Izgradnja makete staklenika i izrada pripadne programske podrške za nadzor i upravljanje

Pervan, Zvonimir

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

IZGRADNJA MAKETE STAKLENIKA I IZRADA PRIPADNE PROGRAMSKE PODRŠKE ZA NADZOR I UPRAVLJANJE

Diplomski rad

Zvonimir Pervan

Osijek, 2017.

SADRŽAJ

1.UVOD

U današnjem, modernom, okruženju, nezamislivo je koračati po svijetu bez **informacije**. Svaki čovjek želi unaprijed znati kakvo će vrijeme biti, najnovije vijesti iz svijeta, otkucaje srca prilikom treninga, kvalitetu zraka u stambenom prostoru itd. Čovjek ima neutažive apetite za brzom informacijom o svemu, a u tome svemu je neizostavan faktor interneta koji omogućava taj brzi prijenos informacije. Internet se usko povezao sa svim područjima znanosti i djelatnosti, te je tako povezan i sa automatizacijom. Naime, automatika, osim što rasterećuje čovjeka od repetitivnih zadataka, daje mu uvid u stanje (pomoću **senzora**) nekog procesa ili postrojenja te na temelju mjernih podataka i programirane logike unutar mikroračunala izravno se utječe na proces ili parametre postrojenja (pomoću **aktuatora**). Zbog još veće funkcionalnosti, povezana su ta dva elementa te je prikazan jedan od primjera u ovom diplomskom radu.

Tema rada je izgradnja **makete staklenika** i pripadne programske podrške. Maketa staklenika predstavlja sustav u kojem je omogućena sadnja raznih poljoprivrednih kultura, pri čemu svaka kultura zahtijeva određene uvjete u kojima će rasti.

Izrada mekete staklenika sastoji se od izrade odgovarajućeg hardvera koja uključuje upravljačko mikroračunalo, senzore i aktuatore, te prikladna programska podrška koja obuhvaća nadzor i upravljanje parametrima unutar makete preko *SCADA*-e[1] (engl. *Supervisory Controla And Data Acquisition*), a nalazi se nalazi na web poslužitelju koji je ostvaren pomoću dodatnog mikroračunalnog sustava.

Diplomski rad podijeljen je na 5 poglavlja. **Drugo** poglavlje opisuje stakleničkI uzgoj biljaka, njene prednosti i bitne stavke oko stakleničkog uzgoja. U **trećem** poglavlju opisan je postupak projektiranja i izgradnje makete staklenika, tj. fizička izvedba makete, korištena mikroračunala za upravljanje i nadzor, senzori za mjerenje odgovarajućih fizikalnih veličina te aktuatori za utjecaj istih parametara. U **četvrtom** poglavlju opisan je postupak izrade programske podrške upravljačkog i nadzornog sustava staklenika, tj. implementacija električnih komponenata navedenih u drugom poglavlju, logika rada upravljačkog sustava, postupak prikupljanja podataka sa ugrađenih senzora te izvedba

[1] ***SCADA*** – računalni sustav za nadzor, mjerenje i upravljanje nekog sustava.

nadzornog sustava uključujući uspostavu web poslužitelja, *MySQL* baze podataka i web aplikacije. U **petom** poglavlju opisan je postupak pokretanja i rad sustava.

2. STAKLENIČKI UZGOJ BILJAKA

Staklenik predstavlja objekt za kontrolirani i zaštićeni uzgoj biljaka pri čemu su stjenke staklenika načinjene od prozirnog materijala. Staklenik se, najčešće, izrađuje od stakla, iako se u novije vrijeme koristi folija kako bi se postigao *stakleničkI efekt*[2], optimalna temperatura te zaštita od padalina i štetnika. Kontroliranjem raznih parametara, kao što je temperatura i vlaga zemlje nastoje se postići optimalni uvjeti za uzgoj biljaka. Staklenici se prvenstveno koriste za **hortikulturalnu** proizvodnju, odnosno komercijalni uzgoj kultiviranih biljaka poput voća, povrća, cvijeća itd. Također se staklenik može koristiti i u **istraživačke** svrhe te u **botaničkim** vrtovima gdje se uzgajaju biljke iz različitih krajeva svijeta (podrazumijeva uzgoj egzotičnih biljaka koje ne bi opstale u području kojem se uzgajaju). Na slici 2.1. prikazani su staklenici u raznim varijacijama [1].

Sl. 2.1. Primjer staklenika [2]

Cijela ideja staklenika je zadržavanje topline, odnosno **solarna radijacija**[3] biljaka koja prolazi kroz stjenke staklenika koju tlo upija. Tvari unutar staklenika s vremenom postaju

[2] **StakleničkI efekt** ili efekt staklenika je zagrijavanje Zemljine površine i donjih slojeva Zemljine atmosfere selektivnim propuštanjem zračenja.
[3] **Solarna radijacija** – vidljivi i susjedni dijelovi infracrvenog i ultraljubičastog spektra

toplije te odašilju infracrvenu energiju veće valne duljine. Staklo i ostali prozirni materijali, koji se koriste u izgradnji staklenika, ne propuštaju infracrveno zračenje što znači se toplina zadržava unutar staklenika. Također, bitne stavke predstavljaju **ventilacija** i **grijanje staklenika** [1].

Ventilacija predstavlja jednu od važnih komponenata za uspješan uzgoj biljaka, osobito u toplim i vlažnim klimatskim uvjetima. Glavna svrha ventilacije je održavanje temperature i vlažnosti na optimalnoj razini te osigurava strujanje zraka što sprječava stvaranje biljnih patogena koji preferiraju mirne uvjete zraka. Ventilacijom se osigurava opskrba svježim zrakom koja pogoduje fotosintezi i disanju biljke [1].

Grijanje predstavlja jednu od neizbježnih stavki za rad sa staklenicima, osobito u hladnijim klimatskim podnebljima jer treba održavati temperaturu na optimalnim vrijednostima za uzgoj pojedine biljke. Glavni problem sa grijanjem staklenika je količina topline koja je potrebna da se ugrije prostor te da se zadrži sama toplina zbog debljine stjenki staklenika, odnosno dolazi do velikog gubitka topline zbog čega nastaju veliki troškovi grijanja. Najčešće korišteno grijanje: prirodni plin i električne peći.

Također postoji mogućnost oplemenjivanje zraka sa **ugljičnim dioksidom** CO_2 kako bi se poboljšao rast biljaka [1].

2.1. Uzgoj biljaka

Temperatura treba biti unutar određenih granica za optimalan rast biljaka. Toplo-sezonsko bilje i povrće imaju najbolji rast na temperaturama između 15°C do 30°C, dok hladno-sezonskom povrću pogoduje rast između 5°C do 20 °C [3].

Svjetlost je potrebna svim biljkama, u velikim količinama, te se preporuča izloženost biljaka sunčevoj svjetlosti između 8 do 10 sati, svakog dana, za pravilan rast. Umjetni izvori svjetlosti, poput žarulja, **ne mogu** nadomjestiti Sunčevu svjetlost [3]!

Određeni **razmak** između biljaka osigurava uspješan rast biljke te primitak odgovarajuće količine svjetlosti. Primjerice, jednoj biljci rajčice potrebno je otprilike $1m^2$ za pravilan rast, krastavcima je potrebno otprilike $2m^2$ itd. Većina biljaka i povrća raste normalno na razmaku preporučanom za vrtnu sadnju [3].

Razdoblje sadnje je bitno. Biljke neće rasti jednako u zimskom i ljetnom periodu zbog temperaturnih razlika i broju sunčanih sati u određenom razdoblju [3].

Biljkama se mora osigurati dovoljna količina **vode**. Primjerice, tijekom vrućih ljetnih dana, biljka rajčice može konzumirat i do 2 litre vode dnevno i ako se korijen biljke ne drži vlažnim biljka će se osušiti [3].

Biljke zahtijevaju **kisik** za respiraciju kako bi obavljale funkcije unosa vode i nutrijenata. Kad se biljka posadi u zemlju nije potrebno naknadno oplemenjivati vodu (kao u hidropontskom sustavu) sa kisikom jer postoji stalan dotok svježe vode. Ovo je kritično u slučaju ako se biljka posadi izravno u vodu jer se s vremenom sav kisik iscrpi unutar vode te može ostaviti trajne posljedice na biljkama [3].

Biljke moraju unositi **minerale** kroz njihov korijen kako bi preživjele. Ti minerali se pružaju biljkama kroz zemlju u obliku komposta ili gnojiva. Sastojci potrebni za pravilan rast u velikim količinama su dušik, fosfor, kalij, kalcij, magnezij i sumpor. Mikronutrijenti – željezo, mangan, bor, cink, bakar, molibden i klor su isto potrebni, ali u vrlo malim količinama [3].

3. PROJEKTIRANJE I IZGRADNJA MAKETE

U ovom poglavlju dan je popis zahtjeva koju mora ispunjavat maketa staklenika, fizička izvedba samog staklenika (materijal i konstrukcija) te korištene električne komponente za obavljanje zadanih zadataka s pripadnom programskom podrškom.

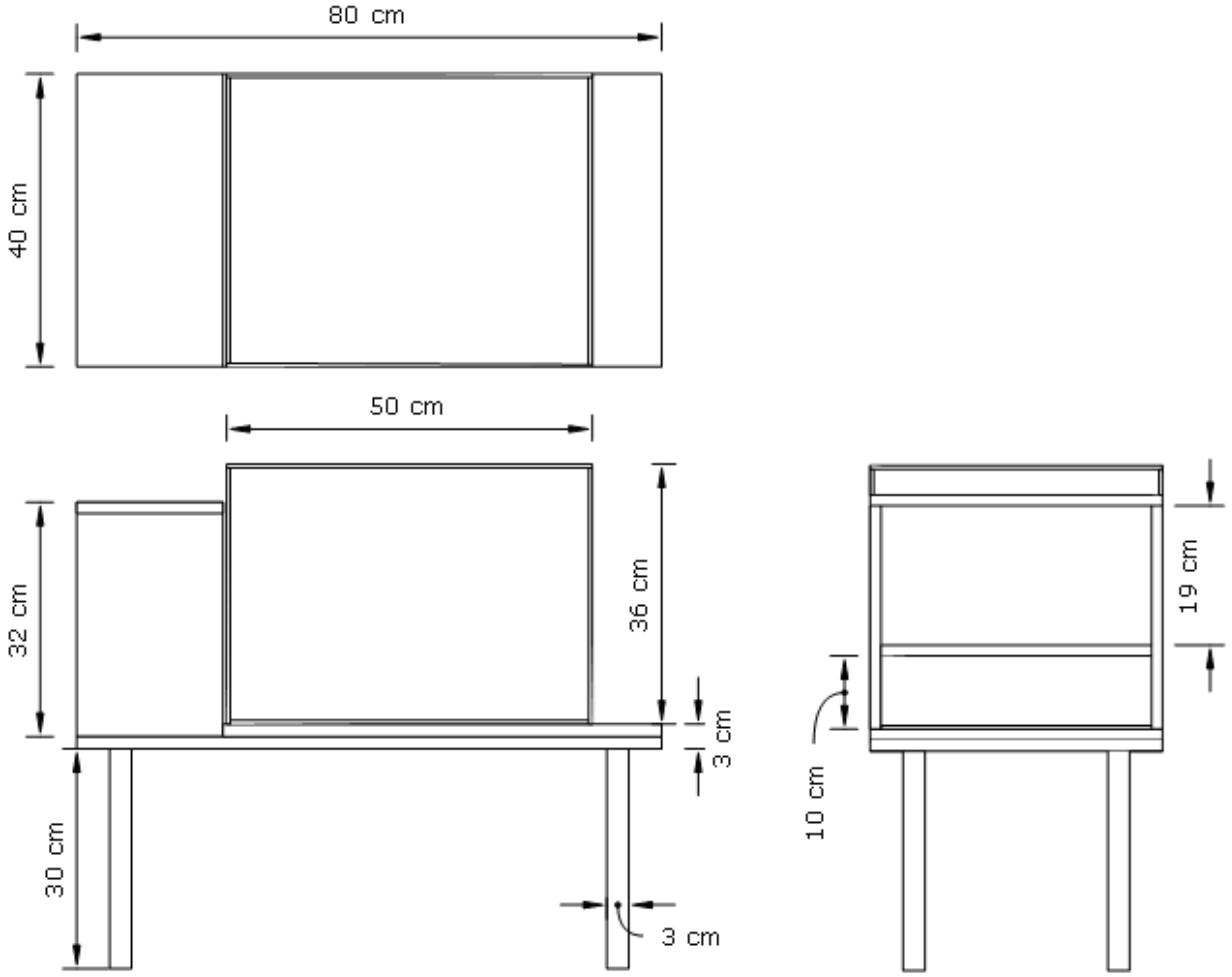
3.1. Zahtjevi

Zahtjevi koje mora ispunjavati staklenik su:

- **Zalijevanje biljaka** – vršiti nadzor vlage u zemlji pomoću senzora (higrometra) i na temelju izmjerenih vrijednosti izvršiti zalijevanje biljaka do zadane vrijednosti pri čemu se voda crpi iz spremnika koji se nalazi pored makete.
- **Upravljanje temperaturom, vlagom i CO_2 –** pomoću senzora vršiti nadzor temperature i vlažnosti zraka te količine CO_2 u zraku. Na temelju izmjerenih vrijednosti izvršiti odgovarajuću radnju (grijanje ili provjetravanje). Raspon graničnih vrijednosti temperature unutar staklenika iznosi od 19°C do 25°C, granični postotak vlažnosti zraka u rasponu od 20% do 60%, vlažnost tla u rasponu od 35% do 50% te udio CO_2 u zraku u rasponu od 5% do 10% [4].
- **Nadzor spremnika vode** – pomoću odgovarajućeg senzora pratiti razinu vode unutar spremnika. U slučaju da se dosegne kritična razina, potrebno je ugasiti pumpu te upozorit korisnika da je spremnik prazan.
- **Nadzor staklenika preko web sučelja** –napraviti *SCADA* sustav koji će prikazivat stanje unutar staklenika i povijesne podatke (npr. vrijednost temperature kroz vrijeme) te promjena parametara, temperature i vlage tla, za novo posađenu biljku.
- **Osvjetljavanje biljaka tijekom noći** – uz pomoć UV LED rasvjetne trake osvjetljavat biljke unutar staklenika kako bi rasle i u odsutnosti sunčevog svjetla. Rasvjeta se pali na temelju mjerenja dobivenog s foto-otpornika.
- **Parametriranje sustava na temelju posađene biljke** – mogućnost podešavanja referentnih vrijednosti (vlaga tla i temperatura) preko *SCADE* koje pogoduju rastu novo posađene biljke u stakleniku.

3.2. Konstrukcija makete

Za realizacija staklenika potrebno je koristiti prozirni materijal kako bi zasađene biljke mogle primati sunčevu svjetlost te zadržavati vlagu i toplinu. U tu svrhu korišten je **pleksiglas**. Na slici 3.1. prikazan je crtež makete s označenim dimenzijama.



Sl. 3.1. Dimenzije makete staklenika

Maketa staklenika postavljena je na postolje, a sastoji se od dva dijela, tj. dio gdje se sade biljke (staklenik), a drugi dio je mjesto za mikroračunala i elektroničke komponente. Na slici 3.2. prikazana je **fizička** izvedba makete. Sa slike je vidljivo da se poklopac makete može podignuti kako bi se moglo pristupit biljkama unutar staklenika. Na slici 3.3. prikazane su **brizgaljke** koje služe za navodnjavanje biljaka unutar staklenika.

Spremnik za vodu predstavlja uobičajeni spremnik za tekućine pri čemu je veličina zapremnine u iznosu od 15 litara i prikazan je na slici 5.6.

Sl. 3.2. Izgled makete

Sl. 3.3. Brizgaljke

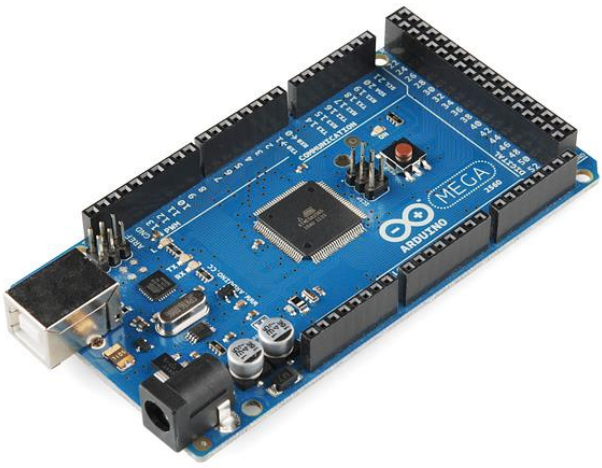
3.3. Električne komponente

U ovom poglavlju opisane su korištene električnih komponenata, tj. korištena mikroračunala, senzori, aktuatori i dodatne električne komponente nužne za ispunjavanje postavljenih zahtjeva.

3.3.1. Mikroračunalni sustav „Arduino MEGA2560“

Arduino predstavlja open-source računalo koje je 2005. godine razvijeno od tvrtke SmartProjects, Italija. „*Open-source*“ platforma omogućava korištenje biblioteka i sadržaja koji nisu napravljeni od proizvođača. Najvažnija stavka ovog mikroračunalnog sustava je mogućnost realizacije upravljačkog sustava pomoću kojeg se upravlja aktuatorima i senzorima. Na temelju izmjerenih vrijednosti dobivenih od senzora, mikroračunalni sustav će na temelju korisničke programske logike izvršit odgovarajuću radnju (npr. temperaturni senzor pokazuje temperaturu veće od zadanih parametara te će zbog toga uključiti ventilator kako bi se rashladila okolina). Arduino se sastoji od **digitalnih** i **analognih** ulazno/izlaznih pinova koji se mogu povezat sa raznim ekspanzijskim tiskanim pločicama (engl. *Shield*) i ostalim osjetilnim i aktuatorskim komponentama. Ima integrirano serijsko komunikacijsko sučelje, pri čemu uključuje USB izlaze na nekim modelima Arduina, koji služi za prijenos programskog koda u memoriju i daljnje upravljanje mikroračunalnim sustavom. Za programiranje mikroupravljača, Arduino pruža integrirano razvojno okruženje zvano ArduinoIDE (engl. *Integrated Development Enviroment*) koje će se koristiti za izradu programske podrške [5].

Za potrebe diplomskog rada korišten je „Arduino MEGA2560“ mikroračunalni sustav, prikazan na slici 3.4.



Sl 3.4. „Arduino MEGA2560“ mikroračunalni sustav [5]

U tablici 3.1 dane su tehničke i fizičke karakteristike danog mikroračunala.

Tab 3.1. Tehničke karakteristike „Arduino MEGA2560“ [5]

Arduino „MEGA 2560“	
Mikrokontroler	ATmega328P
Napajanje	5V
Ulazni Napon (preporučano)	7-12V
Ulazni Napon (limit)	6-20V
Digitalni U/I Pinovi	54 (Od čega su 15 PWM izlazi)
Analogni Ulazni Pinovi	16
DC Struja po U/I Pinu	20 mA
DC Struja za 3.3V Pin	50 mA
Flash Memorija	256 KB (ATmega328P), 8 KB bootloader
SRAM	8 KB (ATmega328P)
EEPROM	4 KB (ATmega328P)
Clock Brzina	16 MHz
Dužina	101,52 mm
Širina	53.4 mm
Težina	37 g

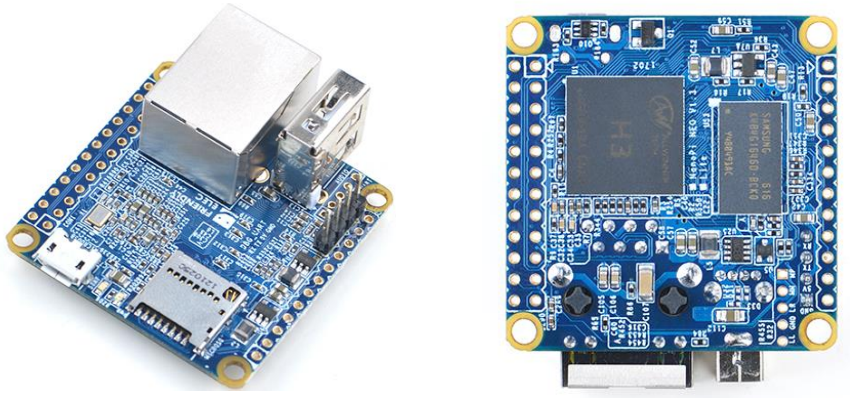
Razlog korištenja ove inačice Arduina je veća **procesorska** i **memorijska** moć zbog velikog broja zadataka koji će se izvršavati i korištenih komponenata (senzori, aktuatori i prikaznik). Arduina predstavlja upravljački sustav pomoću kojeg se obavlja upravljanje i nadzor nad temperaturom zraka, vlažnosti tla, vlažnosti zraka i količine CO_2.

3.3.2. Mikroračunalni sustav „NanoPI NEO“

NanoPi NEO je mikroračunalni *ARM*[4] sustav sa GPIO (engl. *General Purpose Input Output*) pinovima. Posjeduje Allwinner H3 Quad Core A7 procesor na brzini od 1.2 GHz, 512MB DDR3 memorije sa širinom od 32 bit-a. Tvrtka FriendlyElec napravila je

[4] **ARM** (*eng.* Advanced RISC Machine) – pripada obitelji računalnih procesora sa smanjenim brojem instrukcijskih naredbi koji je prilagođen raznovrsnim namjenama.

vlastiti nisko-potrošni Linuxoid (Ubuntu jezgra). Zbog dimenzija (40x40 mm), prilagođenog *software*-a i tehničkih specifikacija često je korišten platforma za *IoT* (engl. *Internet of Things*) projekte od strane hobista i profesionalaca [6].



Sl 3.5. NanoPI NEO [6]

Na slici 3.5. prikazan je NanoPI NEO, a u tablici 3.2. su prikazane tehničke karakteristike samog mikroračunala.

Tab 3.2. Tehničke karakteristike za „NanoPI NEO“ [6]

NanoPI NEO	
Procesor	Allwinner H3, Quad-core Cortex-A7 Up to 1.2GHz
RAM	512MB, DDR3
Brzina veze	10/100M Ethernet
USB	Type-A x 1, 3.54 mm pin x 2
MicroSD utor	1
MicroUSB	Ulaz za napajanje
Napajanje	5V/2A DC
GPIO	36 pinova (uključuje UART, SPI, IIC, IO itd.)
Dimenzije	40x40 mm
Radna temperatura	-40°C do 80°C
Operativni sustav	u-boot, Ubuntu Core

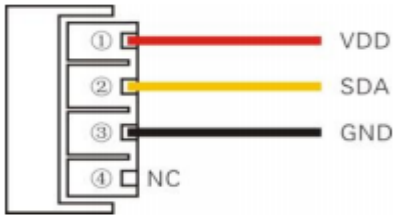
Razlog korištenja NanoPI-a je njegova mogućnost povezivanja na Internet, postavljanje web poslužitelja (baza podataka i web aplikacija) koji će pokretat nadzorni sustav te omogućit praćenje stanja staklenika preko internet preglednika.

3.3.3. Senzor za temperaturu i vlagu - DHT21

DHT21 je kalibrirani digitalni senzor za mjerenje temperature i vlage. Mikroračunalo, koje je smješteno unutar senzora, osigurava veliku pouzdanost te dugoročnu stabilnost. Na slici 3.6. prikazan je izgled DHT21 senzora [7].

Sl. 3.6. DHT21 senzor [7]

Razlog korištenja DHT21 senzora je mjerni opseg temperature i vlage (prikazane u tablici 3.3.) koja ispunjava zahtjeve opisanim u potpoglavlju 3.1. Dodatni razlozi: male dimenzije, niska potrošnja električne energije i prihvatljiva cijena. U stakleniku se koriste dva DHT21 senzora kako bi se smanjila mjerna nesigurnost. Na slici 3.7. prikazan je raspored izlaznih pinova te njihova pojašnjenja funkcija.



PIN	Boja	Ime	Pojašnjenje
1	Crvena	VDD	Napajanje (+)
2	Žuta	SDA	Podaci
3	Crna	GND	Zemlja (-)
4	-	-	Prazno

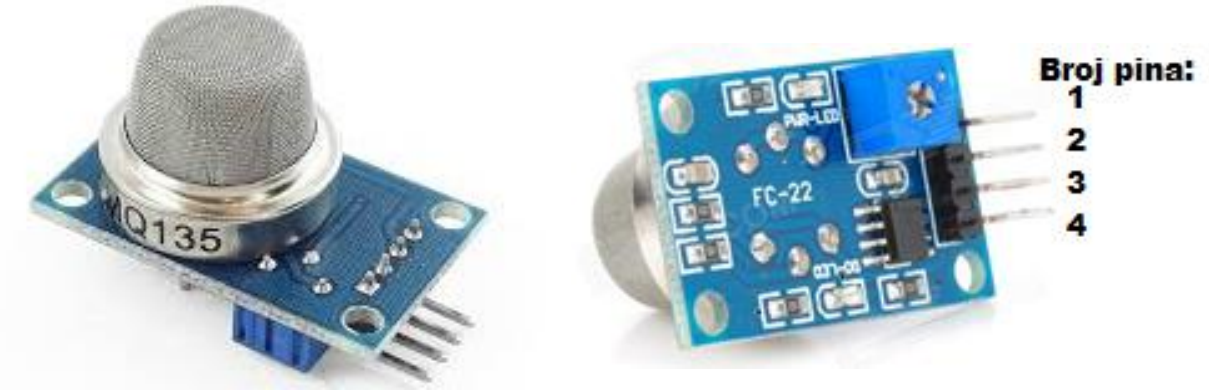
Sl. 3.7. Raspored i pojašnjenje izlaznih pinova [7]

Tab. 3.3. Tehničke karakteristike DHT 21 senzora [7]

DHT 21	
Napajanje	3.3 do 5.2 V (preporučano 5V)
Mjerni opseg	*Vlaga*: 0 do 100 % RH (engl. *Relative humidity*) *Temperatura*: -40°C do + 80°C
Točnost	*Vlaga*: ±3% RH (Max: ±5% RH) *Temperatura*: < ±1°C
Rezolucija	*Vlaga*: 0.1% RH *Temperatura*: 0.1°C
Ponovljivost	*Vlaga*: ±1% RH *Temperatura*: < ±0,2°C

3.3.4. Senzor za mjerenje koncentracije plinova u zraku – MQ135

MQ 135 sa FC22 pločicom je **senzor za mjerenje koncentracije plinova u zraku** poput ugljikovog dioksida (CO_2), benzinskih para, alkoholnih para, NOx, NH3 itd. Senzor se sastoji od čelične mrežice ispod koje je smješten senzor. Senzor se zagrijava i pri kontaktu sa plinom dolazi do ionizacije unutar mrežice pri čemu se mijenja otpor unutar senzora i u konačnici vrijednost izlazne struje. Vrijednost otpora je drugačija za različite plinove te se preporuča podešavanje vrijednost otpora potenciometra smještenog na FC22 pločici. Na slici 3.8. prikazan je MQ135 senzor na FC 22 pločici [8].



Sl 3.8. MQ135 sa FC-22 pločicom [9]

Razlog korištenja ovog senzora je mjerenje **kvalitete zraka** odnosno količinu CO_2 unutar staklenika. Zrak se s vremenom ustali u zatvorenom prostoru te može doći do

razvoja patogenih bakterija (pogledaj potpoglavlje 2.1. i potpoglavlje 3.1.) i na temelju izmjerenih vrijednosti MQ135 senzora izvršava provjetravanje staklenika (nastoji se održavat što manja vrijednost količine CO_2 unutar staklenika). U tablici 3.4. prikazan je raspored pinova.

Tab. 3.4. Raspored pinova [9]

PIN	Ime	Značenje
1	Vcc	Napajanje, +5VDC
2	GND	Uzemljenje
3	DO	Digitalni izlaz
4	AO	Analogni izlaz

3.3.5. Ultrazvučni senzor HC-SR04

HC-SR04 je **ultrazvučni senzor** pomoću kojeg je moguće mjeriti udaljenost objekata. Pruža mogućnost beskontaktnog mjerenja udaljenosti pomoću ultrazvučnih odašiljača, prijemnika te upravljačkog sklopa. Na slici 3.9. prikazan je izgled navedenog senzora [10].

Sl 3.9. HC-SR04 [10]

U tablici 3.6. prikazano su značenje pinova na senzoru.

Tab. 3.6. Značenje pinova [10]

Oznaka	Značenje
V_{CC}	Napajanje, +5V
Trig	Okidački impuls, ulaz
Echo	Impuls odjeka, izlaz
Gnd	Uzemljenje

Osnovni princip rada [10]:

1) Koristi U/I okidačke signale za najmanje 10us,
2) Senzor šalje osam 40kHZ signala te detektira postoji li povratni signal,
3) Ako se signal vrati, tada se mjeri vrijeme od okidanja do povrata signala.

Razlog korištenja ovog senzora je mogućnost mjerenje **razine vode** unutar spremnika vode čije dimenzije ne prelaze maksimalni mjerni doseg senzora (pogledaj tablicu 3.7.).

Tab. 3.7. Tehničke karakteristike HC-SR04 [10]

HC-SR04	
Radni napon	5VDC
Potrošnja struje	15mA
Radna frekvencija	40Hz
Max doseg	400cm
Min doseg	2cm
Kut mjerenja	45°

3.3.6. LC prikaznik 16x02

Prikaznik s tekućim kristalima (engl. *LCD - Liquid Crystal Display*) je prikaznik namijenjen za ugradbene računalne sustave pomoću kojeg se mogu ispisat poruke, stanja unutar sustava i vrijednosti senzora. Izgled prikaznika i redoslijed pinova dan je na slici 3.10., a svrha pojedinog pina dano je u prilogu 3.3.1 [11].

Sl. 3.10. LCD 1602 [12]

Razlog korištenja ovog prikaznika je njegova mogućnost prikaza 16 znakova u dva reda (vidi tablicu 3.8.) što je dovoljno za prikaz imena izmjerenog parametra i vrijednosti izmjerene unutar staklenika te prikaz stanja unutar sustava.

Tab. 3.8. Tehničke karakteristike prikaznika [11]

LCD1602	
Izgled prikaznika	16 znakova, 2 reda
Vrsta prikaznika	Pozitivno, transreflektivno
Radna temperatura	Sobna temperatura, ~21°C
Napajanje	+5V
Vrsta mikroprocesora	COB (Chip On Board)
Znakovi	ASCII

3.3.7. Senzor za vlagu YL-38

YL-38 je modul pomoću kojeg je moguće mjeriti količinu vlage unutar zemlje. Izgled modula prikazan je na slici 3.11.

Sl. 3.11. YL-38 modul

Modul je dizajniran tako da su na jednoj strani smješteni pinovi za spajanje senzora, a druga strana sadrži pinove za napajanje te analogne i digitalne priključke koji šalju izmjerenu vrijednost sa senzora prema mikroračunalu. Na slici 3.11. prikazan je raspored i značenje pinova [13].

Razlog korištenja ovog modula je mogućnost mjerenja **vlažnosti zemlje** te na temelju izmjerenih vrijednosti vrši se navodnjavanje zemlje unutar staklenika (vidi potpoglavlje 3.1.). Koriste se dva YL-38 modula kako bi se smanjila mjerna nesigurnost.

3.3.8. Relejni modul SRD-05VDC-SL-C

Relej predstavlja jednu vrstu prekidača upravljano pomoću zakona **elektromagnetizma**. Relej se koristi za prekidanje strujnog kruga. Modul SRD-05VDC-SL-C je relejni modul koji se sastoji od releja izdržljivosti do 10A što ga čini idealnim u strujnim krugovima s malim vrijednostima struje. Na slici 3.12. prikazan je navedeni modul sa označenim pinovima i stezaljkama. Na slici se također vide tehničke karakteristike releja [14].

Sl. 3.12. Relejni modul sa relejnom sklopkom SRD-05VDC-SL-C

Razlog korištenja ovog modula je **uključivanje/isključivanje** grijača zraka (sušilo za kosu) na temelju izmjerenih temperaturnih vrijednosti unutar staklenika. Sam grijač ima potrošnju struje od 2.5A na 220V AC te je unutar dozvoljenih granica radnog područja releja. Također je na modulu ugrađen **optospržnik** koji galvanski odvaja Arduino od samog releja te tako smanjuje utjecaj elektromagnetskih smetnji koje nastaju kod uključivanja/isključivanja grijača.

3.3.9. Pumpa za navodnjavanje

Pumpa predstavlja jednu od bitnijih dijelova sustava jer je pomoću nje omogućeno zalijevanje biljaka unutar staklenika. Za potrebe ovog rada korištena je utopna pumpa s DC motorom (prikazana na slici 3.13.) jer omogućava navodnjavanje zemlje pri čemu je snaga pumpe (vidi tablicu 3.11.) dovoljna pošto se navodnjavaju male površine zemlje.

.**Sl. 3.13.** Utopna pumpa [15]

Tab. 3.11. Tehničke karakteristike pumpe [15]

Utopna pumpa	
Radni napon	DC 12V
Potrošnja struje	400mA
Max. protok	240L/H
Max. visina dizanja vode	3m
Potrošnja energije	3.6W

3.3.10. Elektro-ventil

Za manipulaciju protoka vode unutar cijevi potreban je **elektro-ventil** koji je prikazan na slici 3.13. Pomoću ovog modula moguće je utjecati na protok vode unutar cijevi.

Sl. 3.13. Elektro-ventil [16]

Osim upravljanja protokom vode unutar cijevi, jedan od razlog korištenja elektro-ventila je **nabijanje pritiska** unutar cijevi, u suprotnom će mlaz vode iz krajnjih brizgaljki biti slab što rezultira nejednakim zalijevanjem zemlje.

3.3.11. UV LED rasvjeta

Za učinkovitiji rast biljaka tijekom noći, koristi se **UV LED rasvjeta**, koja je prikazana na slici 3.14.

Razlog korištenja UV rasvjete je poboljšanje **kvalitete** rasta biljke unutar staklenika u odsutnosti sunčeve svjetlosti, tj. ubrzava postupak **klijanja** biljaka, povećava **otpornost** biljke na bolesti te uništavanje štetnih mikroorganizama unutar samog staklenika [17].

Ova rasvjeta je **vodotoporna** što ju čini idealnim za korištenje unutar staklenika koji je izložen velikoj količini vlage (vidi tablica 3.13.).

Sl. 3.14. UV LED rasvjeta [18]

Tab. 3.13. Tehničke karakteristike [18]

UV LED rasvjeta	
Radni napon	DC 12V
Dužina trake	45cm
Potrošnja struje	350 mA
Namjena	Unutrašnjost
Valna duljina	395 – 405nm
Odlike	Vodootporno

3.3.12. Ventilatori

Kako bi unutar staklenika postojao konstantan dotok svježeg zraka te smanjila temperatura koriste se **ventilatori** pogonjeni istosmjernim motorom. Na slici 3.15. prikazani je takav jedan ventilator. Na poklopcu staklenika (vidljivo na slici 5.6.) postavljena su dva ventilatora pri čemu jedan **upuhuje**, a drugi **izvlači** zrak što stvara bržu izmjenu sastava zraka unutar staklenika.

Sl. 3.15. DC Ventilator [19]

Zbog male zapremnine staklenika, dimenzije ventilatora (vidi tablicu 3.14.) odgovaraju postavljenim zahtjevima (vidi potpoglavlje 3.1.), tj. u relativno kratkom vremenskom razdoblju smanjuju količinu CO_2 unutar staklenika.

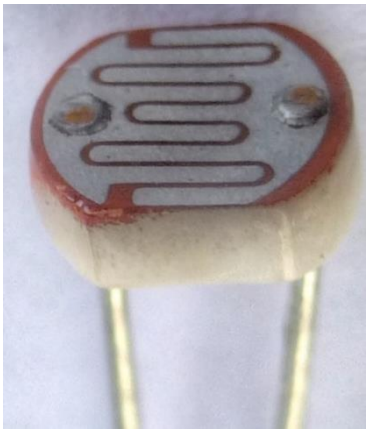
Tab. 3.14. Tehničke karakteristike [19]

DC ventilator	
Radni napon	DC 12V
Potrošnja struje	250mA
Namjena	Unutrašnjost
Veličina	120mm x 25mm
Priključak	2 pina
Akceleracija	981m/s^2

3.3.13. Foto-otpornik

Foto-otpornik predstavlja jednu vrstu otpornika koja na temelju intenziteta svjetlosti (koja pada na otpornik) mijenja vrijednost svog otpora. U odsutnosti svjetla u prostoriji (mraku)

vrijednost otpora foto-otpornika se povećava do nekoliko MΩ, dok je u prisutnosti svjetla vrijednost do par stotina Ω. Na slici 3.16. prikazan je takav jedan otpornik. [20]



Sl. 3.16. Foto-otpornik [20]

Razlog korištenja foto-otpornika je mjerenje trenutne razine svjetlosti unutar staklenika kako bi se, u slučaju niske razine ili odsutnosti svjetla, upalila UV LED rasvjeta koja pospješuje rast i tijekom noći (vidi potpoglavlje 3.1.).

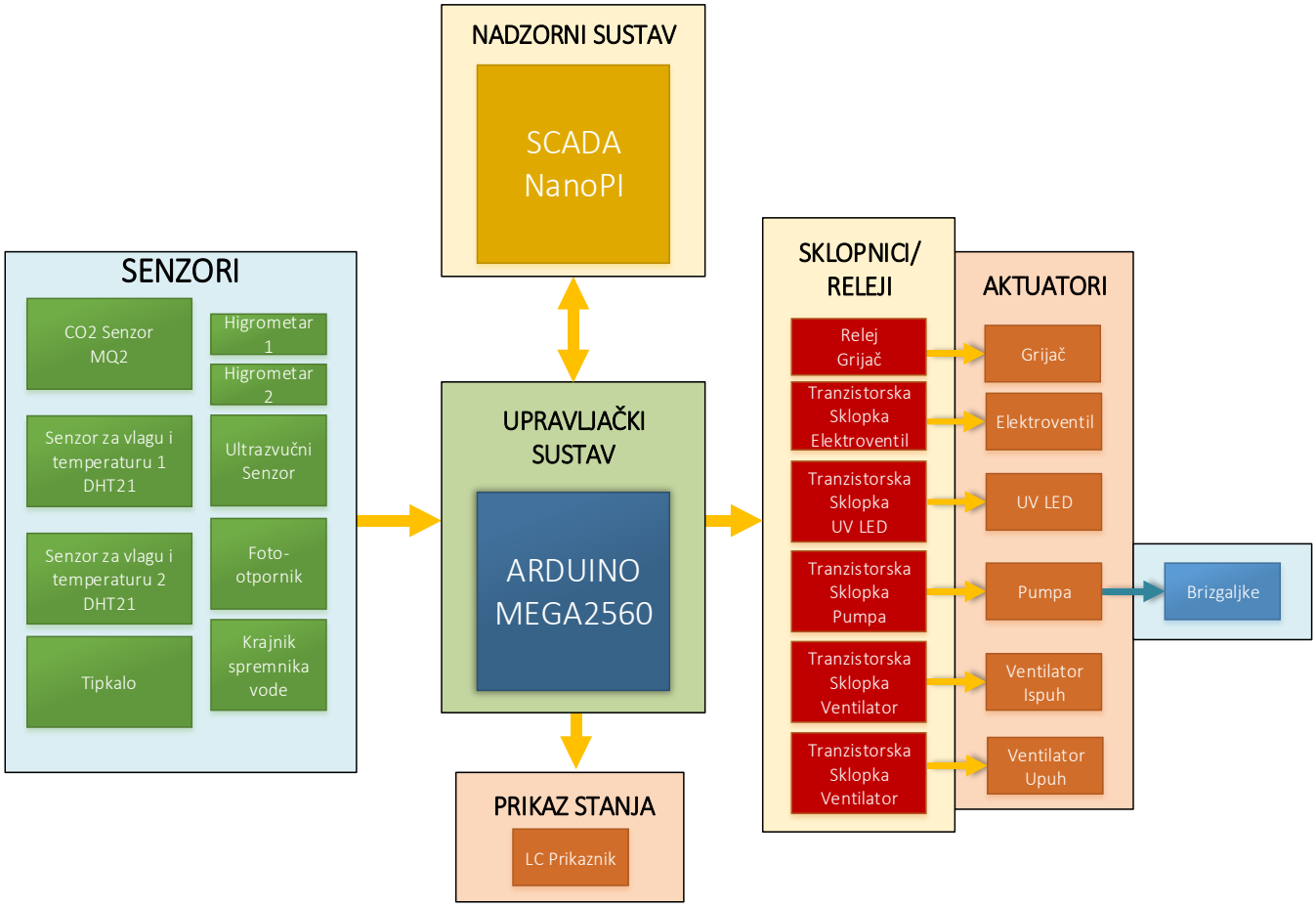
3.3.14. Grijač

Za **grijač** zraka, unutar staklenika, koristi se staro **sušilo za kosu** koje je prikazano na slici 3.17. Grijač, za svoj rad, koristi mrežni napon, tj. 230V AC pri čemu se postiže najveća potrošnja struje do 2.5A. Grijač se uključuje pomoću releja (vidi potpoglavlje 3.3.8.) koji je projektiran za struje većih iznosa (do 10A). Razlog korištenja ovakve vrste grijača je mogućnost brzog zagrijavanja prostora unutar staklenika.

Sl. 3.17. Grijač (sušilo za kosu)

3.4. Blokovski prikaz sustava za nadzor i upravljanje

Sve navedene mikroračunalne sustave, senzore i aktuatore potrebno je povezati u jednu skladnu i smislenu cjelinu.



Sl. 3.18. Blokovski prikaz sustava po logičkim cjelinama

Na slici 3.18. dan blokovski prikaz cjelokupnog sustava po logičkim cjelinama, tj. način na koji su povezani dijelovi sustava. Sustav se sastoji od sljedećih cjelina:

- **Nadzorno i upravljačko računalo** – sastoji se od mikroračunalnog sustava Arduino MEGA2560 te predstavlja jezgru staklenika, tj. podaci se prikupljaju sa senzora te se na temelju izmjerenih vrijednost izvršava odgovarajuća radnja. Također je uspostavljena serijska komunikacija između upravljačkog i nadzornog sustava za razmjenu informacije.
- ***SCADA* sustav** – predstavlja nadzornu jedinicu u kojoj je prikazano stanje unutar sustava odnosno vrijednost temperature zraka, vlažnost zraka, vlažnost tla, razina vode u spremniku, količine CO_2 u zraku i stanje aktuatora (*ON*/*OFF*).

- **Senzori –** služe za mjerenje vrijednosti temperature, vlage tla i zraka te količine CO_2 u zraku unutar staklenika.
- **Aktuatori –** utječu na vrijednosti parametara unutar staklenika, tj. promjena temperatura zraka, vlage tla te količinu vlage i CO_2 u zraku.
- **Sklopno/relejni elementi** – za pravilno upravljanje aktuatorima potrebno je dodat sklopne elemente jer Arduino ne može pružiti dovoljnu količinu struje za pokretanje aktuatora. Na priključke sklopnih elemenata doveden je napon iz vanjskog napajanja koji pokreće aktuatore.

Upravljački sustav predstavlja poveznicu s ostatkom sustava, tj. prikuplja podatke sa senzora, obrađuje ih te, na temelju obrađene vrijednosti, izvršava odgovarajuće radnje (uključuje/isključuje aktuatora), prikazuje izmjerene vrijednosti na LC prikazniku te šalje vrijednosti izmjerenih parametara u nadzorni sustav (web aplikaciju).

4. PROJEKTIRANJE I IZRADA PROGRAMSKE PODRŠKE

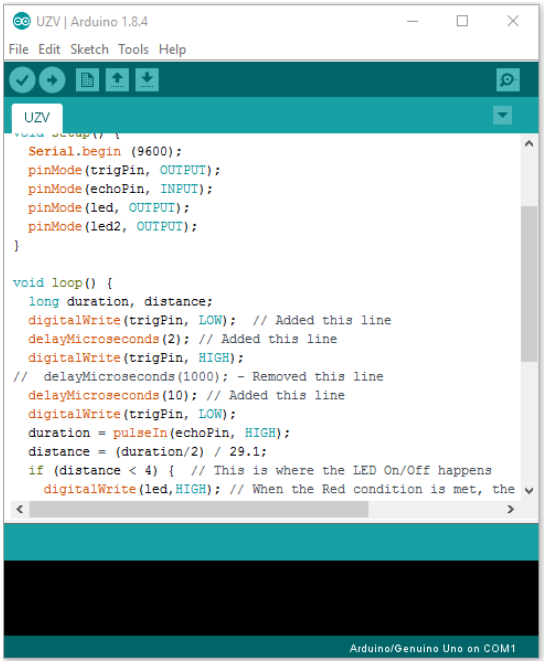
U ovom poglavlju opisan je postupak izrade **programske podrške** za maketu staklenika odnosno programska podrška za rad sa senzorima i aktuatorima te nadzornog sustava, tj. *SCADA* sustava.

4.1. Programska okruženja

U ovom potpoglavlju dan je opis korištenih programskih paketa potrebne za realizaciju upravljačkog sustava (Arduino MEGA2560 i električne komponente) te SCADA sustava (NanoPI NEO i web aplikacija).

4.1.1. Upravljački sustav

Osim što je potrebno fizički povezat Arduino MEGA2560 sa električnim komponentama, bez prikladne programske podrške sustav neće radit. U tu svrhu koristi se **Arduino IDE** (slika 4.1.) razvojno okruženje koji koristi prilagođeni **C-jezik** za rad unutar razvojnog okruženja.



Sl.4.1. Razvojno okruženje Arduino IDE

Razlog korištenja ovog razvojnog okruženja je jednostavnost implementacije koda na Arduino, tj. gotove funkcije za rad sa senzorima, jednostavna uspostava serijske komunikacije te gotove funkcije za obradu sadržaja dobivenih putem serijske komunikacije.

4.1.2. Nadzorni sustav

Za lakše praćenje i razumljiviju interpretaciju informacija dobivenih iz upravljačkog sustava, napravljen je **nadzorni** sustav, tj. *SCADA* sustav. Nadzorni sustav postavljen je na NanoPI mikroračunalu, a sama aplikacija nadzornog sustava sadrži **back-end** i **front-end** sučelje. Pod **back-end** sučeljem podrazumijeva se komunikacija između Arduina i NanoPI mikroračunala, slanje primljenih podataka u bazu te komunikacija između web aplikacije i baze podataka. Pod **front-end** sučeljem podrazumijeva se izrada korisničkog sučelja, tj. web aplikacije gdje korisnik nadzire stanje unutar sustava. Sučelje je izrađeno pomoću *Foundation framework*-a [31], a omogućava izradu *responzivnih* web aplikacija. Za vizualizaciju povijesnih podataka koristi se **Javascript**.

Za rad sa NanoPI mikroračunalom potrebno je na SD karticu instalirati operacijski sustav, tj. koristi se *Snappy Ubuntu Core* i predstavlja Linux Ubuntu 16.04.3 operacijski sustav prilagođen za mikroračunalne sustave pri čemu je izostavljeno grafičko sučelje. Za pristup i programiranje mikroračunala, koristi se **SSH klijent** za uspostavu *SSH*[5] komunikacije sa mikroračunalom.

Za realizaciju nadzornog sučelja, koriste se sljedeće programske tehnologije:

- **Python** – koristi se za serijsku komunikaciju između Arduina i NanoPI-a te zapisivanje i čitanje podataka iz baze podataka.
- **MySQL** – koristi se za upravljanje bazom podataka, tj. pohrana informacije dobivenih od upravljačkog i nadzornog sučelja te za prikaz upisanih podataka na web aplikaciji. Baza se nalazi na mikroračunalu NanoPI.
- **PHP** –služi za dohvaćanje i ispis vrijednosti iz baze podataka te prijavu korisnika u nadzorni sustav.
- **Javascript –** koristi se za vizualizaciju podataka, tj. povijesni prikaz vrijednosti podataka tijekom vremena.

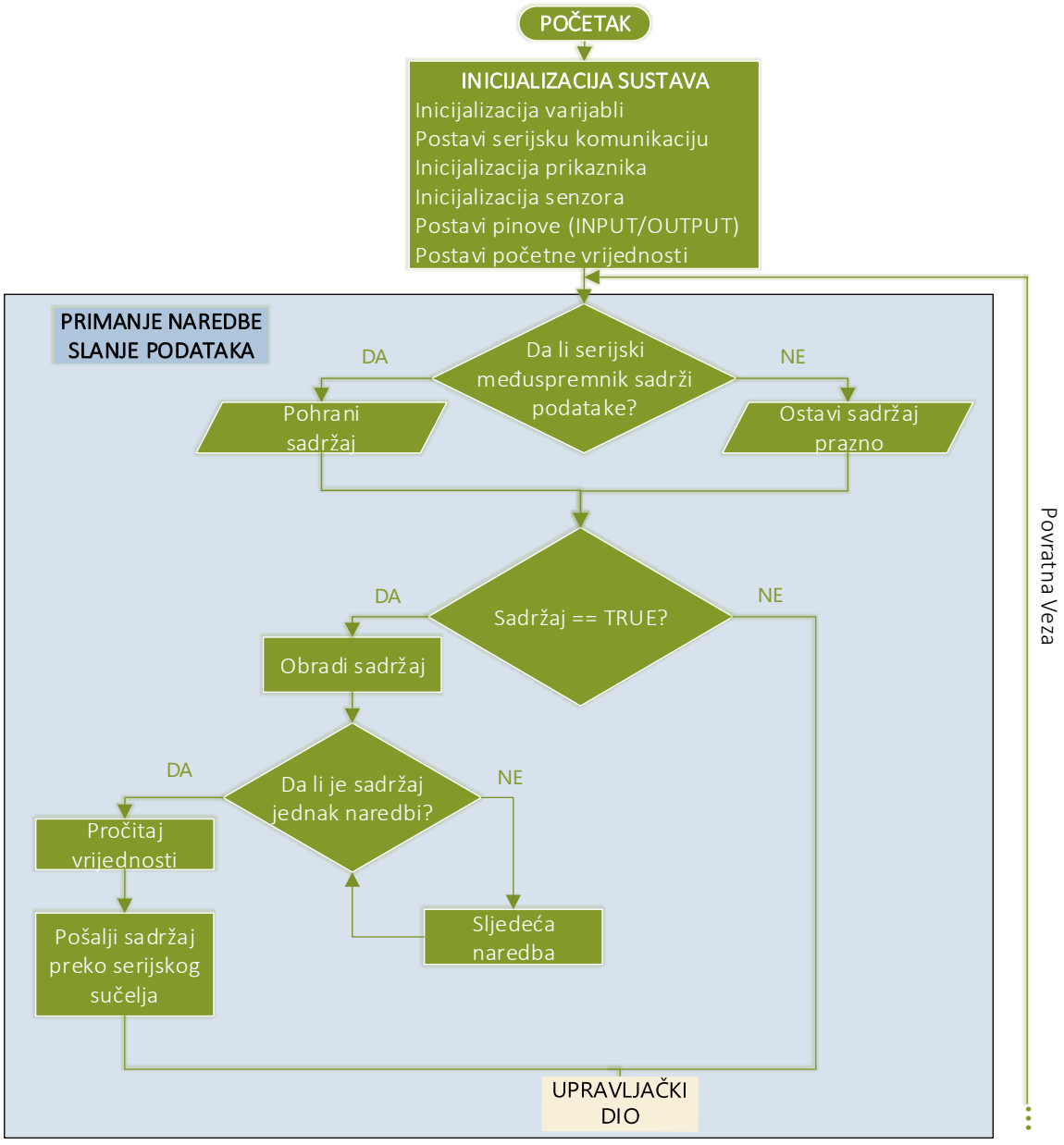
[5] **SSH** ili **Secure Shell** – kriptografski mrežni protokol za rad sa mrežnim servisima preko zaštićene komunikacije na nesigurno mreži

4.2. Upravljački sustav

U ovom potpoglavlju opisana je logika i način rada pojedinih dijelova upravljačkog sustava.

4.2.1 Logika rada upravljačkog sustava

Kao što je spomenuto u prethodnom poglavlju, potrebno je prikupljat podatke sa senzora, obradit ih te izvršiti odgovarajuću radnju na temelju obrađenih vrijednosti, tj. uključiti/isključiti odgovarajući aktuator. Sustav je podijeljen na podatkovni i upravljački dio.

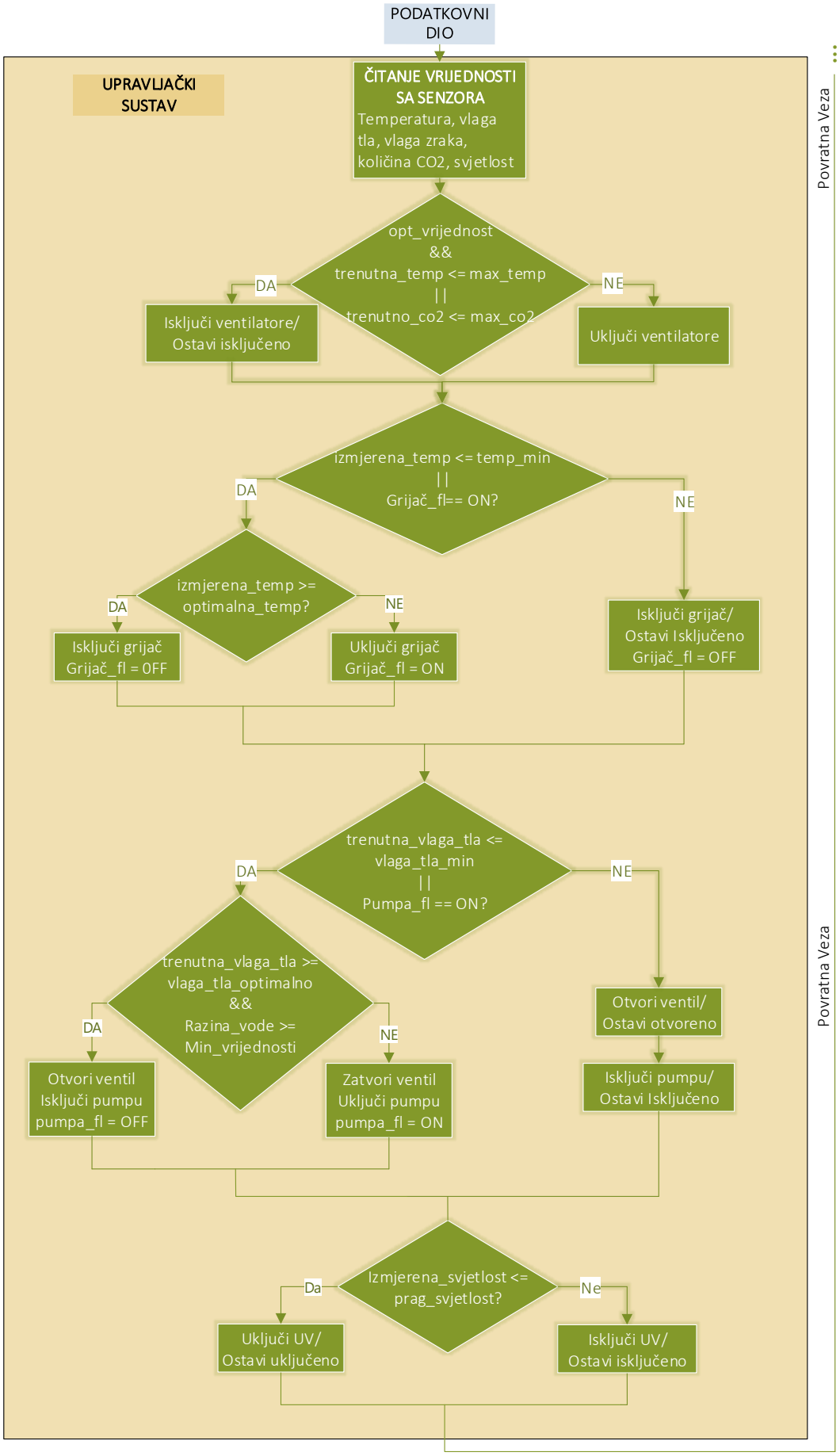


Sl. 4.2. Dijagram toka rada podatkovnog dijela sustava

Na slici 4.2. prikazan je dijagrama toka rada podatkovnog dijela upravljačkog sustava. U podatkovnom dijelu obrađuju se podaci pristigli na serijski međuspremnik (*buffer*). Iz serijskog međuspremnika se čitaju podaci, pri čemu podaci sadrže naredbe i vrijednosti dobivene od nadzornog sustava. U slučaju da postoji sadržaj unutar serijskog međuspremnika, upravljački sustav ih pročita, obradi ih te putem serijske komunikacije vraća odgovor nadzornom sustavu (prikladno dobivenoj naredbi). U potpoglavlju 4.3. detaljnije je opisan postupak obrade i slanje podataka putem serijskog komunikacijskog sučelja.

Na slici 4.3. prikazan je drugi dio sustava, odnosno **upravljački dio**. Upravljački dio nadzire stanje unutar staklenika pomoću senzora i uključuje/isključuje odgovarajuće aktuatore. Unutar upravljačkog dijela se prvotno uzimaju vrijednosti sa senzora, tj. vrijednost temperature zraka, vlažnosti zraka, vlažnost zemlje, količina CO_2 u zraku te trenutna razina svjetlosti u prostoriji. Mjerenja se uzimaju kao što je objašnjeno u potpoglavlju 4.2.2. pri čemu se povećava sigurnost da je izmjerena vrijednost približna stvarnoj. Nakon mjerenja, vrijednosti se uspoređuju sa zadanim graničnim vrijednostima unutar upravljačkog sustava, odnosno sa zadanim graničnim vrijednostima za temperaturu, vlagu zraka, količine CO_2, vlage tla i količine svjetlosti.

Cijeli sustav se nalazi u **povratnoj vezi**, tj. upravljački sustav u svakoj iteraciji provjerava vrijednosti parametara unutar staklenika i u slučaju prelaska min. i max. dopuštenih graničnih vrijednosti, pali se odgovarajući aktuator i izvršava svoju zadaću sve dok se ne postigne optimalna vrijednost. **Primjer** – vrijednost temperature je manja od min. dopuštene temperaturne granice, uključuje se grijač koji izvršava svoju zadaću sve dok se ne postigne **optimalna** temperatura pri čemu je vrijednost optimalne temperature jednaka srednjoj vrijednosti dopuštene min. i max. temperature. Optimalne vrijednosti ovise od biljke do biljke. Za prikaz vrijednosti parametara iz staklenika u nadzornom sučelju, šalju se upiti od strane nadzornog sučelja prema upravljačkom sustavu u vremenskim razmacima od 15 minuta, pri čemu se upiti pohranjuju u bazi podataka i prikazuje na web aplikaciji.



Sl. 4.3. Dijagram toka rada podatkovnog dijela sustava

4.2.2. Prikupljanje podataka sa senzora

Prikupljanje vrijednosti parametara unutar staklenika izvodi se pomoću senzora, ali zbog nesavršenosti električnih komponenata moguće je izmjeriti pogrešne vrijednosti. U tom slučaju potrebno je mjeriti tako da se odbace netočna mjerenja. U programskom isječku koji slijedi prikazano je prikupljanje podataka gdje se odbacuje pogreška.

```
1. float VlagaMedijan() {
2. float vlaga_medijan[6];
3. int i = 0, j = 0, max = 0, temp = 0;
4. vlaga_medijan[0] = dht1.readHumidity();
5. vlaga_medijan[1] = dht2.readHumidity();
6. delay(2000);
7. vlaga_medijan[2] = dht1.readHumidity();
8. vlaga_medijan[3] = dht2.readHumidity();
9. delay(2000);
10.vlaga_medijan[4] = dht1.readHumidity();
11.vlaga_medijan[5] = dht2.readHumidity();
12.for (i = 0; i < 6; i++){
13.  for (j = 1; j < 6; j++){
14.    if (vlaga_medijan[i] < vlaga_medijan[j]){
15.       temp = vlaga_medijan[i];
16.       vlaga_medijan[i] = vlaga_medijan[j];
17.       vlaga_medijan[j] = temp;
18.    }
19.  }
20. }
21.return ((vlaga_medijan[2] + vlaga_medijan[3]) / 2);}
```

Prikazani programski isječak predstavlja funkciju unutar koje se vrši šest uzastopnih mjerenja, vrijednosti se pohrane u vektor, sortiraju po uzlaznoj veličini (pomoću *bubble sort* algoritma), koriste se vrijednosti koje se nalaze na 2. i 3. mjestu u vektoru podataka te se uzme njihova srednja vrijednost i vraća kao rezultat iz funkcije. Gornji primjer predstavlja mjerenje temperature zraka pomoću dva DHT21 senzora.

4.2.3. Upravljanje aktuatorima

Nakon što se podaci prikupe sa senzora, potrebno je donijeti odluku na temelju **dobivenih vrijednosti**. Za primjer, prikazan je postupak upravljanja ventilatorima u programskom isječku koji slijedi. Upravljanje ventilatorima realizirano je pomoću funkcije koja prima 4 argumenta, a to su: trenutna temperatura, trenutna vrijednost CO_2, najveća dozvoljena temperaturna granica i najveća dozvoljeni udio CO_2 unutar staklenika. U slučaju da vrijednost trenutne temperature ili trenutne količine CO_2 nadilazi najveće dopuštene vrijednosti, ventilatori se uključuju. Ventilatori rade sve dok se ne postigne

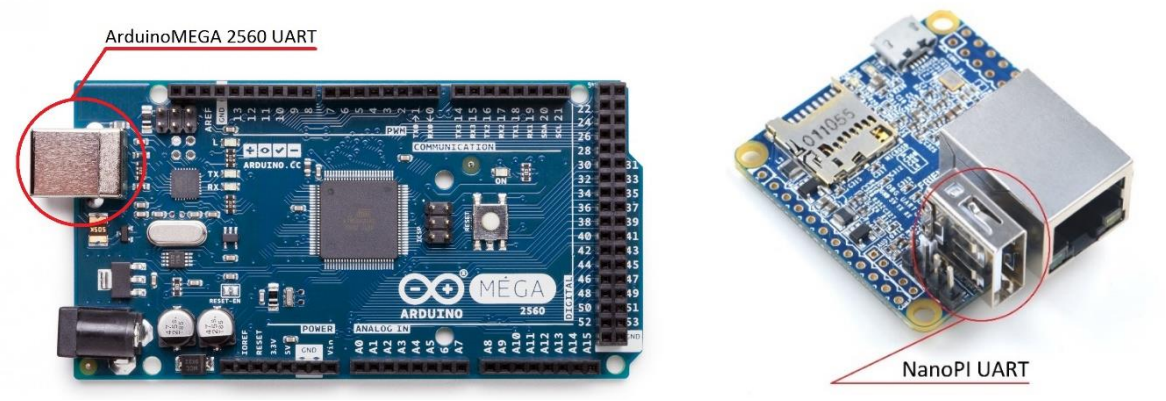
optimalna temperatura (vidi potpoglavlje 4.2.1.) ili zadana granična vrijednost količine CO_2 unutar staklenika.

```
1. boolean Ventilacija(float temp, float co2) {
2.   boolean vent_fl = 0;
3.   Serial.println(temp);
4.   Serial.println(co2);
5.   if (temp > max_temp || co2 > max_co2){
6.     digitalWrite(vent_1_pin, HIGH);
7.     digitalWrite(vent_2_pin, HIGH);
8.     vent_fl = 1;
9.   }
10.  float optimalna_temp = (max_temp + min_temp)/2;
11.  if (temp <= optimalna_temp && co2 <= max_co2 - 5){
12.    digitalWrite(vent_1_pin, LOW);
13.    digitalWrite(vent_2_pin, LOW);
14.    vent_fl = 0;
15.   }
16.  return vent_fl;
17.  }
```

Ostale funkcije za upravljanje aktuatora temelje se na istom principu rada kao i prethodni programski isječak (ako su izmjerene vrijednosti u zadanim granicama - ne čini ništa, inače – uključi odgovarajući aktuator).

4.3. Serijska komunikacija

U ovom potpoglavlju opisana je serijska komunikacije između upravljačkog i nadzornog sustava. Svaki sustav ima svoj način slanja i primanja serijskih podataka. Serijska komunikacije je ostvarena pomoću *UART*[6] sučelja koji se nalaze na Arduinu i NanoPI-u zajedno s „*USB-to-serial*“ pretvornicima kako je prikazano na slici 4.4..



a) b)

Sl. 4.4. UART sučelja: a) ArduinoMEGA2560, b) NanoPI NEO

[6] **UART** (engl. *Universal Asynchronous Receiver-Transmitter*) – računalna komponenta korištena za asinkronu serijsku komunikaciju pri čemu je oblik podataka i brzina slanja podesiva.

4.3.1 Komunikacijski protokol

Za uspješnu komunikaciju između upravljačkog i nadzornog sustava, osmišljen je način komunikacije pomoću podatkovnih paketa koji sadrže odgovarajuće naredbe. Korištene naredbe prikazane su u tablici 4.1.

Tab. 4.1. Naredbe i njihovo značenje

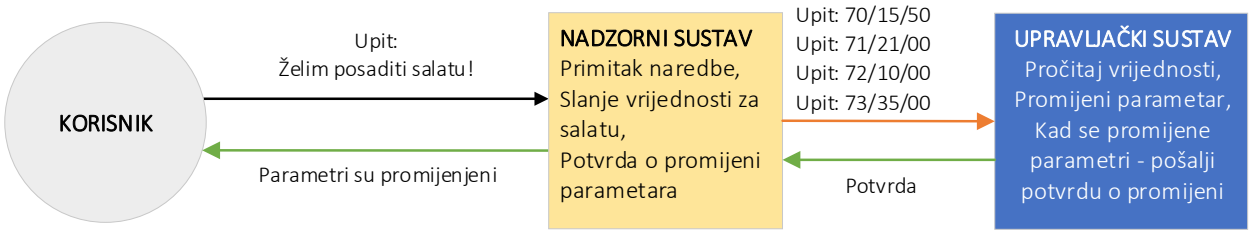
Naredba	Značenje	Naredba	Značenje
10	Trenutna temperatura	**42**	Max. vlažnost zemlje
11	Min. temperatura	**50**	Stanje ventilatora
12	Max. temperatura	**51**	Stanje pumpe
20	Trenutna vlažnost zraka	**52**	Stanje grijaca
21	Min. vlažnost zraka	**53**	Stanje UV LED
22	Max. vlažnost zraka	**60**	Razina vode u spremniku
30	Trenutna količina CO_2	**70**	Promjena min. temperature
31	Min. količina CO_2	**71**	Promjena max. temperature
32	Max. količina CO_2	**72**	Promjena min. vlage tla
40	Trenutna vlažnost zemlje	**73**	Promjena max. vlage tla
41	Min. vlažnost zemlje		

Primjer podatkovnog paketa koji sadrži naredbu sa vrijednostima:

70/15/50

Zelene znamenke predstavljaju broj **naredbe** (u danom primjeru je to promjena min temperature), narančaste znamenke predstavljaju **vrijednost temperature**, odnosno 15.50 °C. Podaci se odvajaju pomoću „ / “ za lakšu obradu podatkovnog paketa unutar sustava.

Primjer - prilikom sadnje nove biljke unutar staklenika, potrebno je promijeniti granične vrijednosti temperature i vlage tla. Korisniku je omogućeno unutar web aplikacije nadzornog sustava odabrat jedan postojeći **recept** pri čemu recepti predstavljaju predefinirane granične vrijednosti temperature zraka i vlage tla koje su postavljene za svaku biljku zasebno prema [3]. Primjer takve komunikacije prikazano je na slici 4.5.



Sl. 4.5. Primjer serijske komunikacije

4.3.2. Implementacija serijske komunikacije unutar Arduina

Za uspostavu serijske komunikaciju u Arduinu, potrebno je izvršit postupak inicijalizacije na slijedeći način:

```
1. Serial.begin(9600);
```

U zagradi se definira brzina serijske komunikacije odnosno *baudrate*[7]. Za pripremu podataka koji se šalju preko serijskog sučelja koristi se Arduinova *String* klasa [21]. Kad nadzorni sustav od upravljačkog sustava zahtjeva neku informaciju, podatak se izravno spremi na serijski međuspremnik (*buffer*) i „čeka“ dok se podatak ne pročita. Pomoću sljedeće naredbe je omogućeno čitanje sa serijskog međuspremnika.

```
1. while (Serial.available()) {
2.   znak = Serial.read();
3.   sadrzaj.concat(znak);
4.   delay(5);
5. }
```

Kad (i ako) postoji podatak u serijskom međuspremniku, program ulazi u „*while*“ petlju i čita znakove sve dok ima podataka unutar međuspremnika. U liniji 2 čitaju se podatci sa serijskog međuspremnika i pohranjuje se znak-po-znak (u varijablu vrste „*char*“). Pohranjeni znakovi pridružuju se varijabli „*sadrzaj*“ koja predstavlja objekt klase „*String*“, pomoću naredbe „*concat*()“ (metoda klase „*String*“). Kad se isprazni međuspremnik, uvjet u prvoj liniji koda postaje neistinit te program nastavlja sa svojim daljnjim radom. Primljeni sadržaj uspoređuje se sa tablicom naredbi te se izvršava odgovarajuća radnja. Primjer takve naredbe prikazan je u slijedećem kodu.

```
1. if (naredba == "10")
2. {
3.   slanje_serial = RastavRealnogBroja(TempMedijan());
4.   Serial.println(slanje_serial);
5.   delay(10);
6. }
```

[7] **Baudrate** - predstavlja brzinu kojom se podaci prenose, odnosno broj simbola koji se šalje po sekundi.

Kako bi se podatak uspješno poslao preko serijskog sučelja, pri čemu su vrijednosti decimalni brojevi tipa „*float*“, potrebno je broj **rastaviti** na dva dijela i poslat ih u posebnom *string* obliku. Rastav brojeva i pretvorba decimalnog broja omogućen je pomoću funckije „*RastavRealnogBroja*()“ prikazano u programskom isječku koji slijedi.

```
1. /*Rastav realnog broja te pretvorba u string za slanje preko seriala*/
2. String RastavRealnogBroja(float vrijednost) {
3.   int prije_dec_tocke = 0, poslije_dec_tocke = 0;
4.   String spojen_dec;
5.   prije_dec_tocke = vrijednost;
6.   poslije_dec_tocke = (vrijednost - prije_dec_tocke) * 100;
7.   spojen_dec = String(prije_dec_tocke)+'.'+String(poslije_dec_tocke);
8. return spojen_dec;
9. }
```

Funkcija prima vrijednost tipa „*float*“. Decimalni broj pretvara i sprema se u varijablu tipa „*int*“ pri čemu se odbacuju vrijednosti poslije decimalne točke. Nakon toga se **predana** vrijednost oduzme sa **pretvorenom** vrijednosti i množi sa 100 što rezultira sa vrijednošću brojeva **poslije** decimalne točke. Nakon toga se brojevi spajaju u vrijednost tipa „*String*“ i odvaja sa „ . “, te je podatak spreman za slanje.

U slučaju primitka paketa koji u sebi sadrži neku **vrijednost**, paket je potrebno **rastaviti** kao što je prikazano slijedećim kodnim isječkom.

```
1. if(sadrzaj[2]){
2.   naredba.concat(sadrzaj[0]);
3.   naredba.concat(sadrzaj[1]);
4.   sadrzaj.remove(0,3);
5.   sadrzaj.replace('/','.');
6.   vrijednost_float = sadrzaj.toFloat();
7. }
```

Uvjet koji se mora ispuniti je da veličina pročitanog sadržaja bude veća od dva znaka jer, po opisanom komunikacijskom protokolu u potpoglavlju 4.3.1., tada se zna da zaprimljeni paket sadrži neku vrijednost. U objekt „*naredba*“ dodaju se prva dva znaka zaprimljenog podatka (naredba), brišu se prva tri znaka unutar objekta „*sadrzaj*“ što na kraju ostavlja **vrijednost** od dobivenog paketa, zamijeni se „ / “ znak sa „ . “ i na kraju se podatak pretvori u varijablu vrste „*float*“ koja je tada spremna za daljnji rad.

4.3.3. Implementacija serijske komunikacije unutar NanoPI-a

Za uspješnu serijsku komunikaciju koristi se biblioteka **PySerial** koja sadržava sve funkcionalnosti za uspješnu serijsku komunikaciju.[22] Prije samog rada, potrebno je odraditi inicijalizacijski postupak za uspostavu serijske komunikacije u kojem se najavljuje

nadzirani *port*[8], brzinu serijske komunikacije (*baudrate*) i vrijeme ispada (engl. *Timeout*[9]). Postupak inicijalizacije prikazan u kodnom isječku koji slijedi.

```
1. import serial
2. #Inicijalizacija serijske komunikacije
3. ser = serial.Serial(port='/dev/ttyUSB0', baudrate=9600, timeout=0.5)
```

Kodni isječak prikazuje postupak osluškivanja *port*-a „*ttyUSB0*“ pri čemu je brzina komunikacije (*baudrate*) 9600, a vrijeme ispada (*timeout*-a) iznosi 0.5 sekundi. S ovim postupkom završen je inicijalizacijski postupak te je *port* spreman za rad. U slijedećem kodnom isječku prikazan je primjer **serijske komunikacije** između Arduina i NanoPI-a.

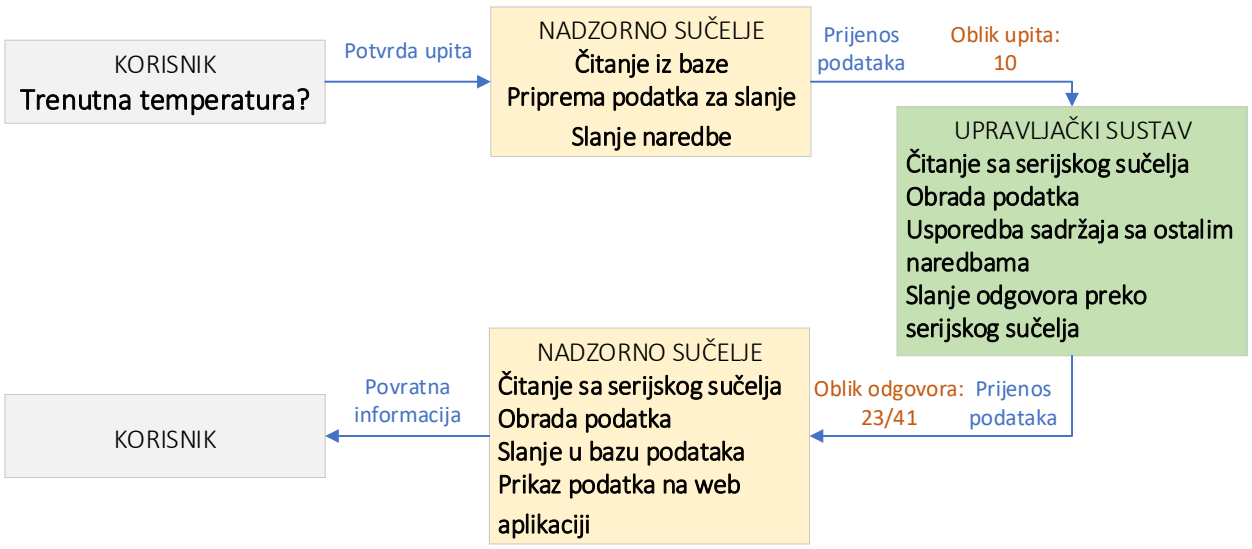
```
1. #Naredba koju saljemo
2. naredba_trenutna_temp = 10
3. #Pretvorba u niz bajtova, pogodno za serijsku komunikaciju
4. slanje_podataka = bytearray(naredba_trenutna_temp,'ascii')
5. #slanje podataka preko USB0 sucelja
6. ser.write(slanje_podataka)
7. #Citanje podataka sa porta u cekanju
8. while (ser.inWaiting()):
9.     #Citanje linije sa serial porta
10.     vrijednost = ser.readline()
11.     #"Ciscenje" porta od sadrzaja
12.     ser.flushInput()
```

U ovom primjeru koristi se naredba za očitavanje trenutne temperature, označeno sa brojem 10, sa upravljačke jedinice. Potrebno pretvoriti navedenu naredbu u oblik koji je pogodan za serijsku komunikaciju. Pomoću naredbe „*bytearray*“ se naredba pretvori **zapakirani bajt podatak** *enkodirano* pomoću *ASCII* standarda. Nakon toga se podatak šalje preko *UART* sučelja na upravljačku jedinicu te se čeka odziv sustava, tj. odgovor. Primitak **odgovora** od upravljačkog sustava je vremenski osjetljivo te je bitno postaviti vrijeme čekanja odgovora. Nakon **slanja** paketa s naredbom, program odlazi u „*while*“ petlju gdje čita podatke sa serijskog međuspremnika i na kraju ispražnjuje međuspremnik.

4.3.4. Primjer serijske komunikacije

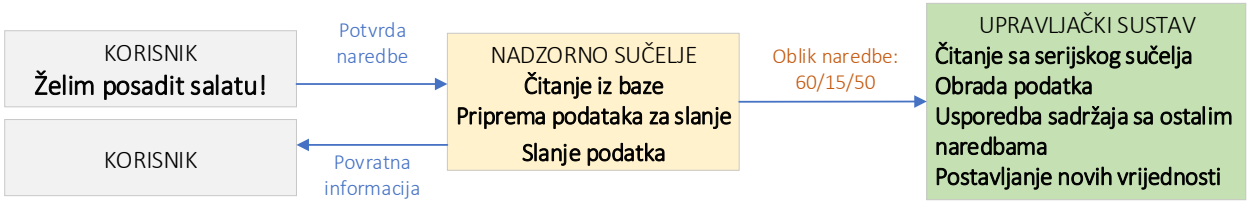
Na slici 4.6. prikazan je primjer serijske komunikacije između upravljačkih sustava gdje korisnik postavlja upit.

[8] **Port** – ulaz UART sučelja na kojem je spojen USB priključak
[9] **Vrijeme ispada** ili **Timeout** – potrebno je kako sustav ne bi ušao u beskonačnu petlju čekajući podatak na serijskom sučelju te s time onesposobi rad sustava,



Sl. 4.6. Primjer serijske komunikacije gdje se očekuje odgovor

Na slici 4.7. prikazan je slučaj gdje se žele promijeniti parametri za sadnju nove biljke, odnosno promjena temperature i vlage tla.



Sl.4.7. Primjer serijske komunikacije gdje se šalje naredba i vrijednost

4.4. Nadzorni sustav

Za prikaz svih dobivenih podataka sa upravljačke jedinice potrebno je napraviti **web aplikaciju** s pripadajućim **web poslužiteljem** i **bazom podataka**.

4.4.1. Web poslužitelj i baza podataka

Unutar lokalne mreže, na NanoPI mikroračunalu, potrebno je postaviti **web poslužitelj** (engl. *Web Server*) kako bi se dobila funkcionalna web aplikacija za nadzor sustava. U tu svrhu koristi se **Apache** koji omogućava prikaz HTML (engl. *HyperText Markup Language*) datoteka putem HTTP (engl. *HyperText Transfer Protocol*) protokola. Navedeni web poslužitelj dostupan je za instalaciju unutar *Ubuntu* operacijskog sustava. Nakon instalacije stvara se mapa od strane web poslužitelja na putanji „/var/www/html“ gdje su smještene sve datoteke potrebne za rad web aplikacije.

Na slici 4.8. prikazana je struktura baze podataka, tj. prikaz svih tablica unutar baze. Napravljeno je 10 tablica u bazi podataka „*scada_gh*“ koje služe za pohranu dohvaćenih vrijednosti iz upravljačkog i nadzornog sustava. Razlog većem broju tablica je lakša **organizacija** i **pristup** podacima. Podaci su grupirani po parametrima kao što su temperatura zraka, vlažnost zraka, vlažnost tla, količina CO_2 i sl.

Tablice „*temp_zraka*“, „*vlaga_zraka*“, „*vlaga_tla*“ i „*co2*“ sadrže vrijednosti parametara koje odgovaraju imenu tablice. Tablica „*trenutni_recepti*“ služi za pohranu trenutnog recepta, odnosno granične vrijednosti parametara za trenutno posađenu biljku unutar staklenika. U slučaju promjene biljke upravljački sustav „zna“ koja je biljka **trenutno** posađena u stakleniku te da se moraju postaviti nove granične parametre za odgovarajuću biljku, tj. biljku koju korisnik želi posaditi. Tablica „*trenutni_podaci*“ služi za pohranu vrijednosti iz upravljačkog sustava na zahtjev korisničkog upita, a to su: trenutna temperatura, trenutna vlažnost zraka, trenutna vlažnost tla te trenutna količina CO_2. Svi dohvaćene (trenutne) vrijednosti prikazane su u web aplikaciji (detaljnije o prikazu vrijednosti u potpoglavlju 4.5.3.).

temp_zraka	
ime	vrsta pod.
datum	date
vrijeme	time
trenutna_temp	float
min_temp	float
max_temp	float

vlaga_zraka	
ime	vrsta pod.
datum	date
vrijeme	time
trenutna_vlaga	float
min_vlaga	float
max_vlaga	float

vlaga_tla	
ime	vrsta pod.
datum	date
vrijeme	time
trenutna_vl_tla	float
min_vl_tla	float
max_vl_tla	float

co2	
ime	vrsta pod.
datum	date
vrijeme	time
trenutno_co2	float
min_co2	float
max_co2	float

ostalo	
ime	vrsta pod.
datum	date
vrijeme	time
uv_stanje	int
pumpa_stanje	int
vent_stanje	int
fen_stanje	int

recept	
ime	vrsta pod.
biljka	varchar
min_temp	float
max_temp	float
min_vlaga	float
max_vlaga	float

trenutni_recepti	
ime	vrsta pod.
biljka	varchar
min_temp	float
max_temp	float
min_vlaga	float
max_vlaga	float

login	
ime	vrsta pod.
ime	varchar
password	varchar

trenutni_podaci	
ime	vrsta pod.
vrijednost	float

razina_vode	
ime	vrsta pod.
razina_vode	float

naredba	
ime	vrsta pod.
naredba	int

Sl. 4.8. Prikaz tablica u bazi podataka „*scada_gh*“

4.4.2. Back-end sučelje

Back-end sučelje sustava sastoji se od serijske komunikacije između mikroračunala, opisana u potpoglavlju 4.3., i upravljanje bazom podataka.

Za upis primljenih podataka sa serijskog *port*-a, od upravljačke jedinice, u bazu podataka, koristi se *MySQLdb Python* [23] modul. Za pravilan rad, potrebno je odraditi inicijalizacijski postupak, a to podrazumijeva uvoz modula te unos osnovnih podataka o *MySQL* bazi. U programskom isječku prikazan je **inicijalizacijski postupak**.

```
1. #Koristeni modul
2. import MySQLdb
3. #Baza podataka na koju se spajamo
4. db = MySQLdb.connect(host="localhost", user="root", passwd="1234", \
   db="scada_ea")
5. #Najava Cursora
6. cursor = db.cursor()
```

Za manipulaciju (čitanje, pisanje i brisanje) podataka unutar baze potrebno je **povezat** *Python* skriptu sa bazom koja primljene podatke sa serijskog *buffer*-a sprema u odgovarajuće tablice unutar baze podataka. Za pravilno izvršenje zadaća, baza radi u pozadini i potrebno je pristupiti bazi sa korisničkim računom koji posjeduje **privilegije** manipulacije podataka iz baze, a u ovom slučaju to predstavlja „*root*“ korisnika koji posjeduje sve te privilegije. Za povezivanje *Python* skripte i baze koristi se *MySQLdb.connect* naredba u koju se unose parametri:

- **host** – IP adresa kojoj se pristupa,
- **user** – ime korisnika s kojim se pristupa,
- **passwd** – lozinka,
- **db** – ime baze kojoj se pristupa.

Naredba u 6. liniji koda, „*cursor*“, predstavlja kontrolnu strukturu koja omogućava kretanje po zapisima u tablicama unutar baze podataka. Nakon inicijalizacijskog postupka može se manipulirati sa podacima unutar baze pomoću standardnih *SQL* naredbi [24]. U programskom isječku koji slijedi prikazan je postupak unosa podataka u bazu tj. **unos podataka** u tablicu „*temp_zraka*“.

```
#Vrijednosti koje se upisuju
trenutna_temp, min_temp, max_temp = 23.56, 19.47, 25.12
#Pokusaj poslat zadane podatke u bazu
try:
  cursor.execute("INSERT INTO scada_gh.temp_zraka(\
  datum, vrijeme, trenutna_temp, min_temp, max_temp)\
  VALUES (now(), now(), %f, %f, %f)" \
  (adresa,trenutni_pritisak, min_pritisak, max_pritisak))
  db.commit()
   print ("Uspjesno poslano u bazu!")
#U slucaju da ne uspije - vrati se na prethodno stanje
except:
   db.rollback()
   print("Nista od slanja")
```

U programskom isječku prikazan je kod koji predstavlja kalup za slanje podataka u bazu, tj. isti pristup se koristi za slanje vrijednosti ostalih parametara u bazu. U ovom slučaju pokušava se poslati 5 vrijednosti u tablicu „*trenutna_temp*“, a to su: datum, vrijeme, trenutna_temp, min_temp i max_temp. Kada se dodijele vrijednosti varijablama, tada slijede *„try-except*“ naredbe koje služe za upravljanje pogreškama (u slučaju da dođe do greške prilikom slanja). U „*try*“ bloku se pomoću naredbe „*INSERT INTO*“ unose podaci u tablicu *scada_gh.temp_zraka* koja se sastoji od slijedećih atributa *datum*, *vrijeme*, *trenutna_temp*, *min_temp* i *max_temp*.. U *Python* i *MySQL* kodu bitno je naglasit vrstu podatka (*int*, *float*, *date* itd.) koja se pohranjuje jer se u suprotnom podatak neće uspješno pohranit te će upisati NULL vrijednost unutar baze. U ovom slučaju se šalju podaci vrste „*float*“ te se u *SQL* naredbi postavlja „*%f*“ argument što *Python* prevoditelju označava da se na tom mjestu nalazi podatak vrste „*float*“, te će upisati vrijednost iz **odgovarajuće** varijable. Pomoću naredbe „*db.commit*()“ podaci se šalju u bazu i ako je sve ispravno postavljeno, podaci će biti vidljivi u bazi.

U programskom isječku koji slijedi prikazan je **ispis podatka**, tj. posljednje naredbe iz baze te pohrana podatka u varijablu.

```
try:
  cursor.execute("SELECT adresa FROM scada_gh.naredba")
  #Pohrana podataka u varijablu
  for i in cursor:
    naredba_t = cursor.fetchone()
    naredba_dek = int(naredba_t[0])
    break
except:
  db.rollback()
  print("Nista od primanja podataka")
```

Naredba “*cursor.fetchone*()“ uzima posljednji podatak iz tablice te ju vraća kao *Python* objekt, a u ovom slučaju je bitan prvi element *Python* objekta.

Manipulacija podacima unutar baze, iz web aplikacije, omogućeno je korištenjem *PHP* programskog jezika. Prije svega, potrebno je povezati web aplikaciju s bazom podataka, kako je prikazano u programskom isječku koji slijedi. Primijeti se da spajanje na bazu izvršava **funkcija** „*mysqli_connect*()“ koja kao parametre prima: naziv web poslužitelja, korisničko ime, lozinku i naziv baze podataka.

```
1. <?php
2. $posluzitelj_name = "localhost";
3. $username = "root";
4. $password = "****";
5. $db = "scada_gh";
6. // stvori vezu
7. $conn = mysqli_connect($poslužiteljname, $username, $password, $db);
8. // Provjera veze
9. if ($conn->connect_error) {
10.   die("Spajanje nije uspjelo: " . $conn->connect_error);
11.   }
12.   ?>
```

Prikupljanje podataka iz baze potrebno je u raznim elementima kao što su tablice, izbornici te prikaz podataka na grafu unutar web aplikacije. Kao i u prethodnom primjeru, sve to omogućava *PHP* kod koji **dohvaća** podatke i prikazuje ih na web aplikaciji. Za **dohvaćanje podataka** iz baze koristi se primjer ispisa imena biljaka koji se nalaze u tablici „*recepti*“. Nakon dohvaćanja podataka, stvara se padajući izbornik u kojem su prikazane biljke sa predefiniranim graničnim vrijednostima. Primjer dohvaćanja podataka prikazan je u programskom isječku koji slijedi.

```
1. <?php
2. $result_biljka = mysqli_query($conn,"SELECT * FROM recepti");
3. while($row_biljka = mysqli_fetch_array($result_biljka)){
4. echo "<option value='" . $row_ biljka ['biljka'] . "'>" . $row_biljka
   ['biljka'] . "</option>"
5. }
6. ?>
```

Nakon odabira biljke potrebno predefinirane vrijednosti upisati u tablicu „*trenutni_recepti*“. Uz pomoć *name* atributa dohvaćaju se imena biljaka i prosljeđuje *POST* proceduri koja podatke sprema u *PHP* varijable te se na kraju pohrane u varijablu koja sadrži cijeli *SQL* upit. Nakon što se vrijednosti pohrane, izvršava se funkcija „*mysql_query*()“ koja kao argumente prima **informaciju** o bazi na koju se šalje i ***SQL* upit**. Kad je *POST* vrijednost postavljena, izvršava se dio programa u kojem se iz tablice „*recepti*“ izvlače vrijednosti za biljku koju je korisnik odabrao, tj. biljku koju želi posaditi. Dohvaćene vrijednosti se spremaju u varijable i ubacuju u *SQL* upit koji vrijednosti šalje u

bazu podataka „*trenutni_recepti*“. U programskom isječku prikazan je prethodno opisan postupak.

```
1.  <?php
2.  if (isset($_POST["promijeni_biljku"])){
3.   $result_biljka = mysqli_query($conn,"SELECT * FROM recepti");
4.   while($row_biljka = mysqli_fetch_array($result_biljka)){
5.     if($row_biljka['biljka']==$_POST["promijeni_biljku"]){
6.       $biljka=$row_biljka['biljka'];
7.       $min_temp=$row_biljka['min_temp'];
8.       $max_temp=$row_biljka['max_temp'];
9.       $min_vlaga=$row_biljka['min_vlaga'];
10.       $max_vlaga=$row_biljka['max_vlaga'];
11.       $sql = "INSERT INTO trenutni_recepti (biljka , min_temp, max_temp,
    min_vlaga, max_vlaga) VALUES ('$biljka', '$min_temp', '$max_temp',
    '$min_vlaga', '$max_vlaga')";
12. mysqli_query($conn, $sql);
13.      }
14.    }
15.  }
16. ?>
```

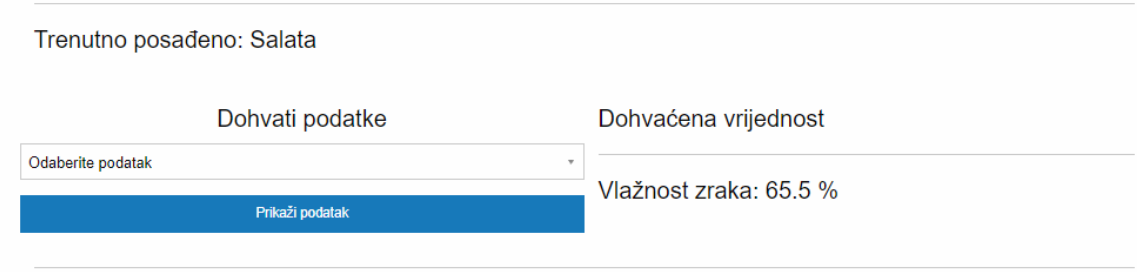
4.4.3. Front-end sučelje

***Front-end* sučelje** sustava izvedeno je uz pomoću *Foundation frontend framework*-a [32] pri čemu je cijela stranica po organizirana po blokovima (engl. *Grid*) kao što je prikazano na slici 4.9.

Sl. 4.9. Grid sustav Foundation framework-a [25]

Početna stranica nadzornog sustava je **prijava korisnika**, prikazano na slici 5.3., koja onemogućava pristup web aplikaciji prije autorizacije korisnika. Nakon unosa korisničkih podatke, sustav provjerava **postoji** li korisnik u bazi podataka (tablica *login*) i ako su podaci ispravni tada započinje „*session*“ koji korisniku omogućava pristup web aplikaciji sve dok je „*session*“ aktivan ili dok ga korisnik sam ne prekine odjavom iz sustava.

Nakon uspješne prijave, korisnik pristupa nadzornoj stranici. Na nadzornoj stranici korisnik ima uvid u **stanje** cjelokupnog sustava (detaljnije u poglavlju 5.).



Sl. 4.10. Dohvaćene trenutne vrijednosti

Na slici 4.10. prikazano je **dohvaćanje trenutnih vrijednosti** unutar sustava. Vrijednosti koje se mogu dohvatiti su: trenutna temperatura zraka, trenutna vlažnost zraka, trenutna vrijednost CO_2 i trenutna vlažnost tla. U programskom kodu što slijedi prikazano je dohvaćanje podataka.

```
<?php
  if (isset($_POST["podatak"])) {
    $podatak = $_POST["podatak"];
  }
  if ($podatak !=0){
    $sql = "INSERT INTO naredba (naredba) VALUES ('$podatak')";
    mysqli_query($conn, $sql);
  }
?>
```

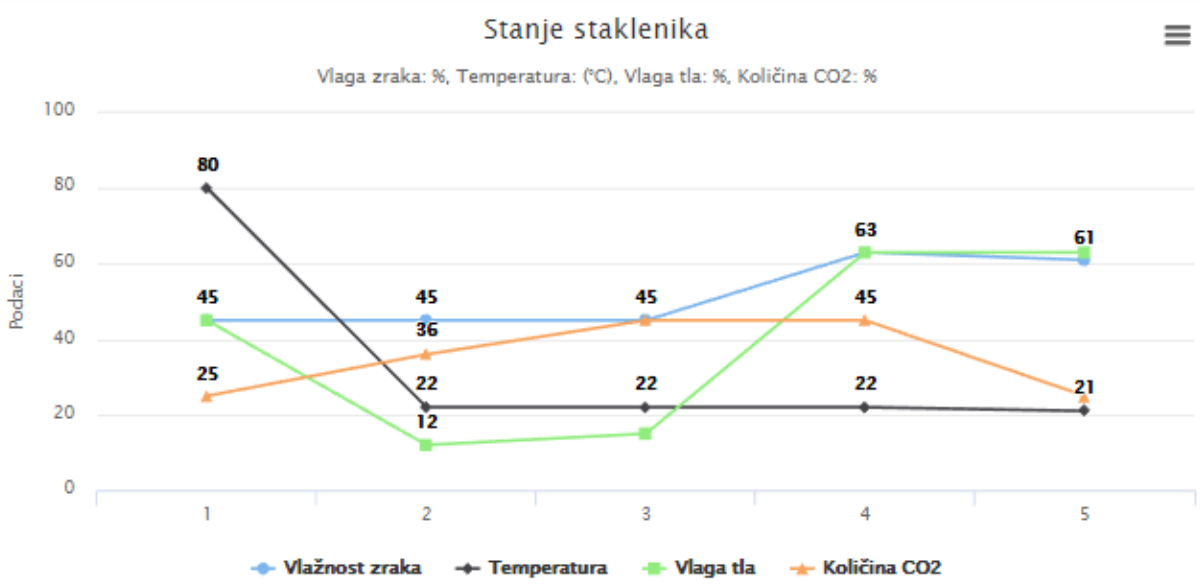
Na temelju **odabrane** vrijednosti šalje se odgovarajuća naredba u tablicu „*naredba*“ i sadržaj te tablice pročita *back-end* dio sustava. Pročitana vrijednost se obrađuje, priprema i šalje preko serijskog sučelja na upravljački sustav. Nakon što se naredba pošalje, čeka se **odgovor** od upravljačkog sustava. *Back-end* dio nadzornog sustava čita primljeni podatak sa serijskog *buffer*-a, obrađuje i zapisuje primljenu vrijednost u bazu, tj. zapisuje u tablicu „*trenutni_podaci*“. U programskom isječku prikazan je prethodno opisani postupak.

```
try:
  cursor.execute("INSERT INTO scada_gh.trenutni_podaci(vrijednost)\
  VALUES (%f)"%(vrijednost_tp))
  db.commit()
  print ("Uspjesno poslano u tablicu trenutni_podaci!")
except:
  db.rollback()
  print("Nista od slanja u tablicu trenutni podaci")
```

Web aplikacija dohvaća podatak iz tablice „*trenutni_podaci*“ i prikazuje na nadzornom sučelju. Postupak je prikazan u programskom isječku koji slijedi.

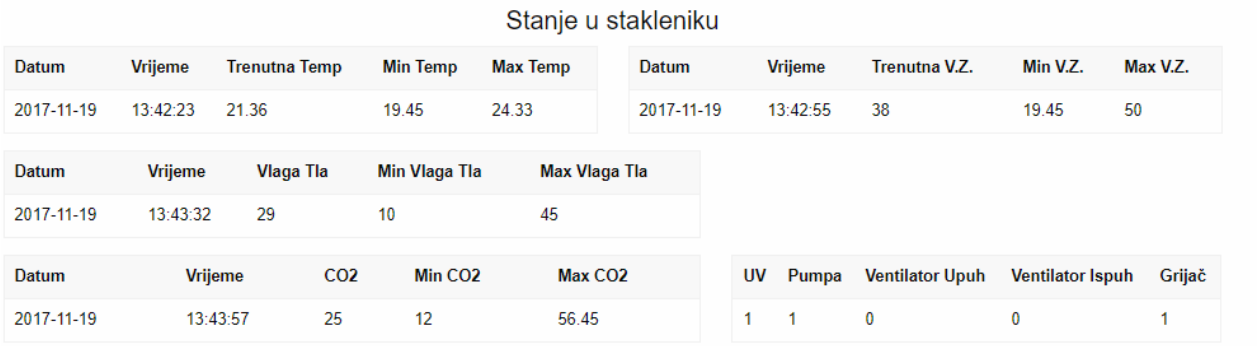
```php
<?php
  $result_trenutni = mysqli_query($conn,"SELECT * FROM trenutni_podaci");
  while($row_trenutni = mysqli_fetch_array($result_trenutni)){
  if($podatak == 10){
    echo "Temperatura: " . $row_trenutni['vrijednost'] . " °C";
  }
  if($podatak == 20){
    echo "Vlažnost zraka: " . $row_trenutni['vrijednost'] . " %";
  }
  if($podatak == 30){
    echo "Količina CO2: " . $row_trenutni['vrijednost'] . " %";
  }
  if($podatak == 40){
    echo "Vlažnost tla: " . $row_trenutni['vrijednost'] . " %";
  }
  if($podatak!=10 && $podatak!=20 && $podatak!=30 && $podatak!=40){
    echo "Odaberite podatak!";
}
?>
```

Grafički i **tablični** prikaz slijedi nakon dohvaćanja podataka. Povijesni prikaz podataka na grafu omogućeno je korištenjem interaktivnih *HighCharts Javascript* [33] grafova koji se jednostavno mogu ugraditi i prilagoditi za različite namjene. U nadzornom sustavu je primijenjen za prikaz **povijesnih podataka** temperature zraka, vlažnosti zraka, vlažnosti tla i količine CO_2. Podaci se dohvaćaju iz odgovarajućih tablice i može se odabrati svaki parametar zasebno za prikaz na grafu. Na slici 4.11. prikaz je graf povijesnih podataka pri čemu se na y-osi nalazi vrijednost parametra, a na x-osi prikaz kroz vrijeme.



Sl. 4.11. Graf sa povijesnim podacima

Prikaz vrijednosti parametara iz baze podataka izvedeno pomoću **tablica** gdje svaka tablica pripada jednoj **skupini parametara** kao što je prikazano na slici 4.12. Može se primjetiti da su ispisani **posljednji** (najnoviji) podaci svih mjerenih parametara unutar sustava, te datum i vrijeme dohvaćanja podataka. Također postoji tablica u kojima su prikazana trenutna stanja aktuatora unutar makete staklenika pri čemu se koristi binarna logika za prikaz stanja aktuatora (1 za uključeno, 0 za isključeno).

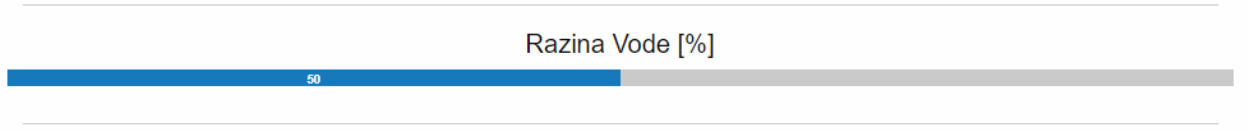


Sl.4.12. Tablice stanja

U programskom isječku koji slijedi prikazano je dohvaćanje i prikaz podataka unutar tablica.

```
1. <?php
2.    $result_co2 = mysqli_query($conn,"SELECT * FROM co2 ORDER BY datum
   DESC LIMIT 1");
3.   while($row_co2 = mysqli_fetch_array($result_co2)){
4.    echo"<tr>";
5.    echo "<td>" . $row_co2['datum'] . "</td>";
6.    echo "<td>" . $row_co2['vrijeme'] . "</td>";
7.    echo "<td>" . $row_co2['trenutno_co2'] . "</td>";
8.    echo "<td>" . $row_co2['min_co2'] . "</td>";
9.    echo "<td>" . $row_co2['max_co2'] . "</td>";
10.   }
11.  ?>
```

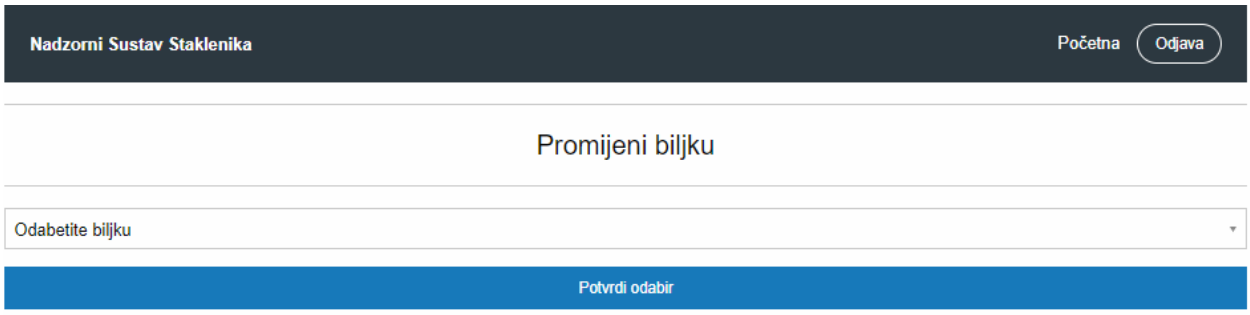
Na slici 4.13. prikazana je **trenutna razina vode** u spremniku vode. Za te potrebe korišten je stupčasti prikaz napretka (engl. *Progress bar*) koji je preuzet od *Foundation framework*-a.



Sl. 4.13. Razina vode

Dohvat vrijednosti se također obavlja kao i u ostalim primjerima. Vrijednost se iščitava iz tablice „*razina_vode*“ te se prikazuje pomoću navedenog *progress bar*-a.

U slučaju da korisnik želi posaditi **novu biljku**, potrebno je postaviti recept. Na slici 4.13. prikazana je stranica za promjenu recepta.



Sl. 4.13. Promjena biljke

Prije svega, potrebno je odabrati biljku koja se sadi te nakon odabira stisnuti „Potvrdi odabir“ kako bi se vrijednosti poslale u bazu podataka, tj. tablicu „*trenutni_recepti*“. Nakon promjene, na nadzornoj stranici se ispiše posađena biljka. U programskom isječku koji slijedi prikazan je način promjene biljke.

```
1. <?php
2.   if (isset($_POST["promijeni_biljku"])){
3.     mysqli_query($conn,"TRUNCATE TABLE trenutni_recepti");
4.     $result_biljka = mysqli_query($conn,"SELECT * FROM recepti");
5.     while($row_biljka = mysqli_fetch_array($result_biljka)){
6.       if($row_biljka['biljka']==$_POST["promijeni_biljku"]){
7.         $biljka=$row_biljka['biljka'];
8.         $min_temp=$row_biljka['min_temp'];
9.         $max_temp=$row_biljka['max_temp'];
10.         $min_vlaga=$row_biljka['min_vlaga'];
11.         $max_vlaga=$row_biljka['max_vlaga'];
12.         $sql = "INSERT INTO trenutni_recepti (biljka , min_temp,
   max_temp, min_vlaga, max_vlaga) VALUES ('$biljka', '$min_temp',
   '$max_temp', '$min_vlaga', '$max_vlaga')";
13.         mysqli_query($conn, $sql);
14.       }
15.     }
16.   }
17.  ?>
```

Nakon potvrde odabira, ime biljke šalje se kao *POST* varijabla. Podaci koji se nalaze u tablici „*trenutni_recepti*“ se **brišu** jer nije potrebno zadržavat recept prethodno posađene biljke. Iz tablice „*recepti*“ dohvaćaju se **svi** podaci te se imena biljaka **uspoređuju** sa *POST* varijablom koja sadrži ime biljke koja se želi posaditi. Kad se pronađe željena biljka, vrijednosti atributa „*min_temp*“, „*max_temp*“, „*min_vlaga*“ i „*max_vlaga*“ se unesu u tablicu „*trenutni_recepti*“. Nakon čitanja, obrade i slanja podataka, upravljački sustav prima podatak te postavi nove granične vrijednosti.

5. RAD SUSTAVA

U ovom poglavlju opisan je postupak postavljanja sustava i osposobljavanja za rad.

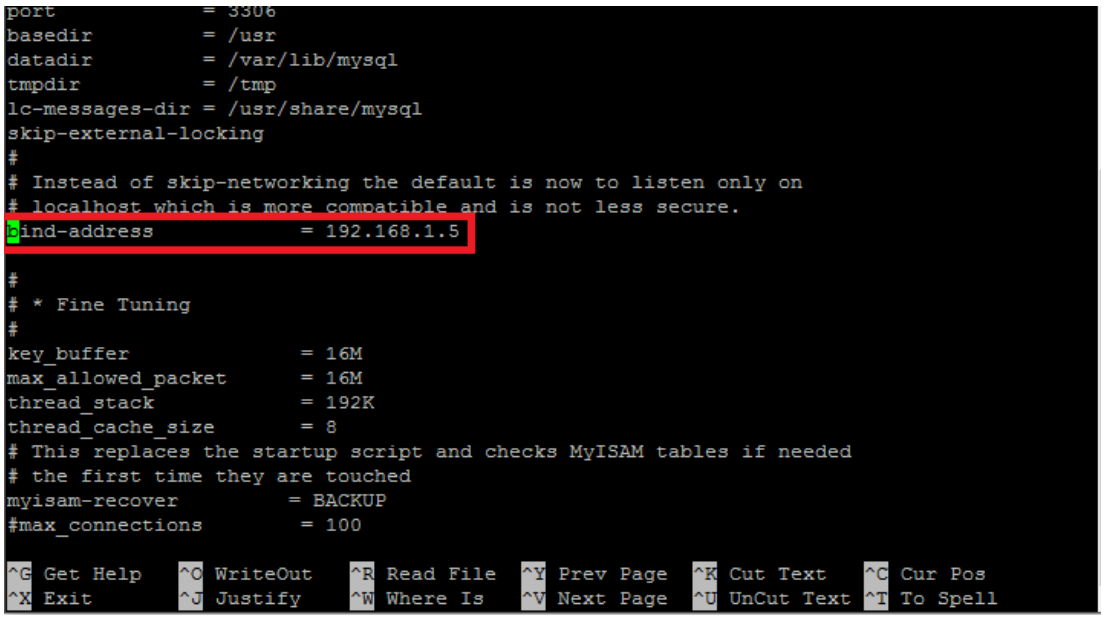
5.1. Postavljanje nadzornog sučelja

Kad se NanoPI uključi, *Apache* web poslužitelj (*web server*) se automatski pokrene. Kako bi se pravilno pokrenula web aplikacija, potrebno je sustav spojiti na **internet**. Nakon što se sustav spoji na internet, potrebno je saznat **IP adresu** *NanoPI*-a jer *MySQL* baza podataka neće raditi i vrijednosti će biti nedostupne web aplikaciji. Postoje dva načina za dohvaćanje IP adrese, **prva** je da se korisnik spoji na *ruter* te u postavkama pronađe IP adresu, kao što je prikazano na slici 5.1.

b8:27:eb:69:70:7c	192.168.1.5	75250	raspberrypi	SSID1

Sl. 5.1. IP adresa dobivena ulaskom u sučelje *rutera*

Drugi način je, nakon spajanja na internet, da se na terminalu od sustava upiše naredba „*ifconfig*“ te se dobije ispis postavki za internet, pod stavkom „*eth0*“ treba pronaći stavku „*inet_addr*“ i tamo se pronaći trenutnu IP adresa sustava. Kad se obavi taj postupak, potrebno je u bazu podataka upisati IP adresu, a u ovom slučaju je „192.168.1.5“. Naredbom „*sudo nano /etc/mysql/my.cnf*“ ulazi se u konfiguracijsku datoteku te se pod stavkom „*bind-adress*“ unosi IP adresa kao što je prikazano na slici 5.2.



Sl. 5.2. Unos IP adrese

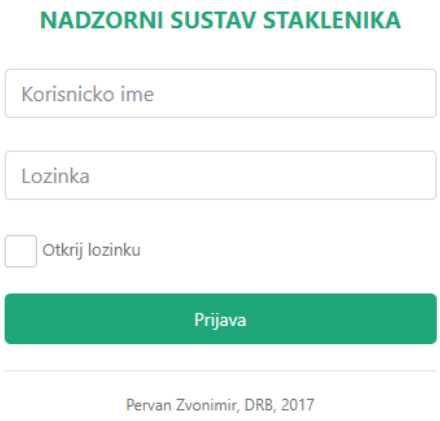
Kad se unese IP adresa, baza podataka i web aplikacija je spremna za rad.

5.2. Rad cjelokupnog sustava

Kad se postave baza podataka i web poslužitelj, potrebno je uključiti upravljački sustav. Kako bi se pristupilo web aplikaciji, potrebno je u web preglednik unijeti IP adresu nadzornog sustava te ime aplikacije „*scada_gh*“ kao što slijedi:

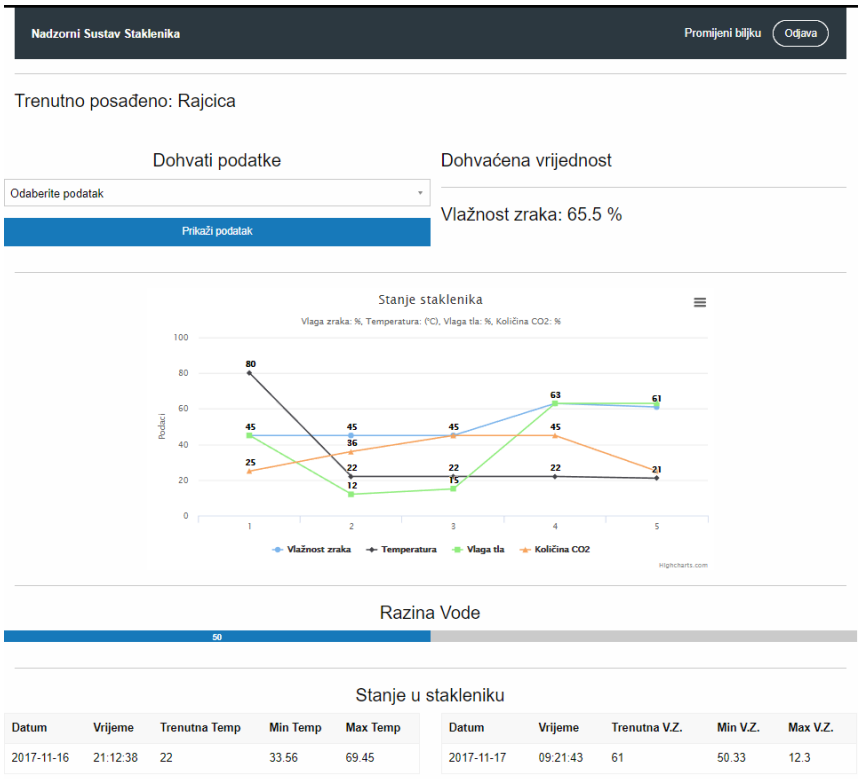
```
192.168.1.5/scada_gh
```

Nakon upisa adrese, otvori se početna stranica odnosno stranica za prijavu korisnika kao što je prikazano na slici 5.3.



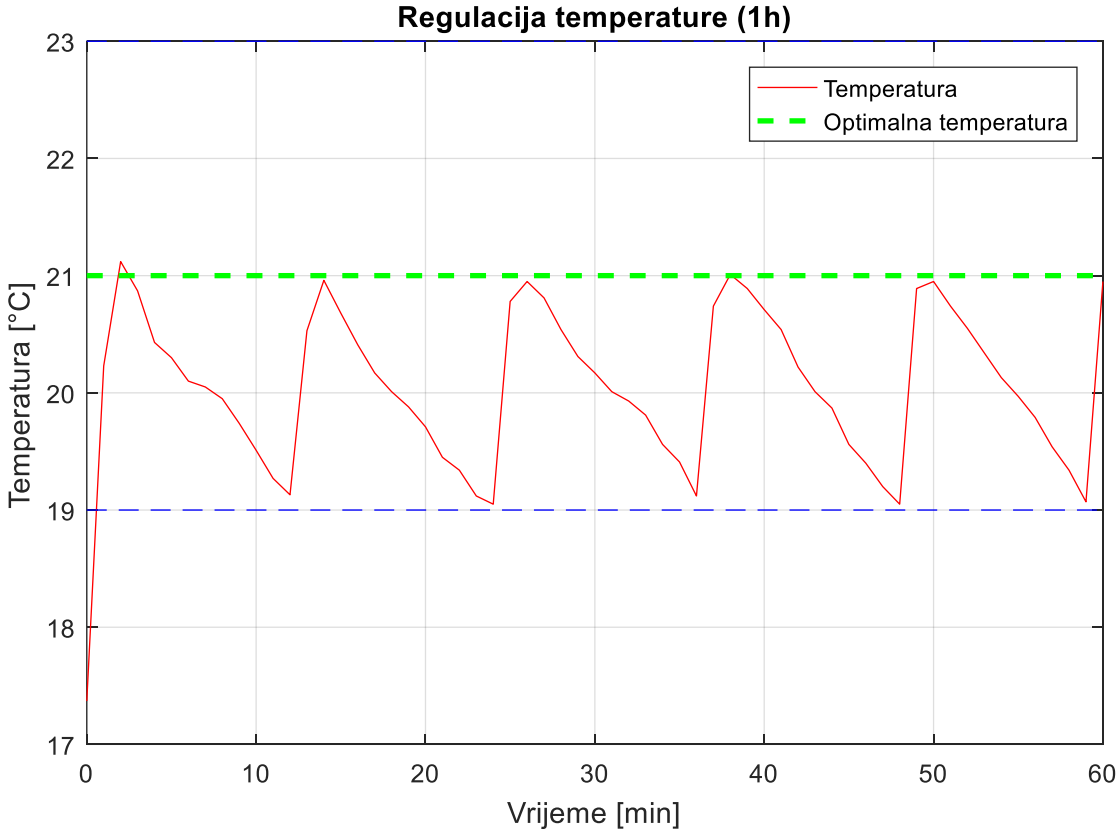
Sl. 5.3. Sučelje za prijavu

Unosom ispravnih korisničkih podataka, pristupa se glavnoj nadzornoj stranici koja je prikazana na slici 5.4.



Sl. 5.4. Nadzorno sučelje

Nadzorno sučelje prikazuje stanje unutar makete staklenika. Nakon pokretanja, sustav **inicijalizira** početne parametre, no prije toga korisnik mora postaviti **parametre za** biljku koju je posadio. Nadzorni sustav preko serijskog komunikacijskog sučelja šalje podatkovne pakete upravljačkom sustavu koji sadrže **naredbe** sa **vrijednostima** graničnih parametara. Nakon primitka vrijednosti, upravljački sustav postavlja granične vrijednosti i započinje sa radom. **Upravljačka logika** i rad sustava opisani su u poglavlju 4. Sustav svakih **15 minuta** mjeri trenutne vrijednosti parametara unutar staklenika te ih šalje nadzornom sustavu koji izmjerene vrijednosti prikazuje u web aplikaciji. Na slici 4.9. prikazano je dohvaćanje **trenutnih** vrijednosti (temperatura, vlažnost tla, vlažnost zraka i količina CO_2 u zraku) unutar staklenika. Nakon odabira parametra, korisnik potvrdi odabir pritiskom na gumb „Prikaži podatak“ te se nakon kratkog vremena ispiše vrijednost sa desne strane, kao što je prikazano na slici 5.4. U slučaju otvaranja poklopca staklenika, postavljeno je **tipkalo** ispod poklopca staklenika koje šalje signal sustavu da je poklopac podignut te postavlja sustav u stanje mirovanja, odnosno, gasi grijače, ventilatore, UV rasvjetu i navodnjavanje jer se pretpostavlja da korisnik želi nešto raditi unutar staklenika.



Sl. 5.5. Primjer regulacije temperature unutar staklenika

Na slici 5.5. dan je primjer regulacije **temperature** unutar staklenika pri čemu y-os predstavlja vrijednost temperature u °C, a x-os vrijeme u minutama. Na grafu su prikazane granične vrijednosti, pri čemu je donja granična temperatura 19°C, a gornja granična temperatura 23°C. Prilikom uključivanja sustava, temperatura unutar staklenika iznosila je 17.37°C što je manje od dopuštene temperature te se zbog toga uključio grijač. Nakon što se postigne **optimalna temperatura** (vidi potpoglavlje 4.2.1.) od 21°C, grijač se isključi. Na slici se primijeti kako temperatura opada s vremenom te svakih 10 do 12 minuta postiže najmanju dopuštenu temperaturnu vrijednost te se grijač ponovno uključuje i grije staklenik. Uzorak temperature se uzimao svake minute, odnosno period uzorkovanja iznosi 60s.

Na slici 5.6. prikazana je gotova maketa staklenika.

Sl. 5.6. Maketa staklenika

6. ZAKLJUČAK

Za uspješan rast biljaka, potrebno je održavat fizičke parametre (temperatura, vlaga tla i zraka, količina CO_2 u zraku) unutar vrijednosti pogodne za rast biljaka. Čovjek je biće koje zaboravlja i kad radi s vremenom mu opada koncentracija te čini pogreške. Automatizacija uzgoja biljaka predstavlja jedno rješenje koje umanjuje ljudske pogreške te uvelike pospješuje rast biljaka jer omogućava stalni nadzor fizičkih parametara te manipulaciju istih (zalijevanje, provjetravanje, grijanje, hlađenje itd.).

Prilikom izrade diplomskog rada javljaju se mnogi problemi tijekom dizajniranja programske i elektroničke podrške. Potrebno je vremena i strpljenja za uspješno dizajniranje i rad kompleksnijih sustava jednog takvog automatiziranog procesa.

Cilj ovog diplomskog rada je iskoristiti znanje stečeno tijekom studiranja te dizajnirati automatizirani sustav staklenika pomoću odgovarajućih senzora i aktuatora te sa odgovarajućim nadzornim sustavom pomoću dostupnih web tehnologija. Svaka biljka optimalno raste u odgovarajućim fizičkim uvjetima, tj. temperatura, vlažnost tla i zraka. Također, jedan od bitnih parametara je kvaliteta zraka, odnosno količina CO_2 u zraku unutar staklenika jer se kod ustajalog zraka mogu razviti patogene bakterije koje mogu uništiti biljku. Izrađeni sustav sastoji se od sustava za ventilaciju, grijanja i navodnjavanja, pri čemu se održavaju sljedeće veličine unutar zadanih granica: temperatura zraka, vlažnost zraka, količina CO_2 u zraku. Izrađena je web aplikacija koja omogućava nadzor izmjerenih fizičkih parametara unutar staklenika, dohvaćanja trenutnih vrijednosti te promjena referentnih vrijednosti.

Moguća poboljšanja postoje. Nadogradnja nadzornog sustava sa kamerom koja bi na dnevnoj bazi slala slike iz staklenika web aplikaciji te korisniku omogućilo veću kontrolu nad rastom biljaka i sustavom.

LITERATURA

[1] Greenhouse, Wikipedia.org, dostupno na: https://en.wikipedia.org/wiki/Greenhouse [22.10.2017]

[2] https://www.greenhousepeople.co.uk/greenhouse/5415/popular-6ft-x-4ft-mill/ [04.12.2017]

[3] Requirements for planth growth, dostupno na: http://www.aces.uiuc.edu/vista/html_pubs/hydro/require.html [29.09.2017]

[4] Agroklub, dostupno na: https://www.agroklub.com/ [15.04.2017]

[5] Arduino MEGA2560, Arduino company, dostupno na: https://www.arduino.cc/en/main/arduinoBoardMega2560 [16.04.2017]

[6] NanoPI NEO, dostupno na: http://wiki.friendlyarm.com/wiki/index.php/NanoPi_NEO [16.04.2017]

[7] DHT21 sensor, dostupno na: https://www.digibay.in/427-dht21-am2301-digital-temperature-and-humidity-sensor [04.12.2017]

[8] MQ-135 Gas Sensor, dostupno na: https://www.olimex.com/Products/Components/Sensors/SNS-MQ135/resources/SNS-MQ135.pdf [04.12.2017]

[9] MQ 135 gas sensor, dostupno na: https://potentiallabs.com/cart/air-quality-control-gas-sensor-mq135 [04.12.2017]

[10] Ultrasonic ranging module, dostupno na: http://www.micropik.com/PDF/HCSR04.pdf [04.12.2017]

[11] Specification for LCD module, dostupno na: https://www.openhacks.com/uploadsproductos/eone-1602a1.pdf [04.12.2017]

[12] LCD 1602, dostupno na: http://www.yourduino.com/sunshop/index.php?l=product_detail&p=62 [04.12. 2017]

[13] YL-38 soil mositure sensor with Arduino, dostupno na: http://www.diyspacepk.com/yl-38-soil-moisture-sensor-with-arduino/ [04.12.2017]

[14] Single 5V relay board, dostupno na: http://microcontrollershop.com/product_info.php?products_id=5919 [04.12.2017]

[15] DC submersible water pump, dostupno na: https://www.ebay.com/itm/Ultra-Quiet-Mini-DC-12V-Lift-3M-240L-H-Brushless-Motor-Submersible-Water-Pump-/132087694620?_trksid=p2349526.m4383.l4275.c10 [04.12.2017]

[16] DC 12V Solenoid, dostupno na: https://www.ebay.com/itm/DC-12V-Solenoid-valve-Mini-electric-valve-Discouraged-valve-Normally-closed-FW/253213204786?epid=1139470053&hash=item3af4aee132:g:JZMAAOSwighZgCSc [04.12.2017]

[17] The Importance of UV Light for Plants Cultivated Indoors, Best LED, dostupno na: https://bestledgrowlightsinfo.com/the-importance-of-uv-light-for-plants-cultivated-indoors/ [19.09.2017]

[18] DC 12V UV light, dostupno na: https://www.ebay.com/itm/DC12V-3528-5050-UV-Ultraviolet-purple-waterproof-60led-m-Strip-lamp-light/262493956304?hash=item3d1ddbe8d0:m:m5ai67U4Dn7nqb0vJcxaGgg [04.12.2017]

[19] DC 12V Brushless fan, dostupno na: https://www.amazon.com/Gdstime-Bearing-Brushless-Cooling-Exhaust/dp/B00N1Y4BMA [04.12.2017]

[20] Photoresistor, Wikipedia.org, dostupno na: https://en.wikipedia.org/wiki/Photoresistor [04.12.2017]

[21] Arduino String(), Arduino company, dostupno na: https://www.arduino.cc/reference/en/language/variables/data-types/stringobject/ [15.11.2017]

[22] PySerial dokumentacija, dostupno na: https://pyserial.readthedocs.io/en/latest/ [15.11.2017]

[23] MySQLdb User's Guide, dostupno na: http://mysql-Python.sourceforge.net/MySQLdb.html [21.09.2017]

[24] SQL Tutorial, W3 Schools, dostupno na: https://www.w3schools.com/sql/ [05.12.2017]

[25] Foundation framework documentation, dostupno na: https://foundation.zurb.com/ [05.12.2017]

[26] When do you use the Arduino's Serial.flush()?, dostupno na: https://www.baldengineer.com/when-do-you-use-the-arduinos-to-use-serial-flush.html [10.11.2017]

[27] Can't send float values from one Arduino to another, dostupno na: https://stackoverflow.com/questions/36598774/cant-send-float-values-from-one-arduino-to-another#36600329 [10.11.2017]

[28] Cursor (databases), Wikipedia.org, dostupno na: https://en.wikipedia.org/wiki/Cursor_(databases) [21.09.2017]

[29] Linux ifconfig command, dostupno na: https://www.computerhope.com/unix/uifconfi.html [19.11.2017]

[30] M. Margolis, Arduino Cookbook, O' Reilly Media, Inc., Sebastopol, 2011.

SAŽETAK

Naslov: Izgradnja makete staklenika i izrada pripadne programske podrške za nadzor i upravljanje

Sažetak: U ovom radu dan je postupak izgradnje staklenika pri čemu je postupak zalijevanja, grijanja i ventilacije automatiziran. Automatizacija je omogućena pomoću Arduino MEGA2560 mikroračunala i odgovarajućih senzora i aktuatora. Također je izrađena odgovarajuća web aplikacija za nadzor sustava tj. mogućnost praćenja temperature, vlažnost zraka i tla te količine CO_2. Web aplikacija, web poslužitelj i serijska komunikacija smještena je na mikroračunalu NanoPI NEO sa „*Snappy Ubuntu Core*“ operacijskim sustavom.

Ključne riječi: ugradbeni računalni sustavi, staklenik, automatizacija, IoT

ABSTRACT

Title: Building a greenhouse model with the adequate software support for supervision and control

Abstract: This paper reviews the building procedure of a greenhouse model in which the tasks of heating, watering and ventilation are fully automated. The automation process is realized with the Arduino MEGA2560 microcontroller unit and the appropriate sensors and actuators. Also, a appropriate web application is made for supervision in which the user can get the information of the temperature, air humidity level, earth humidity level and CO_2 concentration. The web application, web poslužitelj and serial communication is realized on the NanoPI microcontroller with the „*Snappy Ubuntu Core*“ operating system on it.

Keywords: embedded microcontroller system, greenhouse, automation, IoT

ŽIVOTOPIS

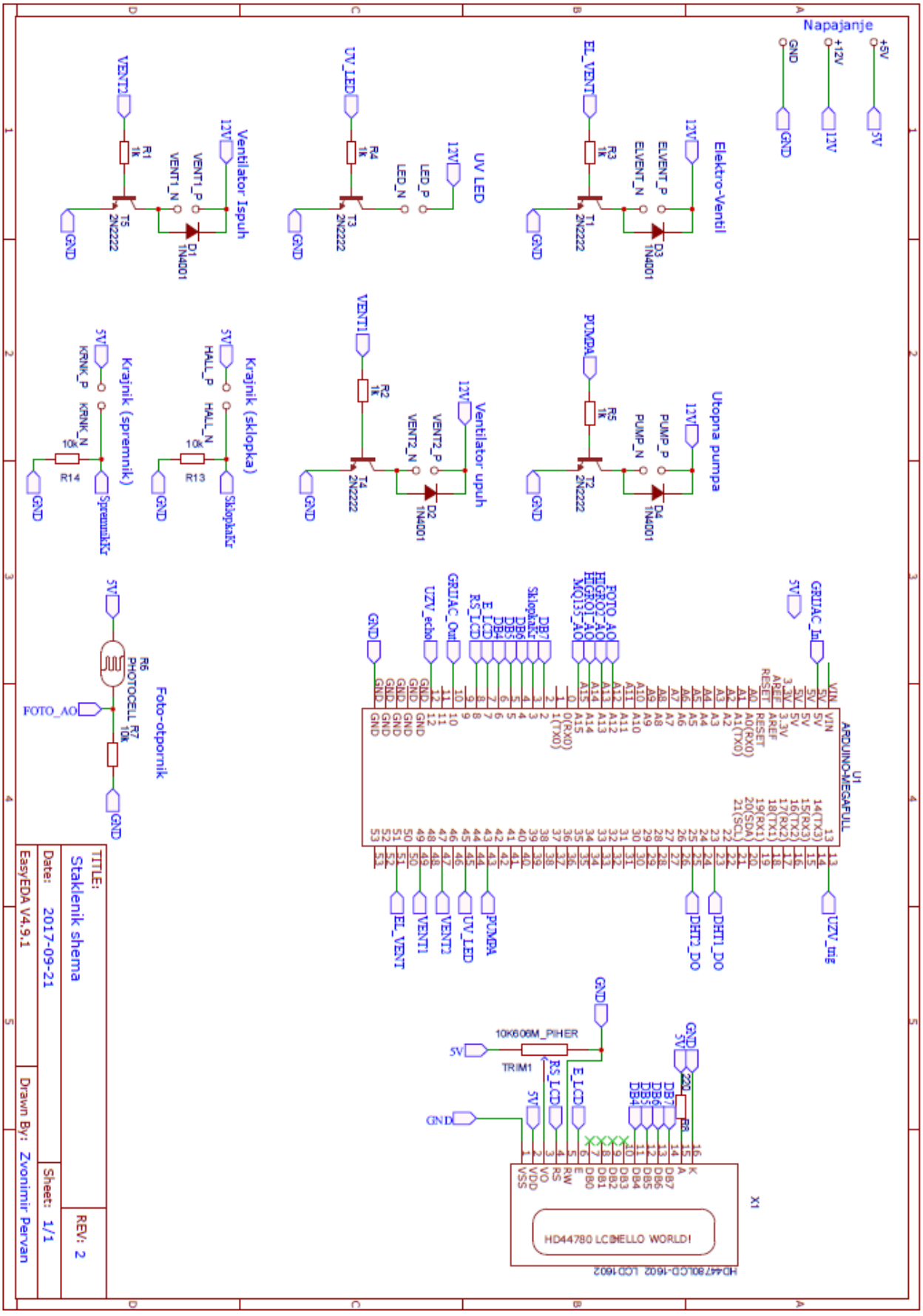
Zvonimir Pervan, rođen je 12.10.1990. godine u Vinkovcima. U 2004. godini završava osnovnu školu „Zrinski“ u Nuštru i iste godine upisuje srednju tehničku školu „Ruđer Bošković“ u Vinkovcima, smjer Tehničar za Mehatroniku te ju završava 2009. godine. Nakon stanke u obrazovanju, 2011. godine upisuje stručni studij smjer Automatika na Elektrotehničkom fakultetu u Osijeku i završava ga 2014. godine. Iste godine upisuje godinu razlikovnih obveza na istom fakultetu, 2015. godine ju uspješno završava i upisuje sveučilišni diplomski studij smjer Procesno računarstvo na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Tijekom diplomskog studija dobiva nagradu za postignut uspjeh tijekom studiranja i postaje stipendist općine Nuštar.

Potpis

_____

PRILOZI

Prilog P.3.1.:



Prilog P 3.3.1. [21]:

Broj pina	Simbol	Logička razina (H/L)	Značenje/funkcija
1	V_{SS}	--	GND
2	V_{DD}	--	+5V
3	V_0	--	Za LC prikaznik
4	RS	H/L	H: Unos znakova, L: Unos instrukcija
5	R/W	H/L	H: Čitanje sa LCD, L: Pisanje na LCD
6	E	H/L	Odobrava unos podataka
7	DB0	H/L	Podatkovna sabirnica koja se koristi za 8 bitni prijenos
8	DB1	H/L	
9	DB2	H/L	
10	DB3	H/L	
11	DB4	H/L	Podatkovna sabirnica za unos 4 i 8 bitnih podataka
12	DB5	H/L	
13	DB6	H/L	
14	DB7	H/L	
15	BLA	--	Pozadinsko svijetlo +5V
16	BLK	--	Pozadinsko svijetlo GND