Pametna lemna stanica

Majdenić, Ilija

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

Diplomski studij

PAMETNA LEMNA STANICA

Diplomski rad

Ilija Majdenić

Osijek, 2017.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 08.12.2017.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Ilija Majdenić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 826 R, 09.10.2015.
OIB studenta:	88743359567
Mentor:	Doc.dr.sc. Tomislav Keser
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Doc.dr.sc. Alfonzo Baumgartner
Član Povjerenstva:	Doc.dr.sc. Ivan Aleksi
Naslov diplomskog rada:	Pametna lemna stanica
Znanstvena grana rada:	**Procesno računarstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	Projektirati i izraditi sustav upravljanja temperaturom lemila koristeći principe brzog, kontroliranog i inteligentnog načina upravljanja sustavom zagrijavanja. Omogućiti automatizirano upravljanje stand-by režimom na temelju analize akceleriranog kretanja lemila u prostoru, tj. očitanja akcelerometra. Provesti funkcionalnu i komparativnu analizu sa postojećim uobičajenim sustavima.
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	08.12.2017.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum: 8.12.2017.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 13.12.2017.

Ime i prezime studenta:	Ilija Majdenić
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 826 R, 09.10.2015.
Ephorus podudaranje [%]:	2

Ovom izjavom izjavljujem da je rad pod nazivom: **Pametna lemna stanica**

izrađen pod vodstvom mentora Doc.dr.sc. Tomislav Keser

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih **koji su izričito priznati navođenjem literature i drugih izvora informacija.**
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je **bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.**

Potpis studenta:

Majdenić Ilija

SADRŽAJ

1. UVOD

Pri izradi nekog elektoničkog uređaja, popravaka ili mijenjanja određene komponente javlja se potreba za lemljenjem komponenti. U tu svrhu najčešće se koristi „meko lemljenje“ kod kojega je talište rastaljenog lema oko 280 °C i manje, te je niže od tališta osnovnog materijala koji se spaja. Da bi se omogučilo taljenje lema u tu svrhu se koristi lemilica. Postoje različiti tipovi lemilica, neke od njih su: štapna lemilica, lemna stanica, lemilice tzv.„pištolji“, USB lemilica, plinska lemilica, lemilica na vrući zrak i druge. S obzirom na neke nedostatke koje imaju standardne lemlice, „pametna lemna stanica“ nastoji otkloniti te nedostatke u vidu dobro kontrolirane temperature vrha lemne drške, prikaza temperature, odabira temperature, „stand-by“ režima rada pomoću akcelerometra, brzog zagrijavanja vrha lemilice, te udovoljiti zahtjevima elektroničara amatera ili profesionalca.

1.1. Zadatak i ciljevi rada

Zadatak ovog diplomskog rada je projektirati i izraditi sustav upravljanja temperaturom lemila koristeći principe brzog, kontroliranog i inteligentnog načina upravljanja sustavom zagrijavanja. Omogućiti automatizirano upravljanje stand-by režimom na temelju analize akceleriranog kretanja lemila u prostoru, tj. očitavanja akcelerometra. Provesti funkcionalnu i komparativnu analizu s postojećim uobičajenim sustavima.

2. SUSTAVI KONTAKTNOG LEMLJENJA

2.1. Lemne legure

Lem je topljiva metalna legura koja se koristi za izradu stalnih veza metalnih kontakata. Pri tome mora imati nižu točku taljenja nego materijal koji se spaja. Lem treba biti otporan na oksidacijske i korozivne učinke koji uništavaju spoj tijekom vremena. Također lem treba imati povoljna električna svojstva kao što je vodljivost. Uobičajene legure koje se koriste za lemljenje dijele se na legure kojima je točka tališta između 90 °C i 450 °C. Legure za lemljenje se dijele na legure koje pripadaju "mekom lemljenju" i legure koje pripadaju "tvrdom lemljenju", a točka tališta im je iznad 450 °C. Vrste koje se koriste kod mekog lemljenja dijele se na legure sa udijelom olova tzv. "olovni lem", legure bez olova tzv."bezolovni lem" i legure s jezgrom zaštitnog sredstva [1].

Olovni lem je komercijalno dostupan s koncentracijama kositra između 5 % i 70 % po masi. Što je veća koncentracija kositra, veća je i vlačna i smična čvrstoća lema. Legura koja se najčešće koristi za lemljenje u elektronici sadržava 60 % kositra (Sn) i 40 % olova (Pb) i ima točku taljenja 188 °C. Također zastupljena legura je legura sa 63 % kositra (Sn) i 37 % olova (Pb) i ima nižu točku taljenja, a iznosi 183 °C. U tablici su dane još neke legure koje u sebi sadrže olovo, a nalaze se u upotrebi [1].

***Tab 2.1**. Lemne legure koje sadrže olovo*

Sastav (%)	Talište (°C)	Upotreba
Sn=10 Pb=90	268-302	Koristi se za izradu automobilskih hladnjaka, spremnike za gorivo
Sn=32 Pb=68	253	Vodoinstalaterski lem
Sn=50 Pb=50	183-216	Lemljenje mjedi, brojila električne energije, plinomjera.
Sn=95 Pb=5	238	Vodovod i grijanje

Sn=62 Pb=36 Ag=2	179	Koristi se za lemljenje u elektronici. Udio srebra smanjuje topivost srebra kod elemenata koji imaju srebrne metalizirane kontakte (npr. SMD kondenzator)

Bezolovni lem za komercijalnu upotrebu može sadržavati kositar (Sn), bakar (Cu), srebro (Ag), bizmut (Bi), indij (In), cink (Zn), antimon (Sb). Bezolovni lem je zamjena za konvencionalne 60/40 Sn-Pb (kositar-olovo lemove) jer je od 1.srpnja 2006. institucija za otpad električne i elektroničke opreme (*eng*. European Union Waste Electrical and Electronic Equipment Directive (WEEE)) i institucija za ograničenje opasnih tvari (*eng*. Restriction of Hazardous Substances Directive (RoHS)) uvela na snagu zabranu uključivanja većih količina olova u većini potrošačke elektronike proizvedene u EU. Najkorištenija legura je Sn-Ag-Cu, naziva se još i "SAC". Većina bezolovnog lema ima 5 do 20 °C viša tališta od lemova koji sadržavaju olovo [1].

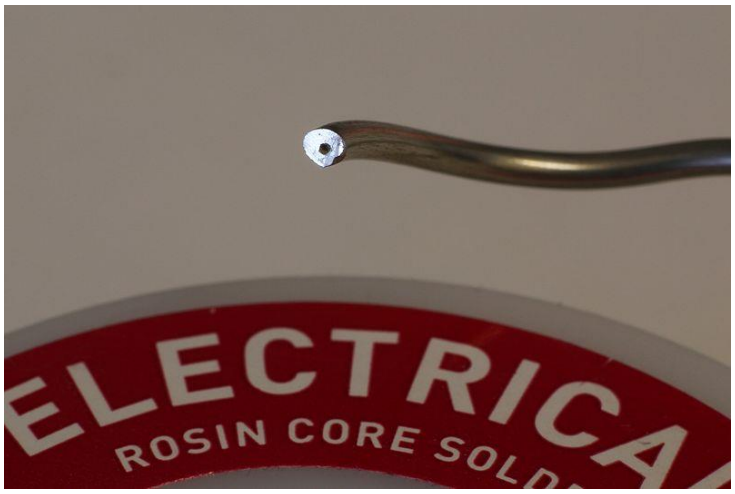
Tab. 2.2. *Lemne legure koje ne sadrže olovo*

Sastav (%)	Talište (°C)	Upotreba
Sn=90 Zn=7 Cu=3	200-222	Koristi se u proizvodnji kondenzatora kao zaštitni sloj od utjecaja elektromotornih sila i elektromagnetskih smetnji.
Sn=96,5 Ag=3,0 Cu=0,5	217-220	Legura koja se koristi u valnom lemljenju, selektivnom i dip lemljenju.
Sn=97 Cu=2,75 Ag=0,25	228-314	Visoka tvrdoča. Izrada radijatora, popravci radijatora, izrada vodovoda.

Slika 2.1. prikazuje razliku u obliku lemne legure sa olovom lijevo i bez olova desno.

Sl. 2.1. *Olovni i bezolovni lem* [2]

Lem sa jezgrom zaštitnog sredstva (*eng*. Flux-core solder) je redukcijsko sredstvo osmišljeno kako bi pomogao smanjiti oksidiranje na mjestima kontakta, poboljšati električni kontakt i mehaničku čvrstoću lemnih spojeva. Dva glavna tipa kiselina (*eng.* acid) i smola (*eng.* rosin). Kiselina se koristi kod lemljenja vodovodnih cijevi i ne smije se koristiti u elektronici. Jezgra sa zaštitnim sredstvom koristi se kod bezolovnih i olovnih lemova [1].



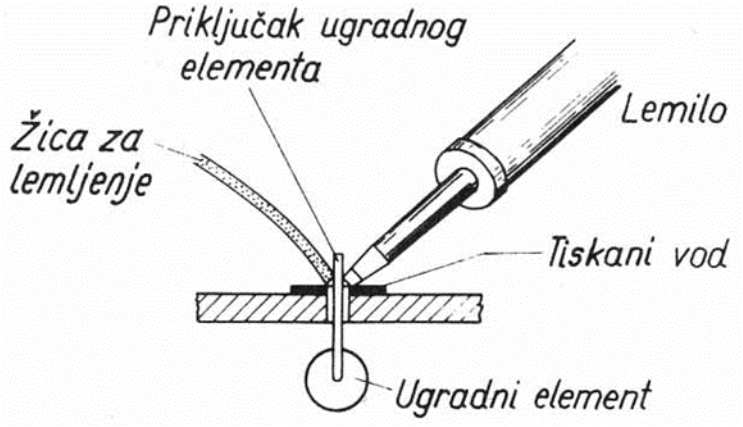
Sl. 2.2. *Lem sa jezgrom zaštitnog sredstva* [1]

2.2. Kontaktni lemni sustavi

Kontaktno lemljenje je proces spajanja dva metalna dijela pomoću lema. Za lemljenje se koriste metali i legure navedeni u poglavlju 2.1.

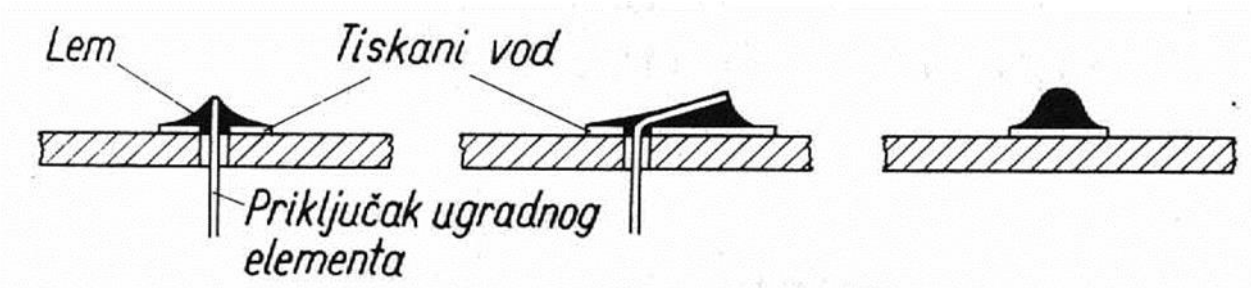
Lemljenje se može izvoditi ručno ili automatski. Kod automatskog lemljenja se na tiskanu elektroničku pločicu (PCB) nanosi solder maska koja predstavlja antikorozivnu zaštitu i zaštitu za kontrolirano nanošenje lemne taljevine i potom se potapa u bazen s rastaljenim lemom na par sekundi. Ručno, lemilicom lem se rastapa dok je u kontaktu sa lemnim mjestom. U nastavku je opisano ručno lemljenje.

Postupak ispravnog lemljenja prikazan je na slici 2.3.



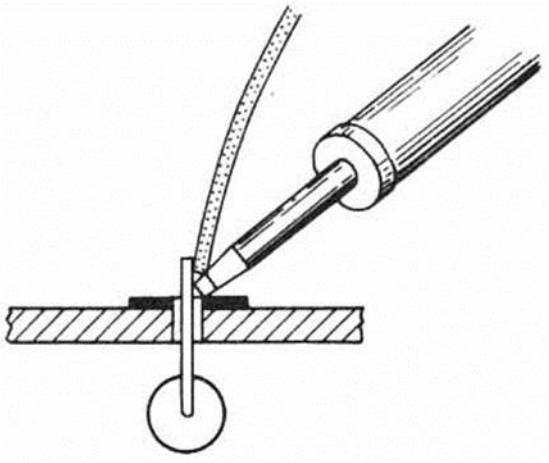
Sl. 2.3.*Postupak ispravnog lemljenja* [3]

Prvi korak je odmašćivanje i čišćenje površine koja se lemi. Slijedi nanošenje paste za lemljenje, zatim se vrhom lemila zagrijava priključak komponente, a ne žica za lemljenje. Zagrijani priključak komponente otopi pastu za lemljenje koja zalije čitavo mjesto spoja, te se razlije rastaljeni lem po mjestu spoja. Vrh lemilice se zadrži nakon toga još dvije do tri sekunde, dok se lem ne formira, tj. dobije finu srebrnu boju. Pri dodavanju lema potrebno je voditi brigu o tome da ne bude previše lema, ali ni premalo, već taman onoliko koliko je potrebno. Slika 2.4. prikazuje primjer dobro izvedenih lemnih mjesta [3].



Sl. 2.4. *Prikaz dobro izvedenih lemnih mjesta* [3]

Kod neispravnog lemljenja pogreška se čini kada se na vrh lemilice izravno prisloni žica za lemljenje, pa se onda rastaljena kapljica lema stavi na mjesto gdje je potrebno ostvariti spoj. Takav spoj ponekad zna biti loš jer postoji vjerojatnost da je takav spoj ustvari tzv. „hladni lem“ kojeg je prouzročila niska temperatura lemljenja pa naizgled formirana kapljica lema izgleda u redu, ali zapravo se unutar kapljice lem nije zavukao u površinu kontakata. Stoga je takav lem električki nevodljiv ili loše vodljiv. Sprečavanje „hladnih lemova“ izbjegava se dužim držanjem vrha lemilice na mjestu lemljenja, ali ne predugo ako se leme poluvodički elementi koji su osjetljivi na temperaturu. Preporučljivo je 350 °C držati oko dvije sekunde [3].



Sl. 2.5. *Prikaz neispravnog lemljenja* [3]

Sl. 2.6. *Prikaz loših lemnih točaka* [3]

Slika 2.6. prikazuje loše lemne točke, vidi se da se lemne točke dodiruju što nije dobro. Prilikom lemljenja poželjno je provjeriti jeli se dodiruju točke provlačenjem listića papira između njih ili pomoću multimetra tako da se ispita otpor između tih točaka [3].

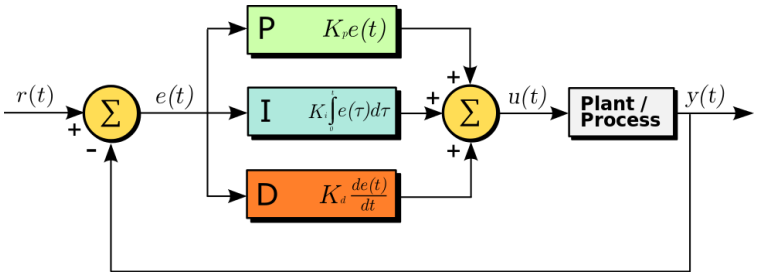
2.3. Lemna stanica

Standardne lemne stanice su uređaji za lemljenje s mogućnosti podešavanja temperature grijaćeg tijela, poneke stanice imaju i mogućnosti očitavanja temperature grijaćeg tijela lemne drške. Kao zadatak koji je stavljen pred pametnu lemnu stanicu potrebno je postići inteligentan način upravljanja lemne stanice, ispis i dijagram temperature na LC pokazniku, te ostvariti komunikaciju s računalom.

Komponente koje se koriste za izradu pametne lemne stanice su sljedeće: Arduino Mega 2560, 128x64 LC pokaznik, HC-05 bluetooth modul, lemna drška HQ SOLDER/IR, MOSFET tranzistor IRFZ44N, izvor napajanja AC-DC 220V-24V, tipkala, kućišta za osigurače na panelima, osigurači, otpornici, kondenzatori, akcelerometar MPU6500, operacijsko pojačalo LM358N, potenciometar, led diode, diode 1N4007, regulator napajanja LM7806 i DC-DC LM2596 pretvarač. Sve ove komponenete detaljnije su opisane u poglavlju 3.2. Sustav i struktura lemne stanice.

Inteligentan način upravljanja lemne stanice ostvaruje se pomoću PID regulatora i žiroskopa pomoću kojeg se detektira položaj lemne drške u prostoru te na temelju promjene položaja donosi odluka o stand-by režimu rada.

PID regulator jedan je od najkorištenijih regulatora u industriji, a sastoji se od tri člana, tj. parametra (proporcionalnog, integracijskog i derivacijskog). Blok dijagram PID regulatora je prikazan na slici 2.7.



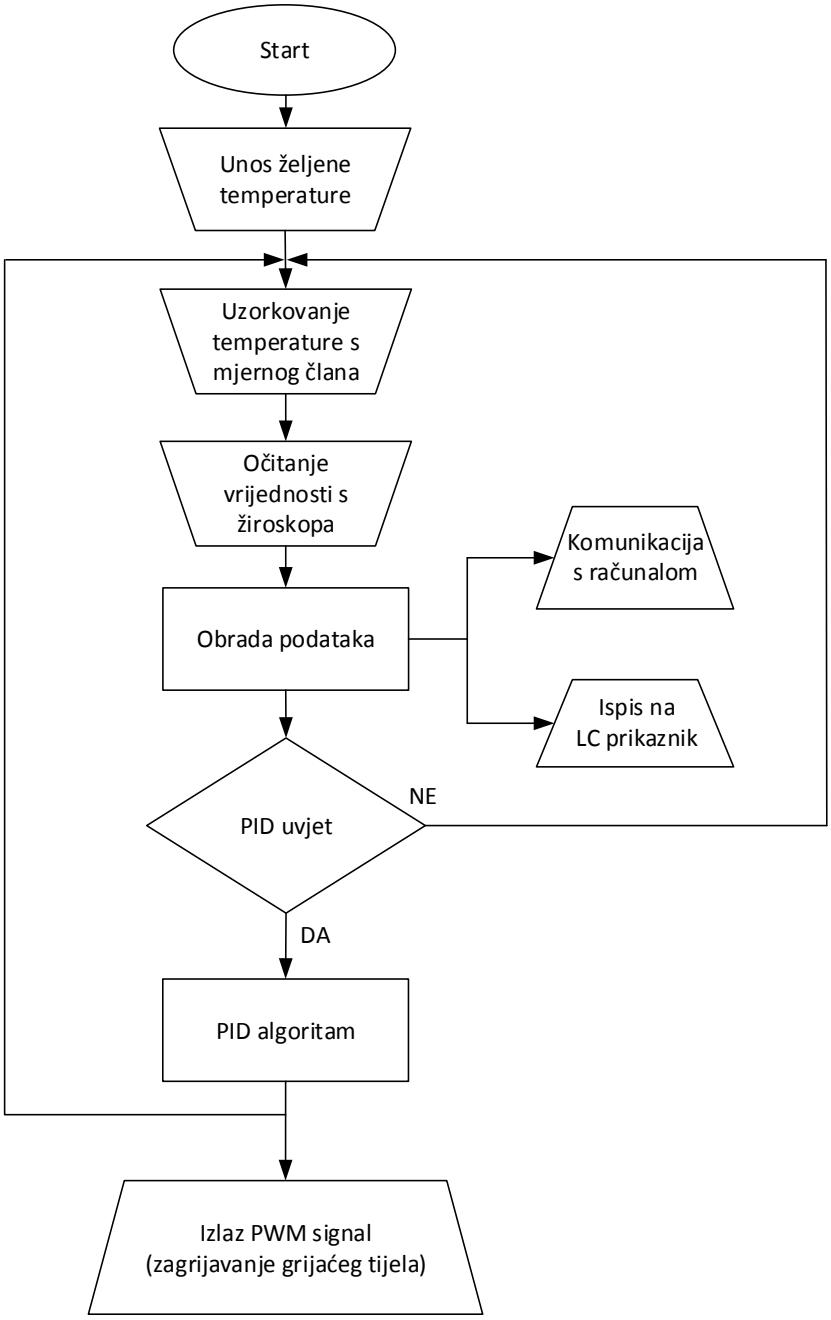
Sl. 2.7. *Blok dijagram PID regulatora* [4]

Zadaća PID regulatora je konstantno izračunavanje pogreške *e(t)* na temelju razlike zadane ulazne vrijednosti *r(t)* i vrijednosti mjerene veličine iz procesa *y(t)*. Pomoću zadanih parametara i regulacijske pogreške dobiva se upravljačka veličina *u(t)* koja nastoji minimizirati pogrešku *e(t)* kako bi mjerena veličina *y(t)* što vjernije pratila ulaznu zadanu vrijednost *r(t).* Ulazna vrijednost *r(t)* zadaje se pomoću tipkala na lemnoj stanici ili pomoću računalne aplikacije koja komunicira bluetooth uređajem s mikrokontrolerom. Upravljačka veličina *u(t)* je PWM (pulsno širinski moduliran) signal pomoću kojega se upravlja tranzistorom koji propušta izvor napajanja u ovisnosti o vrijednosti PWM signala i na taj način zagrijava grijaće tijelo lemne drške. Mjerena procesna veličina *y(t)* dobiva se s temperaturnog mjernog člana lemne drške. Zadavanje željene temperature vrši se pomoću pet tipkala, od čega su tri za „grubo“ podešavanje u iznosu 180 °C, 280 °C i 380°C i dva tipkala za „fino“ podešavanje.

3. REALIZACIJA SUSTAVA

3.1. Funkcije i zadaci lemne stanice

Funkcije i zadaci koje obavlja lemna stanica prikazuju se pomoću dijagrama toka glavnog algoritma upravljanja, te je moguće na osnovu njega dobiti vizualni uvid u sustav lemne stanice.



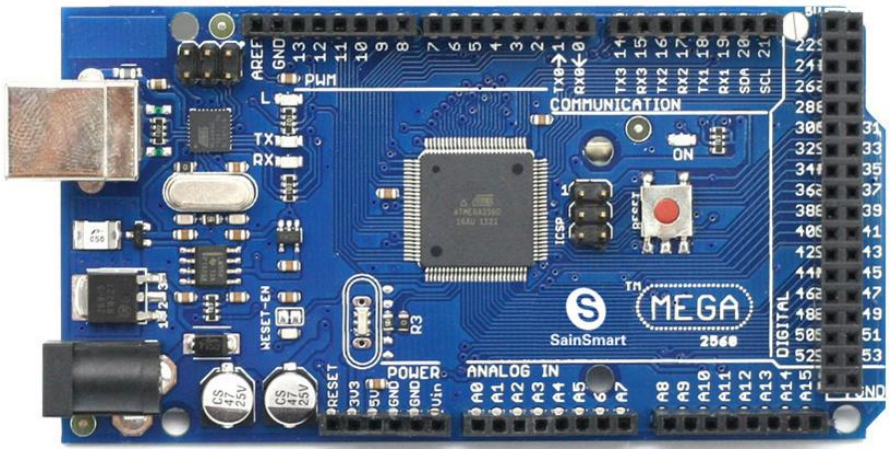
Sl. 3.1. *Algoritam upravljanja lemne stanice*

Slika 3.1. opisuje algoritam upravljanja lemne stanice. Nakon što se priključi lemna stanica na izvor napajanja i uključi sklopka „on/off“ zadaje se željena temperatura kojom se želi lemiti. U sljedećem koraku uzorkuje se temperatura u određenom vremenskom intervalu s mjernog člana koji se nalazi u lemnoj dršci, ta veličina služi za povratnu informaciju PID regulatoru na osnovu koje on uspoređuje sa željenom zadanom temperaturom. Očitavanje vrijednosti sa žiroskopa služi određivanju stanja u kojemu se nalazi lemna drška, tj. prati se aktivnost promjene gibanja lemne drške u određenom vremenu koje iznosi 120 sekundi, ako nema promjene gibanja lemne drške gasi se algoritam upravljanja i grijaće tijelo lemne drške prestaje sa zagrijavanjem. Ukoliko se želi ponovno pokrenut grijanje potrebno je odabrat temperturu te lemna stanica započinje algoritam upravljanja PID regulatora i zagrijavanje grijaćeg tijela lemne drške. Ovaj ciklus se ponavlja stalno i uspoređuje zadanu temperaturu s mjerenom vrijednošću. Vrijednosti temperatura se ispisuju i iscrtavaju na LC pokaznik i šalju računalu.

3.2. Sastav i struktura lemne stanice

U ovom poglavlju opisani su elementi lemne stanice, njihove karakteristike i uloga koju imaju u lemnoj stanici. Elementi su prethodno nabrojani u poglavlju 2.3. Lemna stanica.

Arduino Mega 2560



Sl. 3.2. *Arduino Mega 2560* [5]

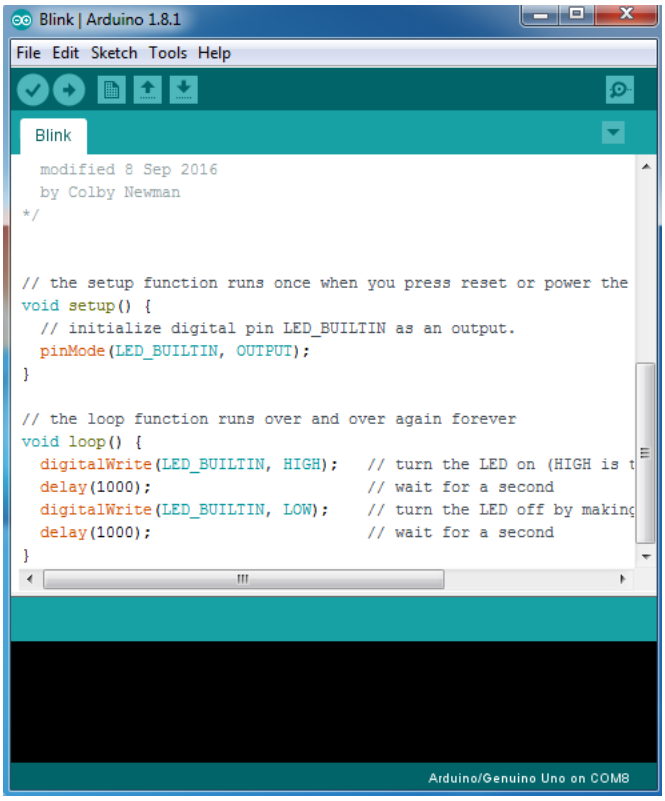
Arduino Mega 2560 je mikrokontroler baziran na atmelovom Atmega2560 čipu. Sadržava 54 digitatalna ulazna i izlazna pina od čega njih 15 se mogu iskoristiti kao PWM izlazi, 16 analognih ulaznih pinova, 4 UART-a (serijskih portova), I2C komunikacijski portovi, SPI

komunikacijski portovi, 16 MHz kristalni oscilator, USB konektor, konektor napajanja, reset tipkalo. Arduino Mega 2560 je središnjica „mozak“ ovoga uređaja koji obrađuje sve informacije i u konačnici upravlja cjelokupnom lemnom stanicom.

Tab. 3.1. *Tehničke specifikacije Arduino Mege 2560* [5]

Mikrokontroler	ATmega2560
Radni napon	5 V
Ulazni napon (preporučeni)	7-12 V
Ulazni napon (granični)	6-20 V
Digitalni U/I pinovi	54 (15 PWM)
Analogni ulazni pinovi	16
DC struja po I/O pinu	40 mA
DC struja za 3,3V pinove	50 mA
Flash memorija	256 KB od kojih 8KB koristi bootloader
SRAM	8 KB
EEPROM	4 KB
Radni takt	16 MHz

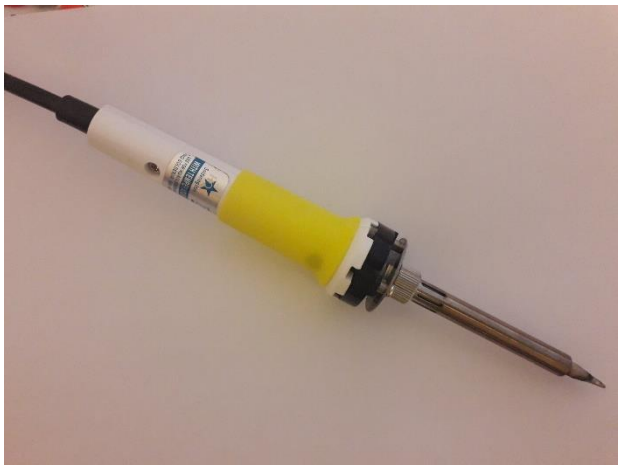
Kao razvojno okruženje Arduino Mega 2560 koristi Arduino IDE (*eng*. Integrated Development Environment), koji je baziran na C programskom jeziku. Arduino IDE je „open-source“ te postoje mnoge biblioteke koje nisu nužno napisane od strane proizvođača, također Arduino IDE podržavaju razni operacijski sustavi kao što su Windows, Linux i Mac.



Sl. 3.3. *Razvojno okruženje Arduino IDE* [6]

Lemna drška HQ SOLDER/IR

Lemna drška u ovome radu čini osnovni element, tj. alat pomoću kojega se vrši kontaktno lemljenje. Ova lemna drška je dosta popularna i rasprostranjena zbog svojstva dobrog zadržavanja topline i mogućnosti zamjene grijaćeg tijela i vrha lemne drške.

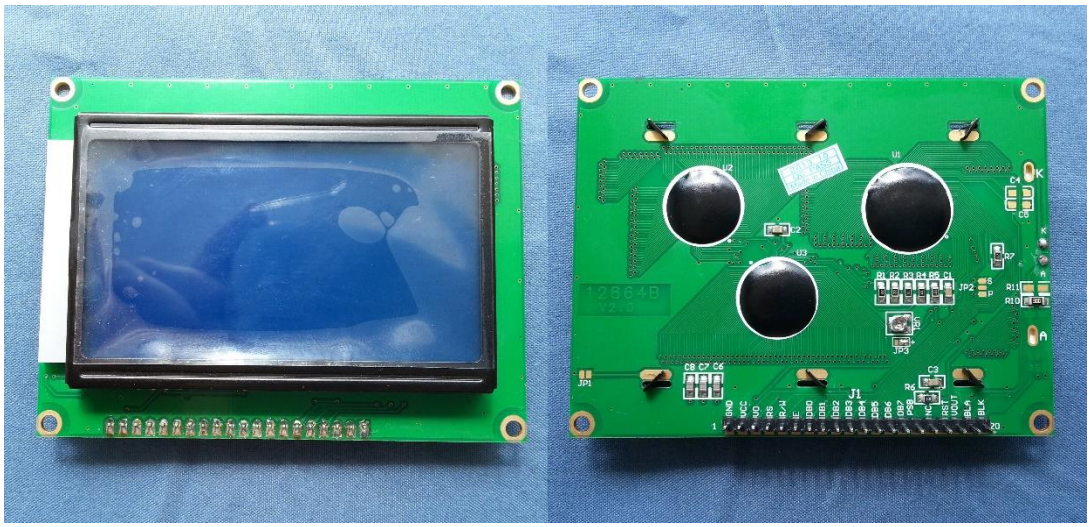


Sl. 3.4. *Lemna drška HQ SOLDER/IR*

Sastoji se od peterožilnog spojnog kabela (*eng*. Cord Assembly) u kojemu crveni i plavi vodič pripadaju temperaturnom mjernom članu koji je u ovom slučaju termopar E-tipa, bijeli i crni pripadaju keramičkom grijačem elementu i zeleni ESD (*eng*. Electrostatic discharge) vodič koji služi za zaštitu od statičkog elektriciteta, čuva sve poluvodičke elemente koji su osjetjivi na statički elektricitet. Sadrži i konektor kojim se povezuje na lemnu stanicu te vrha lemne drške (*eng*. Tip) koji se navlači na vrh grijaćeg tijela.

128x64 LC pokaznik

Služi za prikaz temperature grijaćeg tijela, kao i zadanu temperaturu, te iscrtava dijagram temperature na zaslonu pokaznika. Pokaznik se sastoji od kontrolera ST7920, te pinova pomoću kojih je moguće ostvariti serijsku i paralelnu komunikaciju arduina i LC pokaznika.



Sl. 3.5. *128x64 LC pokaznik* [9]

Izvor napajanja

U radu se koriste dva ovakva izvora napanja povezana u paralelu čime se dobiva izvor napajanja 24 V, sa strujom većom od 6 A koliko je na samo jednom napajanju, a koja je potrebna za početno zagrijavanje grijaćeg tijela lemne drške. Struja pojedinačnog napajanja iznosi 6 A, a snaga 100 W.

Sl. 3.6. *Izvor napajanja AC-DC 220V-24V* [10]

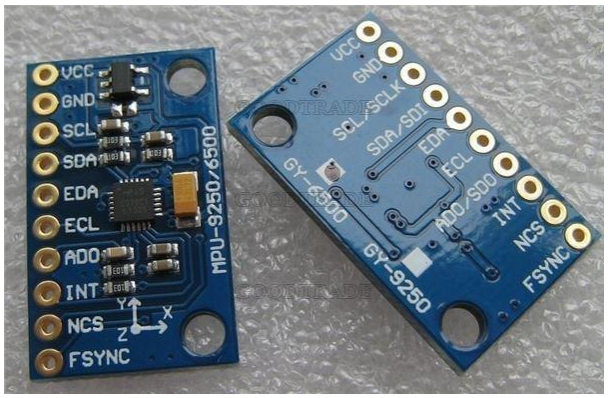
DC-DC pretvarač LM2596

DC-DC pretvarač ima svrhu napajanja Arduino Mega 2560 platforme sa 9 V, te napajanje operacijskog pojačala LM358N pomoću regulatora napajanja LM7806. Vrijednost ulaznog napona ovog pretvarača je u rasponu od 4 V do 35 V, a vrijednost izlaznog napona od 1.23 V do 30 V, dok je maksimalna ulazna struja 3 A. Na slici 3.7. prikazan je DC-DC LM2596 pretvarač.

Sl. 3.7. *DC-DC LM2596 pretvarač* [15]

Žiroskop MPU6500

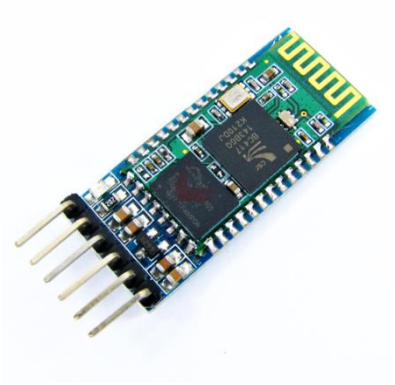
Služi za određivanje promjene položaja lemne drške u prostoru, te na temelju te promjene algoritam lemne stanice uključuje ili isključuje lemilicu. Senzor sadržava akcelerometar i žiroskop u jednom čipu, sadrži 16-bitni ADC (analogno digitalni konverter) za svaki kanal x, y i z. Koristi I2C komunikaciju s Arduinom.



Sl. 3.8. *Žiroskop MPU6500* [11]

Bluetooth modul HC-05

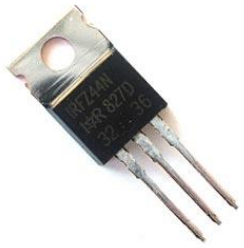
Služi za komunikaciju lemne stanice s računalom bežićnim putem. Spaja se na pinove za serijsku komunikaciju R_X i T_X na Arduino i dovodi mu se napajanje u iznosu 5 V. Vrijednost napona logičke razine kojom komunicira HC-05 bluetooth modul je 3,3 V za logičku „1“, pri tome je potreban logički konverter ili naponsko dijelilo koje će spustiti napon koji se spaja sa T_X pina Arduina na R_X pin HC-05 bluetooth modula.



Sl. 3.9. *HC-05 bluetooth modul* [12]

Tranzistor IRFZ44N

Na MOSFET tranzistor dovodi se PWM signal s Arduina i u ovisnosti o visini napona i propusnosti tranzistora, povećava se napon na grijaćem tijelu s izvora napajanja. Ovaj tranzistor ima vrlo visoko ograničen V_{DS} (*eng*. Drain-source voltage) napon koji iznosi 55 V i vrlo visoku ograničenu struju I_D (*eng*. Drain current) i iznosi 49 A.



Sl. 3.10. *MOSFET tranzistor IRFZ44N* [13]

Operacijsko pojačalo LM358N

Operacijsko pojačalo LM358N ima ulogu diferencijalnog pojačala koje pojačava mjereni napon s temperaturnog mjernog čalana (termopara) koji se kreće u granicama oko dvadesetak mV na raspon vrijednosti od 0 do 5 V pogodne za analogno očitavanje. Ovaj čip u sebi sadrži dva operacijska pojačala i moguće ga je napajati s izvorom napajanja od 3 V do 32 V pri čemu pad napona na izlazu operacijskog pojačala iznosi približno oko 1.5 V.

Sl. 3.11. *Operacijsko pojačalo LM358N* [14]

Ostale komponente

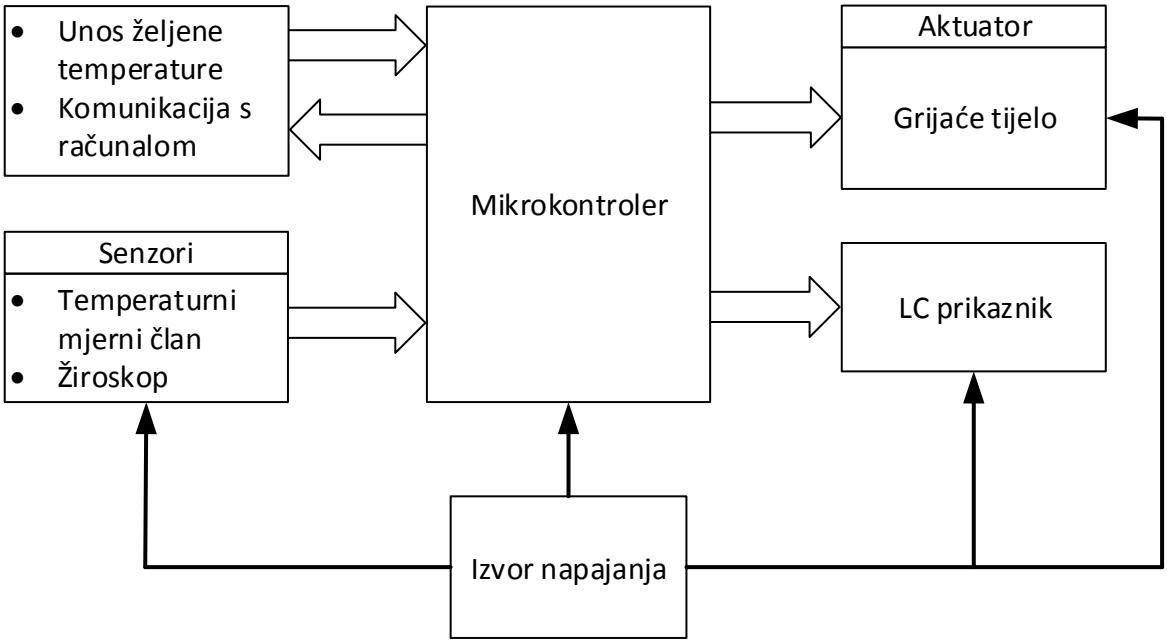
Ostale komponente koje se koriste u radu dane su u tablici 3.2., a njihove vrijednosti i nazive moguće je vidjeti shemi koja je dana poglavlju 3.3. Shema i funkcionalni ustroj.

Tab. 3.2. *Ostale komponente*

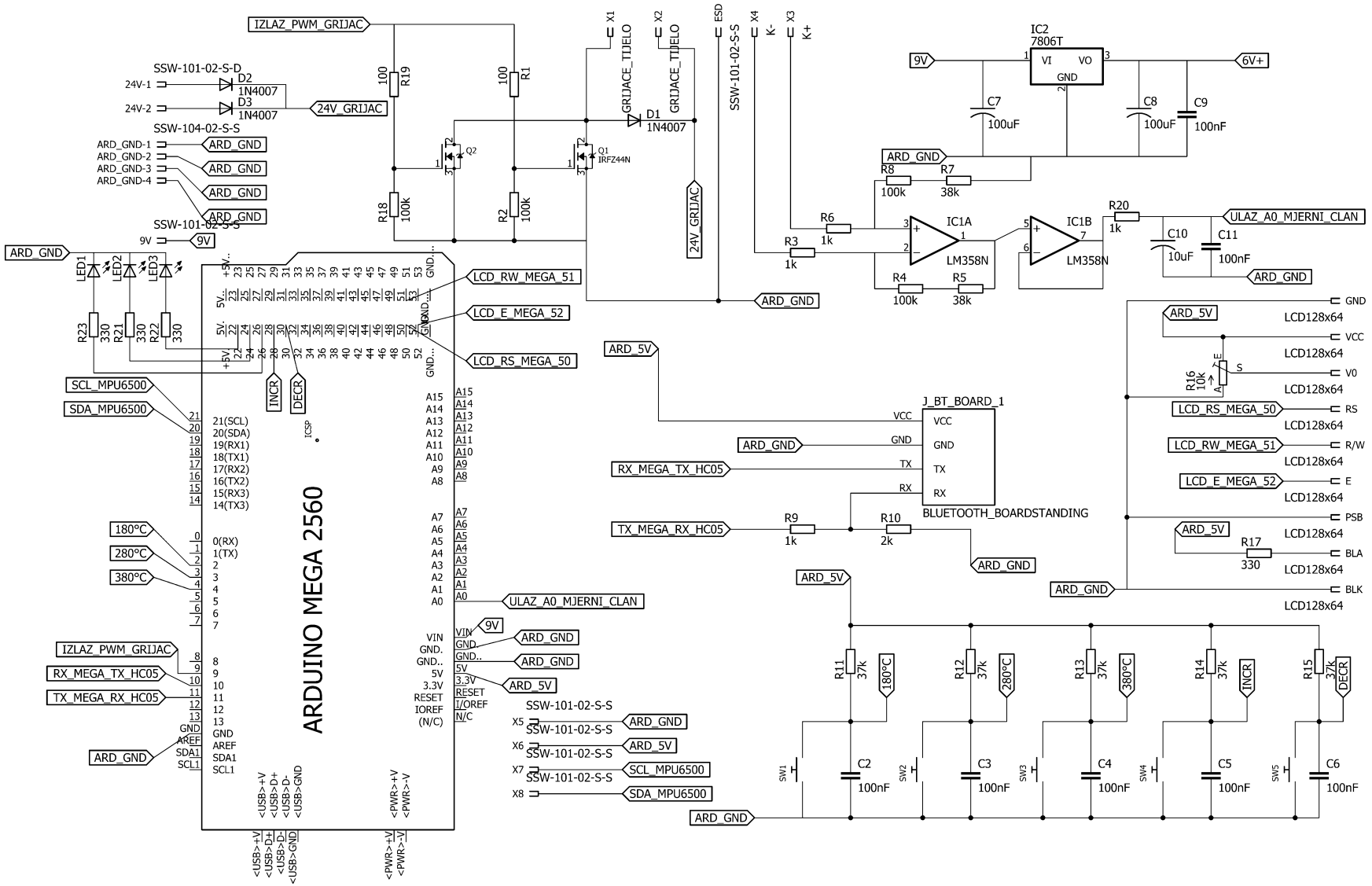
		
Otpornici	Kondenzatori	Diode
		
Regulatori	Osigurači	Kućišta osigurača
		
Svijetleće diode	Tipkala	Držaći LED
		
Konektor	On/Off sklopka	Kabel za napajanje

3.3. Shema i funkcionalni ustroj

Središnjicu sustava čini mikrokontroler Arduino Mega 2560 čija je uloga povezivanja ulaza i izlaza lemne stanice, komunikacije s računalom, obrađivanje informacija te upravljanje cjelokupnim sustavom. Unos željene temperture unosi se pomoću tipkala na lemnoj stanici i pomoću računala koje komunicira s lemnom stanicom putem bluetootha. Ulazi senzora sastoje se od temperaturnog mjernog člana i žiroskopa. Temperaturni mjerni član daje informaciju upravljačkom sustavu o mjerenoj vrijednosti, tj. koja je vrijednost temperature grijaćeg tijela u odnosu na željenu vrijednost. Žiroskop služi za detekciju pomaka lemne drške u prostoru. Izlazi se sastoje od aktuatora, LC pokaznika i komunikacije s računalom. Aktuator predstavlja grijaće tijelo, koje se zagrijava u ovisnosti o izlazu upravljačkog sustava. LC pokaznik služi za ispisivanje vrijednosti temperature i iscrtavanju dijagrama. Komunikacija s računalom omogućuje prikaz temperature na računalu, te mogućnost upravljanja s lemnom stanicom pomoću računala. U nastavku je dan funkcijski blok dijagram na slici 3.12., te cjelokupna shema lemne stanice na slici 3.13



***Sl. 3.12**. Funkcijski blok dijagram sustava*



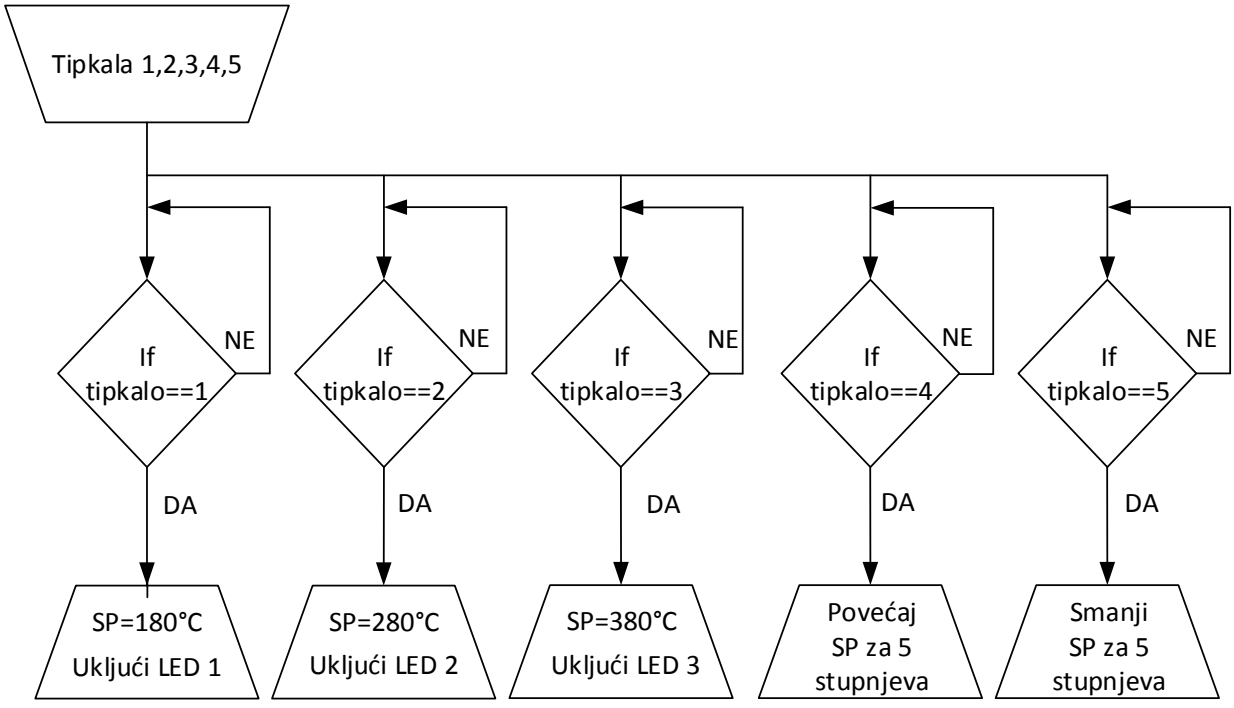
Sl. 3.13. *Shema lemne stanice*

3.4. Programska podrška

Programska podrška pomoću koje se ostvaruje implementacija ovog projekta je Arduino IDE te Visual Studio C++. Upravljački i najvažniji dio projekta realiziran je u Arduino IDE okruženju pomoću kojeg Arduino Mega 2560 upravlja cjelokupnim sustavom lemne stanice te vrši komunikaciju s računalnom aplikacijom o čemu se više informacija nalazi u potpoglavlju 3.5. Računalna aplikacija izrađena je u Visual C++ programskom okruženju i svrha joj je prikupljanje i vizualizacija podataka te upravljanje pomoću računala.

U potpoglavlju 3.1. prikazan je glavni algoritam upravljanja lemnom stanicom na slici 3.1. u nastavku je dano opširnije objašnjenje pojedinih blokova.

Nakon uključivanja lemne stanice pomoću sklopke „On/Off“ odabire se u unos željene temperaturne vrijednosti pomoću tipkala ili pomoću računalne aplikacije koja je detaljnije opisana u potpoglavlju 3.5. Slika 3.14. prikazuje dijagram toka odabira temperaturne vrijednosti pozivom funkcije *tipkala_led()*. Tipkalima 1, 2 i 3 odabire se „SP“, tj. željena temperatura, a tipkalima 4 i 5 povećanje odnosno smanjivanje željene temperature za 5 °C.



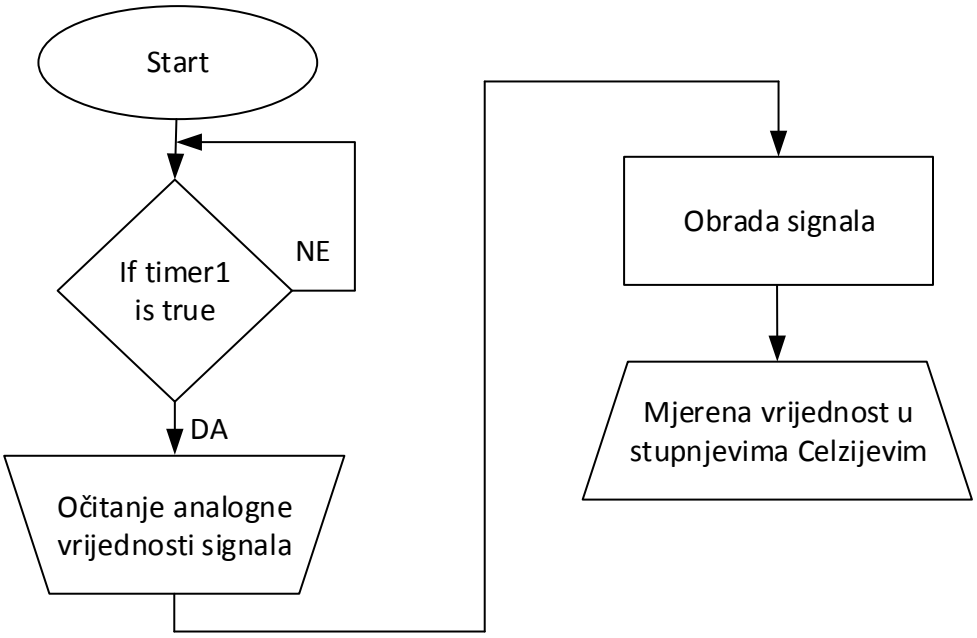
Sl. 3.14. *Dijagram toka odabira temperature pomoću tipkala*

Tijekom cikličkog izvršavanja *loop()* petlje uzorkuje se mjereni signal svakih 100ms kada se aktivira timer1 te se sprema u polje pomoću funkcije *put_temp()* i računa srednja vrijednost posljednjih 19 mjernih uzoraka i trenutnog uzorka pomoću funkcije *moving_avg()* kako bi se dodatno filtrirali mjerni podaci. Očitana mjerena vrijednost se nalazi u rasponu 10-bitnog analognog digitalnog pretvarača (ADC) od 0 do 1023 te ju je potrebno pretvoriti u temperaturni iznos. Stoga se mjerena vrijednost pretvara u iznos napona na analognom ulazu množenjem sa $4.88e^{-3}$, nakon toga se pomoću sljedećih formula dobiva temperatura:

$$skalirajuci\_faktor = \frac{\left(\frac{1}{pojacanje}\right)}{E\_tip\_osjetljivost} \tag{3 - 1}$$

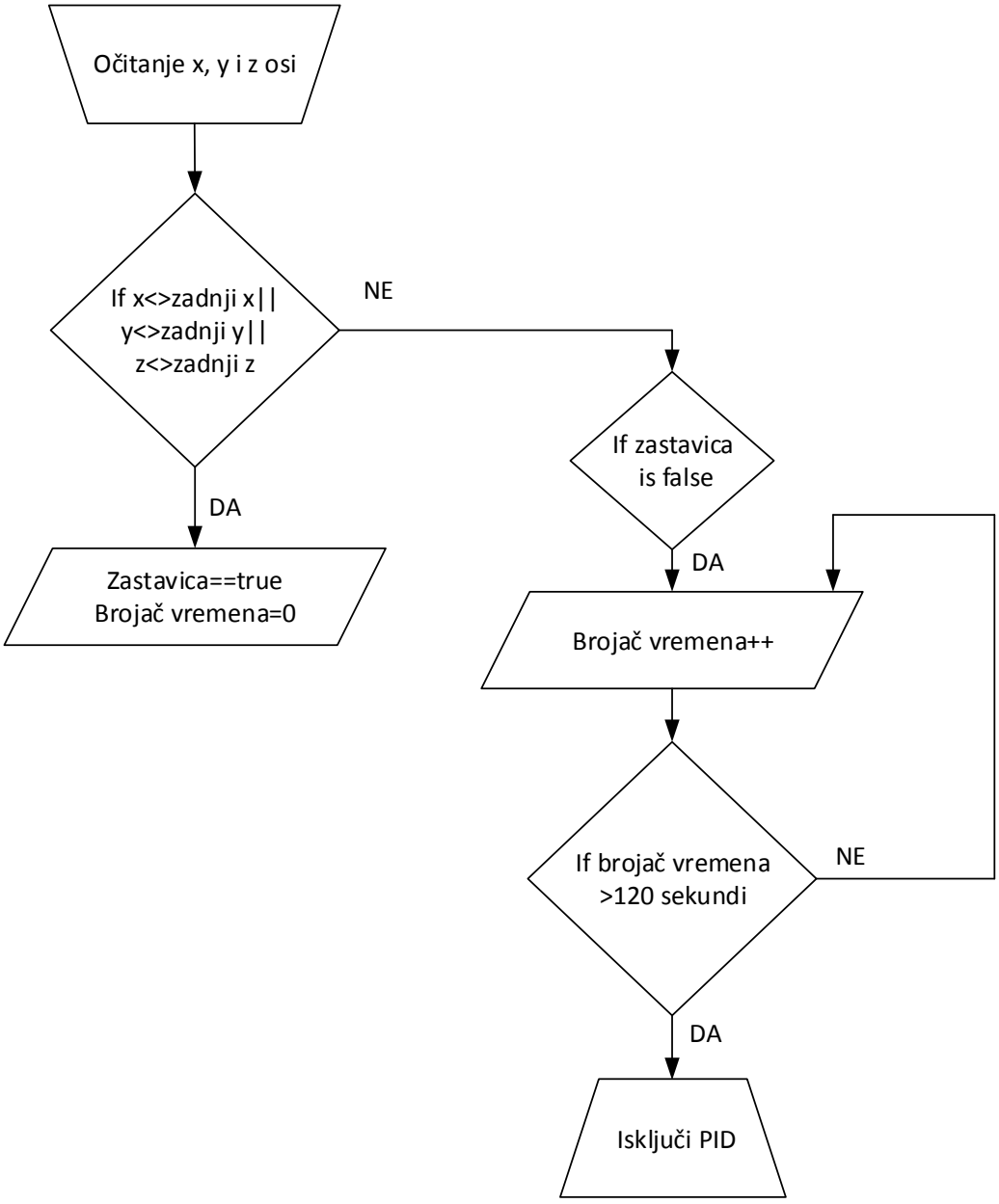
$$MV\_mjereno = 25 + skalirajuci\_faktor * napon \tag{3 - 2}$$

Pojačanje iz izraza (3 - 1) i osjetljivost mjernog člana, tj. termopara dobiveni su testiranjem o čemu više u poglavlju 4. Testiranje i rezultati. Prema testiranju pojačanje iznosi 143, a osjetljivost mjernog člana je 68.5 µV/°C. Konačno iz formule (3 - 2) moguće je odrediti temperaturu u °C.



Sl. 3.15. *Dijagram toka uzorkovanja mjernog signala*

Očitanje sa žiroskopa MPU6500 odvija se pozivom funkcije *ziroskop()* svakh 4 sekunde aktiviranjem timera3. Očitavaju se vrijednosti x, y i z osi, tj. njihove promjene u prostoru, te nakon dodatnog filtriranja *„moving average filtrom“* uspoređuju s prošlom vrijednošću i ukoliko nema promjene vrijednosti na x, y ili z osi unutar 120 sekundi isključuje se PID algoritam.



Sl. 3.16. *Dijagram toka algoritma za žiroskop*

Aktiviranjem timera4 svakih 250 ms vrši se ispis i iscrtavanje dijagrama postavljene i mjerene temperature pomoću funkcije *lcd()* na LC pokaznik. Sličan program se odvija na računalnoj aplikaciji, o čemu je detaljnije opisano u potpoglavlju 3.5. potpoglavlju. Slika 3.17. prikazuje ispis i iscrtavanje dijagrama pomoću LC pokaznika.

Sl. 3.17. *LC pokaznik*

U konačnici, da bi se ostvarilo održavanje zadane temperature potreban je za to efikasan način. U ovom projektu to je učinjeno pomoću PID regulatora čija je svrha unaprijed opisana u potpoglavlju 2.3.

Jednadžba kontinuiranog PID regulatora:

$$u(t) = K_R\left[e(t) + \frac{1}{T_I}\int_0^t e(\tau)d\tau + T_D\frac{de(t)}{dt}\right] \qquad (3\text{ - }3)$$

Gdje je K_R proporcionalno pojačanje regulatora, T_I integralna vremenska konstanta, T_D derivacijska vremenska konstanta. Ovi navedeni parametri imaju fizikalno značenje što znači da su promjenjivi, te ih je moguće podešavati kako bi se dobilo što bolje vladanje i stabilnost sustava [16].

Međutim da bi se PID regulator uspješno implementirao u mikrokontroler potrebno je provesti njegovu diskretizaciju. Postoji više postupaka diskretizacije neki od najpoznatijih su postupak unaprijedne diferencije, unazadne diferencije i Tustinov postupak. Najčešće se primjenjuje postupak unazadne diferencije jer najbolje aproksimira derivacijski član za sve iznose derivacijske vremenske konstante T_D. U nastavku su dane diskretizirane jednadžbe pojedinih članova postupkom unazadne diferencije.

P član:

$$u_P(k) = K_R e_P(k) \tag{3 - 4}$$

k je oznaka diskretnog vremena, a T vrijeme uzorkovanja.

I član:

$$u_I(k) = u_I(k-1) + \frac{K_R T}{T_I} e(k) \tag{3 - 5}$$

D član:

$$u_D(k) = \frac{T_D}{T_D + vT} u_D(k-1) + \frac{K_R v T_D}{T_D + vT}[e_D(k) - e_D(k-1)] \tag{3 - 6}$$

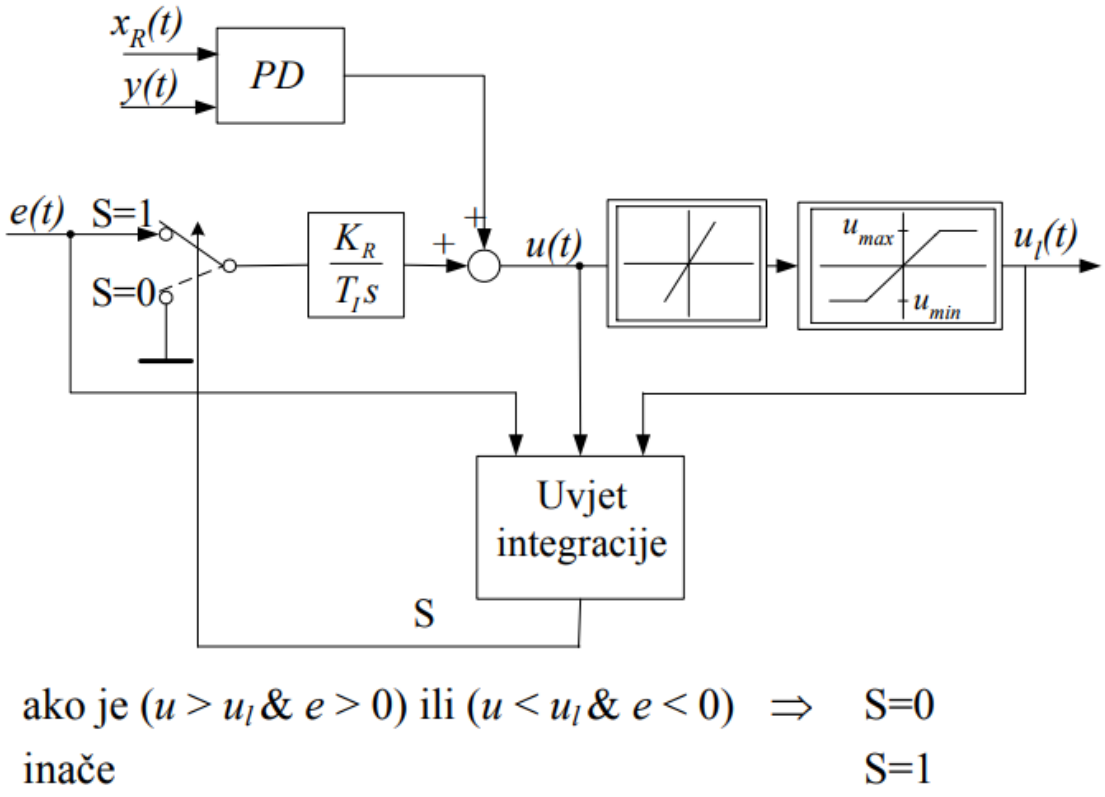
Jednadžba (3 - 6) je diskretizirana jednadžba realnog derivatora gdje je v je skalar u rasponu vrijednosti od 5 do 20. Filtriranjem idealnog derivatora prvog reda vremenske konstante T_D/v dobiven je realni derivator [16].

Zbrojem sva tri člana dobiva se PID regulator:

$$u(k) = u_P(k) + u_I(k) + u_D(k) \tag{3 - 7}$$

U algoritmu je potrebno još zadati vrijeme uzorkovanja T koje iznosi 0.1 s. Potrebno je odrediti i ograničenja, tj. granice u kojima će PID regulator upravljati svojim izlazom. Granice koje su postavljene iznose 0 za najmanju vrijednost i 255 za najveću vrijednost, a to je ujedno i raspon PWM signala na izlazu iz PID regulatora. S obzirom da PID regulator sadrži integralni član on ulaskom u ograničenje može poprimiti velike vrijednosti pa kako se pogreška, tj. regulacijsko

odstupanje smanjuje on sporo izlazi iz ograničenja i premašuje zadanu vrijednost, što može uzrokovati jako velike oscilacije i nestabilan sustav. Ovaj problem se naziva „efekt zaleta“ (*eng*. Wind-up) i moguće ga je riješiti primjenom uvjetnog integriranja koje je dano na slici 3.18. [16]



Sl. 3.18. *Uvjetno integriranje* [16]

Da bi PID regulator zadovoljavajuće obavljao svoj „posao“ potrebno je odrediti i odgovarajuće vrijednosti parametara. Parametri koji se koriste su: K_R=10, T_I=35 s i T_D=0.7s i određeni su eksperimentalnim putem, više o testiranju u poglavlju 4.

PID algoritam se izvršava pozivom funkcije *PID_racunanje()* aktivacijom timera1 svakik 100 ms.

PID_racunanje() funkcija i njen opis dan je u nastavku.

void PID_racunanje()

{

```
error=SP_zadano-MV_mjereno;//računanje pogreške, tj. regulacijskog odstupanja
P_output=Kr*error;//računanje proporcionalnog člana
if((output_1>max_out & error>0) || (output_1<max_out & error<0)) //uvjetno integriranje
{
I_output=0;//postavi vrijednost I člana na nulu, ako je uvjet zadovoljen
}else{
I_output=last_I_output+(Kr*T_sample/Ti)*error;//računanje integralnog člana
last_I_output=I_output;//spremanje koraka u(k-1)
}
deltaError=error-lastError;//razlika trenutne pogreške od posljednje
//računanje derivacijskog člana
D_output=((Td/(Td+v*T_sample))*last_D_output)+((Kr*v*Td)/(Td+v*T_sample))*deltaError;
output = P_output+I_output+D_output;//rezultat zbroja sva tri člana
output_1=output;
if (output>max_out)//ograničavanje najvećeg izlaza
{
output=max_out;
}
if (output<min_out)//ograničavanje najmanjeg izlaza
{
output=min_out;
}
analogWrite(pwm_output,output);//slanje upravljačkog signala, tj. PWM signala
last_D_output=D_output;
lastError=error;
```

}

3.5. Komunikacija i podešavanje

Komunikacija lemne stanice i računalne aplikacije se odvija putem bluetooth HC-05 modula. Za serijsku komunikaciju koristi se već gotova biblioteka „*SoftwareSerial.h*“ koja omogućuje slanje i primanje podataka Arduino Mega 2560 platforme pri čemu je potrebno odabrati pinove za čitanje RX i slanje TX, te brzinu prijenosa podataka (*eng*. Baud rate) koja da bi komunikacija bila uspješna mora biti jednako postavljana na oba uređaja koja komuniciraju. Jedna od najčešće korištenih brzina za prijenos podataka za jednostavne stvari iznosi 9600 bps (*eng*. bits-per-second).

Očitanje u Arduino IDE se odvija konstantno u *loop()* petlji prema sljedećem kodu:

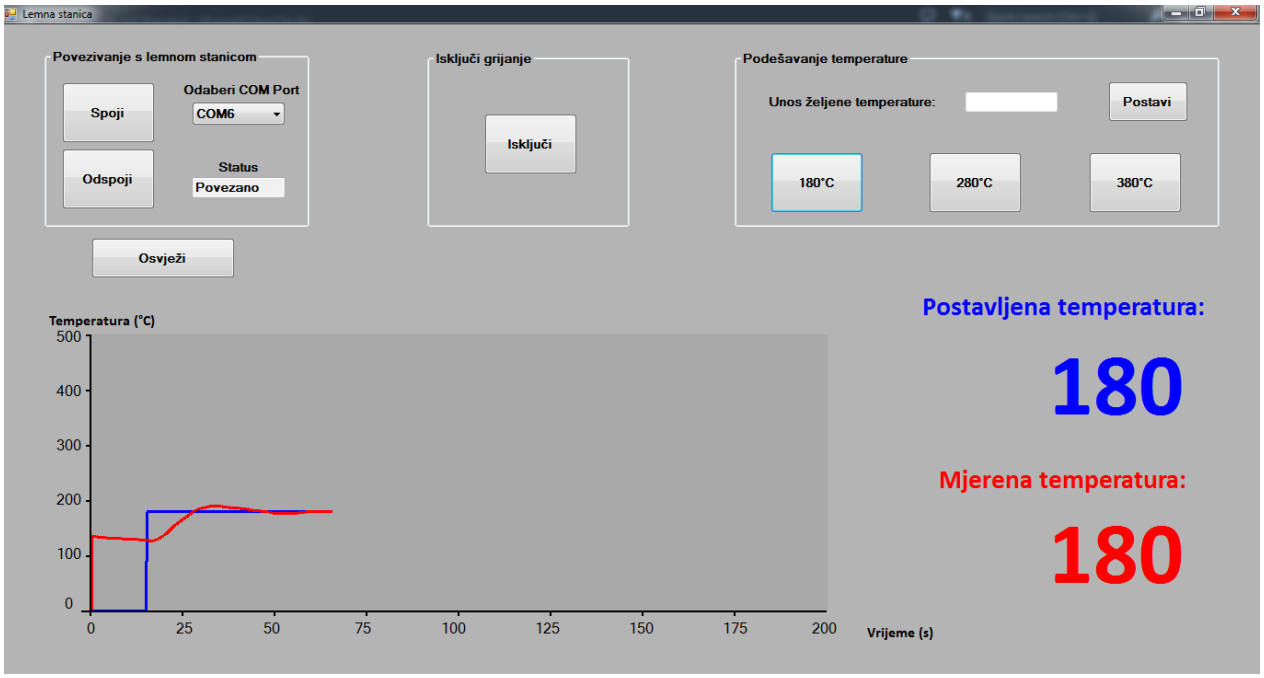
```
while (BTserial.available() > 0) {//sve dok je ulaz omogućen
    inChar = BTserial.read();//očitavanje s ulaza
   if (isDigit(inChar)) {//ako je dolazna vrijednost broj
     // pretvaranje dolazećeg bajta u char i spremanje u string
     inString += (char)inChar;
   }
   // ako dođe novi red spremaj string
     if (inChar == '\n')
     {
       SP_zadano=(inString.toInt());//pretvaranje stringa u int
       inString = "";//inicijalno postavljanje stringa
     }
}
```

Slanje podataka se vrši aktivacijom timera4 svakih 250 ms te se šalje sljedeći paket:

```
BTserial.print(";");BTserial.print(SP_zadano); BTserial.print(":"); BTserial.print(MV_mjereno);
BTserial.print(","); BTserial.print(".");
```

Kod slanja se šalju dva podatka SP_zadano i MV_mjereno što predstavlja zadanu i mjerenu vrijednost signala, te znakovi (;:,.). Točka-zarez označava početak stringa, dvotočka odvaja podatke, zarez i točka se koriste za kraj paketa.

Poslani podaci se pojavljuju u vizualizacijskom obliku u računalnoj aplikaciji prikazanoj na slici 3.19.



Sl. 3.19. *Računalna aplikacija*

Komunikacija s lemnom stanicom se ostvaruje odabiranjem odgovarajućeg komunikacijskog porta u ovome slučaju to je *„COM6“* za HC-05 bluetooth modul, u kodu to se postiže pomoću funkcije *pronadi_port()* koja izlistava popis korištenih portova na računalu.

Funkcija *pronadi_port()*:

```
private: void pronadi_port(void)
    {
        array<Object^>^ objectArray = SerialPort::GetPortNames();
        this->com_port->Items->AddRange(objectArray);
    }
```

Pritiskom na tipku „Spoji“ uspostavlja se komunikacija:

```
private: System::Void btn_Spoji_Click(System::Object^ sender, System::EventArgs^ e) {
            try{
                // ako komunikacijski port nije otvoren
                 if (!this->com_port_1->IsOpen){
                //odabrani port
                this->com_port_1->PortName = this->com_port->Text;
                this->com_port_1->BaudRate = 9600;//brzina prijenosa
                this->com_port_1->Open();//otvaranje porta
                }
             }
             catch (UnauthorizedAccessException^){
                this->txt_Status->Text = "Nije povezano";
            }
                if (this->com_port_1->IsOpen){//ako je port otvoren
                timer1->Enabled = true;//pokreni timer1
                this->txt_Status->Text = "Povezano";
                }
        }
```

Nakon uspostavljene komunikacije aktivira se timer1 svakih 250 ms kako bi se očitali i parsirali podaci:

```
private: System::Void timer1_Tick(System::Object^ sender, System::EventArgs^ e) {
            String^ delimStr = ";:,";
            array<Char>^ delimiter = delimStr->ToCharArray();
            array<String^>^ podatak;
```

```
if (this->com_port_1->IsOpen){//ako je port otvoren
        try{
            //očitanje i spremanje postojećih podataka u spremnik
            spremnik_podataka += this->com_port_1->ReadExisting();
            //obriši nepotpun paket ako se na nultom mjestu ne nalazi ";"
            while (spremnik_podataka[0] != ';')
            {
                spremnik_podataka = spremnik_podataka->Remove(0, 1);
                if (spremnik_podataka->Length < 1) break;
            }
            //sve dok spremnik sadrži zarez, parsiraj podatke
            while (this->spremnik_podataka->Contains(","))
            {
                podatak = spremnik_podataka->Split(delimiter);
                pod_1_SP = podatak[1];//parsiran podatak
                pod_2_MV = podatak[2];//parsiran podatak
                //traži index na kojem se nalazi zarez
                auto index = spremnik_podataka->IndexOf(",");
                //isprazni spremnik
                spremnik_podataka = spremnik_podataka->Remove(0, index + 1)
            }
        }
}
```

Ovi očitani i parsirani podaci se upotrebljavaju za prikaz u obliku dijagrama i numeričkih vrijednosti postavljene i mjerene temperature kao što je prikazano na slici 3.19.

Osim očitanja i vizualizacije računalna aplikacija pruža i mogućnost odabira željene temperature 180 °C, 280 °C, 380 °C pomoću tipkala, te unos pomoću tipkovnice. Pritiskom na određeni gumb postavlja se poruka i aktivira timer2 koji šalje tu poruku mikrokontroleru. U nastavku je dan primjer funkcije za postavljanje 380 °C, te funkcija za slanje podatka.

Pritiskom na gumb 380 °C aktivira se funkcija:

```
private: System::Void postavi_380C_Click(System::Object^ sender, System::EventArgs^ e) {
                message_arduino = "380";//string koji se šalje
                timer2->Enabled = true;//aktivacija timer-a 2
}
```

Kada je timer2 aktiviran, poziva se funkcija:

```
private: System::Void timer2_Tick(System::Object^ sender, System::EventArgs^ e) {
                this->com_port_1->WriteLine(message_arduino);//slanje poruke
                timer2->Enabled = false;//deaktivacija timer-a 2
}
```

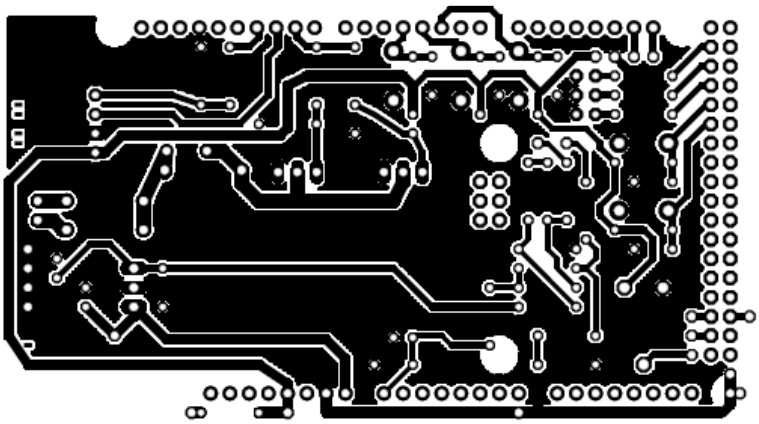
3.6. Izrada sklopa

Kako bi se svi elementi koji čine lemnu stanicu sklopili u jednu funkcionalnu cjelinu potrebno je izraditi tiskanu pločicu. Pločica je izrađena pomoću fotopostupka.

Materijal koji je korišten pri izradi je sljedeći:

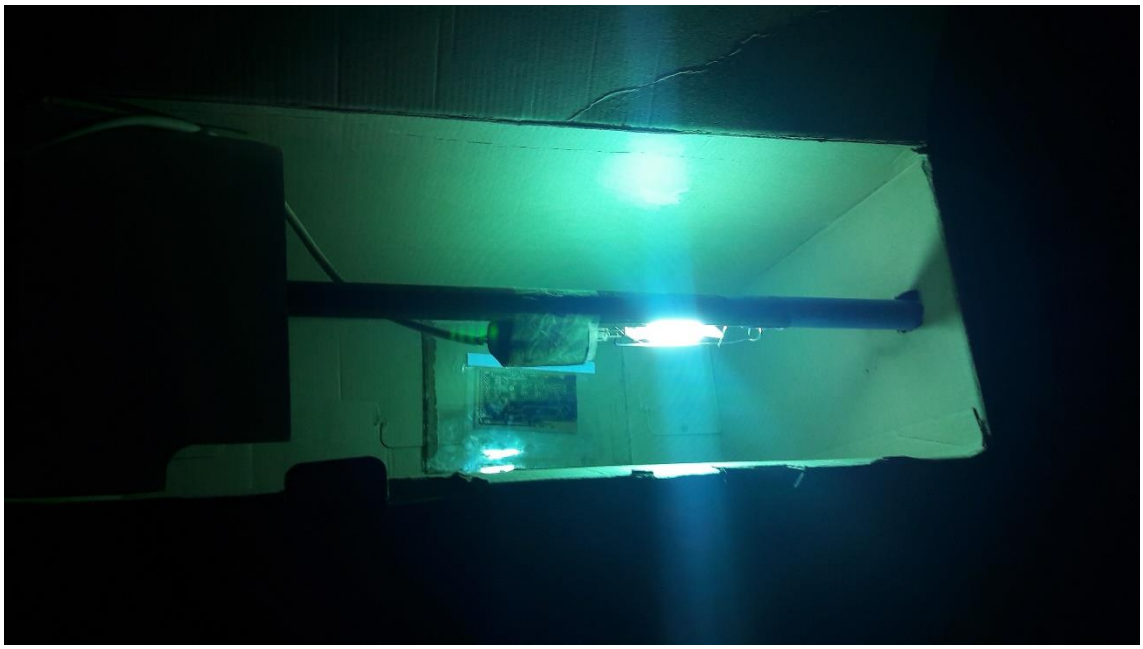
- pertinaks fotooslojena pločica
- prozirnica
- živina žarulja philips HPR 125
- natrijev hidroksid (NaOH)
- feriklorid (FeCl3)

Pločica se razvija u nekoliko koraka. Prvi korak je izrada sheme u programskom alatu „EAGLE“. Shema je dana u potpoglavlju 3.3. te na temelju te sheme izrađuje se predložak koji je prikazan na slici 3.20.



Sl. 3.20. *Predložak za izradu tiskane pločice*

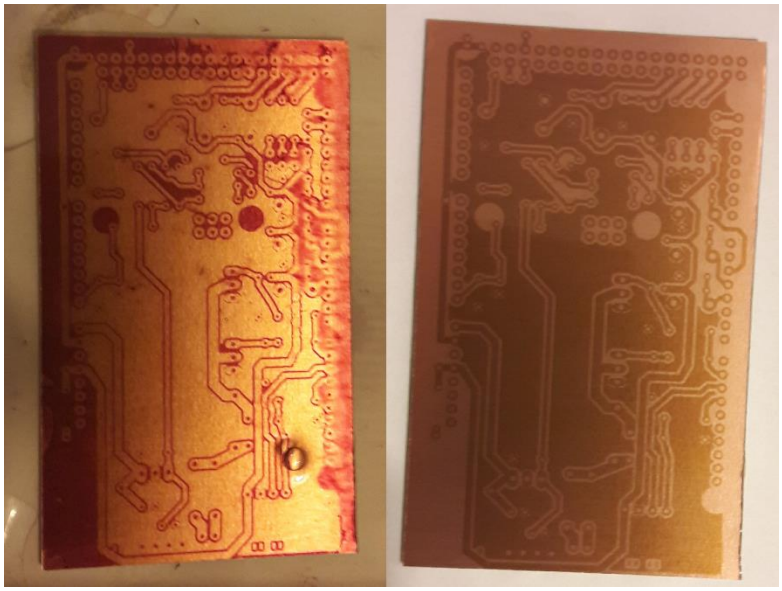
U drugom koraku predložak se printa pomoću laserskog pisaća na prozirnicu koja se stavlja preko fotooslojene pločice. Tako namješten predložak osvjetljava se pomoću živine žarulje philips HPR 125. Žarulja se postavlja na visinu 30 cm, te osvjetljivanje traje oko 3 min. Postupak je prikazan na slici 3.21.



Sl. 3.21. *Osvjetljivanje tiskane pločice*

Nakon osvjetljivanja potrebno je skinuti foto-lak, a to se postiže pomoću natrijevog hidroksida. Natrijev hidroksid se razrjeđuje s vodom te se u tako pripremljenu kupku uranja

pločica. Poželjno je da voda bude topla kako bi ubrzala postupak. Postupak je prikazan na slici 3.22.



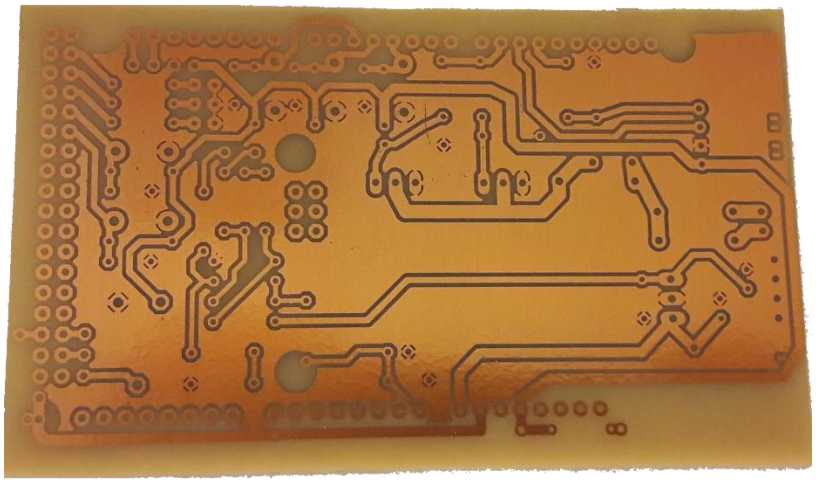
Sl. 3.22. *Lijevo se nalazi pločica za vrijeme kupke u otopini natrijevog hidroksida, desno je izgled pločice nakon gotovog postupka*

Kada je završen postupak skidanja foto-laka pločica se uranja u tekućinu feriklorida koja je prethodno zagrijana kako bi se ubrzao postupak jetkanja. Tijekom postupka je potrebno ljuljati posudicu s tekućinom, cijeli postupak traje do desetak minuta. Postupak je prikazan na slici 3.23.

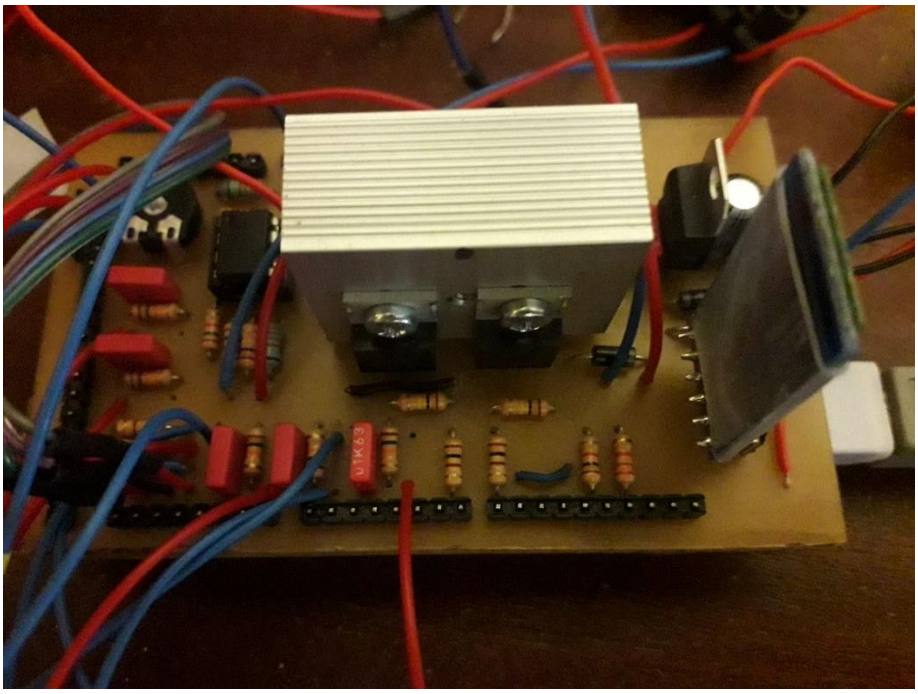
Sl. 3.23. *Jetkanje tiskane pločice*

Nakon jetkanja potrebno je dobro isprati pločicu pod mlazom vode. U završnoj fazi se buše rupe, spajaju komponente te se svi elementi vezani uz sklop stavljaju u izrađenu kutiju lemne stanice.

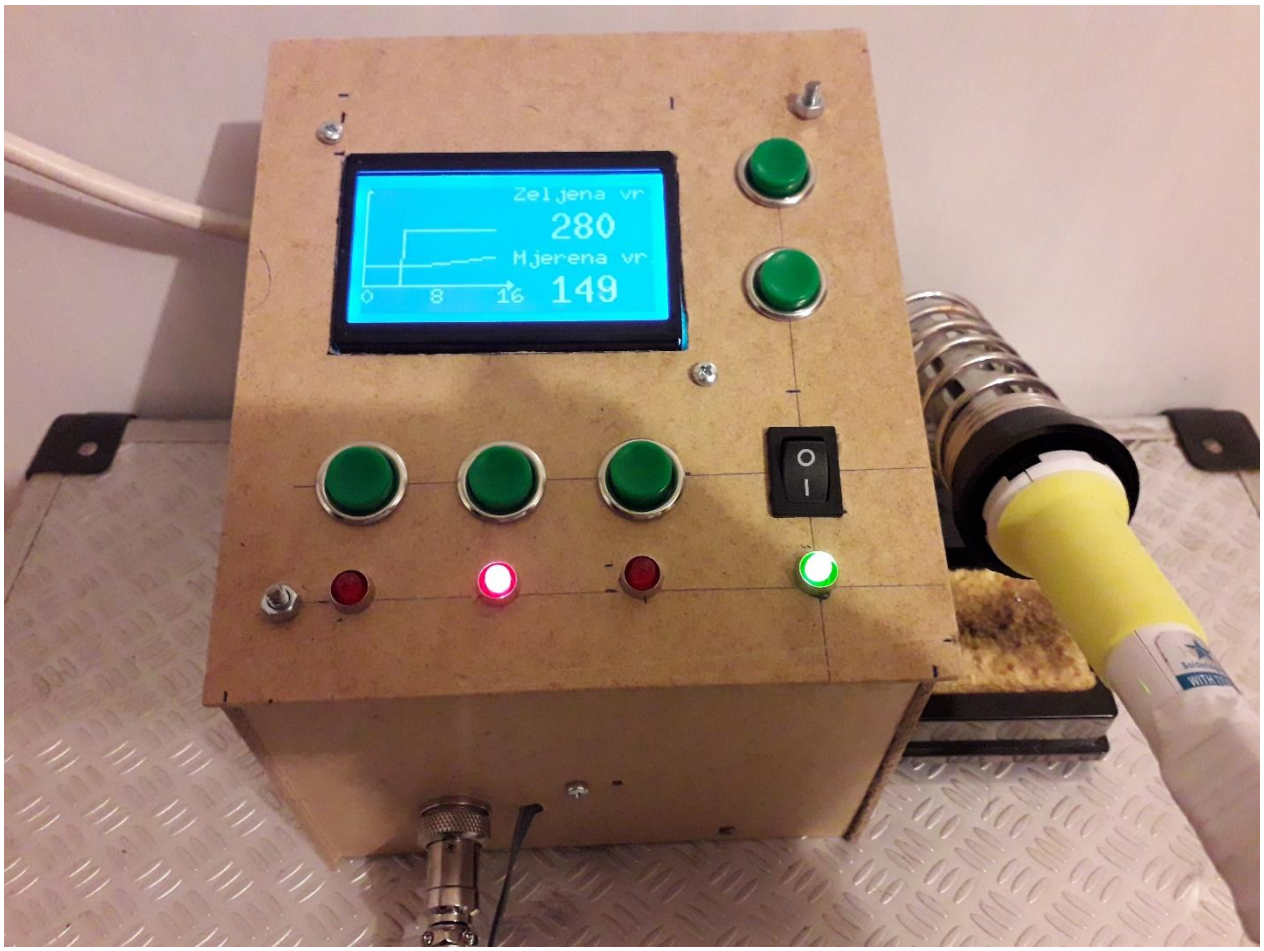
Slike 3.24., 3.25., 3.26. prikazuju izgled gotove tiskane pločice, gotovog sklopa i izgled lemne stanice.



Sl. 3.24. *Izgled tiskane pločice*



Sl. 3.25. *Izgled sklopa*



Sl. 3.26. *Izgled lemne stanice*

4. TESTIRANJE I REZULTATI

4.1. Metode testiranja

Testiranjem se određuju parametri koji služe za određivanje temperature iz očitane analogne vrijednosti. Parametri su: koeficijent pojačanja i tip osjetljivosti termopara koji se koriste u formulama (3 - 1) i (3 - 2). Koeficijent pojačanja se dobije odnosom izlaznog i ulaznog napona $U_{izlazno}/U_{ulazno}$, a tip osjetljivosti mjerenjem napona pri određenim temperaturama. Za oba parametra potrebno je provesti više mjerenja kako bi se dobila bolja aproksimacija temperature. Provedeno je i više mjerenja kako bi se utvrdila točnost očitane temperature sklopa i vanjskog termometra te testiranje PID regulatora.

4.2. Evaluacija rezultata

Određivanje kojeficijenta pojačanja i mjerenja prikazana su u tablici 4.1.

***Tab 4.1**. Tablica mjerenih uzorka pomoću kojih se određuje pojačanje*

$U_{izlazno}$ *(V)*	U_{ulazno} *(V)*	$pojacanje=U_{izlazno}/U_{ulazno}$
1.5	$10.5*10^{-3}$	142.85
2	$13.9*10^{-3}$	143.7
2.5	$17.4*10^{-3}$	142.27
3	$20.9*10^{-3}$	143.8
3.5	$24.6*10^{-3}$	143.5
$pojacanje$=143.2		

Aritmetička sredina pojačanja za pet mjerenja iznosi 143.2. Ova vrijednost se uvrštava u formulu (3-1).

Mjerenja za određivanje tipa osjetljivosti mjernog člana prikazana su u tablici 4.2.

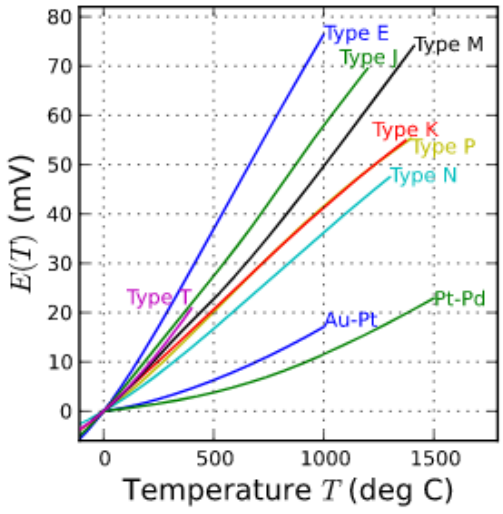
Pomoću izraza (4 - 1) dobiva se tip osjetljivosti, tj. ovisnost o kojoj se mijenja napon ovisno o porastu temperature.

$$tip\_osjetljivosti = \frac{U}{(mjerena\_temp - 25)} \qquad (4 - 1)$$

Tab 4.2. *Tablica mjerenih uzorka pomoću kojih se određuje tip osjetljivosti*

Mjerena temperatura: *mjerena_temp (°C)*	Napon: *U (mV)*	Tip osjetljivosti: *tip_osjetljivosti (µV/°C)*
180	10.6	68.38
197	11.8	68.6
219	13.3	68.5
tip_osjetljivosti=68.49 µV/°C		

Ovisnost o kojoj se mijenja napon ovisno o porastu temperature iznosi 68.49 µV/°C. Ova ovisnost daje uvid da se radi o E-tipu termopara što se može vidjeti iz slike 4.1. Ova vrijednost se također uvrštava u formulu (3-1).



Sl. 4.1. *Ovisnost napona o temperaturi termopara* [17]

Mjerenja kojima je određena točnost temperature provedena su pomoću vanjskog termometra, tj. multimetra „Fluke“ čija temperaturna sonda ima opseg od -20 do 200 °C, maksimalne rezolucije 0.1 °C i točnosti ±(1.0%+10). Ova mjerenja su provedena za 180 °C, te su analizirani podaci prema tablici 4.3. U tablici 4.3. su prikazane aritmetička sredina, apsolutna pogreška mjerenja, maksimalna apsolutna pogreška, relativna pogreška mjerenja, maksimalna relativna pogreška, standardna devijacija pojedinog mjerenja, standardnu devijaciju aritmetričke sredine i relativna nepouzdanost mjerenja.

***Tab 4.3**. Tablica analize mjerenih podataka temperature*

n	T_i	$\overline{T}$	$\overline{T}-T_i$	$\left\|\overline{T}-T_i\right\|$	$\Delta T, \Delta T_{max}$	r_x, r_{max}	$\left(\overline{T}-T_i\right)^2$	m_x, M_x
Mj .jed.	°C	°C	°C	°C	°C	%	°C	°C
1.	181.5		-0.8	0.8			0.64	
2.	181.3		-0.6	0.6			0.36	
3.	180.8		-0.1	0.1			0.01	
4.	179.7		1	1			1	
5.	179.8		0.9	0.9	ΔT=0.62° C	r_x=50%	0.81	$m_{10=}$ 0.73
6.	180.6	180,7	0.1	0.1			0.01	
7.	181.2		-0.5	0.5			0.25	$M_{10=}$
8.	181.8		-1.1	1.1	$\Delta T_{max=}$1.1° C	r_{max}=100%	1.21	0.23
9.	180.1		0.6	0.6			0.36	
10.	180.2		0.5	0.5			0.25	
$\sum_{i=1}^{n}$	10		0	6.2			4.9	

Aritmetička sredina:

$$\overline{T} = \frac{181.5 + 181.3 + 180.8 + 179.7 + 179.8 + 180.6 + 181.2 + 181.8 + 180.1 + 180.2}{10} = 180.7°C$$

Apsolutna pogreška mjerenja: $\Delta T = \frac{6.2}{10} = 0.62°C$

Maksimalna apsolutna pogreška: $\Delta T_{\max} = |180.7 - 181.8| = 1.1°C$

Relativna pogreška mjerenja: $r_x = \frac{0.62}{1} * 100\% = 62\%$

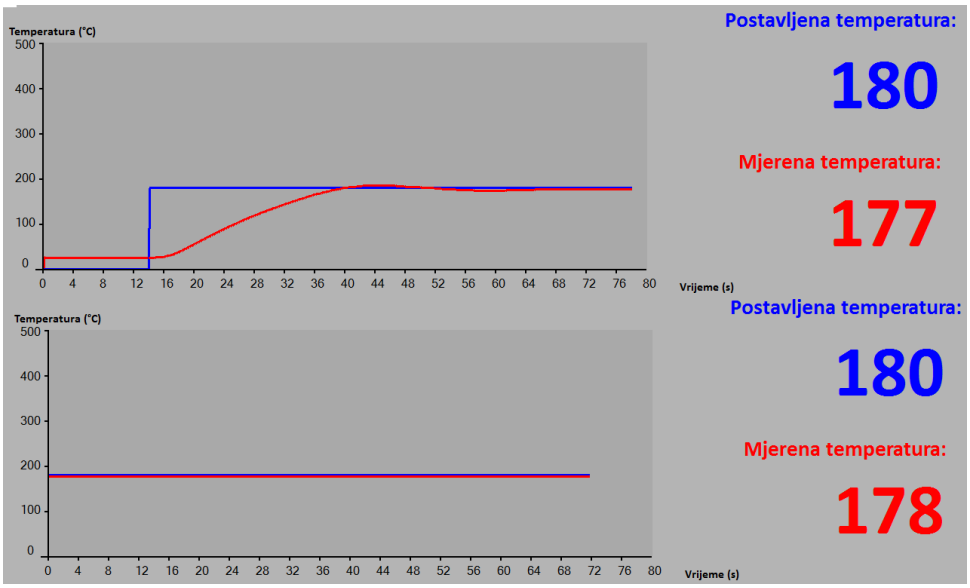
Maksimalna relativna pogreška mjerenja: $r_{\max} = \frac{1.1}{1} * 100\% = 110\%$

Standardna devijacija pojedinog mjerenja: $m_{10} = \sqrt{\frac{4.9}{10-1}} = 0.73°C$

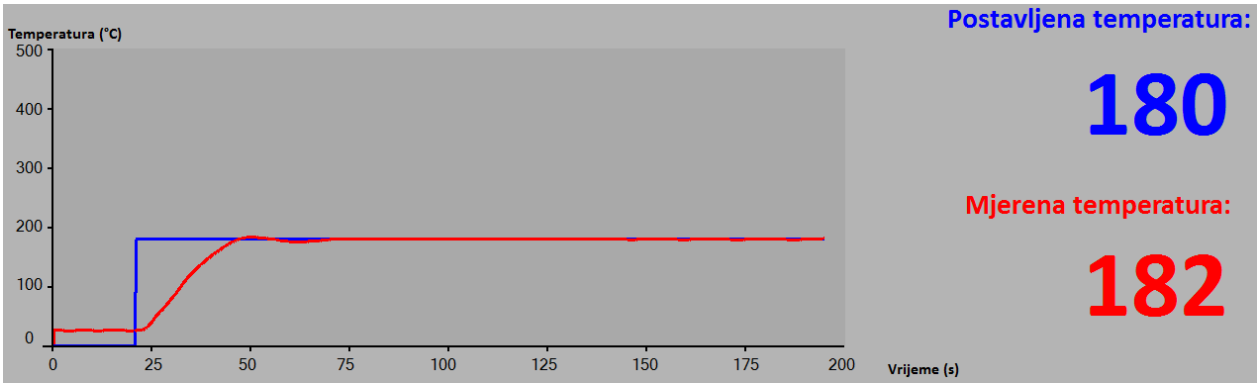
Standardna devijacija aritmetičke sredine: $M_{10} = \sqrt{\frac{4.9}{10*(10-1)}} = 0.23°C$

Relativna nepouzdanost mjerenja: $R_m = \frac{0.23}{1} * 100\% = 23\%$

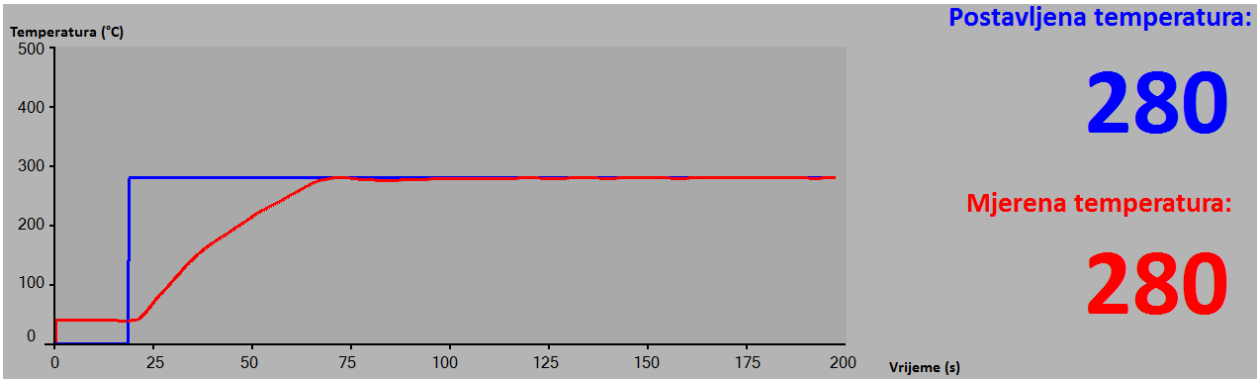
Testiranjem PID regulatora određuju se parametri i dobiva uvid o vladanju sustava. Parametri koji se koriste iznose za K_R=10, T_I=35 s i T_D=0.7 s. Na slici 4.2. prikazan je sami proporcionalni član za zadanu vrijednost 180 °C, međutim nakon dostizanja zadane vrijednosti i smirivanja oscilacije on ostaje u ustaljenom stanju ispod zadane vrijednosti. Dodavanjem integralnog člana otklanja se pogreška ustaljenog stanja. Kada se doda još i derivacijski član skraćuje se vrijeme za tri do četiri sekunde koje je potrebno da temperatura dosegne zadanu vrijednost. Vladanje PID regulatora je prikazano na slikama 4.3., 4.4. i 4.5. za zadanu vrijednost 180 °C, 280 °C i 380 °C. Ovo su ujedno i najbolji rezultati koji su dobiveni i pogreška im je unutar 1 do 2 °C nakon što se PID regulator stabilizira. Ponekad prijelazom iz zadane vrijednosti u drugu pojavljuju se male oscilacije i veće pogreške.



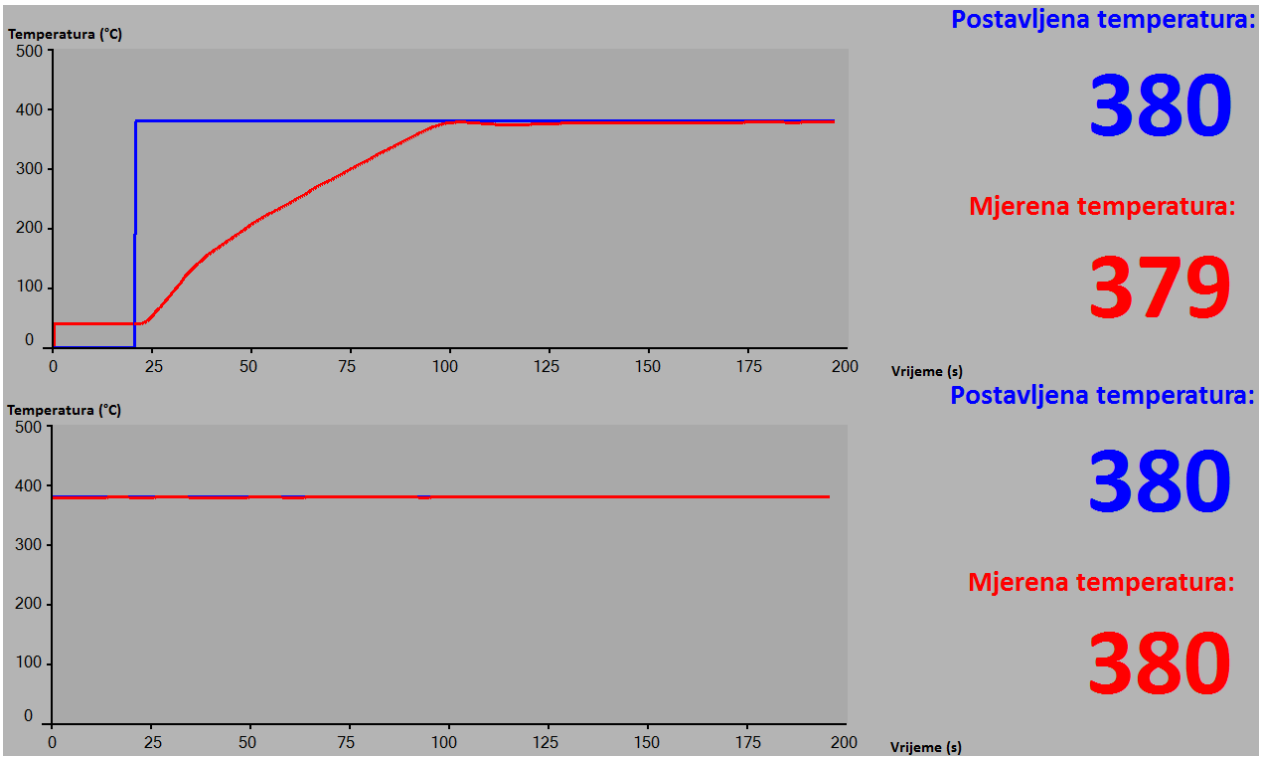
Sl. 4.2. *P regulator*



Sl. 4.3. *PID regulator za SP=180 °C*



Sl. 4.4. *PID regulator za SP=280 °C*



Sl. 4.5. *PID regulator za SP=380 °C*

5. ZAKLJUČAK

Zadatak ovog rada je izraditi lemnu stanicu koja će biti korisna i primjenjiva u praksi. U odnosu na standardne lemne stanice, pametna lemna stanica ima zadaću ukloniti nedostatke i poboljšati rad u vidu dobro kontrolirane temperature, prikaza temperature, odabira temperature, brzog zagrijavanja, "stand-by" režima rada pomoću žiroskopa te omogućiti komunikaciju i vizualizaciju podataka pomoću računala.

Pri izradi rada korištene su dvije vrste lemnih drški, a to su "HAKKO 907" i "HQ SOLDER/IR". Prednost "HAKKO 907" nad "HQ SOLDER/IR" je sposobnost bržeg zagrijavanja, dok joj je nedostatak brzo hlađenje. "HQ SOLDER/IR" bolje zadržava temperaturu, ali ima jako dugo vrijeme zagrijavanja. Prosječno vrijeme zagrijavanja od 25 °C do 380 °C za "HAKKO 907" iznosi oko 40 s, dok kod "HQ SOLDER/IR" ono iznosi oko 70 s. Napon kojim se napajaju ove drške iznosi do 24V.

Pokušano je smanjivanje vremena potrebnog zagrijavanju lemne drške "HAKKO 907" podizanjem napona većeg nego za koji je predviđen njezin rad, te joj je pri jednom takvom zagrijavanju pregorio grijač. Daljnji nastavak rada bazirao se na lemnoj dršci "HQ SOLDER/IR" te time nije ispunjen zadatak brzog zagrijavanja. Zadaci koji su ispunjeni su: "stand-by" režim pomoću žiroskopa ukoliko se detektira neaktivnost lemne drške, komunikacija s računalom putem HC-05 bluetooth modula, računalna aplikacija za vizualizaciju i upravljanje lemnom stanicom, te je implementiran PID regulator radi bolje kontrole temperature drške.

Rezultati koji su dobiveni testiranjem pokazuju da se zadana temperatura kreće s pogreškom unutar od 1 do 3 °C.

LITERATURA

[1] Solder, *https://en.wikipedia.org/wiki/Solder* , (Pristupljeno 19.06.2017)

[2] Quora, *https://www.quora.com/What-are-the-disadvantages-of-lead-free-solder-vs-lead-solder,* (Pristupljeno 21.06.2017)

[3] Židan A, Milobar B, *Spojevi s tranzistorima druga knjiga, IV. izdanje,* Tehnička knjiga Zagreb, Zagreb, 1991., str. od 170 do 172.

[4] PID controller, *https://en.wikipedia.org/wiki/PID_controller*, (Pristupljeno 25.06.2017)

[5] Arduino Mega, *https://www.arduino.cc/en/Main/arduinoBoardMega*, (Pristupljeno 23.06.2017)

[6] Ispis sadržaja ekrana u Arduino IDE-u

[7] HAKKO 907, *http://www.ebay.com/itm/New-Soldering-Station-Iron-Handle-for-HAKKO-907-ESD-907-936-937-928-926/121281474488?hash=item1c3cf06fb8:g:4jUAAOxyJX1TCWv0,* (Pristupljeno 19.06.2017)

[8] Dijelovi HAKKO 907, *http://www.sselec.com/data/Soldering%20Iron%20Accesories.htm*, (Pristupljeno 19.06.2017)

[9] LCD128x64, *http://overskill.alexshu.com/128x64-12864-lcd-w-arduino-serial/*, (Pristupljeno 19.06.2017)

[10] Izvor napajanja, *http://www.ebay.com/itm/100W-AC-DC-Converter-110V-220V-to-24V-DC6APowerSupplySwitchingTransformer/131643563137?hash=item1ea6915c81:g:qPwAAOSw~bFWNspS,* (Pristupljeno 20.06.2017)

[11] Akcelerometar MPU6500, *http://www.electrodragon.com/product/6dof-mpu-6500-3d-accelerogyro-breakout/,* (Pristupljeno 21.06.2017)

[12] HC-05 bluetooth modul, *https://forum.arduino.cc/index.php?topic=334629.0*, (Pristupljeno 21.06.2017)

[13] MOSFET IRFZ44N, *http://www.ebay.com/itm/10x-IRFZ44-N-Channel-Power-MOSFET-49A-55V-IRFZ44N-Transistor-Gate-FET-USA-/181528463011,* (Pristupljeno 22.06.2017)

[14] LM358N, *https://www.evakw.com/en/ic/311-lm358-lm358n-amplifier.html,* (Pristupljeno 22.06.2017)

[15] DC-DC LM2596 pretvarač, *https://www.ebay.com/itm/LM2596-Power-Supply-Output-1-23V-30V-DC-DC-Buck-Converter-Step-Down-Module-/141361451145?hash=item20e9cc9089%3Ag%3AVyUAAOSwVFlT12W7*, (Pristupljeno 30.11.2017)

[16] Perić N, Petrović I, *Automatizacija postrojenja i procesa, predavanja, III.DIO METODE I ALGORITMI UPRAVLJANJA PROCESIMA*, str. od 33 do 77.

[17] Termopar, *https://en.wikipedia.org/wiki/Thermocouple*, (Pristupljeno 4.12.2017)

SAŽETAK

U radu je dan teoretski uvid u lemne legure i kontaktno lemljenje. Na temelju zadataka koji se stavljaju pred pametnu lemnu stanicu izrađen je sustav za upravljanje lemnom drškom koji se zasniva na Arduino Mega 2560 platformi i programiran je u Arduino IDE sučelju. Ostvarena je komunikacija s računalom putem bluetootha, izrađena računalna aplikacija u Visual Studio-u, omogućen "stand-by" režim rada pomoću žiroskopa koji na temelju pokreta lemne drške u prostoru određuje način rada, te implementiran PID regulator kako bi se poboljšala kontrola temperature. Rezultati koji su dobiveni testiranjem zadovoljavajući su s pogreškom unutar 3 °C.

Ključne riječi: pametna lemna stanica, lemna drška, Arduino Mega 2560, Arduino IDE, bluetooth, Visual Studio, žiroskop, PID regulator.

ABSTRACT

The paper gives a theoretical insight into solder alloys and contact soldering. Based on the task assigned to the smart soldering station there was created system for control soldering iron based on the Arduino Mega 2560 platform and is programmed in Arduino IDE interface. There was realized computer communication with bluetooth module, made computing application in Visual Studio, enabled „stand-by“ operating mode with a gyroscope which determines the mode of operation and also implemented a PID controller to improve temperature control. The results obtained by testing are satisfactory with the error within 3 °C.

Keywords: smart soldering station, soldering iron, Arduino Mega 2560, Arduino IDE, bluetooth module, Visual Studio, gyroscope, PID controller

ŽIVOTOPIS

Ilija Majdenić rođen je 17.07.1991. u Osijeku. Živi u Podravskim Podgajcima gdje je polazio Osnovnu školu, nakon koje se upisuje Srednju školu Valpovo u Valpovu, smjer Elektrotehničar. Srednjoškolsko obrazovanje završava 2010. i iste godine upisuje Elektrotehnički fakultet u Osijeku, Stručni studij smjer automatika koji završava 2013. Trenutno je student 2. godine diplomskog studija procesnog računarstva na Fakultetu elektrotehnike računarstva i informacijskih tehnologija u Osijeku. Od stranih jezika koristi se engleskim te je član KUU „Napredak“ iz Podravskih Podgajaca te sudjeluje na brojnim manifestacijama.

PRILOZI

Programski kod korišten u ovom radu (Arduino)

```
#include<Wire.h>
#include "RunningAverage.h"
#include "U8glib.h"
#include <SoftwareSerial.h>
#define polje_1 20

SoftwareSerial BTserial(10, 11); // RX | TX
U8GLIB_ST7920_128X64_1X u8g(52, 51, 50);  // SPI Com: SCK = en = 18, MOSI = rw = 16, CS = di = 17

//varijable LCD
const int sirina=64;
const int visina=64;
const int length_1=sirina;
char val_1[10];
char val_2[10];
char nula[10];
char osam[10];
char sesnaest[10];
int x;
int y[length_1];
int y_1[length_1];
int nula_s=0;
int osam_s=8;
int sesnaest_s=16;
int cnt_tim4=0;

//moving average filter očitane mjerene vrijednosti
void put_temp(int item);
int moving_avg();
int red[polje_1];
int len;
int sum;

//varijable ziroskop
const int MPU_addr=0x68;  // I2C address of the MPU-6050
int16_t Tmp,GyX,GyY,GyZ;
RunningAverage z_x(5);
RunningAverage z_y(5);
RunningAverage z_z(5);
int a_x,a_y,a_z;
int last_a_x=0;
int last_a_y=0;
int last_a_z=0;
int cnt_vrijeme=0;
bool flg_1=false;

//flag timeri
bool update_tim1=0;//timer 1
bool update_tim3=0;//timer 3
bool update_tim4=0;//timer 4
//var za upravljanje
const int ledPin=13;
const int MV=A0;
const int pwm_output=9;
int output=0;
int output_1=0;
int SP_zadano=0;
int MV_mjereno=0;
int min_out=0;
int max_out=255;
float lastError=0;
float deltaError;
float error=0;
float P_output=0;
float I_output=0;
float D_output=0;
float last_I_output=0;
float last_D_output=0;

//PID parametri
float T_sample=0.1;
float Kr=10;
float Ti=35;
float Td=0.7;
int v=5;

//var za komunikaciju s računalnom aplikacijom
String inString = "";
int state;
int incomingByte = 0;
int inChar;
byte byteRead;

//var tipkala i led
const int btnPin[5]={2,3,4,28,30};//pinovi tipkala
const int led_pin[3]={22,24,26};//pinovi tipkala
int btnState[5] = {0,0,0,0,0};
int lastBtnState[5] = {0,0,0,0,0};
//bool tipkalo[5]={0,0,0,0,0}
bool tipkalo_1=false;
bool tipkalo_2=false;
bool tipkalo_3=false;
bool tipkalo_4=false;
bool tipkalo_5=false;
bool _flg_1=false;
bool _flg_2=false;
bool _flg_3=false;
bool _flg_4=false;
bool _flg_5=false;
```

```
//varijable za računanje temperature
float skalirajuci_faktor=102.1;
float ocitani_napon;

//inicijalizacija LCD 128x64
void u8g_prepare(void) {
  u8g.setFont(u8g_font_6x10);
  u8g.setFontRefHeightExtendedText();
  u8g.setDefaultForegroundColor();
  u8g.setFontPosTop();
}
//polje za iscrtvanje skale na LCD 128x64
const uint8_t skala_bitmap[] U8G_PROGMEM = {
0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
```

```
  0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x01, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x01, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x01, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x01, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x80,
  0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff,
0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0x03, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80,
0x01, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x80,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00
};

//inicijalizacija početnih postavki
void setup() {
 // put your setup code here, to run once:
 setPwmFrequency(9, 1024);//postavljanje frekvencije PWM-a
 x = 0;
 clearY();
 z_x.clear(); // explicitly start clean
 z_y.clear(); // explicitly start clean
 z_z.clear(); // explicitly start clean
 //kom sa HC-05
 BTserial.begin(9600);
 //komunikacija sa ziroskopom
 Wire.begin();
 Wire.beginTransmission(MPU_addr);
 Wire.write(0x6B);  // PWR_MGMT_1 register
 Wire.write(0);     // set to zero (wakes up the MPU-6050)
 Wire.endTransmission(true);
 Serial.begin(9600);
//postavljanje ulaza I izlaza
 pinMode(MV,INPUT);
// pinMode(btn_Pin,INPUT);
 pinMode(pwm_output, OUTPUT);
// pinMode(ledPin_1, OUTPUT);
 for (int i=0; i<5;i++)
  {
   pinMode(buttonPin[i], INPUT);
  }
 for (int i=0; i<3;i++)
  {
   pinMode( led_pin[i], OUTPUT);
  }
 cli();//stop interrupts

  //postavi timer1//100ms
  TCCR1A = 0;// set entire TCCR1A register to 0
```

```
 TCCR1B = 0;// same for TCCR1B
 TCNT1  = 0;//initialize counter value to 0
 // set compare match register for 1hz increments
 OCR1A = 1561;
 // turn on CTC mode
 TCCR1B |= (1 << WGM12);
 // Set CS10 and CS12 bits for 1024 prescaler
 TCCR1B |= (1 << CS12) | (1 << CS10);
 // enable timer compare interrupt
 TIMSK1 |= (1 << OCIE1A);

 //postavi timer4//250ms
 TCCR4A = 0;// set entire TCCR1A register to 0
 TCCR4B = 0;// same for TCCR1B
 TCNT4  = 0;//initialize counter value to 0
 // set compare match register for 1hz increments
 OCR4A = 3905;
 // turn on CTC mode
 TCCR4B |= (1 << WGM12);
 // Set CS10 and CS12 bits for 1024 prescaler
 TCCR4B |= (1 << CS42) | (1 << CS40);
 // enable timer compare interrupt
 TIMSK4 |= (1 << OCIE4A);

 //postavi timer3//4000ms
 TCCR3A = 0;// set entire TCCR1A register to 0
 TCCR3B = 0;// same for TCCR1B
 TCNT3  = 0;//initialize counter value to 0
 // set compare match register for 1hz increments
 OCR3A = 62499;
 // turn on CTC mode
 TCCR3B |= (1 << WGM12);
 // Set CS10 and CS12 bits for 1024 prescaler
 TCCR3B |= (1 << CS32) | (1 << CS30);
 // enable timer compare interrupt
 TIMSK3 |= (1 << OCIE3A);

 sei();//allow interrupts
}

ISR(TIMER1_COMPA_vect){//timer1
update_tim1=1;
volatile bool update_tim1=0;
}
ISR(TIMER3_COMPA_vect){//timer3
update_tim3=1;
volatile bool update_tim3=0;
}
ISR(TIMER4_COMPA_vect){//timer3
update_tim4=1;
volatile bool update_tim4=0;
}

void loop() {
 // put your main code here, to run repeatedly:
//očitavanje dolaznih podataka
while (BTserial.available() > 0) {
    inChar = BTserial.read();
  if (isDigit(inChar)) {
    // pretvaranje dolazećeg bajta u char i spremanje
u string
     inString += (char)inChar;
   }
  // ako dođe novi red spremaj string
    if (inChar == '\n')
    {
     SP_zadano=(inString.toInt());
     inString = "";
    }
}
//pozivanje funkcije tipkala_led
tipkala_led();
//pozivanje funkcije ziroskop aktivacijom timer-a3
if (update_tim3==1)
{
 ziroskop();
 update_tim3=0;
}
//očitavanje mjerene vrijednosti i PID računanje
aktivaijom timer-a1
if ( update_tim1==1){
val_mj=analogRead(MV);
put_temp(val_mj);
sum=moving_avg();
ocitani_napon=sum*4.88e-3;
MV_mjereno=25+skalirajuci_faktor*ocitani_napon;
if (SP_zadano==0)
{
 P_output=0;
 I_output=0;
 D_output=0;
 last_I_output=0;
 last_D_output=0;
 lastError=0;
 error=0;
 output=0;
 output_1=0;
}
//pozivanje funkcije PID_racunanje
PID_racunanje();
last_MV_mjereno=MV_mjereno;
update_tim1=0;
 }
//aktivacijom timer-a4 vrši se ispis na LCD te slanje
podataka računalu
 if ( update_tim4==1){
  lcd();
  cnt_tim4++;
  if(cnt_tim4==64)
  {
    nula_s+=16;
```

```
   osam_s+=16;
   sesnaest_s+=16;
   cnt_tim4=0;
  }
  BTserial.print(";");
  BTserial.print(SP_zadano);
  BTserial.print(":");
  BTserial.print(MV_mjereno);
  BTserial.print(",");
  BTserial.print(".");
  update_tim4=0;
 }
}
//PID algoritam
void PID_racunanje()
{
error=SP_zadano-MV_mjereno;
P_output=Kr*error;
if((output_1>max_out & error>0) ||
(output_1<max_out & error<0))
{
I_output=0;
}else{
I_output=last_I_output+(Kr*T_sample/Ti)*error;
last_I_output=I_output;
}
deltaError=error-lastError;
D_output=((Td/(Td+v*T_sample))*last_D_output)+(
(Kr*v*Td)/(Td+v*T_sample))*deltaError;
output = P_output+I_output+D_output;
output_1=output;
if (output>max_out)
{
output=max_out;
}
if (output<min_out)
{
output=min_out;
}
analogWrite(pwm_output,output);
last_D_output=D_output;
lastError=error;
}
//spremanje očitane temp. u polje
void put_temp (int element)
{
  if(len<polje_1){
    red[len++]=element;
  } else{
    for (int i=0; i<len-1;++i) red[i]=red[i+1];
    red[polje_1-1]=element;
  }
}
//računanje srednje vrijednosti
int moving_avg ()
{    int sum;
    for (int i=0;i<len;++i) sum+=red[i];
    sum=sum/len;
    return (sum);
}
//ziroskop
void ziroskop()
{
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);  // starting with register 0x3B
(ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr,14,true);  // request a
total of 14 registers
  GyX=Wire.read()<<8|Wire.read();  //  0x43
(GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
  GyY=Wire.read()<<8|Wire.read();  //  0x45
(GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
  GyZ=Wire.read()<<8|Wire.read();  //  0x47
(GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
  z_x.addValue(GyX);
  z_y.addValue(GyY);
  z_z.addValue(GyZ);
  a_x=z_x.getAverage();
  a_y=z_y.getAverage();
  a_z=z_z.getAverage();

  if((a_x>(last_a_x+500))||(a_x<(last_a_x-
500))||(a_y>(last_a_y+500))||(a_y<(last_a_y-
500))||(a_z>(last_a_z+500))||(a_z<(last_a_z-500)))
  {
    flg_1=true;
    cnt_vrijeme=0;
  }
  else
  {
    flg_1=false;
  }
  if(flg_1==false && SP_zadano>0)
  {
    cnt_vrijeme++;
  }
  if(cnt_vrijeme>30)
  {
    SP_zadano=0;
    cnt_vrijeme=0;
  }
  if(SP_zadano==0)
  {
    cnt_vrijeme=0;
  }
  last_a_x=a_x;
  last_a_y=a_y;
  last_a_z=a_z;
  update_tim3=0;
 }
```

```
//LCD
void lcd()
{
  itoa(MV_mjereno, val_1, 10);
  itoa(SP_zadano, val_2, 10);
  itoa(nula_s,nula,10);
  itoa(osam_s,osam,10);
  itoa(sesnaest_s,sesnaest,10);
  y[x] = map(SP_zadano,0,500,50,0);
  y_1[x]=map(MV_mjereno,0,500,50,0);

  u8g.firstPage();
  do {
    draw();
    drawY();
  } while( u8g.nextPage() );

  x++;
  if(x >= sirina){
    x = 0;
    clearY();
  }
  }
void clearY(){
  for(int i=0; i<length_1; i++){
    y[i] = -1;
  }
}
//funkcija za ispis
void draw(void) {
 u8g.drawXBMP(0, 0, 128, 64,rook_bitmap);
 u8g.setFont(u8g_font_6x10);
 u8g.drawStr(66,7,"Zeljena vr.");
 u8g.drawStr(66,40,"Mjerena vr.");
 u8g.setFont(u8g_font_10x20);
 u8g.drawStr(82,27,val_2);
 u8g.drawStr(82,59,val_1);
 u8g.setFont(u8g_font_6x10);
 u8g.drawStr(0,59,nula);
 u8g.drawStr(30,59,osam);
 u8g.drawStr(59,59,sesnaest);
  }
//crtanje dijagrama
void drawY(){
  u8g.drawPixel(0, y[0]);
  for(int i=1; i<length_1; i++){
    if(y[i]!=-1){
      u8g.drawLine(i-1, y[i-1], i, y[i]);
      u8g.drawLine(i-1, y_1[i-1], i, y_1[i]);
    }else{
      break;
    }
  }
}
//postavljanje PWM frekvencije
void setPwmFrequency(int pin, int divisor) {
 byte mode;
if(pin == 9 || pin == 10) {
  switch(divisor) {
    case 1: mode = 0x01; break;
    case 8: mode = 0x02; break;
    case 32: mode = 0x03; break;
    case 64: mode = 0x04; break;
    case 128: mode = 0x05; break;
    case 256: mode = 0x06; break;
    case 1024: mode = 0x07; break;
    default: return;
  }
  TCCR2B = TCCR2B & 0b11111000 | mode;
 }
}
//tipkala
void tipkala_led()
{
 for (int i=0;i<5;i++){

 btnState[i] = digitalRead(btnPin[i]);

 if (btnState[i] != lastBtnState[i]) {

  if (btnState[i] == LOW) {
   //counter++;
   if(i==0)
   {
    tipkalo_1=true;
    tipkalo_2=false;
    tipkalo_3=false;
    tipkalo_4=false;
    tipkalo_5=false;
    _flg_1=true;
    }
    else if(i==1)
    {
    tipkalo_1=false;
    tipkalo_2=true;
    tipkalo_3=false;
    tipkalo_4=false;
    tipkalo_5=false;
    _flg_2=true;
    }
    else if(i==2)
    {
    tipkalo_1=false;
    tipkalo_2=false;
    tipkalo_3=true;
    tipkalo_4=false;
    tipkalo_5=false;
    _flg_3=true;
    }
    else if(i==3)
    {
    tipkalo_1=false;
```

```
      tipkalo_2=false;
      tipkalo_3=false;
      tipkalo_4=true;
      tipkalo_5=false;
      _flg_4=true;
      }
      else if(i==4)
      {
      tipkalo_1=false;
      tipkalo_2=false;
      tipkalo_3=false;
      tipkalo_4=false;
      tipkalo_5=true;
      _flg_5=true;
      }
   }
  }
  lastBtnState[i] = btnState[i];
 }
if (_flg_1==true)
{
 SP_zadano=180;
 digitalWrite(led_pin[0],HIGH);
 digitalWrite(led_pin[1],LOW);
 digitalWrite(led_pin[2],LOW);
  _flg_1=false;
}
if (_flg_2==true)
{
 SP_zadano=280;
 digitalWrite(led_pin[0],LOW);
 digitalWrite(led_pin[1],HIGH);
 digitalWrite(led_pin[2],LOW);
  _flg_2=false;
}
if (_flg_3==true)
{
 SP_zadano=380;
 digitalWrite(led_pin[0],LOW);
 digitalWrite(led_pin[1],LOW);
 digitalWrite(led_pin[2],HIGH);
  _flg_3=false;
}
if (_flg_4==true)
{
 SP_zadano +=5;
  _flg_4=false;
}
if (_flg_5==true)
{
 SP_zadano -=5;
  _flg_5=false;
}
}
```

Programski kod korišten u ovom radu (Visual Studio C++)

```
#pragma once
#include <iostream>
#include <conio.h>
namespace APP_diplomski_rad {
	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;
	using namespace System::IO::Ports;
	public ref class MyForm : public
System::Windows::Forms::Form
	{
	public:
	//postavljanje varijabli
	System::String^ spremnik_podataka = "";
		System::String^ pod_1_SP;
		System::String^ pod_2_MV;
		String^ message_arduino;
		Graphics^ crtanje_osi;
		Graphics^ crtanje_dijagram;
		//var za crtanje SP
		int visina_SP_graf = 300;
		int xPos_SP_graf =0;//horizontalna
pozicija na grafu
		//varijable za crtanje kontinuirane
linije
		int zadnjixPos_SP = 0;
		int zadnjavisina_SP = 300;
private: System::Windows::Forms::Timer^ timer2;
private: System::Windows::Forms::Button^
btn_osvjezi;
private: System::Windows::Forms::Label^ label5;
private: System::Windows::Forms::Label^ label7;
private: System::Windows::Forms::Label^ label8;
private: System::Windows::Forms::Label^ label21;
private: System::Windows::Forms::Label^ label22;
private: System::Windows::Forms::Label^ label23;
private: System::Windows::Forms::Label^ label24;
private: System::Windows::Forms::Label^ label25;
private: System::Windows::Forms::Label^ label26;
private: System::Windows::Forms::Label^ label27;
private: System::Windows::Forms::Label^ label29;
private: System::Windows::Forms::Label^ label30;
private: System::Windows::Forms::Label^ label39;
private: System::Windows::Forms::Label^ label31;
private: System::Windows::Forms::Label^ label32;
private: System::Windows::Forms::Label^ label33;
private: System::Windows::Forms::Label^ label9;
public:
int zadnjavisina_MV = 300;
MyForm(void)
{
InitializeComponent();
pronadi_port();
//crtanje_osi_x_y();
crtanje_osi = pt_crtanje_osi->CreateGraphics();
crtanje_dijagram = pt_crtanje_dijagram-
>CreateGraphics();
}
protected:
/// Clean up any resources being used.
~MyForm()
{
	if (components)
	{
		delete components;
	}
}
private: System::Windows::Forms::GroupBox^
groupBox1;
private: System::Windows::Forms::Button^
btn_Odspoji;
private: System::Windows::Forms::Button^
btn_Spoji;
private: System::Windows::Forms::Label^ label2;
private: System::Windows::Forms::Label^ label1;
private: System::Windows::Forms::TextBox^
txt_Status;
private: System::Windows::Forms::GroupBox^
groupBox2;
private: System::Windows::Forms::Button^
btn_Iskljuci;
private: System::Windows::Forms::GroupBox^
groupBox3;
private: System::Windows::Forms::Button^
postavi_380C;
private: System::Windows::Forms::Button^
postavi_280C;
private: System::Windows::Forms::Button^
postavi_180C;
private: System::Windows::Forms::Button^
btn_Unos_rucno;
```

```
private: System::Windows::Forms::TextBox^
txt_unesi_temp;
private: System::Windows::Forms::Label^ label3;
private: System::Windows::Forms::Label^ label4;
private: System::Windows::Forms::Label^ lab_SP;
private: System::Windows::Forms::Label^ label6;
private: System::Windows::Forms::Label^ lab_MV;
private: System::Windows::Forms::PictureBox^
pt_crtanje_osi;
private: System::Windows::Forms::PictureBox^
pt_crtanje_dijagram;
private: System::Windows::Forms::ComboBox^
com_port;
private: System::IO::Ports::SerialPort^ com_port_1;
private: System::Windows::Forms::Timer^ timer1;
private: System::ComponentModel::IContainer^
components;

#pragma endregion
//pronađi odgovarajući port
private: void pronadi_port(void)
{
        array<Object^>^ objectArray =
SerialPort::GetPortNames();
        this->com_port->Items-
>AddRange(objectArray);
}
private:System::Voidbtn_Spoji_Click(System::Object
^ sender, System::EventArgs^ e)
{
//crtanje osi
Pen^ pen_osi = gcnew Pen(Color::Black, 2);
crtanje_osi->DrawLine(pen_osi, 50, 50, 50, 350);
 crtanje_osi->DrawLine(pen_osi, 50, 350, 850, 350);
 int startingPoint = 50;
 for (int i = 0; i < 8; i++){
crtanje_osi->DrawLine(pen_osi,startingPoint +
(100*i),350,startingPoint + (100*i),355);
 }
for (int i = 0; i < 6; i++) {
crtanje_osi->DrawLine(pen_osi, 40, startingPoint +
(60*i), 300, startingPoint + (60*i));
}
//spajanje
 try{
// ako port nije otvoren pokreni otvaranje
        if (!this->com_port_1->IsOpen){
        this->com_port_1->PortName = this-
        >com_port->Text;
        this->com_port_1->BaudRate = 9600;
        this->com_port_1->Open();
        }
}
catch (UnauthorizedAccessException^){
         this->txt_Status->Text = "Nije povezano";
        }
if (this->com_port_1->IsOpen){
        timer1->Enabled = true;
         this->txt_Status->Text = "Povezano";
        }
}
private: System::Void timer1_Tick(System::Object^
sender, System::EventArgs^ e) {
String^ delimStr = ";:,";
array<Char>^ delimiter = delimStr-
>ToCharArray();
 array<String^>^ podatak;

if (this->com_port_1->IsOpen){
         try{

        spremnik_podataka += this->com_port_1-
>ReadExisting();
//obriši nepotpun paket
while (spremnik_podataka[0] != ';'){
spremnik_podataka = spremnik_podataka-
>Remove(0, 1);
if (spremnik_podataka->Length < 1) break;
}
//očitaj dok ne dođe kraj stringa
while (this->spremnik_podataka->Contains(","))
{
podatak = spremnik_podataka->Split(delimiter);
pod_1_SP = podatak[1];
pod_2_MV = podatak[2];
//crtanje SP MV
int podatak_1 = Convert::ToInt32(pod_1_SP);
podatak_1 *= 0.6;
int podatak_2 = Convert::ToInt32(pod_2_MV);
podatak_2 *= 0.6;
Pen^ pen = gcnew Pen(Color::Blue, 3);
Pen^ pen_2 = gcnew Pen(Color::Red, 3);
crtanje_dijagram->DrawLine(pen, zadnjixPos_SP,
zadnjavisina_SP, xPos_SP_graf, visina_SP_graf -
podatak_1);
crtanje_dijagram->DrawLine(pen_2, zadnjixPos_SP,
zadnjavisina_MV, xPos_SP_graf, visina_SP_graf -
podatak_2);
```

```
zadnjixPos_SP = xPos_SP_graf;
zadnjavisina_SP = visina_SP_graf - podatak_1;
zadnjavisina_MV = visina_SP_graf - podatak_2;// at
the edge of the window, go back to the beginning:
if (xPos_SP_graf >= 800) {
            xPos_SP_graf = 0;
           zadnjixPos_SP = 0;
            crtanje_dijagram-
>Clear(Color::DarkGray);  //očisti ekran
}else {
// povećaj horizontalnu os:
            xPos_SP_graf++;
}
lab_SP->Text = pod_1_SP;
lab_MV->Text = pod_2_MV;
auto index = spremnik_podataka->IndexOf(",");
spremnik_podataka = spremnik_podataka-
>Remove(0, index + 1);
}
}
 catch (TimeoutException^){
}
}
private: System::Void
btn_Odspoji_Click(System::Object^ sender,
System::EventArgs^ e) {
            this->com_port_1->Close();
            timer1->Enabled = false;
            this->txt_Status->Text = "Nije povezano";}
//isključi grijanje
private: System::Void
btn_Iskljuci_Click(System::Object^ sender,
System::EventArgs^ e) {
            message_arduino = "0";
             timer2->Enabled = true;
}
//pošalji ručno postavljenu temp.
private: System::Void
btn_Unos_rucno_Click(System::Object^ sender,
System::EventArgs^ e) {
            String^ message = this->txt_unesi_temp-
>Text;
// pošalji poruku
if (this->com_port_1->IsOpen)
 this->com_port_1->WriteLine(message);
}
//funkcije za odabir temp.
private: System::Void
postavi_180C_Click(System::Object^ sender,
System::EventArgs^ e) {
            message_arduino = "180";
            timer2->Enabled = true;
}
private: System::Void
postavi_280C_Click(System::Object^ sender,
System::EventArgs^ e) {
            message_arduino = "280";
             timer2->Enabled = true;
}
private: System::Void
postavi_380C_Click(System::Object^ sender,
System::EventArgs^ e) {
             message_arduino = "380";
            timer2->Enabled = true;
}
private: System::Void timer2_Tick(System::Object^
sender, System::EventArgs^ e) {
this->com_port_1->WriteLine(message_arduino);
timer2->Enabled = false;
}
private: System::Void
btn_osvjezi_Click(System::Object^ sender,
System::EventArgs^ e) {
crtanje_dijagram->Clear(Color::DarkGray);  //Clear
the screen.
lab_SP->Text = "0";
 lab_MV->Text = "0";
}
};
}
```