UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Matija Mazalin

Avtopilot plovila na osnovi GPS

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Ljubljana, maj 2012

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Matija Mazalin

Avtopilot za plovila na osnovi GPS

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU
Mentor: prof. dr. Dušan Kodek

Ljubljana, maj 2012

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01823/2012
Datum: 02.04.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MATIJA MAZALIN**

Naslov: **AVTOPILOT ZA PLOVILA NA OSNOVI GPS**
GPS BASED BOAT AUTOPILOT

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Avtomatski krmilni sistem (avtopilot) za plovila je naprava, ki je sposobna samostojno krmiliti plovilo od startne do ciljne točke. Poleg razbremenitve človeškega operaterja mora biti avtopilot sposoben poiskati najkrajšo pot, kar pri daljših razdaljah pomeni, da mora uporabljati sferično trigonometrijo. Raziščite možnosti za izdelavo preprostega avtopilota te vrste. Izberite ustrezne senzorske in aktuatorske komponente ter izdelajte vezje avtopilota vključno z GPS sprejemnikom. Za stabilno uravnavanje krmila uporabite regulator PID. Izdelajte tudi vso potrebno programsko opremo. Zaradi preprostosti vgradite avtopilot v model avtomobila in na njem preverite pravilnost delovanja. Opišite rezultate in podajte možne načine za izboljšave.

Mentor:
prof. dr. Dušan Kodek

Dekan:
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Matija Mazalin,

z vpisno številko 63030174,

sem avtor diplomskega dela z naslovom:

Avtopilot za plovila na osnovi GPS

GPS based boat autopilot

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom
  prof. dr. Dušan Kodek
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 23.05.2012 Podpis avtorja:

Zahvala

Rad bi se zahvalil ožji družini, še posebej staršem, za podporo v dolgih letih študija. Zahvalil bi se tudi mojemu dekletu Jeannine za vso strpnost ob pisanju diplome. Posebno zahvalo za pomoč pri diplomi si zasluži še Marko.

Kazalo

Seznam uporabljenih kratic in simbolov

GPS (angl.) Global Positioning System, sistem globalnega določanja lege
UART (angl.) Universal Asynchronous Receiver/Transmitter, univerzalni asinhronski sprejemnik/oddajnik
PWM (angl.) Pulse Width Modulation
PID (angl.) Proportional Integral Derivative, avtomatski regulator v cikličnih procesih s povratno zvezo

Povzetek

Z diplomskim delom je bila želja realizirati avtomatski avtopilot za plovila, ki je zasnovan na GPS tehnologiji določanja lege. S tem bi se olajšale muke vsakega navtika, ki je kdaj prestal dnevno ali celo večdnevno navigacijo. Hkrati je pa to področje, ki v tem trenutku stagnira ter potrebuje nekaj vzpodbude, posebej z novimi idejami in izvedbami in tako pospeši nadaljni razvoj.

Poudarek pri diplomi je na izdelavi avtopilota na osnovi mikrokrmilnika ter nekaj modulov. Našteta in opisana je uporabljena strojna oprema, teorija računanja razdalje ter azimuta po različnih metodah. Nato je predstavljen regulator PID, razložen njegov pomen v industriji nekoč in danes ter opisan način nastavljanja le-tega. V nadaljevanju je prikazano delovanje avtopilota kot celote, prikazana je tudi blok shema. Sledi še opis kontrolne enote kot veznega člena med človekom in modelom ter prikaz zaslonske maske kontrolne enote. Na koncu je opisan še proces testiranja in težave pri le tem.

Ključne besede:

Avtopilot, PID, GPS

Abstract

The thesis idea was to design and implement an automatic boat autopilot, based on the GPS technology for positioning. This technology already exists, but what was intended here was an encouragement and a push forward of the development and evolution of automated autopilots.

The main part of the thesis consists on building the autopilot from various components accessible on the market. It is made from a microcontroller, GPS and wireless modules. It explains the theory behind the logic of the autopilot, like calculating the distance between two points and getting a course by different methods. Furthermore, it describes the PID controller's role in todays industry. There is a tuning guide for it as well. It is shown how the autopilot is supposed to work. The control unit, an application written in C#, serves as a link between an operator and the autopilot. At the end, there is the process of testing with the description of problems that were observed.

Key words:

Autopilot, PID, GPS

Poglavje 1

Uvod

Plovila, pri tem velja omeniti predvsem tista za prosti čas in užitek, so v večini opremljena s samostojnim krmilnim sistemom, t.i. avtopilotom. Tak avtopilot deluje izključno na podlagi meritve zemljinega magnetnega polja. Ker v prostem času veliko preživim na takšnih plovilih sem se odločil, da nekaj podobnega poskusim narediti tudi sam, vendar z nekaj ključnimi izboljšavami.

Avtopilot je naprava, ki je sposobna samostojno manevrirati premičnino, večinoma je tukaj mišljeno vodno ali zračno plovilo. Ponavadi ima taka naprava azimut ali pozicijo določeno vnaprej, njenemu cilju pa se poskuša le najbolj neposredno približati.

Avtopiloti so v vsakodnevnem življenju pogosto prisotni, predvsem v ladijski in letalski industriji, s pojavom le-teh v osebnih avtomobilih in vlakih bo pa njihovo število le naraščalo. Začetki segajo v prva leta 20. stoletja, ko so želeli razbremeniti del posadke na večjih tovornih in vojaških ladjah. Prvi avtopiloti so bili mehanski, smer so spreminjali glede na veter, kasneje so se jim pridružili tudi elektronski, bliskovit skok pa se je zgodil med drugo svetovno vojno. S pojavom tranzistorjev in mikrokrmilnikov pa se je avtopilot umestil tudi v civilnem življenju.

Obstoječi avtomatski piloti imajo nekaj bistvenih pomankljivosti. Če je bil še pred desetletjem razvoj le-teh na zadovoljivi ravni, zadnjih nekaj let stagnira. V dobi pametnih telefonov in ostalih zmogljivih računalnikov so naprave za krmiljenje plovil ostala brez komunikacije z le-temi. Naslednja neprijetnost izhaja iz dejstva, da namesto novejših, sposobnejših GPS modulov, za določitev smeri neba še vedno uporabljen električni magnetomer. Prednosti in slabosti

obeh bomo prikazali kasneje, je pa v današnjem času GPS modul vsekakor koristno in pomembno orodje, elektronski magnetomer pa le pripomoček.

Izdelana verzija, predstavljena v tej diplomski nalogi, ima nekaj ključnih točk, pri katerih se je vredno ustaviti in jih opisati podrobneje. Sestavljena je iz modelčka avtomobila in kontrolne enote. Za avtomobil sem se odločil zaradi enostavnejšega testiranja.

Poglavje 2

Oprema

2.1 Splošno

Namen avtomatskega pilota je pripeljati vozilo na željeno mesto. Izdelani je sestavljen iz dveh delov, osebnega računalnika z ustrezno aplikacijo ter mikrokrmilnika ATmega128A. Med seboj sta povezana z brezžično povezavo preko UART protokola, ki deluje na frekvenci 433Mhz. Mikrokrmilnik je povezan še z GPS napravo, ki s frekvenco 10 Hz sporoča trenutno pozicijo, hitrost, in azimut. Na mikrokrmilnik je priklopljen preko dodatnih UART vrat. Vsa koda na njem je napisana v programskem jeziku C, v aplikativnem delu na osebnem računalniku pa v C#.

2.2 Mikrokrmilnik

Uporabljen je mikrokrmilnik Atmel ATmega 128A. Je 8-bitni RISC procesor. Ima 32 prosto uporabnih registrov in 128KB notranjega pomnilnika Flash, 4KB EEPROM in 4KB SRAM pomnilnika, 53 V/I vrat, 4 PWM časovnike, 2 UART-a (oba uporabljena) [6]. Deluje na napetosti med 2,7 do 7,2 V. Programirno dostopen je preko JTAG in SPI vrat. V primeru avtomatskega pilota deluje s kristalom na frekvenci 11,0592 MHz. Za razvoj programske kode je uporabljen AVR Studio 4.19.

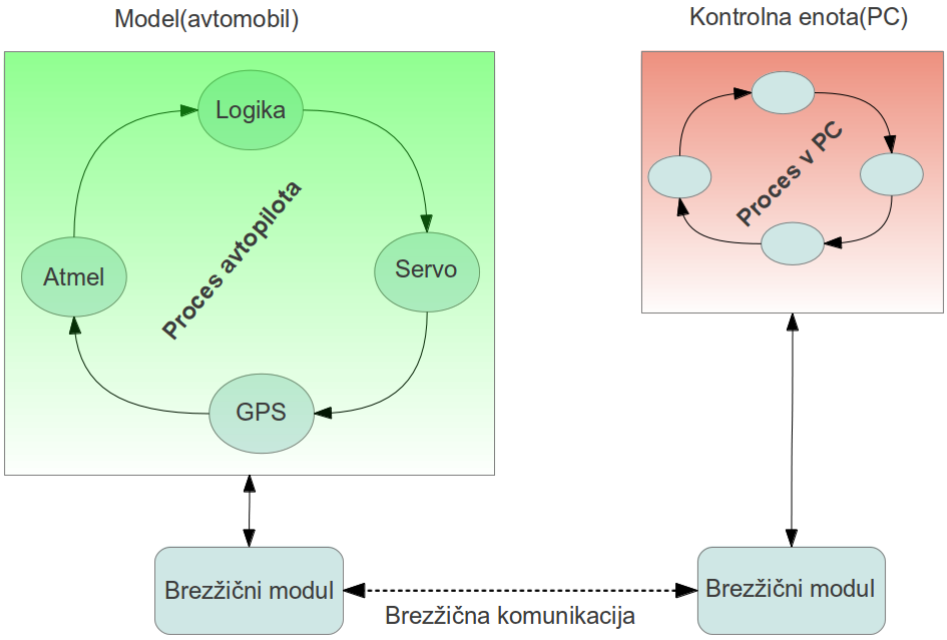
2.2.1 Razvojno orodje AVR Studio 4.19

V AVR Studio-u je mogoče programirati v zbirniku ali višjenivojskemu jeziku C. Prevajalnik jezika C je posebna različica odprtokodnega GCC-ja, prilago-

jena za mikrokrmilnike tega tipa, imenovana AVR-GCC. Postavitev kode na mikrokrmilnik je bila omogočena preko programatorja STK500.

2.3 GPS Venus SkyTRAQ

Modul GPS Venus je namenjen za razne mobilne naprave [5]. Odlikuje ga nizka poraba, visoka občutljivost. Njegova natančnost sega do 2,5 m. Dodano ima zunanjo 50 Ω anteno. V idealnih pogojih je sposoben povezave s sateliti v 29 sekundah. V praksi se izkaže, da je ta čas v povprečju potrojen.



Slika 2.1: Blok shema ideje

2.4 Wireless RF Transciever 431-478 Mhz

Komunikacija z modelom je rešena brezžično. Uporabljen je brezžični modul Wireless RF transciever, delujoč med 431 in 478 MHz. Deluje na razdalji do 1 km in hitrosti do 9600 bps. Ima možnost povezave preko UART/TTL, RS232 in RS485. Deluje na napetosti med 3,3 do 5,2 V [9]. V diplomski nalogi je uporabljen UART na 9600 bps, delujoč na 3,3 V (Zaradi zahteve GPS-a po tej napetosti).

2.5 Microsoft Visual studio 2008

Uporabljen je Microsoftov paket, aplikacija je pa napisana v C#. Paket ponuja raznovrstno paleto že integriranih knjižic in s tem omogoči veliko hitrejše oblikovanje in programiranje aplikacij. Končna aplikacija deluje v okolju operacijskega sistema Microsoft Windows, podrobno preizkušena je bila v okolju Windows XP SP3.

Poglavje 3

Osnovni pojmi

3.1 Splošno

Pred začetkom predstavljanja izdelka bi se veljalo ustaviti pri osnovnih pojmih, s katerimi bo razumevanje naloge v splošnem lažje.

3.2 Koordinate

Naš planet je razdeljen in označen s pomočjo posebnega koordinatnega sistema. Če potegnemo analogijo s kartezičnim koordinatnim sistemom, poznanim v matematiki, sta osi x in y postavljeni z ekvatorjem in Greenwich-em. Ekvator teče od zahoda proti vzhodu, na mestu, kjer ima planet največji radij. Je "naraven"vzporednik in kot tak na našem planetu tudi edini možen. Ekvator razmejuje severno in južno poloblo. Greenwich, poimenovan tudi kot prvi poldnevnik je določen s konvencijo in tako nič drugačen od ostalih poldnevnikov. Poteka od severnega pola skozi Greenwich-ev observatorij v bližini Londona do južnega pola.

Za določanje trenutne zemljepisne lege objekta uporabljamo zemljepisno širino in dolžino (latitude in longitude). Širina je merjena v stopinjah, razprostira se severno in južno od ekvatorja. V matematiki ji v skladu z dogovorom pripada simbol λ. Dolžina je prav tako merjena v stopinjah, razprostira se pa zahodno in vzhodno od Greenwicha. Dolžini pripada simbol L. V primeru bolj natančnega pozicioniranja se uporabljajo manjše enote od stopinj, minute in sekunde, vendar se pri tej nalogi, zaradi enostavnejšega preračunavanja, uporablja le decimalna vejica. Pri diplomski nalogi je uporabljen sistem po-

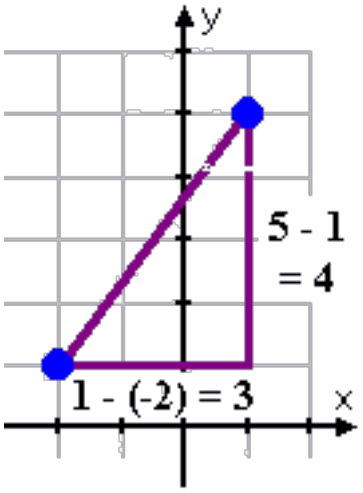
dajanja koordinat, ki je podoben kartezičnemu koordinatnemu sistemu. Tako se v prvem kvadrantu koordinatnega sistema, tj. severni polobli in vzhodno od Greenwich-a uporablja pozitiven predznak pri geografski dolžini in širini. Zahodno od Greenwich-a je dolžina predstavljena negativno, na južni polobli pa širina. Točka v Rio de Janeiru (južna polobla, zahodno od Greenwich-a) je tako podana kot ($\lambda = -22,967475, L = -43,178807$). Enako označitev uporablja tudi Google Earth.

3.3 Določanje razdalje in azimuta

Določanje azimuta na zemeljski obli ni enostavna operacija. Zaradi dobro znanega dejstva, da zemlja ni ploščata, je potrebno posegati po posebnih metodah za določanja azimuta, proti kateremu naj se vozilo obrne.

Pitagorov izrek

Za določanje azimuta in razdalje med točkama se morda zdi Pitagorov izrek primeren ter enostaven način. V praksi se izkaže, da ta teorija deluje samo na zelo kratkih razdaljah, z naraščajočo razdaljo se pa tudi napaka naglo povečuje.



Slika 3.1: Razdalja med koordinatami

Naj bo a razdalja med točkama

$$a = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{3.1}$$

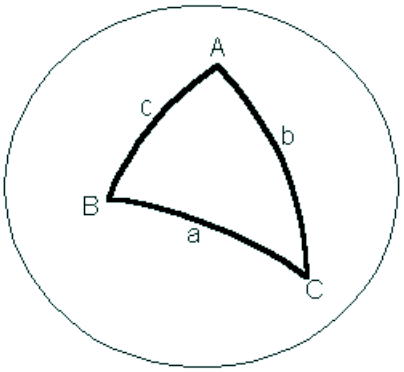
Podobno je tudi pri smeri (kotu α, azimutu).

$$\tan \alpha = \frac{(y_2 - y_1)}{(x_2 - x_1)} \tag{3.2}$$

Za določitev bolj natančnih vrednosti se je potrebno ozreti po sferični trigonometriji.

Sferična trigonometrija

Sferična trigonometrija je veda, ki se ukvarja s trikotniki na sfernih površinah. Izvira iz arabske kulture med 8. in 14. stoletjem, ko se je Arabcem, med osvajanjem severne Afrike ter južne Španije porajalo vprašanje:"V kateri smeri leži Meka?". Razvoj te ideje je prinesel izrazito napredovanje v navtični navigaciji, kartiranju zemlje in zvezd, določanju položaja sončnega vzhoda in zahoda ipd.



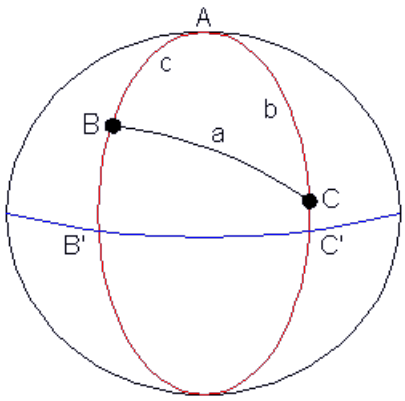
Slika 3.2: Trikotnik na sferi

Za trikotnike, ki se pojavljajo na sferičnih površinah, je značilno, da je vsaka stranica narisana z enakim radijem. To poimenujemo veliki krog. Na taki površini je tako taka črta najbližja razdalja med stranicami. Znana je še ena zanimivost, vsota kotov takega trikotnika meri več kot 180 stopinj.

Razdalja med točkama (metoda velikih krogov)

Z uporabo kosinusnega pravila lahko z lahkoto določimo razdaljo med skoraj vsakima točkama [8]. Izjema so le t.i. antipodne točke, torej točke, ki ležijo

na natančno nasprotni strani zemlje. Ker ta scenarij v praksi ni zelo verjeten, ta izjema ni zajeta v opisu.



Slika 3.3: Točke na zemlji, ter izračun razdalje med točkama

Na sliki 3.3 točki C in B predstavljata 2 točki na zemlji. Definirajmo naslednje:

- Točka A je severni pol.
- Črta ki povezuje točki B in C je najkrajša razdalja med njima (veliki krog).
- Veliki krog, ki povezuje B' in C' je ekvator.
- Veliki krog, ki povezuje A, B in B' je poldnevnik (zemljepisna dolžina) in hkrati zemljepisna dolžina točke B (L_B).
- Veliki krog, ki povezuje A, C in C' je poldnevnik (zemljepisna dolžina) in hkrati zemljepisna dolžina točke C (L_C).
- razdalja med točko B in B' ter C in C' je zemljepisna širina λ_B in λ_C.

Z uporabo kosinusnega pravila lahko določimo razdaljo med točko B in C na naslednji način (slika 3.3)

$$\cos a = \cos b * \cos c + \sin b * \sin c * \cos A \tag{3.3}$$

kjer je:

- A kot. Predstavlja razliko zemljepisne širine med B in C.
- c dolžina loka velikega kroga v radianih, med točko B (λ_B) in polom. Definirana je kot $\frac{\pi}{2}$ - širina (λ_B) in imenovana polarna razdalja.
- b dolžina loka velikega kroga v radianih, med točko C (λ_C) in polom. Definirana je kot $\frac{\pi}{2}$ - širina (λ_C) in imenovana polarna razdalja.

Pri upoštevanju pravila $\cos(\frac{\pi}{2} - x) = \sin(x)$ in $\sin(\frac{\pi}{2} - x) = \cos(x)$ dobimo

$$\cos(b) = \cos(\frac{\pi}{2} - \lambda_B) \Rightarrow cos(b) = sin(\lambda_B) \tag{3.4}$$

To pravilo lahko uporabimo povsod, kjer se pojavlja polarna razdalja in tako gornjo formulo poenostavimo

$$\cos(a) = \sin(\lambda_B) * \sin(\lambda_C) + \cos(\lambda_B) * \cos(\lambda_C) * \cos(L_C - L_B) \tag{3.5}$$

Če je a kotna razdalja, jo lahko pretvorimo v kilometre z uporabo enačbe

$$a_{km} = \frac{a * obseg}{360} \tag{3.6}$$

kjer je obseg zemlje enak 40.074 km(veliki krog).

Določanje azimuta

Kot je bilo videno že pri razdalji, se tudi azimut določa s pomočjo metode velikih krogov. Če želimo, kot je prikazano na sliki 3.3, potovati v smeri od točke B do točke C moramo poznati kot med ravnino in točko. Ta kot je imenovan azimut(kurz). Merjen je v stopinjah v smeri urinega kazalca in kot 0 kaže točno na sever. Za določanje azimuta se uporablja sinusno pravilo

$$\frac{\sin a}{\sin A} = \frac{\sin b}{\sin B} = \frac{\sin c}{\sin C} \tag{3.7}$$

kjer so (slika 3.3) (a, b, c) loki velikih krogov, (A, B, C) pa koti med velikimi krogi, ki sestavljajo sferični trikotnik. Absolutna smer iz B v C glede na sever je tako

$$\sin C = \sin A * \frac{\sin c}{\sin a} \tag{3.8}$$

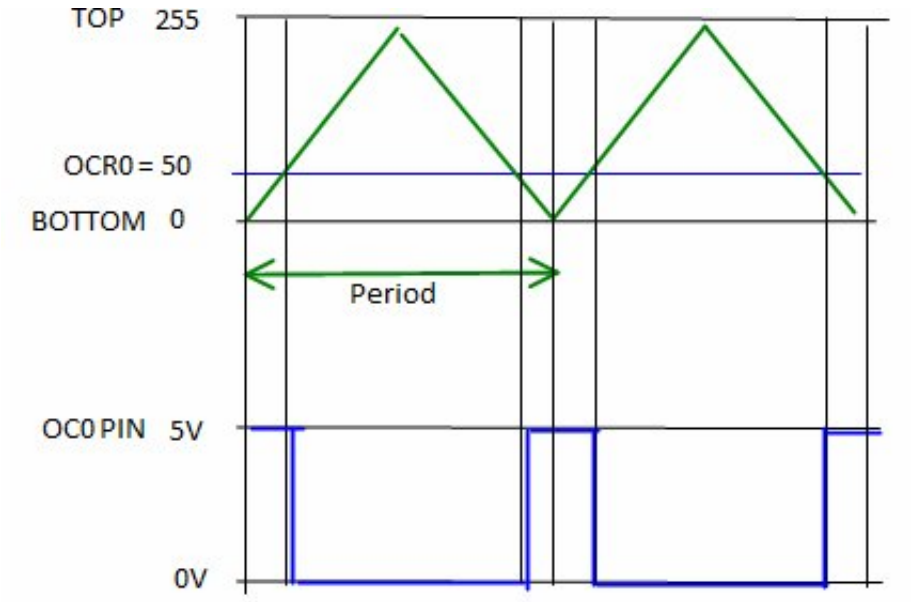
3.4 Krmiljenje servo motorja

Za krmiljenje servo motorja potrebujemo kratke signale napetosti z različno dolžino.

3.4.1 PWM

PWM ali Pulse Width Modulation ustvari signal, katerega dolžino se da poljubno podaljševati ali krajšati [1]. Ker imajo izhodi mikrokrmilnika le stanji 0V in 5V, se da s tem enostavnim periodičnim pulzom ustvariti povprečje napetosti, ki je nekje med tema vrednostima. Torej, če je dolžina visokega stanja enako dolga kot dolžina nizkega stanja, je povprečje izhodne napetosti enako $2,5$ V.

Kako PWM deluje na mikrokrmilnikih? Definirati je potrebno periodo štetja, izbiramo lahko med 1, 8, 64, 256 in 1024. Nato vpeljemo primerjalni register (Output Compare Register ali OCR) kateremu se določi vrednost med 0 in periodo štetja. Števec nato v ozadju "prikrito"šteje in vsakič, ko sta števec in OCR poravnana, se stanje na izhodu mikrokrmilnika spremeni (slika 3.4). Obstaja več različnih tipov PWM števcev, vendar je za krmiljenje servo motorja uporabljen zgoraj opisan model, poimenovan phase PWM. Bistvena razlika med phase PWM in običajnim je, kot je razvidno tudi iz slike 3.4, da je perioda števca dvojna. V tem načinu se tudi stanje na izhodu v vsaki periodi spremeni dvakrat. To je takrat, ko sta OCR in števec poravnana.



Slika 3.4: Prikaz števca s periodo 256 in spremembe stanja izhoda na mikrokrmilniku

Kako izračunati čas trajanja določenega stanja na izhodu? Ta izračun je

preprost, vzamemo le frekvenco procesorja ter jo delimo s števcem

$$f = \frac{f_{CPE}}{stevec} \tag{3.9}$$

kjer je f_{CPE} frekvenca delovanja kristala v mikrokrmilniku, f pa frekvenca števca. Po formuli $t = \frac{1}{f}$ nato dobimo čas trajanja med posameznim povečanjem števca. Tako je čas visokega[1] stanja (po sliki 3.4) enak

$$t_1 = t * OCR \tag{3.10}$$

Za izračun časa trajanja nizkega stanja (ali visokega) pa velja

$$t_2 = t - t_1 \tag{3.11}$$

3.4.2 PWM in servo

V projektu smo uporabili fazni PWM. Ker naj bi, po specifikacijah za upravljanje servo motorjev [2], bila perioda PWM dolga med $10 - 20$ ms, čas visokega stanja pa med $1 - 2$ ms, je uporabljen števec s periodo 256.

Tako je frekvenca

$$t = \frac{stevec}{f_{CPE}} \Rightarrow f = \frac{11.0592 * 10^6}{256} = 43200\,\text{Hz} \tag{3.12}$$

kar nam pove, da je perioda dolga $23\,\mu$s. Ker števec teče do 256 in nazaj sledi

$$23\mu s * 256 * 2 = 11,851\,\text{ms} \tag{3.13}$$

kar še vsebuje pogoj tistih $10 - 20$ ms. Pri upoštevanju zahteve visokega stanja $1\,\text{ms} \leq t_1 \leq 2\,\text{ms}$, izračun za določanje vrednosti registra OCR pokaže

$$t_1 = t * OCR \Rightarrow 23 \leq OCR \leq 45 \tag{3.14}$$

kjer vrednosti OCR med 23 in 34 pomenijo obrat servo motorja v levo, vrednosti med številoma 36 in 45 pa obrat v desno.

[1] Nizko ali visoko stanje glede na števec lahko nastavljamo v registrih mikrokrmilnika

3.5 PID

Za uravnavanje krmila vozila skrbi regulator PID. PID ali Proportional-Integral-Derivative regulator dandanes uporabljamo v vseh avtomatskih cikličnih procesih s povratno zvezo. Uporabljamo ga pri različnih proizvodnih procesih, kot npr za ohranjanje gladine tekočin v rezervoarju, za ogrevanje ali hlajenje prostorov, krmiljenje glave na tiskalnikih, ipd. Deluje ciklično na podlagi napake med željeno in dejansko točko, ki jo sistem sporoči algoritmu.

PID potrebuje tri ločene spremenljivke, za lažjo predstavo si lahko te spremenljivke predstavljamo s pomočjo časovnice. P upošteva trenutno napako, I hrani povprečje preteklih napak in D predpostavlja prihodnje napake. Če se osnovni parametri algoritma nastavijo optimalno, je lahko uravnotežena vsota teh napak zelo lep približek idealnim razmeram. Seveda obstaja veliko različnih vrst PID-ov, celo med avtomatskimi procesi ne poznajo standarda, kar velikokrat privede do zmede.

Prvi regulatorji PID so stara zadeva. V začetku 20. stoletja so regulatorji PID že znali krmiliti vojaške ladje. Prvo teoretično objavo o PID regulatorju je objavil ruski inženir Nicolas Minorsky [3]. Že takrat so ugotovili, da lahko tak proces veliko bolje upravlja barko kot človeška roka. Na trgu so se pojavili med drugo svetovno vojno, po vojni se je pa njihova uporaba razširila.

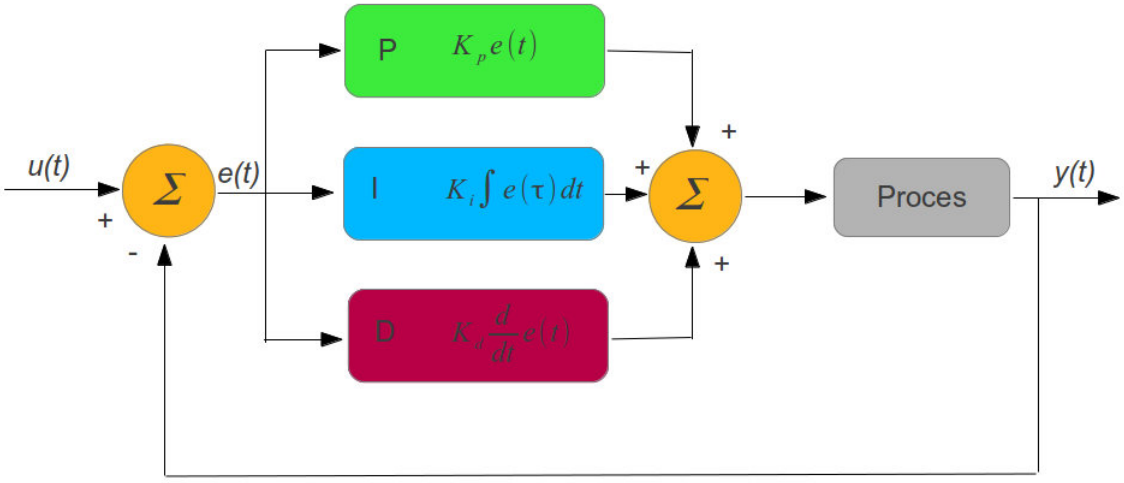
Osnovna formula algoritma PID

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\,\mathrm{d}\tau + K_d \frac{d}{dt} e(t) \tag{3.15}$$

kjer je:

- K_p: P parameter (uravnava trenutno napako)
- K_I: I parameter (uravnava akumulacijo preteklih napak)
- K_D: D parameter (uravnava predikcijo prihodnje napake)
- e: napaka (željeno (s) - sedanje stanje (a), $e = s - a$)
- t: čas

Pri iskanju optimalnega algoritma je potrebno parametre PID prilagoditi lastnostim procesa.



Slika 3.5: Shema delovanja zanke s procesom PID

3.5.1 Določanje parametrov algoritma PID

Vsak proces, ki ga krmili regulator PID ima svoje zahteve in lastnosti. S pomočjo konstant K_P,K_I in K_D se da marsikaj prilagoditi in pripraviti, da ustreza trenutnim razmeram. Največkrat najpomembnejša zahteva je stabilnost PID-a. Stabilnost je v tem primeru definirana kot počasno dušenje oscilacije okrog željene točke. Če ta zahteva ni izpolnjena, proces (plovilo ipd) oscilira okrog iste točke z enako ali celo vse večjo amplitudo. Pri slabo definiranih parametrih se včasih celo zgodi, da do oscilacije sploh ne pride, temveč sistem le divergira, dokler ne pride do saturacijskega filtra (v algoritmu) ali mehanske končne ovire (servo motor se obrne do končne vrednosti).

Kalibriranje procesa PID je, čeprav gre le za tri parametre, težavna naloga. Zaradi izključujoče želje po čimvečji stabilnosti ter hitremu odzivu sistema, je potrebno posvetiti veliko časa poskušanju in popravljanju sistema.

Eden od receptov za ročno kalibriranje se ponavadi prične tako, da se parametra K_I in K_D [4] postavi na vrednost 0. Nato se K_P povečuje, dokler sistem ne začne oscilirati. Po začetku oscilacije se K_P postavi na polovico. Nato se začne popravljati K_I za uravnavanje zamika. Vendar pri prevelikem K_I je

Tabela 3.1: Učinek neodvisnega povečanja posamičnega parametra

Parameter	Čas vzpona	Prestop	Čas umiritve	Stabilnost
K_P	Pada	Narašča	Majhna sprememba	Zmanjšana
K_I	Pada	Narašča	Narašča	Zmanjšana
K_D	Manjši padec	Manjši padec	Manjši padec	Izboljšana

sistem nestabilen. Nato se prične s povečevanjem K_D za hitrejšo postavitev na željeno mesto. Paziti je potrebno le da K_D ni previsok, saj lahko povzroči pretiran odziv in s tem velik prestop čez željeno točko.

Obstaja tudi že več definiranih metod kalibriranja kot je metoda *Ziegler-Nicholsa* in *Cohen-Coona*. Ker je bila v tej nalogi uporabljena ročna metoda kalibracije, so ostale metode le omenjene.

3.5.2 Simulacija regulatorja PID

Zaradi tehničnih omejitev je bila sprva opravljena le simulacija regulatorja PID. Poleg določanja parametrov K_P,K_I in K_D je bilo potrebno najti verodostojno simulacijo poti. Poleg tega je bilo potrebno pri algoritmu regulatorja PID upoštevati, da so strani neba omejene s periodo 360 stopinj ali 2π. Na osnovnem primeru, razlika med azimutom 45 in 315 ni 270 stopinj, ampak le 90 stopinj (slika 3.6). To pomeni, da je največja razlika med željenim in dejanskim azimutom 180 stopinj.

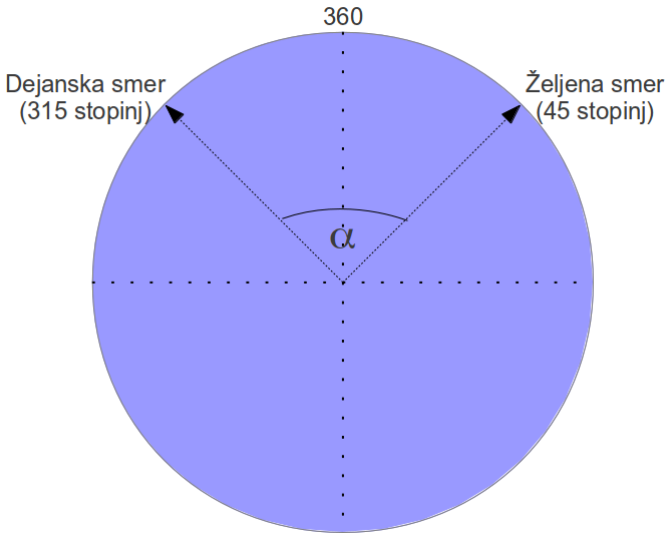
Za spremenljivko e je pri algoritmu regulatorja PID uporabljeno [10]

$$e = (s - a) - 360\lfloor 0.5 + \frac{(s-a)}{360} \rfloor \tag{3.16}$$

kjer je:

- s setpoint ali željeno stanje
- a actual point ali dejansko stanje

Pri prvih poskusih je algoritem, ne glede na parametre regulatorja PID, postavljal na željeno stanje brez prenihajev. Do tega je prihajalo zaradi neskončno kratkega časa med akcijo in reakcijo sistema. Torej, ko je algoritem dal znak, da je potrebno vozilo obrniti za n stopinj, je simulacija to storila

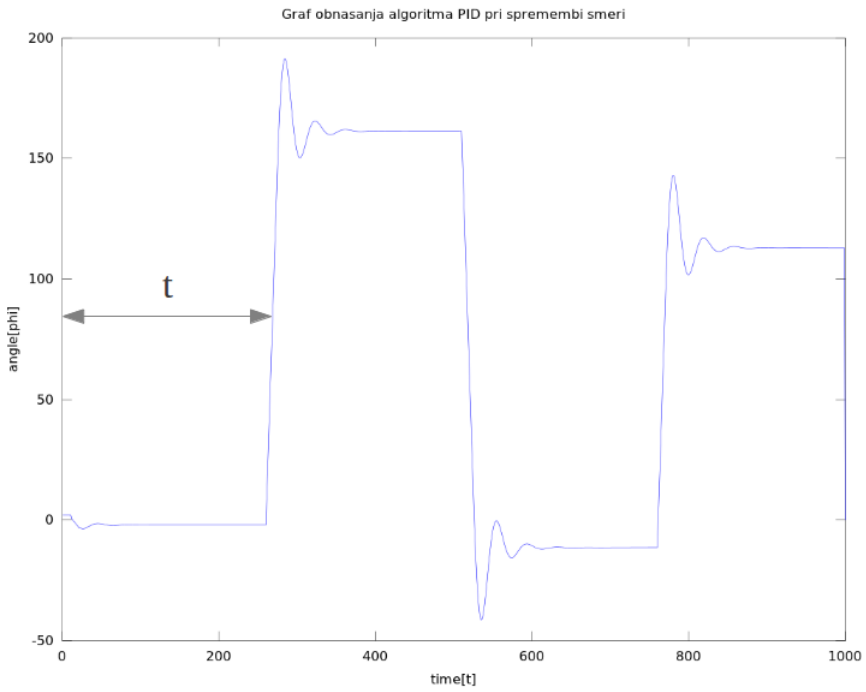


Slika 3.6: Problem periode 360 stopinj

brez časovnega zamika in zatorej ni bilo potrebe po popravljanju. V resnici seveda ni tako, saj je med akcijo in reakcijo potreben določen čas Δt. Ker je ta čas Δt v realnem življenju končno kratek (ali dolg) privede do prekoračenja azimuta ter popravljanja le tega. Pri simulaciji se je bilo potrebno odločiti za nek časovni zamik Δt, kar je pomenilo vpeljavo dodatnega parametera. Izbran je bil zamik $\Delta t = 10$. Nadaljnje določanje parametrov K_P,K_I in K_D pri simulaciji je bilo v skladu s prejšnjim podpoglavjem.

Graf (slika 3.7) prikazuje kako regulator PID spreminja azimut. Prične se z $a = 2$ in željo $s = 358$ $(= -2)$, nato se po času $t = 250$ spremememeni $s = 160$ ter nato še na $s = 350$ $(= -10)$ in $s = 112$. Parametri PID si sledijo, $K_P = 0.1$, $K_I = 0.0025$ in $K_D = 0.0005$.

Pokazalo se je, da poleg standardnih PID parametrov tudi zamik igra pomembno vlogo. V resničnh pogojih je ta parameter različen od primera do primera, je pa velikokrat tudi spremenjljiv (vodni ali zračni pogoji, valovi,..). Če bi bil zamik nelinearen pa najverjetneje regulator PID sploh ne bi deloval. Če je zamik povečan za $\Delta t \geq 15$ ves sistem postane nestabilen. Sistem prične z oscilacijo okrog željene točke ter hkrati povečuje amplitudo. Če je postavljen na zamik $\Delta t \leq 5$ pa je linija praktično idealna. Tako sem bil podvržen



Slika 3.7: Graf simulacije obnašanja vozila pri spreminjanju smeri

subjektivni odločitvi, da je zakasnitev $\Delta t = 10$ najboljša izbira. Po določitvi zamika so bili izbrani tudi preostali parametri regulatorja PID. Zaradi tega so tudi precej odvisni od Δt.

Poglavje 4

Opis delovanja

4.1 Splošno

Vsa zahtevna logika, preračunavanje azimuta, poganjanje servo motorja in regulator PID, se izvajajo na mikrokrmilniku. V splošnem lahko model deluje tudi brez povezave na kontrolno enoto, problem se pojavi le pri postavitvi končne točke (privzeta točka je Marina Portorož, kapitanski most). Je pa to v tem projektu onemogočeno, saj se po 20 sekundah brez signala avtopilot avtomatsko izklopi.

4.2 Osnovna predstavitev delovanja

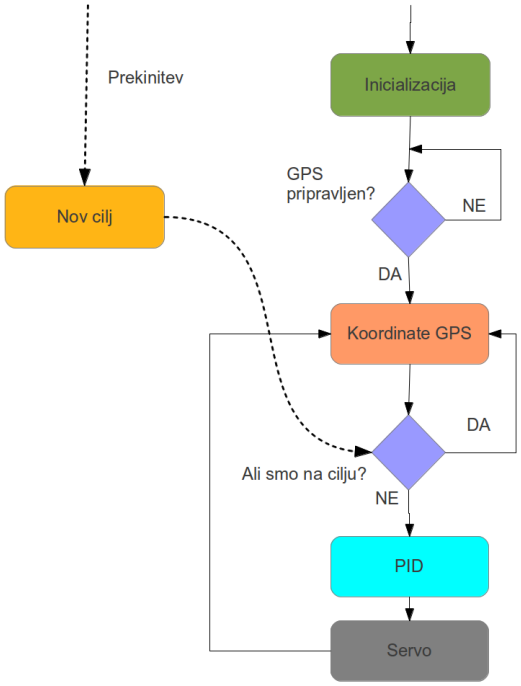
Uporabljeni mikrokrmilnik z ATmega128A deluje na sledeči način:

1. **Inicializacija:**
   Pri inicializaciji se sproži algoritem ki postavi procesor v željeno stanje in poskusi vzpostaviti stik z GPS modulom. Mikrokrmilnik tukaj deluje brez prisotnosti kontrolne enote, torej aplikacije na osebnem računalniku.

2. **Inicializacija GPS-a:**
   Pri GPS modulu je potrebno počakati vsaj 29 sekund, včasih pa modul potrebuje tudi nekaj minut, preden pridobi signal vsaj treh satelitov. Vse dokler sateliti niso pridobljeni, je avtopilot v fazi inicializacije.

3. **Pridobitev koordinat:**
   Ko GPS vzpostavi povezavo s sateliti, začne s frekvenco 10 Hz pošiljati podatke preko UART vrat na mikrokrmilnik. Ta parametre interpretira in shrani, hkrati jih pa pošlje tudi kontrolni enoti.

4. **Zanka:**
   Po pridobitvi končnega cilja (privzet je cilj Marina Portorož, spremeni ga lahko le kontrolna enota) se avtopilot postavi v zanko, kjer izmenično pridobiva koordinate (preko prekinitev), preračunava azimut, hitrost in oddaljenost od cilja ter krmili servo motor, vse izračune pa pošilja tudi preko vrat UART kontrolni enoti.



Slika 4.1: Diagram poteka

4.3 Pridobitev podatkov iz GPS-a

Vsako sekundo GPS pošlje 4 vrstice podatkov skladno z NMEA standardom [11]. NMEA ali Nation Marine Electronics Association je ponudilo standard, s katerim lahko elektronske komponente komunicirajo z ostalimi napravami. Vsaka vrstica se vedno začne z znakom $, konča pa z znakom za novo vrstico(<CR><LF>). Iz teh vrstic ima mikrokrmilnik možnost prebrati

UTC čas, koordinate, hitrost, nadmorsko višino in še nekaj indikativnih parametrov, kot so jakost signala, število satelitov v uporabi ipd. Mikrokrmilnik uporablja le podatke trenutne pozicije in hitrosti ter kurza, preostali parametri se izračunajo naknadno v procesorju. Mikrokrmilnik dobiva posodobljene podatke preko vrat UART, ob vsakem novoprispelem podatku pa se sproži prekinitev. Ko je vrstica polna, jo mikrokrmilnik zaključi in pošlje v obdelavo. Prekinitveni vektor je rešen na naslednji način:

```
ISR(USART0_RX_vect)
{
  char Temp;
  Temp=UDR0;      //Store data to temp
  if(Temp=='$') //After NL, finish old
  {
    test0_string[index0_string]='\0';
    index0_string=0;
  }
  else  //add byte
    test0_string[index0_string++]=Temp;
}
```

Primer GPS vrstice:

```
$GPGGA,111636.932,2447.0949,N,12100.5223,E,1,11<CR><LF>
```

Razčlenitev vrstice je opisana v SkyTraq-ovi dokumentaciji [5].

Mikrokrmilnik je brez večjih zapletov sposoben vrstico razčleniti v sebi razumljivo obliko.Verjetnost napake razčlenitve je okrog enega odstotka.V praksi to pomeni da vsak stoti izračun poskusi speljati avtopilota iz smeri. Zato je dodan varnostni mehanizem, ki nekatere nesmisle odstranja, kar zmanjša možnost napake na polovico. Podobno razčlenitev naredi tudi v primeru podajanja podatkov s strani aplikacije na kontrolni enoti.

```
static void tokenize(char *piki)
{
uint8_t n = 0;
piki = strtok(piki, ",");
if ((strncmp(piki,"GPRMC",4) == 0))  //ce 1.token=GPRMC
{
        while (piki) {
```

```
        strcpy(&words1[n++], piki);    //vsak token posebi
        piki = strtok(NULL, ",");      //najdi nov token.
    }
}
```

4.4 Obdelava podatkov

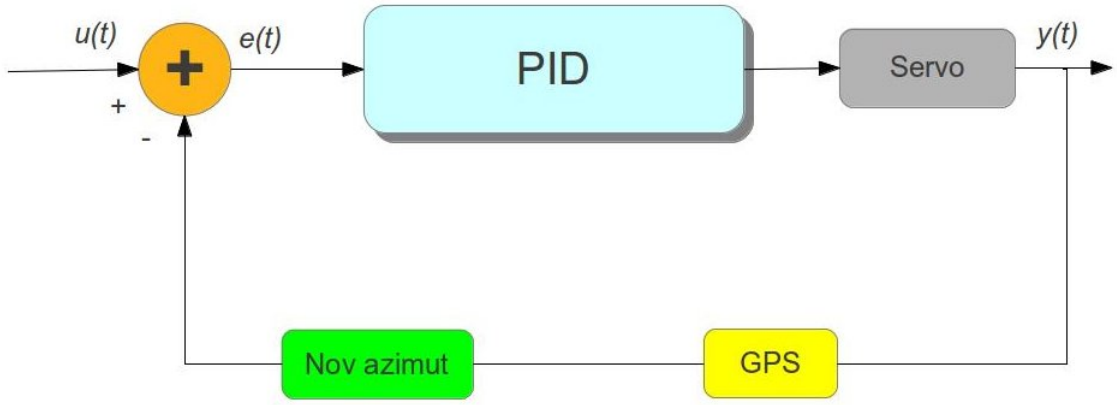
Dobljeni podatki služijo nadaljni obdelavi in posledično sprožanju akcij. Po preračunanju azimuta in razdalje med točkama, to naredi po metodi velikih krogov ter s pomočjo Pitagorovega izreka, pošlje podatke kontrolni enoti, hkrati se pa prične prožiti tudi regulator PID za krmiljenje avtopilota. Naprava manipulira z naslednjimi parametri:

- trenutne koordinate (λ_B, L_B),
- željene koordinate (λ_A, L_A),
- trenutni azimut (pridobljen z GPS-om ter magnetomerom),
- trenutna hitrost.

Med manipulacijo teh parametrov naprava izračuna razdaljo med točkama, potreben azimut za dosego željene koordinate in čas, ki je potreben za potovanje. Ko se naprava približa cilju, tj. željeni koordinati v radiju manjšem od $\epsilon < 1\,\mathrm{m}^2$ se servo motorja izklopita. Če bi bila naprava nameščena na večje vozilo, bi bil ta ϵ večji.

4.5 Krmiljenje

Ves opis postopka v prejšnjih podpoglavjih služi le namenu fizičnega krmiljenja naprave. V regulator PID se pošljeta le dve spremenljivki [7], trenutna in željena koordinata. Algoritem preračuna trenutno, preteklo in prihodnjo napako, ter vrača vrednosti (nominalne OCR vrednosti), s katerimi neposredno krmili servo motor. Kot že omenjeno se vrednosti OCR gibljejo med 21 in 32 za obrat v levo in med 33 in 44 za obrat v desno. Večja kot je napaka (željena-trenutna koordinata), dlje od centra se pojavlja OCR. Za spremenljivki MAX in MIN sta OCR 44 oz 21. Pri testiranju se je pojavil še problem izbiranja konstant K_P,K_I in K_D. Izbrane so bile ročno, po metodi poskušanja, glede na obnašanje modelčka.



Slika 4.2: Notranja shema delovanja avtopilota

```
//Define parameter
#define epsilon 0.01
#define dt 0.01
#define MAX  4
#define MIN -4
#define Kp  0.1
#define Kd  0.01
#define Ki  0.005

float PIDcal(float setpoint,float actual_position)
{
        static float pre_error = 0;
        static float integral = 0;
        float error;
        float derivative;
        float output;

        //Caculate P,I,D
        error = setpoint - actual_position;

        //In case of too small error stop integration
        if(abs(error) > epsilon)
                integral = integral + error*dt;
```

```
        derivative = (error - pre_error)/dt;
        output = Kp*error + Ki*integral + Kd*derivative;

        //Saturation Filter
        if(output > MAX)
                output = MAX;
        else if(output < MIN)
                output = MIN;

  //Update error
  pre_error = error;

  return output;
}
```

4.5.1 Izbiranje konstant K_P, K_I in K_D

Izbiranje parametrov regulatorja PID je potekalo ročno, po postopku opisanem v prejšnjem podpoglavju in samem opazovanju naprave.

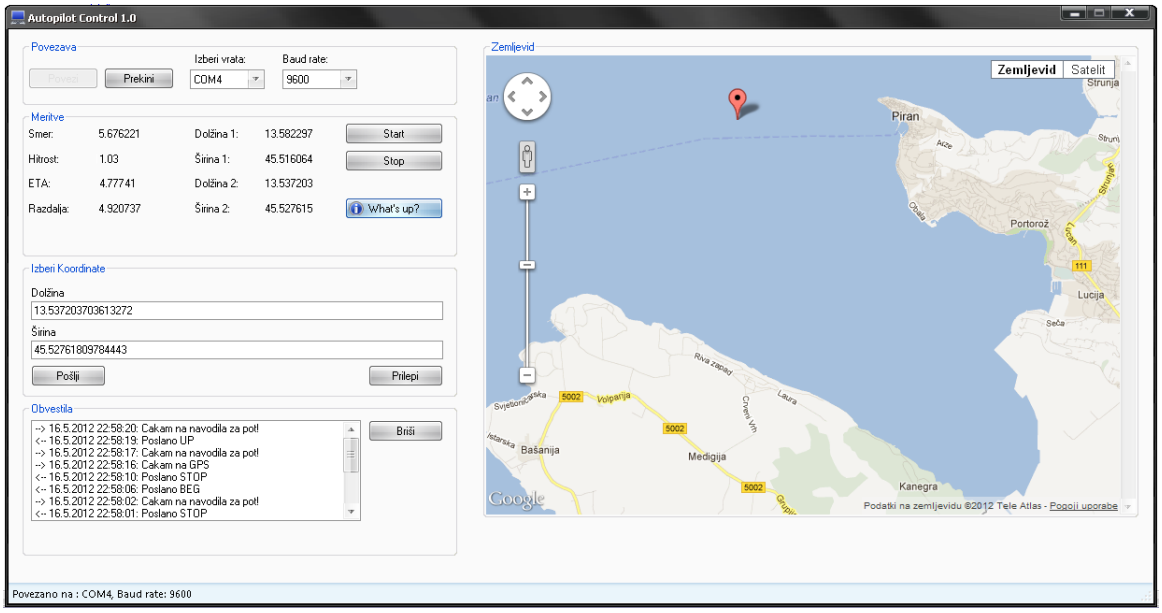
4.6 Kontrolna enota

Kontrolna enota je v osnovi zelo preprosta aplikacija, napisana v programskem jeziku C# in teče v okolju Microsoft Windows. Je nekakšen prikazovalnik novih podatkov o trenutni poziciji in azimutih, edino aktivno vlogo pa poda le pri postavitvi nove ciljne točke. Je posrednik med človekom in avtopilotom. Namesto obstoječe aplikacije bi bila lahko narejena tudi alternativna, delujoča na pametnem telefonu ali tablici.

4.6.1 Delovanje

Za pravilno delovanje kontrolne enote je potrebno imeti poleg osebnega računalnika tudi brezžični modul ter UART/TTL pretvornik. Po zagonu le-te sama poišče seznam serijskih vrat COM. Hkrati se kontrolni enoti določi tudi hitrost prenosa (običajno je to 9600 bps). Za aplikativno povezavo med modelom in kontrolno enoto je uporabljena knjižnica C# serial.COM.

Aplikacija nato prikazuje trenutno in željeno pozicijo, azimut, hitrost ter čas do prihoda. V spodnjem delu programa je okno za vpis novih koordinat. Ker je komunikacija z modelom lahko popačena, aplikacija sproži veliko pasti za lovljenje izjem (ti. exceptions, op.a.). Tako kontrolna enota popačene, neskladne podatke, ki jih prejme, preprosto ignorira. V splošnem je pa le opazovalec ter nadzornik modela.



Slika 4.3: Uporabniški vmesnik kontrolne enote

Podajanje koordinat

Aplikacija sprejme koordinate v obliki stopinj z decimalno vejico, torej brez minut ali sekund. Prvi kvadrant je severna polobla ter vzhodno od Greenwicha. Za pošiljanje pozicije, ki se nahaja zunaj tega kvadranta, je potrebno postaviti pred ustrezno koordinato negativen predznak. Aplikacija pošlje vrstico modelu v naslednji obliki:

```
#POS,X,Y
```

kjer je:

- Koordinata X geografska širina.

- Koordinata Y geografska dolžina.

Uspešnost poslanega se prikaže pri indikatorju cilja v zgornjem delu aplikacije. Izbira koordinat je poenostavljena tudi z manjšim spletnim vmesnikom, na katerem je postavljena spletna stran Google Maps©.

Poglavje 5

Testiranje

Po vmestitvi vseh komponent na model avtomobila je bilo na vrsti testiranje. Za napajanje je bila uporabljena baterija z napetostjo 9,2 V. Že po prvih poskusih je bilo jasno, da je GPS modul za prikazovanje trenutnega azimuta premalo odziven. Tako se je avtomobil, preden se je podatek o njegovi poti osvežil, že nekajkrat zavrtel okoli svoje osi. GPS-u je bila nato dvignjena frekvenca osveževanja na 10 Hz in težava je bila rešena.

5.1 Testna procedura

Procedura testiranja je bila razdeljena na več delov:

- Testiranje elektronskih komponent;
- Preverjanje stikov, spojev in povezav, ki bi lahko pri obremenjenosti popustili, se razdrli ali kako drugače onesposobili mehanizme;
- Preverjanje mehanizmov na modelu avtomobila, od krmilnega mehanizma, diferencialov ter zobnikov;
- Testiranje delovanja regulatorja PID v resničnih pogojih;
- Poskus vodenja in upravljanja vozila na daljavo, samo s podajanjem koordinat;

Prve tri točke so bile preverjane tekom razvoja in so tako preverjeno delovale. Zato sem se osredotočil na zadnji dve točki.

5.1.1 Testni pogoji

Za testiranje sem si izbral veliko asfaltno površino, poligon šole varne vožnje v Izoli, ter nogometno igrišče bližnje osnovne šole. Sprva sem določil 4 koordinate, ki so ustrezale smerem neba S, J, V, Z. Ker je v specifikaciji za brezžični modul podano [9], da oddajnik in sprejemnik lahko komunicirata pri razdalji do 1 km, sem poskusil to tudi v praksi. Izkazalo se je, da pri hitrosti 9600 bps komunikacija deluje le do 100 m zračne, neovirane razdalje.

5.1.2 Testiranje regulatorja PID

Prvi testi so pokazali, da je GPS modul prepočasen ter preokoren za kakršnokoli ažurno informacijo. Za spremljanje preteklih dogodkov je zelo primeren, njegova odzivnost na tekoče dogodke je pa v splošnem zelo slaba. GPS je sicer deloval s frekvenco 1 Hz, kar nikakor ni dovolj za hiter odziv na spremembe. Morda bi delovalo pri manjši hitrosti vozila ali večjem radiju zavoja. Pri prvem testiranju je avtomobil nemudoma pričel naključno krožiti ter zavijati. Po povečanju frekvence osveževanja na 10 Hz, se je odzivnost znatno izboljšala in testiranje se je nadaljevalo. Parametri PID so bili že od začetka postavljeni na vrednosti iz simulacije, $K_P = 0.1$, $K_I = 0.0025$ in $K_D = 0.0005$. Pokazalo se je, da avto s temi PID parametri zavija dokaj primerno. Sledil je poskus testiranja po navodilih iz tabele 3.1, vsi parametri so bili postavljeni na 0 ter ob vsakem novek koraku je bil povečen K_P. Ustalil se je pri $K_P = 0.14$,$K_I = 0.048$, in $K_D = 0.0004$. Razlika med različnimi vrednostmi parametrov zelo hitro vpliva na ves rezultat. Če postavimo $K_D = 0.001$ avto popolnoma izgubi kontrolo in oscilira z vse večjo amplitudo.

Testiranje je potekalo tako, da je bil ob začetku avtomobil vedno postavljen v smer severa. Nato je bila napravi naložena določena točka[1] (točka je bila neskončno daleč, zato je avto ni nikoli dosegel), kateri naj se približa. Avto je moral nato sprva zaviti v zahtevano smer, nato pa to smer držati. Avto je pri lepo izbranih PID parametrih navadno zavil, ko je bilo potrebno voziti v ravni črti, se mu je pa nekoliko zapletlo. Večkrat se je pripetilo, da je avtomobil osciliral, celo z amplitudo 1 m okrog željene črte. Na rezultat testiranja je močno vplivala tudi hitrost. Če so določeni PID parametri pri najnižji hitrosti delovali zadovoljivo, so pri hitrostih $v \geq 5\ \frac{m}{s}$ popolnoma odpovedali.

[1] S, J, V, Z

5.1.3 Poskus vodenja

Ko je krmiljenje avtomobila delovalo, je vodenje potekalo razmeroma enostavno. Večji problem je spet predstavljal GPS modul, saj je avtomobil zlahka zgrešil ciljni radij. To se je dogajalo predvsem pri krajših razdaljah. Veliko vlogo je igrala tudi hitrost modelčka. Ker je modelček napajan s prenosno baterijo, se je hitrost zelo spreminjala glede na dolžino testiranja in padca napetosti v le-tej.

Pri vodenju je pomembno povdariti, da bi pri počasnejših plovilih, katerim je ta avtopilot pravzaprav namenjen, najverjetneje deloval bolje in bolj stabilno. To si upam sklepati na podlagi hitrosti ter možnostih obračanja takšnih plovil.

Poglavje 6

Sklepne ugotovitve

Izvedba samega projekta je zorela v moji glavi že dolgo. Čeprav so se verzije in načini izvedbe skozi čas spreminjale, je ideja ostala enaka. Pripeljati plovilo na točno določen cilj ne glede na spremembe zunanjih faktorjev. Naj bo to morski tok, valovanje ali veter, plovilo mora najti pot do cilja, tudi če mora pri tem spremeniti azimut. Žal sem moral sam avtopilot namestiti na model avtomobila, saj je bilo tako olajšano testiranje, rešilo je pa tudi problem vodotesnosti, vendar v splošnem ideja ostaja ista.

Projekt je sestavljen iz modela avtomobila z mikrokrmilnikom ter kontrolne enote na računalniku. Sam avtopilot ter vsa logika se nahaja v mikrokrmilniku, ki skrbi za vse delovanje, od pridobivanja ter računanja parametrov, pa do pogona in krmiljenja modela. krmiljenje je bilo napisano v programskem jeziku C, aplikacija na PC pa v C#.

V projektu sem se nenehno srečeval s problemi, nemalokrat tudi večjimi, kjer bi večkrat skoraj obupal. Ker sem sprva imel omejeno znanje pri programiranju mikrokrmilnikov, mi je bil v velik izziv že nastavitev registrov za UART vrata. Naslednji, in še večji izziv je bila nastavitev I^2C vodila. Presenetil me je tudi teoretični del, različne teorije določanja azimuta in razdalje na sferičnih površinah.

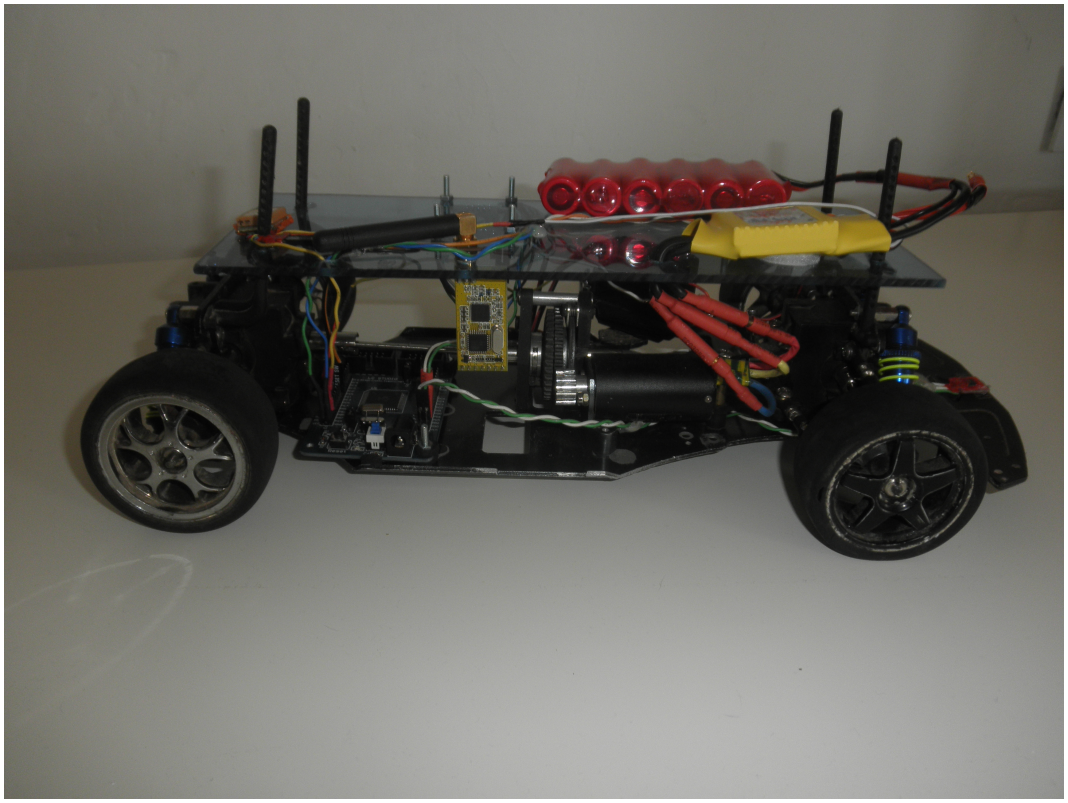
Izboljšave tega projekta so vsekakor možne, prvo in glavno vidim v tem, da bi se kontrolni del preselil na prenosne telefone ali tablice. Pri tem bi uporabil merilec pospeškov, ki je vgrajen v vsak novejši telefon, in modelček vodil samo s premikanjem le-tega. To bi velikokrat lahko prišlo prav navtičarjem, ki morajo ob vsaki nenadejani oviri teči nazaj pred krmilo. Naslednja ideja bi lahko

bila tudi povezava z radarjem, kjer bi se naprava samostojno odločala in umikala morebitnim oviram. Morda bi lahko v poštev prišla tudi sinhronizacija s pomorsko karto, kjer bi logika avtopilota že vnaprej lahko ugotovila pot in se na njej umikala raznim statičnim objektom, kot so kopno, plitvine in svetilniki.

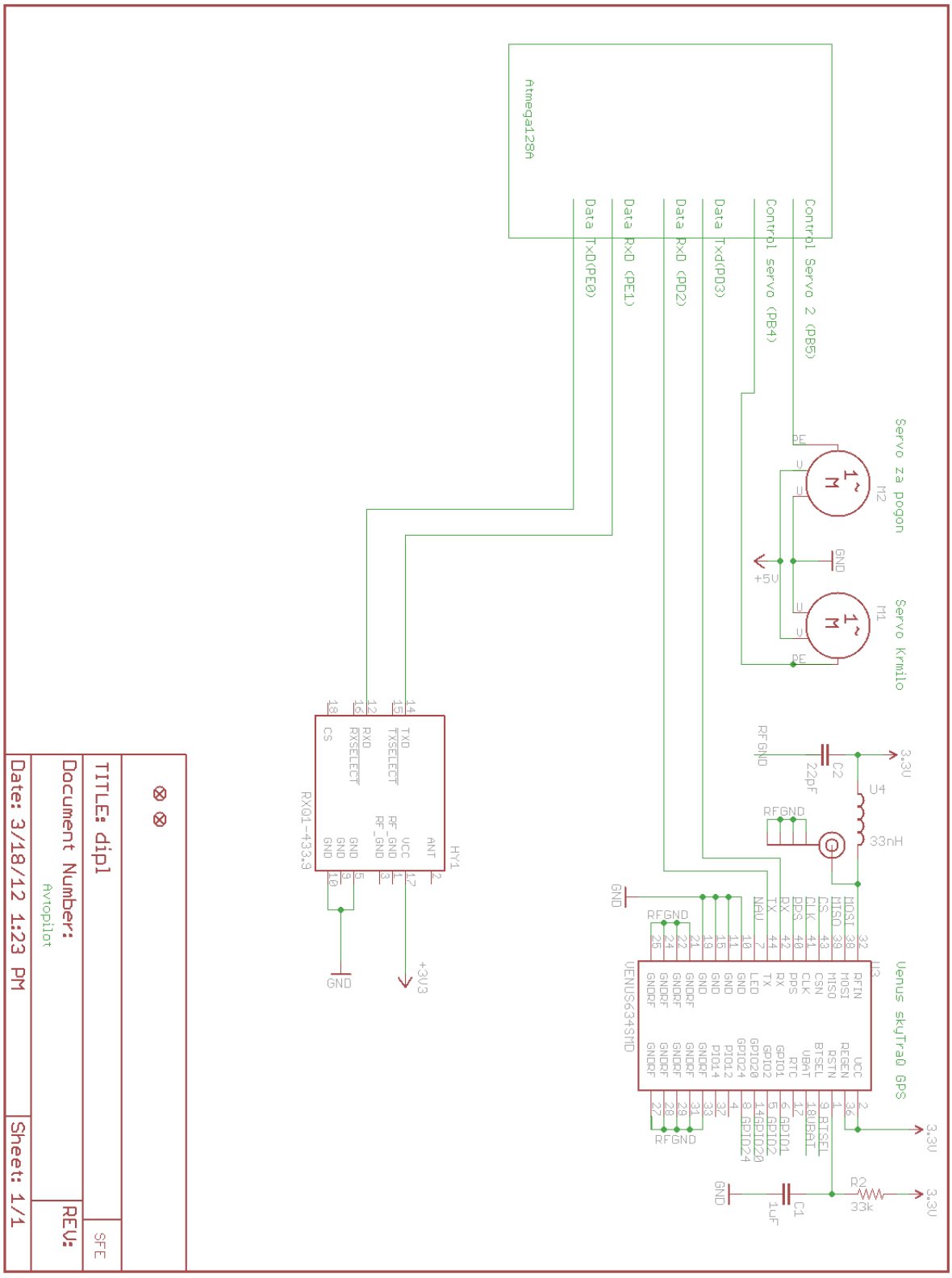
Diplomska naloga bi najverjetneje lahko bila izboljšana tudi z nakupom relativno dražjih komponent. Treba se je zavedati, da je obstoječi avtopilot sestavljen iz najcenejših komponent iz njihovega področja. Že samo s tem bi lahko projekt postal veliko bolj zanesljiv. Vseeno mislim, da je mi je diplomska naloga dobro uspela, saj sem sestavil sistem, ki je sposoben priti, sicer s pomočjo človeka, praktično kamorkoli na zemlji.

Dodatek A

Sheme in slike



Slika A.1: Modelček z vezjem



Slika A.2: Shema vezja

Slike

Literatura

[1] (2011) John Palmisano, "PWM tutorial." Dostopno na: http://www.societyofrobots.com/membertutorials/node/229

[2] (2011) Burford J. Furman, "Interfacing a Servo to the ATmega16." Dostopno na: http://www.engr.sjsu.edu/bjfurman/courses /ME106/ME106pdf/servo-atmel.pdf

[3] Stuart Benneth, "Nicholas Minorsky and the automatic steering of ships," *IEEE Control Systems Magazine* št.4, str. 10–15, november 1984

[4] John A. Shaw, "The PID Control Algorithm, how it works, how to tune it, and how to use it," *Process Control Solutions*, December 2003.

[5] (2011) Venus Skytraq Technology, "VENUS634FLPx Datasheet"

[6] (2011) Atmel Corporation, "ATmega128/L Datasheet"

[7] (2011) Tam87, "PID control algorithm-C language," *Embedded's heaven.* Dostopno na: http://www.embeddedheaven.com /pid-control-algorithm-c-language.htm

[8] (2009) Krystall, "The Great circle." Dostopno na: http://www.krysstal.com /sphertrig.html

[9] (2009) Sure Electronics Co. Ltd.," Wireless RF Transciver 431-478 Mhz GFSK Data Transfer User's Guide"

[10] (2011) Ether, "PID crossing 0." Dostopno na: http://www.chiefdelphi.com/forums /archive/index.php/t-95370.html

[11] (2002) The National Marine Electronics Association, "NMEA 0183 Standard," Dostopno na: http://www.gpsinformation.org/dale/nmea.htm