SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Sveučilišni preddiplomski studij elektrotehnike

Ultrazvučno mjerenje i regulacija razine sa ATMEL ATMega8535

Završni rad

Goran Horvat

Osijek, 2008

Sadržaj:

1. UVOD

U današnjem tehnološki razvijenom svijetu, pojmove mjerenja i regulacije susrećemo na svakom koraku, od jednostavnog mjerenja fizičkih dimenzija tijela, pa sve do regulacije najkompliciranijih sustava svemirskih letjelica. Iz te gomile pojmova sustava i uređaja, u ovom radu biti će izdvojen jedan način mjerenja i regulacije razine, koristeći ultrazvučno mjerenje razine, koje se koristi kada su konvencionalna mjerenja nemoguća.

U ovom radu detaljno će biti prikazan princip ultrazvučnog mjerenja razine određene tvari, također će biti objašnjen princip regulacije razine, način prikaza i određivanja parametara regulatora pomoću osobnog računala, gdje je srce cijelog sustava mikro upravljač Atmel AtMega8535. Koristeći programske alate za programiranje mikro upravljača, i projektiranje programa te dijagrama toka, biti će razrađena električna shema uređaja, i objašnjen program upisan u mikro upravljač. Nakon projektiranja biti će spomenuta izrada i testiranje sustava, ali sustav nije potpuno dovršen niti ispitan, zbog problema sa nabavkom ultrazvučne sonde.

2. PRINCIP ULTRAZVUČNOG MJERENJA

Da bi mogli objasniti princip rada ultrazvučnog mjerenja razine neke tvari, prvo moramo objasniti teorijske osnove zvuka, tj. ultrazvuka, pomoću kojega vršimo spomenuta mjerenja.

2.1 Zvuk i ultrazvuk

Zvukom se u užem smislu značenja te riječi naziva sve ono što čujemo i što zamjećujemo sluhom. Prema fizikalnoj definiciji, zvuk je titranje u plinovitim, tekućim i krutim elastičnim tvarima. Možemo također reći da se zvuk sastoji od ritmičkog njihanja molekula, koje u njihov ravnotežni položaj vraćaju među molekularne elastične sile, prema[1].

Zvučni valovi obuhvaćaju frekvencije unutar područja čujnosti od 16Hz do 20kHz. Područje frekvencija ispod frekvencije od 16Hz naziva se infrazvukom, dok se područje frekvencija iznad frekvencije od 20kHz naziva ultrazvukom.

Za sve vrste valova vrijedi sljedeća relacija koja povezuje valnu duljinu i frekvenciju:

$$v = f \cdot \lambda \tag{2-1}$$

gdje je v- brzina širenja vala, f – frekvencija vala, λ – valna duljina

U sustavima mjerenja udaljenosti pomoću zvuka, najčešće se koristi ultrazvučno područje frekvencija, zbog nekoliko razloga.

Područje infrazvuka je područje frekvencija ispod 16Hz, koje ima svojstvo da prelazi velike udaljenosti i prelazi preko čvrstih prepreka sa vrlo malim gubitcima. Područje infrazvuka se koristi u seizmografiji u svrhu otkrivanja potresa i podrhtavanja, ali je beskorisno u području mjerenja udaljenosti.

Područje čujnosti, od 16Hz do 20kHz pokriva veliki raspon frekvencija, ali problem nastaje u samom pojmu „područje čujnosti“. Svi električni uređaji koji bi mjerili udaljenost koristeći frekvencije od 16 – 20000Hz bi bili neugodno čujni, i ne toliko precizni kao ultrazvučni uređaji.

Ultrazvučno područje, u drugu ruku, ima frekvencije veće od 20kHz, te je teže proizvesti ultrazvučni val nego normalni zvučni val, ali zbog svoje visoke frekvencije ultrazvučni val ima malu valnu duljinu, što je najveća prednost ultrazvučnog područja, jer mala valna duljina znači veću vjerojatnost da će se val odbiti od neko sredstvo, što je osnova u određivanju udaljenosti pomoću zvuka.

2.1.1 Brzina širenja zvuka

Poput svih valova, brzina zvuka ovisi o sredstvu u kojem se širi. Točna brzina širenja zvuka u nekom sredstvu ovisi o njegovoj elastičnosti i gustoći. Što je veća elastičnost medija, a manja njegova gustoća, to će se zvuk brže širiti kroz medij. Na brzinu širenja će utjecati i promjena temperature, i to iz razloga što se povećanjem temperature smanjuje gustoća medija (npr. zraka), bez utjecaja na elastičnost. Dok temperatura ima utjecaja na brzinu širenja zvuka, tlak neznatno utječe. To je zbog toga što promjena tlaka uzrokuje jednake promjene u elastičnosti i gustoći, pa brzina ostaje neznatno promijenjena.

Brzina zvuka kojom se širi zvuk u zraku ili plinovima općenito može se izračunati prema formuli:

$$c = \sqrt{\frac{p_0 \cdot \gamma}{\rho_0}} \tag{2-2}$$

Za zrak je γ=1,4. Sa porastom nadmorske visine sve je manji atmosferski tlak $p_{0,}$ no isto tako i gustoća zraka, odnosno njegova specifična težina ρ_0. Zato promjena atmosferskog tlaka neznatno utječe na brzinu zvuka. Naprotiv, o temperaturi, brzina zvuka ovisi mnogo više. Formula koja približno opisuje brzinu zvuka u zraku, pri vlažnosti od 0% je:

$$c = 331{,}3 + 0{,}606 \cdot \vartheta (m/s) \tag{2-3}$$

gdje je ϑ temperatura zraka.

Brzina zvuka u zraku ovisna je o postotku vlage, ali promjena je između 0,5‰ i 3,3‰ što je zanemarivo u proračunima udaljenosti, prema [1].

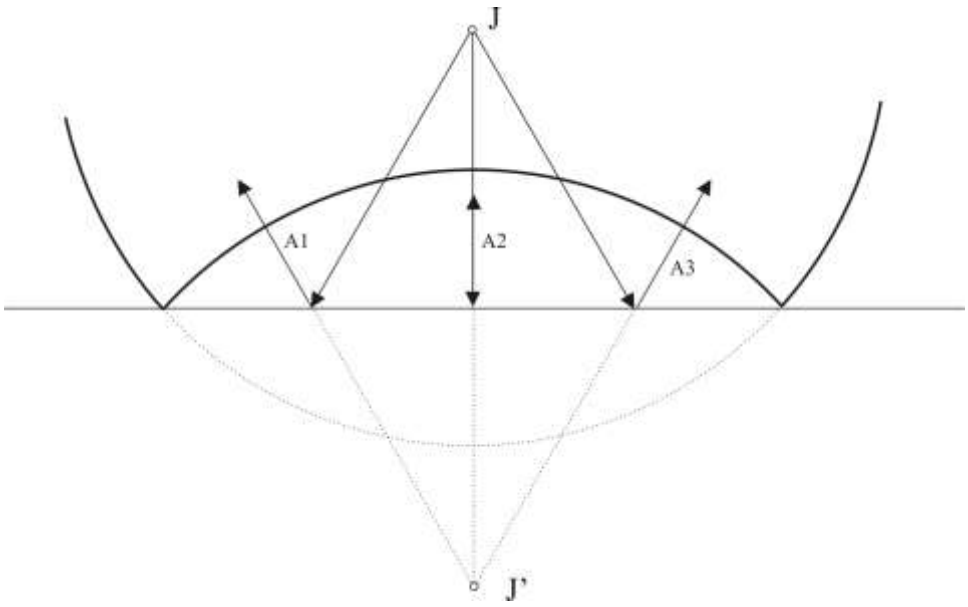
2.1.2 Usmjerenost zvučnog izvora

Kada pričamo o usmjerenosti izvora zvuka, općenito se može reći da je intenzitet valova kojima je valna duljina mnogo veća od dimenzija izvora (od promjera zvučnika), približno jednak u svim smjerovima. Na višim frekvencijama valna duljina je manja od izvora i u tom frekvencijskom području dolazi do usmjeravanja zvučnih valova. To je velika prednost ultrazvučnog područja, jer je moguće lakše usmjeriti zvučne valove u točno određeno područje koje mjerimo.

2.1.3 Refleksija zvuka

Put širenja zvuka može se prikazati zvučnim zrakama. To su zamišljene zrake okomite na čelo vala, koje omogućuju da se optički zakoni vezani za refleksiju svjetlosti primjene na zvuk. Glavni uvjet je da je duljina zvučnog vala, puno manja od dimenzije plohe od koje se val reflektira[1].

Kod ultrazvučnog mjerenja udaljenosti refleksija igra veliku ulogu, jer je potrebno znati na koji način se zvučni val odbija od površine i koliki put prelaze zvučne zrake. Pošto znamo da se kod refleksije zvuka, primjenjuje zakon refleksije svjetlosti, možemo napraviti teoretski model određivanja udaljenosti ravne površine.



Slika 2.1 Princip refleksije zvuka od ravnu podlogu

Uzmimo u obzir da je J izvor ultrazvučnog vala, a podloga da je neka određena razina tvari koju mjerimo. Prema slici 2.1 vidimo da ako iz usmjerenog izvora odašiljemo zvučni val, te ako je taj isti izvor ujedno i prijemnik, samo zvučna zraka koja dolazi pod kutom od 90 stupnjeva na sredstvo se vraća u isti odašiljač, prelazeći time dvostruku udaljenost od sredstva do izvora. S obzirom na ovaj dokaz, vidimo da će ultrazvučni mjerač mjeriti samo najbližu tj. realnu udaljenost do sredstva, dok će sve ostale zrake izbjeći prijemnik, ili u najgorem slučaju ako se reflektiraju od drugih prepreka, stići u prijemnik puno kasnije, te će biti zanemarene. Zahvaljujući refleksiji, možemo mjeriti razinu u jako uskim spremnicima, bez straha od detektiranja sporednih zraka.

2.1.4 Izvori i prijemnici ultrazvučnih valova

Izvori ultrazvučnih valova su uređaji koji pretvaraju energiju u ultrazvuk, dok su prijemnici uređaji koji imaju obrnuti princip od izvora. Izvori su u principu zvučnici koji proizvode zvučne valove frekvencija iznad 20kHz, a prijemnici ultrazvučnih valova su mikrofoni koji primaju frekvencije iznad 20kHz.

Poznato nam je pravilo da zvučnik proizvodi zvuk na način da promjena struje u zavojnici stvara promjenjivi magnetski tok na permanentni magnet, te se stvara sila koja pomiče membranu i stvara titranje membrane koja tada proizvodi zvuk. Ako ovu situaciju gledamo iz druge perspektive, zvučni val koji dolazi iz izvora prema zvučniku rezultira titranje membrane koja pomiče zavojnicu unutar magnetskog polja, te se inducira napon. Iz priloženog se vidi da su u principu zvučnik i mikrofon jedan te isti uređaj, samo ovisi dali je taj uređaj postavljen kao potrošač, ili kao izvor električne energije. Zato ultrazvučne prijemnike i predajnike zovemo zajedničkim nazivom primopredajnici ili Trancducers (eng.). U pravilu postoje dvije vrste ultrazvučnih primopredajnika, a to su: piezoelektrični i eletromagnetski primopredajnici.

Elektromagnetski primopredajnici su zvučnici koji se sastoje od permanentnog magneta i zavojnice te membrane koja proizvodi (ili prima) zvuk. Ovaj tip ultrazvučnih primopredajnika je pogodan za frekvencije od 20kHz do 40kHz, i mogu stvoriti zvučne valove velike snage.

Drugi tip ultrazvučnih primopredajnika su piezoelektrični primopredajnici koji rade na načelu Piezoelektričnog efekta. Piezoelektrični efekt je svojstvo nekih materijala, najčešće kristala, da induciraju napon kada su izloženi mehaničkom naprezanju. Također efekt je prisutan i u obrnutom procesu, kada se piezoelektričnom materijalu narine napon, on se deformira. Ovo je osnova rada piezoelektričnih ultrazvučnih primopredajnika. Kada se određenom piezoelektričnom materijalu (najčešće PbZrTi) narine sinusni napon frekvencije iznad 20kHz , on počinje oscilirati narinutom frekvencijom i počinje proizvoditi ultrazvučne valove, i obrnuto.

Svi tipovi ultrazvučnih primopredajnika su konstruirani za točno određenu frekvenciju (najčešće 40kHz), sa točno danim dijagramom zračenja i točno izraženim podatcima jačine zvučnih valova, te se prema danim podatcima odabiru ultrazvučni primopredajnici, ovisno o primjeni.

2.2 Teorija ultrazvučnog mjerenja

Nakon uvoda u teoriju ultrazvuka dolazi se do dijela teoretske osnove ultrazvučnog mjerenja razine, tj. principa rada ultrazvučne sonde koja je zaslužena za mjerenje razine određene tvari. Kao što je pokazano u poglavlju 2.1, zvuk se kroz zrak širi brzinom koja ovisi o temperaturi, te na sobnoj temperaturi (22ºC) iznosi 344,6 m/s. Podatak o brzini zvuka nam mora stalno biti dostupan, te treba biti kompenziran temperaturom prostorije da bi mogli dobiti točno očitanje razine koju mjerimo. No zašto nam je potrebna brzina zvuka te na kojem principu radi ultrazvučno mjerenje razine? Proučimo način ultrazvučnog mjerenja udaljenosti.

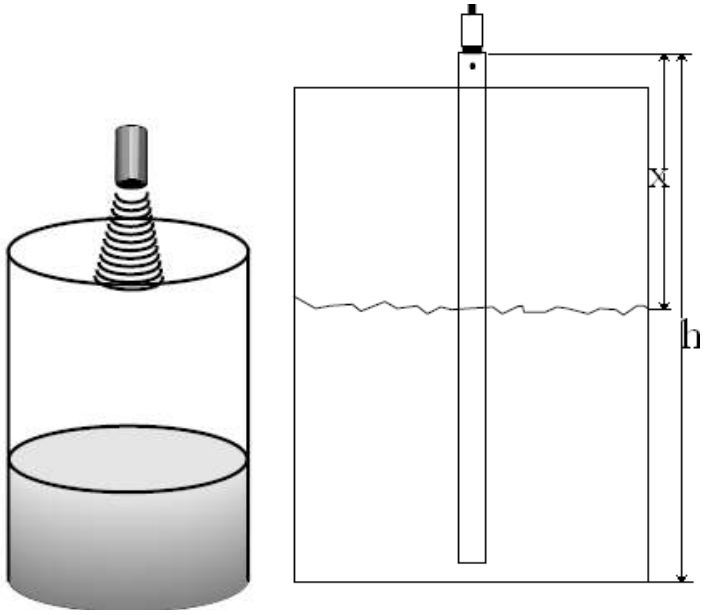
2.2.1 Ultrazvučno mjerenje udaljenosti

Kako bi odredili udaljenost do nekog predmeta, najjednostavnija metoda je pomoću ultrazvuka. Princip se bazira na slanju zvučnog vala kratkog trajanja prema određenom objektu, te mjerenja vremena potrebnog da se val odbije i vrati na mjesto slanja. Uz poznavanje brzine zvuka, dobivamo put koje zvučni val prijeđe od odašiljanja do primanja reflektiranog vala i taj je put dvostruko veći od udaljenosti ultrazvučnog primopredajnika do prepreke od koje se zvučni val odbio. Na ovaj način možemo mjeriti udaljenost od čvrstih objekata, tekućina, prepreka itd. No kao i kod ostalih metoda mjerenja udaljenosti, kod ove metode postoje četiri glavna problem koji mogu utjecati na točnost rezultata:

1) Površina- idealna površina je glatka i tvrda, okrenuta okomito na lice senzora. Ovakva će površina odbijati puno više signala nego meka površina koja upija zvučne valove. Površina sa slabim karakteristikama refleksije zvučnih valova smanjuje domet, i smanjuje točnost mjerenja.
2) Udaljenost – što je udaljenost manja to će odbijeni zvučni val biti jači. Stoga, kako se udaljenost povećava predmet zahtjeva bolja svojstva refleksije zvučnih valova da bi povratni zvučni val bio dovoljne jačine.
3) Veličina – veći predmet ima veću površinu za refleksiju signala nago manji stoga će se veći predmet moći mjeriti sa veće udaljenosti nego manji. Površina koja se prepoznaje kao cilj je najčešće površina najbliža primopredajniku.
4) Kut – nagib površine predmeta koji gleda prema ultrazvučnom senzoru utječe na refleksiju zvuka od objekta (opisano u 2.1). Samo dio površine koji je okomit na zvučnu zraku vraća zvučni val koji se detektira. Ako je kut pre velik zvučni val se neće vratiti natrag u senzor i tako se neće moći izmjeriti udaljenost. U pravilu predmet sa kutom od 5º od okomice neće biti detektiran.

2.2.2 Ultrazvučna sonda – mjerenje razine

Na principu ultrazvučnog mjerenja udaljenosti radi i ultrazvučna sonda, koja je ključni dio mjerenja razine. Ultrazvučna sonda je programabilni uređaj koji se koristi u industriji za mjerenje udaljenosti, razine tekućina, mjerenje visine objekata na trakama itd., a u našem slučaju se koristi kao senzor za mjerenje razine tekućina, raznih smjesa, emulzija idr.



a) b)

Slika 2.2 a) Ilustrirani prikaz sonde i b) tehnički prikaz dimenzija i razine

Sonda radi na principu ultrazvučnog mjerenja udaljenosti, tako da šalje niz ultrazvučnih pulsova od nje do razine tekućine, te oduzimajući tu udaljenost od udaljenosti sonde do dna spremnika daje razinu u spremniku, slika 2.2 b).

Sonda ima analogni izlaz u obliku strujnog izvora karakteristika strujne petlje, koji daje od 4mA do 20 mA ovisno o razini spremnika, te se programira pomoću računalnog sučelja. Sonda također ima kompenzaciju utjecaja temperature, kojim se postiže otpornost mjernog rezultata na promjene vanjske temperature.

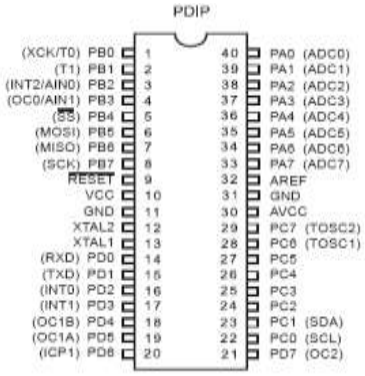
3. PREGLED REGULACIJSKOG SUSTAVA SA ATMEL ATmega8535

Kao što je pokazano u prethodnom poglavlju, mjerenje razine vrši se pomoću programirljive ultrazvučne sonde, koja nam na temelju visine razine tvari koju mjerimo, vraća analogni strujni signal u području od 4mA do 20mA. Raspon od 4mA do 20mA je u linearnom odnosu na razinu spremnika, tako je 4mA istovjetno sa 0% popunjenosti spremnika, a 20mA istovjetno sa 100% popunjenosti spremnika, tj. razine spremnika. Da bi analogni strujni signal mogli koristiti za regulaciju razine, moramo ga pretvoriti u digitalni signal. Za pretvorbu se koristi mikro upravljač Atmel AtMega8535, koji sa svojim analogno-digitalnim pretvornikom pretvara strujni signal u digitalni signal. No analogno-digitalna pretvorba nije jedini zadatak Atmel-a AtMega8535. Naprotiv, Atmel Atmega8535 je srce ovog sustava regulacije i mjerenja, jer on obrađuje analogne podatke, vrši regulaciju i vrši komunikaciju sa osobnim računalom. Stoga ćemo pogledati princip rada Atmela AtMega8535

3.1 Atmel ATMega8535

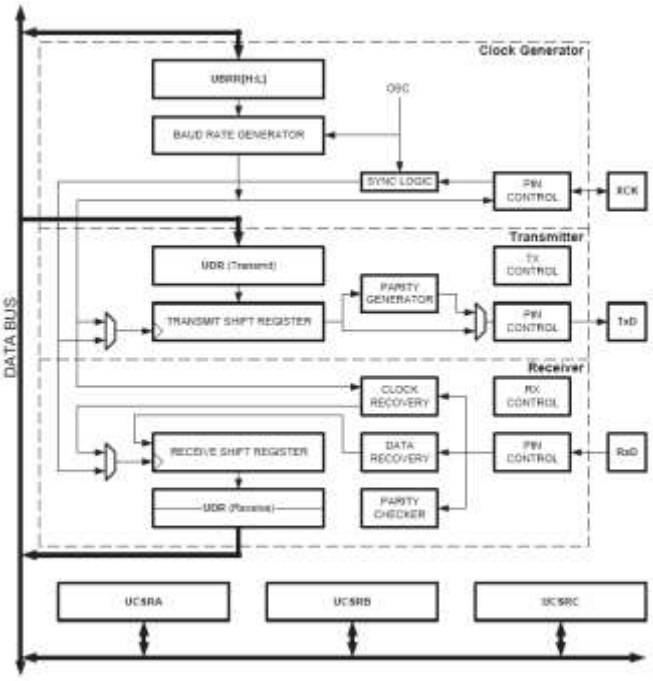
Atmel ATmega8535 je niskonaponski CMOS 8-bitni mikro upravljač baziran na AVR poboljšanoj RISC arhitekturi. Izvršavajući jednu instrukciju u jednom ciklusu takta, ATmega8535 postiže protok od milijun instrukcija u sekundi uz frekvenciju takta 1MHz.

Atmel ATmega8535 ima sljedeća svojstva: 8kB Flash programske memorije, 512B EEPROM memorije, 512B SRAM memorije, 32 ulazno-izlazne linije za opću namjenu, 32 registra za opću namjenu, tri Timera/Brojača sa načinima rada komparacije, unutarnje i vanjske prekidne rutine, serijski programirljivi USART (Universal Syncronus and Asyncronus Reciever and Transmiter) koji koristimo za komunikaciju prema osobnom računalu, ugrađeni oscilator, koji određuje frekvenciju takta, 8 kanalni 10-bitni analogno-digitalni pretvarač koji koristimo za pretvorbu signala ultrazvučne sonde u digitalni signal, i mnogo drugih svojstava, koja nam trenutno nisu bitna. Na slici 3.1 vidimo fizički izgled mikro upravljača i raspored nožica na čipu, prema [3]



Slika 3.1 Prikaz i oznake nožica mikro upravljača

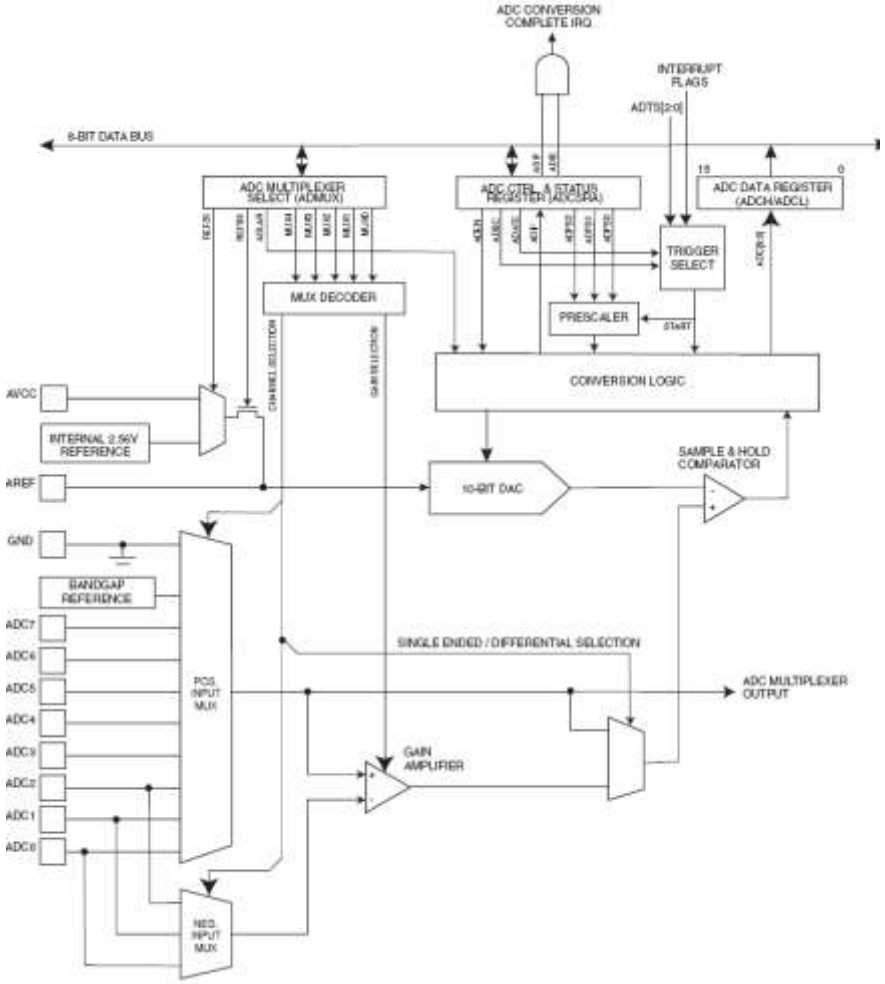
Od nabrojanih svojstava Atmega8535 izravno koristimo samo USART i ADC pretvornik, dok nam ostali sustavi indirektno pomažu. USART je univerzalni serijski asinkroni i sinkroni prijemnik i predajnik, koji je osnova u RS-232 sučelju. On se sastoji od tri dijela: Generator takta, odašiljač i prijemnik, slika 3.2.



Slika 3.2 Unutarnja struktura USART sklopovlja sa registrima

Vidimo da se svaki dio USART-a sastoji od registara i pomoćnih logičkih sklopova, koji u suradnji tvore USART. Postoje četiri glavna registra koji se koriste za slanje i primanje podataka pomoću USART-a a to su: UDR – podatkovni registar u koji se upisuje oktet koji se šalje ili je upisan oktet koji je primljen, UCSRA, UCSRB, UCSRC – upravljački statusni registri, koji nam govore stanje prijema, predaje, te koji se programiraju radi odabira različitih načina rada USART-a, koji će biti opisani u poglavlju programiranja ATMEL-a.

Analogno-digitalni pretvornik je sklop koji pretvara analogni naponski signal u digitalni, koristeći metodu sukcesivne aproksimacije, koja znatno ubrzava proces konverzije. On radi na principu da se analogni napon komparira, umjesto sa rastućim naponom iz DA (digitalno analognog) pretvornika, s mogućim doprinosom pojedinih bitova, i to počevši od najvećega. Sklop se sastoji od naponskog komparatora, registra s DA pretvornikom i kontrolne logike za upravljanje procesom, slika 3.3. [2]

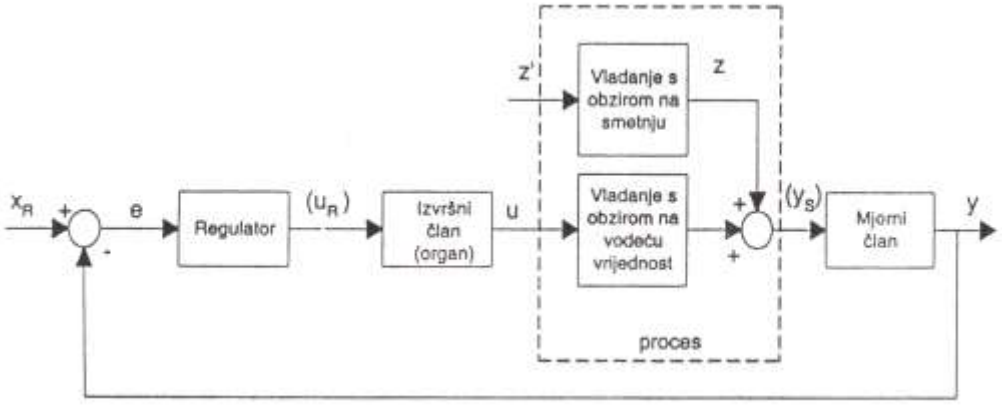


Slika 3.3 Unutarnji prikaz sklopovlja i registara ADC pretvornika

ADC se još sastoji of multipleksera, koji služi za odabiranje pojedinog kanala na kojem želimo izvršiti AD pretvorbu. ADC ima interni referentni naponski izvor od 2.56V, no moguće je povećati referentni napon na napon napajanja mikro upravljača koristeći AVCC ulaz. ADC ima četiri glavna registra: ADMUX - koji služi za odabir kanala na kojemu se vrši pretvorba, ADCSRA – je kontrolni i status registar koji služi za provjeru statusa pretvorbe i za pokretanje pretvorbe idr. Detaljniji rad ADC-a biti će opisan u poglavlju o programiranju ATMEL-a.

3.2 Regulacija razine

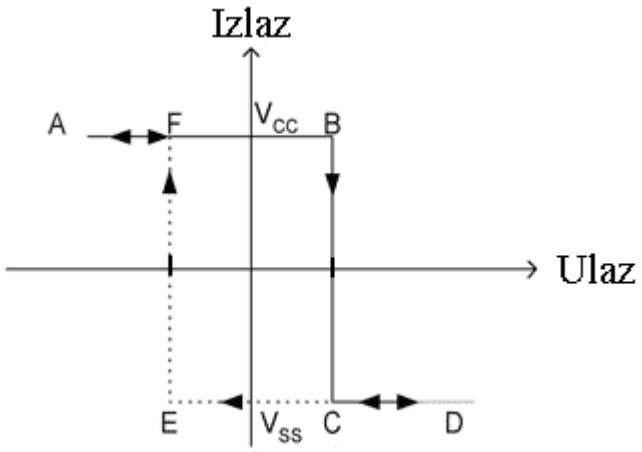
Pojam regulacije susrećemo u mnogim tehničkim disciplinama, od mehanike do elektrotehnike. Regulacija je postupak upravljanja željenom veličinom, na temelju referentne veličine. Načelo regulacije u osnovi nije nikakav tehnički izum. Ovo načelo utjelovljeno je u živim bićima i na temelju kojeg se održavaju na životu. Isto tako, ovo je načelo prisutno i u mnogim drugim procesima – npr. ekonomskim, prema [4]. Počeci automatskog upravljanja sežu iz 18. stoljeća: centrifugalni regulator brzine vrtnje parnih strojeva, J. Watt. Princip regulacije u nekom sustavu prikazan je na slici 3.4.



Slika 3.4 Princip regulacije u nekom sustavu, prema [4]

Danas, područje automatskog upravljanja je toliko široko da ga je nemoguće opisati ukratko. Regulatori se projektiraju za točno određene svrhe, uz točne parametre sustava i ne mogu se primjenjivati za druge sustave, osim za one za koje su projektirani. Regulatori moraju održavati proces (visinu razine u našem slučaju) na referentnoj vrijednosti, uz dozvoljena odstupanja, koja se zadaju tokom projektiranja.

No problem nastupa kada se jedan sustav, kao što je kod nas regulacija razine, uzima općenito, jer ne znamo točne parametre sustava, ne znamo način upravljanja sa izvršnim članovima, također ne znamo koje su vrste izvršni članovi, da li su to npr. crpke koje imaju mogućnost linearnog upravljanja brzinom crpljenja, ili da li su to elektromehanički ventili. Ovdje dolazi do problema projektiranja regulatora standardnim metodama. Zato umjesto korištenja standardnih regulatora kao npr. PID (proporcionalno integralno derivativni) regulator koristimo diskretni tip regulatora, koji radi na principu komparatora sa histerezom, slika 3.5.



Slika 3.5 Rad komparatora sa histerezom

Kod ovog tipa regulatora ulaz u regulator je kontinuiran, a izlaz je diskretan sa dva stanja, '1' ili '0' (Vcc i Vss). Regulator radi na principu zadavanja dvije razine, razina uključivanja i razina isključivanja, a cilj je da regulirana veličina bude unutar definiranih razina. Uzmimo za primjer sliku 3.5. Na slici se vide dvije definirane razine, razina uključenja je potez E-F, jer se pri toj razini stanje mijenja iz '0' (Vss) u '1' (Vcc), a razina isključenja potez B-C, jer se pri toj razini stanje mijenja iz '1' (Vcc) u '0' (Vss). Na os apscise nanose se podatci trenutne razine, dok se sa osi ordinate očitava vrijednost izlaza. Uzmimo za primjer da je trenutna regulirana veličina (razina) otprilike na položaju D i da počinje opadati prema C. Regulirana veličina će opadati sve do položaja E, kada će regulator promijeniti stanje izlaza u '1' (Vcc), (prijelaz iz E u F) i time uključiti izvršni član, kojemu je zadatak da povisi reguliranu veličinu (razinu). Kada regulirana veličina dosegne gornju granicu, položaj B, regulator mijenja izlazno stanje u '0' (Vss) čime onemogućuje izvršni član, i zaustavlja rast regulirane veličine (razine). Ovaj tip regulatora je primjenjiv u skoro svim područjima regulacije razine, ali njegov nedostatak je nemogućnost linearne regulacije izvršnog člana. No u mnogim područjima regulacije razine nije moguće ostvariti linearnu prijenosnu karakteristiku izvršnog člana, (solenoidi, kompresori, elektromagnetski ventili itd.) tako da je ovaj način regulacije najpovoljniji.

Ponekad u regulaciji dolazi do problema da se regulirana veličina ne može održati na referentnoj vrijednosti, da li zbog prevelike smetnje u sustavu, ili zbog nedovoljnog djelovanja izvršnog člana. Tada je dobro postaviti određene sigurnosne točke uzbune na razini koju mjerimo. Te točke mogu biti iznad ili ispod referentne razine, ovisno o načinu rada izvršnog člana i o opasnosti od prevelike ili premale razine. Te točke uzbune aktiviraju redundantne izvršne uređaje ili zaustavljaju proces da ne bi došlo do oštećenja postrojenja.

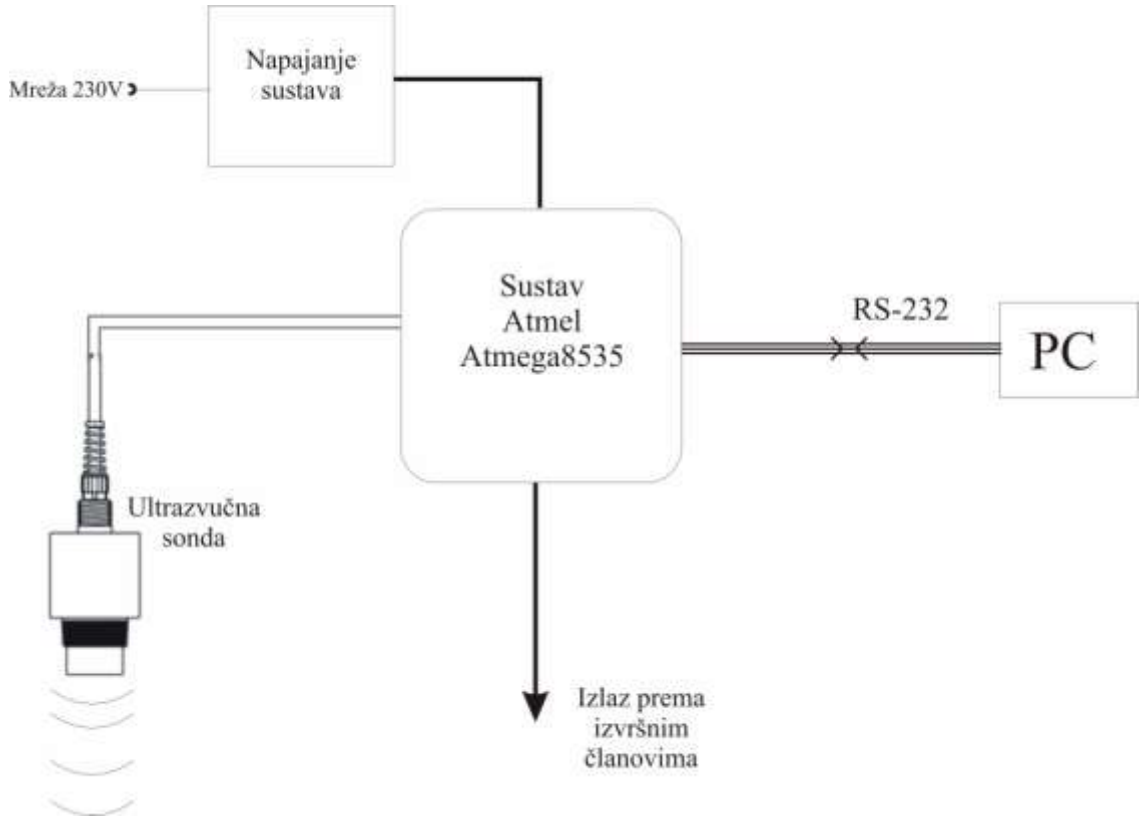
3.3 Prikaz i upravljanje pomoću osobnog računala

Sustav regulacije i mjerenja razine ima predviđenu vezu sa osobnim računalom, radi mogućnosti prikaza stanja razine spremnika i mogućnosti programiranja razina regulacije te položaje alarma. Sustav se spaja sa osobnim računalom preko serijske veze, i RS232 protokola za razmjenu informacija.

Razmjena radi na principu da osobno računalo šalje niz podataka, u kojim se nalaze programirane razine alarma i regulacije, koje se programiraju u računalnom programu na osobnom računalu. Nakon što je osobno računalo poslalo podatke, ono čeka odgovor od sustava mjerenja i regulacije. Gledajući od strane sustava, računalo šalje poruku sa zadanim razinama i alarmima, sustav prima i sprema potrebne podatke u memoriju. Nakon toga vraća poruku osobnom računalu, u kojoj su podatci o trenutnoj razini i potvrdni podatci o alarmima i razini regulacije. Ovaj način upita i odgovora odvija se sve dok je računalni program povezan sa sustavom regulacije i mjerenja, a kada se osobno računalo odvoji od sustava za regulaciju i mjerenje, nastavlja sa normalnim radom samo bez prikaza trenutne razine na zaslonu osobnog računala. Važno je naglasiti da spoj računala i mikro upravljačkog sustava nije neophodan za rad regulacije, nego je samo dodatna funkcija za vizualizaciju razine i korekciju željene razine regulacije.

3.4 Blok dijagram sustava ultrazvučnog mjerenja i regulacije razine

Kako smo u poglavljima prije razmotrili svaki aspekt ovog sustava, sada ga možemo spojiti u cijelo. Povezujući ultrazvučnu sondu, Atmel ATmega8535 i osobno računalo dobivamo sustav ultrazvučnog mjerenja i regulacije razine. No da bi se mogli svi dijelovi uspješno povezati potrebni su određena prilagođena i dodatni pasivni elementi. Ovdje ćemo teoretiziranja radi ispustiti te elemente i prikazati blok dijagram sustava mjerenja i regulacije razine, gdje će biti vidljiv način rada mjerenja i regulacije pomoću ultrazvučne sonde i osobnog računala. Blok dijagram je prikazan na slici 3.6.



Slika 3.6 Izgled sustava mjerenja i regulacije razine sa Atmel Atmega8535

4. ZASNIVANJE SUSTAVA

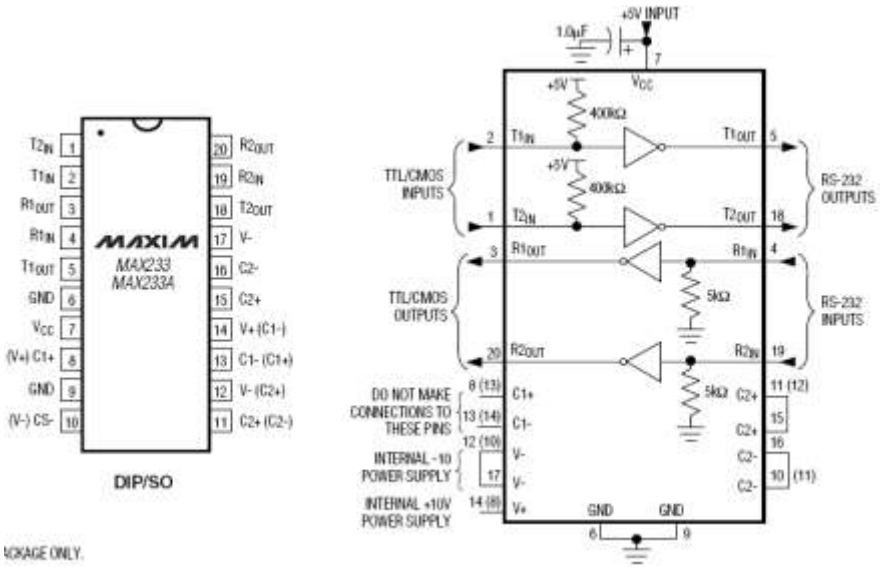
Nakon što smo prošli teoretske dijelove mjerenja i regulacije, prelazimo na projektiranje elektroničkog sklopa prije programiranja mikro upravljača. U ovom poglavlju ćemo obraditi spajanje ultrazvučne sonde sa Atmel-om, spajanje osobnog računala sa Atmel-om, spajanje izvršnih članova sa Atmel-om, projektiranje napajanja i projektiranje ostalih potrebnih sklopova za normalan rad cijelog sustava.

4.1 RS232 sučelje

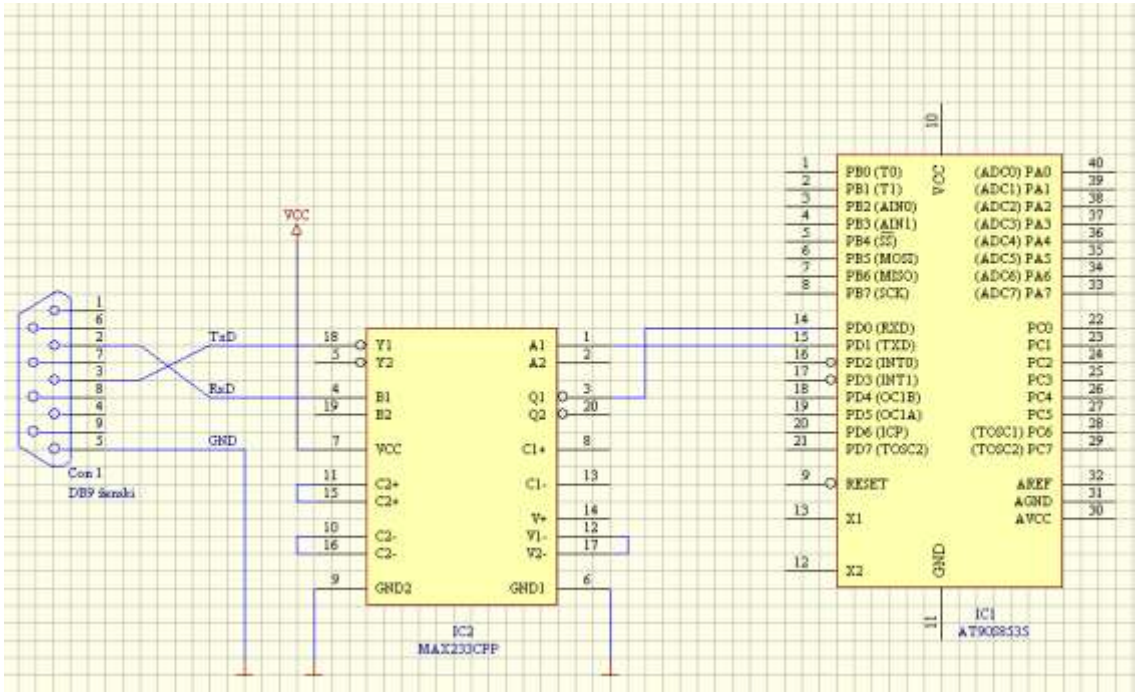
RS232 je serijsko sučelje koje koristi napone od -3V do -12V za logičku jedinicu, i napone od 3V do 12v za logičku nulu. Pošto se ti naponi puno razlikuju od standardnih 5V CMOS napona, potreban je pretvarač, koji će omogućiti pretvorbu CMOS u RS232. Taj sklop mora imati svojstvo da CMOS logičku jedinicu od 5V, pretvori u logičku jedincu RS232, koja mora biti u području od -3V do -12V i također da CMOS logičku nulu od 0V, prenese u RS232 područje od 3V do 12V.

Postoji mnogo izvedbi ovog sklopa no najjednostavnija i najčešće korištena je integrirana verzija od proizvođača Maxim. Ta izvedba u sebi sadrži dvije nabojne pumpe. Prva podiže napon sa +5V na otprilike +10V dok druga invertira +10V da bi se dobio negativni napon od -10V. Najčešće su nabojne pumpe neregulirane, što rezultira da napon od 10V padne puno ispod njegove nazivne vrijednosti, što ovisi o dužini RS232 vodova i njihovom kapacitetu. Druge inačice integriranih sklopova imaju regulirani napon ali nije veći od 5.4V, što je dovoljno, ako je napon stabilan. Također, ovisno o izvedbi postoje inačice sa eksternim kapacitetima za nabojne pumpe, koji se koriste za veće udaljenosti i inačice sa integriranim kapacitetima, za manje brzine i kratke udaljenosti[5].

Pošto nama brzina prijenosa ne igra veliku ulogu, a niti udaljenost možemo izabrati inačicu sa nereguliranim nabojnim pumpama i integriranim kondenzatorima. Naziv IC-a: MAX233. Prema slici 4.1 vidimo raspored nožica, i unutrašnji raspored sklopova te prema tome spajamo na Atmel ATmega8353.[6]



Slika 4.1 Raspored nožica i unutrašnja struktura Max233 [6]



Slika 4.2 Shema spajanja mikro upravljača sa serijskim sučeljem pomoću Max233

Na slici 4.2 vidimo shemu spoja RS232 sučelja sa mikro upravljačem Atmel ATmega8535. Ulazi i izlazi iz integriranog sklopa Max233 spojeni su na mikro upravljač na pinove 14 i 15, jer su to točno određeni pinovi za RS232 USART sučelje, dok su na drugoj strani spojeni na konektor na pinove 2 i 3, i to na taj način, da je za spajanje računala sa mikro upravljačem potreban poseban tip serijskog kabla, tzv. Nul-Modem kabl, kojemu su Txd i Rxd veze unakrsno spojene.

4.2 Spajanje ultrazvučne sonde s mikro upravljačem

Kao što smo vidjeli u prijašnjim poglavljima, ultrazvučna sonda ima analogni strujni izlaz od 4mA do 20mA, a Atmega8535 ima naponski ulaz u analogno digitalni pretvornik. Stoga potreban nam je sklop koji pretvara strujni izvor u naponski, a to je najjednostavniji otpornik spojen paralelno sa ulazom u ADC i masom. Da bi mogli izračunati otpornik moramo si postaviti pitanje što je najvažnije kod pretvorbe analognog strujnog signala, u analogni naponski signal, te potom u digitalni signal? Da bi pretvorba bila što točnija, razlika napona dobivena iz stanja od 4mA, do stanja 20mA, na ulazu u ADC mora biti što veća, uz uvjet da pri stanju od 20mA napon ne bude veći od 5V.

$$\Delta U = I_2 R - I_1 R = R \cdot (I_2 - I_1) = R \cdot 16 \cdot 10^{-3} V \tag{4-1}$$

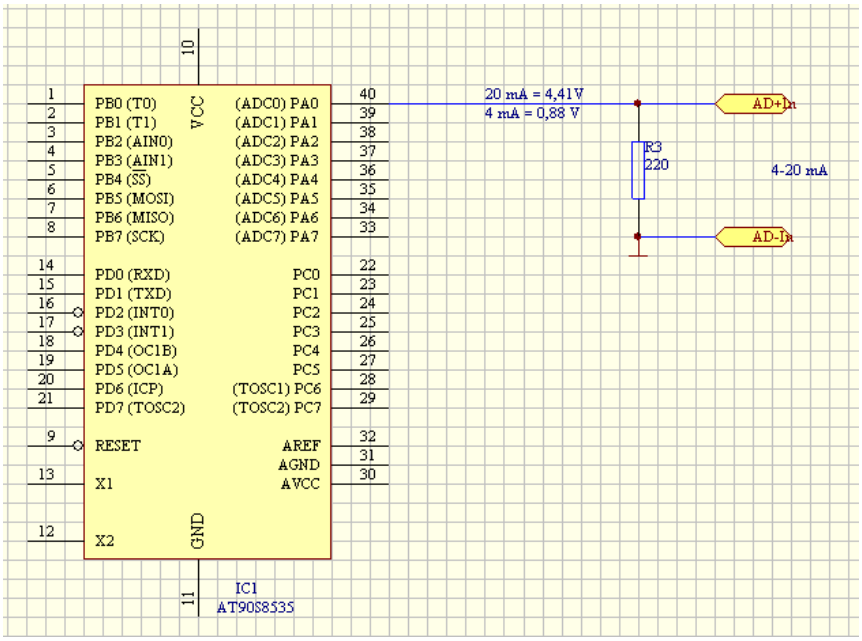
Iz jednadžbe vidimo da će razlika napona biti to veća što je otpor R veći, ali moramo uzeti u obzir ograničenje sklopa ADC-a na 5V, stoga treba izračunati koliki će otpor biti ako je napon na ADC-u 5V, a struja najveća moguća 20mA.

$$R = \frac{U_2}{I_2} = \frac{5V}{20mA} = 250\Omega \tag{4-2}$$

Kako otpornik od 250 Ohm-a ne postoji po E seriji (E6 za 10% tolerancije), uzimamo prvi manji koji je 220 Ohm-a i računamo napone za stanje 4mA i stanje 20mA.

$$\begin{aligned} U_1 &= I_1 R = 4mA \cdot 220 = 0.88V \\ U_2 &= I_2 R = 20mA \cdot 220 = 4.4V \end{aligned} \tag{4-3}$$

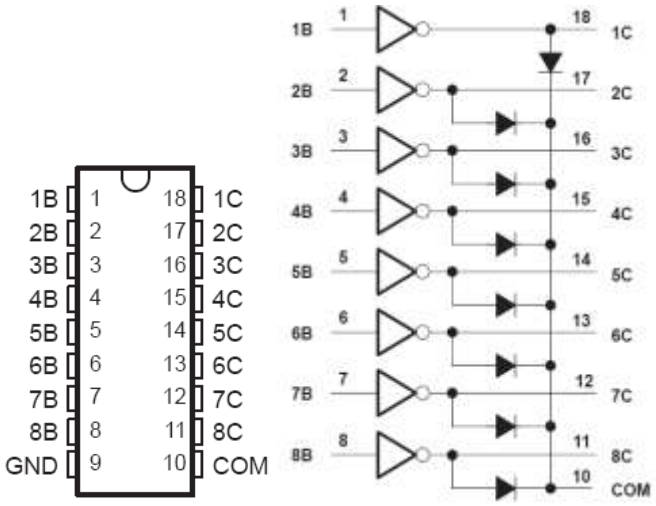
Sada znamo naponske razine koje odgovaraju strujnim razinama od 4mA i 20mA, te pri pisanju programa mikro upravljača možemo jednostavno napraviti prijelaz, iz naponskog područja, u razinu spremnika kojeg mjerimo sa ultrazvučnom sondom. Slika spoja sonde sa Atmega8353 prikazana je na slici 4.3.



Slika 4.3 Shema spoja ultrazvučne sonde sa mikro upravljačem

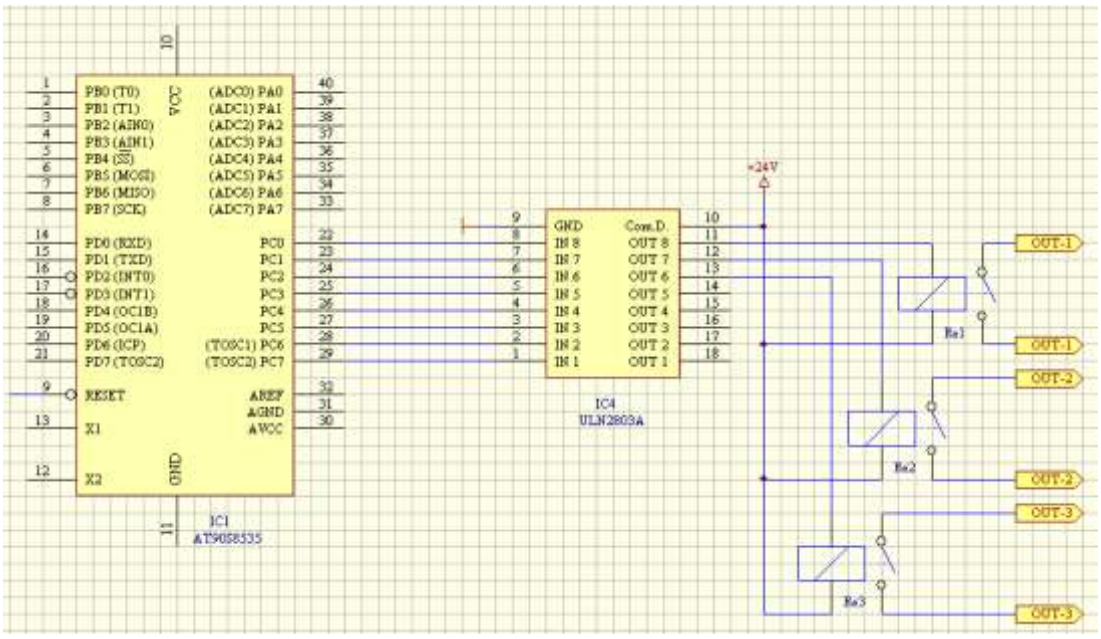
4.3 Sučelje prema izvršnim članovima

Da bi regulator funkcionirao, mora imati sučelje prema izvršnim članovima. Kao što smo vidjeli, ovaj tip regulatora ima diskretne izlaze, što znači da izlazi mogu poprimiti stanje '1' ili '0'. Za takvo upravljanje najjednostavnije je koristiti elektromehaničke sklopke zvane releji, koji imaju svojstvo propuštanja velikih struja i napona, nevezanih za točno određeni krug, tj galvanski odvojeni od sklopa mikro upravljača. No releji su dizajnirani kao mali elektromagneti koji troše od 10mA do 100mA, ovisno o tipu releja. Stoga je potrebno napraviti izlazni stupanj koji će pobuđivati releje, koristeći signale iz mikro upravljača. Najčešća izvedba ovih izlaznih stupnjeva su tranzistorske sklopke, no ako imamo veći broj releja dolazi do nepraktičnosti. Zato se koriste integrirane tranzistorske mreže, uparene točno za ovakve tipove opterećenja, kao što su releji. Mogu dati puno veće struje od CMOS izlaznih stupnjeva, imaju zaštitu od visokih napona i kompatibilne su sa CMOS pobudom, bez dodatnih komponenata. Jedna od češćih tranzistorskih mreža koju koristimo i ovdje je ULN2803, slika 4.4.[7].



Slika 4.4 a) Raspored nožica i b) unutrašnja struktura ULN2803

Prema slici 4.4, vidimo da postoji mreža dioda, koje se koriste za zaštitu sklopa od visokih napona, nastalih isključenjem releja. Zaštitne su diode orijentirane na način, da je COM izvod spojen na veći potencijal, te prema tome releji moraju biti spojeni od izlaza (xC) prema većem potencijalu, paralelno sa diodama, zbog samoindukcije koja nastaje pri isključivanju napona sa zavojnice releja. Ulaze (xB) spajamo na mikro upravljač, na željeni port, u ovom slučaju PortC, i to možemo spojiti sve ulaze bez greške, jer time ostavljamo mogućnost nadogradnje u budućnosti. U ovome regulatorskom sustavu koristit ćemo tri releja, od toga jedan za kontrolu glavnog izvršnog člana, a ostala dva za uzbunjivanje niske ili visoke razine u spremniku, ili za redundantne izvršne članove. Slika 4.5 prikazuje shemu spajanja.



Slika 4.5 Shema spajanja sučelja prema izvršnim članovima

4.4 Središnji dio sa mikro upravljačem

U središnjem dijelu postoje pasivne komponente, koje omogućuju rad mikro upravljača. Tri su dijela mikro upravljača kojega čine ove komponente: oscilator takta, sustav ponovnog pokretanja (RESET) i referentni napon ADC pretvarača.

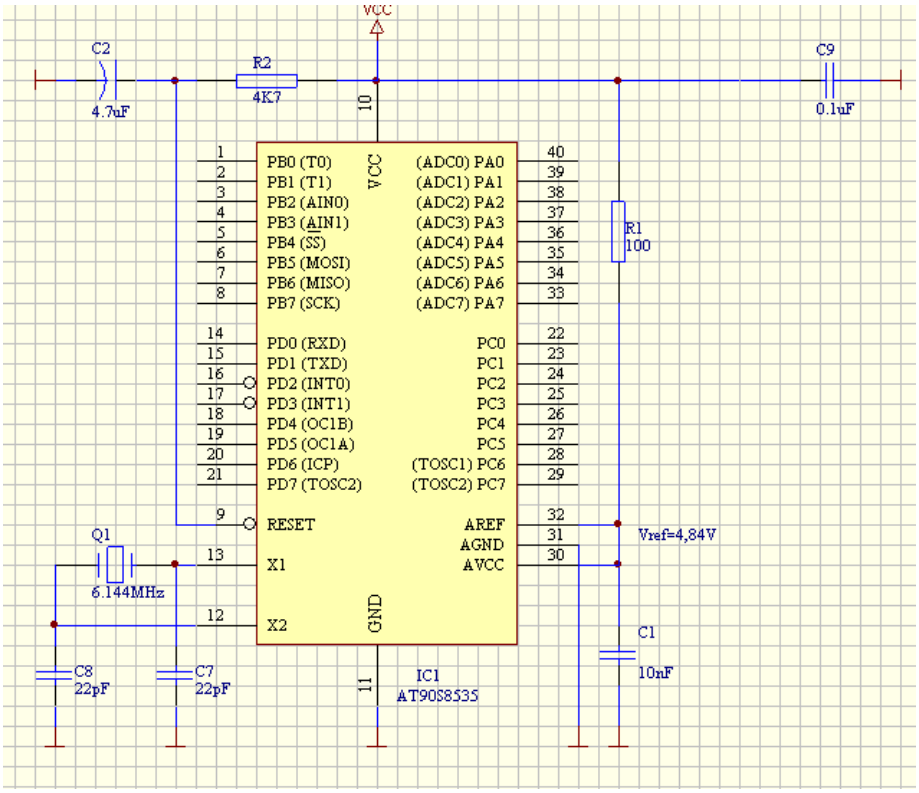
Oscilator takta, kod mikro upravljača, koristi kristal da bi osigurao što stabilniju frekvenciju takta, koja je važna za izvođenje svih instrukcija unutar mikro upravljača, a posebno za RS232 prijenos, jer je on asinkroni. Frekvencija kristala kvarca koji koristimo, mora odgovarati cijelom višekratniku od frekvencije RS232 porta, jer ako ne odgovara, pojavljuje se greška u prijenosu podataka na RS232 sučelju. Najčešće korištene frekvencije su 4MHz, 11.0592MHZ, 6.144MHZ itd. no pošto nam 4MHz ne daje cijeli višekratnik, a 11.0592MHz je relativno visoka i nepotrebna frekvencija, najbolji izbor daje kristal od 6.144MHz. Da bi oscilator pravilno radio, potrebno je spojiti kondenzatore veličine 22pF, od izlaza kristala prema masi, prema [3].

Sustav ponovnog pokretanja ili RESET sustav služi ne samo za ponovno pokretanje, nego i za pokretanje mikroprocesora općenito. Nakon što je uključeno napajanje i napon dođe od mikro upravljača, potreban je određeni period da bi istitravanja i nestabilnosti napajanja iščezle. U tom periodu mikro upravljač mora biti ugašen, tj. RESET nožica mora biti na potencijalu nule. To se postiže na način da se spoji serijski spoj otpornika i kondenzatora na pin RESET, koji omogućuju da mikro upravljač starta nakon što se napajanje stabilizira. Uobičajeni izbor komponenata je 4,7uF i 4,7kOhm, što ovisi o kvaliteti napajanja i količini smetnji.

Analogno-digitalni pretvornik koristi vanjski referentni napon, s kojim uspoređuje analogni signal na ulazu u pretvarač, stoga je važno da referentni signal bude stabilan. Da bi se referenti signal stabilizirao od smetnji napajanja i od samog oscilatora mikro upravljača, koristi se nisko propusni RC filter sa vrijednostima otpora 100Ohm, i kondenzatora 10nF. Zbog unutrašnjeg otpora, referentni napon pada na otprilike 4.8V, što je u granicama.

Kapacitet od 100nF spojen između napajanja i mase, osigurava, da se uklone vršci napona koji nastaju naglim preklapanjem, te naglom promjenom struje u mikro upravljaču, koja je posljedica prebacivanja sklopova, iz logičkih stanja 1 u 0, i obratno.

Shema središnjeg dijela sa mikro upravljačem prikazana je na slici 4.6



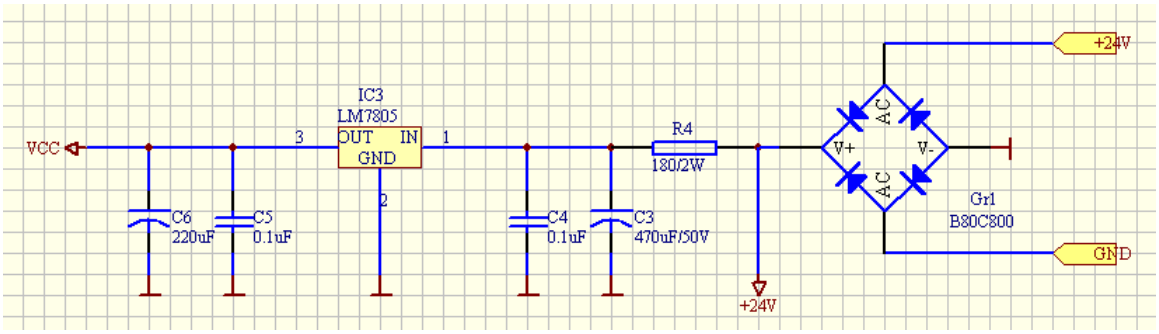
Slika 4.6 Shema spajanja dodatnih pasivnih komponenata na AtMega8535

4.5 Napajanje

Neophodni dio svakog elektroničkog uređaja je izvor napona tj. napajanje. Napajanje mora biti projektirano da zadovoljava određene uvjete: stabilnost napona, mala izmjenična komponenta, dovoljan izvor struje.

Mikro upravljači rade na naponu od 5V, imaju jako puno preklapanja, stoga napajanje mora imati svojstvo filtriranja vrhova napona. Pošto je potrošnja kod mikro upravljača relativno mala, napajanje nema velike zahtjeve za visoku struju. Ostatak integriranih krugova nisu veliki potrošači, pa se prema [3, 6, 7] potrošnja procjenjuje na maksimalnih 100mA.

Prema tome, najjednostavnija izvedba napajanja i najčešće korištena je stabilizacija napona sa integriranim krugom LM7805. Iznose kapaciteta i princip rada može se pronaći u [8], a sama izvedba ne zahtjeva puno pasivnih komponenata. Napon prije stabilizacije iznosi 24V i koristi se za napajanje ultrazvučne sonde i izlaznih releja. Shema napajanja prikazana je na slici 4.7



Slika 4.7 Shema spajanja ispravljača

4.6 Cjelokupni sustav ultrazvučnog mjerenja i regulacije

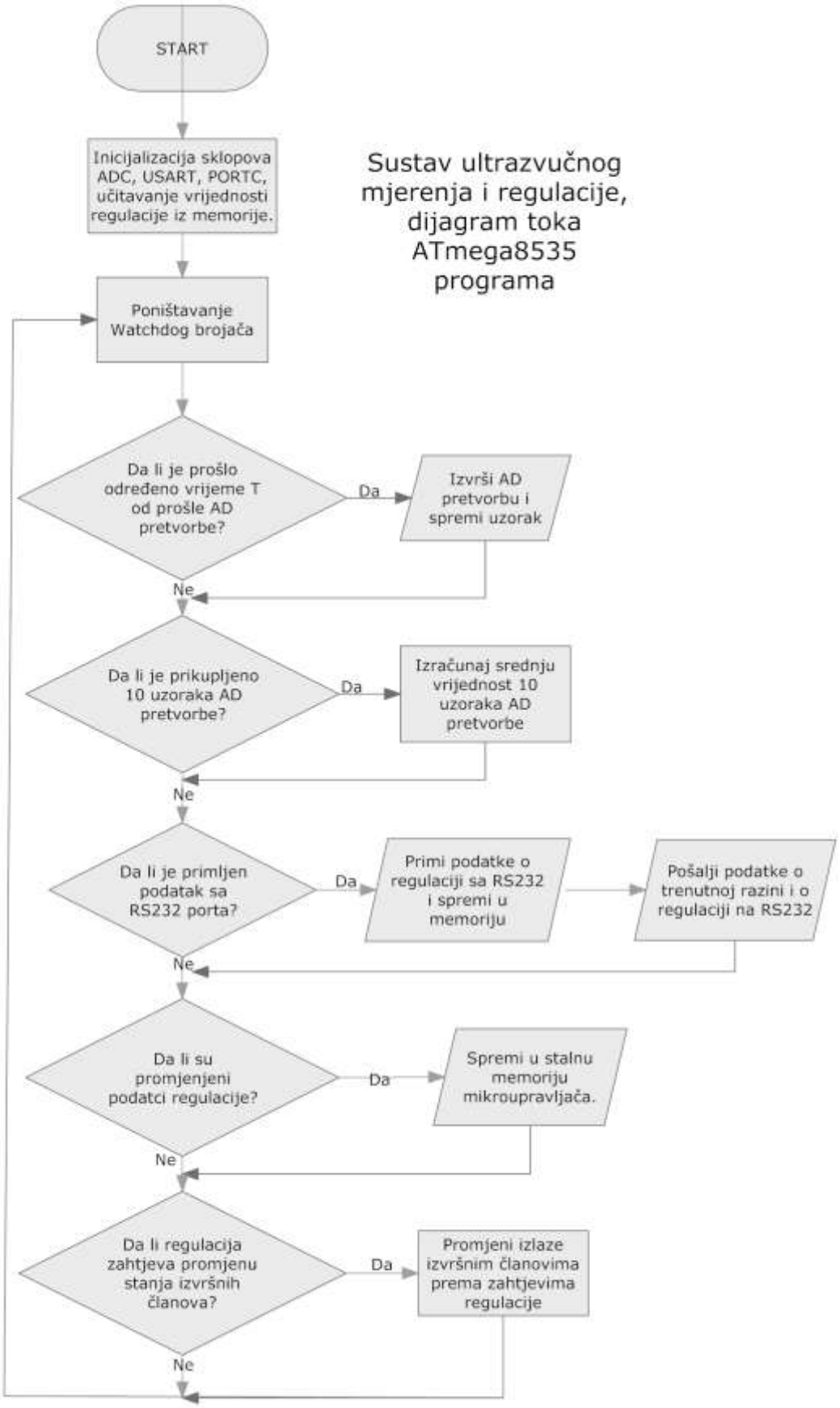
Nakon obrađenih svih dijelova sustava ultrazvučnog mjerenja i regulacije možemo prikazati njegovu potpunu električnu shemu, prilog 4.1.

5. PROGRAMSKA PODRŠKA

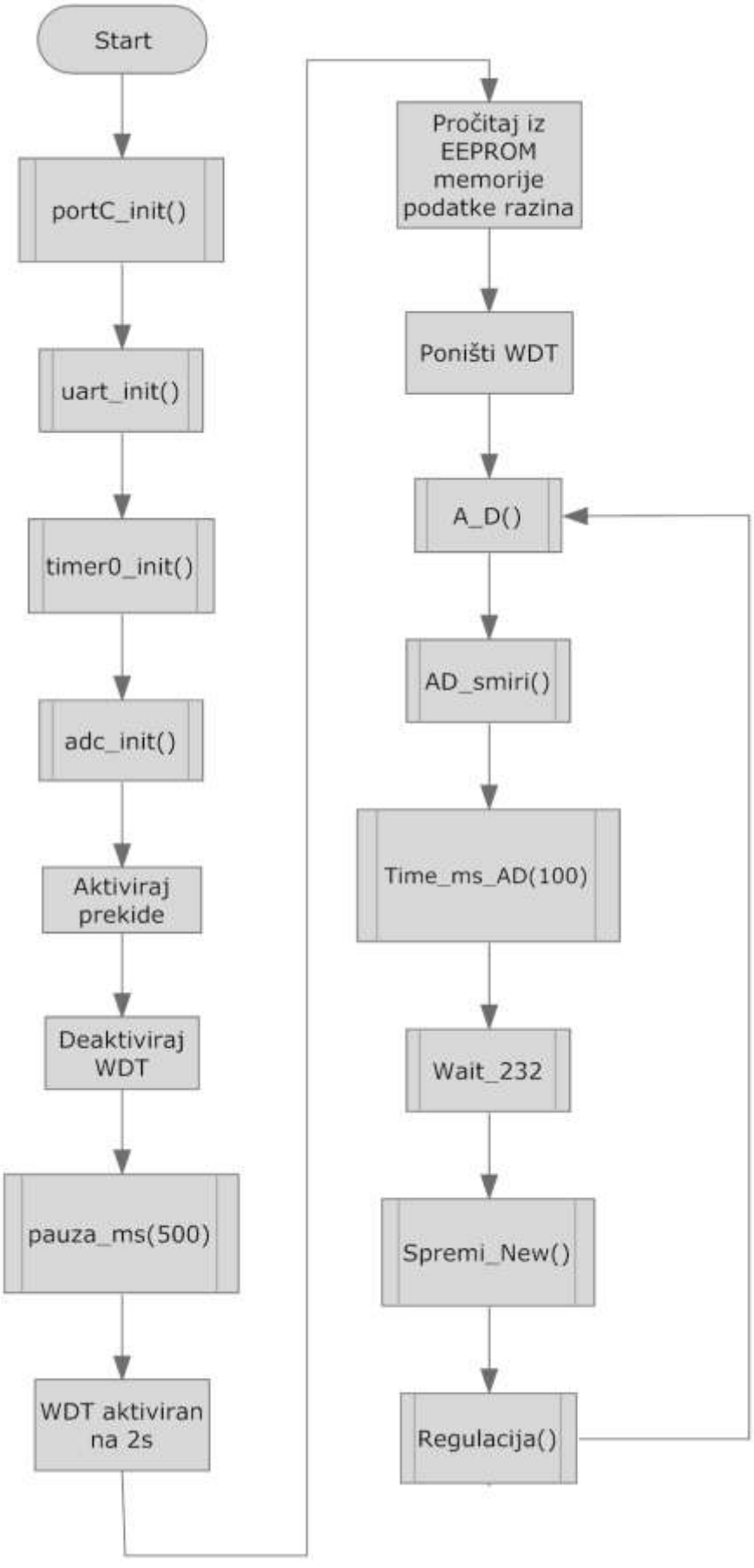
Prolazeći kroz prijašnja poglavlja, vidimo sve potrebne teorijske osnove i izvedbene dijagrame za izradu i simulaciju sustava ultrazvučnog mjerenja i regulacije. No glavni dio ovog sustava nije u sučeljima i pasivnim komponentama, nego u programu koji čini mozak ovog sustava, Atmel ATmega8535 funkcionalnim. Gledajući Atmel ATMega8535 kao aktivnu komponentu bez programa, on je kao računalo bez operativnog sustava, funkcionalan ali beskoristan. Da bi ovaj sustav mogao funkcionirati, potrebno je napisati program za ATMega8535, koji će utjelovljivati svojstva mjerenja, regulacije, sučelja sa ultrazvučnom sondom, sučelja sa računalom i sučelja sa izvršnim uređajima. No osim programa predviđenog za ATmega8535, potrebno je napisati aplikaciju za osobno računalo, sa ciljem povezivanja sa mikro upravljačem. Početi ćemo sa najvažnijim programom, a to je za mikro upravljač ATMega8535.

5.1 Program Atmel ATmega8535

Najvažniji dio ovog sustava je program za mikro upravljač. Program mora pokrivati nekoliko cjelina i sve ih objediniti u jednu funkcionalnu cjelinu, koja se upisuje u mikro upravljač. Da bi si olakšali pisanje programa, umjesto programiranja u asembleru, koristiti ćemo prevodilac iz programskog jezika C++. Program koji ima integrirani prevodilac iz jezika C++ u asembler, i u binarnu datoteku potrebnu za upisivanje programa u mikro upravljač, je AVRStudio. Ovaj program koristi bibliotečne datoteke programirane sa mnogim naredbama, koje koriste mikro upravljači i time nam olakšava rad sa registrima, USART-on, ADC-om itd. Program ćemo podijeliti u nekoliko cjelina te na kraju integrirati. Grubi prikaz rada programa prikazan je na slici 5.1 koja prikazuje blok dijagram programa i glavne odluke koje se provode. Ovo je simbolički prikaz rada, dok će pravi program djelomično odstupati od dan strukture. Princip rada je sljedeći: Nakon inicijalizacije potrebnih sklopova, mikro upravljač ulazi u beskonačnu petlju u kojoj se stalno provjeravaju zadani parametri. Prvo nastupa poništavanje budilice (Watchdog timer), koja ima svrhu da ponovo pokrene mikro upravljač i sam program, ako program prestane raditi prema željenim parametrima. Ako se nakon određenog vremena, budilica (Watchdog timer) periodički ne poništi, ona rezultira ponovnom pokretanju mikro upravljača. Nakon toga počinju sa radom ostali potprogrami, koje ćemo detaljno objasniti u daljnjim poglavljima, a prikazani su na slici 5.2.



Slika 5.1 Pojednostavljeni dijagram toka mikro upravljača



Slika 5.2 Glavni dijagram toka programa

5.1.1 Analogno-digitalna pretvorba

AD pretvorba je proces koji obuhvaća pretvorbu analognog signala u digitalni, no prije same pretvorbe potrebno je podesiti sklopovlje analogno digitalno pretvornika. U poglavlju 3.1. opisano je načelo rada ADC-a, ali sada to moramo prenijeti u praksu, na način da programiramo određene registre sa određenim podacima i na taj način podesiti ADC pravilnom radu. Kao što smo vidjeli ADC ima četiri registra, od kojih su dva podatkovni registri ADCH i ADCL koji nam vraćaju vrijednost analognog napona. Ostala dva registra su registar multipleksera ADMUX koji ostaje na vrijednosti 0 jer on upućuje koji ADC kanal odabiremo, a drugi registar je ADCSR, upravljački i statusni registar u koji upisujemo vrijednost 10000110 što prema [3, str 221] aktivira ADC i postavlja djelitelj takta pretvornika sa sukcesivnom aproksimacijom na 64. To obavlja potprogram za inicijalizaciju ADC-a naziva adc_init(). Nakon inicijalizacije sklopova način uzimanja uzorka analognog signala je sljedeći: u registar ADCSR se upisuje vrijednost 1 na mjesto bita sa oznakom ADSC, koji označuje početak konverzije analognog u digitalno. Nakon što je započela konverzija program čeka da se konverzija završi i iz registra ADCH i ADCL čita vrijednost, s tim da je ADCH registar sa gornjim bitovima rezultata, a ADCL registar sa donjim bitovima. Konverziju radi potprogram nazvan AD_konv(). No pošto nije potrebno da se pretvorba događa stalno, nego svakog određenog vremenskog intervala, dodaje se potprogram vremenske baze, koji svakih 100ms dopušta programu da započne konverziju, na način da promjeni vrijednost zastavice Time_Tick_AD, koja omogući početak pretvorbe. Potprogram vremenske baze ne zaustavlja program na način da čeka zadano vrijeme pa onda ponovo pokreće program, jer time bi se smanjila dostupnost serijskog porta i podataka osobnom računalu, nego on omogućava tok programa u svim potprogramima u petlji, a onemogućuje samo rad AD pretvornika. Način na koji potprogram vremenske baze računa točno vrijeme, je na temelju brojača Timer0, koji se podešava tako da se aktiviraju prekidi programa na preliv brojača. U podatkovni registar brojača se upisuje početna vrijednost, koja se svaki put kada se brojač prelije upisuje ponovo. Time se stvara generator frekvencije ovisno o djelitelju takta koji napaja brojač, frekvencije koja se dijeli i o početno upisanoj vrijednosti. Ako znamo frekvenciju koja je 6.144MHz, i uzmemo da je djelitelj te frekvencije 1024, tada ako upišemo u registar vrijednost 249 (255-(6144000/1024)/1000) dobivamo da brojač pravi preliv svake milisekunde i poziva potprogram SIGNAL (SIG_OVERFLOW0), koji zbraja nadolazeće impulse u varijablu brojp_AD i time se omogućuje točno određivanje trajanja pauze. Potprogram za inicijalizaciju brojača je timer0_init().

AD pretvornik uzima analogni signal i vraća digitalni u rasponu od 0-1024, što poistovjećuje naponsku razinu od 0-Uref. Pošto naša sonda daje napon, od 0.88V do 4.4V na ulazu ADC-a, moramo programski prilagoditi da izmjereni napon od 0.88-4.4V, odgovara postotcima 0-100%:

$$ADC = \frac{U_{ADC}}{U_{REF}} \cdot 1024 \rightarrow U_{ADC} = \frac{ADC \cdot U_{REF}}{1024}$$

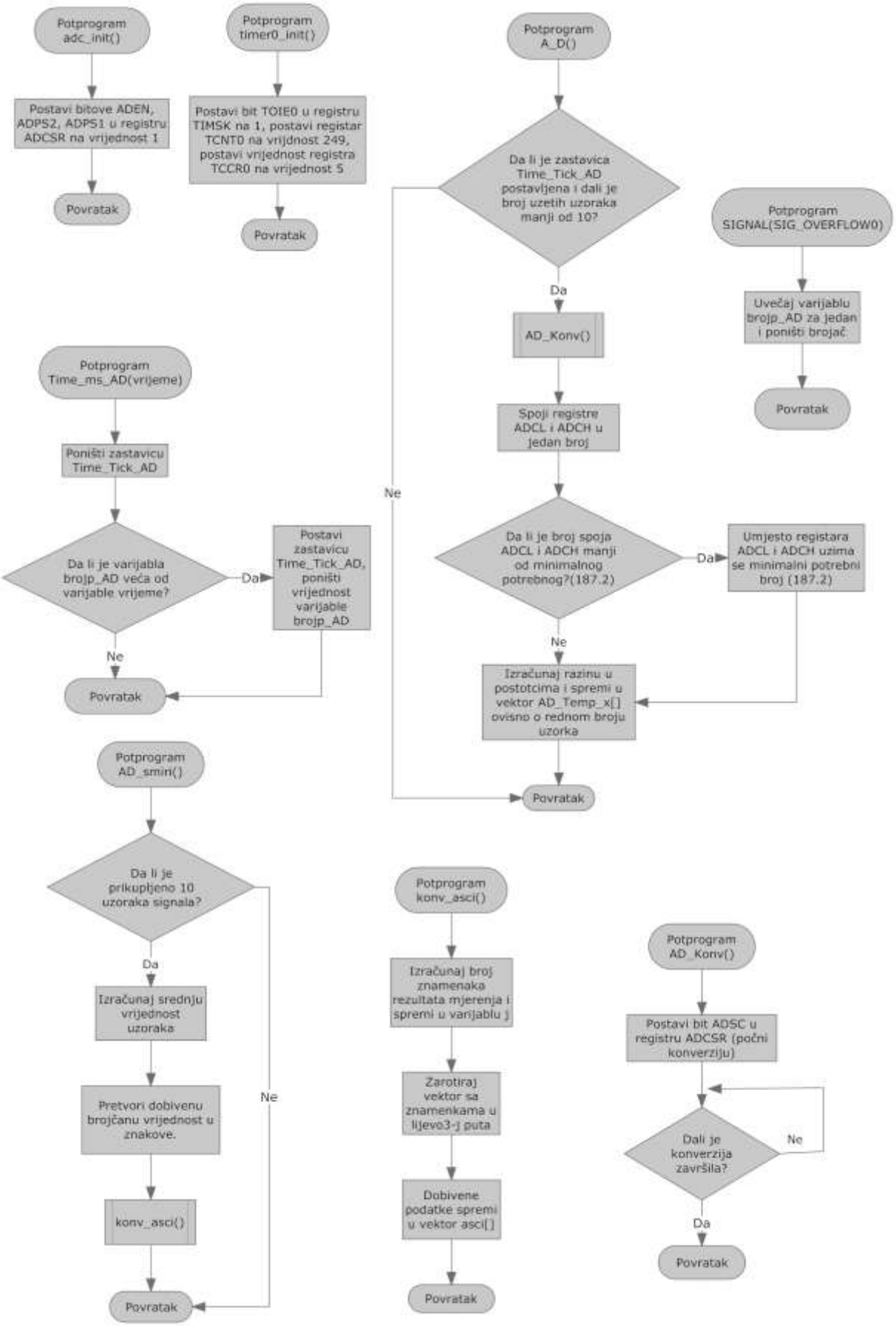
$$R(\%) = \frac{U_{ADC} - 0.88}{3.52} \cdot 100\% = \frac{\frac{ADC \cdot U_{REF}}{1024} - 0.88}{3.52} \cdot 100\% = \left| U_{ref} = 4.84V \right| = \frac{ADC - 186.2}{7.45}\% \quad (5\text{-}1)$$

gdje je ADC-vrijednost registara ADC-a, Uadc- analogni napon koji se pretvara (ulaz), Uref- referentni napon ADC-a, R(%)- razina spremnika u postotcima.

Nakon uzimanja uzorka analognog signala, potprogram A_D() uzima uzorke i dalje, u razmaku definiranom potprogramom vremenske baze i to 10 uzoraka, te ih sprema u vektor AD_Temp_x[], na kojima se kasnije računa srednja vrijednost da bi se smanjila greška i smirilo osciliranje signala.

Srednju vrijednost računa potprogram AD_smiri(), kojemu je glavna zadaća da spriječi osciliranje signala, koje je posljedica greške AD pretvornika ili je posljedica varijacije stvarne razine. Uzimanjem srednje vrijednosti skupa uzoraka, smanjujemo mjernu nesigurnost mjerenja. Potprogram AD_smiri() računa aritmetičku srednju vrijednost i sprema vrijednost u varijablu AD_Temp. Nakon izračunate srednje vrijednosti, brojčana vrijednost razine pretvara se u znakovnu vrijednost, koja se dalje šalje osobnom računalu preko serijskog RS232 porta. Pretvaranje iz brojčane varijable u znakovni vektor vrši se pomoću naredbe „itoa(brojčani podatak, znakovni vektor, brojčana baza)“ koja vraća rezultat u znakovnom vektoru. No ovdje problem nastaje u tome što se vrijednost razine mijenja od jednoznamenkastog do troznamenkastog broja, a naredba „itoa()“ vraća cijeli broj u obliku znakova, u troznamenkasti vektor, ali bez potrebnih razmaka ispred broja. Takav rezultat je nepregledan i nejasan, te da bi se poboljšao izgled ispisa koristimo potprogram konv_asci(), koji rotira znamenke, ovisno dali je broj jednoznamenkasti ili dvoznamenkasti, da bi se dobili potrebni razmaci ispred broja. Nakon izvršenih korekcija, razina u obliku znakovnog troznamenkastog vektora sprema se u vektor aski[], koji je tada spreman za slanje na osobno računalo, koristeći RS232 serijsku vezu.

Prikaz dijagrama toka potrebnih za analogno digitalnu pretvorbu prikazan je na slici 5.3



Slika 5.3 Blok dijagrami potprograma korištenih za AD pretvorbu

5.1.2 RS232 Komunikacija

RS232 komunikacija obuhvaća dvije faze, a to su prijem i predaja podataka. RS232 je u načelu serijska veza, stoga svi podatci koji se šalju ili primaju moraju se slati u slijedu. Slijed podataka koji govore o razinama regulacije i trenutnoj razini nazivamo paket. Svaki paket sastoji se od zaglavlja, podatkovnog dijela i od kraja paketa. Zaglavlje i podatkovni dio razlikuju se u ovisnosti da li je paket poslan od strane osobnog računala, ili od strane mikro upravljača. Prijenos podataka započinje na način, da osobno računalo pošalje paket mikro upravljaču, u kojemu šalje podatke o razinama regulacije, ili šalje prazan paket mikro upravljaču. Ako je paket prazan, mikro upravljač ne uzima podatke, nego samo vraća stanje trenutne razine i razina regulacije. Ako osobno računalo pošalje paket sa podatcima o regulaciji, tada mikro upravljač uzima potrebne podatke, sprema ih u memoriju te vraća osobnom računalu paket sa podatcima trenutne razine i podatcima razina regulacije, za potvrdu uspješnog programiranja razina. Izgled paketa slanog od osobnog računala prikazan je na slici 5.4.

=XT	Gornja razina regulacije	Donja razina regulacije	Gornja razina alarma	Donja razina alarma	CR
Zaglavlje |	Podatkovni dio				|Kraj paketa

Slika 5.4 Izgled paketa poslanog od strane osobnog računala

RS232 način prijenosa radi na principu znakovnog terminala, tj. podatci koji se šalju, kodirani su ASCII kodnom tablicom i prikazani u obliku simbola. Da bi prijenos razina bio moguć preko RS232 porta, potrebno je numeričke vrijednosti pretvoriti u znakovne vrijednosti, sa tri znamenke, koje se tada šalju na RS232 port u sljedećem formatu:

„=XT$x_1x_2x_3y_1y_2y_3z_1z_2z_3q_1q_2q_3$<CR>“

gdje je „=XT“ zaglavlje koje označuje da je paket poslan od osobnog računala, x_n - gornja razina regulacije n-ta znamenka, y_n – donja razina regulacija n-ta znamenka, z_n – gornja razina alarma n-ta znamenka, q_n - donja razina alarma n-ta znamenka, <CR> – simbol kraja paketa.

Na sličan način mikro upravljač odgovara osobnom računalu i to u sljedećem obliku:

>XT	Gornja razina regulacije	Donja razina regulacije	Gornja razina alarma	Donja razina alarma	Stvarna razina	CR
Zaglavlje|		Podatkovni dio				|Kraj paketa

Slika 5.5 Izgled paketa poslanog od strane mikro upravljača

Format slanja podataka od mikro upravljača sličan je formatu paketa osobnog računala, sa dodatkom stvarne razine, koja se dobije u potprogramu konv_asci():

„>XT$x_1x_2x_3y_1y_2y_3z_1z_2z_3q_1q_2q_3w_1w_2w_3$<CR>“

gdje je „>XT“ zaglavlje koje označuje da je paket poslan od mikro upravljača, x_n - gornja razina regulacije n-ta znamenka, y_n – donja razina regulacija n-ta znamenka, z_n – gornja razina alarma n-ta znamenka, q_n - donja razina alarma n-ta znamenka, w_n – stvarna razina spremnika n-ta znamenka, <CR> – simbol kraja paketa.

Da bi komunikacija sa osobnim računalom putem RS232 sučelja bila moguća, prvo je potrebno inicijalizirati RS232 sklopovlje mikro upravljača. To radi potprogram uart_init(), koji ima zadaću omogućiti rad USART sklopovlja, i to na način da prema [2 str 166] omogući programske prekide i omogući prijem i predaju na RS232 port. To ostvarujemo na način da upišemo u registar UCSRB, (u AVRStudiu UCR) na mjesta bitova RXCIE, TXCIE, RXEN, TXEN logičke jedinice. Drugi dio konfiguracije USART-a je određivanje brzine slanja podataka, koja ovisi o UBBR registru, koji je zadužen za generiranje frekvencije.Za asinkroni standardni rad sklopa, vrijednost UBBR registra ovisno o brzini jednaka je:

$$UBBR = \frac{f_{OSC}}{16 \cdot BAUD} - 1 \qquad (5\text{-}1)$$

Gdje je fosc – frekvencija glavnog oscilatora mikro upravljača (6.144MHz), BAUD- brzina prijenosa u bitovima u sekundi.

Kao što smo vidjeli iz izgleda paketa, broj byte-ova paketa osobnog računala je 13, a broj byte-ova paketa slanog od mikro upravljača je 19, što daje maksimalni broj bitova za prenijeti preko RS232 152 + 1 stop bit. Pošto je to relativno velik niz, a cilj nam je što prije poslati i primiti podatke, koristit ćemo brzinu prijenosa od 19200 bitova u sekundi, uz jedan stop bit, bez provjere pariteta i byte-ove veličine 8 bita: 19200,N,8,1. Tada iz (5-1) dobivamo vrijednost UBBR registra: UBBR=19.

Glavni dio programa za primanje i slanje podataka sastoji se od dva potprograma, koji se pozivaju na programske prekide USART sklopovlja:

Potprogram SIGNAL(SIG_UART_RECV) poziva se kada je primljen znak sa serijskog RS232 porta. Primljeni znak se provjerava da li je jednak prvom znaku iz zaglavlja, ili je jednak znaku kraja paketa. Ako znak označava početak zaglavlja, tada se svi naredni znakovi spremaju u vektor, sve do znaka kraja paketa <CR> i do točno određenog broja okteta u zaglavlju (16). Tada se provjerava ispravnost zaglavlja i dali je podatkovno polje prazno, ili puno. Ako je zaglavlje ispravno, postavlja se zastavici Primio, koja označava da su podatci primljeni sa serijskog RS232 porta. Ako je podatkovno polje prazno, tada se razine regulacije i alarma ne

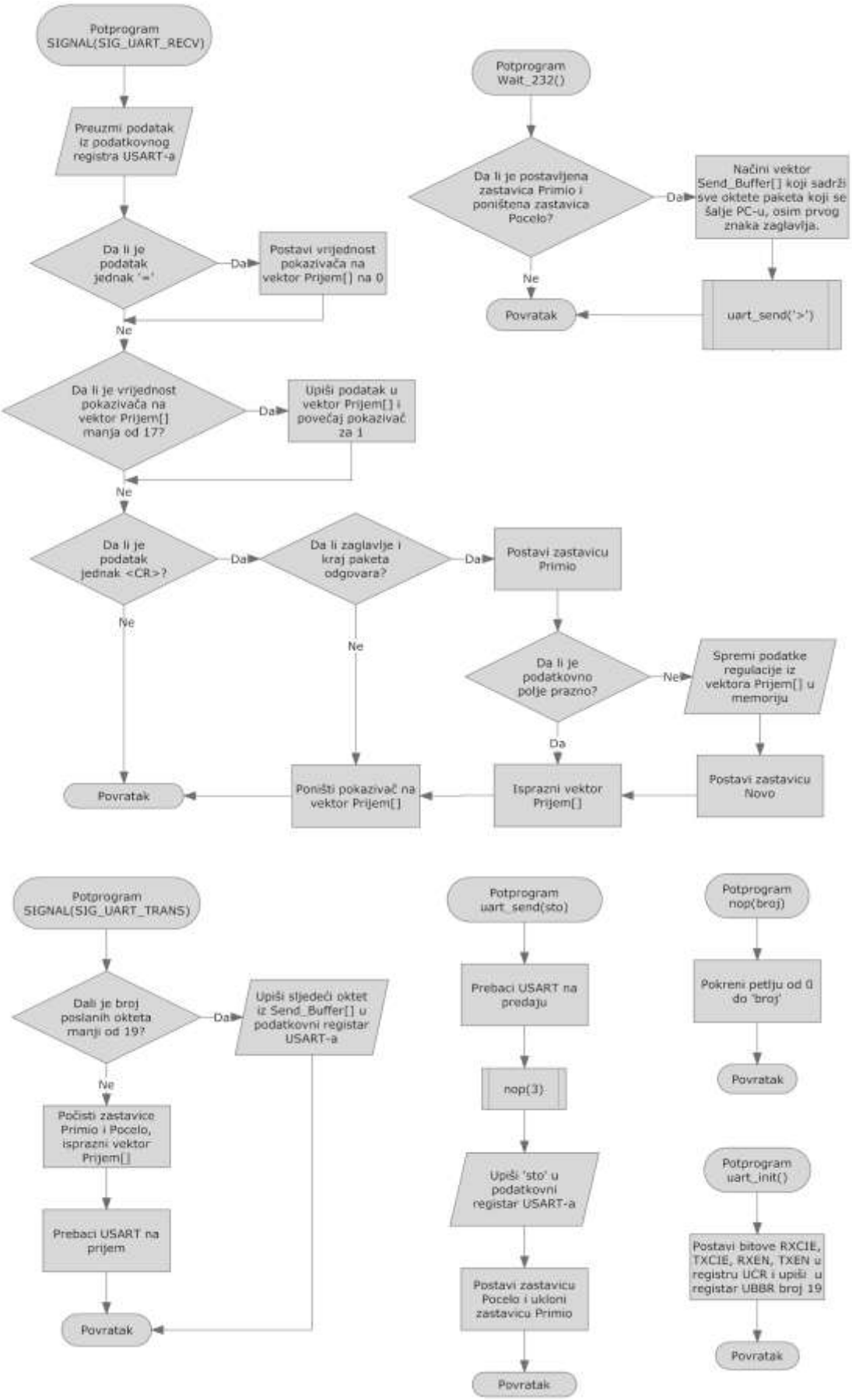
mijenjaju, u suprotnom, razine u mikro upravljaču se prepisuju sa razinama dobivenima sa serijskog porta i postavlja se zastavica Novo, koja označuje da su nastupile nove razine na snagu.

Potprogram SIGNAL(SIG_UART_TRANS) poziva se kada je USART spreman za slanje podatka. Pošto je RS232 serijska veza, slanje se mora vršiti oktet po oktet. Za početak slanja okteta koristi se potprogram uart_send(..), koji prebacuje USART na predaju, šalje prvi znak iz zaglavlja mikrokontolera („>“), te postavlja zastavicu Počelo, koja nam govori da je u tijeku slanje podataka na serijski port. Nakon poslanog prvog znaka, USART okida prekide nakon svakog poslanog znaka i poziva SIGNAL(SIG_UART_TRANS), koji svaki put kada se pozove, šalje na serijski port idući oktet iz vektora Send_Buffer[], koji predstavlja paket koji se šalje. Nakon svih poslanih okteta, zastavice Primio i Počelo se skidaju i USART se prebacuje na prijem.

Način razmjene podataka obuhvaća potprogram Wait_232(), koji se poziva periodički u radu, tj. nalazi se u glavnoj petlji. On pri pozivanju provjerava zastavice i to, da li je primljen paket od strane osobnog računala i ako je te u tom vremenu nije započelo slanje podataka osobnom računalu, potprogram Wait_232() oblikuje paket sa zaglavljem i krajem te ga sprema u vektor Send_Buffer[] i započinje slanje podataka osobnom računalu, na način, da poziva potprogram uart_send('>'), koji šalje prvi simbol zaglavlja. Nakon što je poslan prvi simbol, prekidni potprogram SIGNAL(SIG_UART_TRANS) nastavlja sa slanjem paketa osobnom računalu. Potprogram uart_send(..) se ponaša kao okidač, koji nakon pozivanja, omogućuje okidanje potprogram SIGNAL(SIG_UART_TRANS).

Od potprograma korištenih za slanje podataka na RS232 port, treba spomenuti potprogram nop(…), koji ima svrhu da stvori određeni broj NOP (No operation) naredbi u aritmetičko logičkoj jedinici mikro upravljača. Ova naredba nam služi kada USART prebacujemo sa prijema na predaju, jer pri upisivanju podatka na port mikro upravljača, treba proći određeni broj ciklusa takta, da bi se sklopovlje konfiguriralo, tako da početak prijenosa ne bude prije nego je port konfiguriran.

Svi potprogrami opisani u ovom poglavlju imaju određenu zadaću i u krupnom planu omogućuju prijenos podataka, od, i prema osobnom računalu, koristeći RS232 sučelje. Na slici 5.6 prikazani su dijagrami toka svih potprograma koji sudjeluju u RS232 komunikaciji.



Slika 5.6 Blok dijagram potprograma potrebnih za rad RS232 serijske veze

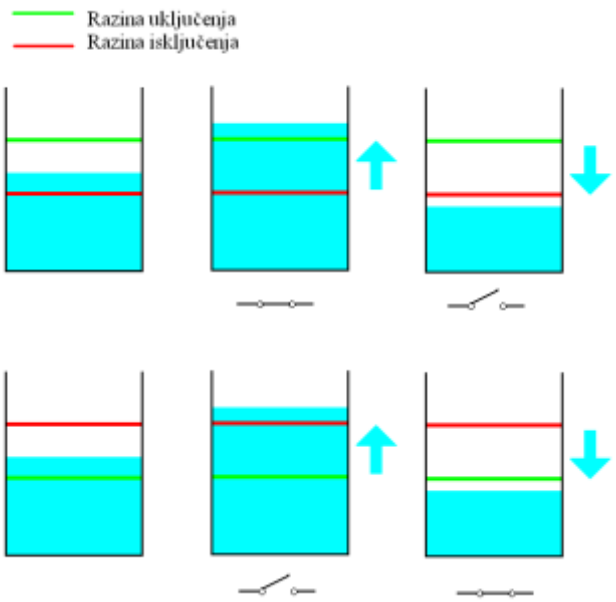
5.1.3 Regulacija razine i podešavanje kritične razine (alarmi)

Glavni dio regulacije razine implementiran je u potprogramu regulacije. Princip same regulacije vidjeli smo u prethodnom poglavlju, a ovdje ćemo vidjeti na koji način radi program koji vrši regulaciju i upravlja alarmima. Kao što smo vidjeli, ova regulacija se bazira na dvije razine, razina uključenja i razina isključenja izvršnog člana. Svaka razina se definira u postotcima kako kod regulacije, tako i kod kritične razine.

Princip rada alarma kritične razine je sljedeći: postoje dva vrsta alarma kritične razine. Prvi je gornja kritična razina, gdje se alarm aktivira kada razina pređe dopuštenu razinu, definiranu od korisnika. Drugi alarm je alarm donje kritične razine. On se aktivira kada razina padne ispod određene razine, definirane od korisnika. Alarm gornje kritične razine i alarm donje kritične razine ostaje uključeni sve dok se razina ne dovede u granice regulacije, između razine uključenja i razine isključenja. Tada se alarmi poništavaju, jer se smatra da se proces vratio u granice regulacije. Iz priloženog vidimo da nikada ne mogu biti oba alarma uključena, stoga su izlazi prema izvršnim članovima napravljeni na taj način, da su dva releja predviđena za alarme gornje i donje razine, a treći relej je predviđen za glavni izvršni član regulacije.

Glavni dio regulacije razine i kontrole alarma odvija se u potprogramu Regulacija(), no da bi se moglo upravljati relejima preko izlaznog Porta C, Port C se mora inicijalizirati za izlazni način rada, jer su svi portovi mikro upravljača dvosmjerni. Inicijalizacija porta odvija se u potprogramu portC_init().

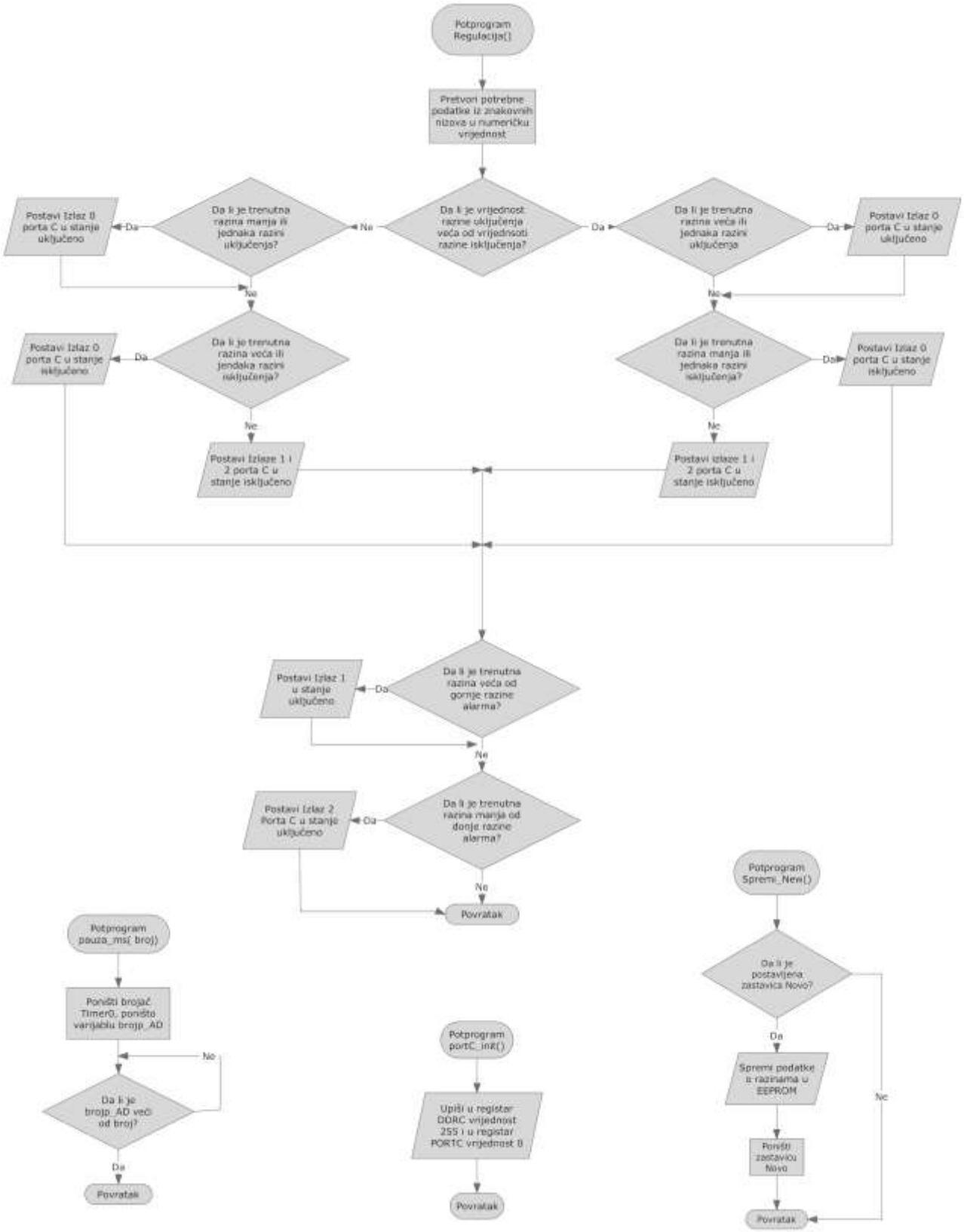
U potprogramu Regulacija(), glavnom dijelu regulacije i kontrole alarma odvija se uspoređivanje trenutne razine, sa razinama alarma i regulacije danim od korisnika, i na temelju usporedbe, upravlja se sa izlazima prema izvršnim članovima. Upravljanje sa regulacijom vrši se na način: ako je razina uključenja iznad razine isključenja, tada, ako stvarna razina pređe iznad razine uključenja, uključuje se izlaz prema izvršnim članovima, a ako stvarna razina padne ispod razine isključenja, tada se izlaz prema izvršnim članovima isključuje. Isto tako ako je razina isključenja iznad razine uključenja, tada, ako stvarna razina padne ispod razine uključenja, uključuje se izlaz prema izvršnim članovima, a ako se razina digne iznad razine isključenja, izlaz prema izvršnim članovima se isključuje. U ovom načinu regulacije, razina uključenja i isključenja se ponašaju kao okidači, koji mijenjaju stanje izvršnog člana, ovisno o stvarnoj razini. Ilustrirani prikaz rada regulatora prikazan je na slici 5.7.



Slika 5.7 Ilustrirani prikaz rada regulatora

Osim dijela regulacije, postoji dio programa koji se veže na regulaciju, ali ne izravno. Ovdje govorimo o dijelu programa čija je zadaća da pohrani podatke o razinama regulacije i razinama alarma, u stalnu memoriju mikro upravljača. Stalna memorija mikro upravljača je dio memorije koju nazivamo EEPROM, ili Električni izbrisivi programirljivi ROM, prema [2] jer ima svojstva permanentne memorije. Upisivanje i brisanje podataka postiže se tuneliranjem elektrona kroz suženi sloj izolatora, između lebdeće elektrode i odvoda, prema [2 str 393]. Kapacitet EEPROM-a ATmega8535 je 512Byta, što je više nego dovoljno za pohranjivanje podataka o razinama regulacije. Potprogram naziva Spremi_New(), ima zadatak da provjerava da li je došlo do promjene u regulacijskim razinama, i ako da, pohraniti nove podatke u EEPROM memoriju. Ne bi bilo dobro kada bi se periodički upisivali podatci u EEPROM, jer je vijek trajanja EEPROM memorije, prema [3] procijenjen na 100,000 pisanja/brisanja, što se čini puno ali ako je frekvencija pisanja velika brzo se memorija može uništiti. Pri pokretanju mikro upravljača se također čitaju podatci iz EEPROM memorije i pohranjuju u RAM memoriju u obliku varijable.

Na slici 5.8 možemo vidjeti zadnji dijagram toga ATmega8534-ovog koda. Na njemu su prikazani svi potprogrami korišteni za regulaciju, i potprogram naziva pauza_ms(…), koji ima svrhu da pri pokretanju osigura pauzu od pola sekunde, da bi se pravilno konfigurirala budilica (WatchDog timer). Spojimo li sve dijagrame toka koje smo postupno iznositi u tijeku ovog rada, dobivamo kompletni program mikro upravljača ATmega8535, koji zauzima 4.55KB FLASH memorije, te ga možemo usporediti sa listingom programa, prikazanom u privitku 5.1.



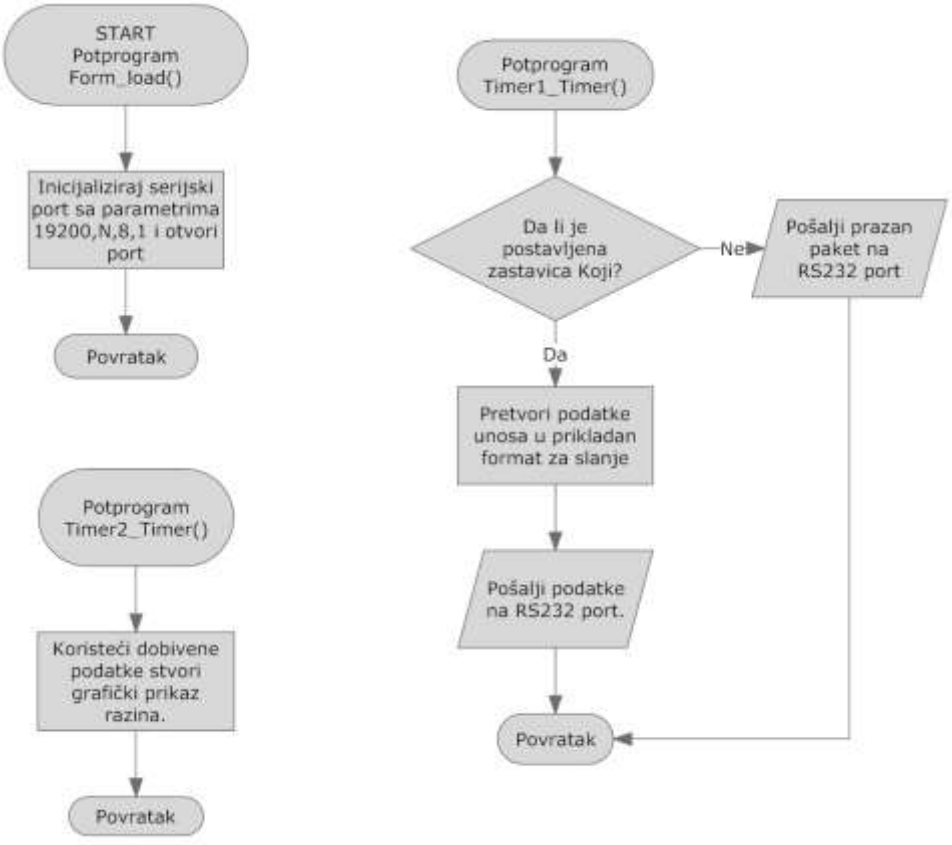
Slika 5.8 Blok prikaz potprograma potrebnih za regulaciju razine

5.2 Program za osobno računalo

Nakon detaljno opisanog programa mikro upravljača, jedini dio koji nije obrađen je upravljački program na osobnom računalu. Upravljački program ima svojstvo prikaza trenutne razine, prikaza razina regulacije i razina alarma dobivenih iz mikro upravljača i svojstvo programiranja novih razina alarma i razina regulacija sustava.

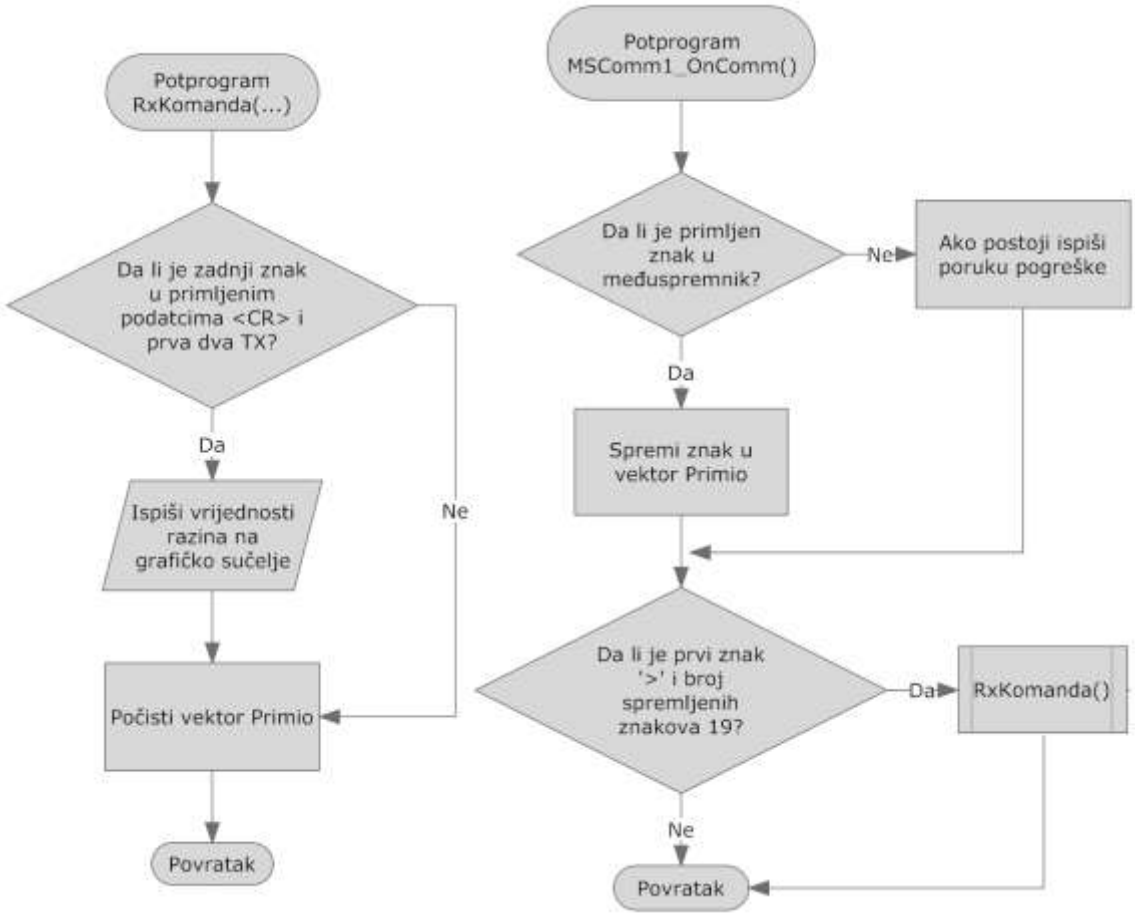
Program je napisan u programskom jeziku Visual Basic i radi na sljedeći način: Pri podizanju programa prvo se incijalizira serijski port i otvori veza, slika 5.9. Nakon što je veza uspostavljena pokreće se pozadinski brojač Timer1, koji ima zadaću da svaki dani vremenski interval pozove određeni potprogram (Timer1_timer()). U našem slučaju, potprogram koji se poziva ima zadaću slanja paketa, sa zaglavljem i podatcima, na serijski RS232 port.

Program periodički šalje paket svakih 1s, ali šalje prazan paket, bez podataka, jer paket bez podataka ne utječe na trenutne razine koje su upisane u mikro upravljač, nego samo govori mikro upravljaču da vrati paket sa upisanim razinama regulacije i sa trenutnom razinom. Na taj način, program periodički dobiva informacije o stanju trenutne razine i stanju regulacijskih i alarmnih razina, te ih grafički prikazuje za što je zaslužan potprogram Timer2, slika 5.9.



Slika 5.9 Potprogrami inicijalizacije i slanja podataka

Kada računalo pošalje prazni (ili puni) paket, on čeka na nadolazeće oktete od mikro upravljača. Na svaki primljeni oktet događa se prekid, ili događaj, koji okida potprogram MsComm1_OnComm(). U teoriji, isti je princip kao kod mikro upravljača. Program prikuplja sve oktete i kada završi primanje, provjerava zaglavlje i kraj, te ako je sve u redu poziva potprogram RxKomanda(…), koji nadalje dodatno provjerava zaglavlje pristiglog paketa i ako je zaglavlje ispravno prikazuje podatne na grafičko sučelje. Dodatna stvar koja nema kod mikro upravljačevog programa je ako se zamijeti neka greška na nadolazećem nizu okteta, trenutni niz se odbacuje i čeka se sljedeći. Dijagram toka prikazan je na slici 5.10.

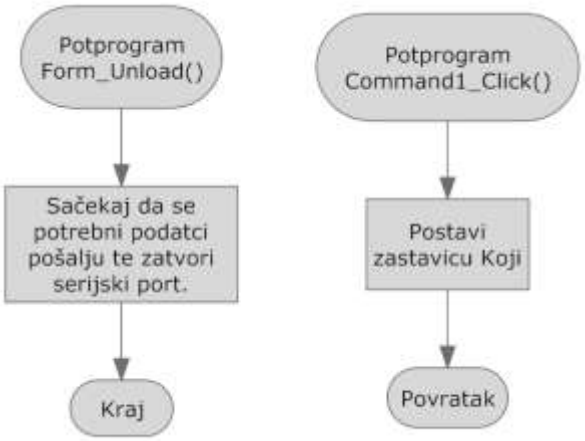


Slika 5.10 Potprogrami serijske komunikacije i obrade podataka

Ako poželimo promijeniti neku od programiranih razina, potrebno je upisati željene promjene u odgovarajuća, polja i pritisnuti „Pošalji razine“. Nakon što je komanda „Pošalji razine“ (Command1_Click()) pritisnuta, ona postavlja zastavicu Koji, koja utječe na ponašanje programa u Timer1 potprogramu, slika 5.11. Kada se detektira Koji zastavica, tada potprogram više ne šalje prazan paket, nego pretvara vrijednosti iz polja za unos u potrebne podatke za slanje na serijski port, na isti način na koji radi potprogram asci_konv() u mikro upravljaču. Nakon

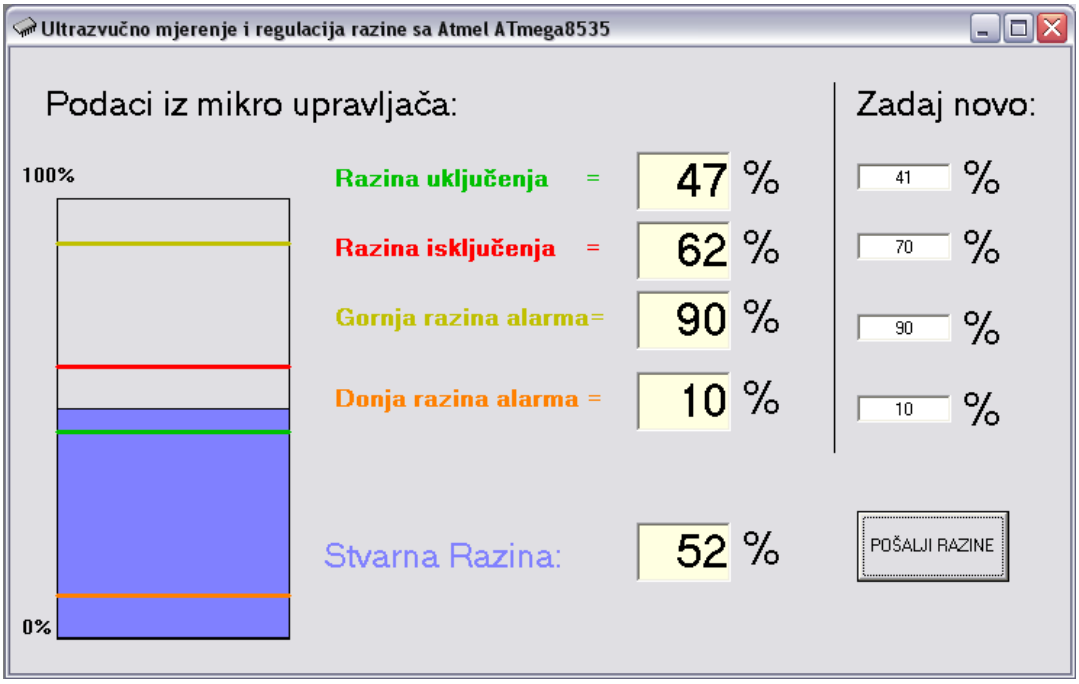
slanja novih razina, zastavica Koji se skida i program dalje šalje prazne pakete, isto kao i prije slanja novih razina.

Nakon završetka rada sa programom, prilikom zatvaranja, poziva se potprogram Form_Unload(), koji nakon završetka slanja trenutnog paketa zatvara serijsku vezu i gasi sučelje, slika 5.11.



Slika 5.11 Potprogrami koji se pokreću pri izlazu iz programa i pri pozivanju komande slanja razina.

Nakon opisanog rada programa za osobno računalo, potrebno je opisati i prikazati grafičko sučelje programa. Izgled prikazan je na slici 5.12.



Slika 5.12 Izgled programa za osobno računalo

Na grafičkom sučelju prikazanom na slici 5.12 vidimo prikazane vrijednosti razina dobivenih iz mikro upravljača, sa stvarnom razinom, dobivenom od ultrazvučne sonde. Lijevo od razina vidimo grafički prikaz samih razina i to na način, da su određenim linijama u bojama izražene određene razine, a stvarna razina prikazana je u obliku volumena koji pokriva spremnik (plava boja). Razine na grafičkom prikazu odgovaraju razinama dobivenim iz mikro upravljača.

Drugi dio programa je dio za zadavanje novih razina i slanje u mikro upravljač. Razine se zadaju na način da se upišu željene razine u predviđena mjesta i zatim pritisne komanda „Pošalji razine“, koja šalje upisane razine mikro upravljaču, enkapsulirane u paket, opisano u poglavlju 5.3.

Na ovaj način se mjeri i regulira razina, koristeći Atmel Atmega8535, i programiraju željene razine prema potrebi.

Sa završetkom programskog dijela, završavamo sa teorijskom obradom sustava ultrazvučnog mjerenja i regulacije sa Atmel Atmega8535. Sada su postavljenje sve teorijske osnove za praktičnu izradu sklopa i testiranje, no praktični dio u ovom radu neće biti obrađen zbog kompliciranosti izrade, nabave dijelova i testiranja ovog uređaja.

6. ZAKLJUČAK

Ultrazvučno mjerenje i regulacija imaju veliku ulogu u današnjim sustavima automatskog upravljanja i u sustavima mjerenja, gdje nije moguće mjerenje razine na konvencionalan način. Sa pojmom ultrazvučnog mjerenja razine upoznali smo se u početku, te je prikazan način određivanja udaljenosti koristeći ultrazvuk, što je osnova ultrazvučnog mjerenja. Vidjeli smo prednosti ultrazvuka i način primanja i odašiljanja, što je bilo osnova za daljnje razmatranje. Nakon teorijskog dijela zvuka, prikazan je praktični način mjerenja; ultrazvučna sonda. Spomenut je princip rada ultrazvučne sonde u području mjerenja, davanja razine i prijenosu podataka od sonde, do mikro upravljača, strujnom vezom. Pošto je ultrazvučna sonda praktični proizvod, baziran na principu ultrazvučnog mjerenja, detaljni podatci proizvođača i specifikacije sonde nisu dani, zbog primarno teorijske osnove ovog rada.

Pregled regulacijskog sustava započet je sa upoznavanjem sa mikro upravljačem Atmel ATmega8535, gdje objašnjen princip pojedinih sklopovlja integriranih u sam mikro upravljač, koji su neophodni za rad ovog sustava i to sklopovlje analogno-digitalne pretvorbe, serijske komunikacije i ostalo pomoćno sklopovlje. U daljnjem razmatranju, opisan je način regulacije sa razinama i usporedba sa ostalim metodama, te je objašnjen način prikaza i upravljanja pomoću osobnog računala. Na kraju je prikazan blok dijagram cijelog sustava, radi lakšeg snalaženja.

Idući korak, koji je slijedio, je projektiranje sklopovlja sustava iz prije danih teorijskih osnova. Projektiranje se izvodilo u pet dijelova, redom, od serijske komunikacije, spoja sa sondom, sučelja prema izvršnim članovima, središnjeg dijela, te naposljetku napajanja sustava. Pokazano je da su sklopovi i dijelovi odabrani prema zahtjevima koji su dani na sustav, te prema potrebnoj funkcionalnosti. Treba napomenuti da su neki sklopovi uzeti automatizmom, bez dodatnih proračuna, kao npr. napajanje, jer nije bilo potrebno praviti posebni projekt napajanja za standardne CMOS i TTL sklopove, kada postoje gotova rješenja. Neke veličine pasivnih elemenata, i njihovo spajanje na mikro upravljač uzeti su iz tvorničkih podataka samog mikro upravljača, te iz praktičnih iskustava konstruktora. Nakon razrade i proračuna, prikazan je cjelokupni shematski prikaz. Moguće je od shematskog prikaza razviti tiskanu pločicu za izradu..

Nakon elektroničkog projektiranja dolazi programski dio. Programski dio treba obuhvaćati programske dijelove mikro upravljača kao i program za osobno računalo uz detaljno objašnjene algoritme pomoću dijagrama toka.

Prvi dio je program za mikro upravljač koji se sastoji od grubo tri dijela: AD pretvorba, serijska komunikacija i regulacija. Svaki dio detaljno je opisan u poglavljima, i to na način da je

opisan dijagram toka, koji je izravno povezan sa listingom programa priloženim u prilogu. Drugi dio je program za osobno računalo, koji je načinjen u programskom jeziku Visual Basic, i pruža grafičko sučelje prema mikro upravljaču i sustavu mjerenja i regulacije. Program je objašnjen na isti način, pomoću dijagrama tokova i izravno povezan sa listingom u prilogu. Treba napomenuti da su oba programa simulirana, u cilju provjeravanja njihovog rada, i to program za mikro upravljač je simuliran u programu AvrStudio, gdje je na simuliranu analognu pobudu promatran rad. Program za računalo je ručno simuliran na principu debug-a. Pošto praktična izvedba nije predviđena, nije moguće kvalitetno ispitati rad ovog sustava i moguće greške u programu.

Ovaj rad, zamišljen kao teorijska osnova, omogućuje u potpunosti izradu ovog cijelog sustava u praksi, te primjenu na regulaciju razine tekućina, krutih tvari, pokretnih traka i mnogih drugih mjesta regulacije. Kao uređaj mjerenja i alarmiranja može se primijeniti i na sustave koji ne zahtijevaju regulaciju, nego samo ograničenja razina i potrebe za alarmiranje kritičnih razina. Iako ovaj sustav ne pruža kontinuiranu regulaciju sa određenim odzivom, nego regulaciju unutar određenog intervala, on se svejedno može primjenjivati u mnogim sustavima koji ne zahtijevaju veliku preciznost i točno definiranu razinu, te je time pogodan za manje profesionalne i amaterske izvedbe, te za sustave gdje kontinuirano upravljanje izvršnim članovima nije moguće!

Literatura

[1] T. Jelaković, Zvuk · Sluh · Arhitektonska akustika, Školska knjiga Zagreb, 1978

[2] U. Periško, Digitalna elektronika, logičko i elektroničko projektiranje, Školska knjiga Zagreb, 1996

[3] Specifikacije mikro upravljača Atmel ATmega8535, www.atmel.com/dyn/resources/prod_documents/doc2502.pdf

[4] N. Perić, Automatsko upravljanjem, FER Zagreb, 2004

[5] MAXIM, Choosing the Right RS-232 Transceiver, Maxim Integrated Products, 2003

[6] Specifikacije integriranog kruga MAX233

[7] Specifikacije integriranog kruga UMN2803

[8] Specifikacije integriranog kruga 78L05

Sažetak

U ovom radu opisan je način regulacije i mjerenja razine (fluida, krutina itd.) koristeći ultrazvučno mjerenje udaljenosti, mikro upravljač i osobno računalo. Opisan je princip ultrazvučnog mjerenja koristeći ultrazvučnu sondu, te način rada same sonde. Prikazani su dijelovi sustava, i princip regulacije razine koristeći mikro upravljač Atmel Atmega8535. Detaljno je opisano projektiranje električnog sustava, serijske veze sa osobnim računalom, sučelja prema izvršnim članovima i sučelje prema ultrazvučnoj sondi. Koristeći dijagrame toka, opisan je program za mikro upravljač, te program za osobno računalo, koji međusobno komuniciraju preko serijske RS232 veze, koja služi za prikaz razina na osobnom računalu i zadavanje željenih razina u mikro upravljač.

Abstract

In this paper, a method of regulation and level measurement is described, using an ultrasonic distance measuring, micro controller and personal computer (PC). An ultrasonic measuring principle is described using an ultrasonic probe, and probe's theory of operation is described as well. System segments are shown and the principle of regulation using a micro controller Atmel AtMega8535. A detail description of electronic system is presented such as a serial link with PC, interface towards the actuators and the interface toward the ultrasonic probe. Using the flow diagrams, software for microcontroller is described and the software for the PC too, which communicate in-between over the RS232 serial link. Serial link is used for the display of level data on PC and for the programming of regulation levels into the microcontroller.

Životopis

Osobni podatci:

Goran Horvat
Vukovarska 2
34000 Požega
Telefon: +38534273123
Mobitel: +385915659877
E-Mail: goran.horvat@etfos.hr

Datum rođenja: 23.veljače 1987.god.
Mjesto rođenja: Požega

Obrazovanje:
1993 – 2001 Osnovna škola Julija Kempfa, Požega
2001 - 2005 Tehnička škola Požega, smjer Elektrotehničar, maturalni rad: „Dvostupanjsko unipolarno J-FET pojačalo u kaskadnom spoju“
2005 - … Elektrotehnički fakultet, Osijek, smjer: Elektrotehnika, komunikacije

Natjecanja:
2001 – 2002 ACSL(American Computer Science League) Međunarodno natjecanje, Classroom division, 1. mjesto pojedinačno, 3. ekipno
2002 Osnove elektrotehnike, Državno natjecanje, 4. mjesto
2002 – 2003 ACSL međunarodno natjecanje, Intermediate division, 1. mjesto pojedinačno
2003 – 2004 ACSL međunarodno natjecanje, završno natjecanje u SAD, Chicago, Intermediate division, 1. mjesto pojedinačno
2002 – 2004 Županijsko natjecanje „Mladih tehničara“, 1. mjesto, 3 godine za redom

Dodatna znanja:
Rad na PC-u:
-C++, Visual Basic, Basic, BASCOM, Ms OFFICE
Strani jezici:
-Engleski
Vozačka dozvola:
-kategorija B

Goran Horvat

_____

Prilog 5.1 : Listing programa mikro upravljača ATMega8535

```
/*
        Program za ultrazvučno mjerenje i regulaciju sa Atmel ATmega8535
        RS232  -  19200,N,8,1 (veza s PC-om)
                1. PC posalje:    "=XTx1x2x3y1y2y3z1z2z3q1q2q3<CR>"
                2. uC vrati:     ">XTx1x2x3y1y2y3z1z2z3q1q2q3w1w2w3<CR>"
                                        (x=1.nivo, y=2.nivo, z=3.nivo, q=4.nivo, w=stvarni nivo)
                Ako PC posalje "=XT000000000<CR>" uC ne uzima podatke nego samo salje stanje
*/
#include <avr/eeprom.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/signal.h>
#include <avr/pgmspace.h>
#include <avr/wdt.h>    // watchdog
#include <stdlib.h>
#include <math.h>
#define DEBUG 1
#define CPU_F           6144000L
#define UART_BAUD_RATE     19200L
#define UART_BAUD_SELECT (CPU_F/(UART_BAUD_RATE*16))-1
#define ms1 255-((CPU_F/1024)/1000)
#define Time_Out 2000
#define Tx_En PD2   // PD2 pin 1=Tx, 0=Rx (nema ECHO)
#define BEL 7
#define HT 9
#define LF 10
#define CR 13
#define ESC 27
#define Nula 186.2
#define Konst 7.45
typedef unsigned char  u08;
typedef         char  s08;
typedef unsigned short u16;
typedef         short s16;
/* globals variable */
static volatile u08 Pocelo=0;
static volatile u08 Primio=0;
static volatile u08 uart_sent=0;
static volatile u08 uart_rec=0;
static volatile u08 uart_sen=0;
static volatile u08 smir_count=0;
static volatile u08 Time_Tick_AD=0;
static volatile u16 brojp_AD=0;
static volatile u08 brojT=0;
static volatile u08 Novo=0;
static volatile int AD_Temp;                    // A/D 1 nivo (prosjek 10 mjerenja)
static volatile int AD_Temp_x[10];              // A/D 10 mjerenja
static volatile u08 aski[3];                    // 3 broja sa 3 znamenke
static volatile u08 askj[5];                    // za konveziju broja
static volatile u08 ask_a1[4];          // zadana razina uključenja
static volatile u08 ask_a2[4];          // zadana razina isključenja
static volatile u08 ask_a3[4];          // zadana gornja razina alarma
static volatile u08 ask_a4[4];          // zadana donja razina alarma
static volatile u08 Prijem[18];         // Prijemni bufer od PC-a
static volatile u08 Send_Bufer[18];     // Predajni bufer

/* ------------------------------------------------------------*/
SIGNAL(SIG_OVERFLOW0)
// signal handler for tcnt0 overflow interrupt
{
   brojp_AD++;
   outp(brojT, TCNT0);          // reset counter to get this interrupt again
}

SIGNAL(SIG_UART_TRANS)
// signal handler for uart TxD ready interrupt
{
        uart_sent = 0;
   uart_sen++;
   if (uart_sen < 19){
        outp(Send_Bufer[uart_sen], UDR);
```

```
   }
   else{
      uart_sen = 0;
      Primio = 0;
      Pocelo = 0;
      Prijem[0] = 0;      // =
      Prijem[1] = 0;      // X
      Prijem[2] = 0;      // T
      Prijem[3] = 0;      // 1.1
      Prijem[4] = 0;      // 1.2
      Prijem[5] = 0;      // 1.3
      Prijem[6] = 0;      // 2.1
      Prijem[7] = 0;      // 2.2
      Prijem[8] = 0;      // 2.3
      Prijem[9] = 0;      // 3.1
      Prijem[10] = 0;     // 3.2
      Prijem[11] = 0;     // 3.3
      Prijem[12] = 0;     // 4.1
                  Prijem[13] = 0;     // 4.2
                  Prijem[14] = 0;     // 4.3
      Prijem[15] = 0;     // CR
      cbi(PORTD,Tx_En);   // prebaci na prijem
   }
}

SIGNAL(SIG_UART_RECV)
// signal handler for receive complete interrupt
{
   u08 i, j = 0;
   i = inp(UDR);
   if (i == '=') uart_rec = 0;
   if (uart_rec < 17) Prijem[uart_rec++] = i;
   if (i == CR){
         if (Prijem[0] == '='){
                  if (Prijem[1] == 'X'){
                           if (Prijem[2] == 'T'){
                                    if (Prijem[15] == CR){
                                             Primio = 1;
                                             uart_sen = 0;
                                             for (i=0;i<12;i++)
                                                      j = j + (Prijem[3+i]-0x30);
                                             if (j != 0){
                                                      ask_a1[0] = Prijem[3];
                                                      ask_a1[1] = Prijem[4];          // 1.nivo
                                                      ask_a1[2] = Prijem[5];
                                                      ask_a2[0] = Prijem[6];
                                                      ask_a2[1] = Prijem[7];          // 2.nivo
                                                      ask_a2[2] = Prijem[8];
                                                      ask_a3[0] = Prijem[9];
                                                      ask_a3[1] = Prijem[10];         // 3.nivo
                                                      ask_a3[2] = Prijem[11];
                                                      ask_a4[0] = Prijem[12];
                                                      ask_a4[1] = Prijem[13];         // 4.nivo
                                                      ask_a4[2] = Prijem[14];
                                                      ask_a1[3] = 0;
                                                      ask_a2[3] = 0;
                                                      ask_a3[3] = 0;
                                                      ask_a4[3] = 0;
                                                      Novo = 1;
                                             }
                                             Prijem[0] = 0;      // =
                                             Prijem[1] = 0;      // X
                                             Prijem[2] = 0;      // T
                                             Prijem[3] = 0;      // 1.1
                                             Prijem[4] = 0;      // 1.2
                                             Prijem[5] = 0;      // 1.3
                                             Prijem[6] = 0;      // 2.1
                                             Prijem[7] = 0;      // 2.2
                                             Prijem[8] = 0;      // 2.3
                                             Prijem[9] = 0;      // 3.1
                                             Prijem[10] = 0;     // 3.2
                                             Prijem[11] = 0;     // 3.3
                                             Prijem[12] = 0;     // 4.1
                                             Prijem[13] = 0;     // 4.2
                                             Prijem[14] = 0;     // 4.3
                                             Prijem[15] = 0;     // CR
```

```
                                                }
                                            }
                                        }
                                    }
                                    uart_rec = 0;
    }
}
void pauza_ms(u16 broj)
{
    cli();                                                  // disable interrupts
    outp((brojT=ms1), TCNT0);   // reset counter
            brojp_AD = 0;
    sei();                              // enable interrupts
            while(brojp_AD < broj);
            brojp_AD = 0;
}
void adc_init(void)
// initialize uart
{
    // ADC enable, CLK/64
    outp((1<<ADEN)|(1<<ADPS2)|(1<<ADPS1),ADCSR);
}
void uart_init(void)
// initialize uart
{
    // enable RxD/TxD and ints
    outp((1<<RXCIE)|(1<<TXCIE)|(1<<RXEN)|(1<<TXEN),UCR);
    // set baud rate
    outp(UART_BAUD_SELECT, UBRR);
}
void timer0_init(void)
// initialize timer 0
{
    outp((1<<TOIE0), TIMSK);    // enable TCNT0 overflow
    outp((brojT=ms1), TCNT0);   // reset TCNT0
    outp(5, TCCR0);                     // count with cpu clock/1024
}
void portC_init(void)
{
    outp(0xFF,DDRC);
    outp(0x00,PORTC);
}
void nop(u08 broj)
{
  u08 i;
  for(i=0;i<broj;i++);
}
void AD_Konv(void)
{
    sbi(ADCSR,ADSC);                                    // start konverzije
    while((inp(ADCSR) & (1<<ADIF)) == 0); // sacekaj kraj konverzije
    sbi(ADCSR,ADIF);                                    // obrisi fleg kraja konverzije
    cbi(ADCSR,ADSC);                                    // obrisi fleg kraja konverzije
}
void A_D(void)
{
    u08 AD_H, AD_L;
    u16 Temp_u16;
    int AD_Temp_tmp;
    float Temper;
    if ((Time_Tick_AD == 1) && (smir_count < 10)){
                AD_Konv();                                                  // pokreni konverziju i sacekaj kraj
                AD_L = inp(ADCL);       // D1-D8
                AD_H = inp(ADCH);               // D9-D10
                Temp_u16 = (AD_H * 256) + AD_L;
                if (Temp_u16 < Nula) Temp_u16 = Nula;
                Temper = (float)(Temp_u16);
                Temper = (Temper - Nula) / Konst;
                AD_Temp_tmp = (int)Temper;
                AD_Temp_x[smir_count] = AD_Temp_tmp;
                smir_count++;
    }
}
void konv_asci(void)
{
  int i,j;
```

```
    j=0;
    while (askj[j] != 0) j++;
    j=3-j;

    for(i=0;i<j;i++){
            askj[2]=askj[1];
            askj[1]=askj[0];
            askj[0]=' ';
    }
    aski[2]=askj[2];
    aski[1]=askj[1];
    aski[0]=askj[0];
}
void AD_smiri(void)
{
    u08 i;
    u16 Temper;
            if (smir_count == 10){
                    smir_count = 0;
                    Temper = 0;
                    for(i=0;i<10;i++){
                            Temper = Temper + AD_Temp_x[i];
                    }
                    Temper = Temper / 10;
                    AD_Temp = Temper;
                    itoa(AD_Temp,askj,10);
                    konv_asci();

            }
}
void Time_ms_AD(u16 broj)
{
    Time_Tick_AD = 0;
    if (brojp_AD > broj){
            Time_Tick_AD = 1;
            brojp_AD = 0;
    }
}
void uart_send(u08 sto)
{
    if (uart_sent == 0){
        uart_sent = 1;
            sbi(PORTD,Tx_En);  // prebaci na predaju
            nop(3);
            outp(sto, UDR);                     // posalji zadani bajt
            Primio = 0;
            Pocelo = 1;
    }
}
void Wait_232(void)
{
    if ((Primio==1) && (Pocelo==0)){
                    Send_Bufer[1] = 'X';
                    Send_Bufer[2] = 'T';
                    Send_Bufer[3] = ask_a1[0];     // 1.nivo
                    Send_Bufer[4] = ask_a1[1];     //
                    Send_Bufer[5] = ask_a1[2];     //
                    Send_Bufer[6] = ask_a2[0];     // 2.nivo
                    Send_Bufer[7] = ask_a2[1];     //
                    Send_Bufer[8] = ask_a2[2];     //
                    Send_Bufer[9] = ask_a3[0];     // 3.nivo
                    Send_Bufer[10] = ask_a3[1];    //
                    Send_Bufer[11] = ask_a3[2];    //
                    Send_Bufer[12] = ask_a4[0];     // 4.nivo
                    Send_Bufer[13] = ask_a4[1];    //
                    Send_Bufer[14] = ask_a4[2];    //
                    Send_Bufer[15] = aski[0];      // stvarni nivo
                    Send_Bufer[16] = aski[1];      //
                    Send_Bufer[17] = aski[2];      //
                    Send_Bufer[18] = CR;
                    uart_send('>');
    }
}
void Spremi_New(void)
{
            if (Novo == 1){
```

```
                        eeprom_write_block(ask_a1, 1, 12);          // zapiši nove podatke u EEPROM
                        eeprom_write_block(ask_a2, 20, 12);
                        eeprom_write_block(ask_a3, 40, 12);
                        eeprom_write_block(ask_a4, 60, 12);
                        Novo = 0;
                }
}
void Regulacija(void)
{
int ON_r, OFF_r, ga, da;
ON_r=atoi(ask_a1);// pretvori podtake iz znakova u broj
OFF_r=atoi(ask_a2);
ga=atoi(ask_a3);
da=atoi(ask_a4);
if (ON_r > OFF_r){   // usporedi parametre potrebne za regulaciju
                if (AD_Temp >= ON_r) sbi(PORTC,PC0);
                if (AD_Temp <= OFF_r) cbi (PORTC, PC0);
                if ((AD_Temp <= ON_r) && (AD_Temp >= OFF_r)){
                        cbi(PORTC, PC1);
                        cbi(PORTC, PC2);
                        }
                }
                else{
                if (AD_Temp <= ON_r) sbi(PORTC,PC0);
                if (AD_Temp >= OFF_r) cbi (PORTC, PC0);
                if ((AD_Temp >= ON_r) && (AD_Temp <= OFF_r)){
                        cbi(PORTC, PC1);
                        cbi(PORTC, PC2);
                }
                }
if (AD_Temp > ga) sbi (PORTC, PC1);
if (AD_Temp < da) sbi (PORTC, PC2);
}
/*  ************************************************************** */
int main(void)
{
                portC_init();                   // PORTB postavi u radno stanje
                uart_init();            // init UART
                timer0_init();                  // init TIMER 0
                adc_init();                     // init ADC convertor
                sei();                                  // enable interrupts
                wdt_disable();                  // onemoguci watchdog
                pauza_ms(500);                  // sacekaj 500ms
                wdt_enable(7);                  // 2 sekunde za watchdog
                eeprom_read_block(ask_a1, 1, 12);           // ocitaj zadane vrijednosti iz eeprom-a
                eeprom_read_block(ask_a2, 20, 12)
                eeprom_read_block(ask_a3, 40, 12)
                eeprom_read_block(ask_a4, 60, 12)
                for(;;){                                // loop forever
                        wdt_reset();                    // reset watchdog
                        A_D();                                  // A/D konverzija
                        AD_smiri();                             // Umiri A/D konverziju
                        Time_ms_AD(100);  // Vremenska baza za AD
                        Wait_232();                             // Komunikacija s PC-om
                        Spremi_New();           // spremi nove vrijednosti u EEPREOM
                        Regulacija();
                }
}
```

Prilog 5.2 : Listing programa za osobno računalo

```
Dim Novo As Byte
Dim Koji As Byte
Dim Brojac As Integer
Dim Primio As String

Private Sub Command1_Click()
   Koji = 1
End Sub

Private Sub Form_Load()
   Novo = 0
   Koji = 0
   Brojac = 0
   CommPort = "1"            'Com Port
   Settings = "19200,n,8,1"   '19200,n,8,1
```

```
    Handshaking = "2"          'RTS/CTS
    Echo = False               'Echo
    MSComm1.CommPort = Val(CommPort)
    MSComm1.Settings = Settings
    MSComm1.Handshaking = Val(Handshaking)
    On Error Resume Next
    MSComm1.PortOpen = True
    Exit Sub
End Sub

Private Sub Form_Unload(Cancel As Integer)
    Dim Counter As Long
    Dim NumFile As Integer
    Dim i As Integer, j As Integer
    If MSComm1.PortOpen Then
      ' Wait 2 seconds for data to be transmitted.
      Counter = Timer + 2
      Do While MSComm1.OutBufferCount
        Ret = DoEvents()
        If Timer > Counter Then
          Select Case MsgBox("Data cannot be sent", 34)
            ' Cancel.
            Case 3
              Cancel = True
              Exit Sub
            ' Retry.
            Case 4
              Counter = Timer + 2
            ' Ignore.
            Case 5
              Exit Do
          End Select
        End If
      Loop
      MSComm1.PortOpen = 0
    End If

End Sub

Private Sub MSComm1_OnComm()
    Dim EVMsg$
    Dim ERMsg$

     ' Branch according to the CommEvent property.
    Select Case MSComm1.CommEvent
       ' Event messages.
       Case comEvReceive
          Primio = Primio & StrConv(MSComm1.Input, vbUnicode)
          Text1 = Brojac & ".  " & Primio
          If Left(Primio, 1) <> ">" Then Primio = ""
          If Left(Primio, 1) = ">" And Len(Primio) >= 19 Then Call RxKomanda(Primio)

       Case comEvCTS
          EVMsg$ = "Change in CTS Detected"
       Case comEvDSR
          EVMsg$ = "Change in DSR Detected"
       Case comEvCD
          EVMsg$ = "Change in CD Detected"
       Case comEvRing
          EVMsg$ = "The Phone is Ringing"
       Case comEvEOF
          EVMsg$ = "End of File Detected"
       ' Error messages.
       Case comBreak
          ERMsg$ = "Break Received"
       Case comCDTO
          ERMsg$ = "Carrier Detect Timeout"
       Case comCTSTO
          ERMsg$ = "CTS Timeout"
       Case comDCB
          ERMsg$ = "Error retrieving DCB"
       Case comDSRTO
          ERMsg$ = "DSR Timeout"
       Case comFrame
          ERMsg$ = "Framing Error"
       Case comOverrun
```

```
            ERMsg$ = "Overrun Error"
        Case comRxOver
            ERMsg$ = "Receive Buffer Overflow"
        Case comRxParity
            ERMsg$ = "Parity Error"
        Case comTxFull
            ERMsg$ = "Transmit Buffer Full"
        Case Else
            ERMsg$ = "Unknown error or event"
    End Select
    If Len(EVMsg$) Then
    ElseIf Len(ERMsg$) Then
        Primio = ""
        End If
End Sub

Private Sub RxKomanda(MojBuf As String)
    Brojac = Brojac + 1
    If (Val(Asc(Right(MojBuf, 1))) = 13) And (Mid(MojBuf, 2, 1) = "X") And (Mid(MojBuf, 3, 1) = "T") Then
          Label1 = Mid(MojBuf, 4, 3)
          Label2 = Mid(MojBuf, 7, 3)
          Label3 = Mid(MojBuf, 10, 3)
          Label13 = Mid(MojBuf, 13, 3)
          Label4 = Mid(MojBuf, 16, 3)
    End If
    Primio = ""
End Sub
Private Sub Timer1_Timer()
    Dim l As Integer
    Dim arry(4) As String
    Dim x As Byte
    Dim y As Byte
    Dim tex As String
    If Koji = 1 Then
        arry(0) = Text2
        arry(1) = Text3
        arry(2) = Text4
        arry(3) = Text5
        For x = 0 To 3
        arry(x) = Trim((Str(Val(arry(x)))))
        l = Len(arry(x))
        l = 3 - l
            For y = 1 To l
            arry(x) = " " + arry(x)
            Next y
        Next x
        tex = ""
        For x = 0 To 3
        tex = tex + arry(x)
        Next x
        MSComm1.Output = "=XT" + tex + Chr(13)
        Koji = 0
    Else
        MSComm1.Output = "=XT000000000" + Chr(13)    'daj nivoe
    End If

End Sub

Private Sub Timer2_Timer()
Line2.Y1 = 6123 - Val(Label1.Caption) * 4560 / 100
Line2.Y2 = Line2.Y1
Line4.Y1 = 6123 - Val(Label2.Caption) * 4560 / 100
Line4.Y2 = Line4.Y1
Line3.Y1 = 6123 - Val(Label3.Caption) * 4560 / 100
Line3.Y2 = Line3.Y1
Line5.Y1 = 6123 - Val(Label13.Caption) * 4560 / 100
Line5.Y2 = Line5.Y1
Shape2.Height = (Val(Label4.Caption) / 100) * 4575
Shape2.Top = 6120 - Shape2.Height
End Sub
```