UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Sašo Frančeškin

Sistem za nadzor in upravljanje procesa vrenja v vinski cisterni

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2016

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Sašo Frančeškin

Sistem za nadzor in upravljanje procesa vrenja v vinski cisterni

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: doc. dr. Mojca Ciglarič

Ljubljana, 2016

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Opišite postopek alkoholne fermentacije v vinski cisterni. Pojasnite, katere težave nam lahko pomaga rešiti avtomatiziran nadzor dogajanja v cisterni. Predstavite osnovne koncepte interneta stvari in navežite na nadzor vinske cisterne. Zasnujte sistem, ki bo uporaben za ta namen, opišite in utemeljite izbiro tehnologije in orodij, pojasnite potreben način komunikacije. Pojasnite, kako poteka krmiljenje naprav ter izdelajte in opišite spletno aplikacijo za interakcijo z uporabnikom. Izdelek kritično ovrednotite in nakažite možnosti za nadaljnji razvoj.

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a Sašo Frančeškin,

z vpisno številko 63110029,

sem avtor/-ica diplomskega dela z naslovom:

Sistem za nadzor in upravljanje procesa vrenja v vinski cisterni

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)

  doc. dr. Mojca Ciglarič

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela

- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne _____ Podpis avtorja/-ice: _____

Ob tej priložnosti bi se rad zahvalil svoji mentorici doc. dr. Mojci Ciglarič, ki mi je s svojimi mnenji in nasveti pomagala pri izboljšavi mojega končnega izdelka ter mi pomagala odpraviti določene nerazumljivosti, ki so se pojavile med samo izdelavo. Obenem bi se rad zahvalil tudi vsem svojim prijateljem, ki so mi z različnimi idejami svetovali in pomagali pri izdelavi diplomske naloge. Na koncu bi se zahvalil tudi svojim staršem in sestri, ki so me ves čas trajanja izdelave diplomske naloge podpirali.

Kazalo

Kazalo slik

Seznam uporabljenih kratic

kratica	angleško	slovensko
IoT	Internet of Things	Internet stvari
FTP	File Transfer Protocol	Protokol za prenos datotek
RWD	Responsive Web Design	Odzivna spletna stran
HDMI	High-Definition Multimedia Interface	Multimedijski vmesnik visoke definicije
CO	Carbon mnoxide	Ogljikov monoksid
PHP	Personal Home Page	Orodja za osebno spletno stran
HTML	HyperText Markup Language	Jezik za označevanje nadbesedila
USB	Universal Serial Bus	Univerzalni podatkovni medij
SCL	Serial Clock Line	Serijski signalni tok
SDA	Serial Data Line	Serijski podatkovni tok
ACK	Acknowledgement	Potrditev
TCP	Transmission Control Protocol	Protokol za nadzor prenosa
IP	Internet Protocol	Internetni protokol
PPM	Parts Per Million	Število delcev na milijon
RFID	RadioFrequency IDentification	Radiofrekvenčna identifikacija
API	Application Programming Interface	Aplikacijski programski vmesniki
JSON	JavaScript Object Notation	JavaScriptni objektni zapis
I^2C / I2C	Inter-Integrated Circuit	Protokol komunikacije med integriranimi vezji

Povzetek

Internet stvari je povezano omrežje, ki ga sestavljajo pametne naprave, opremljene z različnimi senzorji. Diplomska naloga se ukvarja s sistemom za upravljanje vinske cisterne, s katerim lahko celoten proces alkoholne fermentacije, ki se dogajala v cisterni, nadziramo in krmilimo. Sistem je opremljen z različnimi senzorji in stikali.

Uporabniški vmesnik uporabniku omogoča upravljanje s celotnim sistemom in je edini modul, ki omogoča interakcijo med uporabnikom in sistemom. Sistem je razdeljen na tri glavne module. Poleg uporabniškega vmesnika sta tu še krmilna in strežniška aplikacija. Krmilna aplikacija je namenjena upravljanju senzorjev in treh sistemov, medtem ko strežniška aplikacija predstavlja vmesnik med uporabniškim in krmilnim modulom.

Diplomska naloga se ukvarja s konceptom Interneta stvari, ki predstavlja novo tehnološko evolucijo interneta in zagotavlja komunikacijo med objekti in ljudmi ter med objekti samimi. Diplomska naloga se osredotoča na problem nadzora alkoholne fermentacije v vinski cisterni. V celoti je opisan sistem za upravljanje vinske cisterne, ki omogoča spremljanje temperature in gladine mošta v vinski cisterni ter spremljanje koncentracije ogljikovega monoksida v vinski kleti. Sistem omogoča tudi upravljanje hladino-grelnega sistema in mešalnega sistema. Predstavljene so vse uporabljene naprave in senzorji ter vsa programska oprema, s katero je bil sistem razvit.

Ključne besede: Internet stvari, krmilni sistem, vinska cisterna, Arduino, Raspberry Pi.

Abstract

Title: The automatisation of the wine reservoir with Arduino and Raspberry Pi

The Internet of things is a connected network that consists of smart deviced, equipped with different sensors. The thesis is dealing with a system fort he management of a wine reservoir, with which the whole process of alcohol rementation that is going on in the reservoir can be supervised and operated. The system is equipped with different sensors and switches.

The user interface allows the user to manage the whole system and is the only module that allows the interaction between the user and the system. The system consists of three main modules. Apart from the user interface there are also the control and server applications. The control application is intended for the management of the sensors and the three systems, while the server application represents an interface between the user module and the control module.

The thesis focuses of the question of the control of alcohol fermentation in a wine reservoir. The system for the management of a wine reservoir that allows the supervision of the surface of must in a wine reservoir, and also the concentration of carbon monoxide in a wine cellar is described in its entirety. The system also allows the management of the cooling and heating system and the mixing system. The thesis describes the basic building blocks of the system for the management of a wine reservoir – the sensors and other devices, and the overviews of the programming tools and technologies used.

Key words: Internet of things, control / management system, wine reservoir, Arduino, Raspberry Pi.

Poglavje 1 Uvod

Fermentacija, ali po domače alkoholno vrenje mošta je dolgotrajen postopek, ki zahteva veliko časa in pozornosti. Predstavljajmo si, da smo vinogradniki in je to naš glavni vir dohodka. Za to dejavnost potrebujemo ogromne vinograde, kjer vino pridelujemo in tudi skladišče, kjer ga hranimo. Med samim procesom alkoholne fermentacije moramo biti zelo pozorni, da je temperatura mošta primerna, saj lahko med vretjem pride do nenadzorovanega dviga mošta in posledično do razlitja mošta. Da bi se temu izognili, bi morali vretje fizično nadzorovati štiriindvajset ur na dan, kar pa bi nam vzelo preveč časa [1].

Internet stvari predstavlja mrežo, v kateri so fizični objekti integrirani v informacijsko omrežje in so tako pomemben člen poslovnega procesa. Pametni objekti so povezani z internetom in omogočajo iterakcijo z objekti prek vmesnikov. Namen Interneta stvari je torej varna izmenjava podatkov med pametnimi objekti in ljudmi in med objekti samimi [23]. Glede na to, da živimo v digitalni dobi, bi lahko vinsko cisterno avtomatizirali in ustvarili »pametno« cisterno. Z mobilno napravo ali računalnikom bi stanje cisterne enostavno preverili ter po potrebi ponastavili določene nastavitve. Vse to pa bi lahko naredili kar z domače sedežne garniture ne, da bi sploh vstopili v klet ter fizično preverjali vrenje mošta.

V diplomski nalogi bomo tako poskušali najti primerno rešitev za uporabnika. Cilj diplomske naloge je ustvariti sistem, ki bo omogočal direktno povezavo s cisterno ter uporabniku dopuščal enostavno upravljanje te. Poleg tega bo sistem moral biti prilagodljiv tako, da bo ustrezal vsem vrstam cistern. S pomočjo različnih senzorjev bomo poskušali razbrati stanje mošta v cisterni. Vse podatke bomo hranili v podatkovnem skladišču, ki bo dostopno uporabniku. S pomočjo stikal bo uporabniku omogočeno tudi vključevanje in izključevanje različnih sistemov. Vse skupaj pa bo dostopno prek enostavnega uporabniška vmesnika.

Diplomsko nalogo sestavlja pet poglavij.

Drugo poglavje namenjeno Internetu stvari. Opisano je kaj sploh je Internet stvari ter sam čas in način razvoja tega. Na koncu so predstavljeni tudi možni komunikacijski modeli.

V tretjem poglavju so opisane vse naprave in senzorji, ki so bili uporabljeni za izdelavo prototipa ter tudi programska oprema, s katero je bil razvit celoten sistem za upravljanje.

Četrto poglavje je namenjeno načrtu in opisu izvedbe sistema; skupaj s slikami je jasno predstavljen celoten upravljalni in krmilni sistem. Opisana in predstavljena je tudi podatkovna baza, namenjena hranjenju podatkov.

Peto poglavje predstavlja zaključek. V njem je predstavljen končni izdelek, težave, ki so nastale med samo izdelavo ter reševanje teh. Govora je tudi o nadgradnji sistema za nadaljnje delo.

Poglavje 2 Internet stvari

Internet je veliko računalniško omrežje, ki je sestavljeno iz več manjših omrežij. Glavna lastnost interneta je ta, da omogoča komunikacijo in izmenjavo podatkov med uporabniki oziroma napravami [6]. Internet stvari predstavlja omrežje, v katerem so med seboj povezani vsakodnevni pametni objekti, ki omogočajo iterakcijo z drugimi objekti in ljudmi [24].

Vsak večer, preden zaspimo, nastavimo budilko, ki nas bo zjutraj zbudila. Zjutraj, ko nas budilka prebudi pa opazimo, da se je alarm oglasil pet minut kasneje od prvotno nastavljene ure. Naprava je namreč preverila vozni red avtobusa in ugotovila, da ima avtobus zamudo in nam na podlagi tega omogočila dodatnih pet minut spanca. Ko zapuščamo stanovanje opazimo, da ročaj na dežniku rdeče utripa. Dežnik je samodejno preveril vremensko napoved ter predvidel, da bo danes deževalo. To sta dva tipična primera Interneta stvari (v nadaljevanju IoT), kjer naprava komunicira z drugimi napravami, ki so povezane s svetovnim spletom [2].

Internet stvari predstavlja vizijo interneta, kjer so vse naprave povezane med sabo, kar pomeni, da so fizične naprave prosto dostopne in jih lahko upravljamo iz vseh koncev sveta, kjer je internetni dostop seveda omogočen. Ključnega pomena pa so ravno pametne naprave, saj z današnjim razvojem komunikacijske opreme ter informacijskih sistemov lahko resnično izkoristimo njihovo uporabnost. Z uporabo različnih senzorjev in naprav, ki so sposobne shraniti podatke, ter jih s pomočjo interneta posredovati drugim napravam ali uporabnikom, ustvarjamo skupno omrežje, v katerem vse naprave komunicirajo med sabo [7].

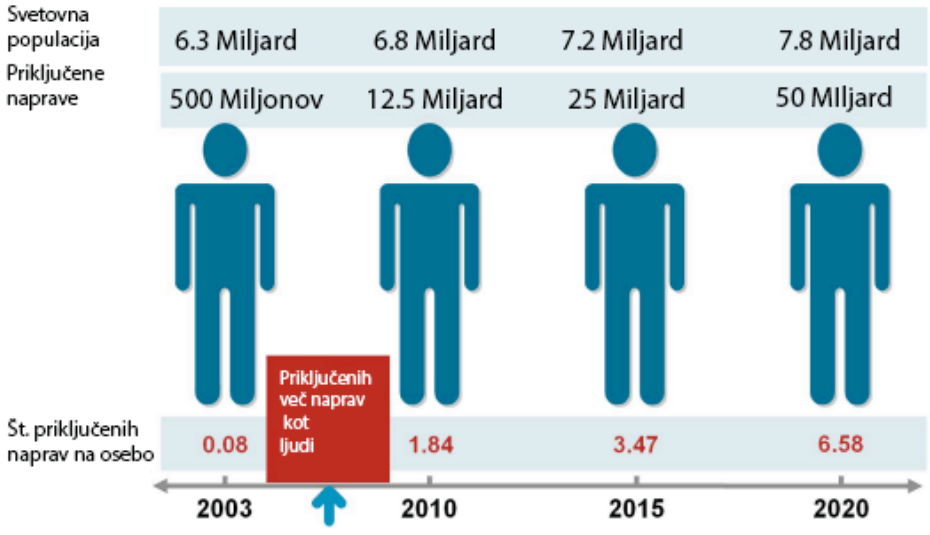
Izraz Internet stvari je star približno 16 let, tehnologijo pa je že davnega leta 1926 napovedal Nikola Tesla in dejal: »Ko bo brezžična komunikacija dodelana, bo celoten planet deloval kot veliki možgani, ki bo združeval predmete in naprave v celoto. Človek bo lahko svoj telefon nosil kar v svojem žepu« [4]. Leta 1999 pa je skovanko »Internet stvari« prvič uporabil Britanec Kevin Ashton, ki je sistem opisal kot omrežje računalnikov, ki so sposobni zaznavati in identificirati različne objekte. Štiri leta kasneje, pa je na EPC simpoziju predstavil nov koncept EPC omrežja, ki bi napravam omogočalo zaznavanje drugih objektov ter tako ustvarilo učinkovito omrežje IoT. K boljšemu in hitrejšemu razvoju Interneta stvari je zelo

pripomogel tudi razvoj mobilnih naprav, saj so mobilni telefoni postopoma postajali vse bolj zmogljivi. S procesno močjo in kapaciteto spomina ter senzorjev so mobilne naprave postale zmogljive aparature, ki so cenovno sprejemljive in dostopne vsem.

Podjetje Cisco, ki se ukvarja predvsem z razvojem in prodajo omrežne opreme je napovedalo, da bo do leta 2020 internetno dostopnih več kot 50 milijard naprav. Če pogledamo graf (Slika 2.1: Povprečno številno naprav na osebo skozi čas.), lahko vidimo, da je bilo leta 2003 na svetu približno 6,3 milijarde ljudi in 500 milijonov vseh naprav, ki so bile priključene na internet. Če ti števili delimo med sabo, dobimo vrednost 0,08, kar označuje število naprav na osebo (2.1).

$$število\ naprav\ na\ osebo = \frac{priključene\ naprave}{svetovna\ populacija} \tag{2.1}$$

Tu še ne moremo govoriti o pravem IoT, saj je število naprav bistveno prenizko. Vedeti moramo namreč, da je bil razvoj pametnih telefonov takrat šele na samem začetku. Prvi iPhone je podjetje Apple predstavilo šele leta 2007. Leta 2010 se je začela prava mobilna norija, ko so proizvajalci poleg mobilnih telefonov začeli prodajati tudi tablice. Iz Slike Slika 2.1 je jasno razvidno, da se je v obdobju 7 let število naprav bistveno povečalo. Danes lahko vidimo, da je skoraj vsak uporabnik lastnik več kot treh naprav [5].



Slika 2.1: Povprečno številno naprav na osebo skozi čas.

2.1 Funkcionalni vidik Interneta stvari

Internet stvari sestavljajo unikatno razpoznavni objekti. Da se objekti lahko »predstavijo« v internetni strukturi, pa morajo vsebovavi module kot so:

- modul za interakcijo z lokalni napravami: naloga tega modula je pridobivanje in posredovanje podatkov strežniku ter njihovo shranjevanje (nameščen je lahko na mobilni napravi, ki omogoča vzpostavitev povezave prek vmesnika);
- modul za lokalno analizo podatkov,
- modul za interakcijo z zunanjimi napravami: komunikacija lahko poteka prek interneta ali pa kar prek strežnika proxy, prav tako pa je zadolžen za pridobivanje in pošiljanje podatkov oddaljenemu strežniku,
- modul za obdelavo specifičnih podatkov: nameščen je na strežniški strani in je namenjen vsem uporabnikom, ki do njega dostopajo prek spletne ali mobilne aplikacije; v ozadju modula se izvajajo določeni algoritmi, ki uporabniku vrnejo ustrezne rezultate;
- uporabniški vmesnik: gre za modul, ki vizualno predstavi podatke uporabniku in predstavlja komunikacijski vmesnik med napravo in uporabnikom **Error! Reference source not found.**.

Z operativnega vidika je dobro tudi vedeti kako so naprave IoT povezane in kako komunicirajo med seboj. Obstajajo štirje komunikacijski modeli, ki se med seboj razlikujejo po ključnih karakteristikah, ki so opisane spodaj [4].

2.2 Komunikacijski model: naprava z napravo

Ta komunikacijski model predstavlja komunikacijo med dvema ali več napravami, ki so neposredno povezane med sabo brez posrednega medija (npr.: strežnika). Naprave lahko komunicirajo prek različnih protokolov, najpogosteje pa se uporablja protokol »modrega zoba« (angl. Bluetooth), Z-Wave ali ZigBee.

Slika 2.2: Skica komunikacijskega modela naprava z napravo.

Ta model se v večini primerov uporablja za bolj osebne namene, kjer naprave ne potrebujejo velike količine podatkov za upravljanje. Tipičen primer takih naprav so stikala, žarnice, termostati, ključavnice, ipd.[4].

2.3 Komunikacijski model: naprava z oblakom

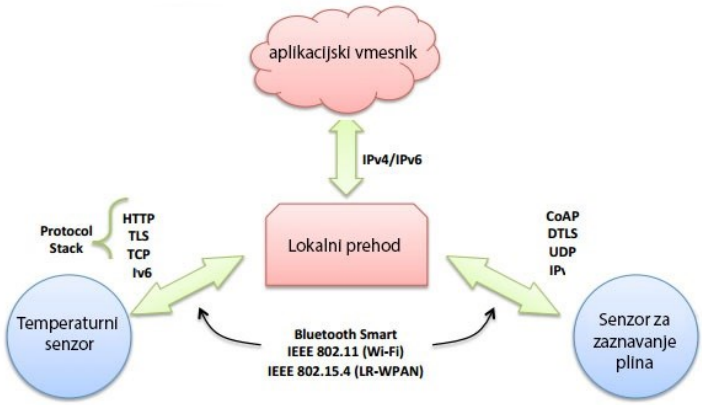
Za ta model je značilno, da so vse naprave neposredno povezane z oblačnim aplikacijskim vmesnikom, ki omogoča izmenjavo podatkov in nadzira podatkovni kanal. Za komunikacijo med napravami in internetnim oblakom model uporablja tradicionalne protokole kot so Ethernet oziroma Wi-Fi.

Slika 2.3: Skica komunikacijskega modela naprava z oblakom.

Ta model lahko uporabimo v primeru, ko imamo doma termostat, ki je povezan v oblak. Termostat pošilja izmerjene podatke v podatkovno bazo, kjer se jih uporabi za razne izračune ter analize nihanja temperature v stanovanju.[4]

2.4 Komunikacijski model: naprava s prehodom

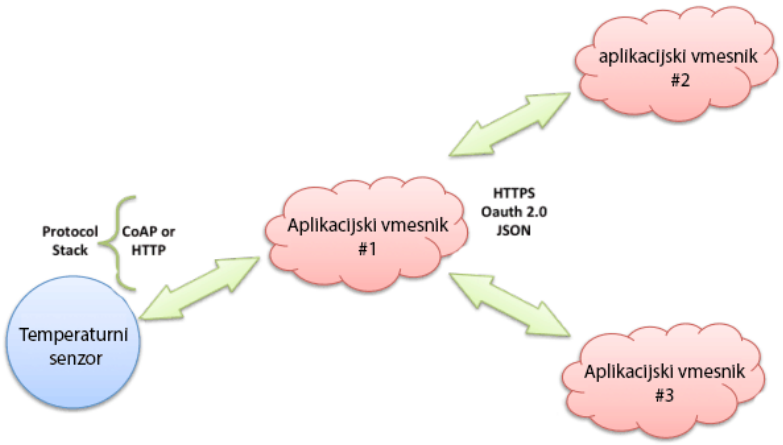
Za ta model je značilno to, da je med napravo in oblakom dodatna naprava oziroma storitev, ki omogoča komunikacijo med njima. To pomeni, da je na lokalnem omrežnem prehodu nameščena aplikacija, ki upravlja s prometom med napravo in oblakom ter zagotavlja določeno stopnjo varnosti [4].



Slika 2.4: Skica komunikacijskega modela naprava s prehodom.

2.5 Komunikacijski model izmenjevanja podatkov

Komunikacijski model ponuja komunikacijski arhitekturo, ki uporabniku omogoča, da lahko analizira podatke iz oblaka v kombinaciji z drugimi aplikacijami. Arhitektura dovoljuje uporabniku dostop do vseh podatkov naprav, ki se nahajajo v objektu. V bistvu gre za nadgradnjo komunikacijskega modela naprava – oblak, kjer lahko pooblaščen uporabnik dostopa do podatkovnega skladišča, v katerem so shranjeni podatki vseh naprav.



Slika 2.5: Skica komunikacijskega modela izmenjavanja podatkov.

Za dostop do podatkov drugih pametnih naprav je priporočeno, da se uporablja oblačne aplikacijske programske vmesnike (API) ali pa se dostop opravi prek storitve oblaka. [4].

Poglavje 3 Opis problema in zasnova sistema

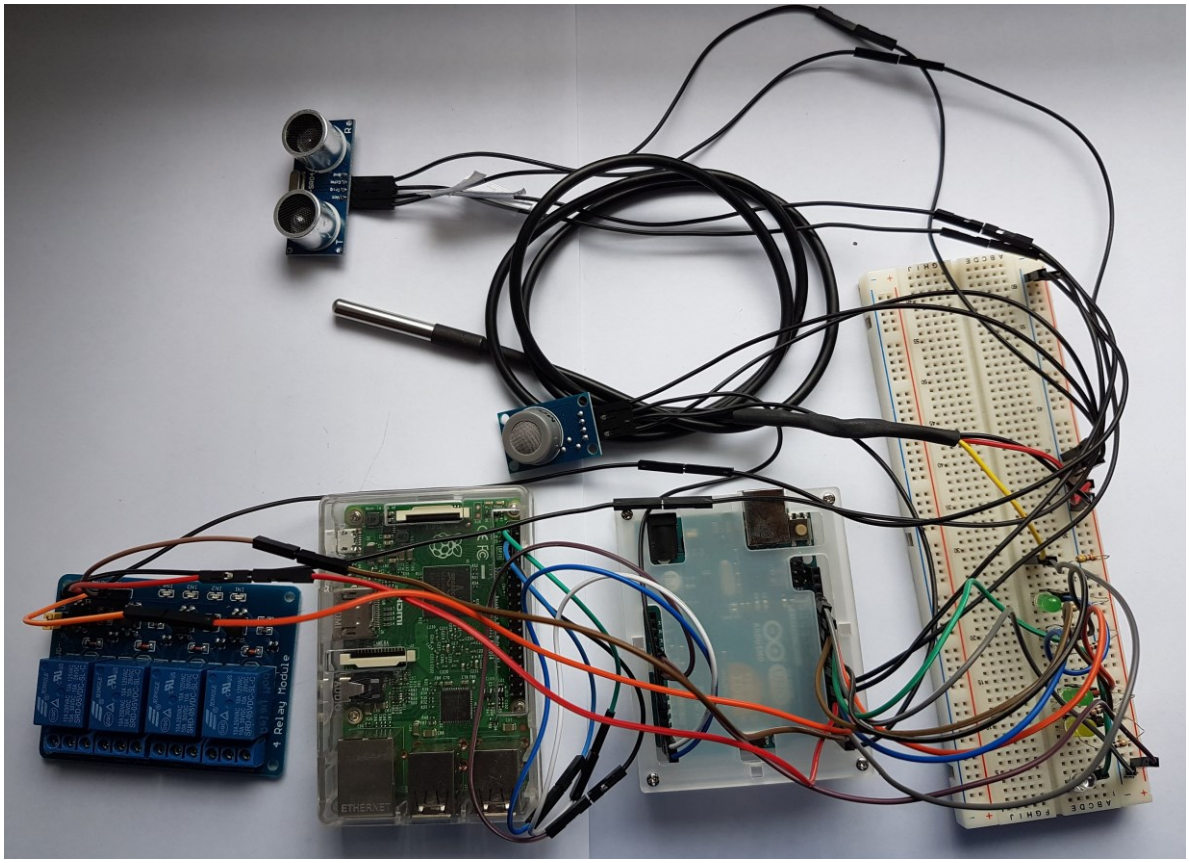
V tem poglavju je predstavljen problem, katerega bomo poskušali odpraviti v sklopu te diplomske naloge ter predstavljena je tudi arhitektura sistema za upravljanje vinske cisterne. Opisane so vse uporabljene naprave ter programska oprema, s katero je bil razvit celoten sistem.

3.1 Opis problema in postopek alkoholne fermentacije

Alkoholna fermentacija je proces pridelovanja vina iz grozdnega mošta. Postopek pridobivnaja vina iz grozdnega mošta se začne s vstavljanjem zmletega grozdja v cisterno. Postopek pa ni vedno enak. V odvisnosti od barve in vrste grozdja je potrebno nastaviti in vzdrževati primerno temperaturo mošta v cisterni med celotnim biološkim procesom. S tem zagotovimo konstantno temperaturo med celotnim procesom prodobivanja vina in se poskušamo izogniti nenadnim »izbruhom« mošta, ki bi rezultirali v izlitju vsebine cisterne. Poleg tega je potrebno celotno vsebino cisterne mešati, da ne pride do nastanka usedlina in da je celotna vsebina cisterne ohlajena / segreta na isto temperaturo. Med fermentacijo nastaja tudi človeku nevaren plin ogljikov monoksid. Potrebujemo torej cisterno, ki ima vgrajen mešalni in hladilno-grelni sistem. S pomočjo temperaturnega senzorja moramo nadzorovali temperaturo mošta v vinski cisterni ter izmerjene podatke shranjevati za kasnejšo obdelavo in statistiko. Hladilno-grelni sistem se mora vključiti, ko je temperatura mošta previsoka ali prenizka, da ohranja konstano temperaturo mošta. Potrebno je nadzorovati nivo gladine mošta, da ne pride do izlitja mošta. Poleg tega pa je potrebno še spremljanje koncentracije ogljikovega monoksida.

3.2 Zasnova sistema

Sistem za upravljanje vinske cisterne je namenjen nadzorovanju alkoholne fermentacije grozdnega mošta. Sistem je izdelan tako, da s pomočjo senzorjev, ki se nahajajo v notranjosti cisterne pridobiva podatke. Poleg tega mora sistem biti sposoben tudi upravljanja sistema za hlajenje in gretje cisterne. Sama izdelava mora biti cenovno ugodna in konkurenčna ostalim obstoječim sistemom.



Slika 3.1: Prototip sistema za krmiljenje cisterne.

3.3 Arhitektura sistema

Sistem je razdeljen na 3 glavne module:

- strežnik,
- krmilne naprave,
- spletna aplikacija, ki omogoča upravljanje sistema.

Strežnik je naprava, ki služi za obdelovanje podatkov, ki jih nato posreduje končnemu uporabniku. V našem projektu smo uporabili Raspberry Pi 2, ki je trenutno najbolj priljubljen ter izredno majhen in zmogljiv računalnik. Strežniška aplikacija je bila razvita s pomočjo programskega jezika Python. Aplikacija omogoča enostavno komuniciranje končnega uporabnika s krmilnikom naprav ter obratno. Kot je razvidno iz slike, končnemu uporabniku ni potrebno poznati zahtevnosti samega sistema.

Na strežniku je nameščen operacijski sistem Raspbian, ki temelji na osnovi Debian [9]. Na strežniku je nameščena tudi podatkovna baza MySql, ki omogoča trajno shranjevanje podatkov bodisi aplikacije bodisi (pooblaščenega) uporabnika. Poleg tega je na strežniku

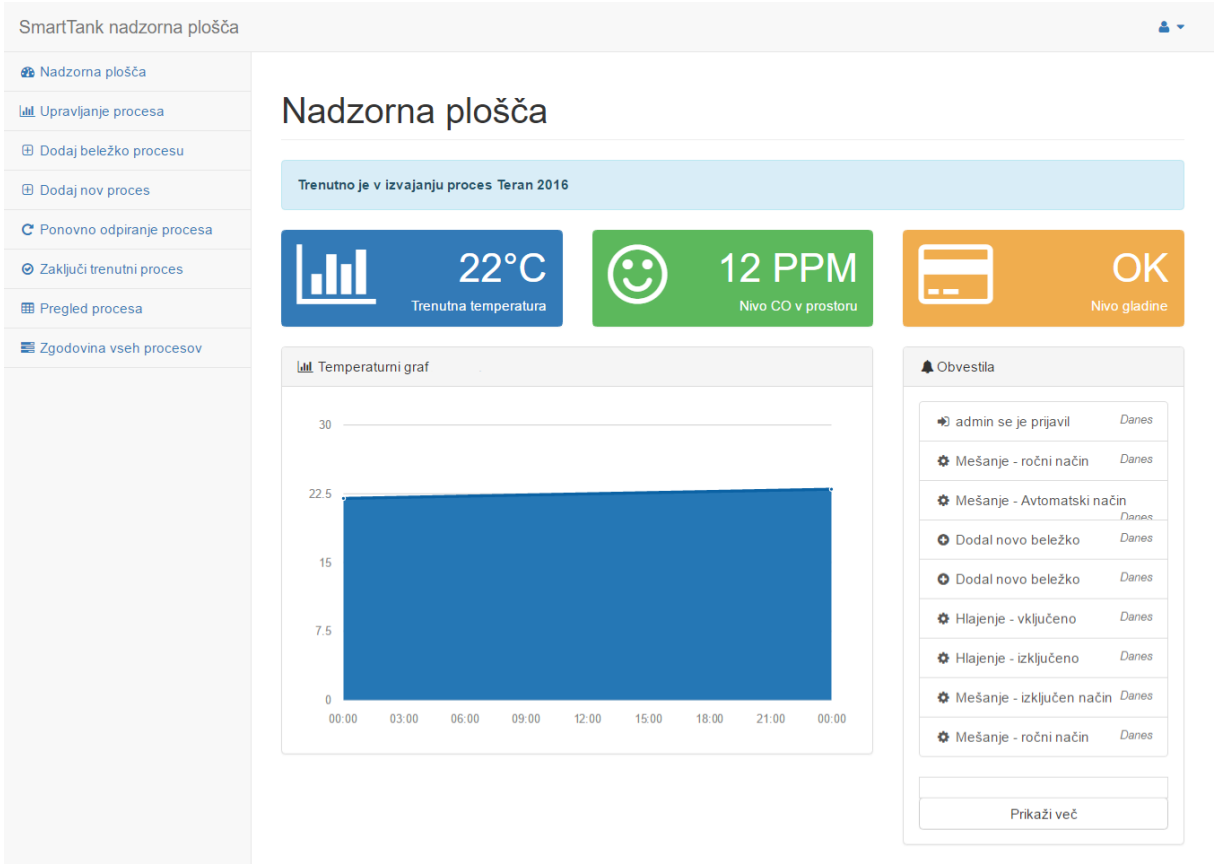
omogočena tudi izmenjava datotek po protokolu FTP, s katerim lahko uporabnik enostavno in hitro prenese datoteke na svoj računalnik ali z njega. Na strežniku gostimo tudi spletno aplikacijo za upravljanje sistema.

Krmilnik je naprava, ki skrbi, da vse naprave, ki so priključene nanj, delujejo pravilno, oziroma jim omogoča, da lahko nemoteno opravljajo svojo funkcijo. Za projekt smo izbrali Arduino Uno, ki je zelo priljubljen in enostaven krmilnik za upravljanje. Glede na to, da bi lahko uporabili Raspberry Pi 2 tako za strežnik kot tudi za krmilnik naprav smo se raje odločili za oba, ker ima krmilnik Arduino večjo podprtost senzorjev kot pa Raspberry. Poleg tega je tudi bistveno hitrejši, saj na njem ni nameščenega operacijskega sistema in zato ne potrebuje dodatnega časa za vzpostavitev sistema.

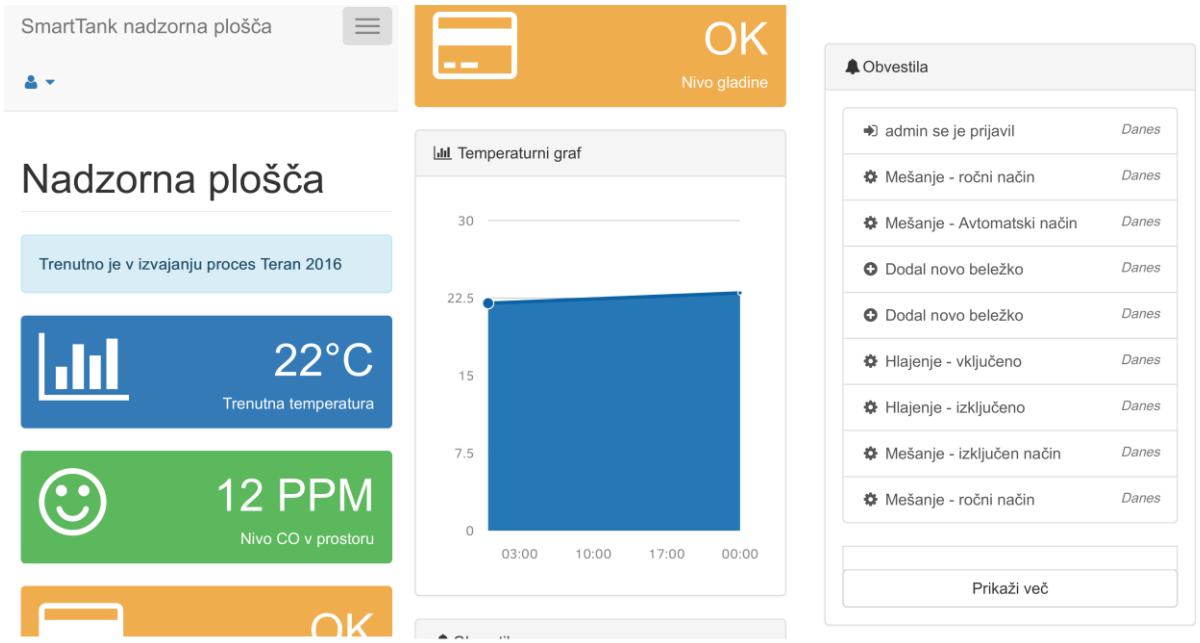
Strežnik in krmilnik sta med seboj fizično povezana po protokolu I^2C. Komunikacija poteka prek dveh vodil ter ene skupne mase. Eno vodilo je namenjeno prenosu podatkov in ukazov, drugo vodilo pa določa hitrost prenosa podatkov in določa, Kater naprava je v vrsti za pošiljanje in katera za prejemanje podatkov.

Za vključevanje hladilno-glelnega sistema in mešalnega sistem smo uporabili rele. Rele je stikalo, ki se vklopi ali izklopi, ko dobi signal od krmilnika [10]. Rele je bil uporabljen zato, ker je izhodna krmilna napetost na Arduino krmilniku samo 5 V in je bistveno premajhna za upravljanje hladilnega in grelnega sistema.

Spletna aplikacija je namenjena upravljanju in spremljanju procesa alkoholnega vretja mošta v vinski cisterni. Na spletu smo dobili brezplačno temo za spletno stran. Narejena je po standardu RWD, kar pomeni, da se spletna stran prilagaja resoluciji naprave, s katero dostopamo do nje. Če torej dostopamo s pametnega telefona, se bodo vsi elementi na spletni strani prilagodili zaslonu telefona (SlikaSlika 3.3) in bodo nekoliko drugačni od tistih, ki se bodo prikazali na zaslonu osebnega računalnika (Slika 3.2).



Slika 3.2: Spletna stran kot uporabniški vmesnik.



Slika 3.3: Mobilna spletna stran.

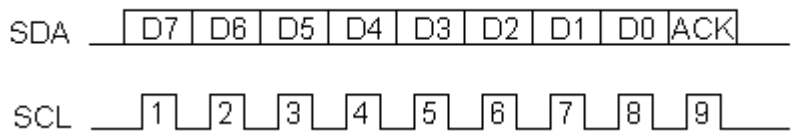
3.4 I2C Protokol

Pri komunikaciji med strežnikom in krmilnikom smo imeli nemalo težav. Na začetku smo uporabljali USB vodilo tako za napajanje kot tudi za sporazumevanje. Težava je bila v tem, da je prišlo do prevelikih izgub. To pomeni, da podatek, ki smo ga poslal iz strežnika ni prišel do cilja v pravi obliki. Zato smo se odločili za I2C protokol.

I2C oziroma I^2C je protokol, ki se ga uporablja pri komunikaciji procesorjev z mikrokrmilniki ter drugih digitalnih pretvornikov. Najpogosteje se ga uporablja pri komunikaciji v televizorjih, za kar ga je prvotno tudi naredilo podjetje Philips.

Vsa komunikacija poteka po dveh dvosmernih žičkah (SCL(Serial Clock Line) in SDA-(Serial Data Line)). Poleg je še žica za maso. SCL žica sporoča napravam signal, ki signalizira prenos podatkov med dvema moduloma, Žica SDA pa je namenjena izključno prenosu podatku med moduloma. Protokol deluje po principu godpodat/suženj, kar pomeni, da je ena naprava vedno podrejena, druga pa nadrejena. To pa ne pomeni, da podrejena naprava ne more pošiljati podatkov, je pa odvisna od nadrejene naprave, saj ta določa, kdaj pošilja in kdaj podrejena naprava prejema podatke [12].

Kot sem že dejali, napravi delujeta po principu gospodar/suženj. Nadrejena naprava je tista naprava, ki upravlja SCL uro. Podatke pošiljamo v sekvenci 8 bitov. Poleg 8 bitov je tukaj še 1 bit, ki služi kot ACK oziroma potrditev, da je naprava prejela 1 bajt podatkov (Slika 3.4).



Slika 3.4: Prenos podatkov v sklopu 8 bitov ter potrditveni bit.

Protokol I2C uporablja 7 bitni zapis za naslavljanje, kar pomeni, da nanj lahko priključimo do 128 naprav. Vedno, ko naprava pošilja 7 bitni naslov doda še 1 bit. Ta dodatni bit je namenjen podrejeni napravi in ji sporoča ali nadrejena naprava trenutno bere ali pošilja podatke [20].

Da smo lahko protokol uporabili na našem sistemu je bilo potrebno, da smo poleg fizičnega priklopa obeh naprav uvozili tudi potrebne knjižnice. Za Arduino Uno obstaja brezplačna knjižnica Wire.h, implementacija katerega je zelo enostavna. Na začetku programa nastavimo skupni naslov, po katerem bo potekala komunikacija. Podatke beremo s funkcijo »onReceive!, funkcija »onRequest« pa služi za pošiljanje podatkov nazaj na Raspberry.

```
Wire.begin(address);
Wire.onReceive(receiveEvent);
Wire.onRequest(requestEvent);
```

Primer 3.1: Primer uporabe knjižnice Wire.h v C++ jeziku.

Na strežniški strani smo protokol implementirali s programskim jezikom Python. SMBus je knjižnica namenjena I2C protokolu. Najprej je potrebno nastaviti skupni komunikacijski naslov. Branje podatkov poteka s funkcijo »read_byte«, ki potrebuje komunikacijski naslov za vhodni parameter ter funkcija »write_i2c_block_data«, ki omogoča pošiljanje podatkov. Vsi podatki, ki se prenašajo, so v bajtnem zapisu, zato pa potrebujemo še dodatno funkcijo za kodiranje podatkov.

```
import smbus#knjižnica za I2C protokol
bus = smbus.SMBus(1)#inicializacija
addresss = 0x07#nastavimo naslov
bus.read_byte(address)#branje iz naslova
bus.write_i2c_block_data(address,    messageInBytes)#pošiljanje    bajtnega
zapisa na naslov
```

Primer 3.2: Primer uporabe SMBus knjižnice v programskem jeziku Python.

3.4.1 Hladilno-grelni sistem

Namen sistema je, da vsebino cisterne ohladi oziroma segreje na določeno temperaturo. Sistema sta po principu delovanja enaka, vendar sta med seboj neodvisna. Za delovanje sistema je potreben zalogovnik. V tem zalogovniku se nahaja hladilna oziroma grelna tekočina. Tekočini se razlikujeta samo po temperaturi, kar pomeni, da bo hladilna tekočina hladnejša od grelne. Sistema ločeno dovedemo do cisterne, kar pomeni, da je grelni sistem speljan po drugi poti kot hladilni. Cisterna je narejena tako, da ima poleg stranice še dodatni plašč. Plašč je namenjen ravno hladilno-grelnemu sistemu. Hladilna tekočina potuje med plaščem in stranico ter tako hladi oziroma greje. Na cisterni so nameščene štiri cevi, vsakemu sistemu pripadata dve. Ena cev služi za dovod tekočine, druga tekočino odvaja. Sistem krmilimo s pomočjo rele stikala, ki je povezano z elektromagnetnim ventilom, ki omogoča odpiranje in zapiranje dovoda tekočine po plašču.

Slika 3.5: Elektromagnetni ventil.

3.4.2 Mešalni sistem

Namen mešalnega sistema si lahko predstavljamo s primerom Cedevite. Ko Cedevito stresemo v vodo, jo moramo dobro premešati, da se enakomerno raztopi v vodi, v nasprotnem primeru bo vsa zmes ostala na dnu kozarca. S pomočjo mešalnega sistema torej preprečimo, da bi v cisterni nastajala usedlina. Sistem sestavljajo motor, ki služi kot pogon sistema, reduktor, ter glavna gred, ki se nahaja v notranjosti cisterne. Motor in reduktor omogočata obračanje grede. Na gredi so lopate, ki so namenjene mešanju vsebine. Sistem upravljamo z rele stikalom. Ko vključimo sistem stikalu pošljemo signal, ta pa sproži napetost na motor in s tem se celotni sistem zažene.

3.5 Uporabljene naprave

Tukaj so opisane naprave, ki so bile uporabljene za krmiljenje in procesiranje celotnega sistema. Poleg tega so opisane lastnostni in osnovne karakteristike vseh uporabljenih senzorjev. Predstavljena so tudi vsa programska orodja, s katerimi smo razvili celoten sistem.

3.5.1 Arduino Uno

Arduino Uno je mikrokrmilnik, ki omogoča branje podatkov z različnih senzorjev ter lahko upravlja tudi z drugimi fizičnimi napravami. Razvit je bil v Italiji in je prvotno bil izdelan za študijske namene. Narejen je bil na osnovi ATmega328P mikrokrmilnika, z delovno frekvenco 16 MHz. Razvijalna ploščica ima 14 digitalnih ter 6 analognih priključkov, ki jih je mogoče uporabiti tako za vhodne kot tudi za izhodne naprave. Naprava nima nameščenega operacijskega sistema, kar pomeni, da takoj, ko napravo priključimo na električno napajanje začne izvajati program, ki je naložen na internem pomnilniku. Nanj lahko shranimo program do velikosti 32 KB. Napajamo ga preko zunanjega 5 V napajanja ali pa kar prek USB vhoda [11].

Slika 3.6: Mikrokrmilnik Arduino Uno.

3.5.2 Raspberry Pi 2

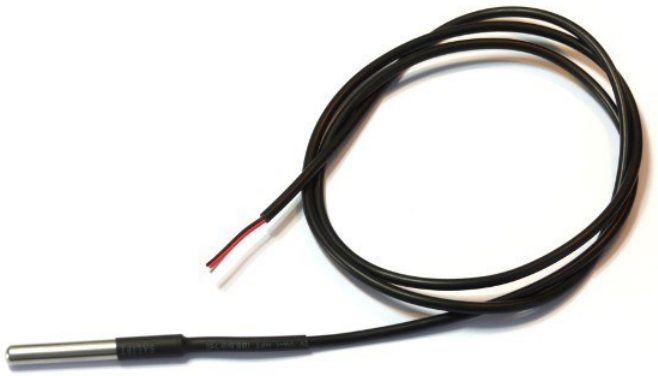
Če smo rekli, da je Arduino mikrokrmilnik, potem lahko rečemo, da je Raspberry Pi mikroračunalnik. Pravimo mu pa tudi kartični računalnik, saj ji je s svojimi dimenzijami zelo podoben. Razvili so ga v Veliki Britaniji za potrebe učenja. Kot pravi star pregovor »strup je v majhnih stekleničkah« in isto velja za ta mikroračunalnik. S štiri jedrnim, 1,2 GHz Arm Cortex-A7 procesorjem ter 1 GB velikim pomnilnikom je kos tudi zahtevnejšim operacijam. Nanj je mogoče priklopiti štiri USB naprave ter omogoča priklop zaslona prek HDMI priključka. Da pa ga lahko začnemo uporabljati, je nanj potrebno namestiti operacijski sistem. Za potrebe tega mikroračunalnika je bil razvit operacijski sistem Raspbian, ki temelji na UNIX osnovi. Namestimo ga tako, da programsko opremo naložimo na micro SD kartico ter sledimo navodilom, izpisanim na zaslonu. Sama namestitev traja približno petnajst minut [13].

Slika 3.7: Računalnik Raspberry Pi 2.

3.5.3 Uporabljeni senzorji

3.5.3.1 Temperaturni senzor

Temperaturni senzor služi za merjenje temperature mošta, znotraj cisterne. Uporabili smo temperaturni senzor oznake ds18b20, ki je povsem plastificiran in tudi vodoodporen. Deluje po principu eno-žičnega protokola, kar pomeni, da senzor uporablja samo eno žico za komunikacijo s krmilnikom. Poleg komunikacijske žičke vsebuje še dve drugi: ena se uporablja za napajanje, druga za ozemljitev. Senzor deluje v razponu od -55°C do + 125°C in ima ± 0,5°C odstopanja [15].



Slika 3.8: Temperaturni senzor oznake ds18b20.

3.5.3.2 Senzor za nadzor ogljikovega monoksida (CO)

Pri alkoholnem vretju mošta nastaja smrtonosni plin CO. Senzor MQ-7 je namenjen prav nadzorovanju tega. Deluje tako, da ciklično izmenjuje visoko (do 5 V) in nizko napetost (do

1,5 V). S tem povzroči segrevanje grelca znotraj senzorja MQ-7. Pri visoki napetosti se grelec segreje na višjo temperaturo, pri nizki pa se ohladi na nižjo tempretaruro. Zaznavanje plina CO poteka samo v ciklu nizke napetosti. V ciklu visoke napetosti poteka »čiščenje« senzorja, ki odstrani vse zajete pline, absorbirane v ciklu nizke napetosti [14].

Slika 3.9: Senzor za zaznavanje plina CO.

3.5.3.3 Ultrazvočni senzor

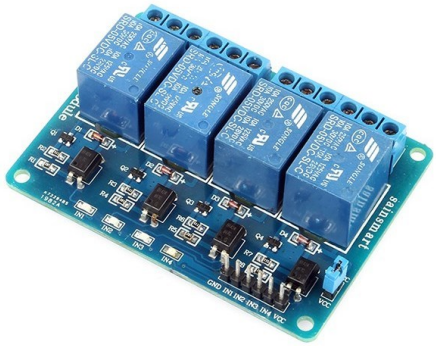
Ultrazvočni senzor se uporablja za nadzor višine gladine mošta v sami vinski cisterni. Izbrali smo senzor HC-SR04, ki je dovolj zmogljiv za naš projekt. Dodatno je bilo potrebno narediti še vodotesno ohišje, ki bo varovalo senzor pred tekočino. Domet senzorja je od 1 cm do 400 cm, z natančnostjo 3 mm [22].

Slika 3.10: Senzor za nadzor višine gladine v cisterni.

3.5.4 Rele

Pri izdelavi projekta smo uporabili ploščico s štirimi stikali oznake MSP430. Namen te je upravljanje hladilnih sistemov ter sistemov za mešanje zmesi znotraj cisterne. Rele je vrsta stikala pri katerem sta močnostni tokokrog ter krmilni tokokrog galvansko ločena. Tok, ki gre skozi magnetno navitje pa krmili rele. Za krmiljenje sistema ga potrebujemo zato, ker sta

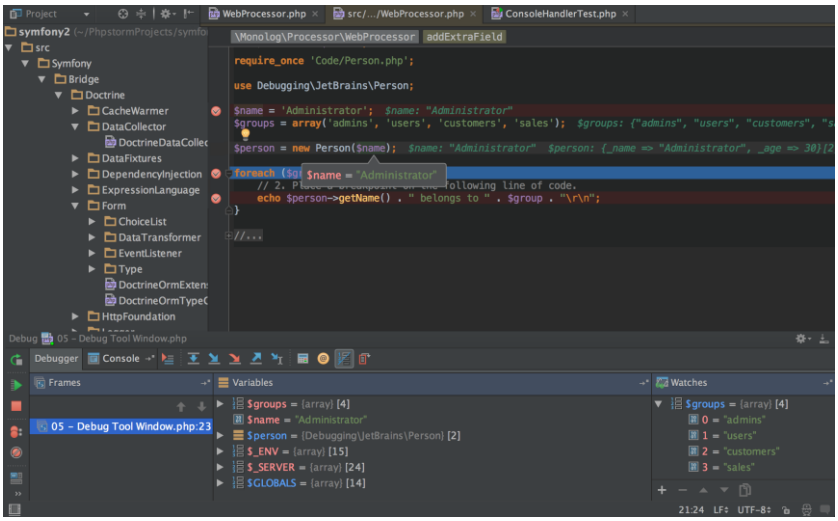
izhodna napetost in tok na Arduino ploščici premajhna. Stikalo omogoča krmiljenje z nizko napetostjo, na končno napravo pa privede potrebno napetost [10].



Slika 3.11: Štiri-kanalno stikalo.

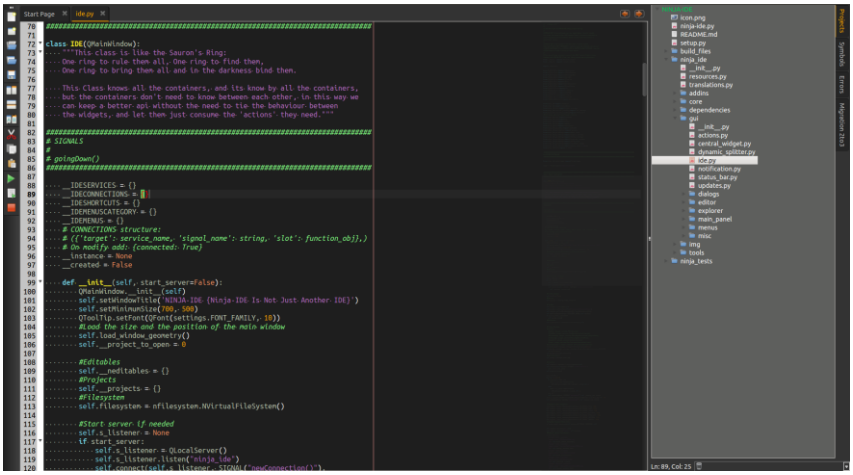
3.6 Uporabljena programska oprema

Za razvoj programske kode smo uporabili različna programska orodja. Za razvijanje spletne strani smo uporabili JetBrains PhpStorm. Razvojno okolje je narejeno na JetBrains platformi, celotna aplikacija jenarejena v Javanskem programskem jeziku in omogoča vključevanje različnih spletnih jezikov, kot so PHP, HTML in JavaScript. V veliko pomoč je tudi vgrajena funkcija razvojnega okolja, ki predlaga oziroma popravi določene dele kode [18].



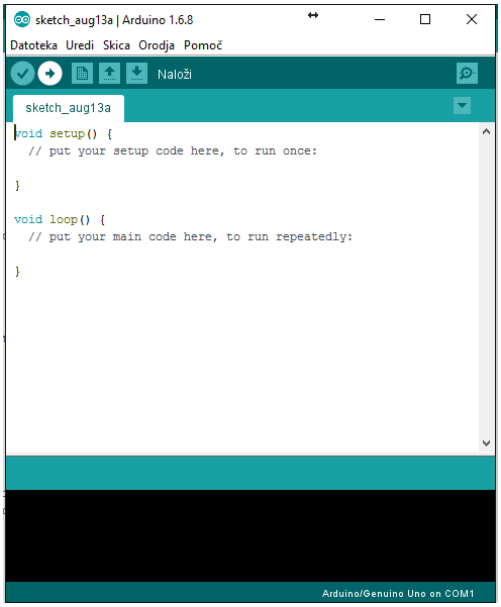
Slika 3.12: Razvojno okolje JetBrains PhpStorm.

Za razvijanje strežniške aplikacije smo uporabili razvojno orodje Ninja-IDE, ki pa je brezplačeno. Razvojno okolje je namenjeno predvsem razvijanju Python aplikacij. Sama aplikacija pa je razvita ravno v programskem jeziku Python [17].



Slika 3.13: Razvojno okolje Ninja-IDE.

Za potrebe razvijanja Arduino krmilne aplikacije smo uporabili kar Arduinovo integrirano razvojno orodje, ki je narejeno v programskem jeziku Java. Razvojno okolje podpira razvijanje krmilne aplikacije v C ali pa v C++ programskem jeziku. Vsak Arduino program ima dve glavni funkciji, in sicer funkciji »setup« in »loop«. Razlika med njima je ta, da se funkcija »setup« izvede samo enkrat na začetku programa (inicializacija spremenljivk), funkcija »loop« pa se, kot pove že samo ime funkcije, izvaja v neskončni zanki. Tako torej celotno logiko, ki jo želimo izvajati na krmilniku, postavimo znotraj funkcije »loop« [16].



Slika 3.14: Razvojno okolje Arduino IDE.

Poglavje 4 Opis sistema

Namen tega projekta je, da uporabniku čim bolj olajšamo delo pri upravljanju cisterne. Omogočeno je pregledovanje vsebine ter urejanje in spreminjanje stanj. Glavni poudarek je na upravljanju, kar pomeni, da uporabniku ni potrebno biti fizično prisotnemu, temveč lahko zadevo upravlja kar preko računalnika, oziroma pametnega telefona. Edino, kar uporabnik potrebuje, je internetni dostop, saj je celoten sistem priključen na internet. Sistem torej omogoča pregled stanj senzorjev, omogoča pa tudi upravljanje določenih sistemov kot so recimo hladilni sistemi ter pregled vseh dogajanj v zvezi s trenutnim procesom ali pa s procesi, ki so se izvajali predhodno.

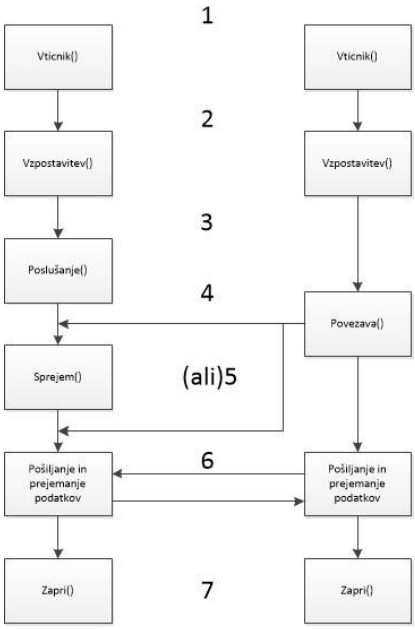
Kot smo že omenili uporabnik upravlja sistem prek spletne aplikacije. Za dostop do nadzorne plošče mora uporabnik imeti svoje uporabniško ime in geslo, saj je dostop onemogočen brez avtentikacije uporabnika. Aplikacija je nameščena na strežniku in s pomočjo strežniške aplikacije neposredno upravlja celoten sistem. Vsi podatki so shranjeni v interni podatkovni bazi. Strežniška aplikacija deluje kot posrednik, saj na podlagi argumenta, ki ga dobi iz uporabniškega vmesnika, posreduje podatke krmilni aplikaciji ter sprejme njene povratne informacije in jih prikaže v uporabniku prijazni obliki.

Krmilna aplikacija je shranjena na krmilnem pomnilniku in deluje na podlagi od strežnika prejetega sporočila. Lahko bi dejali, da sta strežnik in krmilnik povezana po principu razmerja gospodar - suženj, kjer je strežnik gospodar, krmilnik pa suženj.

4.1 TCP/IP Odjemalec - strežnik komunikacija

TCP/IP Server – client komunikacijo uporabljamo za sporazumevanje med spletno in strežniško aplikacijo. Deluje na podlagi IP naslova in vrat po katerih poteka komunikacija.

Na strežniški strani določimo IP naslov naprave ter vrata, prek katerih bo strežniška aplikacija spremljala promet. Na drugi strani imamo odjemalca, ki poskuša komunicirati s strežnikom. Pomembno je, da obema nastavimo iste komunikacijske podatke, da lahko modula komunicirata. Ko strežnik sprejme odjemalčev zahtevek, se povezava med njima vzpostavi.



Slika 4.1: Potek komunikacije med odjemalcem in strežnikom.

Strežniško aplikacijo smo realizirali v programskem jeziku Python. Za pošiljanje in prejemanje podatkov med odjemalcem in strežnikom je potrebno ustvariti vtičnik. V Pythonu to naredimo tako, da uvozimo knjižnico »socket«. Potrebno je nastaviti IP naslov ter vrata, po katerih bo potekala komunikacija. Primer vzpostavitve komunikacije med odjemalcem in strežnikom si lahko ogledamo na spodnjem primeru kode.

```
import socket
IP = 'localhost'#IP naslov, kjer se nahaja strežniška aplikacija
PORT = 5005#vrata, na katerih posluša strežniška aplikacija
SIZE = 1024#velikost povratne informacije
MESSAGE = "hello"
soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)#inicializacija
vtičnika
soc.connect((IP, PORT))#vzpostavimo povezavo s strežnikom
soc.send(MESSAGE)#pošljemo podatke strežniku
data = soc.recv(SIZE)#povratno sporočilo od strežnika
```

Primer 4.1:Primer odjemalčeve aplikacije.

```
import socket

IP = '127.0.0.1'#nastavitev IP naslova
PORT = 5005#vrata komunikacije
BUFF_SIZE = 1024#velikost sporočila
soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)#deklaracija
vtičnika
soc.bind((IP,PORT))#konfiguriranje vtičnika
soc.listen(5)#poslušanje na vtičniku
data = conn.recv(BUFF_SIZE)#prebrani podatki
```

```
conn.send('Helo')#povratno sporočilo
conn.close()#zapiranje povezave
```

Primer 4.2: Primer strežniške aplikacije.

4.2 Strežniška aplikacija

Strežniška aplikacija predstavlja osrednji del sistema, saj povezuje vse elemente v celoto. Poleg tega omogoča tudi nadziranje in upravljanje sistema.

Strežniška aplikacija povezuje dva pomembna sistema, in sicer:

- spletno aplikacijo in
- krmilno aplikacijo.

Spletna aplikacija predstavlja uporabniški vmesnik, s katerim uporabnik upravlja celoten sistem. Dobra lastnost uporabniškega vmesnika je ta, da uporabniku ni potrebno poznati tehničnih zahtev ter procesov, ki se izvajajo v ozadju. Na podlagi uporabnikove zahteve spletna stran pošlje argument strežniški aplikaciji, ta pa na podlagi uporabniške izbire posreduje zahtevan ukaz krmilni aplikaciji, ki odgovori z ustreznimi podatki, ki se nato uporabniku prikažejo na spletni strani.

4.3 Spletna aplikacija

Spletna stran je namenjena uporabniku in je edini modul, prek katerega lahko uporabnik upravlja sistem. Predloga spletne strani je narejena po načinu odzivnega spletnega oblikovanja, kar pomeni, da se spletna stran prilagaja mediju, s katerim dostopamo do nje. Prednost tega je, da imamo lahko samo eno aplikacijo, ki se prilagaja zaslonu medija.

Spletna aplikacija je sestavljena iz več podstrani. Sestavljajo jo:

- prijavna stran,
- začetna stran,
- krmilna stran,
- upravljalna stran,
- pregledna stran in

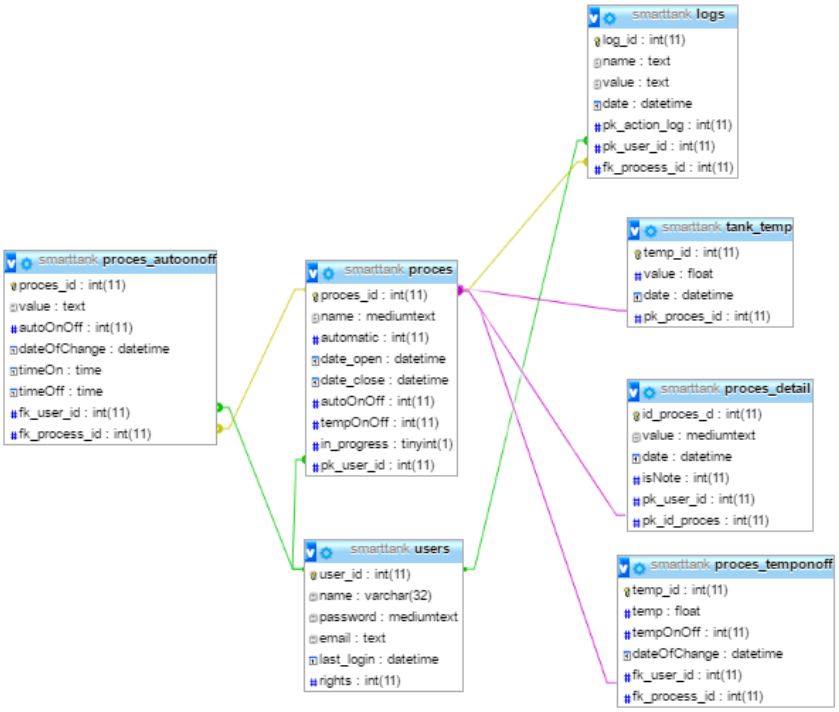
- uporabniška stran.

Spletna vsebina pa ni namenjena vsakomur. Vsak uporabnik se mora predstaviti z uporabniškim imenom in geslom. Za dodatno varnost smo geslo preoblikovali s pomočjo zgoščevalne funkcije Sha. Vsi podatki se nahajajo v podatkovni bazi.

Spletno aplikacijo smo v celoti razvili s HTML, PHP in JavaScript programskimi jeziki. Za oblikovanje smo uporabili CSS.

4.3.1 Podatkovna baza

Podatkovna baza je zbirka podatkov, kateri so shranjeni v posameznih tabelah podatkovne zbirke. Da lahko podatke uporabljamo in spreminjamo, moramo do njih dostopati. To nam omogoča SUPB (sistem za upravljanje podatkovne baze), ki predstavlja zbirko programov za upravljanje in vzdrževanje podatkovnih baz. V podatkovni bazi hranimo vse podatke, ki jih potrebujemo za upravljanje sistema [21].



Slika 4.2:Shema podatkovne baze.

Uporabili smo podatkovno bazo MySQL, ki se nahaja na strežniku. Povezavo s podatkovno zbirko smo izvedli s PHP skriptnim jezikom. Na spodnjem primeru je prikazan primer vzpostavitve povezave s podatkovno bazo.

```
$povezi=mysqli(Streznik,pb,pb_UpIme,pb_UpGeslo)
if($povezi -> connect_error){
    echo("Napaka pri povezavi na podatkovno bazo!");
}
```

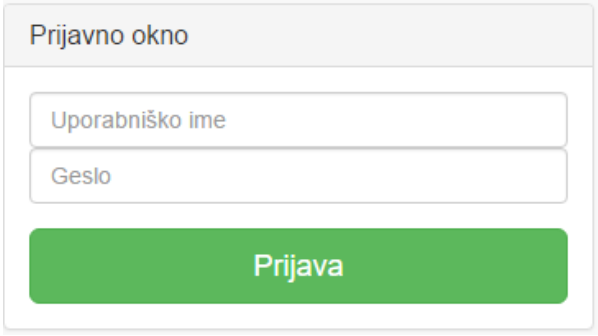
Primer 4.3: Primer enostavne povezave na podatkovno bazo.

Našo podatkovno zbirko sestavlja sedem različnih tabel.

- Uporabniki (»users«).
    - Tukaj hranimo podatke kot so uporabniško ime in geslo, ki uporabniku omogočata prijavo v sistem ter podatke o avtorizacijah.
- Proces (»process«).
    - Tukaj hranimo osnovne podatke o imenu procesa, kdaj je proces nastal, kdaj se je zaključil ter katere funkcije so vključene.
- Podrobno o procesu (»process_detail«).
    - V tej tabeli se shranjene vse podrobnosti o nekem procesu, kdo ga je urejal, kaj je bilo spremenjeno,…
- Logi (»logs«).
    - V to tabelo zapisujemo vsako dejanje, ki ga izvede uporabnik.
- Temperatura cisterne (»tank_temp«).
    - Tukaj se shranjujejo podatke o temperaturi mošta znotraj cisterne.
- Sistem za hlajenje in gretje (»proces_tempOnOff«).
    - Tukaj hranimo nastavitve za hladilni in grelni sistem, kot sta recimo način in režim izvajanja.
- Mešalni sistem (»proces_autoOnOff«).
    - Ta tabela ima enak namen kot tabela »proces_tempOnOff«, le da tukaj hranimo podatke, ki so namenjeni mešalnemu sistemu.

4.3.2 Prijavna stran

Prijavna stran služi za avtentikacijo uporabnika v sistem. Uporabnik mora vpisati svoje uporabniško ime in geslo. Če sistem uporabnika prepozna, ga preusmeri na začetno stran. V nasprotnem primeru dobi povratno opozorilo o napačnem vnosu podatkov. Po uspešni prijavi v sistem si aplikacija zapomni čas in uporabniško identifikacijo ter vse potrebne podatke zapiše v tabelo »logs«, poleg tega se ustvari še seja. Sejo ustvarimo tako, da pokličemo funkcijo »Session_start()«, ter v globalno spremenljivko »$_SESSION('tank')« shranimo vrednost prijave, v »$_SESSION('rights')« pa shranimo avtorizacije uporabnika. Če bi ob uspešni prijavi ponovno poskušali dostopati do prijavne strani, bi nas sistem samodejno preusmeril na začetno stran, saj se seja še ni zaključila. Seja traja tako dolgo, dokler se uporabnik ne odjavi. Slika 4.3 prikazuje prijavni obrazec, ki ga mora uporabnik izpolniti za prijavo v sistem.



Slika 4.3: Prijavno okno.

4.3.3 Začetna stran

Po uspešni prijavi nas sistem preusmeri na začetno stran, ki nam omogoča hiter pregled najpomembnejših podatkov.

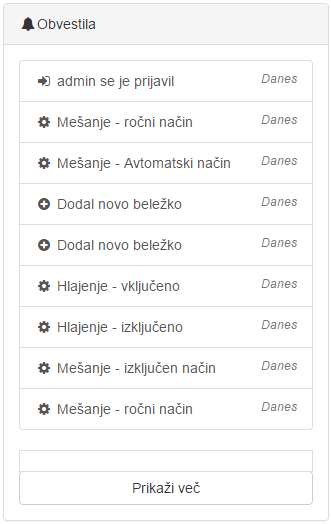
Skrajno desno zgoraj se nahaja gumb z motivom osebe, ki nam omogoča odjavo iz sistema ter urejanje osebnih uporabniških podatkov. Odjava deluje tako, da pokličemo funkcijo »session_destroy()« ter globalni spremenljivki »$_SESSION('tank')« spremenimo vrednost. S tem, ko pobrišemo sejo, je stran za neprijavljenega uporabnika onemogočena.

Na zgornji strani se nahajajo tri informativna okenca. Prvo nam sporoča trenutno temperaturo mošta v cisterni. Srednje predstavlja trenutno vrednost ogljikovega monoksida v ozračju;. podatek je podan v PPM enoti, oziroma v enoti za merjenje koncentracije. V primeru, da je koncentracija ogljikovega monoksida previsoka se ozadje okna obarva rdeče. Na skrajni desni strani se nahaja še zadnje okence, ki ponazarja trenutno stanje višine gladine v cisterni. V

primeru, da se nivo gladine nenadno spremeni, se ozadje okenca prav tako obarva rdeče. Podatke preberemo iz krmilne naprave. Preden se spletna stran naloži se izvede »odjemalec« aplikacija, ki pridobi trenutna stanja v cisterni. Odjemalčevi aplikaciji podamo zahtevane argumente, ta pa nam vrne podatke iz krmilne naprave.

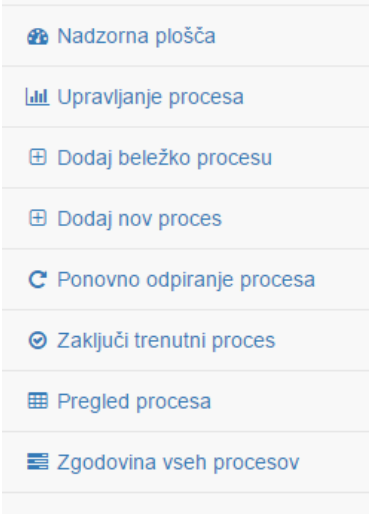
Slika 4.4: Informativna okna.

Desno, tik pod informativnimi okenci se nahaja polje z obvestili, ki služi za hitri pregled nedavnih dejanj uporabnikov, ki so jih počeli uporabniki. Podatke beremo iz podatkovne baze, natančneje iz tabele »logs«. Dostop do teh zapisov ima samo administrator. Za preverjanje avtorizacij si pomagamo s spremenljivko »$_SESSION('rights')«, v kateri hranimo uporabniške avtorizacije.



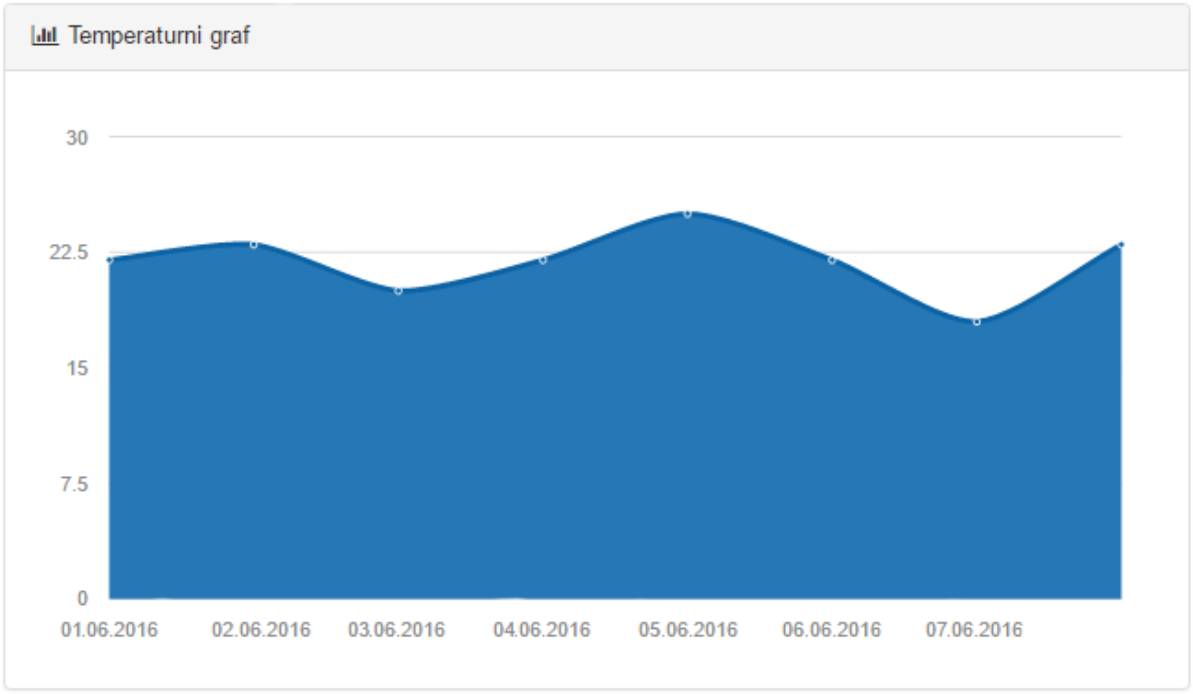
Slika 4.5: Obvestila.

Na levi strani se nahaja meni, s katerim se lahko sprehajamo po vseh straneh naše spletne strani. Meni se pojavi na vsaki podstrani, kar pomeni, da ima uporabnik dostop do vse vsebine vseh spletnih strani.



Slika 4.6: Navigacijski meni.

Na sredini imamo graf gibanja temperature, ki prikazuje nihanje temperature mošta v cisterni. Podatke beremo neposredno iz podatkovne baze. Ker je graf narejen v JavaScriptu, moramo podatke, prebrane iz podatkovne baze, shraniti v asociativno tabelo.



Slika 4.7: Temperaturni graf.

```
$s = $povezava_naPb->prepare($sql_querry);
$s->execute();
$rezultat = $s->fetchAll(PDO::FETCH_ASSOC);
```

```
json_encode($rezultat);
```

Primer 4.4: Primer shranjevanja podatkov v asociativno tabelo.

S pomočjo JSON formata nato posredujemo tabelo funkciji, ki je namenjena izrisovanju grafa. Še pred tem pa moramo asociativno tabelo pretvoriti v JSON zapis s pomočjo funkcije »JSON. Stringify()«.

4.3.4 Odpiranje novega procesa

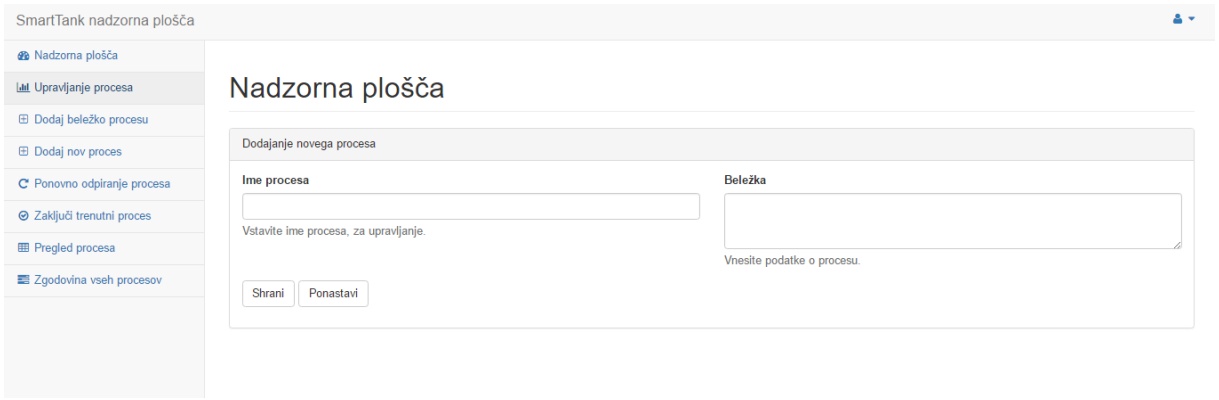
S procesom označujemo dogajanje, ki se izvaja v cisterni. V našem primeru se ukvarjamo s procesom fermentacije oziroma procesom alkoholnega vrenja grozdne drozge.

Preden začnemo z upravljanjem moramo najprej dodati nov proces v primeru, če noben proces ni v stanju izvajanja. Vnos je zelo enostaven. Uporabnik mora vnesti ime procesa, ki bo služil za vodenje. Poleg tega ima uporabnik še eno vnosno polje »Beležka«. To polje je opcijsko in ni obvezno. Beležka je podrobneje opisana v podpoglavju 4.3.6.

Ko uporabnik potrdi podatke, se aplikacija odzove na različne načine. V primeru, da je nek proces že v izvajanju, aplikacija uporabnika o tem opozori in mu svetuje, da naj proces najprej zaključiti, če želi odpreti novega. Če uporabnik ne vnese imena procesa, ga spletna stran opozori, da obvezno polje mora biti izpolnjeno. V kolikor je uporabnik pravilno izpolnil podatke in v izvajanju ni nobenega predhodnega procesa, se v podatkovno bazo shranijo podatki o procesu ter aplikacija vrne obvestilo o uspešnem odpiranju procesa.

```
<?phpif($napaka==0) {?>
    <div class="alert alert-success">
        Nov proces je bil uspešno dodan v sistem.
    </div>
<?php}else if($napaka==1) {?>
    <div class="alert alert-info">
        Napaka pri vnosu v sistem. Preverite vnesene podatke!
    </div>
<?php}else if($napaka==2){?>
    <div class="alert alert-info">
        Če želite odpreti nov proces, morate naprej starega zapreti!
    </div>
    <?php}?>
```

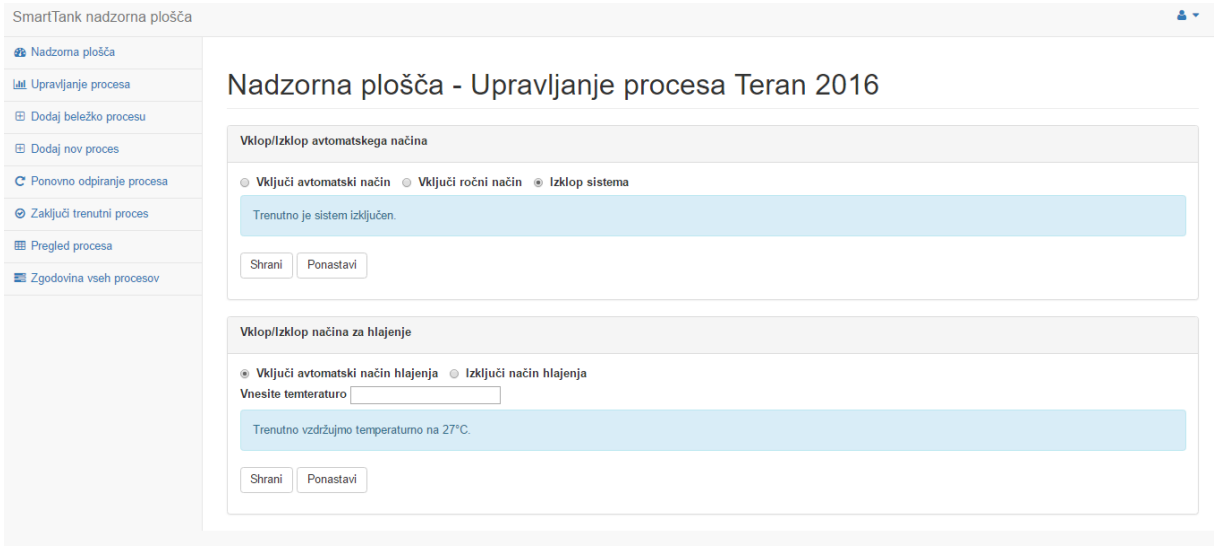
Primer 4.5: Primer izpisa pravilnega obvestila uporabniku.



Slika 4.8: Dodajanje novega procesa.

4.3.5 Krmilna stran

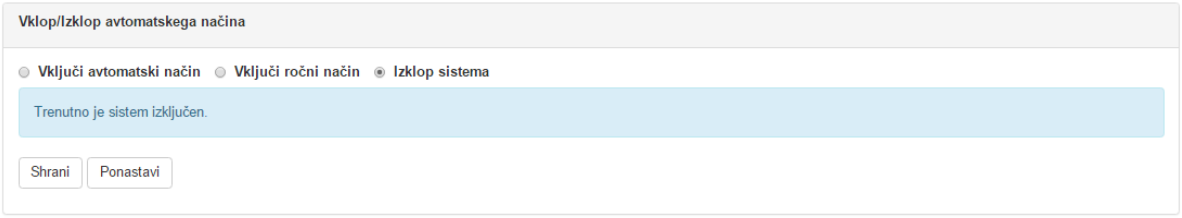
Krmilna stran je stran, na kateri lahko uporabnik upravlja krmilne naprave. Stran je razdeljena na dva dela. Zgornji del je namenjen samo mešalnemu sistemu, spodnji pa je namenjen izključno hladilno-grelnemu sistemu. V primeru, da noben proces ni v izvajanju, nam aplikacija to sporoči in onemogoči njegovo upravljanje.



Slika 4.9: Stran za upravljanje.

Za nastavitev mešalnega sistema imamo na voljo tri režime:

- avtomatski način,
- ročni način,
- izklop.

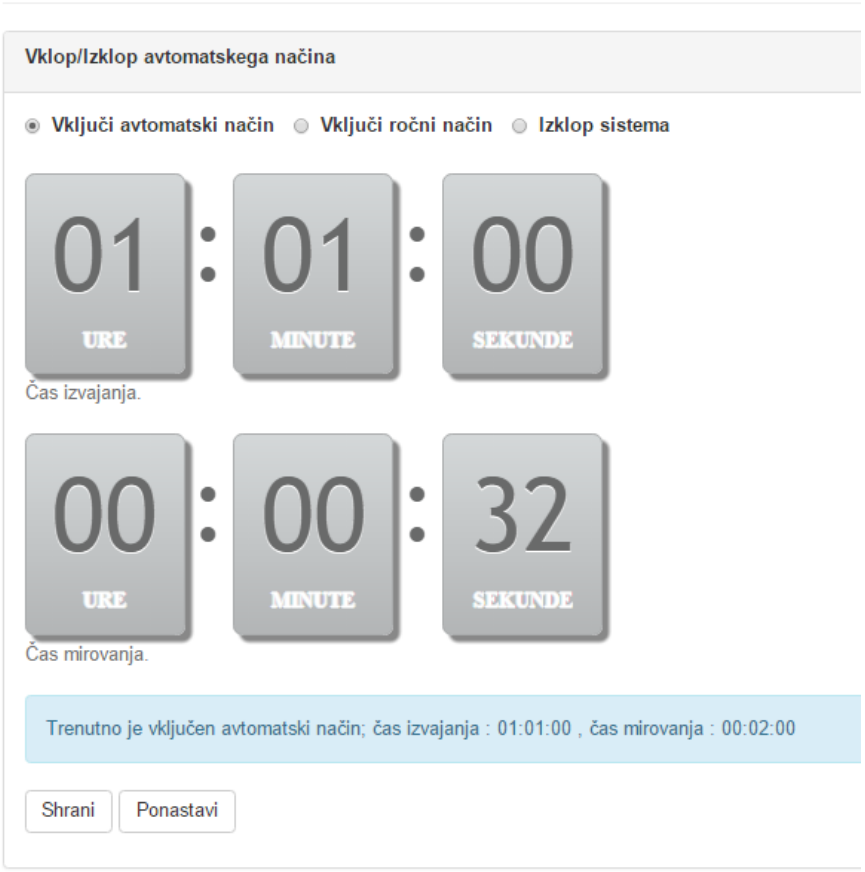


Slika 4.10: Upravljanje mešalnega sistema.

Pri avtomatskem načinu imamo na voljo dva pogleda, ki se prikazujeta v odvisnosti od stanja samega sistema. V primeru, da je mešalni sistem izključen, se v aplikaciji prikažeta dve vnosni polji. Ti dve polji služita za vnos izvajalnega časa in časa mirovanja. Ta dva podatka potrebujemo za to, da vemo, koliko časa bomo izvajali dejansko mešanje in koliko časa bo trajalo mirovanje sistema. Po uspešnem vnosu se podatka preneseta v podatkovno bazo in na krmilno aplikacijo. Drugi pogled prikazuje dva odštevalnika. Zgornji odšteva izvajalni čas, spodnji pa je namenjen mirovnemu času. Odštevalnika sta med seboj odvisna, kar pomeni, da naenkrat deluje samo eden. Naslednji prične odštevati šele, ko predhodni zaključi oziroma pride do 0. Odštevalni čas izračunamo ob vsakem ponovnem nalaganju spletne strani. Čas odštevalnika izračunamo po enačbi (4.1).

$$odštevalni\_čas = čas\_pb - čas\_krmilnik - čas\_nalaganja\_strani \tag{4.1}$$

»Odštevalni_čas« predstavlja naš končni čas za odštevalnik, »čas_pb« pa je podatek, ki ga preberemo iz podatkovne baze in predstavlja izvajalni čas za določeno stanje. Od krmilne naprave dobimo podatek »čas_krmilnik«, ki nam pove koliko časa se sistem že izvaja, poleg tega pa moramo upoštevati tudi čas, ki ga spletna stran potrebuje za nalaganje celotne vsebine. Ta čas je predstavljen s »čas_nalaganja_strani«. Nalagalni čas izmerimo s pomočjo JavaScript funkcije »Date«. Na začetku nalaganja se v spremenljivko shrani trenutni čas. Ko je vsebina dokončno naložena trenutni čas odštejemo s časom spremenljivke in tako dobimo čas nalaganja spletne strani.



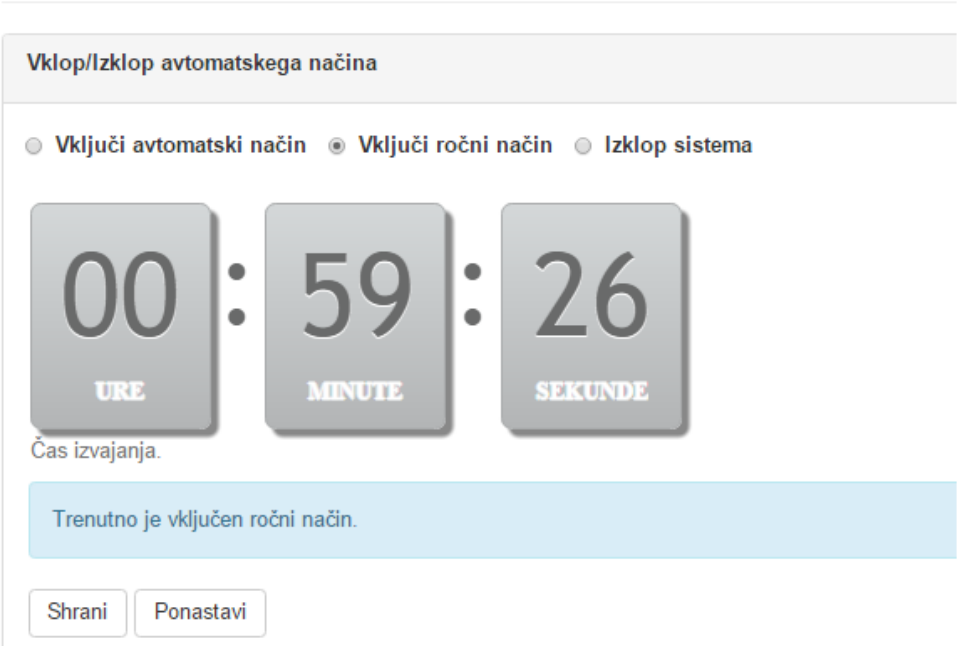
Slika 4.11: Odštevalnik časa v avtomatskem načinu.

Aplikacija mora vedeti kateri izmed odštevalnikov mora najprej začeti odštevati. Podatek dobimo od krmilne naprave, ki nam sporoči v katerem stanju se sistem nahaja. Na podlagi tega aplikacija zažene ustrezen odštevalnik.

```
if(sistemMesanja == true) {
        timer_mesanje
}else{
        timer_mirovanje
}
```

Primer 4.6: Enostaven primer preverjanja stanja sistema.

Ročni način deluje nekoliko drugače od avtomatskega načina. Bistvena razlika med njima je ta, da v ročnem načinu mešanje poteka ves čas. Uporabniku ni potrebno vnašati nobenega izvajalnega časa, saj mora samo izbrati možnost »Ročni način«, ki se izvaja do uporabnikovega preklica. Predpostavimo, da je uporabnik izbral ročni način. Ob ponovnem nalaganju spletne strani se prikaže, podobno kot pri avtomatskem načinu, ura, ki prikazuje čas izvajanja. Podatek je prebran s krmilne naprave.



Slika 4.12: Pretečeni čas v ročnem načinu.

Za izklop sistema imamo možnost »Izklop«. Funkcija postavi mešalni sistem v fazo mirovanja in tam ostane, dokler uporabnik ne izbere avtomatskega ali ročnega načina. Če želimo preklopiti iz avtomatskega v ročni sistem in obratno, moramo vedno najprej izključiti sistem in šele nato nam aplikacija omogoči spremembo načina..

V modrem okvirčku nam sistem sporoča kateri način je izbran. Tako je uporabnik vedno seznanjen s stanjem cisterne.

Za razliko od mešalnega sistema nam hladilno-grelni sistem nudi samo dve možnosti:

- avtomatski način,
- izklop.

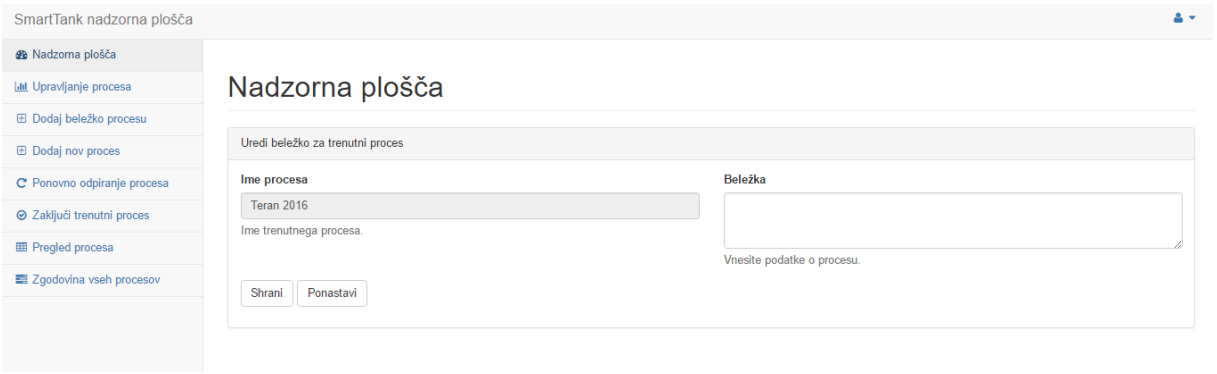
Pri avtomatskem načinu je zahtevan vnos temperature. Ta podatek sistemu sporoči na katero temperaturo mora cisterno ohladiti oziroma segreti. Uporabniku ni potrebno vedeti ali mora zagnati grelni ali hladilni sistem, saj se na podlagi trenutne temperature sistem sam odloči, kateri sistem je primernejši.

Možnost »Izklop« je precej podobna tisti pri mešalnem sistemu. Razlika je le ta, da sistema ni potrebno izključiti vsakič, ko želimo spremeniti temperaturo. Uporabljamo ga samo, ko želimo izključiti hladilno-grelni sistem.

Tudi tukaj uporabljamo modri okvirček za sporočanje o stanju hladilno-grelnega sistema.

4.3.6 Dodajanje beležke

Beležka se uporablja za zapisovanje informacij o procesu. Vanjo uporabnik lahko zapisuje različne podatke kot so sestavine, ki jih je uporabil, različna opažanja v času procesa in tako naprej.



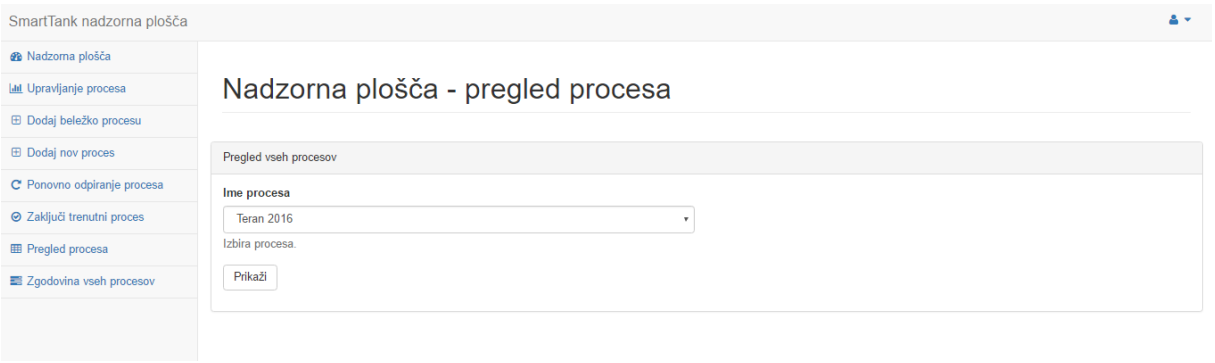
Slika 4.13: Dodajanje beležke procesu.

Stran precej spominja na tisto za dodajanje novega procesa, vendar se po funkcionalnosti nekoliko razlikujeta. Uporabniku ni potrebno ročno izbirati, h kateremu procesu želi dodati beležko, saj aplikacija samodejno izbere proces v izvajanju in onemogoča dodajanje beležke drugim, zaprtim procesom. Uporabnik mora samo izpolniti vnosno polje za beležko in s potrditvijo na gumb »Shrani« se podatki prenesejo v podatkovno skladišče. Gumb »Ponastavi« pobriše vso vneseno vsebino, tako da je uporabniku ni potrebno ročno brisati.

4.3.7 Pregled procesov

Pregled procesov prikazuje vse podatke oziroma informacije, ki so bile zapisane v okviru določenega procesa. Tukaj se prikazujejo vse vnesene beležke ter tudi informacije, ki jih je sistem samodejno zapisal. V samodejne zapise sodijo informacije o odprtju procesa, o vključitvi in izključitvi določenega sistema ter zaključitvi procesa samega.

Ko se stran naloži, se nam na zgornji strani prikaže spustni meni. V meniju se nahajajo vsi vneseni procesi.



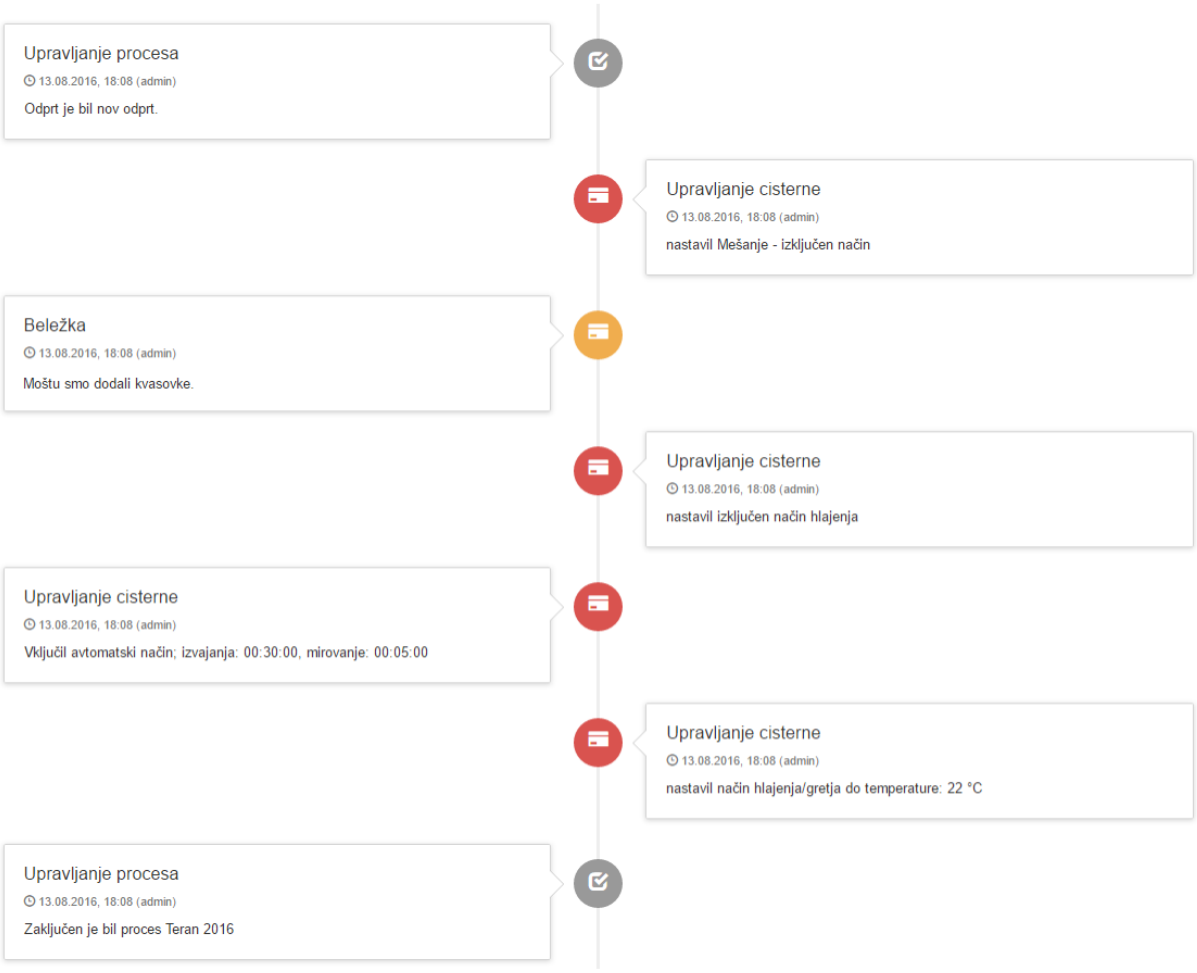
Slika 4.14: Izbira procesa za prikaz podrobnosti.

Na prvem mestu je prikazan trenutno izvajajoči se proces, sicer pa se prikažejo procesi po datumu nastanka v padajočem vrstnem redu. Na podlagi uporabniške odločitve se prikažejo vse informacije o izbranem procesu. To naredimo tako, da iz podatkovne baze preberemo identifikacijsko številko izbranega procesa ter jo zapišemo v trenutni spletni naslov. Na konec spletnega naslova dodamo še ime spremenljivke ter njeno vrednost. Nato spletno stran ponovno naložimo ter s pomočjo »GET« metode preberemo podatek s spletnega naslova in prikažemo vse informacije o tem procesu.

```
$id_procesa = $_GET['id'];
$querySelect = "SELECT * FROM tabela WHERE id='" . $id_procesa . "' ORDER
BY pd.date";
```

Primer 4.7: Primer delovanja metode »GET« ter iskanje vseh podatkov v zvezi s tem procesom s prebranim podatkom v podatkovnem skladišču.

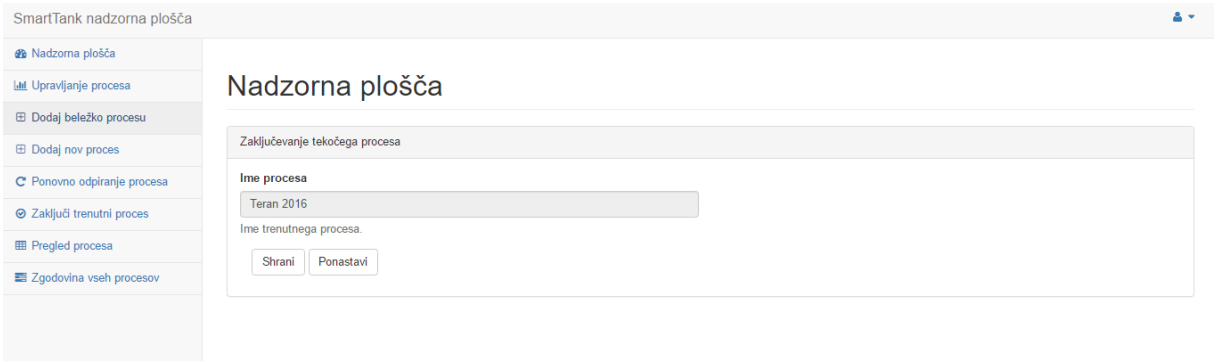
Podatke izpisujemo na časovnici. Prikazujemo jih po datumu v naraščajočem vrstnem redu. Na sredini se izrisuje črta, ki ponazarja časovni trak. Vsak sodi podatek prikazujemo na levi strani časovnice, vsak lihi podatek pa na desni strani časovnice. Da je seznam polj pregleden smo poleg informacij dodali še ikone. Ikone sive barve prikazujejo začetek ali konec procesa, vse rdeče ikone pa prikazujejo informacije, ki jih je sistem zapisal samodejno in oranžne ikone, ki prikazujejo vnesene beležke, ki jih je uporabnik vnesel sam.



Slika 4.15: Prikaz podrobnosti o procesu.

4.3.8 Zapiranje procesa

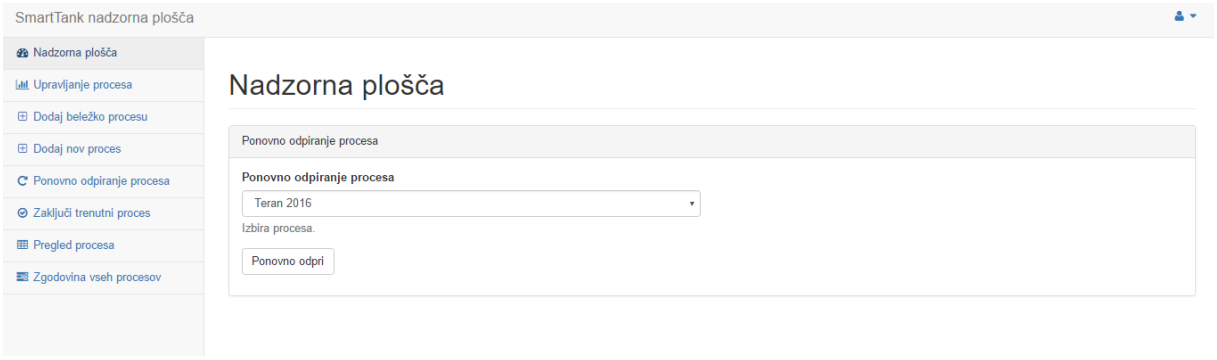
Če uporabnik želi proces zapreti, to stori s pomočjo strani za zapiranje procesa. S tem se proces zaključi in ga ni mogoče več spreminjati ali nadzirati. V času, ko se noben proces ne izvaja, so določene funkcije sistema onemogočene: dodajanje beležk, saj ni procesa, kateremu bi jo lahko zapisali in upravljanje same cisterne zaradi istega razloga.



Slika 4.16: Zaključevanje procesa.

4.3.9 Ponovno odpiranje procesa

Ker pa uporabnik lahko naredi napako in ponesreči zapre proces, mu stran za ponovno odpiranje procesa omogoči ponovno odpiranje. S tem se proces lahko ponovno upravlja. Predpogoj za uporabo te strani je le ta, da noben proces ni v izvajanju, saj v nasprotnem primeru uporabnik dobi primerno opozorilo.



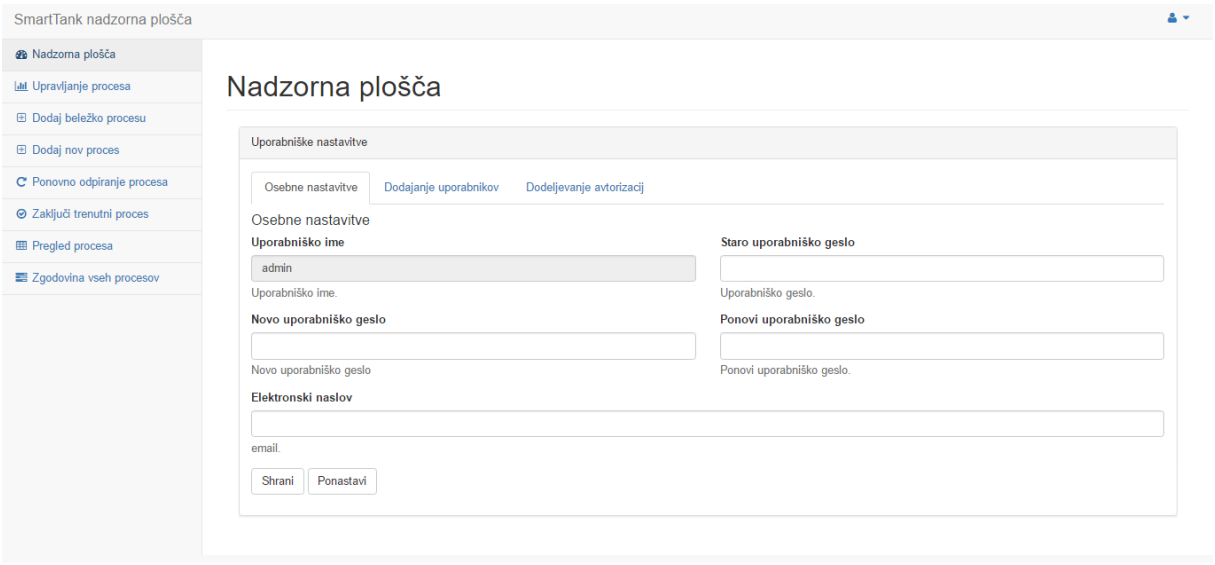
Slika 4.17: Ponovno odpiranje procesa.

4.3.10 Uporabniške nastavitve

Uporabniške nastavitve se uporablja za spreminjanje in upravljanje teh. Pogled se od uporabnika do uporabnika razlikuje v odvisnosti od avtorizacij, ki so posameznemu uporabniku dodeljene. Administrator oziroma glavni uporabnik ima vse pravice in so mu

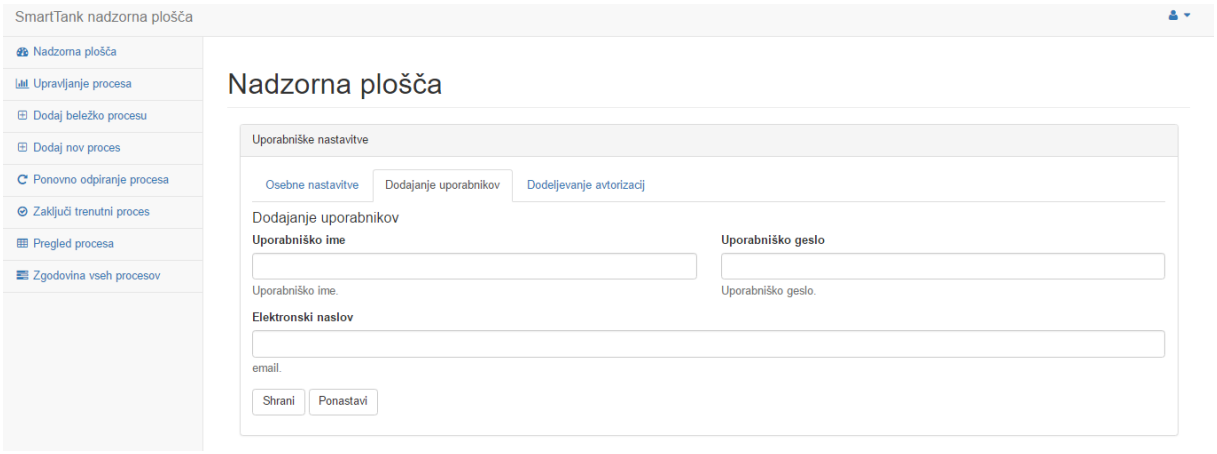
dostopni vsi podatki na spletni strani. urejevalec lahko samo dodaja beležke ter spremlja krmilni sistem, ne more ga pa upravljati, ter upravitelj, ki lahko samo upravlja krmilne sisteme.

Stran za urejanje uporabniških nastavitev najdemo v zgornjem desnem kotu, kjer se nahaja ikona z motivom uporabnika. S klikom na ikono se nam prikaže spustni seznam, kjer imamo poleg odjave še možnost »uporabniške nastavitve«.



Slika 4.18: Uporabniške nastavitve.

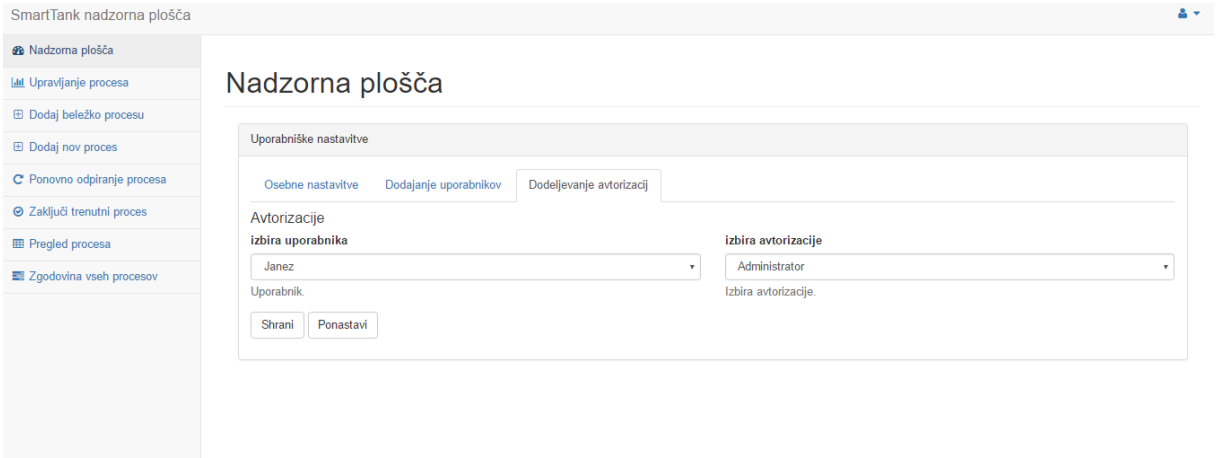
Kot je že bilo omenjeno, se pogled strani prikazuje v odvisnostni od uporabniških avtorizacij. Administratorju se prikažejo trije zavihki (Slika 4.18: Uporabniške nastavitve.). Prvi zavihek je namenjen osebnim nastavitvam in se prikaže vsem uporabnikom. Namenjen je urejanju osebnih podatkov kot so sprememba gesla ter elektronskega naslova. Elektronski naslov je namenjen za komunikacijo z uporabnikom, v primeru, da pozabi svoje geslo in mu s tem omogočimo ponoven dostop do aplikacije. Drugi zavihek je »dodajanje uporabnikov«. To možnost vidi samo administrator in samo on lahko dodaja nove uporabnike. Novemu uporabniku dodeli uporabniško ime in geslo ter elektronski naslov.



Slika 4.19: Dodajanje novega uporabnika.

Zadnji zavihek nam ponuja dodeljevanje uporabniških pravic posameznim uporabnikom. Administrator mora samo izbrati ustrezno osebo ter tej osebi dodeliti pravice. Na voljo ima tri možnosti:

- administrator,
- urejevalec,
- upravitelj.



Slika 4.20: Dodajanje avtorizacij uporabniku.

4.4 Krmiljenje naprav

V sistemu je en krmilnik, ki je namenjen krmiljenju naprav. Naprava ima šest analognih ter štirinajst vhodno-izhodnih priključkov. Spremljamo lahko trenutno temperaturo mošta v cisterni in nivo ogljikovega monoksida v ozračju ter nivo gladine v cisterni. Poleg tega pa z njim upravljamo še hladilno-grelni in mešalni sistem.

V nadaljevanju bomo opisali še kako aplikacija deluje ter kako krmilimo in spremljamo stanja naprav.

4.4.1 Krmilna aplikacija

Krmilna aplikacija je shranjena na 32 MB velikem pomnilniku, ki se nahaja na Arduino razvojni ploščici. Aplikacija je sestavljena iz dveh glavnih ter štirih dodatnih funkcij. Glavi funkciji, ki sta nujno potrebni za delovanje Arduino aplikacije sta funkcija »Setup in funkcija »Void«. Poleg tega smo dodali še funkcije, s katerimi smo dopolnili funkcionalnost aplikacije.

Funkcija »receiveEvent« sprejme podatke od strežniške aplikacije, kar se izvede takoj, ko strežniška aplikacija pošlje signal za sprejemanje podatkov. Podatke beremo enega za drugim, v bajtnem zapisu. Vse prebrane podatke shranimo v tabelo, ki je dinamične velikosti. Nato izvedemo funkcijo »chooseAction«. Bajtni zapis pretvorimo v znakovnega in ga hranimo v tabeli prebranih podatkov. Prvi znak v tabeli predstavlja akcijo, ki jo strežniška aplikacija zahteva od krmilne. Na podlagi tega torej ve katere podatke mora strežniški aplikaciji posredovati nazaj. To naredi s pomočjo funkcije »requestEvent«, ki potrebne podatke najprej pretvori v znakovni zapis s pomočjo funkcije »ConvFloatToChar« ter v bajtnem zapisu vrne podatke. Prvi bajt v nizu predstavlja dolžino celotnega sporočila, ki ga bo sprejela strežniška aplikacija.

4.4.2 Merjenje temperature

Za spremljanje temperature uporabljamo temperaturni senzor. oziroma senzor z oznako DS18B20. Komunikacijo med krmilnikom in senzorjem smo izvedli z 1-Wire Bus protokolom, ki poteka prek enega vhoda. Uporabili smo knjižnico OneWire. Senzor omogoča branje temperature vsako sekundo.

1-Wire Bus protokol uporablja definicijo enega podatkovnega toka, kar pomeni, da lahko na isti podatkovni tok priključimo več istih senzorjev. Vsi senzorji imajo različne MAC naslove, s čimer lahko identificiramo vsak senzor.

Glavna funkcija senzorja je ta, da samodejno pretvori analogno vrednost v digitalno. Sam senzor omogoča resolucijo od 9 do 12 bitov, kar pomeni, da pri resoluciji 9 bitov senzor prikazuje temperaturno spremembo na vsake 0,5°C, pri resoluciji 12 bitov pa spremembo na 0,0625°C. Senzor hrani temperaturni podatek v temperaturnem registru. Velikost registra je 16 bitov. Prvi 4 biti predstavljajo signalne bite. V primeru, da je temperatura negativna imajo vsi signalni biti vrednost 1, sicer pa je vrednost 0. Če imamo nastavljeno resolucijo 12 bitov pomeni, da bodo vsi ostali biti v registru imeli svojo vrednost. Če pa izberemo manjšo resolucijo, kot je recimo resolucija 10 bitov, bosta prva 2 bita nedoločena [19].

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
BIT 15	**BIT 14**	**BIT 13**	**BIT 12**	**BIT 11**	**BIT 10**	**BIT 9**	**BIT 8**
S	S	S	S	S	2^6	2^5	2^4

Slika 4.21: Podatek shranjen v registru senzorja.

4.4.3 Nadzor gladine

Za nadzor spremljanja nivoja gladine v cisterni uporabljamo ultrazvočni senzor oznake HC-SR04. Meritve izvajamo neprekinjeno in s tem zagotovimo popolni nadzor nad gladino v cisterni.

Senzor uporablja ultrazvočne valove, ki jih pošlje v določeno smer. V istem času se sproži štoparica. Ultrazvočni val deluje tako, da se razprši po zraku, ob prvi oviri pa se žarek odbije nazaj proti senzorju. Hitrost samega vala je 340 m/s. Funkcija za izračun razdalje je:

$$s = 340 * \frac{t}{2} \tag{4.1}$$

V enačbi »s« predstavlja spremenljivko za razdaljo, vrednost 340 je konstanta, ki predstavlja hitrost ultrazvočnega vala, »t« pa označuje pretečen čas od poslanega do povratnega ultrazvočnega vala [22].

V aplikaciji imamo shranjeno konstanto »MAX_DISTANCE«, ki označuje najvišjo točko, ki jo gladina mošta lahko doseže. V primeru, da je izmerjena razdalja višja od dovoljene podatek v spremenljivki »warnDist« povečamo za 1. Če »warnDist« doseže vrednost 10 to pomeni, da je bilo vseh deset zadnjih meritev višjih od željne. Na podlagi tega podatka obvestimo strežniško aplikacijo o nenadzorovanem dvigu nivoja gladine v cisterni. V nasprotnem primeru spremenljivko »warnDist« postavimo nazaj na začetno vrednost, saj se včasih lahko zgodi, da je senzor nenatančen.

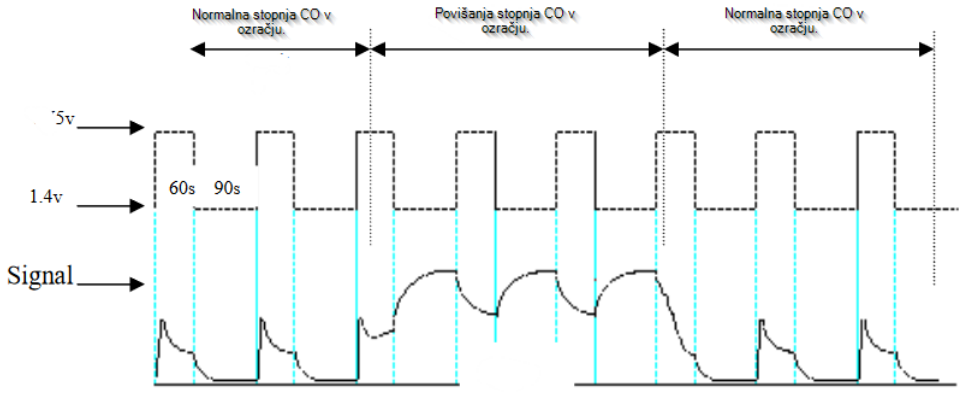
```
dur = pulseIn(LedPin6, HIGH);
dist = (dur/2)
if (dist < MAX_DISTANCE) {
    warnDist++;
}else{
  if(warnDist!=0){
      warnDist=0;
  }
}
delay(500);
```

Primer 4.8: Primer merjenja nivoja gladine.

4.5 Spremljanje stopnje koncentracije ogljikovega monoksida v kleti

Za spremljanje koncentracije ogljikovega monoksida v ozračju smo uporabili senzor MQ-7. Merjenje koncentracije ogljikovega monoksida se izvaja neprekinjeno.

Senzor MQ-7 deluje tako, da ciklično izmenjuje visoko (do 5 V) in nizko napetost (do 1,5 V). S tem povzroči segrevanje grelca znotraj senzorja MQ-7. Pri visoki napetosti se grelec segreje na višjo temperaturo, pri nizki pa se ohladi na nižjo temperaturo.



Slika 4.22: Menjenje koncentracije ogljikovega monoksida v ozračju.

Sistem smo implementirali tako, da ciklično izmenjuje visoko in nizko napetost. V intervalu dolgem 60 sekund, senzor napajmo s 5 V napetostjo. V tem času se senzor »očisti« vseh zajetih plinov, absorbiranih v ciklu nizke temperature. Nato pa v ciklu dolgem 90 sekund napajamo senzor z 1,5 V napetostjo. V tem času poteka zajemanje plinov. V primeru prisotnosti ogljikovega monoksida, se prevodnost senzorja poveča .

Poglavje 5 Sklepne ugotovitve

V sklopu diplomske naloge smo uspešno izdelali sistem za upravljanje vinske cisterne. Sistem je sestavljen iz strežnika in Arduino krmilnika, na katerega so priključeni trije senzorji. Poleg senzorjev so še tri rele stikala, ki omogočajo upravljanje različnih sistemov. Naredili smo tudi spletno aplikacijo, ki služi kot uporabniški vmesnik in je edini komunikacijski vmesnik med uporabnikom in sistemom. Aplikacija je dostopna vsem napravam, ki jim je omogočena prijava na svetovni splet ter vsebujejo brskalnik. Sistem je bil narejen za upravljanje manjših cistern.

Med samim razvojem smo naleteli na veliko težav. Prva težava se je pojavila na začetku ob nameščanju operacijskega sistema na Raspberry. Problem je bil ta, da operacijski sistem Raspbian nastavi privzeto velikost korenskega imenika enako desetim odstotkom celotnega prostora, kar pomeni, da smo hitro zapolnili dani pomnilnik. Težavo smo rešili tako, da smo odstranili celotno particijo ter ustvarili novo, z maksimalno velikostjo, ki jo ponuja pomnilnik. Naslednja težava se je pojavila pri prenosu podatkov med strežnikom in krmilnikom. Najprej smo uporabljali USB protokol za izmenjavo podatkov, a smo hitro ugotovili, da je to napačna izbira, saj se je podatek, ki smo ga poslali iz izvora bistveno razlikoval od podatka, ki smo ga dobili na ponoru. Težavo smo rešili z implementacijo protokola I2C. Največjo težavo pa je predstavljal senzor za zaznavanje CO plina, saj nam je senzor ves čas prikazoval napačne podatke. Kasneje smo ugotovili, da senzor ne deluje pravilno in so podatki posledično bili napačni. Kljub temu smo imeli nekaj sreče in pravočasno dobili novega, ki smo ga uspešno implementirali in tudi testirali.

V sistem je mogoče vpeljati še veliko nadgradenj, prav tako bi vanj lahko vključili še več drugih senzorjev. Eden takih je senzor za merjenje koncentracije sladkorja v moštu. V sistem bi vključil še video nadzor, ki bi uporabniku omogočil direkten pogled na določeno cisterno. Trenutno je do sistema mogoče dostopati samo prek naprave, zato bi bilo potrebno na objekt namestiti še zaslon, saj bi v primeru, da bi se internetna povezava prekinila lahko še vedno dostopalo do sistema prek zaslona. V sistem bi dodali še »čarovnika«, ki bi uporabniku pomagal pri izbiri načina za določeno vrsto vina ter mu svetoval pri nastavitvah sistema. Za večje cisterne bi bilo potrebno zamenjati določene naprave. Rele, ki ga uporabljamo za krmilni sistem manjše cisterne ni primeren za večje, saj so rotacijski motorji večji in močnejši. Glavni del, kjer je možna nadgradnja pa je hladilno-grelni sistem. Sistem bi nadgradil tako, da bi ta sam predvidel kdaj se mora vključiti ali izključiti. Sistem mora torej predvideti kdaj hladilno-grelni sistem izključiti, da se željena temperatura ohrani.

Sistem je pripravljen na začetna testiranja s pravo cisterno. Predvidevamo, da bo testiranje trajalo vsaj toliko časa, kot je trajal celoten razvoj sistema. Za celoten projekt smo porabili približno 150 evrov, od tega smo 80 evrov porabili za Raspberry Pi in Arduino. Sam razvoj je trajal približno 6 mesecev.

Kljub vsem nadgradnjam, ki so še potrebne vseeno menimo, da nam je uspelo ustvariti kvaliteten sistem s cenovno ugodnimi napravami in senzorji, ki omogoča enostavno upravljanje ter ga lahko z manjšimi spremembami prilagodimo vsaki cisterni.

Viri in literatura

[1] Alkoholno vretja mošta. *Enolyse* [Online]: Dosegljivo http://enolyse.com/sl/2015/05/18/alkoholno-vrenje-belih-mostov-priloznosti-in-tveganja/.

[2] Adrian McEwen, Hakim Cassimally, Designing the Internet Of Things, Chichester: John Wiley and Sons, Ltd., 2014.

[3] Ovidiu Vermesan, Peter Friess, *Internet of Things - From Research and Innovation to Market Deployment,* Algade: River Publishers, 2014.

[4] Povzetek Interneta Stvari, *Internet society* [Online] Dosegljivo: *https://www.internetsociety.org/sites/default/files/ISOC-IoT-Overview-20151014_0.pdf.*

[5] Prihodnost interneta stvari, *The Internet of Things How the Next Evolution of the Internet Is Changing Everything,* Cisco [Online] Dosegljivo: *https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf.*

[6] Osnovno o internetu. *Wikipedia* [Online] Dosegljivo: *https://en.wikipedia.org/wiki/Internet.*

[7] Ovidiu Vermesan, Peter Friess, *Internet of things-Converging Technologies for Smart Environments and Integrated Ecosystems*, Algade: River Publishers, 2013.

[8] Internet stvari. *Datafloq* [Online] Dosegljivo: *https://datafloq.com/read/where-does-the-internet-of-things-come-from/524.*

[9] Raspbian operacijski sistem. *Wikipedia* [Online] Dosegljivo: *https://en.wikipedia.org/wiki/Raspbian.*

[10] Delovanje rele stikal. *Wikipedia* [Online] Dosegljivo: *https://sl.wikipedia.org/wiki/Rele.*

[11] Osnovno o Arduino ploščici. *Wikipedia* [Online] Dosegljivo: *https://en.wikipedia.org/wiki/Arduino.*

[12] Protokol I2C. *Revija svet elektronike* [Online] Dosegljivo: *http://www.svet-el.si/o-reviji/samogradnje/835-i2c-monitor.*

[13] Raspberry Pi kartični računalnik. *Wikipedia* [Online] Dosegljivo: *https://en.wikipedia.org/wiki/Raspberry_Pi.*

[14] Senzor MQ-7. *Sparkfun* [Online] Dosegljivo: *https://cdn.sparkfun.com/datasheets/Sensors/Biometric/MQ-7%20Ver1.3%20-%20Manual.pdf.*

[15] Temperaturni senzor. *Cactus* [Online] Dosegljivo: *http://cactus.io/hookups/sensors/temperature-humidity/ds18b20/hookup-multiple-ds18b20-temp-sensor-arduino-pins.*

[16] Arduino razvojno okolje. *Wikipedia* [Online] Dosegljivo: *https://en.wikipedia.org/wiki/Arduino#Software.*

[17] IDE-Ninja razvojno okolje. *Wikipedia* [Online] Dosegljivo: *https://en.wikipedia.org/wiki/Ninja-IDE.*

[18] JetBrains PhpStorm razvojno okolje. *Wikipedia* [Online] Dosegljivo: *https://en.wikipedia.org/wiki/PhpStorm.*

[19] Implementacija temperaturnega senzorja. *Adafruit* [Online] Dosegljivo: *https://cdn-shop.adafruit.com/datasheets/DS18B20.pdf.*

[20] Protokol I2C. *Robot Electronics* [Online] Dosegljivo: *http://www.robot-electronics.co.uk/i2c-tutorial.*

[21] Podrobno o podatkovni bazi. *Wikipedia* [Online] Dosegljivo: *https://en.wikipedia.org/wiki/Database.*

[22] Ultrazvočni senzor. *Micropik* [Online] Dosegljivo: *http://www.micropik.com/PDF/HCSR04.pdf.*

[23] Rolf H. Weber, Romana Weber, *Internet of Things Legal Perspectives*, Zurich-Basel-Geneva: Schulthess Juristische Medien AG , 2010.

[24] Feng Xia, Laurence T. Yang, Lizhe Wang, Alexey Vinel, Internet stvari [Online] Dosegljivo: *https://www.homeworkmarket.com/sites/default/files/q5/04/07/danainfo.acppwiszgmk2n0u279qu76contentserver.pdf*