Dizajn i realizacija sustava za održavanje termalne radne točke tiskane pločice aktivnim hlađenjem

Piskač, Tibor

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

Sveučilišni studij

DIZAJN I REALIZACIJA SUSTAVA ZA ODRŽAVANJE TERMALNE RADNE TOČKE TISKANE PLOČICE AKTIVNIM HLAĐENJEM

Diplomski rad

Tibor Piskač

Osijek, 2020.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 19.02.2020.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Tibor Piskač
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-913R, 04.10.2019.
OIB studenta:	89146917743
Mentor:	Izv. prof. dr. sc. Marijan Herceg
Sumentor:	
Sumentor iz tvrtke:	Pavao Lubina
Predsjednik Povjerenstva:	Doc.dr.sc. Ratko Grbić
Član Povjerenstva:	Izv. prof. dr. sc. Mario Vranješ
Naslov diplomskog rada:	Dizajn i realizacija sustava za održavanje termalne radne točke tiskane pločice aktivnim hlađenjem
Znanstvena grana rada:	**Procesno računarstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	Prilikom integracije softvera i hardvera, potrebno je verificirati integraciju, tj. uvjeriti se da hardver zaista čini ono što je definirano softverom. Pri tome određeni testovi zahtijevaju mjerenje signala na određenim hardverskim testnim točkama za odgovarajuće operacije koje se odvijaju u softveru. Hardverske testne točke su realizirane u obliku proširenja vodova na određenim mjestima tiskane pločice, u svrhu pristupa sondom osciloskopa tijekom postupka testiranja. Kako bi se ubrzao i automatizirao postupak testiranja, testni objekt postavlja u posebno izrađene adaptere tzv. needle adaptere. To su uređaji posebno izrađeni za testnu pločicu koji na sebi imaju igle, a svaka igla se prilikom zatvaranja spušta na točno određenu testnu točku. Tijekom ovakvog testiranja se testna ploča mora izvaditi iz oklopa koji služi kao
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Vrlo dobar (4)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/bodaPostignuti rezultati u odnosu na složenost zadatka: 2 bod/bodaJasnoća pismenog izražavanja: 1 bod/bodaRazina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	19.02.2020.

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, 07.03.2020.

Ime i prezime studenta:	Tibor Piskač
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-913R, 04.10.2019.
Ephorus podudaranje [%]:	1

Ovom izjavom izjavljujem da je rad pod nazivom**: Dizajn i realizacija sustava za održavanje termalne radne točke tiskane pločice aktivnim hlađenjem**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Marijan Herceg

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

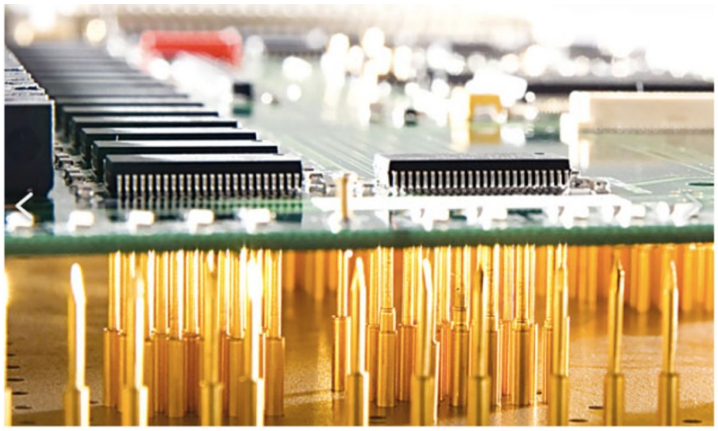
Testiranje ugradbenoga sustava i njegove programske podrške može biti zahtjevan posao, i sklopovski i izvedbeno. Danas govorimo o složenim informacijski moćnim sustavima pa je često potrebno automatizirano testiranje koje je najčešće *black-box* vrste.

Za testiranje tiskanih pločica koristan je *needle testing* odnosno *nail testing* [1] uređaj koji je prikazan na slici 1.1. Takozvani čavli ili igle poravnate su s testnim točkama na pločici i spojene s mjernim uređajima ili električnim izvorima. Prislanjanjem pločice na igle moguće je simulirati ulazne signale i pratiti izlazne signale.

Takav pristup često zahtijeva uklanjanje sustava hlađenja s pločice kako bi igle mogle pravilno dotaknuti testne točke te kako bi se izbjegao kratki spoj. Stoga su pločice na *needle testing* uređajima sklone pregrijavanju te je testiranje potrebno nadzirati. Međutim, u slučaju složenih računalnih sustava na pločici, potrebno testiranje može biti dugo i iscrpno pa je poželjno da se ono može odvijati bez izravnog nadzora.

Cilj je ovoga rada napraviti prototip sustava koji će automatski pratiti temperaturu na pločici te zaustaviti testiranje kad se pločica pregrije. Potom, kad se pločica ohladi, testiranje je potrebno nastaviti. Ukoliko je detektirana neispravnost u sustavu za temperaturni nadzor, testiranje je potrebno zaustaviti. Primjenom takvoga sustava, testiranje se može vršiti bez nadzora.

U sljedećem poglavlju dan je pregled postojeće tehnologije hlađenja. Slijedi ga lista zahtjeva razvijenog sustava za hlađenje te okvirni opis načina na koji su ti zahtjevi zadovoljeni. Zatim slijedi četvrto poglavlje gdje su dodatno opisane najvažnije korištene tehnologije. U petom poglavlju je dan detaljniji uvid u programsku realizaciju sustava, a u šestom poglavlju se daje uvid u testiranje ispravnosti. Slijedi ga eksperimentalna analiza, u kojoj je eksperiment 1 važan za realizaciju konačne verzije sustava, eksperiment 2 je preduvjet za izvršavanje eksperimenta 3, a eksperiment 3 zaključuje određeni parametar.



Slika 1.1. Needle testing uređaj [1]

2. OPĆENITO O TEHNOLOGIJAMA HLAĐENJA

Za sustave koji se zbog električnih razloga griju, potrebno je implementirati sustav hlađenja [2]. Međutim, zbog kvarova ili tehnoloških ograničenja sustava hlađenja sustav se može pregrijati te je temperaturu potrebno konstantno pratiti, davati upozorenja u slučaju detekcije pregrijavanja ili čak ugasiti sustav nakon detekcije pregrijavanja.

Postoje dvije vrste hlađenja: pasivno i aktivno. Pasivno hlađenje koristi materijale visoke temperaturne vodljivosti, visokog toplinskog kapaciteta materijala i velikih površina. Samo po sebi ne uključuje aktivne dijelove kao što su ventilatori. Za procesore se koristi aluminij koji je preko toplinski vodljive paste pričvršćen na procesor. Aktivno hlađenje sadrži aktivan element koji osigurava tok fluida kao što su zrak (primjenom ventilatora) ili tekućina. Aktivno i pasivno hlađenje često je moguće kombinirati.

U slučaju aktivnoga hlađenja pomoću ventilatora cilj je postići cirkulaciju što veće količine „svježeg zraka" preko komponente, kako bi taj zrak apsorbirao što više topline s komponente. Status rada ventilatora moguće je provjeriti tahometrom koji je najčešće ugrađen u ventilatore. Primjerice, za hlađenje procesora osobnog računala, tahometarski signal se očitava pa je vrijednost brzine vrtnje moguće očitati preko pomoćnoga softvera, a moguće je implementirati i posebnu komponentu za upravljanje hlađenjem.

Hlađenje pomoću tekućine koristi tekućinu visokog temperaturnog kapaciteta, a male električne vodljivosti. Uz dovoljnu električnu izolaciju moguće je koristiti i vodu. Sustav koji se hladi može biti uronjen u tekućinu, koja može biti stacionarna (pasivno hlađenje) ili u protoku (aktivno hlađenje).

Najvažniji aspekt sustava zapravo je mjerenje temperature na komponentama koje se jako zagrijavaju. Najčešći oblik implementacije je temperaturni senzor na komponenti ili senzor ugrađen u komponentu. Ponekad je kontaktno mjerenje nepraktično pa se mjeri temperatura izlaznoga zraka, tekućine ili termalno zračenje (dio infracrvenog spektra). Detekcija pregrijavanja važna je zbog sprječavanja termičkog oštećenja tih komponenti.

Za regulaciju potrošnje električne energije i regulaciju temperature procesora najčešće se koristi *overvolting* i *undervolting*. Radi se o dinamičkom skaliranju napona, a *CPU throttling* se koristi za dinamičko skaliranje frekvencije. Spomenute metode koriste se za regulaciju potrošnje električne energije i regulaciju temperature procesora.

Formula za dinamičku snagu zagrijavanja P, u ovisnosti o naponu i frekvenciji rada, glasi:

$$P = C V^2 A f \tag{2-1}$$

C je ukupni kapacitet digitalnog sklopovlja, V je napon digitalnog sklopa u stanju 1, A je faktor aktivnosti (srednji broj događaja promjena stanja na tranzistorima), a f frekvencija procesora.Potrošnjom energije, i proizvodnjom topline kao posljedicom disipacije energije, može se upravljati promjenom veličina V (dinamičko skaliranje napona) i f (dinamičko skaliranje frekvencije). Najčešće će se potrebna frekvencija skalirati po potrebi, a pri tome će se potrebni napon smanjiti.

Međutim, stvarna snaga zapravo je viša, jer regulator sam po sebi troši električnu energiju, te zbog fenomena curenja električne struje. No snaga se može smanjiti i programski, primjerice primjenom HLT instrukcije u x86 arhitekturi koja može zaustaviti rad procesora do sljedećeg prekida. Poznate tehnologije koje koriste opisane metode su Intelov *SpeedStep* [3] i AMD-ov *Cool'n'Quiet* [4].

AMD *Cool'n'Quiet* primarno služi regulaciji temperature, a ta informacija dodatno može koristiti smanjenju brzine ventilatora za hlađenje procesora i tako smanjiti glasnoću koju generira. Temperaturna regulacija u tom se slučaju ostvaruje namještanjem frekvencije i napona, a primarno pripada *dynamic frequency scaling* tipu tehnologije. Prvi put se primijenio za AMD-ovu *Athlon XP* (*Palomino*) liniju procesora. U *Linux* operacijskom sustavu upravljački program za navedenu tehnologiju podržan je *powernow-k8* modulom.

Intelov *Enhanced SpeedStep* funkcionira na sličan način, ali za razliku od *Cool'and'Quiet* nije primarno usredotočen na smanjenje topline, već samo na smanjenje potrošnje električne energije. Poznata mana je to što je u nekim slučajevima moglo doći do problema s reprodukcijom multimedije. Od godine 2012. zaštitni znak *SpeedStep* je istekao.

3. OPĆENITO O OSTVARENOM REGULATORU

U nastavku su korišteni sljedeći pojmovi. **Upravljačkom jedinicom** naziva se cijeli sustav za temperaturnu regulaciju, uključujući temperaturne senzore, ventilatore i komponentu za upravljanje napajanjem testnog objekta (ili signalnu LED diodu umjesto toga). **Upravljačkim uređajem** se naziva mikroupravljački uređaj za hlađenje (Raspberry Pi 3 B) koji je glavni dio upravljačke jedinice. **Upravljački program** sastoji se od **inicijalizacijske sekvence** i **upravljačke sekvence** koje su definirane u izvršnoj datoteci *controller.o* u datotečnom sustavu upravljačkog uređaja.

3.1. Zahtjevi na sustav

Za automatizirano testiranje pločica i njihovih programa jedna od opcija je korištenje *needle testing* uređaja. No tijekom takvog testiranja pločica je izvan vlastitog sustava hlađenja, a neke pločice se jako griju te su potrebne intenzivnije metode hlađenja. Drugim riječima, hlađenje kakvo se može koristiti u *needle testing* okruženju često nije dovoljno.

Zahtjevi koje treba zadovoljavati sustav za rješavanje tog problema su:

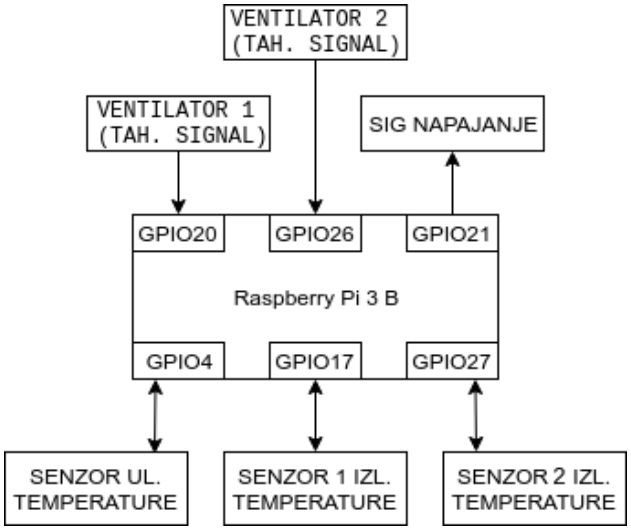
1. Upravljački program će prekinuti napajanje testnog objekta u slučaju detekcije pregrijavanja.
2. Kad je zaštita od pregrijavanja aktivirana, ventilatori za hlađenje će nastaviti raditi.
3. Upravljačka jedinica će stalno biti napajana.
4. Ako se napajanje upravljačke jedinice prekine, napajanje testnog objekta mora biti isključeno.
5. Upravljački program će imati korisničko sučelje preko kojeg će se podesiti i pokrenuti njegova upravljačka sekvenca. Ako pri pokretanju programa signal sa senzora temperature nije primljen, program će biti prekinut.
6. Upravljački program će mjeriti vrijeme neprekidnog rada upravljačke sekvence i neprekidnog napajanja testnog objekta.
7. Upravljački program će izvijestiti o vremenu komunikacije između upravljačkog uređaja i senzora temperature prilikom svakog ciklusa mjerenja tijekom njegove upravljačke sekvence.
8. Nakon što temperatura testnog objekta padne ispod temperature pregrijavanja, napajanje testnog objekta će biti uključeno.

9. Upravljačka jedinica bit će u mogućnosti uključivati i isključivati napajanje testnog objekta čija je struja do 5A.
10. Upravljačka jedinica će imati dva ventilatora s međusobno okomitim protocima zraka.
11. Nasuprot svakom ventilatoru bit će namješten po jedan temperaturni senzor.
12. Upravljačka jedinica će izvješćivati o temperaturama koje su izmjerene pomoću temperaturnih senzora.
13. Ako jedan od ventilatora nije pokrenut, upravljačka jedinica neće uključiti glavno napajanje i javit će pogrešku.

3.2. Opis rješenja u odnosu na zahtjeve

U ovome poglavlju opisano je kako su realizirana rješenja za navedene zahtjeve. Ovaj rad je usredotočen na dizajn upravljačke jedinice tako da su neki elementi samog upravljačkog sustava modificirani u svrhu testiranja. Primjerice, korišten je jedan ventilator umjesto dva, ali je njegov tahometarski signal povezan s GPIO pinovima za oba ventilatora na upravljačkom uređaju, a sustav je bez posebnoga kućišta kako bi se događaji mogli simulirati brže nego u realnoj upotrebi sustava za hlađenje (osim u eksperimentalnoj fazi ovog rada, poglavlje 7). Signal za napajanje testnog objekta je spojen na LED diodu umjesto na jedinicu za uključivanje i isključivanje napajanja stvarnog testnog objekta.

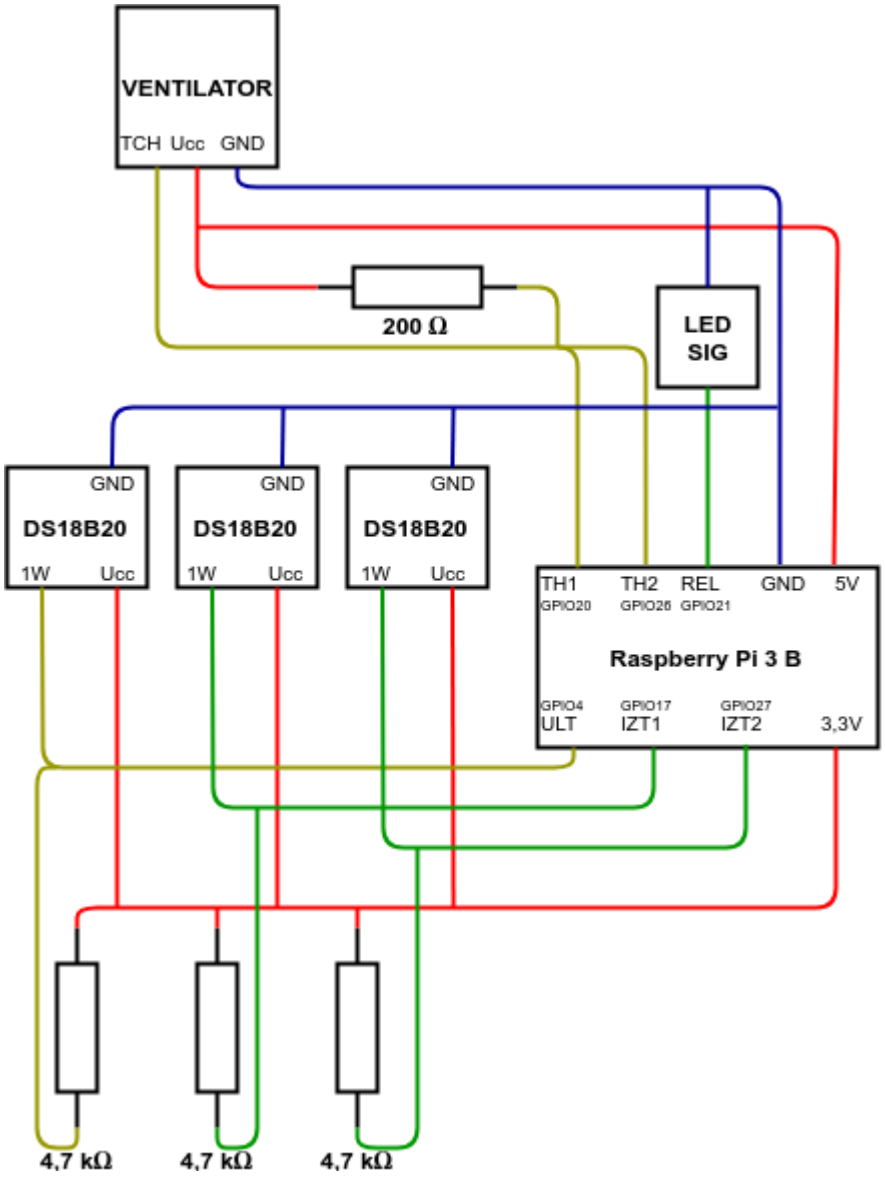
3.2.1. Sklopovska struktura sustava za regulaciju



Slika 3.1: Blok dijagram sustava za temperaturnu regulaciju

Na slici 3.1 prikazan je blok dijagram ugradbenog upravljačkog sustava realiziranog u ovome radu. Korišteni upravljački uređaj je Raspberry Pi 3 B. Kako bi se lakše opisalo rješenje u odnosu na zahtjeve, prvo je dan uvid u povezanost sustava.

Tahometarski vodovi ventilatora spojeni su na GPIO portove 20 i 26 upravljačkog uređaja. Senzori temperature spojeni su na odvojene GPIO pinove (4, 17 i 27). Jedan senzor mjeri temperaturu prostorije, a ostala dva senzora temperature mjere temperaturu izlaznoga zraka (zagrijanog zraka, odnosno zraka iskorištenog za hlađenje). Budući da, zbog *1-Wire* komunikacijskog protokola, upravljački uređaj mora zatražiti očitanje temperature prije nego senzori mogu poslati informaciju o temperaturi, veza između senzora temperature i upravljačkog uređaja je dvosmjerna. Signal za napajanje (uključenje napajanja) testnog objekta je povezan s GPIO portom 21.



Slika 3.2. Ožičenje sustava

Na slici 3.2 prikazano je ožičenje sustava kao detaljniji prikaz u usporedbi s blok dijagramom. Za mjerenje temperature korištena su tri senzora koja implementiraju *1-Wire* komunikacijski protokol *DS18B20* serije: jedan koji nije vodootporan (malih dimenzija) i dva vodootporna (velikih dimenzija). Komunikacijski vod svakog senzora spojen je preko *pull-up* veze za primjenu *1-Wire* protokola, a vod za napajanje svakog senzora spojen je na 3.3V.

Za hlađenje je korišten 3-žični ventilator za PC, *MS PC COOL, LED computer fan* serijskog broja *BLUELEDFANSM12180000340*. Tahometarski signali s ventilatora također su spojeni preko *pull-up* veze jer su realizirani pomoću detektora Hallovog efekta bez pojačala, čiji je čisti izlazni signal samo 0,2V. Napajanje ventilatora je 5V.

Radi lakšeg prototipiranja i testiranja, tahometarski signalni vod jednog ventilatora spojen je na oba pina za tahometarske signale. Signalni pin za uključenje napajanja spojen je na LED diodu.

Upravljačku sekvencu moguće je opisati pomoću tri stanja rada:

1. Stanje napajanja - kad je napajanje testnog objekta uključeno. Program izvješćuje o izmjerenim temperaturama i temperaturnoj granici. Ima najmanji prioritet (može biti prekinuto bilo kojim od drugih stanja).
2. Stanje odmora - kad je napajanje testnog objekta neprekidno isključeno određeno vrijeme. Pobuđuje se detekcijom pregrijavanja. Program izvješćuje o izmjerenim temperaturama, temperaturnoj granici i preostalom vremenu stanja odmora.
3. Stanje pogreške - kad je napajanje testnog objekta isključeno. Pobuđuje se detekcijom neispravnosti ventilatora ili temperaturnih senzora. Ono ne utječe na vrijeme odmora te ne izvješćuje niti o preostalom vremenu odmora, niti o izmjerenim temperaturama, niti temperaturnoj granici, već samo o detektiranim neispravnostima. Ima najveći prioritet (prekida svako od preostalih stanja).

Radi informatičke skalabilnosti, a opet radi očuvanja performansi, upravljački uređaj je opremljen minimalističkom varijantom sustava baziranog na *Linux* jezgri. Uputstva za konfiguraciju i izgradnju koja su većinom korištena dolaze s Loomen stranice FERIT-ova kolegija *Linux u ugradbenim sustavima, DA1-03* [5].

Korištena je primarna konfiguracija *Linux* sustava dostupna u sklopu kolegija, *raspberrypi3_ferit_defconfig*, a kompilator *gcc-arm-linux-gnueabihf*. Za izgradnju slike je korišten alat *Buildroot*, a kao bootloader *U-Boot*.

3.2.2. Napajanje upravljačkog uređaja (zahtjevi 3, 4, 9):

Upravljačka jedinica spojena je na vlastito napajanje, a upravljačka sekvenca koristi GPIO pin kao signal za uključenje napajanja testnog objekta. Umjesto komponente za uključivanje i isključivanje napajanja testnog objekta (npr. releja), korištena je LED dioda kako bi upravljački program bio lakše dizajniran.

Za slučaj otkazivanja upravljačkog programa napravljena je skripta *watchdog.sh*. Prilikom svakog ponavljanja beskrajne petlje tijekom upravljačke sekvence, program zapisuje trenutno vrijeme u datoteku *watchdog_notify.txt*. Ukoliko *watchdog* skripta pročita zapis star 5 sekundi ili više, isključuje signal za napajanje. Ukoliko je datoteka nedostupna, upravljački program isključuje signal za napajanje. Tako je izlazni pin osiguran u slučaju otkazivanja upravljačkog programa.

3.2.3. Integracija ventilatora (zahtjevi 2, 10, 13):

U radu je korišten 3-žični ventilator s tahometrom koji je spojen na vlastito napajanje od 5V te je konstantno uključen. Tahometarski signal spojen je na dva pina upravljačke jedinice. Tahometar je realiziran pomoću senzora Hallovog efekta te daje napon od samo 0.2V pa je spojen pull-up vezom od 5V. Tahometarski GPIO portovi na upravljačkoj jedinici su 26 i 20.

Ukoliko se ventilatori ne okreću, aktivno hlađenje ne funkcionira, a manjak strujanja zraka može dovesti do prevelike ili premale temperature na temperaturnim senzorima, što može dovesti do stalne isključenosti glavnog napajanja ili pregrijavanja.

Napravljen je i *Linux* modul za hvatanje prekida s tahometarskih portova te su definirani prekidni zahtjevi s praznim *handler* procedurama (koje bi inače služile obradi svakog prekidnog zahtjeva). Broj prekida se, u *Linux* sustavu, automatski zapisuje u pripadnu *spurious* datoteku. Čitanjem tih *spurious* datoteka tijekom upravljačke sekvence, program dobiva informaciju okreću li se ventilatori. Detalji su opisani u poglavlju 5.

Pri pokretanju program korisniku daje na izbor zahtijeva li da oba tahometarska signala moraju ispravno raditi ili samo jedan. Ukoliko je broj tahometarskih prekida konstantan (na jednom ili oba porta, što ovisi o izboru korisnika), upravljačka sekvenca prelazi u stanje pogreške i ispisuje obavijest dok se problem ne otkloni. U protivnom sekvenca može biti u stanju napajanja.

3.2.4. Regulacija temperature (zahtjevi 1, 5, 8, 11, 12):

Za regulaciju temperature korištena su tri temperaturna senzora, koji koriste *1-Wire* komunikacijski protokol. Jedan senzor služi za mjerenje temperature okolnog (ulaznog) zraka, dok ostali služe za mjerenje temperature izlaznog zraka.

Na upravljačkoj jedinici korišten je operacijski sustav *Linux* čiji je modul za *1-Wire* komunikaciju osposobljen preko *Device Tree* konfiguracije. Kad bi komunikacijski vodovi spojeni na isti GPIO port, senzor bi morao biti referenciran preko serijskog broja. Kako bi se to izbjeglo, svaki senzor je bio spojen na vlastiti GPIO port te tako definiran u *Device Tree* konfiguraciji. Detalji o tome su opisani u poglavlju 5.2. Informacija o temperaturi na pojedinom senzoru zapisuje se u datoteku generiranu od strane modula za *1-Wire* komunikaciju nakon što je zatraženo čitanje te datoteke od strane upravljačkog programa.

Za temperaturu izlaznog zraka, pri temperaturi okoline od 25°C, temperaturna granica je zapisana u datoteku *boards.txt* od strane korisnika. Budući da pločica može imati više procesora, a maksimalno opterećenje jednog procesora može rezultirati nižom temperaturom izlaznog zraka nego polovično opterećenje više procesora, temperaturne granice za okolinu od 25°C potrebno je empirijski odrediti za pojedinu pločicu.

Određivanje temperaturne granice za okolinu od 25°C moguće je izvesti tako da se okolina podesi na zadanu temperaturu, a pločica se podesi na pripadni *needle testing* uređaj. Nakon toga bi se svaki procesor na pločici trebao odvojeno dovesti do točke pregrijavanja, pri čemu bi se bilježile temperature izlaznoga zraka. Najmanja izmjerena temperatura zapisuje se u datoteku *boards.txt*. Format teksta navedene datoteke treba biti *oznaka(tab)temperatura(novi_red)* s temperaturom u °C, ali napisanoj bez mjerne jedinice. Demonstracija je dana u **eksperimentu 2** (poglavlje 7.3.).

U realnim uvjetima okolna temperatura ne mora biti 25°C pa je u tom slučaju potrebno operativnu graničnu temperaturu promijeniti u ovisnosti o okolnoj temperaturi. Primijenjena matematička funkcija, koja prima graničnu temperaturu pri temperaturi okoline od 25°C i trenutnu temperaturu okoline kao parametre, dobivena je **eksperimentom 1** (poglavlje 7.2).

Prilikom pokretanja sekvence upravljanja, ako je temperatura izlaznoga zraka na oba senzora izlaznoga zraka manja od granične temperature, upravljačka sekvenca može biti u stanju napajanja. Ako je temperatura izlaznoga zraka jednaka ili veća od granične temperature u bilo kojem trenutku i na bilo kojem senzoru temperature izlaznoga zraka, sekvenca prelazi u stanje odmora. Ukoliko je krajem stanja odmora temperatura izlaznoga zraka na oba senzora temperature izlaznoga zraka manja od granične temperature, uređaj se može vratiti u stanje napajanja, a u protivnom se ponavlja stanje odmora. Ukoliko s jednim od senzora nije moguće komunicirati, sustav prelazi u stanje pogreške te program o pogrešci izvješćuje. Uklanjanjem pogreške sekvenca iz tog stanja izlazi (osim ako je stanje pogreške potaknuto neispravnosti okretanja ventilatora).

Ukoliko je potrebno testirati samo ispravnost upravljačkog sustava, signalni pin za napajanje može biti spojen i na, primjerice, LED diodu. Isto to je napravljeno u prototipu upravljačkog sustava povezanog s ovim radom.

3.2.5. Analiza trenutnog rada sustava (zahtjevi 6, 7)

Nakon uspješnog pokretanja upravljačke sekvence, upravljački program bilježi vrijeme početka rada. Korištenjem te informacije, tijekom upravljačke sekvence se vrijeme njenoga rada ispisuje u konzolu. Kao što je prethodno spomenuto, temperatura se očitava čitanjem prikladnih datoteka, po jedne za svaki senzor, ali je potrebno određeno vrijeme za to očitavanje. Zato se programski bilježi vrijeme prije i nakon čitanja pojedine datoteke, a razlika tih vremena se ispisuje u konzolu, za pojedini senzor. Dodatno se ispisuje vrijeme koje je sekvenca provela u stanju napajanja, ali je vrijednost maksimalne pogreške jedna sekunda u slučaju prelaska u stanje odmora ili stanje pogreške.

4. KORIŠTENE TEHNOLOGIJE

U ovom poglavlju opisane su sklopovske i programske komponente koje su korištene i koje je važno istaknuti.

4.1. *Linux Device Tree*

Kako bi programer lakše i bolje iskoristio uređaje ugrađene ili prikopčane na matičnu ploču, operacijski sustav koristi specijalne programe koji nude sučelja za komunikaciju s određenim vrstama uređaja. Komunikacija može biti usmjerena samo prema računalu, samo prema uređaju ili u oba smjera. Takvi programi se nazivaju upravljačkim programima uređaja (engl. *device driver*), a u okruženju *Linux* ih se još naziva i modulima.

U ranoj fazi evolucije sustava baziranih na *Linuxu* svi moduli i opisi njihove povezanosti sa sklopovljem bili su definirani u *Linux* jezgri. No kako broj platformi s različitim sklopovskim strukturama raste i prilagodbe *Linux* sustava zahtijevaju veću fleksibilnost, tako je postalo poželjno te opise povezanosti kompilirati u posebnu datoteku i na taj način preskočiti kompilaciju velikog dijela jezgre. U primjenu je ušla struktura poznata pod nazivom *Device Tree*. [6]

Device Tree opisuje organizaciju potrebnih uređaja i postavlja parametre za njihove module. Odnos između uređaja i matične ploče opisan je stablastom strukturom, zbog čega dolazi i ime *Device Tree*. Neispravna organizacija te neispravna konfiguracija parametara dovodi do neispravnog i neiskoristivog ponašanja modula.

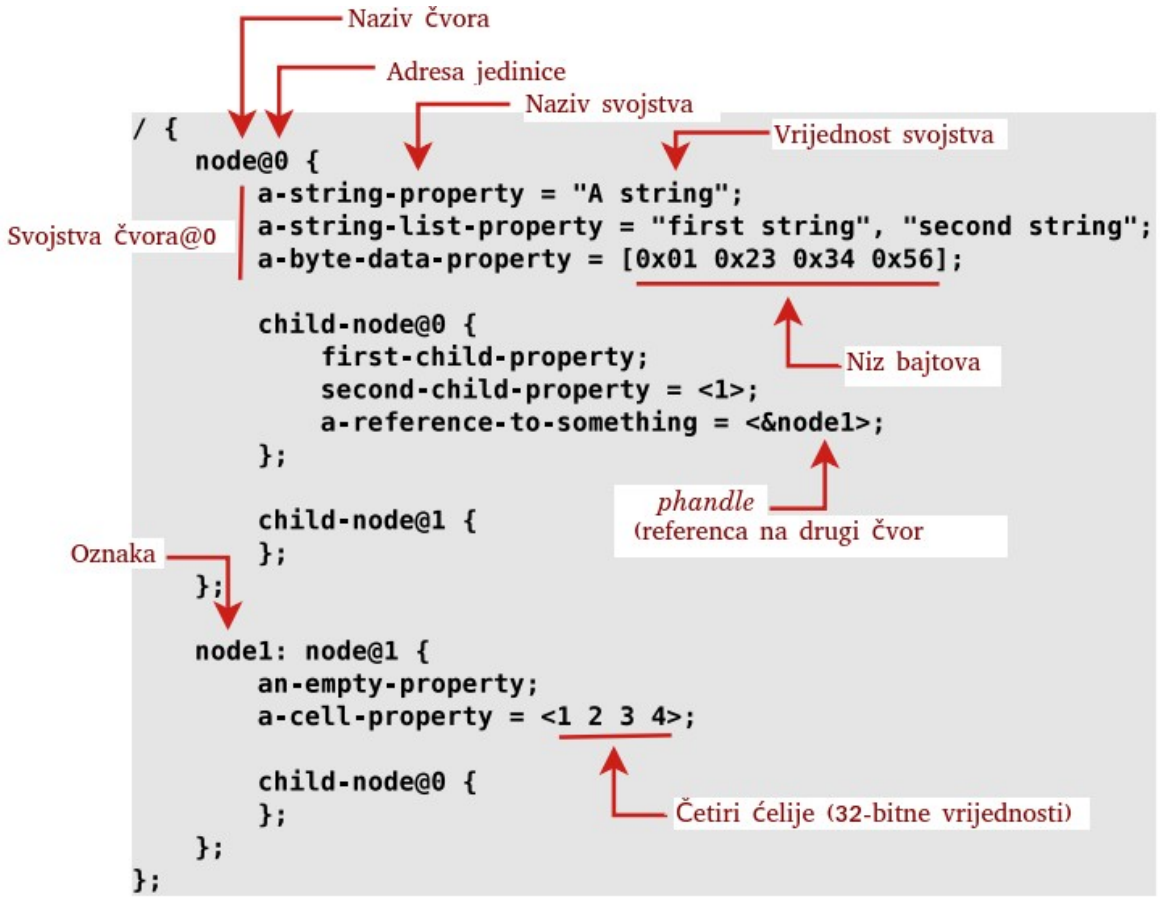
Glavna prednost *Device Tree* koncepta je mogućnost opisa sklopovlja u datoteci za skup platformi te specifikacija podskupa platformi i njegove organizacije sklopovlja u novoj datoteci, i tako dalje sve do definicije sklopovlja specifične platforme.

Izvorni kod *Device Tree* strukture zapisuje se u *dts* i *dtsi* datoteke. Sintaksa je slična programskom jeziku C te je moguće naslijediti nadređene grupe platformi, a konačni izlaz je *dtb* datoteka s potpunim specifikacijama za platformu, uključujući i naslijeđene specifikacije.

Kôd je organiziran u strukturu stabla, gdje čvorovi mogu imati svoje oznake (engl. *label*), svojstva (engl. *properties*) te djecu (engl. *child node*), kao što je prikazano na slici 4.1. Čisti prikaz čvora, bez oznake, sastoji se od imena vrste sklopa i njegove adrese. Čvorovi se mogu referencirati jedni na druge u svojim parametrima.

Opis sklopovlja dodatno se može podešavati i proširivati *dts* datotekama koje služe za preklapanje i najčešće imaju *-overlay* dodatak prije ekstenzije te su kompilirane u *dtbo* datoteke bez *-overlay* dodatka.

Module je moguće aktivirati i ručno, no u nekim slučajevima je to puno teže. Neki moduli, kao što su *1-Wire* moduli, zahtijevaju opis sklopovlja unutar *Device Tree* strukture ili preko *overlaya* kako bi ispravno radili te se rijetko aktiviraju ručno.



Slika 4.1. Primjer Device Tree kôda [6]

4.2. *1-Wire* protokol

Kada govorimo o senzorima za mikroupravljače, najčešće se to odnosi na senzore s digitalnim izlazom te analogne senzore koji mogu poslati signal različitih naponskih vrijednosti ili različitih frekvencija. No postoji potreba i za digitalnim senzorima koji će poslati dodatne informacije, primjerice identifikacijski broj, broj modela te općenito informacije koje će omogućiti operacijskom sustavu da ih brzo prepozna preko postojećeg upravljačkog programa uređaja.

U ovom radu govori se o *Dallas Semiconductor Corp. 1-Wire* protokolu [7], baziranom na integriranom krugu koji omogućava digitalnu komunikaciju više uređaja preko samo jednoga signalnog voda. Takav protokol pogotovo je važan za sustave kao što je *Raspberry Pi 3 B*, koji nema analogne ulaze.

Komunikacijski vod spojen je *pull-up* [8] vezom na GPIO pin ugradbenoga sustava. Početna stanja oba kraja komunikacijske veze trebala bi biti visoke impedancije te početno stanje na komunikacijskom vodu u stanju 1. Uređaji koji sudjeluju u komunikaciji nazivaju se *master* i *slave*. *Master* uređaj najčešće je mikroupravljač ili PC te ima mogućnost zatražiti početak komunikacije, a *slave* uređaj je senzor ili aktuator. Ukoliko je više *slave* uređaja spojeno na isti signalni vod, potrebno ih je referencirati preko njihovih serijskih brojeva.

Kod *1-Wire* komunikacije bitovi se prenose u vremenskim periodima. U normalnom režimu rada period traje 60 mikrosekundi, a u *overdrive* režimu traje 8 mikrosekundi. Ukoliko napon na komunikacijskom vodu padne početkom perioda, poslana je logička nula, a padne li krajem perioda, poslana je logička jedinica.

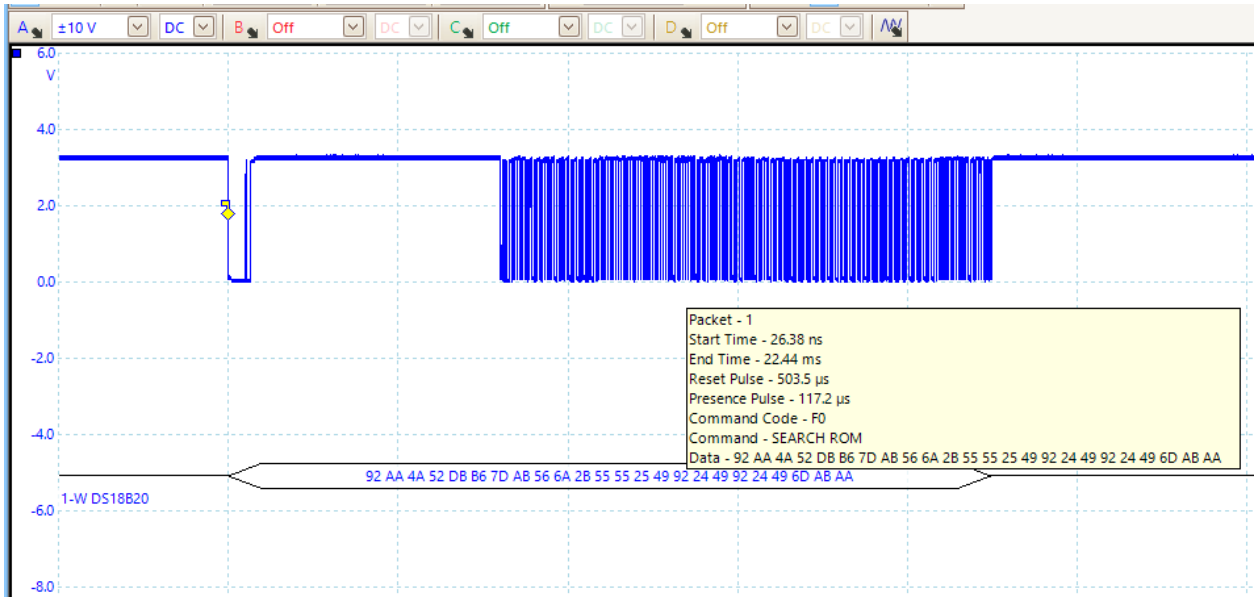
Posebni signali su *master reset pulse* i *slave presence pulse*. U normalnom režimu rada, ukoliko *master* uređaj drži napon veze niskim dulje od 480 mikrosekundi, *slave* uređaj to interpretira kao *master reset pulse* i zahtjev za početkom komunikacije. *Master* napon vraća u visoko stanje. Ubrzo nakon toga, *slave* uređaj drži napon veze niskim sljedećih 60 do 240 mikrosekundi, što *master* interpretira kao *slave presence pulse* što mu daje informaciju da je *slave* uređaj dostupan za komunikaciju. Na slici 4.2. je grafički prikaz opisanog pozdrava s Pico Technology Ltd. mrežne stranice [7].

Zatim *slave* uređaji mogu *master* uređaju poslati svoje 64-bitne *ROM ID* brojeve koji se sastoje od 8-bitne oznake porodice, 48-bitnog serijskog broja i 8-bitnog CRC broja. Nakon toga *master* i *slave*-ovi mogu komunicirati preko takozvanih ROM komandnih sekvenci. Primjer komunikacije može se vidjeti na slici 4.3.

Većina *slave* uređaja zahtijeva samo signalnu žicu i uzemljenje za napajanje i komunikaciju. Oni koji mogu ili ne moraju koristiti dodatno napajanje imaju dodatne mogućnosti kada ga koriste.

Primjerice, temperaturni senzori s vanjskim napajanjem temperaturu mogu mjeriti bez zahtjeva od strane *mastera* te za to vrijeme mogu javiti *masteru* da su zauzeti.

Slika 4.2. Pozdrav u 1-Wire protokolu [7]



Slika 4.3. Općeniti primjer 1-Wire komunikacije [7]

4.3. Tahometar

Kod hlađenja je često potrebna informacija o funkcionalnosti i trenutnom stanju uređaja za hlađenje. Neplanirano usporenje ili kvar ventilatora može dovesti do ubrzanoga grijanja dijela sustava koje, ukoliko upravljački sustav ne registrira previsoku temperaturu, može dovesti do oštećenja. Pomoću informacije o uređajima za hlađenje moguće je poslati obavijest operaterima, a informacija može biti poslana i hlađenom sustavu koji se po potrebi može zaustaviti ili promijeniti režim rada. U slučaju ventilatora, informaciju je moguće dobiti pomoću tahometra [9].

Tahometri mogu biti:

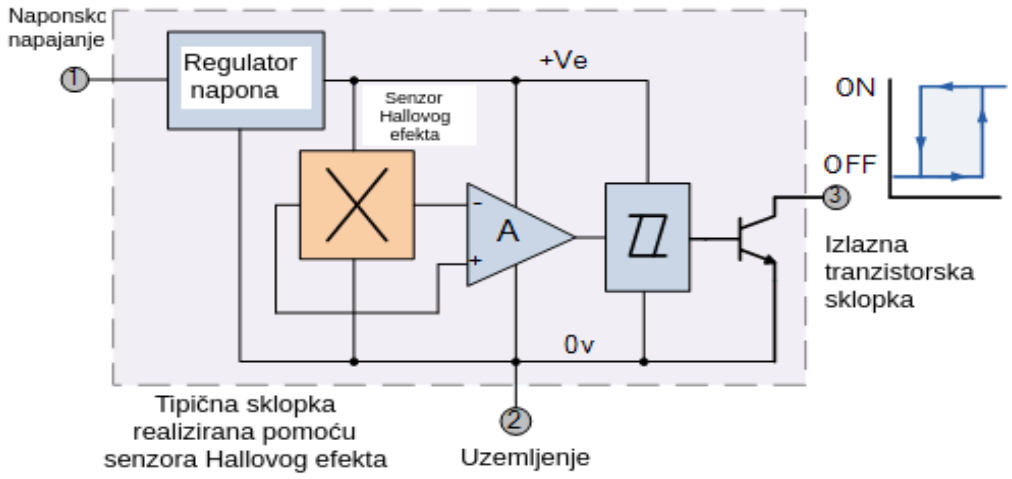
- **Analogni -** za veću brzinu daje veći napon na izlazu, pomoću konverzije iz frekvencije u napon.
- **Digitalni -** koristi memoriju te daje srednju vrijednost temperature, a po mogućnosti i statističke podatke o okretanju.
- **Kontaktni -** povezan s osovinom te koristi optički enkoder ili magnetski senzor.
- **Beskontaktni -** mjeri vrtnju bez dodira s osovinom pomoću lasera ili optičkog diska.

U ovom radu korišten je tahometar baziran na senzoru Hallovog efekta [10]. Radi se o analognom tahometru, osim što u ovom slučaju nema konverzije frekvencije u napon, već se brzina očitava samo iz frekvencije.

Tahometar na bazi senzora Hallovog efekta sadrži tanku metalnu traku koja je spojena u strujni krug. U prisutnosti magnetskog polja dolazi do poprečnog električnog napona (razlika potencijala, odnosno odsustvo elektrona na jednoj strani, a višak na drugoj). Dakle, ukoliko rotor sadrži trajne magnete ili elektromagnete, a traka senzora Hallovog efekta je u njegovoj blizini, njegova vrtnja će dovoditi do promjene poprečnog napona izazvanog na traci pa se tako može detektirati brzina vrtnje. Ukoliko rotor ne sadrži trajne magnete, potrebno je dodati magnet na osovinu motora.

Druga opcija je u tahometar ugraditi trajni magnet ili elektromagnet. U tom slučaju blizina nekog metala utječe na intenzitet Hallovog efekta jer osovina najčešće ima ugrađene metalne zupce.

U svakom slučaju, napon dobiven Hallovim efektom često je potrebno pojačati (odvojenim pojačalom). Budući da se za prekide na računalima često koriste uzlazni i silazni brid napona, postoji i potreba za okidnim sklopom. Pojednostavljeni prikaz takvog senzora dan je na slici 4.4.



Slika 4.4. Pojednostavljena shema senzora Hallovog efekta [10]

5. REALIZACIJA TEMPERATURNOG REGULATORA

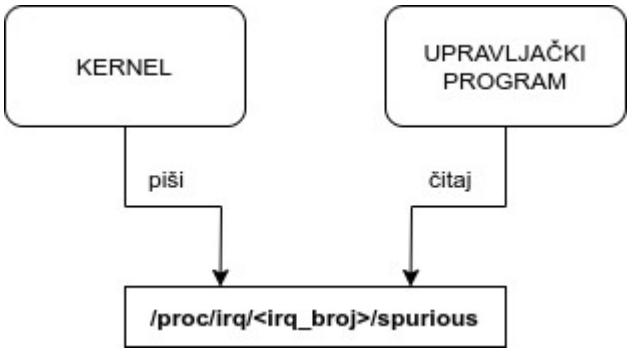
5.1. Modul za prekide tahometra

Upravljački program mora imati informaciju o vrtnji ventilatora za hlađenje kako bi isključio napajanje testnog objekta u slučaju da se ventilatori ne okreću ili nisu dostupni. Zato su portovi za tahometarske signale ventilatora spojeni na prekidne rutine koje je moguće uključiti aktivacijom posebno napisanog *Linux* modula *3w_counter.ko*.

Izvorni kôd je u datoteci *3w_counter.c*. Prilikom pokretanja modula inicijaliziraju se prekidni zahtjevi (engl. *interrupt requests*) za svaki od dva tahometarska pina i pridružuju im se rukovatelji (engl. *handlers*), a prilikom njegova uklanjanja oni se deinicijaliziraju.

Rukovatelji, u ovom slučaju, zapravo ništa posebno ne rade, već samo služe tome kako bi operacijskom sustavu omogućili brojanje prekida kad se oni dogode. Informacija o broju prekida može se pronaći u datoteci */proc/irq/<irq_broj>/spurious* te je potrebno pročitati prvi redak tijekom izvođenja upravljačkoga programa. Primjena datoteke vizualno je prikazana slikom 5.1.

Modul je moguće aktivirati pomoću *insmod* naredbe i nije ga potrebno posebno podešavati niti podesiti u *Device Tree*-u, osim ako se to baš želi. Prekidi su povezani s GPIO portovima 26 i 20.



Slika 5.1. Primjena „spurious" datoteke

5.2. *Linux* konfiguracija

Kao što je već spomenuto, kako bi bili iskorišteni bez posebnih komplikacija, moduli za *1-Wire* komunikaciju podešeni su preko *Device Tree* konfiguracije. Za realizaciju *Device Tree* konfiguracije modificirane su dvije datoteke.

U *bcm283x.dtsi* datoteci definirane su reference na pinove. Parametar *brcm,pins* definira broj GPIO pina, *brcm,function* parametar funkciju pina (0 za ulaz, 1 za izlaz) koji je zadan na 0, a

brcm,pull primjenu *pull-up* ili *pull-down* veze za pin (0 za ništa, 1 za *pulld-down*, 2 za *pull-up*). *Pull-up* veza je spojena eksterno pa je konfiguracija tog parametra 0.

Datoteka *bcm2710-rpi-3-b.dts* je konačna konfiguracija za Raspberry Pi 3 B platformu u koju dodajemo informaciju o modulu koji se koristi za senzore. Parametar *compatible* opisuje koji se upravljački program točno koristi. U ovom slučaju to je *w1-gpio*. Parametar *pinctrl-<broj>* definira moguće stanje uređaja i pridružuje mu konfiguraciju pina (referenciranjem oznake definirane u *bcm283x.dtsi*), a *pinctrl-names* definira ime stanja. U ovom slučaju svaki temperaturni senzor može imati samo jedno stanje, a konfiguracija pina posebno je definirana za svaki uređaj. Parametar *gpios* opisuje na kojem portu i kojim signalom se aktivira instanca uređaja u sklopu *Linux* modula. Parametar *rpi,parasitic-power* sličan je *brcm-pull* pa je iz istog razloga u stanju 0. Prevođenje u *DTB* datoteku za željenu platformu omogućeno je parametrom *status* koji može biti samo u stanjima *ok* ili *okay*.

Potom je vrijednost temperature moguće očitati preko pripadne datoteke za pojedini GPIO port na koji je spojen pojedini senzor. Modifikacija *DTS* datoteka dana je na slici 5.2. Kako su senzori fizički i konfiguracijski odvojeni jedni od drugih, tako im je moguće pristupati preko mape za pojedinu *1-Wire* instancu.

Korist odvajanja senzora dana je na slici 5.3, gdje je moguće dobiti i osnovna ideja o preslikavanju senzora na datotečni sustav te pristupanju njima. Naime, za svaki GPIO port je posebno definirana mapa *w1_bus_master<1,2, ili 3>* pa će *w1_bus_master1* uvijek biti povezana s *GPIO4* i služiti informaciji o okolnoj temperaturi, a *w1_bus_master2* i *w1_bus_master3* informaciji o temperaturi izlaznog zraka na portovima *GPIO17* i *GPIO27*.

5.3. Datoteke za pokretanje i izvođenje programa

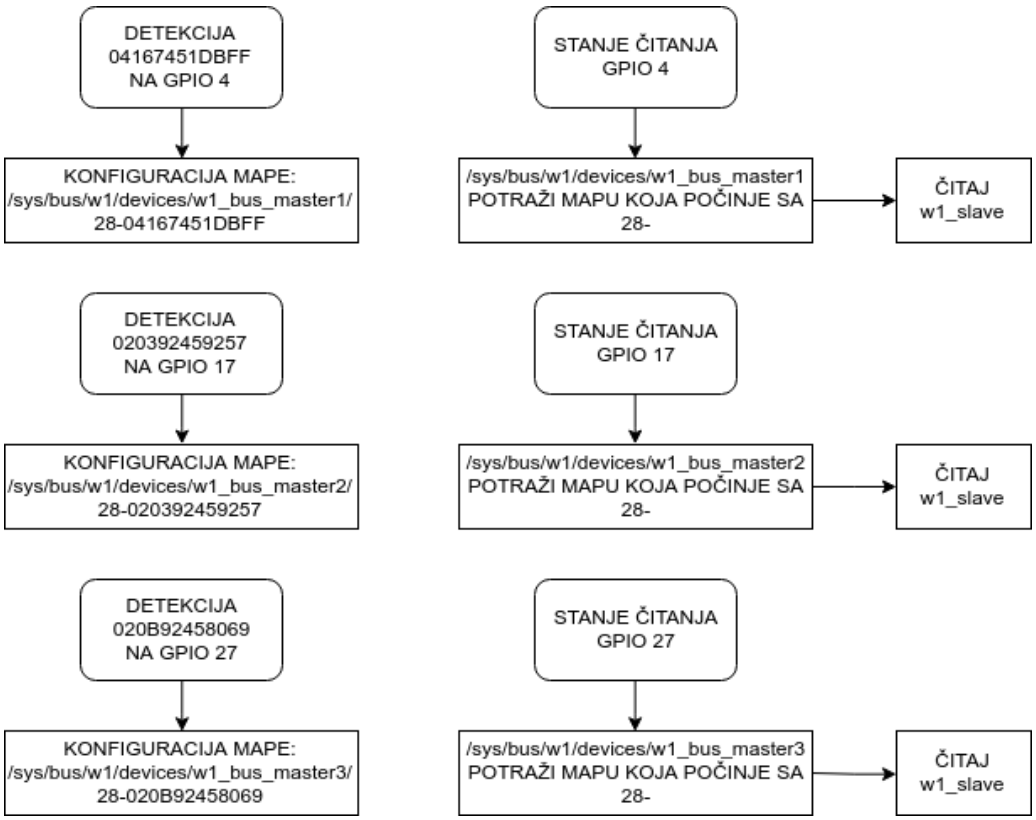
Upravljački program nalazi se u mapi */controller* i pokreće se preko datoteke *controller.o*. No potrebno je pripremiti tahometarske prekidne rutine i *watchdog* skriptu za sigurnosno izvršavanje programa.

Skripta */start_controller_safe.sh* preporučena je za pokretanje programa. Ona prvo pokuša ukloniti pa ponovno aktivirati modul *3w_counter* (*rmmod*, *insmod*) za omogućavanje prekida na pinovima za tahometarske signale. Pronalazi *spurious* datoteke za pojedini tahometarski prekidni zahtjev, napravi virtualnu mapu na lokaciji */mnt* te datoteku */mnt/watchdog_notifier.txt*. Nakon toga pokreće */controller/watchdog.sh* skriptu u pozadini, osim ako je već pokrenuta. Na kraju pokreće upravljački program i šalje mu putanje IRQ datoteka kao argumente. Ukoliko je poslan broj

```
/ {
soc {

…
        gpio: gpio@7e200000 {
                        w1_pins_room: w1_pins@0 {
                                brcm,pins = <4>;
                                brcm,function = <0>;
                                brcm,pull = <0>;
                        };
                        w1_pins_board_1: w1_pins@1 {
                                brcm,pins = <17>;
                                brcm,function = <0>;
                                brcm,pull = <0>;
                        };
                        w1_pins_board_2: w1_pins@2 {
                                brcm,pins = <27>;
                                brcm,function = <0>;
                                brcm,pull = <0>;
                        };
        };

…
};
};
```

```
{
        w1_room: w1_room {
                        compatible = "w1-gpio";
                        pinctrl-names = "default";
                        pinctrl-0 = <&w1_pins_room>;
                        gpios = <&gpio 4 0>;
                        rpi,parasitic-power = <0>;
                        status = "okay";
        };

        w1_board_1: w1_board_1 {
                        compatible = "w1-gpio";
                        pinctrl-names = "default";
                        pinctrl-0 = <&w1_pins_board_1>;
                        gpios = <&gpio 17 0>;
                        rpi,parasitic-power = <0>;
                        status = "okay";
        };

        w1_board_2: w1_board_2 {
                        compatible = "w1-gpio";
                        pinctrl-names = "default";
                        pinctrl-0 = <&w1_pins_board_2>;
                        gpios = <&gpio 27 0>;
                        rpi,parasitic-power = <0>;
                        status = "okay";
        };
;};
```

Slika 5.2. Modifikacija „bcm283x.dtsi” (lijevo) i „bcm2710-rpi-3-b.dts” (desno)



Slika 5.3. Korist odvojenih GPIO portova kod 1-Wire komunikacije

kao argument, upravljački program će dodatno dobiti taj argument koji mu predstavlja vrijeme odmora.

Skripta */controller/watchdog.sh* nadzire izvršava li se program, te ukoliko se ne izvršava, podešava GPIO port signala za napajanje na 0. Prvo inicijalizira GPIO port 21 pa čita sadržaj datoteke */mnt/watchdog_notifier.txt* u koju upravljački program zapisuje vrijeme (u sekundama) sustava nakon svakog ponavljanja upravljačke sekvence. Uspoređuje sadržaj datoteke s trenutnim vremenom koje je sama dohvatila. Ukoliko je trenutno vrijeme barem 5 sekundi više, postavlja GPIO port 21 na 0 preko datotečnog sustava. Koraci od čitanja */mnt/watchdog_notifier.txt* datoteke se beskrajno ponavljaju.

5.4. Organizacija izvornog koda upravljačkog programa

Radi čitljivosti i skalabilnosti, kôd upravljačke aplikacije organiziran je u više datoteka.

- **controller (controller.c)**

Datoteka s *main* funkcijom u kojoj su realizirane komunikacija s korisnikom, priprema za izvođenje upravljačke sekvence te sama upravljačka sekvenca.

- **temperature (temperature.c, temperature.h)**

Za komunikaciju s temperaturnim senzorima. Sastoji se od funkcije za lociranje datoteke preko koje se očitava temperatura senzora te funkcije za čitanje temperature, koja umeće i vrijeme čitanja pomoću *unsigned long long* pokazivača.

- **tested_dev (tested_dev.c, tested_dev.h)**

Služi za komunikaciju s testnim objektom. Može podesiti vrijednost GPIO portu 21 preko datotečnog sustava. Ima i inicijalizacijsku funkciju za slučaj da *watchdog* još nije inicijalizirao port.

- **local (local.c, local.h)**

Za komunikaciju s lokalnim datotekama koje nisu izravno povezane s posebnim *Linux* modulima. Uključuje čitanje datoteke *boards.txt* s popisom testnih objekata i njihovih početnih temperaturnih granica. Funkcija *watchdogNotify* zapisuje trenutno vrijeme u datoteku */mnt/watchdog_notifier.txt* te tako sprječava *watchdog.sh* skriptu da ugasi signal napajanja.

- **tachometer (tachometer.c, tachometer.h)**

Sadrži funkciju za čitanje broja tahometarskih prekida preko */proc/irq/<irq_broj>/spurious*.

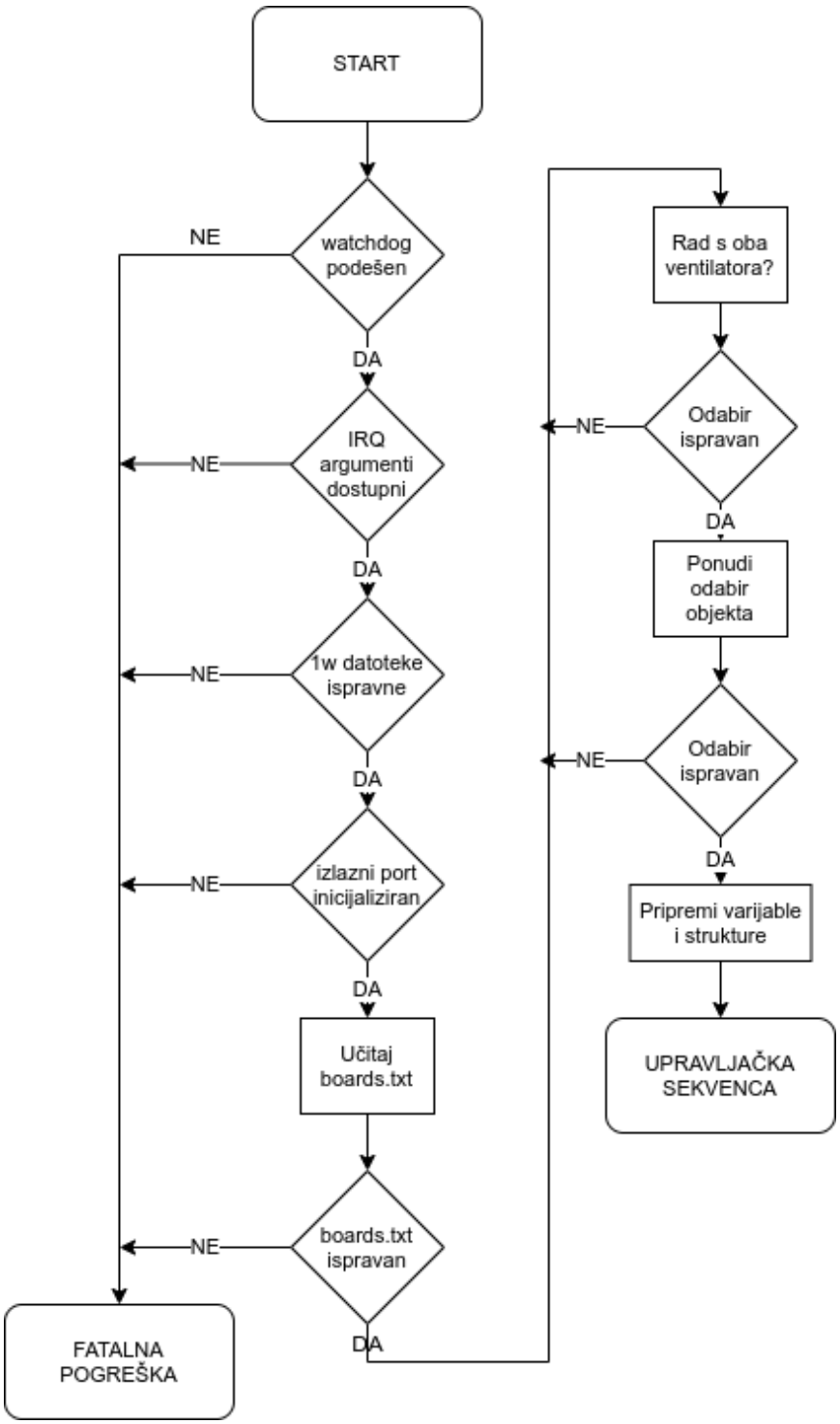
- **helpers (hepers.c, helpers.h)**

Sadrži pomoćne funkcije: funkciju za pretvaranje niza znakova u broj, funkciju za oduzimanje vremena između dva objekta *timespec* strukture te *stopwatch* strukture i funkcije (štoperica).

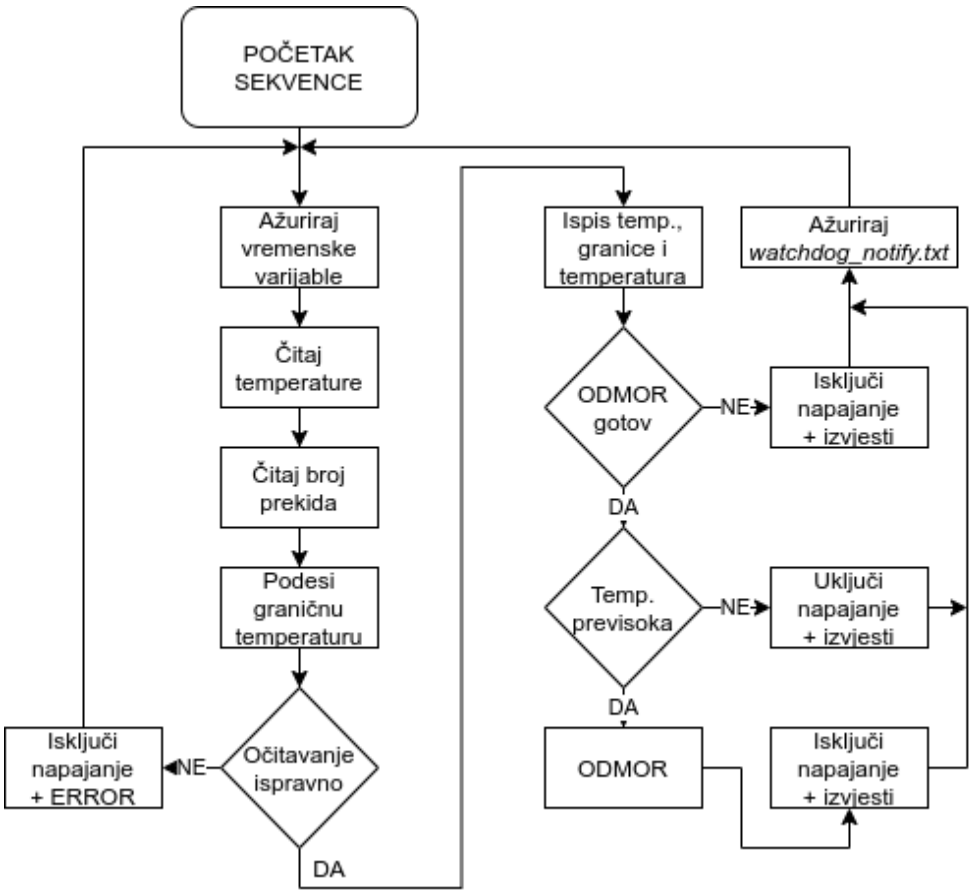
5.5. Struktura upravljačkog programa

Upravljački program moguće je odvojiti u dvije sekvence:

- **Inicijalizacijska sekvenca** (slika 5.4)
- **Upravljačka sekvenca** (slika 5.5)



Slika 5.4. Pojednostavljena shema inicijalizacijske sekvence



Slika 5.5. Pojednostavljena upravljačka sekvenca

Inicijalizacijska sekvenca prikuplja argumente komandne linije, provjerava ispravnost potrebnih datoteka, ispisuje listu mogućih testnih objekata zajedno s njihovim početnim graničnim temperaturama te nudi izbor objekta. Zatim priprema varijable za izvršavanje upravljačke sekvence.

Upravljačka sekvenca čita temperature sa senzora, broj prekida na tahometarskim portovima te detektira pregrijavanje i neispravnost senzora ili nedostatak okretanja ventilatora. Ispisuje i vrijeme vlastitog neprekinutog izvršavanja, vrijeme napajanja testnog objekta, vrijeme očitavanja sa senzora te vrijeme odmora između čitanja. Radnje se ponavljaju u beskrajnoj petlji.

6. PROVJERA ISPRAVNOSTI UPRAVLJAČKOG SUSTAVA

U sklopu provjere ispravnosti sustava napravljena su testiranja na zahtjeve na sustav navedene u poglavlju 3.1. vezane uz temperaturnu regulaciju i ispravnost komponenti, a to su zahtjevi 1, 5, 8, 12 i 13.

Testiranje je izvršeno pomoću simulacije potrebnih događaja. Na primjer, neispravnost komponente je simulirana njenim ukopčavanjem i iskopčavanjem, a grijanje izlaznog zraka je simulirano dodirivanjem temperaturnih senzora prstima (za senzor koji nije vodootporan) ili zatvorenim šakama (za vodootporne senzore). Bitan je isključivo odziv sustava na određene fizički simulirane događaje.

Upravljački sustav bio je spojen i uključen. Korištena je verzija programa osposobljena eksperimentom 1. Okolna temperatura bila je konstantna, između 19°C i 20°C, a u *boards.txt* datoteku je bila dodana konfiguracija *soba*.

U konzoli, log datotekama i slikama sljedeća izvješća znače:

total uptime – trajanje upravljačke sekvence.

test uptime – ukupno vrijeme koje je sustav proveo u stanju napajanja. Moguća je pogreška od 1 sekunde prilikom izlaska sekvence iz stanja napajanja.

read: (broj) – ukupno trajanje komunikacije s temperaturnim senzorima kod svakog ciklusa provedenog u stanju napajanja ili odmora.

(broj) ms read – trajanje komunikacije s pojedinim temperaturnim senzorom.

room – temperatura očitana na GPIO portu namijenjenom za senzor sobne temperature.

brd 1 – temperatura očitana na GPIO portu namijenjenom za prvi senzor temperature izlaznog zraka.

brd 2 – temperatura očitana na GPIO portu namijenjenom za drugi senzor temperature izlaznog zraka.

limit – trenutna granična temperatura za izlazni zrak, za koju se aktivira stanje odmora.

PAUSE - Preostalo vrijeme odmora.

...moving on... – indikator stanja napajanja.

...starting pause... – indikator početka stanja odmora.

...taking a break... – indikator stanja odmora.

HOLD-UP – indikator stanja pogreške.

HOLD-UP: Fan rotation malfunction! – broj prekida na tahometarskom GPIO portu je konstantan. Pojavljivanje je detaljno opisano u pripadnom poglavlju.

HOLD-UP: Lost contact with room temperature detector. Please fix it! – neispravnost na senzoru sobne temperature. Pojavljuje se i kad je očitana temperatura u intervalu <*-0,1°C, 0,1°C*>.

HOLD-UP: Lost contact with board temperature detector (broj). Please fix it!
– neispravnost na [redni broj] senzoru temperature izlaznog zraka. Pojavljuje se i kad je očitana temperatura u intervalu <*-0,1°C, 0,1°C*>.

ERROR: File (broj) for board temperature not found. – neispravnost na [redni broj] senzoru temperature izlaznog zraka prije pokretanja upravljačke sekvence. Pojavljuje se i kad je očitana temperatura u intervalu <*-0,1°C, 0,1°C*>. Ne logira se i tretira se kao fatalna pogreška.

ERROR: File for room temperature not found. – neispravnost na senzoru sobne temperature prije pokretanja upravljačke sekvence. Pojavljuje se i kad je očitana temperatura u intervalu <*-0,1°C, 0,1°C*>. Ne logira se i tretira se kao fatalna pogreška.

6.1. Test dinamičke promjene granične temperature za izlazni zrak

Ovaj test neizravno je povezan s testiranjem zahtjeva 1 i 8. Kako u realnoj situaciji temperatura okolnog zraka nije nužno blizu 25°C, program dinamički mijenja graničnu temperaturu za izlazni zrak u ovisnosti o okolnoj temperaturi, po matematičkoj formuli dobivenoj eksperimentom 1. Kao senzor okolne temperature korišten je nevodootporni *DS18B20,* koji ima kraće vrijeme odziva od vodootpornih senzora.

Koraci testa su:

1. Pokrenuti upravljačku sekvencu u stanju napajanja ili odmora.
2. Bilježiti *room* i *limit* vrijednost više od jedan ciklus.
3. Zagrijavati senzor okolne temperature i bilježiti *room* i *limit* vrijednosti.
   Očekivanje: *limit* vrijednost raste povećanjem *room* vrijednosti.
4. Zaustaviti grijanje senzora okolne temperature, bilježiti *room* i *limit* vrijednosti.
   Očekivanje: *limit* vrijednost pada smanjenjem *room* vrijednosti.

Izvješća je moguće iščitati iz priloga P.6.1. Slijedi opis izvršene procedure testiranja po *total uptime* vrijednosti.

0,0s – 2,7s : Završetak koraka 1 (prvog mjerenja nakon pokretanja upravljačke sekvence).

2,7s – 11,5s: Korak 2.

14,5s – 32,3s Korak 3.

Rezultat: *limit* i *room* vrijednosti rastu, u skladu s očekivanjem u koraku 3.

35,3s – 68,3s Korak 4.

Rezultat: *limit* i *room* vrijednosti padaju, u skladu s očekivanjem u koraku 3.

Zaključak: Sva očekivanja su ispunjena. Sekvenca nije ni u jednom trenutku bila u stanju pogreške.

6.2. Test pregrijavanja

Ovaj test napravljen je za zahtjeve 1, 8 i 12. Ako su *brd 1* ili *brd 2* vrijednost iznad *limit* vrijednosti, glavno napajanje se isključuje, započinje stanje odmora i signal za napajanje je isključen. Ako je *brd 1* ili *brd 2* vrijednost i dalje iznad *limit* vrijednosti, stanje odmora se ponavlja. U protivnom se sustav vraća u stanje napajanja. Vrijeme odmora je 10 sekundi, s mogućom pogreškom od 1 sekunde početkom stanja odmora.

Koraci testa su:

1. Pokrenuti upravljačku sekvencu. Osigurati da je pokrenuta u stanju napajanja.
2. Zagrijavati prvi senzor temperature izlaznog zraka.
   Očekivanje: Nakon što *brd 1* vrijednost dosegne ili prijeđe preko *limit* vrijednost, sustav prelazi u stanje odmora.
3. Zaustaviti grijanje prvog senzora temperature izlaznog zraka.
   Očekivanje: Nakon što *brd 1* vrijednost padne ispod *limit* vrijednosti, a trenutno stanje odmora je završeno, sustav prelazi u stanje napajanja.
4. Zagrijavati drugi senzor temperature izlaznog zraka.
   Očekivanje: Nakon što *brd 2* vrijednost dosegne ili prijeđe preko *limit* vrijednost, sustav prelazi u stanje odmora.
5. Zaustaviti grijanje drugog senzora temperature izlaznog zraka.
   Očekivanje: Nakon što *brd 2* vrijednost padne ispod *limit* vrijednosti, a trenutno stanje odmora je završeno, sustav prelazi u stanje napajanja.

Izvješća je moguće iščitati iz priloga P.6.2. Slijedi opis izvršene procedure testiranja po *total uptime* vrijednosti.

0,0s – 2,6s: Završetak koraka 1 (prvog mjerenja nakon pokretanja upravljačke sekvence). Sustav započinje u stanju napajanja.

5,6s – 38,6s: Prazni hod.

41,6s – 44,5s: Korak 2, prvi dio.

47,5s – 68,8s: Korak 2, drugi dio, i korak 3, prvi dio.

Rezultat: *brd 1* vrijednosti je veća ili jednaka *limit* vrijednosti, sustav je u stanju odmora. Očekivanje u koraku 2 je zadovoljeno.

71,8s – 74,8s: Korak 3, drugi dio.

Rezultat: *brd 1* vrijednosti je manja od *limit* vrijednosti, posljednje stanje odmora je završilo, sustav prelazi u stanje napajanja. Očekivanje u koraku 3 je zadovoljeno.

77,7s – 77,7s: Korak 4, prvi dio.

80,7s – 126,1s: Korak 4, drugi dio, i korak 5, prvi dio.

Rezultat: *brd 2* vrijednosti je veća ili jednaka *limit* vrijednosti, sustav je u stanju odmora. Očekivanje u koraku 4 je zadovoljeno.

128,9s – 131,8s: Korak 5, drugi dio.

Rezultat: *brd 2* vrijednosti je manja od *limit* vrijednosti, posljednje stanje odmora je završilo, sustav prelazi u stanje napajanja. Očekivanje u koraku 5 je zadovoljeno.

Zaključak: Sva očekivanja su ispunjena. Sekvenca nije ni u jednom trenutku bila u stanju pogreške.

6.3. Test neispravnosti senzora tijekom upravljačke sekvence

Ovo je prvi test za zahtjev 5. Ukoliko je bilo koji od temperaturnih senzora nedostupan, može doći do pregrijavanja koje upravljačka sekvenca ne može primijetiti. Iz toga razloga, ona u tom slučaju prelazi u stanje pogreške ispisujući prikladnu poruku:

HOLD-UP: Lost contact with room temperature detector. Please fix it!

HOLD-UP: Lost contact with board temperature detector 1. Please fix it!

HOLD-UP: Lost contact with board temperature detector 2. Please fix it!

Tijekom ovog stanja, kao i bilo kojeg stanja pogreške, signal za napajanje je isključen.

Koraci testa su:

1. Pokrenuti upravljačku sekvencu. Smije biti pokrenut u stanju napajanja ili odmora.
2. Isključiti jedan senzor.

   Očekivanje: Stanje pogreške s prikladnim ispisom.
3. Ukloniti pogrešku.
4. Isključiti još jedan senzor.

   Očekivanje: Stanje pogreške s prikladnim ispisom.

5. Ukloniti pogrešku.
6. Isključiti zadnji senzor.
   Očekivanje: Stanje pogreške s prikladnim ispisom.
7. Ukloniti pogrešku.

Izvješća je moguće iščitati iz priloga P.6.3. Slijedi opis izvršene procedure testiranja po *total uptime* vrijednosti.

0,0s – 2,6s: Završetak koraka 1 (prvog mjerenja nakon pokretanja upravljačke sekvence). Sustav započinje u stanju napajanja.
5,6s – 8,6s: Prazni hod.
10,7 – 12,8: Isključen senzor izlazne temperature 2.
Rezultat: Stanje pogreške s prikladnim ispisom.
15,7 – 15,7: Uključen senzor izlazne temperature 2.
Rezultat: Izlazak iz stanja pogreške.
18,1 – 18,1: Isključen senzor izlazne temperature 1.
Rezultat: Stanje pogreške s prikladnim ispisom.
20,9 – 23,8: Uključen senzor izlazne temperature 1.
Rezultat: Izlazak iz stanja pogreške.
26,8 – 30,9: Isključen senzor okolne temperature.
Rezultat: Stanje pogreške s prikladnim ispisom.
33,7 – 33,7: Uključen senzor okolne temperature.
Rezultat: Izlazak iz stanja pogreške.

Zaključak: Sva očekivanja su ispunjena. Sekvenca nije ni u jednom trenutku bila u drugim stanjima pogreške.

6.4. Test neispravnosti senzora prije pokretanja programa

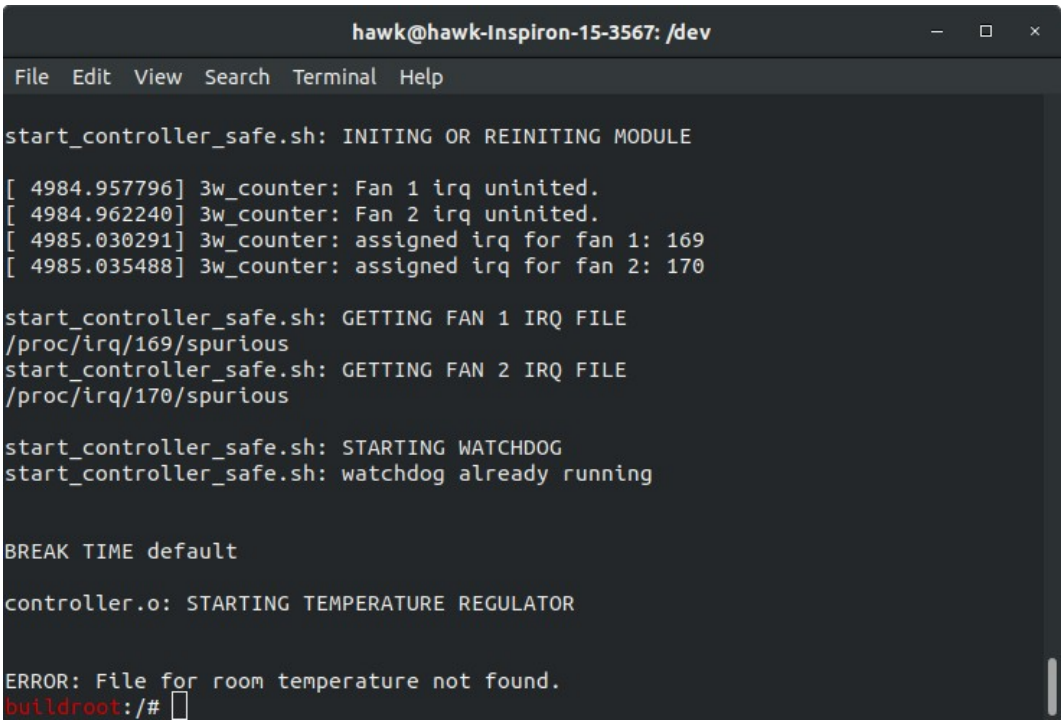
Ovo je drugi test za zahtjev 5 i izvršen je iz istog razloga kao i prvi. Naime, ukoliko su senzori nedostupni prije pokretanja programa, to se tretira kao fatalna pogreška te se prekida program. Upravljačka sekvenca ne može biti pokrenuta pa ništa ne može biti zapisano u log datoteku. Iz toga razloga za ovaj test nije dostupan prilog, već slike konzole.

Koraci izvršavanja:

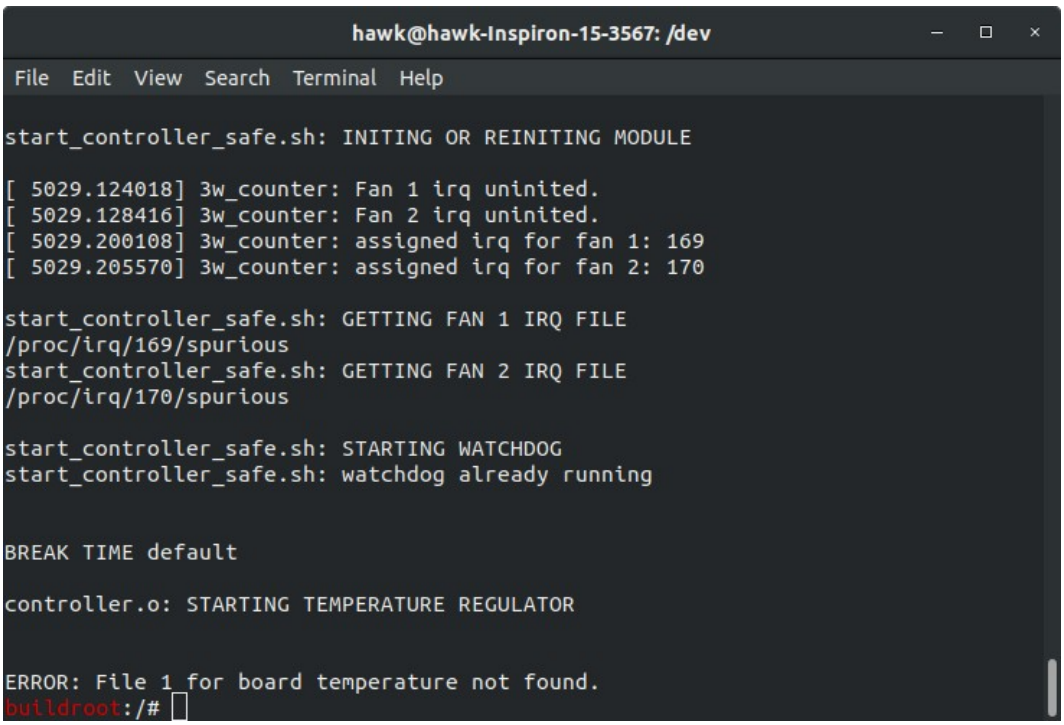
1. Isključiti senzor sobne temperature i pokrenuti program.
   Očekivanje: Fatalna pogreška s prikladnim ispisom.
2. Uključiti senzor sobne temperature.
3. Isključiti senzor 1 temperature okolnog zraka i pokrenuti program.
   Očekivanje: Fatalna pogreška s prikladnim ispisom.
4. Uključiti senzor 1 temperature okolnog zraka.
5. Isključiti senzor 2 temperature okolnog zraka i pokrenuti program.
   Očekivanje: Fatalna pogreška s prikladnim ispisom.
6. Uključiti senzor 2 temperature okolnog zraka.

Na slikama 6.1. - 6.3. demonstrirani su rezultati izvođenja koraka 1, 3 i 5.

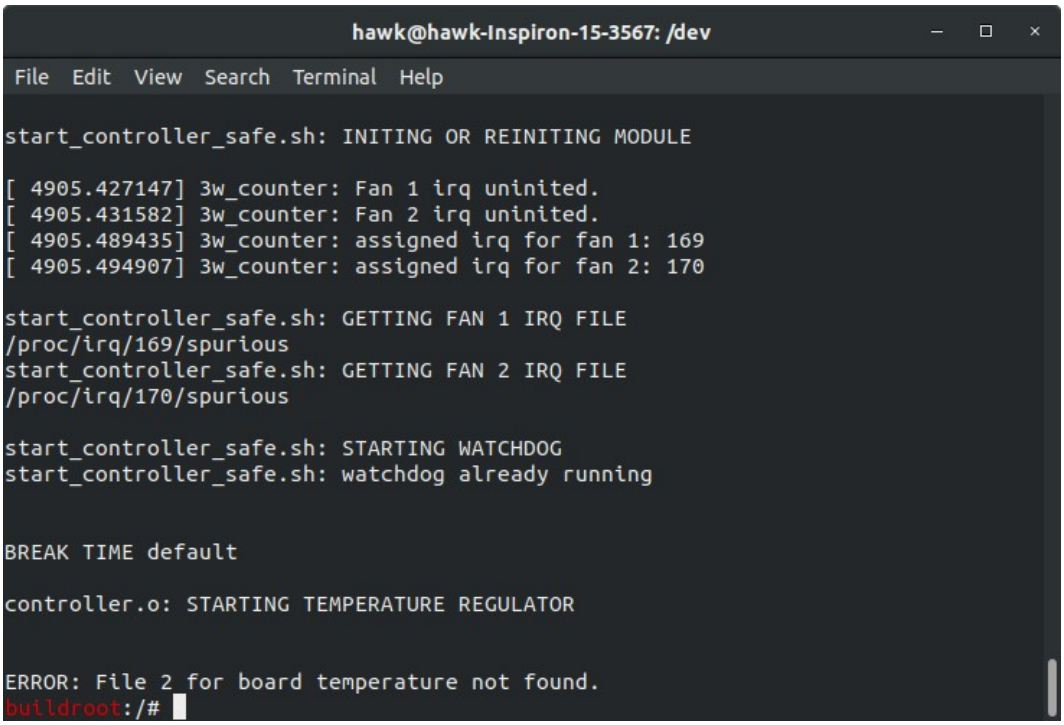
Zaključak: Sva očekivanja su ispunjena.



Slika 6.1. Pokretanje programa bez senzora okolne temperature



Slika 6.2. Pokretanje programa bez senzora 1 izlazne temperature



Slika 6.3. Pokretanje programa bez senzora 2 izlazne temperature

6.5. Test neispravnosti ventilatora, kad su oba potrebna

Ovim testom je provjeren zahtjev 13. Neispravnost ventilatora imala bi utjecaj na zagrijavanje senzora pa bi sustav mogao zapeti u stanju odmora ili čak dovesti do pregrijavanja testnog objekta. Iz tog razloga dizajniran je modul za primjećivanje prekida s pinova namijenjenih

za tahometarski signal ventilatora, a upravljačka sekvenca prati broj prekida kako bi imao informaciju okreću li se ventilatori ili ne. Ukoliko se ne okreću, sekvenca prelazi se u stanje pogreške sa sljedećim ispisom:

HOLD-UP: Fan rotation malfunction!

Naime, stanje navedene pogreške može stići sa zakašnjenjem od cijelog jednog ciklusa mjerenja. Koraci za izvođenje testa su sljedeći:

1. Pokrenuti upravljački program.
2. Kad program upita jesu li potrebna oba ventilatora, odgovoriti pozitivno.
3. Pokrenuti upravljačku sekvencu u stanju napajanja ili odmora.
4. Isključiti jedan od tahometarskih vodova.
   Očekivanje: Stanje pogreške s prikladnim ispisom.
5. Ponovno uključiti tahometarski vod.
   Očekivanje: Izlazak iz stanja pogreške.
6. Isključiti drugi tahometarski vod.
   Očekivanje: Stanje pogreške s prikladnim ispisom.
7. Ponovno uključiti tahometarski vod.
   Očekivanje: Izlazak iz stanja pogreške.

Izvješća je moguće iščitati iz priloga P.6.4. Slijedi opis izvršene procedure testiranja po *total uptime* vrijednosti.

Prije 0,0s: Koraci 1 i 2.

0,0s – 2,6s: Završetak koraka 3 (prvog mjerenja nakon pokretanja upravljačke sekvence). Sustav započinje u stanju napajanja.

5,5s – 5,5s: Prazni hod.

8,4s – 14,1s: Prvi tahometarski vod isključen.

Rezultat: Stanje pogreške s prikladnim ispisom, kašnjenje 1 ciklus mjerenja.

16,9s – 22,8s: Prvi tahometarski vod ponovno uključen.

Rezultat: Izlazak iz stanja pogreške.

25,8s – 31,5s: Drugi tahometarski vod isključen.

Rezultat: Stanje pogreške s prikladnim ispisom, kašnjenje 1 ciklus mjerenja.

37,4s – 40,6s: Drugi tahometarski vod ponovno uključen.

Rezultat: Izlazak iz stanja pogreške.

Zaključak: Sva očekivanja su ispunjena. Sekvenca nije ni u jednom trenutku bila u drugim stanjima pogreške.

6.6. Test neispravnosti ventilatora, kad je samo jedan potreban

Uz zahtjev 13, program implementira i dodatnu uslugu, a to je odabir je li potreban samo jedan ventilator (tahometarski signal) ili dva. Ovaj je test izvršen iz istoga razloga kao i prošli.

Stanje pogreške može stići sa zakašnjenjem od cijelog jednog ciklusa mjerenja. Koraci za izvođenje testa su sljedeći:

1. Pokrenuti upravljački program.
2. Kad program upita jesu li potrebna oba ventilatora, odgovoriti pozitivno.
3. Pokrenuti upravljačku sekvencu u stanju napajanja ili odmora.
4. Isključiti jedan od tahometarskih vodova.
   Očekivanje: Sekvenca nije u stanju pogreške.
5. Ponovno uključiti tahometarski vod.
6. Isključiti drugi tahometarski vod.
   Očekivanje: Sekvenca nije u stanju pogreške.
7. Ponovno uključiti tahometarski vod.
8. Isključiti oba tahometarska voda.
   Očekivanje: Stanje pogreške s prikladnim ispisom.
9. Ponovno uključiti jedan ili oba tahometarska voda.
   Očekivanje: Izlazak iz stanja pogreške.

Izvješća je moguće iščitati iz priloga P.6.5. Slijedi opis izvršene procedure testiranja po *total uptime* vrijednosti.

Prije 0,0s: Koraci 1 i 2.

0,0s – 2,7s: Završetak koraka 3 (prvog mjerenja nakon pokretanja upravljačke sekvence). Sustav započinje u stanju odmora.

5,7s – 11,5s: Prazni hod.

14,4s – 21,2s: Prvi tahometarski vod isključen.

Rezultat: Sekvenca nije u stanju pogreške.

24,2s – 27,1s: Prvi tahometarski vod ponovno uključen.

30,2s – 36,1s: Drugi tahometarski vod isključen.

Rezultat: Sekvenca nije u stanju pogreške.

39,0s – 42,1s: Drugi tahometarski vod ponovno uključen.

45,0s – 50,8s: Oba tahometarska voda isključeni.

Rezultat: Stanje pogreške s prikladnim ispisom, kašnjenje 1 ciklus mjerenja.

53,6s – 56,6s: Tahometarski vodovi ponovno uključeni.

Rezultat: Izlazak iz stanja pogreške.

7. EKSPERIMENTALNA ANALIZA

7.1. Eksperimentalno okruženje

Eksperimenti u ovom radu napravljeni su kako bi se odgovorilo na sljedeća pitanja:

1. Ukoliko okolna temperatura odstupa od 25°C, kako promijeniti graničnu temperaturu definiranu u *boards.txt* datoteci tijekom upravljačke sekvence?
2. Kako bi korisnik odredio graničnu temperaturu pri 25°C?
3. Koje je preporučeno vrijeme odmora (kad je pregrijavanje detektirano i glavno napajanje isključeno), za testne objekte sa sličnim fizikalnim svojstvima kao u ovom dizajnu?

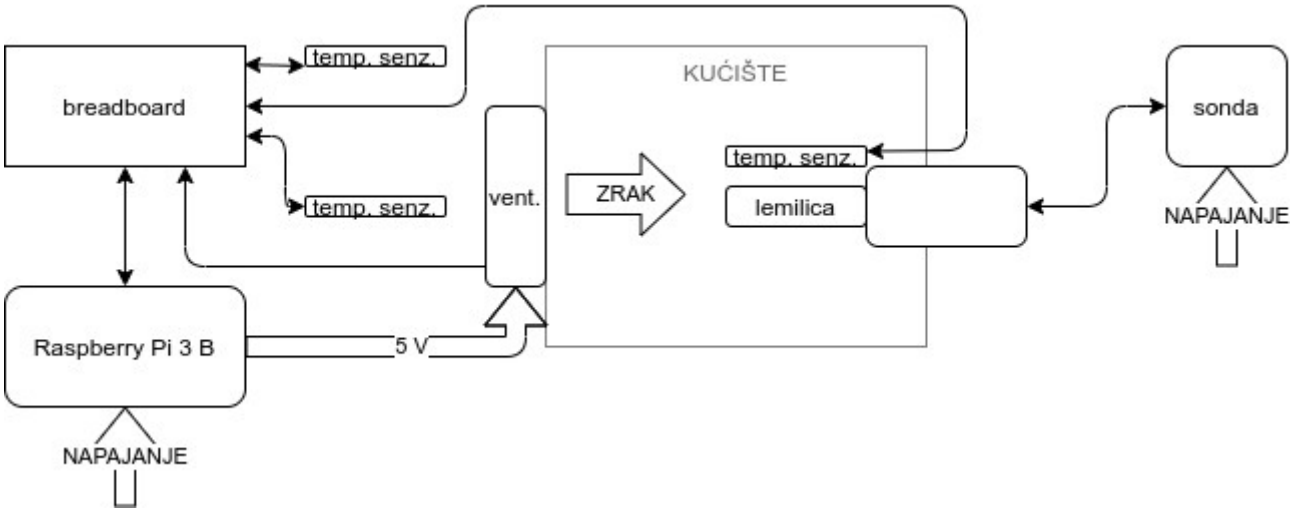
Kako bi eksperimentalna analiza bila napravljena, potreban je testni objekt ili sustav koji funkcionira na sličan način (lažni testni objekt). Kako se radi o čistoj regulaciji temperature, potreban je element koji se zagrijava u kućištu otvorenom na suprotnim stranama kroz koje bi trebao protjecati zrak s ventilatora. Element koji se aktivno zagrijava potrebno je staviti u kućište i jedan od senzora temperature izlaznog zraka je potrebno pričvrstiti u njegovoj blizini. Na stranu koja je udaljenija od elementa i senzora potrebno je namjestiti ventilator, a iza ventilatora pričvrstiti senzor temperature ulaznog zraka.

Kao element koji se zagrijava korištena je lemilica sa sondom, preko koje se može upravljati njenom temperaturom. Lemilica može raditi pri temperaturama od 150°C do 450°C. Ona se grije postepeno, tako da je njome moguće simulirati i dinamiku sustava za koji je ovaj upravljački sustav namijenjen.

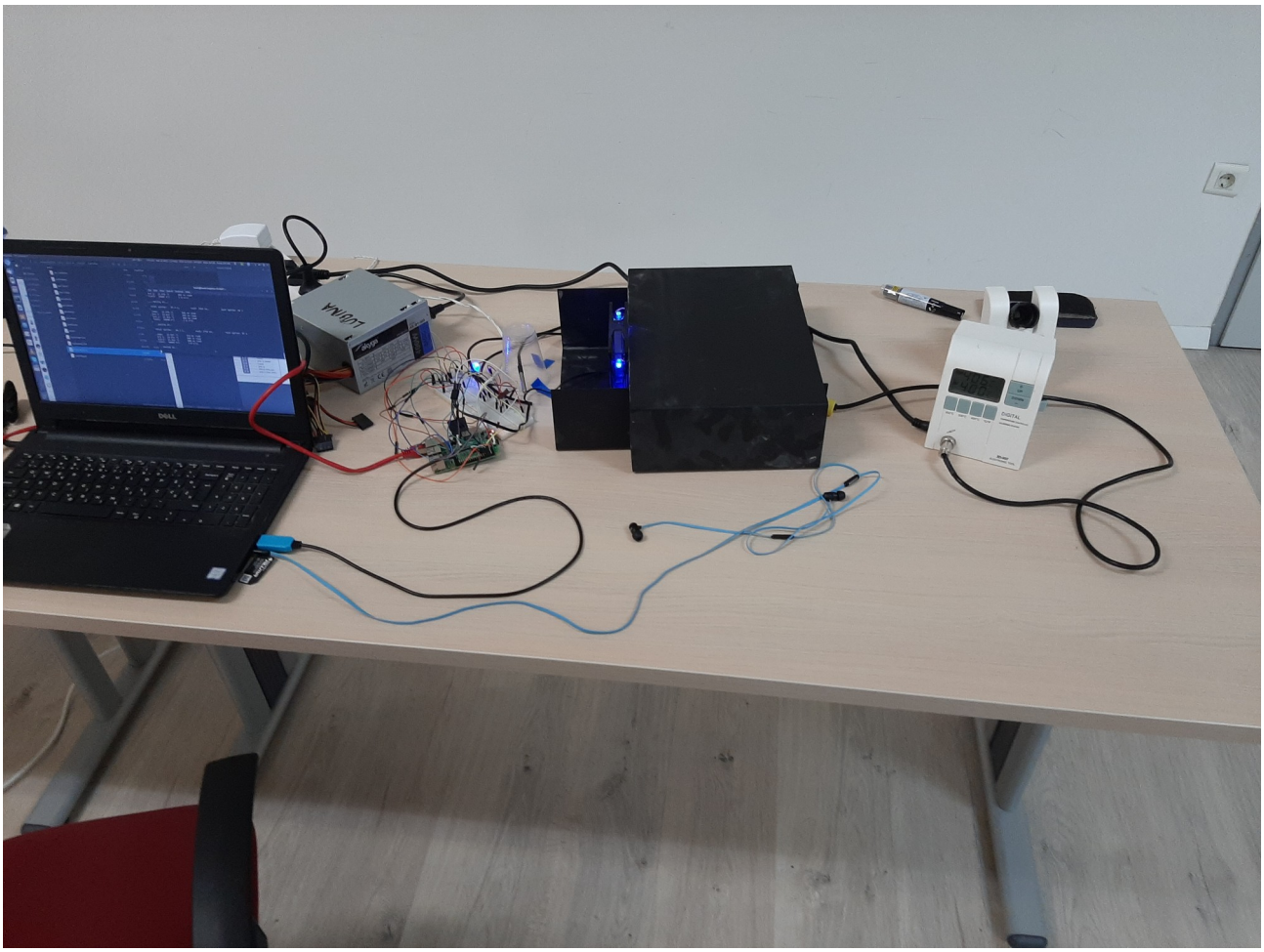
Za pričvršćivanje senzora korištene su dvije plastične čaše, a za kućište dva metalna elementa. Drugi senzor za mjerenje izlazne temperature korišten je kao pričuvni senzor za mjerenje temperature okolnog zraka te u eksperimentu 1 za to i primijenjen. (Naime, identifikacija namjene senzora ovisi o GPIO portu na koji je njegov signalni vod spojen.) Ventilator je spojen na izvor od 5V.

Prvim promatranjima uočeno je da vodootporni senzori imaju sporiji odziv na promjenu temperature okolnog zraka. Ali u realnoj situaciji, korisnik bi mogao koristiti 1-*Wire* senzore sličnih svojstava.

Analiza je napravljena uz pomoć Python skripti, a za generiranje grafičkih prikaza korišteno je *plotly* [14] sučelje. Pojednostavljena shema eksperimentalnog okruženja prikazana je na slici 7.1., a na slici 7.2. je njegova fotografija.



Slika 7.1. Pojednostavljena shema



Slika 7.2. Fotografija eksperimentalnog okurženja

7.2. Eksperiment 1 - Ovisnost izlazne temperature o okolnoj temperaturi

7.2.1. Organizacija eksperimenta

U datoteci *boards.txt* moguće je zadati gornje temperaturne granice za izlazni zrak, za pojedini testni objekt, pri u okolini od 25°C. No, nakon puštanja sustava u pogon okolna temperatura ne mora biti blizu 25°C, za koju su zadane temperaturne granice.

Kako se sustav napravljen u ovom radu uvelike oslanja na temperaturu izlaznog zraka, koja je posljedica temperature ulaznog (okolnog) zraka i procesa hlađenja testnog objekta (odnosno zagrijavanja zraka), potrebno je izvesti matematičku funkciju koja će zadržati zadanu temperaturnu granicu za okolnu temperaturu od oko 25°C, a promijeniti ju za svako odstupanje okolne temperature od 25°C. Dakle radi se o identifikaciji sljedeće funkcije:

$$t_l = f(t_z, t_o) \tag{7-1}$$

U jednadžbi je t_l primijenjena temperaturna granica, t_z granica zadana za okolinu od 25°C, a t_o okolna temperatura. Budući da različiti testni objekti s istim temperaturama pregrijavanja procesora mogu rezultirati različitim t_z vrijednostima, temperatura pregrijavanja procesora ili u ovom slučaju lemilice, nije uzeta u obzir kod određivanja ove funkcije.

Odabran je eksperimentalni pristup određivanja potrebne matematičke funkcije sa sljedećim varijablama:

- nezavisna varijabla je t_o (okolna temperatura)
- zavisna varijabla je t_{iz} (izlazna temperatura)
- kontrolna varijabla je t_p (temperatura testnog objekta, odnosno lemilice)

Eksperiment je bio ponovljen za različite vrijednosti kontrolne varijable. Svrha različitih vrijednosti kontrolne varijable je povećanje količine podataka kako bi bilo moguće odabrati najbolju seriju podataka za određivanje *f* funkcije. Proces eksperimenta je sljedeći:

1. Ohladiti prostoriju ispod 21,5°C.
2. Aktivirati upravljački uređaj, ukoliko još nije aktiviran, kako bi se pokrenuo ventilator.
3. Podesiti lemilicu na određenu temperaturu i pričekati da se njena temperatura stabilizira oko zadane.
4. Pričekati da se očitana vrijednost temperature izlaznog zraka stabilizira.
5. Započeti automatizirano bilježenje okolne i izlazne temperature.
6. Započeti zagrijavanje prostorije.
7. Nakon što okolna temperatura dosegne 28,5°C, prekinuti bilježenje temperatura.

Za hlađenje prostorije u koraku 1 su se jednostavno otvorili njeni prozori, a za njeno zagrijavanje u koraku 6 korištena je sobna peć s upuhivanjem i ispuhivanjem zraka te dodatno klima uređaj podešen na zagrijavanje.

Za izvršavanje koraka 1, 4 i 5 korištena je privremena verzija upravljačkog programa koja ne uključuje pravilan način određivanja granične temperature u ovisnosti o okolnoj temperaturi (koristi *placeolder* formulu). Kako signal za uključenje napajanja testnog objekta nije važan za ovaj eksperiment, tako je odabir opcije iz *boards.txt* datoteke nevažan.

Logging je bio uključen za korak 5. Nakon završetka eksperimenta log datoteka je preimenovana kako bi se za ponavljanje eksperimenta generirala nova *log.txt* datoteka i lakše izvršila analiza.

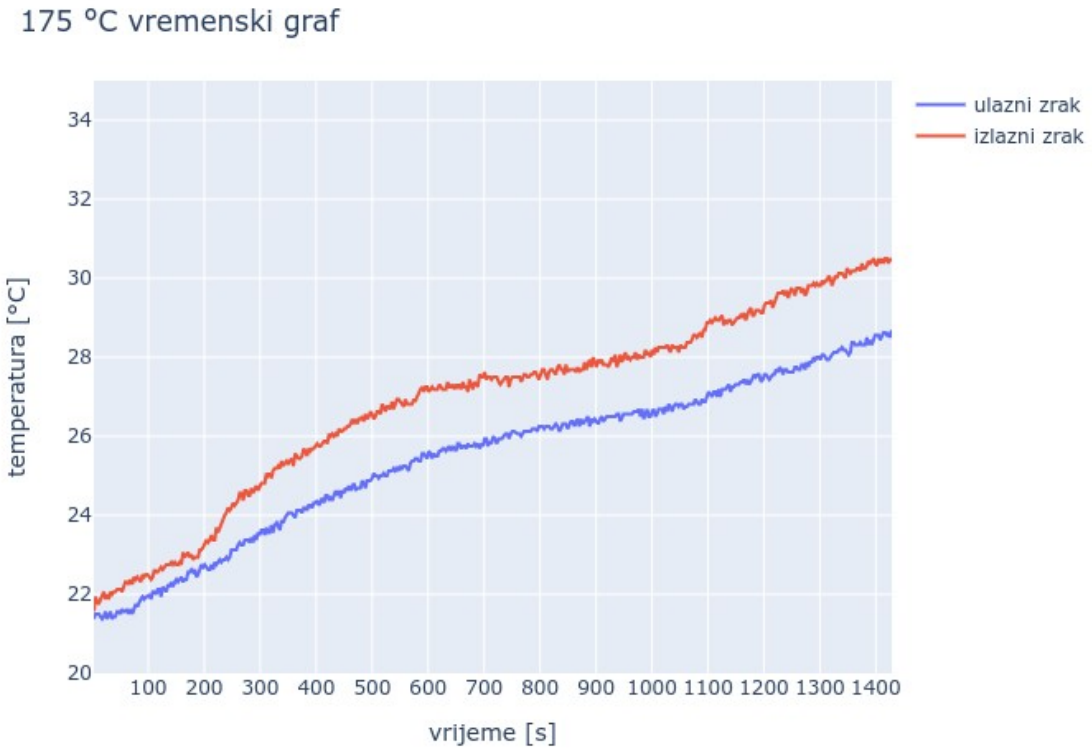
Eksperiment je bio ponovljen za različite temperature lemilice (koja je zadana u koraku 2). U tablici 7.1. su opisane duljine izvršenja eksperimenta (zaokružene na sekundu) za pojedine temperature lemilice, kao i redoslijed primjena temperatura lemilice. Za povećanu duljinu izvršavanja prva dva ponavljanja eksperimenta odgovorne su temperature zidova (koji još nisu bili dovoljno zagrijani jer je eksperiment počeo u 9 ujutro) te temperatura zraka izvan zgrade (koja je tad bila niska).

Broj ponavljanja	Temperatura lemilice	Trajanje eksperimenta
1	150	31min, 11s
2	175	23min, 48s
3	200	14min, 39s
4	225	11min, 37s
5	250	12min, 24s
6	275	10min, 75s
7	300	12min, 50s
8	350	13min, 35s
9	400	12min 44s

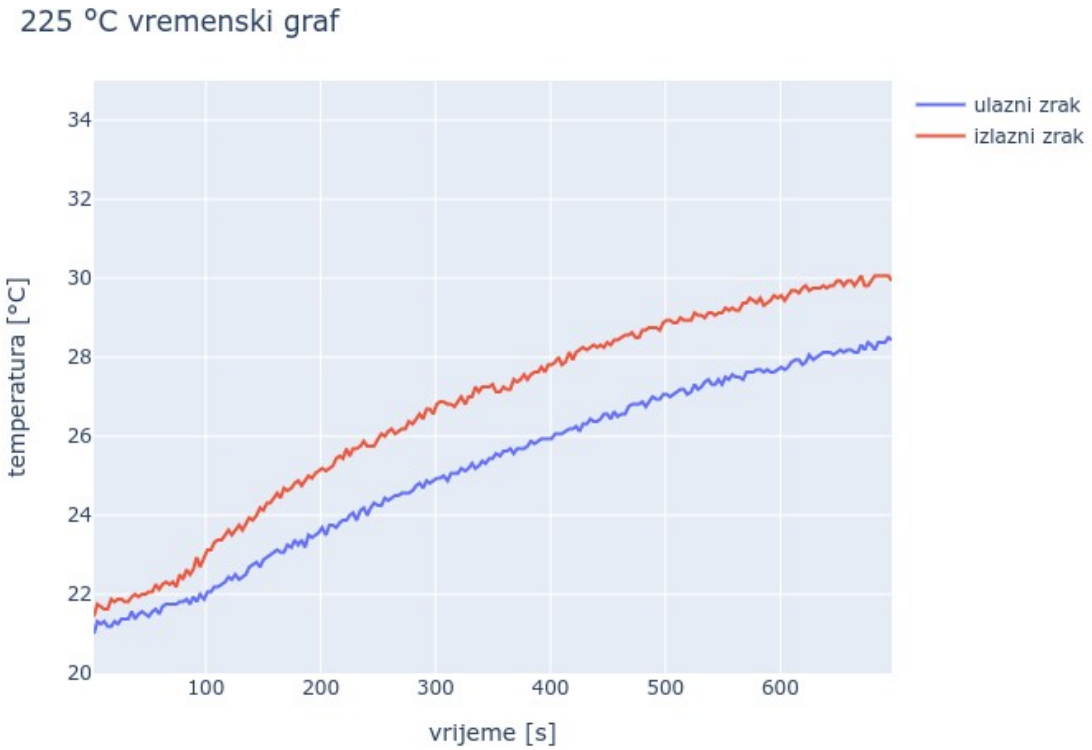
Tablica 7.1

Skripta *expparse.py* korištena je za parsiranje log datoteka, a skripta *exp1graph.py* za generiranje grafičkih prikaza. Za okolnu temperaturu korišteni su podatci drugog senzora izlazne temperature (koji je za to bio i pozicioniran i mogao se tako primijeniti). U ostalim eksperimentima su korišteni podatci senzora sobne temperature (nakon prespajanja). Na slikama 7.3. - 7.6. su primjeri vremenskog odziva temperatura okoline i izlaznog zraka. Ista aproksimacija može biti napravljena za realni testni objekt.

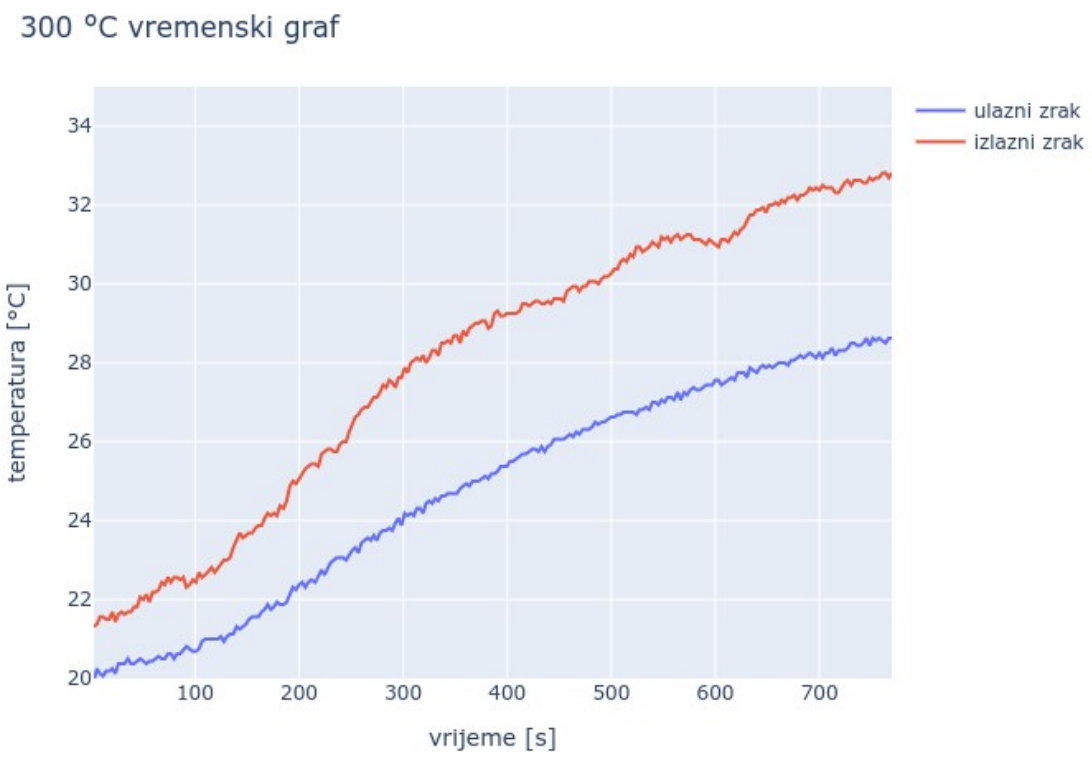
7.2.2. Analiza eksperimenta



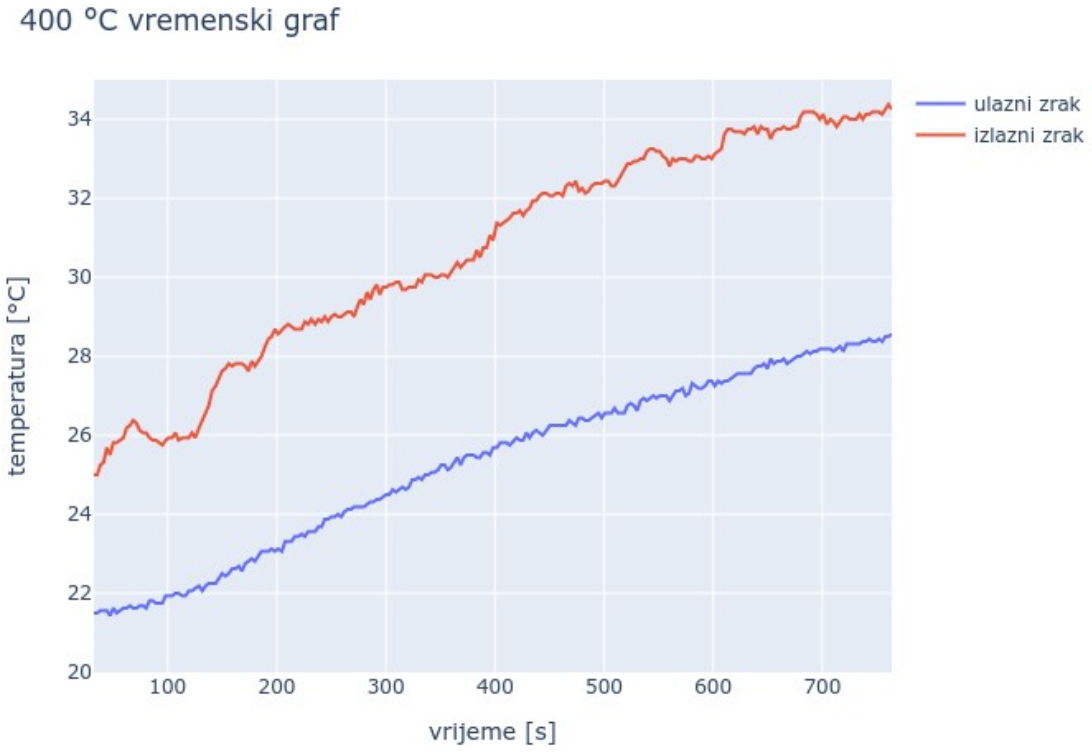
Slika 7.3. Vremenski odziv za kontrolnu varijablu 175°C



Slika 7.4. Vremenski odziv za kontrolnu varijablu 225°C

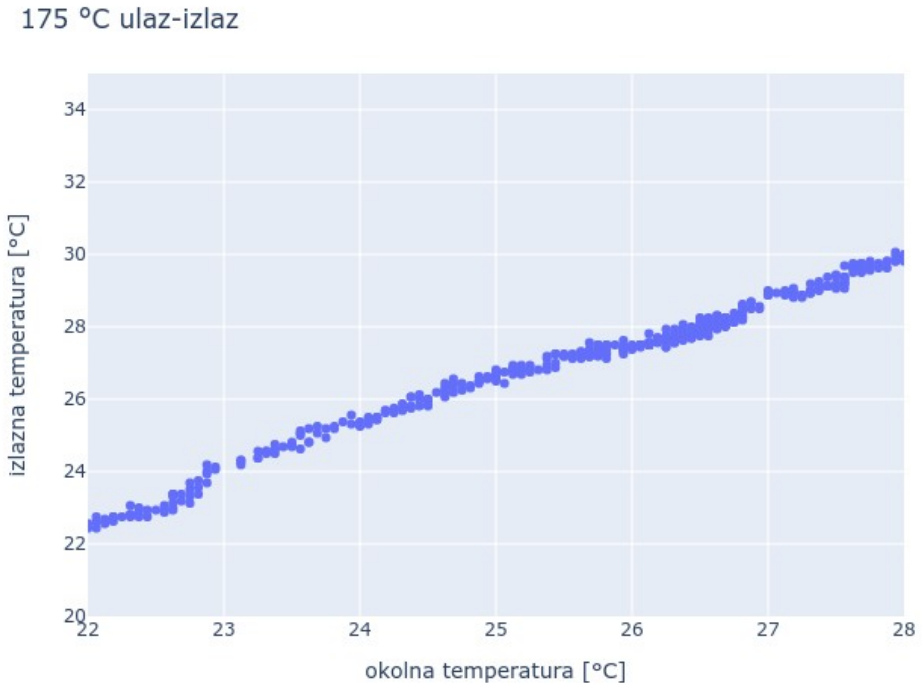


Slika 7.5. Vremenski odziv za kontrolnu varijablu 300°C

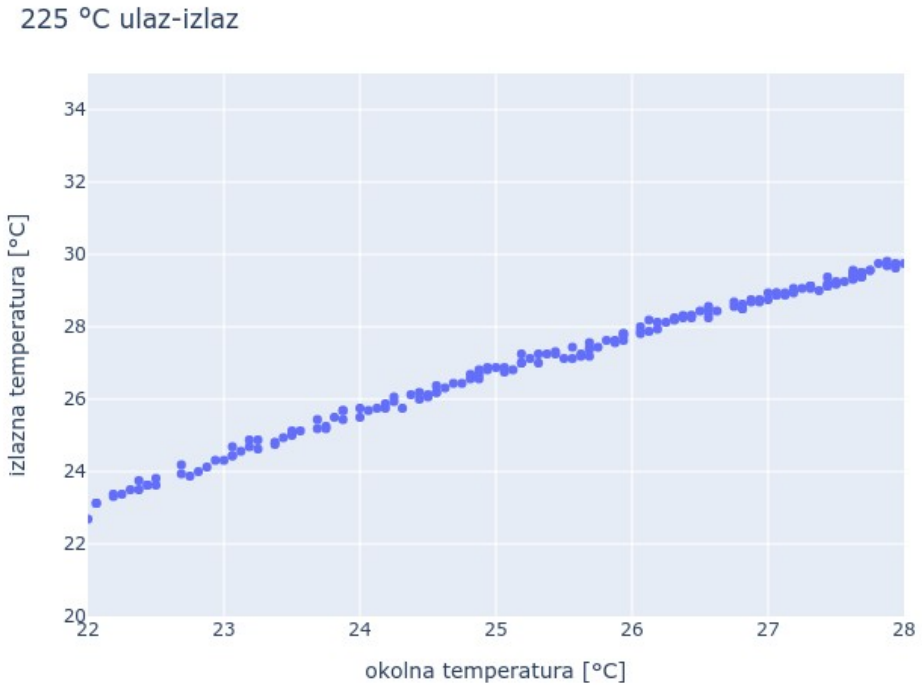


Slika 7.6. Vremenski odziv za kontrolnu varijablu 400°C

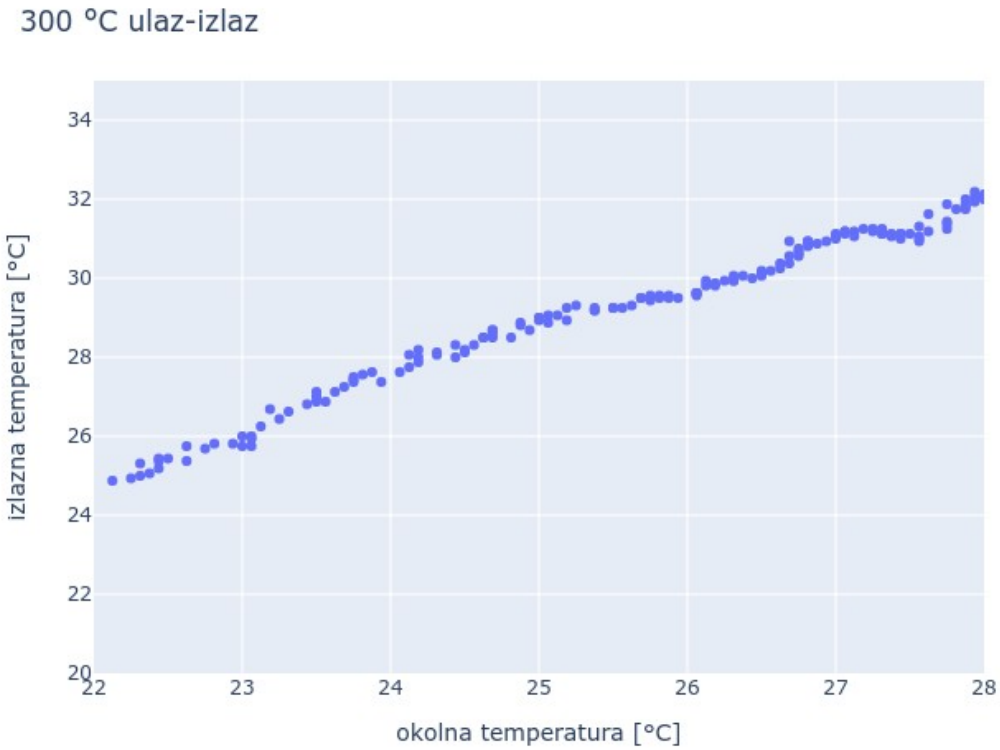
Na slikama 7.7. - 7.10. prikazane su temperature izlaznog i ulaznog zraka izmjerene u istim vremenskim trenutcima. U obzir su uzeti samo podatci uz okolne temperature 22°C - 28°C. Radi smanjenja broja slika, prikazane su serije mjerenja za kontrolne varijable 175°C, 225°C, 300°C i 400°C.



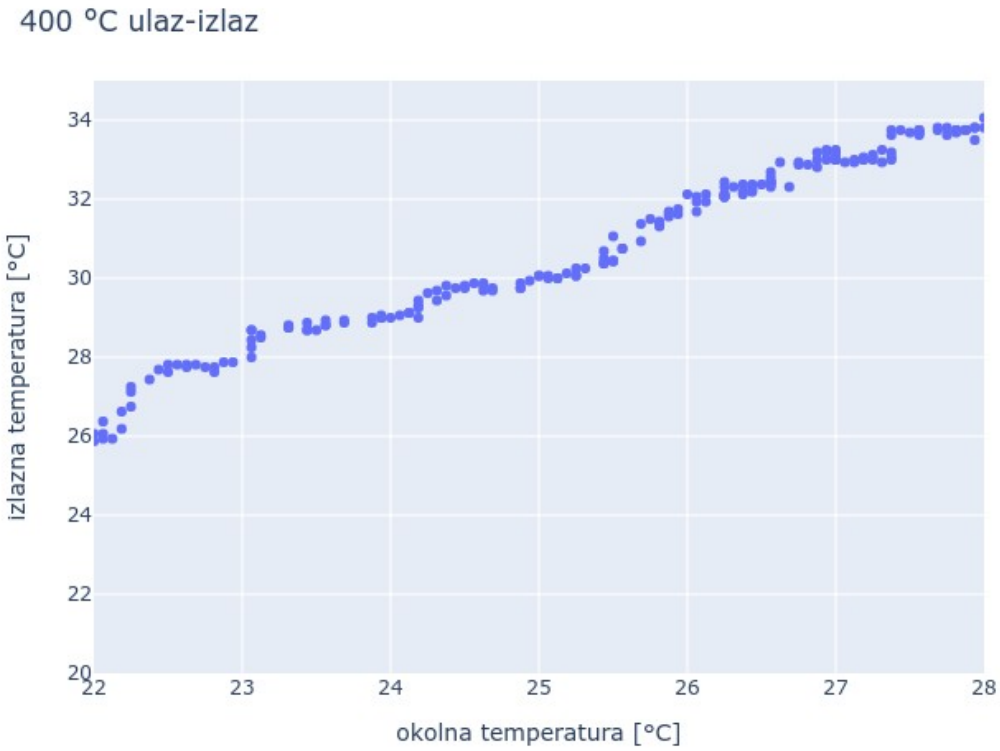
Slika 7.7. Prikaz okolnih i izlaznih temperatura zraka za kontrolnu varijablu 175°C



Slika 7.8. Prikaz okolnih i izlaznih temperatura zraka za kontrolnu varijablu 225°C



Slika 7.9. Prikaz okolnih i izlaznih temperatura zraka za kontrolnu varijablu 300°C



Slika 7.10. Prikaz okolnih i izlaznih temperatura zraka za kontrolnu varijablu 400°C

Kako bi se odredila funkcija $f(t_z,t_o)$, prvo je bilo potrebno odrediti funkciju odnosa okolne i izlazne temperature zraka, koja se naziva $g(t_o)$.

$$\widehat{t_{iz}} = g(t_o) \tag{7-2}$$

Iz slika 7.7. - 7.10. moguće je uočiti zakrivljenost krivulja te je za aproksimaciju dobivenih rezultata uzeta kvadratna funkcija $g(t_o)$.

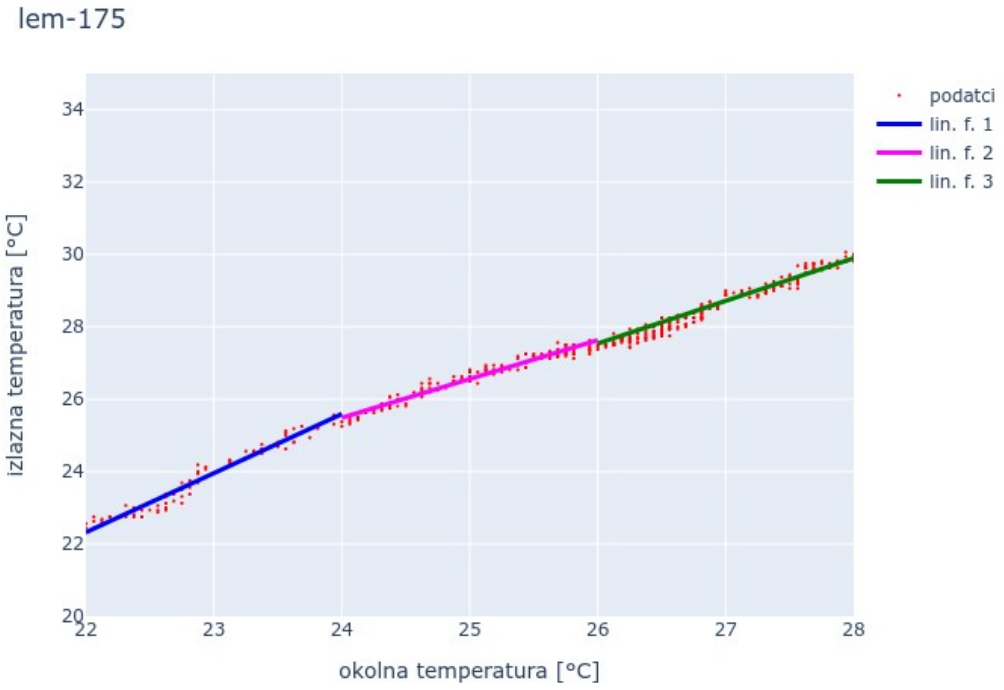
$$g(t_o) = a t_o^2 + b t_o + c \tag{7-3}$$

U prethodno prikazanim podatcima vidljivi su niskofrekvencijski poremećaji. Primjerice, na slici 7.7. oko 22,5°C okolne temperature, a na slici 7.9. oko 27°C. Navedeni poremećaji pretežito su ograničeni na sljedeće intervale: *[22, 24]*, *[24, 26]* i *[26, 28]*. Navedeni poremećaji se uočavaju u samo jednom od navedenih intervala, osim kad je temperatura lemilice 400°C gdje su takvi poremećaji posvuda pa se u sljedećim koracima ta serija mjerenja ne uzima u obzir.

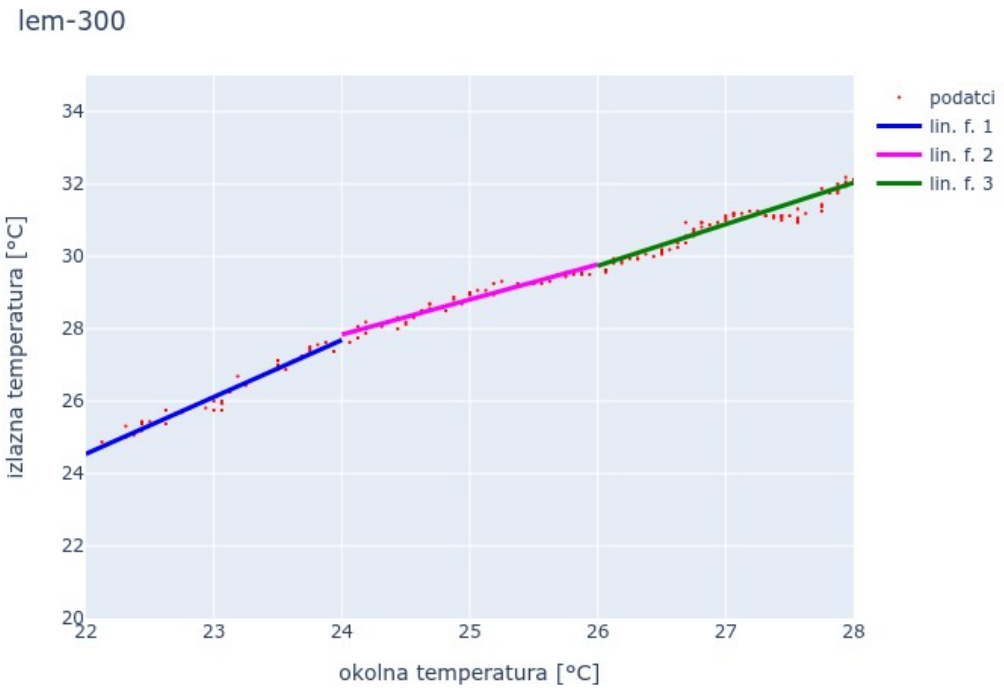
Iz toga razloga aproksimacija je izvršena u sljedećim koracima:

1. Rastaviti podatke u tri skupa za t_o u sljedećim intervalima:
$$t_o \in [22,24],\ t_o \in \langle 24,26 \rangle\ i\ t_o \in [26,28] \tag{7-4}$$
2. Napraviti jednostavnu linearnu regresiju (izračunavanjem koeficijenata nagiba *a* i pomaka *b*) za svaki od ta 3 skupa. Rezultirajuće funkcije su označene s l_1, l_2 i l_3.
$$l_{broj}(t_o) = a_{broj} t_o + b_{broj} \tag{7-5}$$
$$l_1(t_o)\ za\ t_o \in [22,24], \quad l_2(t_o)\ za\ t_o \in \langle 24,26 \rangle, \quad l_3(t_o)\ za\ t_o \in [26,28] \tag{7-6}$$
3. Ukloniti jednu aproksimaciju *l*, sa prevelikim ili premalim nagibom.
   Ako točno jedna aproksimacija *l* ima nagib izvan intervala <1.0, 1.6>, ukloni ju.
   U protivnom ukloni aproksimaciju *l* čiji nagib najviše odstupa od 1.5. (Naime, manji nagibi se više „kažnjavaju" pa se ne traži odstupanje od sredine spomenutog intervala, već od 1,5.)
4. Za preostale dvije funkcije, izračunati sljedeće točke po sljedećim formulama.
$$p_1 = l_1(22),\ p_2 = l_2(25),\ p_3 = l_3(28) \tag{7-7}$$
5. Za vrijednost *p* koja nije mogla biti izračunata napraviti sljedeće:
$$(nedostaje\ p_1) \Rightarrow p_1 = \begin{cases} l_2(22), & za\ l_2(22) < l_3(22) \\ l_3(22), & za\ l_3(22) < l_2(22) \end{cases}$$
$$(nedostaje\ p_2) \Rightarrow p_2 = \begin{cases} l_1(25), & za\ l_1(25) < l_3(25) \\ l_3(25), & za\ l_3(25) < l_1(25) \end{cases} \tag{7-8}$$
$$(nedostaje\ p_3) \Rightarrow p_3 = \begin{cases} l_1(28), & za\ l_1(28) < l_2(28) \\ l_2(28), & za\ l_2(28) < l_1(28) \end{cases}$$
6. Izračunati kvadratnu funkciju kroz točke *p*.

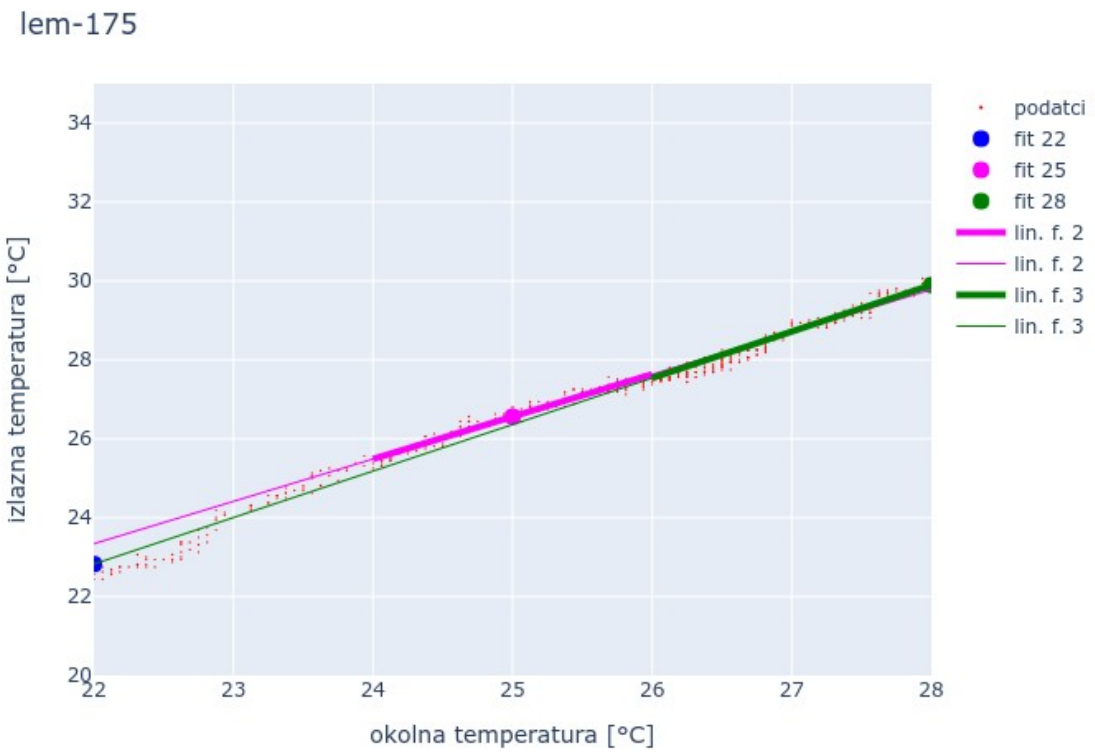
Aproksimacija za pojedinu seriju mjerenja je nazvana $g_{vlastita}(t_o)$. Za demonstraciju su uzete serije za temperature lemilice 175°C i 300°C. Na slikama 7.11. i 7.12. prikazane su aproksimacije svih triju linearnih funkcija (korak 2), na slikama 7.13. i 7.14. određivanje nedostajuće *p* točke (korak 5), a na slikama 7.15 i 7.16 izračunate parabole s naznačenim točkama *p* (korak 6).



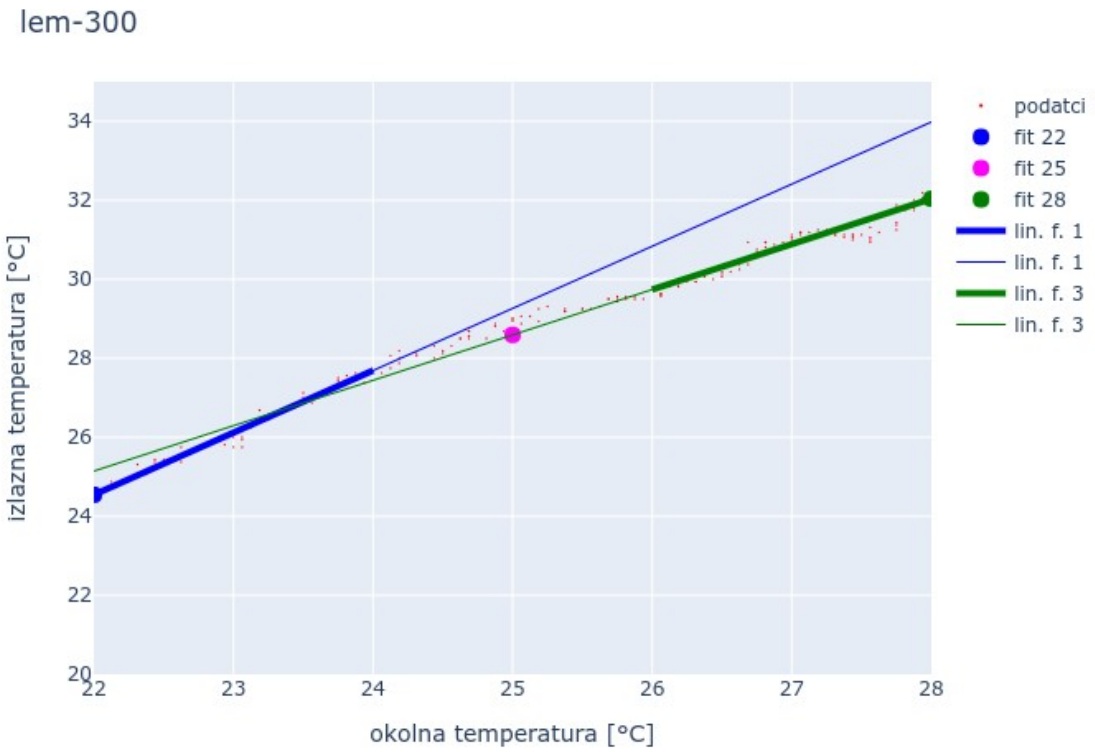
Slika 7.11. Aproksimirane linearne funkcije za kontrolnu varijablu 175°C



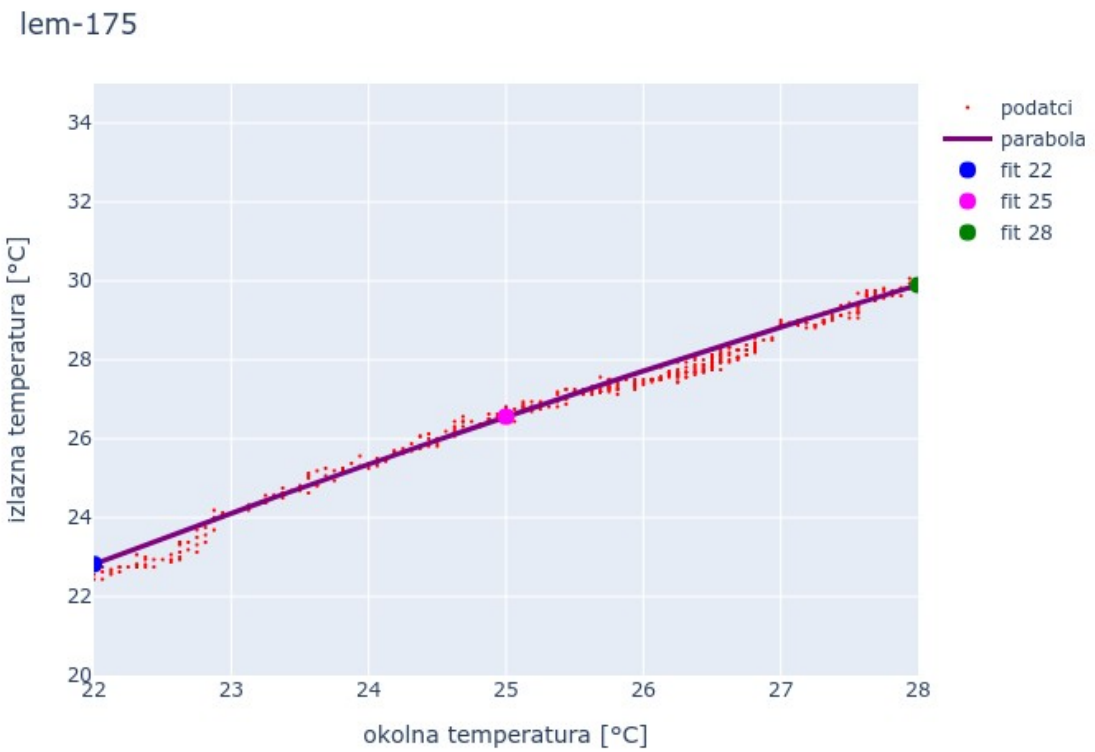
Slika 7.12. Aproksimirane linearne funkcije za kontrolnu varijablu 300°C



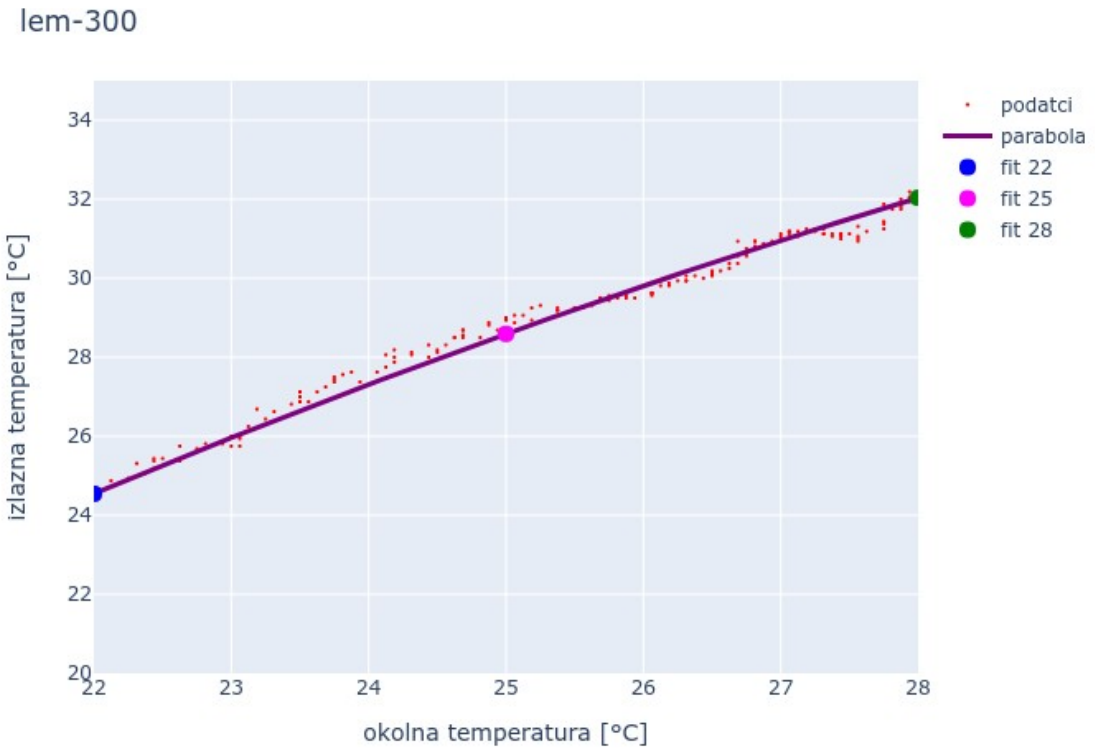
Slika 7.13. Određivanje nedostajuće točke p za kontrolnu varijablu 175°C



Slika 7.14. Određivanje nedostajuće točke p za kontrolnu varijablu 300°C

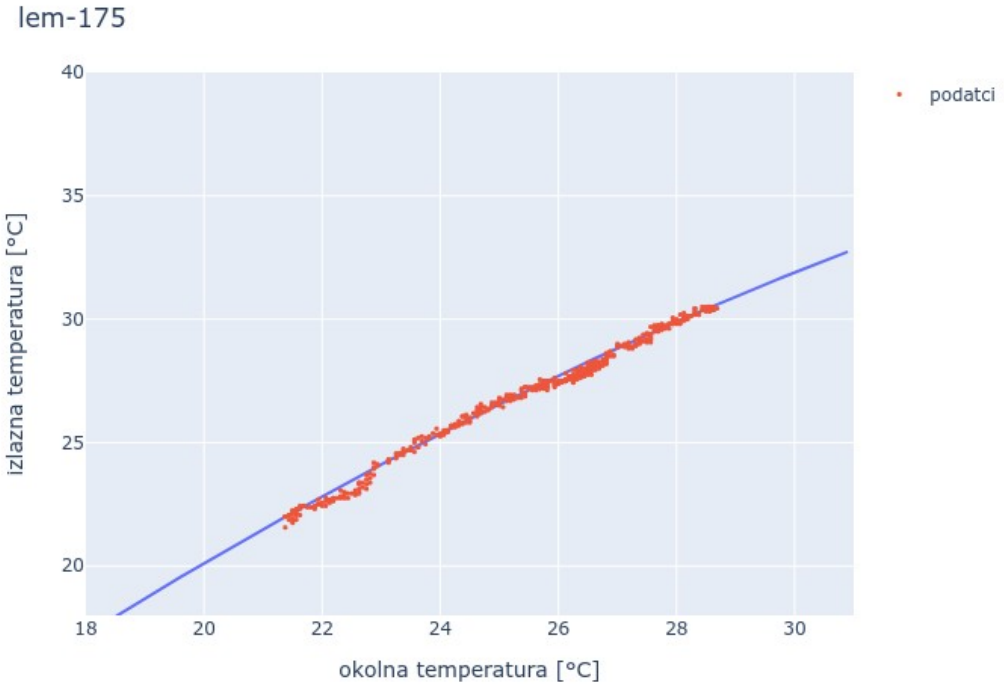


Slika 7.15. Parabola kroz 3 točke za kontrolnu varijablu 175°C

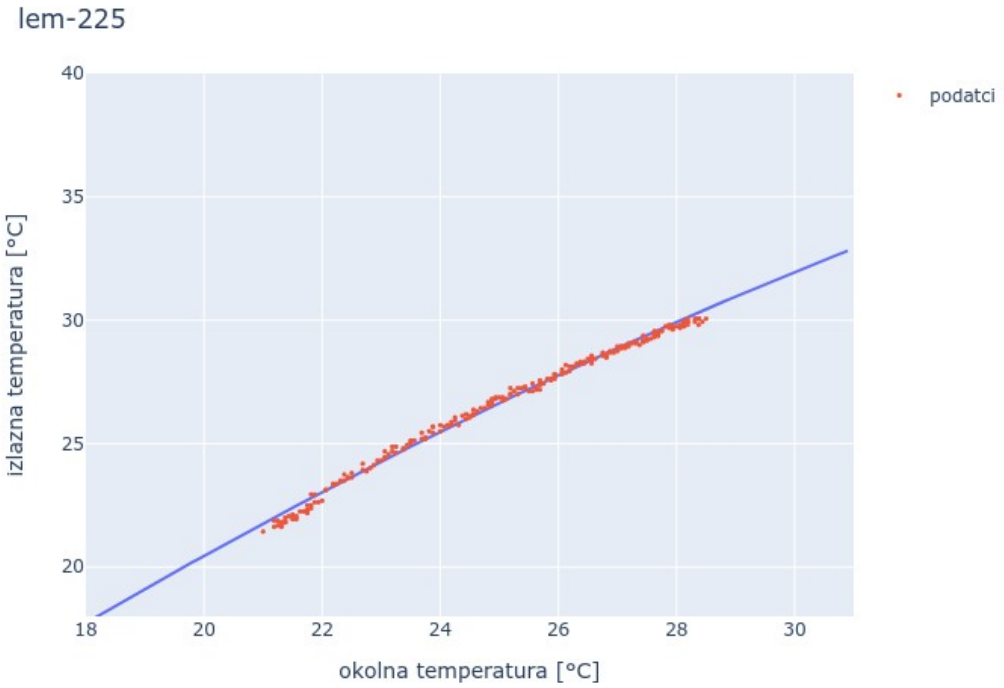


Slika 7.16. Parabola kroz 3 točke za kontrolnu varijablu 300°C

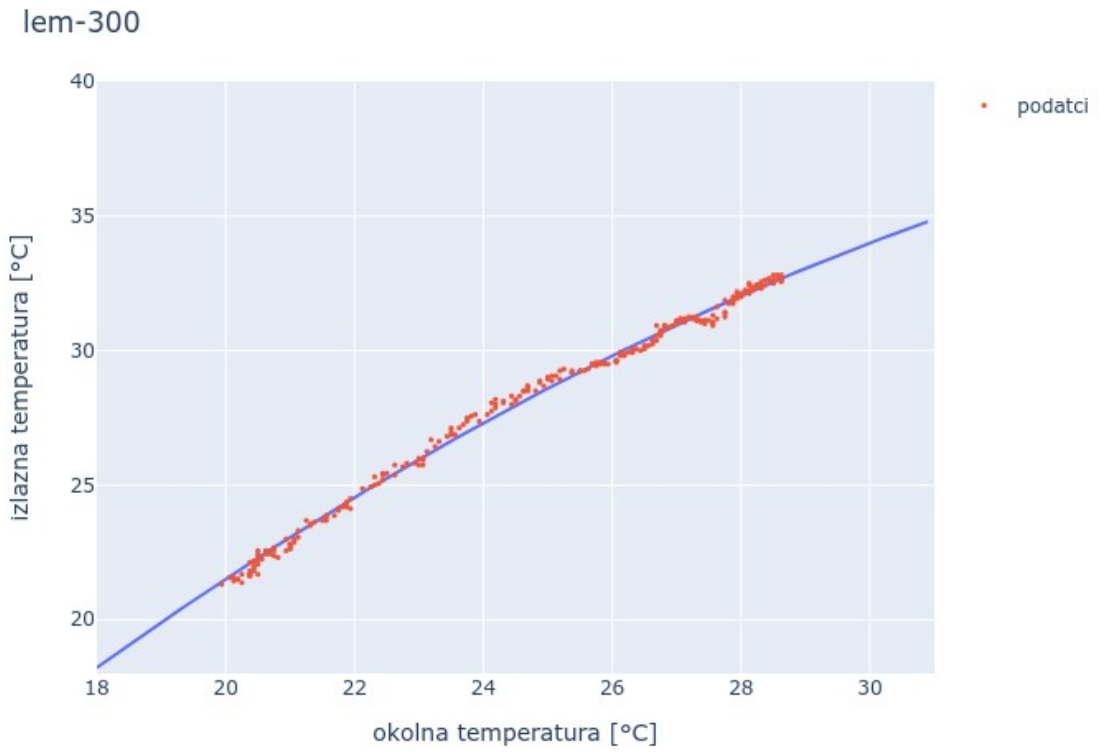
Tom metodom dobivena je aproksimacija $g_{vlastita}(t_o)$. Na slikama 7.17 - 1.19 prikazane su aproksimacije funkcije $g_{vlastita}(t_o)$ za kontrolne varijable 175°C, 225°C i 300°C, u proširenom intervalu t_o.



Slika 7.17. Aproksimacija g(t_o) uz kontrolnu varijablu 175°C



Slika 7.18. Aproksimacija g(t_o) uz kontrolnu varijablu 225°C



Slika 7.19. Aproksimacija $g(t_o)$ uz kontrolnu varijablu 300°C

Kako bi se efektivno spriječilo pregrijavanje u primjeni, pri određivanju konačne funkcije $g(t_o)$ korisno je uzeti $g_{vlastita}(t_o)$ aproksimaciju s većim nagibom u intervalu [22,28]. Stoga je uzeta aproksimacija koja demonstrira veće razlike nagiba. Iz aproksimacija $g_{vlastita}(t_o)$ se odabire funkcija $g_{ori}(t_o)$ po sljedećem kriteriju: apsolutna vrijednost kvadratnog koeficijenta treba biti što veća, a izračun $g_{vlastita}(18)$ što manji. Odabrana je $g_{vlastita}(t_o)$ aproksimacija uz kontrolnu varijablu 300°C.

$$g_{ori}(t_o) = -0{,}033359205995710774\ t_o^2 + 2{,}9169264896641725\ t_o - 23{,}482259640699475 \quad (7\text{-}9)$$

Za primjenu u upravljačkom programu bilo bi dobro da su svi koeficijenti zaokruženi na 4 decimalna mjesta. No za slučaj prvog i drugog koeficijenta, takvo zaokruživanje i modifikacija udaljuju funkciju od empirijskih podataka pa je za to iskorištena sljedeća korekcija:

1. Zadati privremenu $g_{pr}(t_o)$ funkciju zaokruživanjem koeficijenata funkcije $g_{ori}(t_o)$:

$$g_{pr}(t_o) = -0{,}0334\,t_o^2 + 2{,}9169\,t_o - 23{,}4822 \quad (7\text{-}10)$$

2. Odrediti konačnu $g(t_o)$ funkciju promjenom slobodnog koeficijenta funkcije $g_{pr}(t_o)$, tako da $g(25)$ i $g_{ori}(25)$ budu jednaki.

$$g(t_o) = g_{pr}(t_o) + (g_{ori}(25) - g_{pr}(25)) \quad (7\text{-}11)$$

3. Zaokružiti slobodni koeficijent u konačnoj $g(t_o)$ funkciji.

Konačna *g* funkcija glasi:

$$g(t_o) = -0{,}0334\,t_o^2 + 2{,}9169\,t_o - 23{,}4561 \tag{7-12}$$

Za ostvarenje cilja eksperimenta, a to je određivanje funkcije $f(t_z,t_o)$, gdje je t_z granica zadana za okolinu od 25°C, a t_o okolna temperatura, korišten je isti oblik parabole kao i kod $g(t_o)$ funkcije, a namješten je primjenom funkcije $h(t_z)$ koja pomiče funkciju $g(25)$ po t_l ili t_{iz} osi (t_l je konačna, primijenjena temperaturna granica, a t_{iz} je prethodno spominjana izlazna temperatura) tako da t_z i $f(t_z,25)$ budu jednaki.

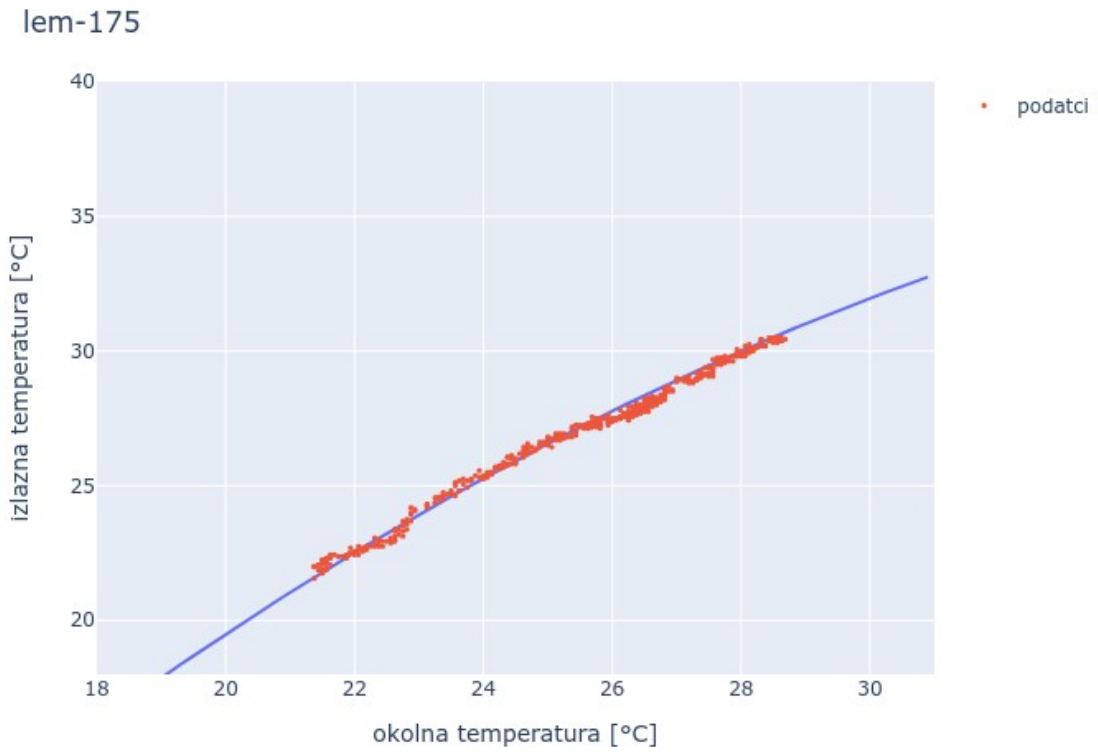
$$h(t_z) = t_z - g(25) = t_z - 28{,}5914 \tag{7-13}$$

$$t_l = f(t_z,t_o) = g(t_o) + h(t_z) \tag{7-14}$$

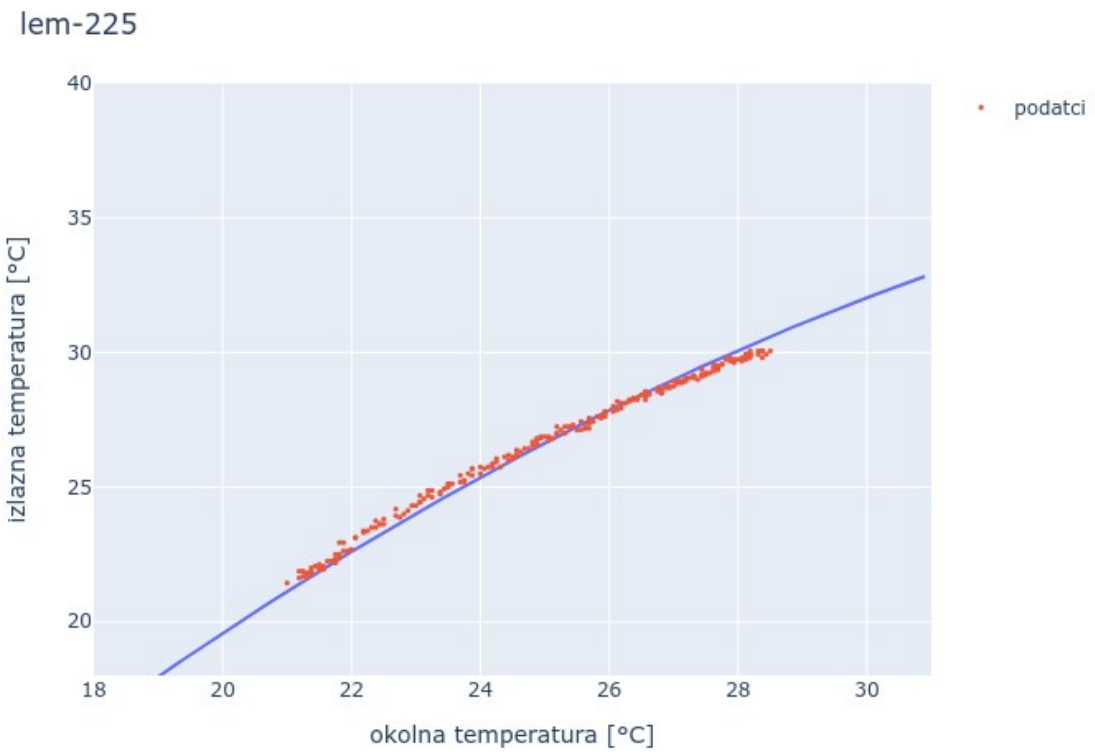
$$t_l = f(t_z,t_o) = -0{,}0334\,t_o^2 + 2{,}9169\,t_o - 23{,}4561 + h(t_z) \tag{7-15}$$

Prikaz funkcije $f(t_z,t_o)$ uz empirijske podatke dan je na slikama 7.20. - 7.22. uz sljedeće vrijednosti t_z:

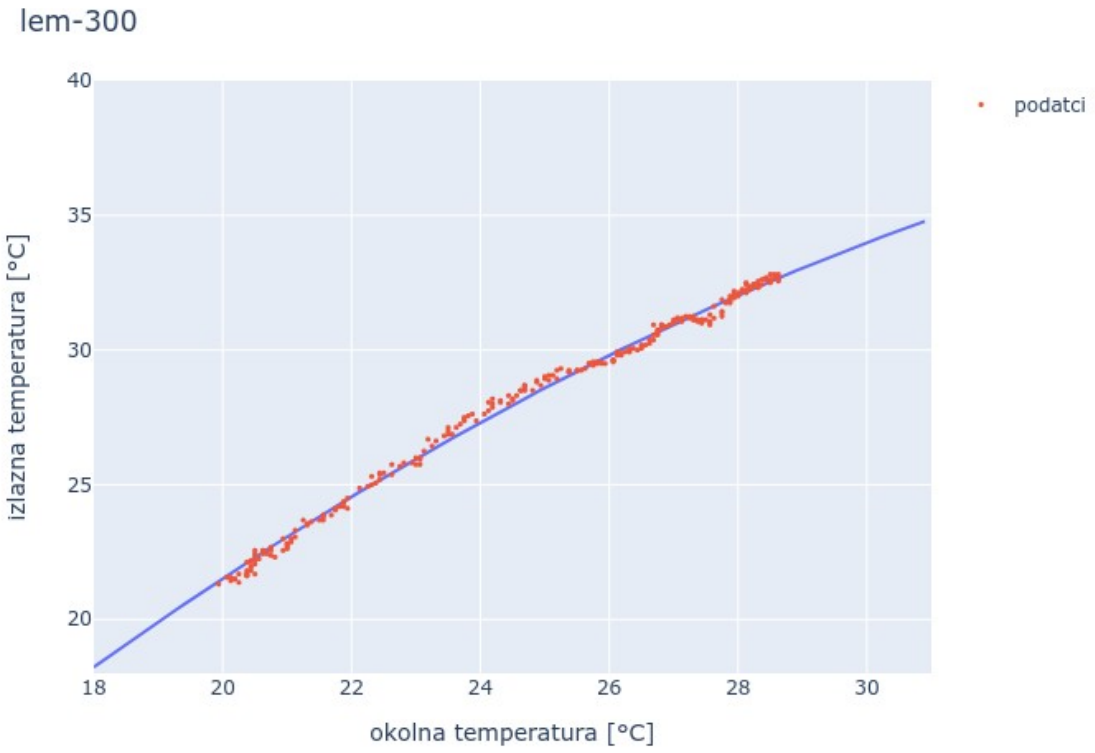
$$t_z = g_{vlastita}(25) \tag{7-16}$$



Slika 7.20. $f(t_z,t_o)$ uz kontrolnu varijablu 175°C



Slika 7.21. $f(t_z,t_o)$ uz kontrolnu varijablu 225°C



Slika 7.22. $f(t_z,t_o)$ uz kontrolnu varijablu 300°C

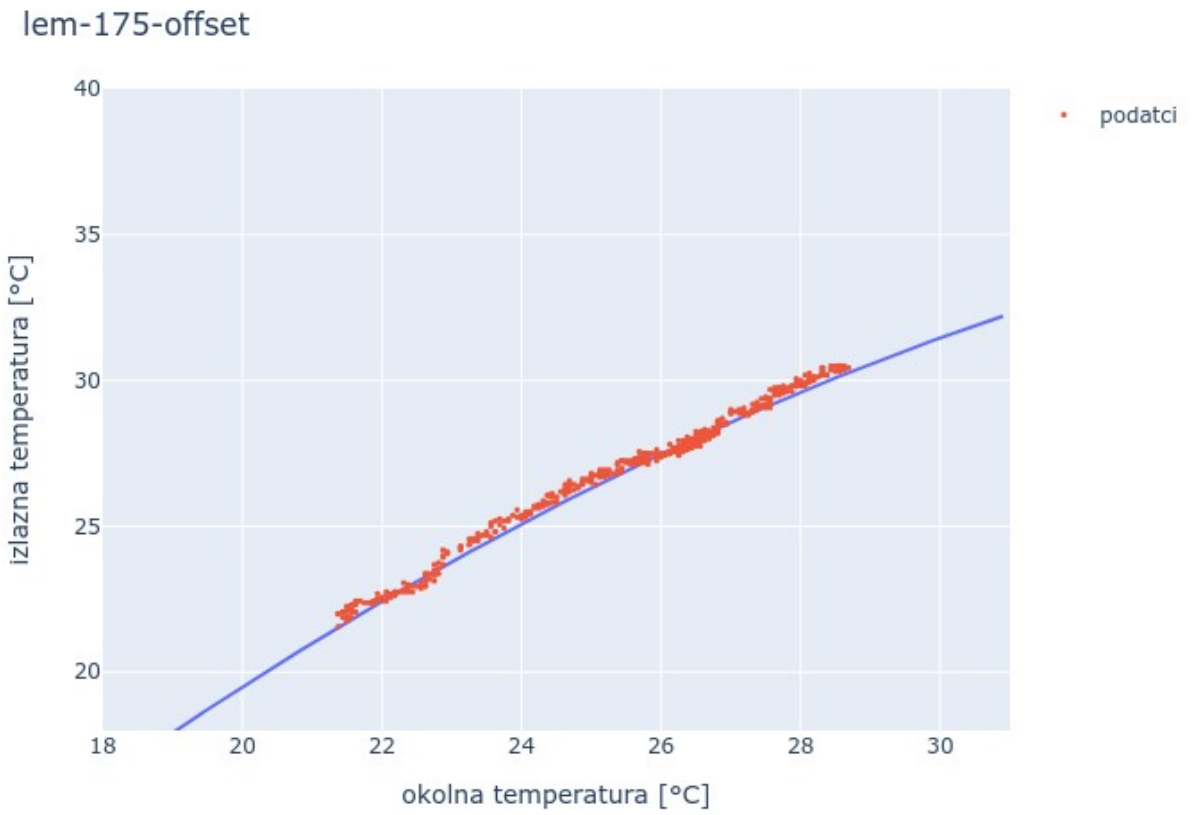
No, za kontrolnu varijablu 225°C, primijećeno je kako pri višim temperaturama funkcija $f(t_z,t_o)$ može predvidjeti previsoke t_l vrijednosti, što riskira pregrijavanje. Zato je definirana nova funkcija $f_{sigurna}(t_z,t_o)$ koja implementira linearno smanjenje izlazne vrijednosti: 0°C za okolinu od 20°C, a 0,4°C za okolinu od 28°C.

$$k(t_o)=0.4-0.4\frac{(28-t_o)}{8} \tag{7-17}$$

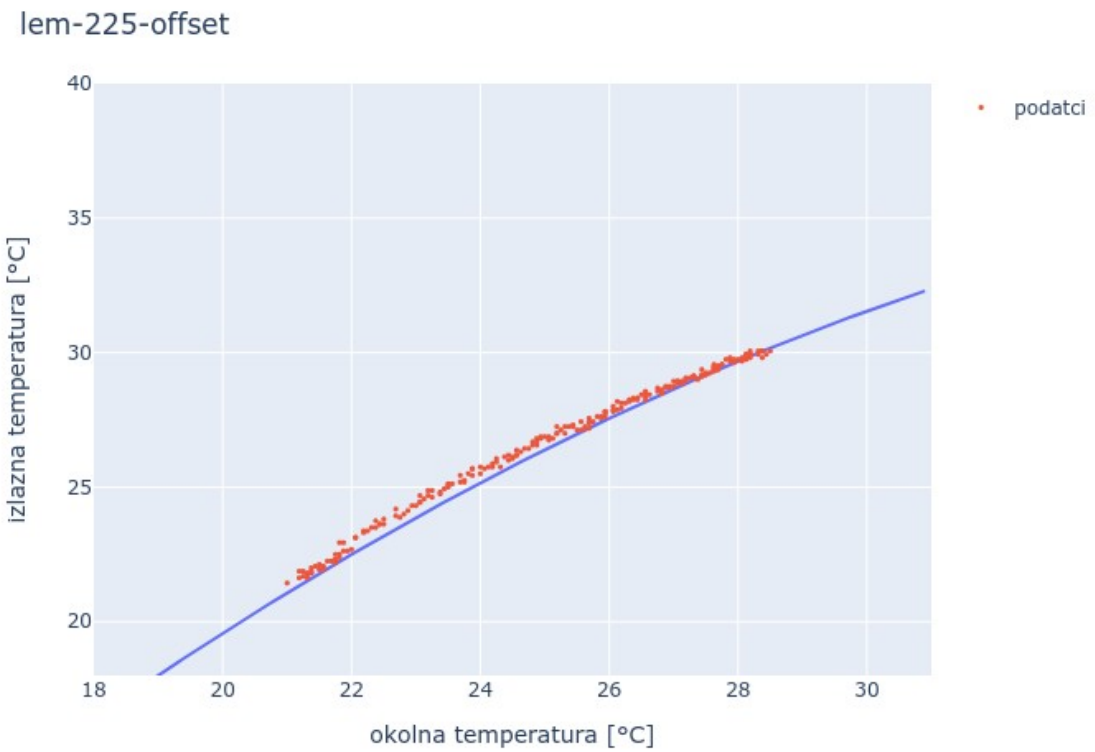
$$f_{sigurna}(t_z,t_o)=g(t_o)+h(t_z)-k(t_o) \tag{7-18}$$

$$f_{sigurna}(t_z,t_o)=f(t_z,t_o)-k(t_o) \tag{7-19}$$

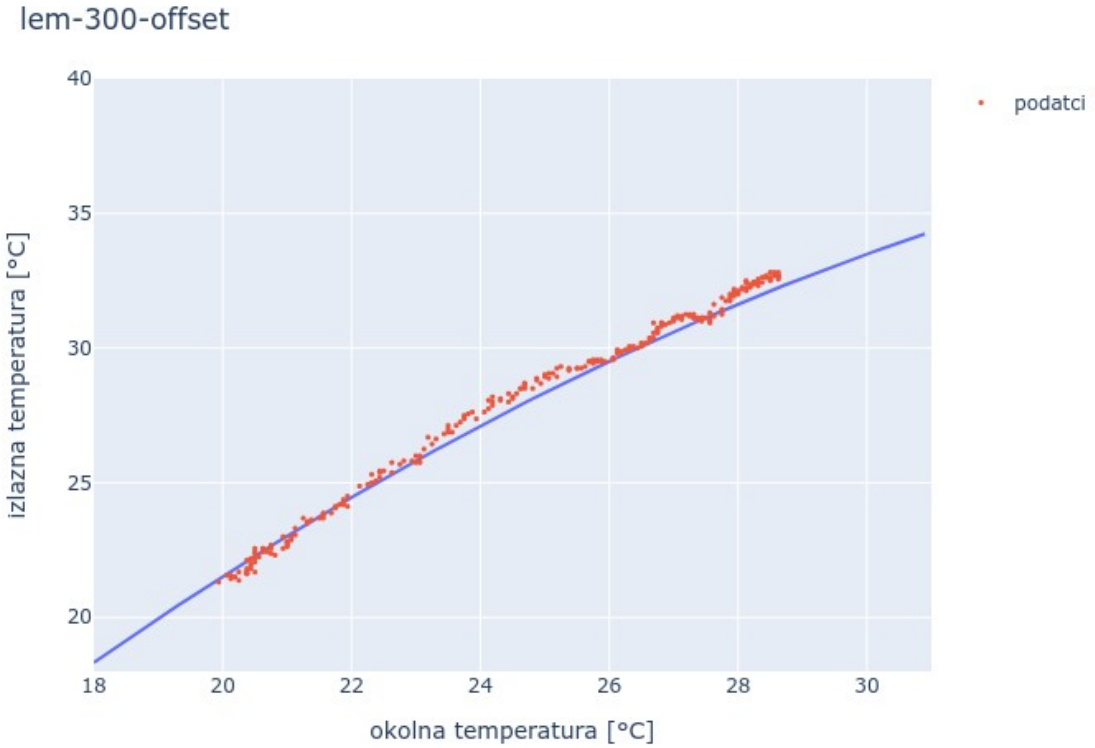
Na slikama 7.23. - 7.25. su usporedbe $f_{sigurna}(t_z,t_o)$ s prvotnim empirijskim podatcima.



Slika 7.23. $f_{sigurna}(t_z,t_o)$ uz kontrolnu varijablu 175°C



Slika 7.24. $f_{sigurna}(t_z,t_o)$ uz kontrolnu varijablu 225°C



Slika 7.25. $f_{sigurna}(t_z,t_o)$ uz kontrolnu varijablu 300°C

7.2.3. Zaključak eksperimenta

Zakonitost između okolne i izlazne temperature bilo je moguće aproksimirati kvadratnom parabolom $g(t_o)$ te ju je na isti način moguće aproksimirati za realan sustav. Ponavljanje eksperimenta s različitim temperaturama lažnog testnog objekta dovelo je do različitih aproksimacija parabole pa je tako bilo moguće odabrati aproksimaciju $g(t_o)$ koja djeluje najsigurnije u (hipotetskoj) praksi. Iz te aproksimacije, uz pomoć funkcije $h(t_z)$ moguće je dobiti funkciju $f(t_z,t_o)$ koja će regulirati konačnom graničnom temperaturom izlaznog zraka t_l. Moguće joj je dodati funkciju $k(t_o)$ koja će ju učiniti još sigurnijom funkcijom, $f_{sigurna}(t_z,t_o)$.

Konačna formula zaključena ovim eksperimentom glasi:

$$f_{sigurna}(t_z,t_o) = -0{,}0334\,t_o^2 + 2{,}9169\,t_o - 23{,}4561 \; + \; (t_z - 28{,}5914) \; - \; \left(0.4 - 0.4\,\frac{(28 - t_o)}{8}\right) \quad \text{(7-20)}$$

Formula je implementirana u programski kôd te je konačna verzija programa, s regulacijom temperature t_i, implementirana u sljedećim eksperimentima. U praksi se ne preporučuje korištenje te verzije programa i navedene formule izvan intervala okolne temperature t_o [22, 28], a zbog odabira funkcije $k(t_o)$ je kritični interval [20, 30]. Navedeni intervali nisu bitni za testove ispravnosti, osim ako su izvedeni u realnoj situaciji.

7.3. Eksperiment 2 - Određivanje granične temperature za 25°C

7.3.1. Organizacija eksperimenta

Za primjenu sustava za određeni testni objekt, korisnik mora odrediti graničnu izlaznu temperaturu za okolnu temperaturu od 25°C. Iz toga razloga će to biti demonstrirano i u ovom radu. Preporučuje se sljedeći postupak:

1. Ukoliko je upravljačka jedinica već spojena na način da može prekinuti napajanje testnog objekta, osigurati da u *boards.txt* datoteci postoji konfiguracija npr. 999°C.
2. Podesiti klima uređaj za održavanje sobne temperature oko 25°C i uključiti ga.
3. Uključiti upravljački uređaj kako bi se upalio ventilator.
4. Pričekati da se okolna temperatura stabilizira oko 25°C.
5. Preopteretiti procesor na ploči kako bi završio u granici pregrijavanja.
6. Bilježiti izlaznu i okolnu temperaturu dok se izlazna temperatura ne stabilizira, i jedno vrijeme nakon toga.
7. Završiti proces i pustiti procesor da se ohladi.

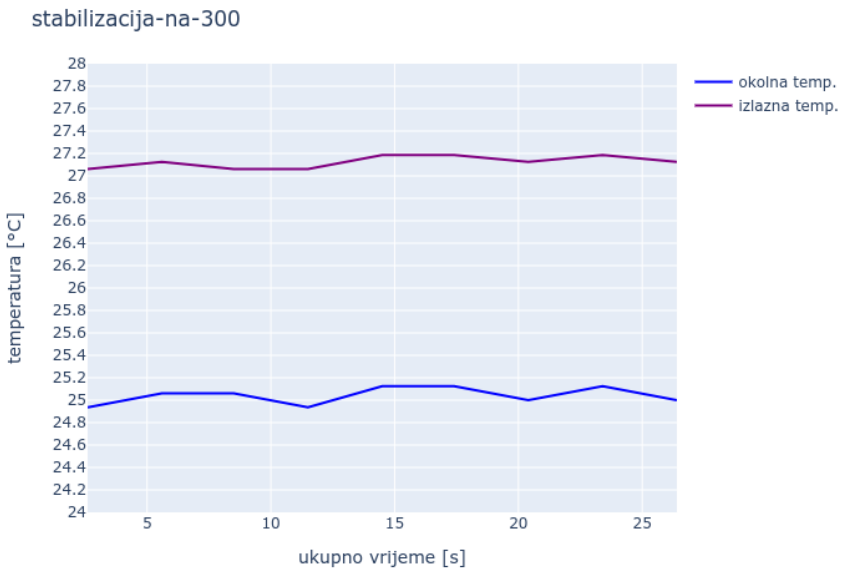
Eksperiment bi trebao biti ponovljen za svaki procesor. Serija podataka koja zaključuje najmanju vrijednost granične temperature (iz sigurnosnih razloga) zapisuje se u *boards.txt* datoteku kao *ime_ploče[tab]temperatura[novi_red]*.

No u ovome slučaju raspolažemo lemilicom umjesto pravim testnim objektom pa je **prilagođena varijanta tog postupka** lakša za izvođenje. Naime, lemilica, za razliku od procesora, ne može pregorjeti na temperaturi na koju je bila podešena pa se nije moralo žuriti. Slijedi postupak koji je bio primijenjen:

1. Podesiti lemilicu u malo drugačiju poziciju nego u eksperimentu 1 (kako bi se dobila drugačija granična temperatura za okolinu 25°C nego u eksperimentu 1).
2. **Pretpostaviti** da temperatura lemilice ne smije doseći 300°C.
3. Podesiti klima uređaj za održavanje sobne temperature na 25°C i uključiti ga.
4. Uključiti upravljački uređaj kako bi se upalio ventilator.
5. Uključiti lemilicu i zadati joj, preko sonde, temperaturu 300°C.
6. Pričekati da se okolna temperatura stabilizira oko 25°C.
7. Bilježiti izlaznu i okolnu temperaturu dok se izlazna temperatura ne stabilizira (ukoliko se još nije u koraku 6), i jedno vrijeme nakon tog.

Koristila se verzija programa napisana nakon eksperimenta 1, tj. ona koja pravilno izračunava konačnu graničnu temperaturu, ali to nije bilo bitno jer je signal za napajanje spojen na LED diodu radi lakše demonstracije (kao i u ostatku rada). Bilo je bitno uključiti logiranje. Skripta *expparse.py* parsira log datoteku, a *expstab.py* za generira grafički prikaz.

7.3.2. Analiza eksperimenta



Slika 7.26. Vremenski odziv okolne i izlazne temperature nakon stabilizacije

Iz slike 7.26. uočavamo da je izlazna temperatura (za malo drugačiju poziciju lemilice) oko 27,1°C. Zbog manjka točnosti očitavanja temperature, moguće je u *boards.txt* zapisati temperaturu između 27.0°C i 27.2°C.

7.3.3. Zaključak eksperimenta

Moguće je eksperimentalno odrediti graničnu izlazni temperaturu t_z pri okolnoj temperaturi t_o od 25°C za svaki pojedini testni objekt ili lažni testni objekt, koja u ovom slučaju može biti u intervalu [27.0°C, 27,2°C]. Preporučuje se iščitavanje granične temperature preko prikladno podešenog grafičkog prikaza (da su bitne znamenke vidljive i mogu se iščitati).

7.4. Eksperiment 3 - Određivanje idealnog vremena odmora

7.4.1. Organizacija eksperimenta

U ovom eksperimentu pokušava se odrediti vrijeme odmora za koje vrijedi sljedeće:

- Slučajeva kratkih napajanja između stanja odmora treba biti što manje
- Slučajeva ponavljanja stanja odmora treba biti što manje
- Ukupno vrijeme napajanja treba biti što veće

Pod pojmom „kratko napajanje između stanja odmora" podrazumijeva se stanje napajanja koje, od svog početka, traje 30 sekundi ili manje. Upravljački program ispisuje i logira trenutno vrijeme izvršavanja upravljačke sekvence od njenog početka (*total uptime*) i vrijeme koje je sekvenca provela u stanju napajanja (*test uptime*). Postava je ista kao i u eksperimentu 2. Za temperaturu okoline 25°C korištena je granična temperatura 27,2°C.

Program je pokrenut naredbom *sh start_controller_safe.sh vrijeme_odmora* i eksperiment je ponovljen za sljedeća vremena odmora: 60, 90, 120, 180, 210, 240 (sekundi). Korištena je verzija programa izgrađena nakon uspješnog završetka eksperimenta 1. Sve promjene programa su se ticale isključivo popravaka malih problema koji ne utječu na regulacijska svojstva niti na ispis tijekom ovog eksperimenta. Logiranje je uključeno. Nakon svake serije mjerenja je log datoteka bila preimenovana kako bi bila odvojeno parsirana skriptom *expparse.py*, a grafički prikazi su bili generirani skriptom *exp3graph.py*.

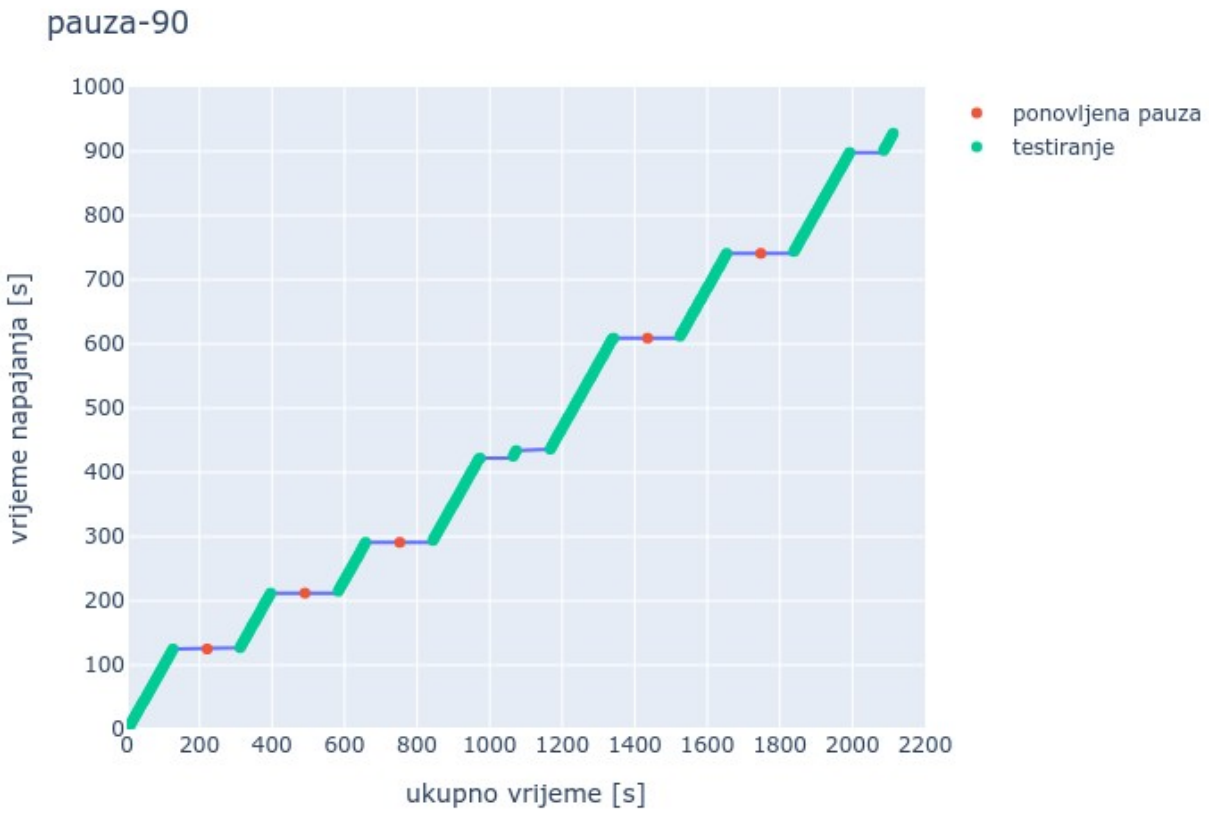
Koraci izvođenja bili su sljedeći:

1. Osigurati da je temperatura okoline između 24,5°C i 27°C.
2. Osigurati da je upravljačka jedinica uključena (kako bi ventilator bio uključen).

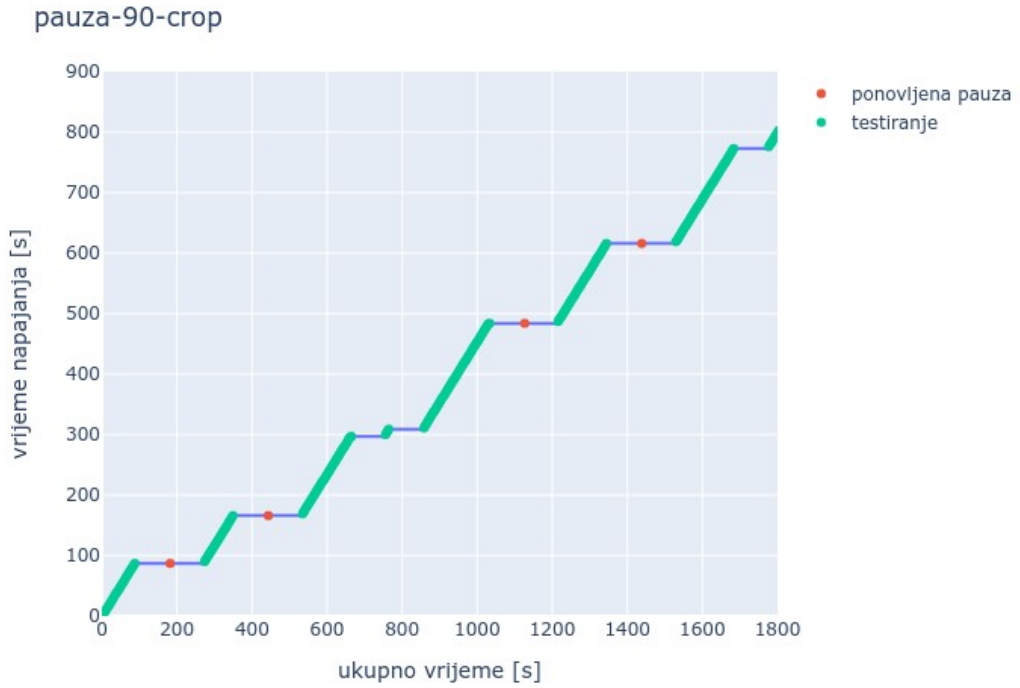
3. Podesiti lemilicu na 300°C preko sonde.
4. Pokrenuti upravljački program, omogućiti logiranje i odabrati pripadnu konfiguraciju iz *boards.txt* te pokrenuti upravljačku sekvencu.
5. Pričekati prvo stanje odmora pa isključiti lemilicu.
6. Nakon uspješnog povratka u stanje napajanja uključiti lemilicu.
7. Sljedećih 30 minuta isključivati lemilicu kad ona prijeđe u stanje odmora, a uključivati ju kad prijeđe u stanje napajanja.

Ukoliko je, pri kraju izvršavanja eksperimenta, upravljačka sekvenca u stanju odmora i očito je da će program biti prekinut tijekom tog stanja odmora, program je moguće ranije prekinuti. Stanje odmora indicirano je izvješćem (u konzoli i log datoteci) *...starting pause...* i *...taking a break...* te stanje napajanja izvješćem *...moving on...* Analiza se vrši na temelju podataka o vremenu izvršavanja sekvence i vremenu napajanja.

Za analizu su uzeti samo podatci prikupljeni tijekom koraka 7. Sva vremena izvršavanja sekvence i napajanja su smanjena za one zabilježene u koraku 6. Na slici 7.27. je primjer neobrađenih podataka (od početka upravljačke sekvence), a na slici 7.28. je primijenjena prethodno opisana operacija kako bi bitne informacije bile lakše prikazane, za istu seriju podataka.



Slika 7.27. Neobrađeni podatci



Slika 7.28. Obrađeni podatci (korak 7)

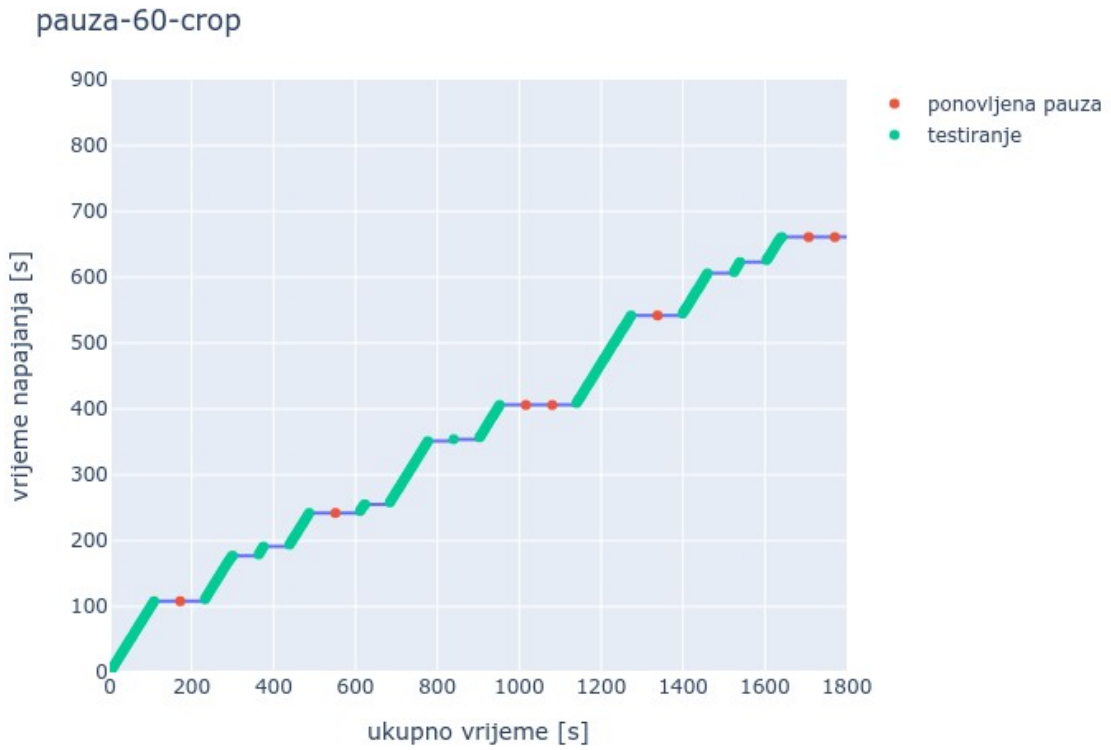
7.4.2. Analiza eksperimenta

Prema preimenovanim log datotekama te slikama 7.29. - 7.34. napisana je tablica 7.2. Može se primijetiti kako mjerenja za 240s završavaju ranije. Naime, pred kraj te serije mjerenja sustav je bio u stanju odmora te bi isto tako bio u stanju odmora planiranim krajem serije mjerenja.

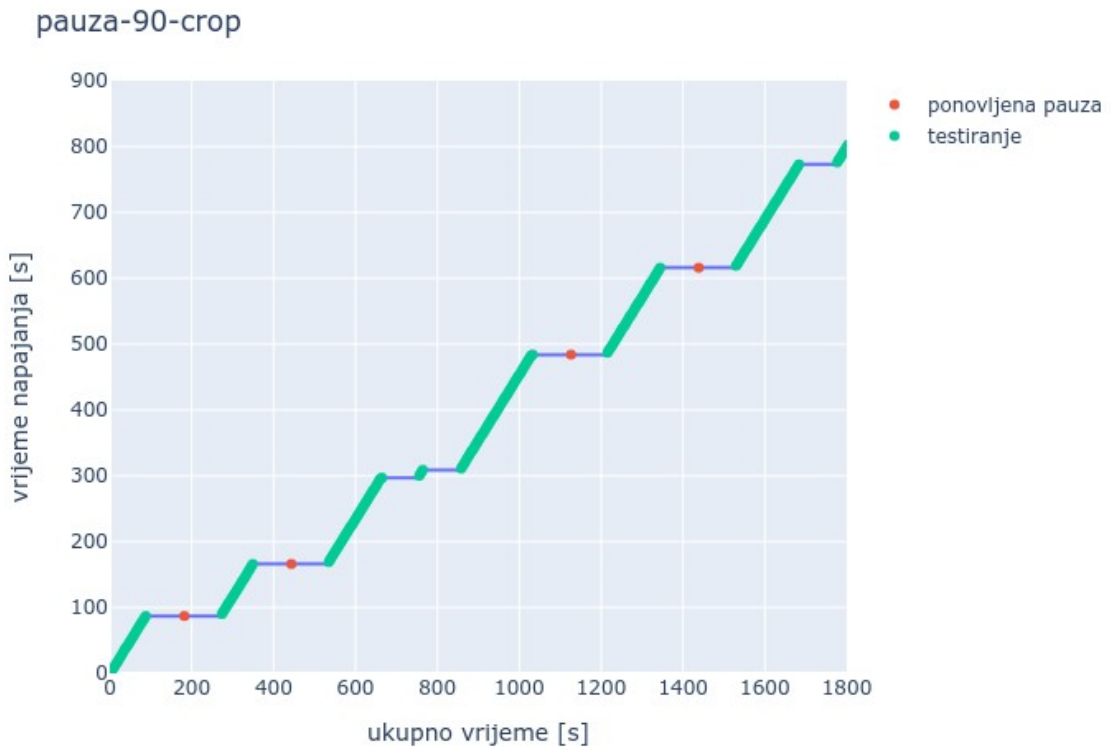
Za vrijeme odmora od 90s je vrijeme napajanja tijekom koraka 7 najveće (800s), ali su dobivena 4 slučaja ponavljanja odmora, što znači da bi u stvarnoj situaciji vrijeme napajanja moglo biti nepredvidivo. Dodatno, u jednom trenutku je napajanje bilo prekratko uključeno.

Za vrijeme odmora od 210s nisu uočene nikakve mane tijekom 30 minuta, a vrijeme napajanja tijekom koraka 7 (774s) je samo malo manje od onog u prethodnom odlomku. Za ukupni sustav koji je korišten u ovom eksperimentu, preporučeno vrijeme odmora bilo bi 210s. Prekratko napajanje nije uzeto u obzir pred kraj mjerenja.

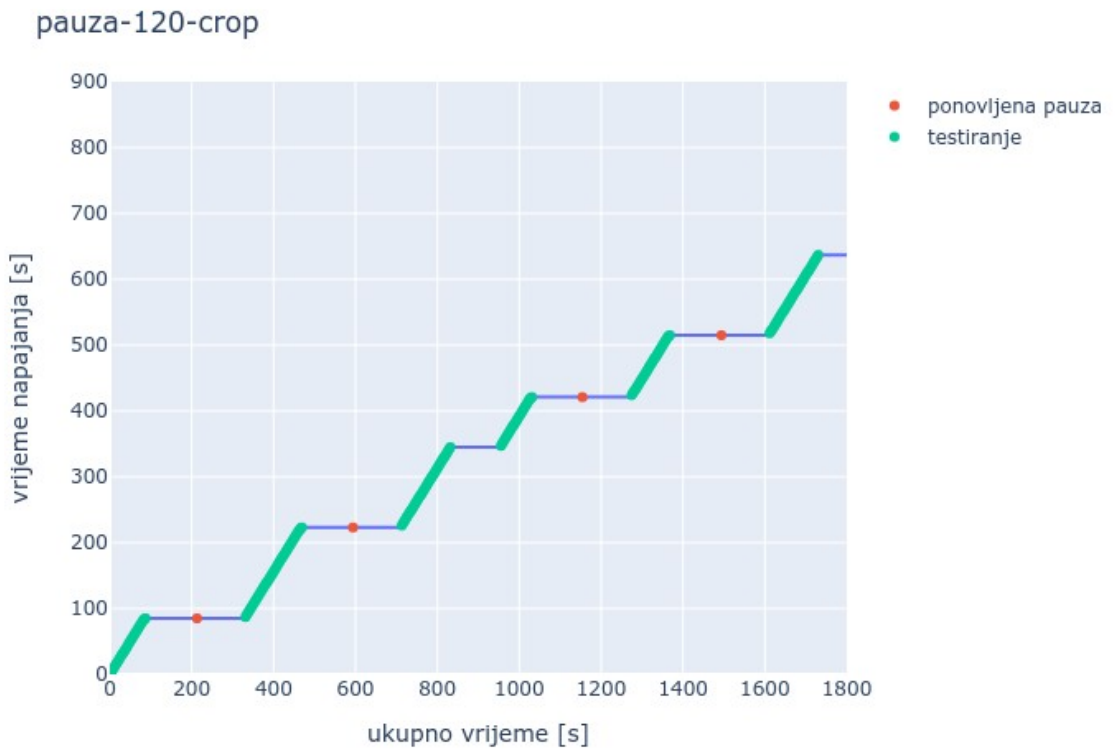
Vrijeme odmora	Početak koraka 7	Početno v. napajanja	Kraj koraka 7	Krajnje v. napajanja	**Bitno v. napajanja**	**Prekratka napajanja**	**Ponavlj. odmora**
60	62,5 s	0 s	1862,7 s	661 s	**661 s**	**4**	**7**
90	309,0 s	125 s	2019,4 s	925 s	**800 s**	**1**	**4**
120	296,4 s	51 s	2097,4 s	688 s	**637 s**	**0**	**4**
180	268,3 s	83 s	2068,2 s	639 s	**556 s**	**1**	**1**
210	320,1 s	106 s	2119,6 s	850 s	**744 s**	**0**	**0**
240	350,2 s	107 s	2102,2 s	735 s	**628 s**	**0**	**0**



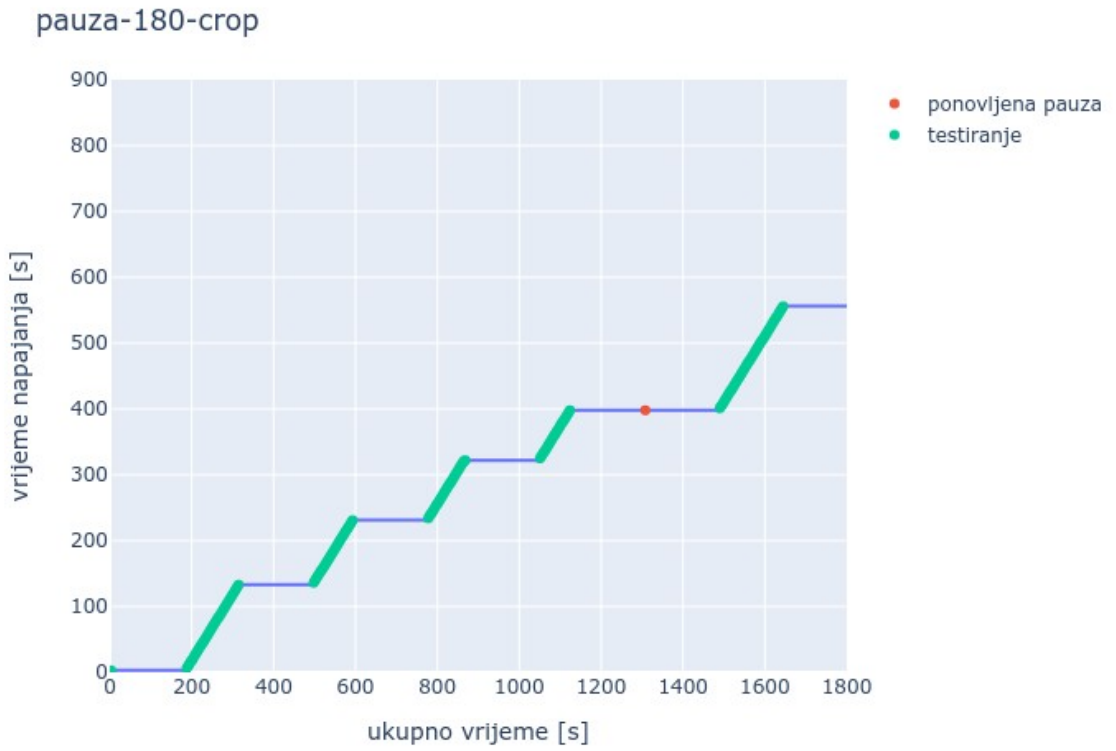
Slika 7.29. Ovisnost vremena napajanja o vremenu izvršavanja za 60s vremena odmora



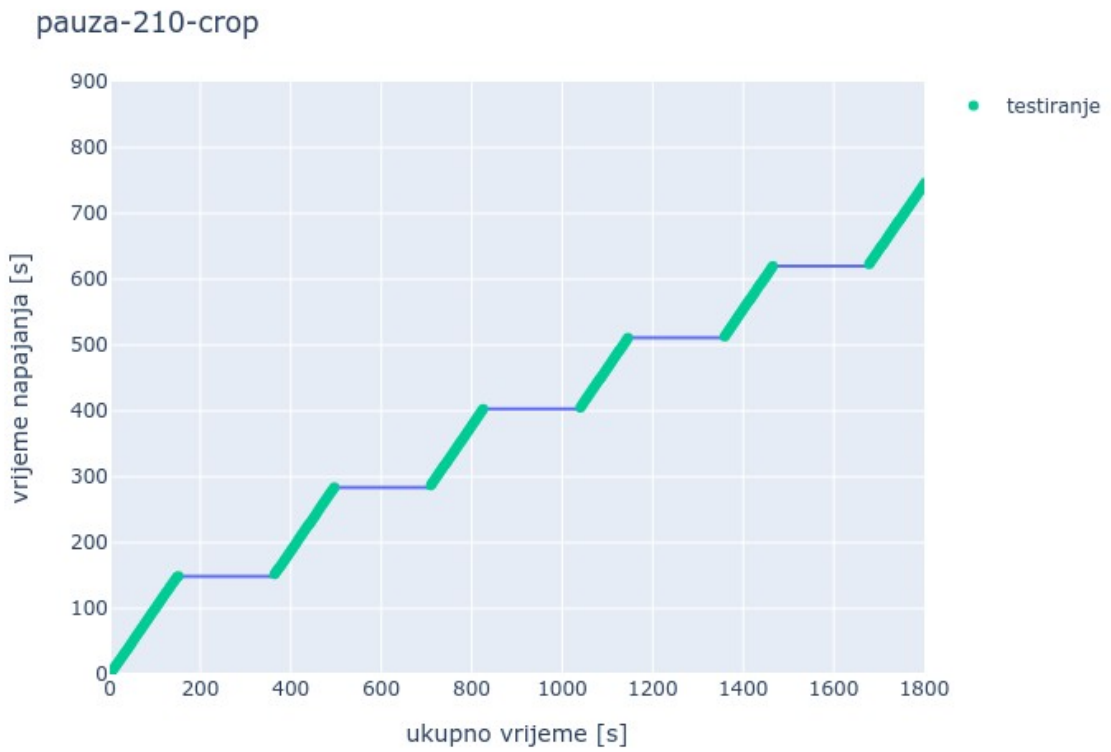
Slika 7.30. Ovisnost vremena napajanja o vremenu izvršavanja za 90s vremena odmora



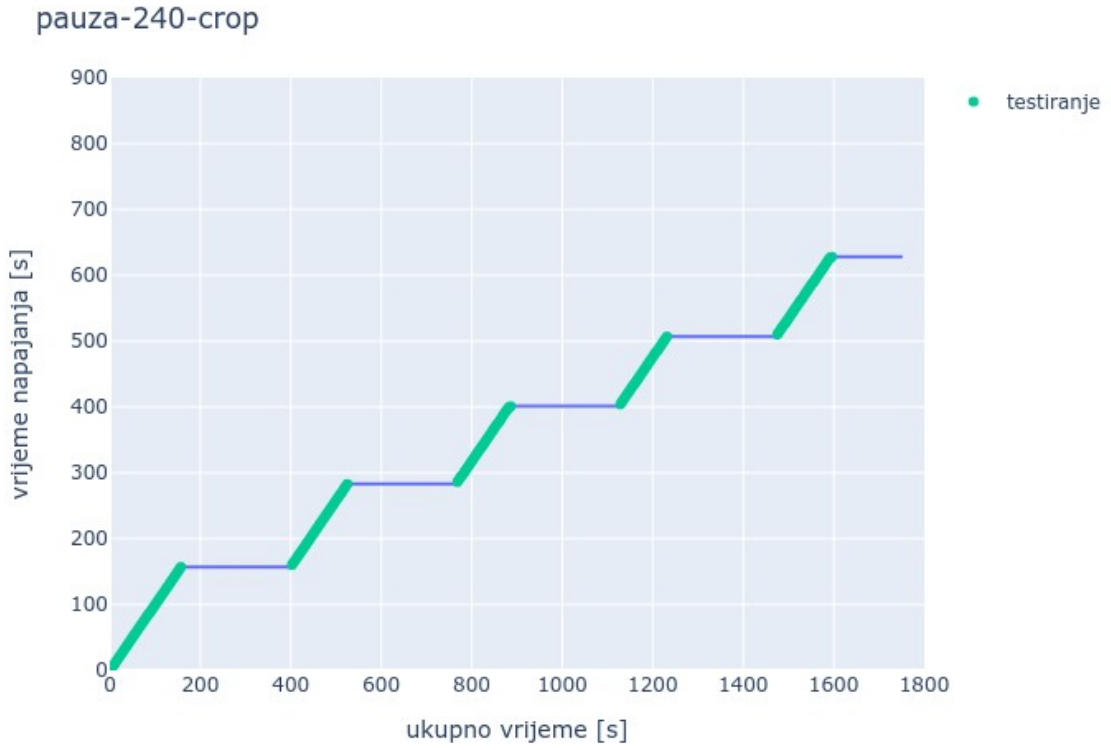
Slika 7.31. Ovisnost vremena napajanja o vremenu izvršavanja za 120s vremena odmora



Slika 7.32. Ovisnost vremena napajanja o vremenu izvršavanja za 180s vremena odmora



Slika 7.33. Ovisnost vremena napajanja o vremenu izvršavanja za 210s vremena odmora



Slika 7.34. Ovisnost vremena napajanja o vremenu izvršavanja za 240s vremena odmora

7.4.3. Zaključak eksperimenta

Kada vrijeme napajanja između stanja odmora treba zadovoljiti određeni minimum, simulacijom u realnim uvjetima moguće je odrediti najbolje vrijeme odmora za navedeni sustav, tj. vrijeme odmora koje će vrlo rijetko ili nikad rezultirati kršenjem tog uvjeta, a da ima što veće ukupno vrijeme napajanja i što manje ponovljenih stanja odmora.

Za ovaj sustav bi preporučeno vrijeme odmora bilo 210 sekundi (3 i pol minute). Pretpostavi li se da je uvjet potpuno kritičan, moguće je odabrati vrijeme odmora od 240 sekundi uz žrtvovanje ukupnog vremena napajanja. Prema ovom istraživanju može se očekivati da bi, u realnoj situaciji, ono bilo između 3 i 4 minute.

8. ZAKLJUČAK

U radu je opisan prototip sustava za temperaturnu regulaciju tiskane pločice tijekom *needle testing* procesa. Upravljanje je postignuto uključivanjem i isključivanjem glavnog napajanja koristeći *Raspberry Pi 3 B* uređaj kao upravljački uređaj. Temperaturni senzori koji su spojeni na *Raspberry Pi 3 B* i koji implementiraju *1-Wire* komunikacijski protokol su korišteni za detekciju pregrijavanja, a ventilator s tahometrom za neprekidno hlađenje sustava. Upravljački program na *Raspberry Pi 3 B* je osposobljen za detekciju neispravnosti temperaturnih senzora i ventilatora. U slučaju detekcije pregrijavanja, *Raspberry Pi 3 B* isključuje napajanje testnog objekta na predefinirano vrijeme (stanje odmora), a ukoliko je nakon toga pregrijavanje i dalje detektirano, napajanje testnog objekta ponovno se isključuje na to isto predefinirano vrijeme. U slučaju detekcije neispravnosti senzora ili ventilatora, glavno napajanje se isključuje. Uključenost glavnog napajanja simulirana je LED diodom.

Kako bi program osposobljen te kako bi se zaključile neke od preporuka za njegovu primjenu, napravljena su tri eksperimenta za koje je korišten lažni testni objekt – lemilica u kućištu. Senzori su pričvršćeni probušenim plastičnim čašama. Eksperimenti su opisani u zadnjem poglavlju prije zaključka, poglavlju 7, radi čitljivosti.

Prvim eksperimentom dobivena je funkcija 7-20 za određivanje vrijednosti izlazne temperature pri kojoj bi program trebao detektirati pregrijavanje. Prvo su određene funkcije ovisnosti temperature izlaznog zraka o temperaturi okolnog zraka za različite temperature lemilice. Funkcija 7-20 je određena pomoću najbolje funkcije prethodno spomenute ovisnosti, 7-12, funkcije pomicanja po osi izlazne temperature u ovisnosti o zadanoj temperaturi pregrijavanja za okolinu od 25°C 7-13 te funkcije za povećanje sigurnosti 7-17. Ona je primijenjena za izrađivanje konačne verzije upravljačkog programa na *Raspberry Pi* uređaju. Isti eksperiment i istu analizu moguće je napraviti i za stvarni testni objekt.

Nadalje, napravljen je eksperiment određivanja prethodno spomenute temperature pregrijavanja za okolinu od 25°C, kako bi se sličan proces demonstrirao korisniku te kako bi treći eksperiment mogao biti izvršen. Zatim je napravljen treći eksperiment, gdje je simuliran lažni testni objekt u pogonu i kojim se, za njega, tražilo najbolje vrijeme odmora. Što je manja učestalost vremena napajanja između stanja odmora kraćeg od 30 sekundi, što je manja učestalost ponavljanja detekcije pregrijavanja i što je veće ukupno vrijeme napajanja, to se zadano vrijeme odmora više cijeni. Isti eksperiment moguće je napraviti i za stvarni testni objekt.

U fazi testiranja ispravnosti u poglavlju 6, koja koristi konačnu verziju programa, testirala se reakcija programa na *Raspberry Pi* uređaju na promjenu okolne temperature, fizički simulirano

pregrijavanje, neispravnost senzora i neispravnost ventilatora. Program je reagirao u skladu sa zahtjevima i navedenim očekivanjima.

Daljnjim istraživanjem moguće je odrediti koji su senzori bolji i koji gori za primjenu u navedenoj situaciji. Nadalje, moguće je istražiti kako dinamička svojstva temperaturnih senzora utječu na upravljačka svojstva ovakvog sustava te kako se mijenja zakrivljenost funkcije za detekciju pregrijavanja, u ovisnosti o temperaturi procesora i grijane površine. Zatim je moguće ponoviti treći eksperiment ili istražiti poboljšanja algoritama određivanja vremena za koje je napajanje isključeno u slučaju detekcije pregrijavanja.

LITERATURA

[1] Hackaday, Test PCBs On A Bed Of Nails
https://hackaday.com/2019/02/09/test-pcbs-on-a-bed-of-nails/
(pristup ostvaren 2.12.2019.)

[2] Dealna, Types of Computer Cooling Systems: Which to Choose?
https://dealna.com/en/Article/Post/722/Types-of-Computer-Cooling-Systems-Which-to-Choose
(pristup ostvaren 1.7.2019.)

[3] Phoronix, Intel EIST SpeedStep
https://www.phoronix.com/scan.php?page=article&item=397&num=1
(pristup ostvaren 12.3.2020.)

[4] Phoronix, AMD Cool n Quiet
https://www.phoronix.com/scan.php?page=article&item=391&num=1
(pristup ostvaren 12.3.2020.)

[5] FERIT, E-kolegij: Linux u ugradbenim sustavima DA1-03
https://loomen.carnet.hr/course/view.php?id=7444
(pristup ostvaren 15.11.2019.)

[6] Thomas Petazzoni, Device Tree for Dummies, prezentacija
https://events.static.linuxfound.org/sites/events/files/slides/petazzoni-device-tree-dummies.pdf
(pristup ostvaren 14.7.2019.)

[7] Pico Technology Ltd., 1-Wire serial protocol decoding - PicoScope from A to Z
https://www.picotech.com/library/oscilloscopes/1-wire-serial-protocol-decoding
(pristup ostvaren 4.11.2019.)

[8] Electronics Tutorials, Pull-up Resistor and Pull-down Resistor Explained
https://www.electronics-tutorials.ws/logic/pull-up-resistor.html
(pristup ostvaren 14.7.2019.)

[9] AzoSensors, What is a Tachometer?
https://www.azosensors.com/article.aspx?ArticleID=310
(pristup ostvaren 23.7.2019.)

[10] HowToMechatronics, What is Hall Effect and How Hall Effect Sensors Work
https://howtomechatronics.com/how-it-works/electrical-engineering/hall-effect-hall-effect-sensors-work/
(pristup ostvaren 23.7.2019.)

[11] Hrvatska enciklopedija, Hallov efekt
http://www.enciklopedija.hr/natuknica.aspx?ID=70173
(pristup ostvaren 19.8.2019.)

[12] Electronics Tutorials, Hall Effect Sensor and How Magnets Make It Work
https://www.electronics-tutorials.ws/electromagnetism/hall-effect.html
(pristup ostvaren 23.7.2019.)

[13] Peter Vis, CPU Cooling Fan Tachometer Standard
https://www.petervis.com/electronics%20guides/cpu%20fan%20tacho/cpu%20fan%20tacho.html
(pristup ostvaren 23.7.2019.)
[14] Python API reference for plotly
https://plot.ly/python-api-reference/
(pristup ostvaren 13.2.2020.)

SAŽETAK

Rad opisuje implementaciju sustava održavanja temperaturne radne točke tiskane pločice tijekom *needle/nail* testiranja. Sustav je realiziran pomoću mjerenja temperature okolnog zraka i izlaznog zraka te aktivnoga hlađenja. Na temelju toga sustav određuje treba li testiranje nastaviti ili uzeti odmora. Mjerenje temperature je izvedeno senzorima koji koriste *1-Wire* protokol, hlađenje ventilatorom s tahometrom, a upravljačka platforma je *Raspberry Pi 3 B*. Program je realiziran na minimalističkoj *Linux* platformi i u radu su opisani testovi njegove ispravnosti. Eksperimentalnim postupkom određena je funkcija za detekciju pregrijavanja te se opisuju druga dva eksperimenta kao primjeri postupka za uvođenje sustava u pogon – eksperiment za određivanje granične izlazne temperature u okolini od 25°C te eksperiment za određivanje idealnog vremena za koje testni objekt mora biti ugašen u slučaju detekcije pregrijavanja.

Ključne riječi: Raspberry Pi 3 B, Linux, Device Tree, 1-Wire, aktivno hlađenje.

Design and realization of the system for maintaining the thermal working point of the printed circuit board with active cooling

ABSTRACT

This thesis describes an implementation of a system for working temperature point conservation of a PCB board during needle/nail testing. The system was designed through temperature measurements of the surrounding air, output air and active cooling. Based on that, the system decides should the testing continue or take a break. Temperature measurement is accomplished with temperature sensors that use 1-Wire protocol, the cooling using a fan with a tachometer, and the controller platform is Raspberry Pi 3 B. The program is implemented on a minimal Linux platform and the thesis includes testing of the correct behavior of the control software. Using an experimental procedure, function for overheating detection was determined and the other two experiments describe an example of the process of setting up the system for operation – an experiment for determining output temperature margin in 25°C environment and the experiment for determining the ideal time for which the test object should be shut down in case of overheating detection.

Key Words: Raspberry Pi 3 B, Linux, Device Tree, 1-Wire, active cooling.

ŽIVOTOPIS

Tibor Piskač rođen je 13.08.1994. u Zagrebu u Republici Hrvatskoj. Osnovnu školu Jure Kaštelana završio je 2009., a XV. gimnaziju u Zagrebu 2013. Godine 2013. upisuje Stručni studij računarstva na Tehničkom veleučilištu u Zagrebu, smjer Programskog inženjerstva, koji završava 2016. godine. Prije završetka studija (ljeto 2016.), radi za Corvus Info u Zagrebu na web projektu za e-Kupi.

Zbog interesa za ugradbenim sustavima i matematikom, upisuje i završava Razlikovne obveze na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku te godine 2017. tamo upisuje diplomski studij Računarstva, smjer Procesno računarstvo. Prima stipendiju Instituta RT-RK već na prvoj godini diplomskog studija, a tijekom ljeta 2018. radi, kao stipendist i programer, na odvojenom pomoćnom web projektu. Na drugoj godini radi na diplomskom radu zadanom od strane stipendista. Tijekom edukacije i prakse stekao je znanja i vještine za razvoj jednostavnih web aplikacija, a na diplomskom studiju i znanja iz automatizacije, robotike, strojnog učenja i dizajna ugradbenih sustava.

.....

Tibor Piskač

PRILOZI

Prilog P.6.1. datoteka „test limit vrijednost.txt”

Nalazi se na priloženim CD-u, u direktoriju „prilozi.”

Prilog P.6.2. datoteka „test pregrijavanja nakon.txt”

Nalazi se na priloženim CD-u, u direktoriju „prilozi.”

Prilog P.6.3. datoteka „test nedostupni nakon.txt”

Nalazi se na priloženim CD-u, u direktoriju „prilozi.”

Prilog P.6.4. početak datoteke „test ventilator, 2 potrebna.txt”

Nalazi se na priloženim CD-u, u direktoriju „prilozi.”

Prilog P.6.5. datoteka „test ventilator, 1 potreban”

Nalazi yse na priloženim CD-u, u direktoriju „prilozi.”