"Mladi za napredek Maribora 2018"
35. srečanje

IP preko vsega

RAČUNALNIŠTVO

Raziskovalna naloga

Prostor za nalepko

Avtor: PETER BERBERIH, MIHA FRANGEŽ
Mentor: BRANKO POTISK
Šola: SREDNJA ELEKTRO-RAČUNALNIŠKA ŠOLA

Maribor, februar 2018

“Mladi za napredek Maribora 2018”
35. srečanje

IP preko vsega

RAČUNALNIŠTVO

Raziskovalna naloga

Prostor za nalepko

Maribor, februar 2018

Kazalo vsebine

Kazalo slik

POVZETEK

Prenašanje digitalnih informacij preko večjih razdalj je izjemno pomemben del modernega življenja. A kot pri večini mrežne tehnike so naprave, potrebne za to, zelo drage in prezapletene za izdelavo doma.

Mii se bomo problema lotili z druge strani – raziskali bomo načine vzpostavljanja mrežne povezave na nekaj poceni in relativno enostavnih načinov, ki jih lahko uporabimo brez zapletene strojne opreme. Prav tako bomo pri tem uporabili izključno odprtokodno programsko opremo.

IP (Internetni Protokol) povezave bomo poskusili vzpostaviti laserskih žarkov, radijskih povezav in drugih signalnih metod, ki prenosu digitalnih podatkov sicer niso namenjene.

1. CILJ

Cilj te naloge je raziskati čim več bolj ali manj enostavnih načinov vzpostavitve decentraliziranega omrežja med računalniki. Čeprav imamo v današnjih časih povezavo z internetom za nekaj samoumevnega, pogosto temu ni tako. V slabše razvitih ali oddaljenih krajih, še posebej pa v primerih naravnih katastrof, uporaba javne mrežne infrastrukture ni mogoča. V takšnih primerih nam ostane le možnost, da omrežne povezave izdelamo sami.

Za vsak tip povezave bomo začeli s teoretičnim pregledom medija samega ter teoretičnih osnov, potrebnih za njegovo uporabo. Najprej bomo vzpostavili povezavo na enem računalniku, nato pa sprejemnik in oddajnik podvojili ter vzpostavili povezavo med dvema računalnikoma. Ko bomo povezavo izboljšali tako, da bo dovolj stabilna, bomo preko te povezave poskusili vzpostaviti ad-hoc IP omrežje.

Če bomo ugotovili, da so omrežne povezave dovolj hitre in stabilne za uporabo, bomo poskusili še več teh povezav združiti v distributirano *mesh* omrežje.

2. Osnove računalniških omrežij

Preden se lotimo raziskovanja specifičnih tehnik, ki jih bomo uporabili za vzpostavitev omrežji, si bomo ogledali kako, natančno, delujejo računalniška omrežja.

2.1. TCP/IP

Paket internetnega protokola je konceptualni model in niz komunikacijskih protokolov, ki se uporabljajo na internetu in podobnih računalniških omrežjih . Običajno je znan kot TCP / IP, saj so temeljni protokoli v paketu protokol za nadzor prenosa (TCP) in internetni protokol (IP).

Paket internetnega protokola zagotavlja podatkovno komunikacijo od konca do konca, ki določa, kako je treba podatke pakirati, naslavljati, prenašati, preusmeriti in prejeti. Ta funkcionalnost je organizirana v štiri abstrakcijske sloje, ki razvrstijo vse povezane protokole glede na obseg vključenega omrežja. Od najnižjega do najvišjega sloja so povezovalni sloj , ki vsebuje komunikacijske metode za podatke, ki ostanejo v enem segmentu omrežja (povezava); internetni sloj , ki zagotavlja povezovanje med neodvisnimi omrežji; transportni sloj, ki obravnava komunikacijo med gostitelji in gostitelji; in aplikacijski sloj , ki zagotavlja izmenjavo podatkov med procesi v procesih.

Tehnični standardi, ki opredeljujejo zbirko internetnih protokolov in mnoge njene protokole, vzdržujejo Internet Engineering Task Force (IETF). Paket internetnega protokola je pred modelom OSI , bolj celovit referenčni okvir za splošne omrežne sisteme.

TCP/IP (ang. TCP:"Transmission Control Protocol", IP:"Internet Protocol") ali internetni sklad protokolov je množica protokolov prek katerih "teče" internet. Ta množica ali nabor internetnih protokolov se deli na: aplikacijsko plast, prenosno plast, omrežna plast ter povezovalna plast.

Povezovalni sloj ima obseg omrežja lokalne omrežne povezave, na katero je pritrjen gostitelj. Ta režim se imenuje povezava v literaturi TCP / IP. To je najnižja komponentna plast internetnih protokolov, saj je TCP / IP zasnovan kot neodvisen od strojne opreme. Posledično se lahko TCP / IP izvaja poleg skoraj vsake strojne tehnologije omrežja. Povezovalni sloj se uporablja za premikanje paketov med vmesniki internetne plasti dveh različnih gostiteljev na isti povezavi. Procesi pošiljanja in sprejemanja paketov na določeni povezavi se lahko nadzorujejo tako v gonilniku programske opreme za omrežno kartico kot tudi na vdelani ali specializirani čipnici . Ti opravljajo funkcije podatkovne povezave , kot je dodajanje glave paketa za pripravo za prenos, nato pa dejansko prenašajo okvir preko fizičnega medija . Model TCP / IP vključuje specifikacije prevajanja metod za mrežno naslavljanje, ki se uporabljajo v internetnem protokolu, za povezavo naslovov na ravni, kot so naslovi za nadzor dostopa do medijev (MAC). Vse druge vidike pod to raven pa implicitno domnevajo, da obstajajo v povezavi, vendar niso izrecno opredeljene. To je tudi sloj, v katerem so lahko izbrani paketi poslani prek navideznega zasebnega omrežja ali drugega mrežnega tunela . V tem scenariju se lahko podatki o povezovalni plasti obravnavajo kot podatki o aplikacijah, ki prehajajo na drugo instantiacijo IP-ja za prenos ali sprejem prek druge IP-povezave. Takšna povezava ali navidezna povezava se lahko vzpostavi s transportnim protokolom ali celo s protokolom za področje uporabe aplikacij, ki služi kot predor v povezovalni plasti protokola.

Tako model TCP / IP narekuje strogo hierarhično zaporedje zapiranja. Povezovalni sloj modela TCP / IP ustreza fizičnim slojem podatkovnih povezav in podatkovnim povezavam medsebojnega povezovanja odprtih sistemov (OSI), slojev enega in dveh OSI modelov.

Spletni sloj je odgovoren za pošiljanje paketov v potencialno več omrežij. Medmrežje zahteva pošiljanje podatkov iz izvornega omrežja v ciljno omrežje. Ta proces se imenuje usmerjanje . Internetni protokol izvaja dve osnovni funkciji:

- Naslov in identifikacija gostitelja: To se doseže s hierarhičnim sistemom za naslove IP
- Paketno usmerjanje: To je osnovna naloga pošiljanja paketov podatkov (datagramov) od vira do cilja, tako da jih posredujete na naslednji usmerjevalnik omrežja bližje končnemu cilju.

Internetni sloj ni le agnostik podatkovnih struktur na transportni plasti, temveč tudi ne razlikuje med delovanjem različnih protokolov prometne plasti. IP prenaša podatke za različne protokole zgornjega sloja . Ti protokoli se identificirajo z edinstveno številko protokola : na primer protokoli 1 in 2 sta na primer protokol protokola internetnega nadzora (ICMP) in protokol za upravljanje internetnih skupin (IGMP). Nekateri protokoli, ki jih prenaša IP, kot je ICMP, ki se uporablja za posredovanje diagnostičnih informacij, in IGMP, ki se uporabljajo za upravljanje podatkov IP Multicast , so večplastno nad IP-jem , vendar opravljajo interne funkcije. To ponazarja razlike v arhitekturi stack TCP / IP interneta in OSI modela. Internetni sloj modela TCP / IP ustreza tretjemu sloju modela odprtih sistemov (OSI), kjer se imenuje omrežna plast. Spletni sloj zagotavlja nezanesljiv prenos podatkovnega datagrama med gostitelji, ki se nahajajo v potencialno različnih omrežjih IP, tako da posredujejo datagrame prometnega sloja ustreznemu usmerjevalniku naslednjega omrežja za nadaljnjo posredovanje do cilja. S to funkcionalnostjo internetni sloj omogoča internetno povezovanje, medsebojno delovanje različnih omrežij IP in v bistvu vzpostavlja internet. Internetni protokol je glavna komponenta internetne plasti in definira dva sistema za naslavljanje, ki prepoznata računalnike omrežnih gostiteljev, in jih najdejo v omrežju. Prvotni naslovni sistem ARPANET-a in njegovega naslednika, internet, je različica 4 protokola Internet (IPv4). Uporablja 32-bitni IP-naslov in zato lahko identificira približno štiri milijarde gostiteljev. To omejitev je leta 1998 odpravila standardizacija internetnega protokola različice 6 (IPv6), ki uporablja 128-bitne naslove. Implementacija proizvodnje IPv6 se je pojavila približno v letu 2006.

Transportna plast vzpostavlja osnovne podatkovne kanale, ki jih aplikacije uporabljajo za izmenjavo podatkov, specifičnih za posamezne naloge. Ta sloj vzpostavlja povezljivost med postopki, kar pomeni, da zagotavlja storitve od konca do konca, ki so neodvisne od strukture uporabniških podatkov in logistike izmenjave informacij za določen namen. Njena odgovornost vključuje prenos od konca do konca, neodvisen od osnovnega omrežja, skupaj z nadzorom napak , segmentacijo , nadzorom pretoka, nadzorom prezasedenosti in adresiranjem aplikacij (številke vrat). Prenos sporočil od konca do konca ali povezovanje aplikacij na prometni plasti je mogoče razvrstiti bodisi v povezavo usmerjeno , izvajati v TCP ali brez povezave , ki se izvaja v UDP. Za zagotavljanje procesnih kanalov za prenos za aplikacije, plast določa koncept pristanišča . To je oštevilčen logični konstrukt, dodeljen posebej za vsak komunikacijski kanal, ki ga potrebuje aplikacija. Za številne vrste storitev so številke teh vrat

standardizirane, tako da lahko odjemalski računalniki naslovijo specifične storitve strežniškega računalnika brez vključevanja obvestil o storitvah ali imeniških storitev. Ker IP zagotavlja le najboljše napore , nekateri protokoli transportne plasti nudijo zanesljivost. Vendar lahko IP preide preko zanesljivega podatkovnega protokola, kot je nadzor visoke ločljivosti podatkovne povezave (HDLC). TCP je na primer povezovalno usmerjen protokol, ki obravnava številne težave z zanesljivostjo pri zagotavljanju zanesljivega toka bajtov :

- podatki prispejo po naročilu
- podatki imajo minimalno napako (tj. pravilnost)
- podvojeni podatki se zavržejo
- izgubljeni ali zavrženi paketi
- vključuje nadzor prometa

Novi protokol za nadzor prenosa toka (SCTP) je tudi zanesljiv prometni mehanizem, usmerjen v povezavo. Je usmerjen v sporočilni tok, ki ni usmerjen na byte-stream, kot je TCP, in omogoča večkratno prevajanje več tokov preko ene same povezave. Zagotavlja tudi podporo za več homingov , pri kateri lahko končni priključek predstavlja več naslovov IP (ki predstavljajo več fizičnih vmesnikov), tako da se povezava ne prekine, če ena ne uspe. Prvotno je bil razvit za telefonske aplikacije (za prenos SS7 preko IP-ja), vendar se lahko uporablja tudi za druge aplikacije. Protokol User Datagram je protokol brez povezovalnega podatkovnega protokola. Tako kot IP je to najboljši napor, "nezanesljiv" protokol. Zanesljivost se obravnava z odkrivanjem napak z uporabo šibkega kontrolnega algoritma. UDP se običajno uporablja za aplikacije, kot so pretočni mediji (avdio, video, glasovno prek IP itd.), Pri katerih je časovni prihod pomembnejši od zanesljivosti ali za preproste aplikacije poizvedbe / odziva, kot so pregledi DNS , zanesljiva povezava je nesorazmerno velika. Protokol prenosa v realnem času (RTP) je protokol datagrama, ki je zasnovan za podatke v realnem času, kot so pretakanje zvoka in videa. Aplikacije na katerem koli omrežnem naslovu odlikujejo njihova vrata TCP ali UDP. Po dogovoru so nekatera znana pristanišča povezana s specifičnimi aplikacijami. Model transporta TCP / IP ali gostitelj-gostitelj ustreza četrti plast v modelu za povezovanje odprtih sistemov (OSI), ki se imenuje tudi transportna plast.

Aplikacijska plast vključuje protokole, ki jih uporablja večina aplikacij za zagotavljanje uporabniških storitev ali izmenjavo podatkov aplikacij prek omrežnih povezav, ki jih določijo protokoli nižjega nivoja. To lahko vključuje nekatere osnovne omrežne podporne storitve, kot so protokoli za usmerjanje in konfiguracijo gostitelja. Primeri protokolov aplikacijske plasti vključujejo protokol prenosa za hipertekst (HTTP), protokol prenosa datotek (FTP), protokol preprostega prenosa po pošti (SMTP) in protokol dinamičnega gostiteljskega konfiguracije (DHCP). [26] Podatki, kodirani po protokolih aplikacijske plasti, so vdelani v enote protokola transportne plasti (na primer sporočila TCP ali UDP), ki nato uporabijo protokole nižjega nivoja za dejansko prenos podatkov. Model TCP / IP ne upošteva specifičnosti oblikovanja in predstavljanja podatkov ter ne določa dodatnih plasti med aplikacijo in transportnimi sloji, kot je v modelu OSI (predstavitveni in sejni sloji). Takšne funkcije so področje knjižnic in aplikacijskih programskih vmesnikov . Protokoli aplikacijskega sloja običajno obravnavajo protokole transportne plasti (in nižje) kot črne škatle, ki zagotavljajo stabilno omrežno

povezavo, prek katere se komunicirajo, čeprav se aplikacije običajno zavedajo ključnih lastnosti povezave prometnega sloja, kot so naslov IP in vrata številke. Protokoli aplikacijskega sloja so pogosto povezani s posameznimi aplikacijami odjemalec-strežnik , skupne storitve pa imajo znane številke vrat, ki jih rezervira pooblaščena internetna pooblastila (IANA). Na primer, protokol prenosa za HyperText uporablja vrata strežnika 80 in Telnet uporablja vrata strežnika 23. Odjemalci, ki se povezujejo s storitvijo, običajno uporabljajo krajša vrata , to so številke vrat, dodeljene samo za čas trajanja transakcije naključno ali iz določenega obsega, nastavljenega v aplikacija. Transportni sloj in sloji nižjega nivoja se ne zanimajo s posebnostmi protokolov aplikacijskih slojev. Usmerjevalniki in stikala običajno ne preučujejo zaprtega prometa, temveč samo zagotovijo kanal za to. Vendar pa morajo nekateri požarni zidovi in aplikacije za zmanjševanje pasovne širine razlagati uporabniške podatke. Primer je Protokol Reservation Reservation (RSVP). Prav tako je včasih potrebno, da prestopnik omrežnega naslova (NAT) preuči uporabniško korist. Aplikacijska plast v modelu TCP / IP se pogosto primerja kot enakovredna kombinaciji pete (seje), šeste (predstavitve) in sedmega (aplikacijskega) sloja modela OSI (Open Systems Interconnection - OSI). Poleg tega referenčni model TCP / IP razlikuje med uporabniškimi protokoli in podpornimi protokoli . Podporni protokoli zagotavljajo storitve sistemu. Uporabniški protokoli se uporabljajo za dejanske uporabniške aplikacije. FTP je na primer uporabniški protokol in DNS je podporni protokol.

Paket internetnega protokola ne predvideva nobenega posebnega strojnega ali programskega okolja. Zahteva samo, da obstaja strojna in programska plast, ki lahko pošilja in sprejema pakete v računalniškem omrežju. Rezultat je, da je programski paket implementiran na v bistvu vsaka računalniška platforma. Minimalno izvajanje TCP / IP vključuje naslednje: Internet Protocol (IP), Address Resolution Protocol (ARP), Internet Protocol Control Protocol (ICMP), TCP, UDP in Internet Group Management Protokol (IGMP). Poleg IP-ja, ICMP, TCP, UDP, Internet Protocol različica 6 zahteva Protokol za odkrivanje sosveta (NDP), ICMPv6 in IGMPv6 in ga pogosto spremlja integrirana varnostna plast IPSec .

Aplikacijski programerji se običajno ukvarjajo samo z vmesniki v aplikacijski plasti in pogosto tudi v transportni plasti, medtem ko so spodnji sloji storitve, ki jih ponuja strežnik TCP / IP v operacijskem sistemu. Večina implementacij IP je dostopna programerjem preko vtičnic in API-jev .

Edinstvene izvedbe vključujejo lahki TCP / IP , odprtokodni sistem , zasnovan za vgrajene sisteme , in KA9Q NOS , zlog in pripadajoče protokole za amaterske paketne radijske sisteme in osebne računalnike, ki so povezani prek serijskih linij.

Vgrajena programska oprema mikrokrmilnika v omrežnem vmesniku običajno obravnava vprašanja povezav, ki jih podpira gonilnik programske opreme v operacijskem sistemu. Neprogramabilna analogna in digitalna elektronika sta navadno zadolžena za fizične komponente pod povezavo, običajno z uporabo vezij za integrirano vezje (ASIC) za vsak posamezni omrežni vmesnik ali drug fizični standard. Visoko zmogljivi usmerjevalniki v veliki meri temeljijo na hitro neprogramabilni digitalni elektroniki, ki izvaja preklapljanje ravni povezave.

2.2. Usmerjanje

Usmerjanje (ang. routing) je proces izbire poti za promet v omrežju ali med ali v več omrežjih. Routing se izvaja za številne vrste omrežij, vključno z omrežji z vklopljenimi vezji , kot so javno komutirano telefonsko omrežje (PSTN), računalniška omrežja , kot je internet .

V omrežjih za paketno preklapljanje je usmerjanje odločitev na višji ravni, ki usmerja omrežne pakete od njihovega vira do cilja preko vmesnih omrežnih vozlov s posebnimi mehanizmi za posredovanje paketov. Posredovanje paketov je tranzit logično usmerjenih omrežnih paketov iz enega omrežnega vmesnika na drugega. Vmesna vozlišča so običajno omrežne strojne naprave, kot so usmerjevalniki, mostovi, prehodi, požarni zidovi ali stikala. Računalniki splošnega namena posredujejo tudi pakete in opravljajo usmerjanje, čeprav nimajo posebej optimirane strojne opreme za to nalogo. Proces usmerjanja običajno usmerja posredovanje na podlagi usmerjevalnih tabel, ki vodijo evidenco poti do različnih omrežnih destinacij. Tako je izdelava usmerjevalnih tabel, ki so v pomnilniku usmerjevalnika, zelo pomembna za učinkovito usmerjanje. Večina usmerjevalnih algoritmov hkrati uporablja samo eno omrežno pot. Tehnike večsmernega usmerjanja omogočajo uporabo več alternativnih poti.

Routing v ožjem pomenu izraza je pogosto v nasprotju s premostitvijo v predpostavki, da so omrežni naslovi strukturirani in da podobni naslovi pomenijo bližino v omrežju. Strukturirani naslovi omogočajo, da en sam usmerjevalni vnos predstavlja pot do skupine naprav. V velikih omrežjih strukturirano naslavljanje (usmerjanje v ozkem smislu) presega nestrukturirano naslavljanje (premostitev). Usmerjanje je postalo prevladujoča oblika naslavljanja na internetu. Premostitev se še vedno pogosto uporablja v lokaliziranih okoljih.

Načrti za usmerjanje se razlikujejo glede na to, kako dostavljajo sporočila:

- unicast prinaša sporočilo enemu posameznemu vozlišču
- oddaja prenaša sporočilo vsem vozliščem v omrežju
- multicast pošlje sporočilo skupini vozlišč, ki so izrazila zanimanje za prejem sporočila
- anycast prenaša sporočilo kateremu koli iz skupine vozlišč, običajno najbližje vira
- geocast pošilja sporočilo geografskemu območju

Unicast je prevladujoča oblika dostave sporočil na internetu. Ta članek se osredotoča na algoritme za unicast usmerjanje.

2.3. Ad-hoc omrežja

Ad hoc je latinski izraz, ki pomeni “za to”. Le ta dosti krat opisuje rešitev za točno določen problem ali nalogo, katera se ne more prenašat na druge probleme ali naloge. Isti koncept se lahko uporabi pri računalniških omrežjih. Ad hoc omrežje se ne zanaša na prej obstoječo infrastrukturo kot so usmerjevalniki temveč nastane iz več Point-to-point povezav kjer vsaka

naprava sodeluje v prenosu podatkov za druge naprave. Izbira katere naprave bodo sodelovale pri prenosu podatkov nastane dimačno z pomočjo usmerjevalnih algoritmov. Ad hoc omrežja so najpogosteje brezžična.

Načini usmerjanja v brezžičnih Ad hoc omrežjih se delijo v tri kategorije:

- “Proactive routing”:

Ta kategorija protokolov uporablja seznam naprav in pot do njih tako da periodično distribuira t.i. “usmerjevalne tabele (ang. routing tables)” po omrežju. Glavne težave takšnih protokolov so: velika količina podatkov za vzdrževanje ter počasna “reakcija” na možne izpade (npr.: Dva računalnika sta povezana preko večih “vozlišč” nakar eno od teh izgine z mreže. Težava je v tem da bosta napravi, ki sta povezani rabila kar nekaj časa da ponovno vzpostavita povezavo saj se mora usmerjevalna tabela prvo ustvariti nato pa še poslat vsem napravam).

- “Reactive routing”:

Protokoli, ki spadajo v to kategorijo iščejo pot do sprejemnika tako da “poplavijo omrežje (ang. flooding the network)” z paketi za iskanje poti. Glavni težavi z takšnimi protokoli sta: čas iskanja poti je zelo dolg ter da odvečno “zalivanje” lahko vodi do zamašitve omrežja.

- “Hybrid routing”:

Hibridni tipi kombinirajo prednosti “proactive” ter “reactive” usmerjanja tako da v začetku uporabijo proaktivno iskanje poti nato pa ob izključitvi ali priključitvi naprav uporabijo reaktivno iskanje poti najpogosteje z “poplavljanjem”.

4. PRENOS PREKO RADIJSKIH POSTAJ

Ena izmed najosnovnejših modernih metod komunikacije na daljavo je prenos zvoka preko radijskih valov. Kljub hitremu razvoju boljših tehnologij, so prenosne radijske postaje še vedno izjemno pogoste. Njihova enostavnost, priročnost in nenaveznost na kakršno koli infrastrukturo jih naredi perfektne za uporabo v primerih, ko do te nimamo dostopa (naravne nesreče, izpadi elektrike…). Zaradi njihovega relativno enostavnega delovanja so tudi zelo poceni, za silo pa jih je možno tudi izdelati iz nekaj standardnih komponent.

4.1. Teorija

4.1.1. FM radio

FM oddajanje je metoda radijske radiodifuzije s tehnologijo frekvenčne modulacije (FM). Izumil leta 1933, ki ga je izdelal ameriški inženir Edwin Armstrong , se uporablja po vsem svetu, da zagotovi visoko zvestobran zvok prek radiodifuznega radia . FM oddajanje je sposobno boljše kakovosti zvoka kot AM oddajanje , glavna konkurenčna radijska radiodifuzna tehnologija, zato se uporablja za večino glasbenih oddaj. FM radijske postaje uporabljajo frekvence VHF . Izraz "FM-band" opisuje frekvenčni pas v določeni državi, ki je posvečen FM-oddajanju.

Frekvenčna modulacija ali FM je oblika modulacije, ki prenaša informacije tako, da spreminja frekvenco nosilnega vala ; starejša amplitudna modulacija ali AM spreminja amplitudo nosilca, njegova frekvenca pa ostane konstantna. Pri FM je frekvenčni odklon od dodeljene nosilne frekvence v vsakem trenutku neposredno sorazmeren z amplitudo vhodnega signala, ki določa trenutno frekvenco oddanega signala. Ker poslani FM signali uporabljajo več pasovne širine kot AM signali, se ta oblika modulacije običajno uporablja z višjimi (VHF ali UHF) frekvencami, ki jih uporabljajo TV , oddajni pas FM in zemeljski mobilni radijski sistemi .

FM radijski valovi ne potujejo daleč preko vidnega obzorja , zato so razdalje sprejema FM postaj običajno omejene na 48,3-64,4 km. Hribi jih lahko blokirajo. To je manj kot obseg radijskih valov AM, ki zaradi svoje nižje frekvence lahko potujejo kot zemeljski valovi ali odsevajo ionosfero , tako da lahko AM radijske postaje sprejemamo na stotine (včasih na tisoče) milj. To je značilnost tipične frekvence (in moči) nosilnega valja, ne njegov modulacijski način. Območje mono FM prenosa je povezano z RF močjo oddajnika , z višino antene in višino antene . US FCC objavlja krivulje, ki pomagajo pri izračunu te največje razdalje kot funkcije moči signala na sprejemnem mestu. Mnoge FM postaje, še posebej tiste na hudih večpotemnih območjih, uporabljajo dodatno zvočno stiskanje, da ohranijo bistven zvok nad šumom v ozadju za poslušalce, občasno na račun celotne zaznane kakovosti zvoka. V takih primerih pa je ta tehnika pogosto presenetljivo učinkovita pri povečanju uporabnega območja postaje.

4.1.2. Packet radio

Paketna radijska postaja je oblika tehnologije za paketno preklapljanje, ki se uporablja za prenos digitalnih podatkov prek brezžičnih komunikacij. Paketni radio uporablja iste koncepte prenosa podatkov z uporabo datagramov, ki so temeljnega pomena za komunikacijo na internetu , v nasprotju s starejšimi tehnikami, ki jih uporabljajo namenski ali komutirani krogi. Paketni radio lahko uporabljate na daljših razdaljah brez potrebe po fizični povezavi med postajami. Paketna radijska postaja se lahko uporablja tudi za mobilne komunikacije.

Paketni radio je način digitalne radijske zveze. Predhodni digitalni načini so bili telegrafija (Morseova koda), teleprinter (koda Baudot) in faksimile. Vsaka paketna postaja lahko deluje kot digipeater , ki povezuje oddaljene postaje med seboj prek ad hoc omrežij . Zaradi tega je paketni radio še posebej uporaben za komunikacijo v sili. Mobilne paketne radijske postaje lahko samodejno prenesejo svojo lokacijo in se redno preverjajo z omrežjem, da pokažejo, da še vedno delujejo. Najpogostejša uporaba paketnega radia je v amaterskem radiju, za izgradnjo brezžičnih računalniških omrežij. Paketni radio uporablja protokol za podatkovno povezavo AX.25 (Amateur X.25), izpeljan iz paketa protokolov X.25 in prilagojen za radioamatersko radijsko uporabo. AX.25 je bil razvit v sedemdesetih letih. AX.25 vključuje polje digipeater, ki drugim postajam omogoča samodejno ponovitev paketov za razširitev obsega oddajnikov. Edina prednost je, da vsak poslan paket vsebuje oddajnik in prejemnika radijski pozivni znak, s čimer zagotavlja identifikacijo postaj z vsakim prenosom.4.1.3. AX.25

AX.25 je protokol v omrežni plasti, ki je osnovan na X.25 protokolih in je v uporabi z strani radio amaterjev v paketno radijskijh omrežjih. kot je bilo že prej povedano, deluje AX.25 na omrežni plasti kar pomeni, da je odgovorno za vzpostavitev povezave, prenos podatkov ter zaznavanje napak v podatkih. Le ta pa se dosti krat uporablja za prenos drugih omrežnih plasti v “point-to-point” povezavah med paketnimi radijih. Najpogosteje uporabljen omrežni protokol v paketno radijskih omrežjih je IPv4, ki je uporabljen v tako imenovanem “AMPRNet” (ang.: Amateur Packet Radio Network) včasih tudi poimenovan “Network 44” saj so IP adrese v tem bloku v 44.0.0.0/8. Zaradi omejitev pasovne širivne v radijskih valovih, VHF in UHF pozevave so povprečno 1200 baud, in je mnogokrat omejen na 9600 baud. AMPRTNet popolnoma podpira TCP/IP kar pomeni da omrežje podpira vse omrežne protokole.

Za vpostavitev teh omrežji radijo amaterji uporabljajo naprave kot so “TNC” (ang.: Terminal Node Contrroller). Funkcionalno je zelo podoben “sestavljalcu/razstavljalcu” paketov v X.25 omrežjih z adicijo modema za pretvorbo digitalnega signala v zvočne tone.

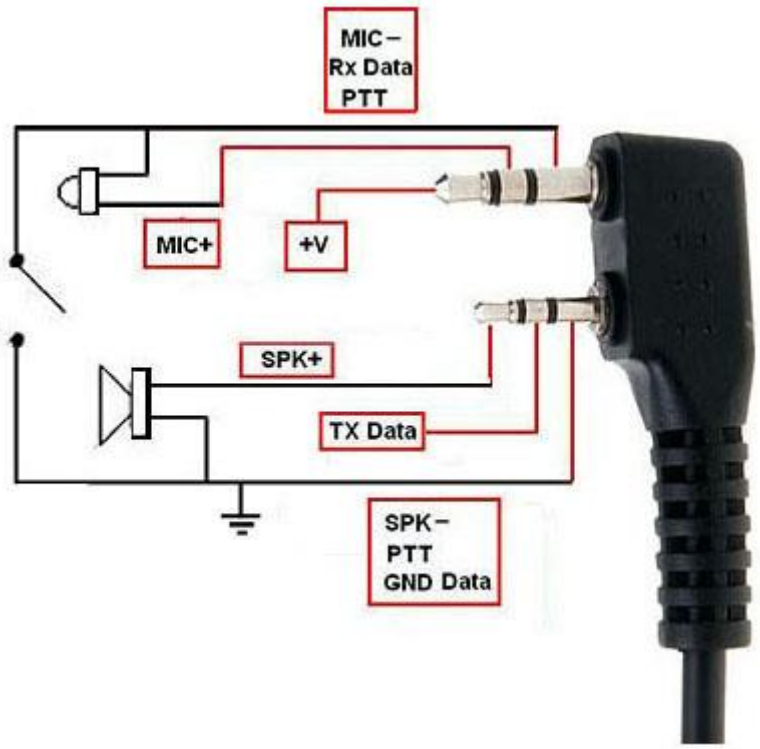
4.2. Povezava na računalnik

Tradicionalno, so bile za AX.25 povezavo potrebne posebne naprave, t.i. *TNC (Transmit Node Controller)* modemi. Te so bile v začetku izdelane kar iz starih telefonskih modemov, pogosteje pa se uporabljajo namenski TNC modemi, ki implementirajo *KISS (Keep It Simple, Stupid)* ali, redkeje, *6PACK* komunikacijski protokol. Preko *KISS* protokola, ki deluje po serijski povezavi, lahko računalnik komunicira s TNC napravo. Uporablja se za konfiguracijo, kot tudi za pošiljanje in prejemanje podatkov preko radijske postaje, priključene na TNC napravo.

Ker pa se AX.25 uporablja precej redko, so TNC naprave, tako kot večina opreme za radioamaterje, precej drage in redke. Na srečo, pa je za povezavo na AX.25 omrežje ne potrebujemo nujno. Z modernimi računalniki je namreč delovanje teh naprav možno emulirati kar v programski opremi. Več o temu namenjenih programih bomo govorili kasneje, najprej pa moramo radijsko postajo povezati na računalnik.

To storimo tako, da radijsko postajo priklopimo v zvočno kartico računalnika. Izhod za slušalke na postaji priključimo na *line-level* vhod naše zvočne kartice. Če takšnega vhoda nimamo, lahko uporabimo vhod za mikrofon le, da ga povežemo z 500ohm uporom, da ne poškodujemo zvočne kartice. Prav tako z 500ohm uporom povežemo še izhod za zvočnik računalnika na vhod za mikrofon postaje.

Za priključke na strani računalnika smo prerezali kabel starih slušalk, na strani radija pa smo uporabili kar kabel od priložene slušalke. Pomagali smo si tudi s spodnjim diagramom, ki prikazuje notranje povezave originalnih slušalk za radijsko postajo ter namen vseh povezav.



Slika 1: Povezava priključka radijske postaje[1]

Ko smo preko končanega kabla radijsko postajo povezali na zvočno kartico, smo lahko dogajanje na radijskih valovih le poslušali. Za oddajanje pa smo potrebovali način za aktivacijo PTT (*Push To Talk*) na radijski postaji.

Zaradi omejitev programske opreme, o kateri bomo govorili kasneje, smo imeli na voljo le tri načine aktivacije oddajanja: preko DTS priključkov serijske povezave, preko GPIO priključkov sistema in z uporabo glasovne aktivacije (VOX). Prva opcija ni prišla v poštev, saj naši računalniki nimajo RS-232 priključkov, USB adapterji pa DTS funkcije ne podpirajo.

[1] http://www.miklor.com/COM/UV_Technical.php

Najboljša opcija bi bila uporaba GPIO priključkov, ker pa jih standardni računalniki nimajo, smo potrebovali razvojno ploščo kot je Raspberry Pi. V ta namen smo naročili nekaj plošč Orange Pi (podobne kot Raspberry Pi, le cenejše, odprtokodne in v večih variacijah). Do konca roka za oddajo naloge te, zaradi težav pri pošiljanju s Kitajske, še niso prispele.

Ostane nam torej najslabša metoda – uporaba vgrajene VOX funkcije naših radijskih postaj. V tem načinu radijska postaja samodejno začne oddajati ko sprejme signal od mikrofona (v našem primeru, računalnika). VOX vezje, pa je bilo narejeno in optimizirano za zaznavanje govora, zato se odziva le na frekvence blizu frekvenc človeškega govora, hkrati pa ima opazen zamik pri začetku oddajanju ter še hujši zamik pred koncem.

V praksi bi bila uporaba vgrajenega VOX vezja za prenos podatkov neuporabno počasna, ker pa je to le začasna rešitev, dokler ne dobimo razvojnih plošč, smo jo lahko za silo uporabili.

3.3. Programska oprema

Med programskimi TNC modemi sta najbolj znana dva: Dire Wolf[2] in Soundmodem[3]. Zaradi priloženega grafičnega vmesnika, ki naredi konfiguracijo dosti lažjo, smo se odločili za Soundmodem.

S konfiguracijskim orodjem soundmodemconfig smo ustvarili konfiguracijsko datoteko. Za zvočni gonilnik smo določili standardni Linux gonilnik ALSA ter zanj izbrali ustrezno napravo (našo zvočno kartico). Ker uporabljamo le en kanal in naši rediji ne morejo istočasno oddajati in sprejemati, smo odkljukali opcijo *Half Duplex*. Kerše vedno nismo imeli načina kontrole PTT smo to v programu izklopili in le nastavili zelo dolg zamik pred začetkom pošiljanja podatkov. Tako bo Soundmodem za določen čas pred začetkom oddajanja predvajal glasen ton, ki bo aktiviral VOX funkcijo radija.

Tej konfiguraciji smo dodali le en kanal. Nastavitve modulacije in demodulacije smo pustili privzete za AX.25, pod zavihkom *Packet IO* pa smo način nastavili na *MKISS*, kar nam je omogočilo vpis IP naslova in drugih nastavitev AX.25 povezave.

Ko smo radio vklopli in nastavili na frekvenco z AX.25 prometom ter pognali program Soundmodem smo takoj začeli prejemati podatke. Tukaj pa se je praktičen del preizkusa tudi zaključil. Za oddajanje namreč potrebujemo radioamatersko licenco, ki pa je v času pisanja nobeden od nas še ni imel.

---

[2] https://github.com/wb2osz/direwolf

[3] http://gna.org/projects/soundmodem

4. PRENOS PREKO LASERJEV

Za razliko od radijskih povezav, ki smo jih uporabili v prejšnjem poglavju, za prenos digitalnih podatkov preko laserjev ni standardizirane programske in strojne opreme, ki bi jo lahko uporabili. Raziskali smo dva načina povezave - prvega v obliki surovega toka bitov med dvema mikrokrmilnikoma, drugega pa z uporabo protokola RS-233.

4.1. Teorija

4.1.2. Laserske diode

Polprevodniški laser (tudi diodni laser ali laserska dioda) je laser, pri katerem je ojačevalno sredstvo polprevodnik. Po delovanju je zelo podoben svetleči diodi, vendar oddajanje svetlobe pri svetleči diodi temelji na spontanem sevanju, pri laserski diodi pa na stimuliranem sevanju. Zato oddaja laserska dioda koherentno lasersko svetlobo. Izbira materiala določa barvo izsevane svetlobe. Najpogostejše svetijo laserske diode v infrardečem območju, poznamo pa tudi take, ki oddajajo vidno ali ultravijolično svetlobo. Poznamo dve vrsti polprevodniških laserjev. Mejno sevajoči (edge emitting) in površinsko sevajoči (surface emitting) laserji. Mejno sevajoči laserji imajo laserski snop vzporeden z ravnino P-N spoja, ogledala pa so narejena z uporabo površinskih kristalov na obeh koncih. Površinsko sevajoči laserji sevajo laserski snop pravokotno glede na normalno ravnino kristala.

V osnovi potrebuje diodni laser za svoje delovanje tako imenovani p-n stik. Najpreprostejši tak stik je meja med dvema območjema istega polprevodnika, kjer je v enem območju presežek elektronov (N-tip), v drugem pa pomanjkanje elektronov oziroma presežek vrzeli (P-tip). Prosti nosilci električnega naboja se neurejeno gibajo in ob takem stiku vrzeli prehajajo iz področja P v področje N, prosti elektroni pa iz področja N v področje P. Ob trku vrzeli s prostim elektronom pride do rekombinacije in v nekaterih primerih pride do oddaje fotona in pri dovolj nizkih temperaturah ali dovolj močnem električnem toku tako lahko dobimo diodni laser. Vendar so laserji, ki temeljijo na tako imenovanem homospoju, niso optimalni.

4.1.3. Fotodiode

Navadno se za zaznavanje svetlobe uporabljajo fotoupori (t.j. upori, ki svojo upornost spremenijo glede na moč svetlobe). Njihova uporaba je zelo enostavna, imajo pa eno slabost, ki jih naredi neprimerne za tovrstno vezje - dolg odzivni čas. Povprečni fotoupori za prehod iz nižjega v višje stanje potrebujejo okoli 10ms, kar bi že samo po sebi omejilo največjo možno hitrost prenosa na praktično neuporabnih 100 bitov na sekundo, za prehod na nižje stanje pa nekateri potrebujejo tudi do 1s.

Za zaznavanje hitrih sprememb v svetlobi nam torej ostaneta le dve možnosti: fototranzistorji in fotodiode. Čeprav sta obe dovolj dobri za uporabo pri hitrostih, ki jih omogočajo plošče Arduino, se bomo osredotočili na fotodiode zaradi ene pomembne prednosti: fotodiode so skoraj identične LED diodam. To pomeni, da lahko namesto fotodiode uporabimo čisto navadno LED diodo, ki jih lahko najdemo povsod. Še en razlog za uporabo LED diod pa je njihov vgrajen barvni filter. Ker laserska dioda oddaja le eno valovno dolžino svetlobe, je

dober način da odstranimo motnje to, da sprejemnik omejimo le na tisto valovno dolžino. To je zelo zapleteno s fotodiodami in zahteva precej drage barvne filtre, LED diode pa imajo takšne filtre že vgrajene. Tako bo zelena LED dioda sprejemala le zeleno svetlobo, redeča le rdečo, itd. Če uporabimo enako barvo LED diode kot je barva laserja bo signal čistejši in močnejši, kot če bi uporabili navadno fotodiodo.

Fotodiode, kot tudi LED diode, lahko za zaznavanje svetlobe uporabimo v dveh konfiguracijah. V fotovoltaičnem načinu dioda sama proizvaja zelo nizko napetost, ki je odvisna od moči svetlobe. Če pa diodi dodamo napetost v zaporni smeri *(ang. "reverse bias"),* se njen odzivni čas še skrajša, kar pa dvigne tudi nivo šuma.

4.1.4. Operacijski ojačevalci

Operacijski ojačevalnik (pogosto op-amp ali opamp) je enosmerni ojačevalnik z visokim vzajemnim ojačenjem z diferencialnim vhodom in običajno enkratni izhod. V tej konfiguraciji op-amp proizvaja izhodni potencial (glede na vezje), ki je ponavadi stotisoče tisočkrat večji od potencialne razlike med vhodnimi terminali. Operacijski ojačevalci so izhajali iz analognih računalnikov , kjer so bili uporabljeni za izvajanje matematičnih operacij v mnogih linearnih, nelinearnih in frekvenčno odvisnih vezjih.

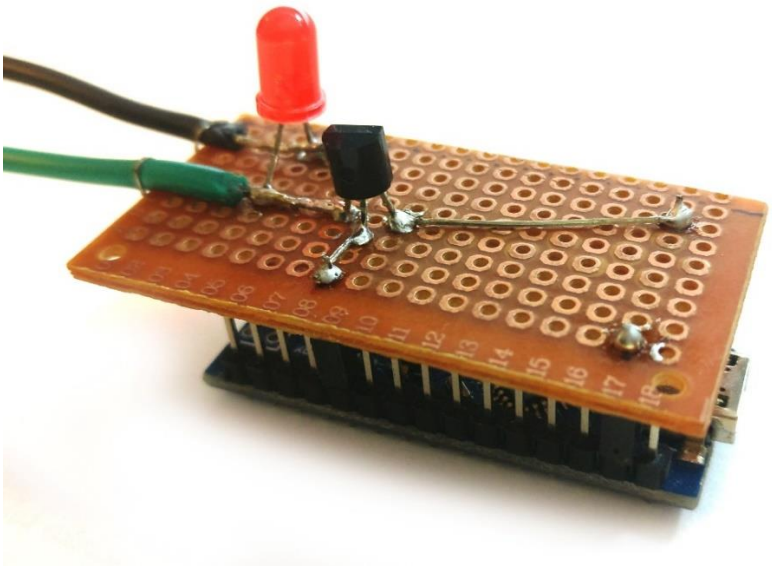
Op-amp so danes med najbolj razširjenimi elektronskimi napravami, ki se uporabljajo v širokem naboru potrošniških, industrijskih in znanstvenih naprav. Veliko standardnih IC op-ampov stane le nekaj centov v zmernem obsegu proizvodnje; vendar pa lahko nekateri integrirani ali hibridni operacijski ojačevalniki s posebnimi specifikacijami zmogljivosti v majhnih količinah stanejo več kot 100 USD. Op-amp se lahko pakirajo kot sestavni deli ali uporabljajo kot elementi bolj zapletenih integriranih vezij.

Primerjalnik Op-amp primerja en analogni nivo napetosti z drugo analogno napetostno stopnjo ali določeno prednastavljeno referenčno napetostjo, VREF, in proizvaja izhodni signal, ki temelji na tej primerjavi napetosti. Z drugimi besedami, op-amp napetostni primerjalnik primerja velikosti dveh napetostnih vhodov in ugotovi, katera je največja od obeh. Napetostni komparatorji na drugi strani uporabljajo bodisi pozitivno povratno povratno sporočilo ali povratne povratne informacije (način odprte zanke), da preklopijo njegovo izhodnost med dvema nasičenima stanjima, ker je v načinu odprte zanke ojačanje napetosti v bistvu enako AVO. Potem zaradi te visoke odprtosti z odprto zanko se izhod iz komparatorja popolnoma spremeni v svojo pozitivno oskrbo, + Vcc ali v celoti do negativne dovodne tirnice, -Vcc pri uporabi različnega vhodnega signala, ki prehaja nekaj prednastavljene mejne vrednosti. Komparator op-amp open-loop op-amp je analogno vezje, ki deluje v nelinearni regiji, saj spremembe v dveh analognih vhodih, V + in V- povzroča, da se obnaša kot digitalna bistabilna naprava, saj sprožitev povzroči, da ima dva možna izhoda države, + Vcc ali -Vcc. Nato lahko rečemo, da je napetostni primerjalnik v bistvu 1-bitni analogni digitalni pretvornik, saj je vhodni signal analogen, vendar se izhod digitalno obnaša.

4.2. Osnovna digitalna povezava

Za najosnovnejšo digitalno lasersko povezavo potrebujemo dva sprejemnika in dva oddajnika.

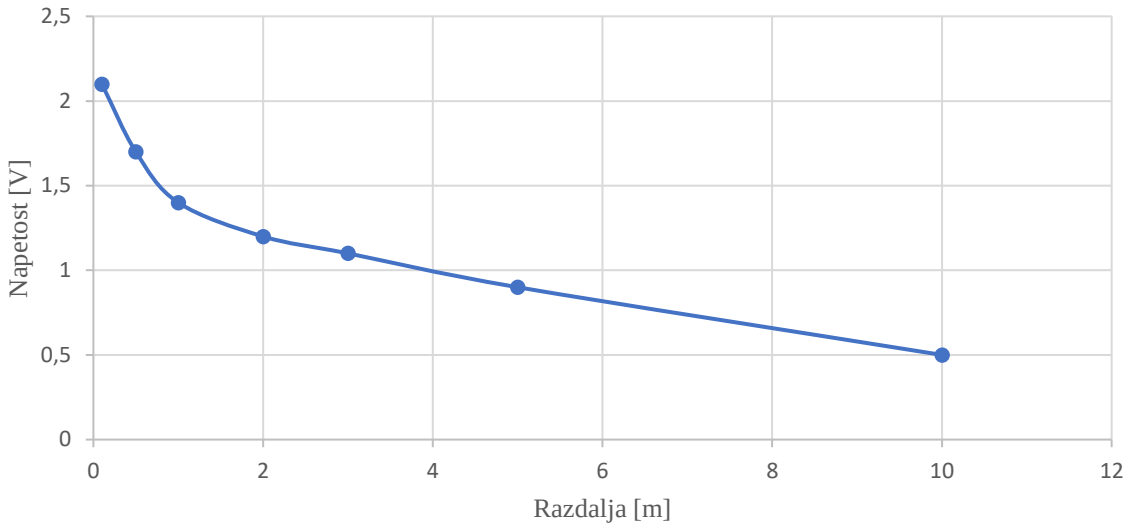
Oddajnik je precej enostaven. Zanj potrebujemo le lasersko diodo, en tranzistor, nek mikrokrmilnik ter seveda ustrezen vir napajanja. Za lasersko diodo lahko uporabimo čisto navaden laserski kazalnik, mikrokrmilniki (npr. Arduino) pa so postali tako pogosti, da jih lahko dobimo v praktično vsaki trgovini z elektroniko. Ker laserska dioda potrebuje dosti več toka, kot ga lahko dobimo na izhodnem pinu, jo preko tranzistorja povežemo na ustrezen napajalnik, bazo tranzistorja pa na izhodni priključek mikrokrmilnika. V našem primeru smo delali z laserskim kazalnikom, ki ga navadno napajata dve AAA bateriji. Ti imata skupno napetost 3V (2x 1.5V, saj sta 2 bateriji povezani zaporedno), kar je dovolj blizu 3,3V napajalnem priključku na plošči.



Slika 2: Oddajnik, nameščen na ploščo Arduino Nano (žice vodijo do laserja)

Sprejemnik pa je malo bolj zapleten. Ker nismo imeli na voljo dovolj natančnih merilnih naprav (npr. osciliskopa) nismo mogli izmeriti odzivnih časov, zato smo za vsak slučaj vezja načrtovali z diodo v *reverse bias* konfiguraciji. Laserska dioda je namreč dovolj močna, da dodaten šum še vedno ni dovolj za motenje delovanja.

Da smo lahko začeli načrtovati sprejemno vezje smo morali najprej opraviti nekaj meritev. Z multimetrom smo izmerili napetost na zeleni LED z zelenim laserjem na večih razdaljah in tako ugotovili razpon vrednosti ter največjo razdaljo, na kateri bi bil sprejemnik učinkovit.



Slika 3: Graf napetosti na fotodiodi v odvisnosti od razdalje od laserja

Ker nas zanima le, če je laser vklopljen ali izklopljen, moramo analogni signal fotodiode še pretvoriti v digitalnega. To lahko storimo v programski opremi tako, da povežemo analogni signal na enega izmed šestih ADC (*Analog-Digital Converter*) priključkov na našem mikrokrmilniku, izmerimo signal z laserjem vkloljenim in izklopljenim ter določimo prag, nad katerim je vrednost 1, pod pa 0.

Vgrajen analogni pretvornik pa ima zelo omejeno hitrost in potrebuje vsaj 1 milisekundo, da se stabilizira po meritvi. Zato moramo uporabiti digitalne vhode, ki imajo hitrost meritev omejeno le s hitrostjo procesorja mikrokrmilnika.

Večina mikrokrmilnikov, vključno z našimi, delujejo na standardnih logičnih nivojih 5V ali 3.3V za *HIGH* in 0V za *LOW*. Na srečo pa digitalni vhodi ne zahtevajo strogo 5V ali strogo 0V. Za visoko vrednost (*HIGH*) mikrokrmilnik šteje vse, kar je višje od napetosti *VIH*, za nizko vrednost (*LOW*) pa vse, nižje od napetosti *VIL*. Te vrednosti lahko najdemo v uradni dokumentaciji mikrokrmilnika, ki ga uporabljamo. Kot je razvidno iz spodnjega dela tabele, je zgornji prag nizke napetosti $VIL = 0{,}2 \times VCC - 0{,}1V = 0{,}2 \times 5V - 0{,}1V =$ ***0,9V***, spodnji prag visoke napetosti pa $VIH = 0{,}2 \times VCC + 0{,}9V = 0{,}2 \times 6V + 0{,}9V =$ ***1,9V***.

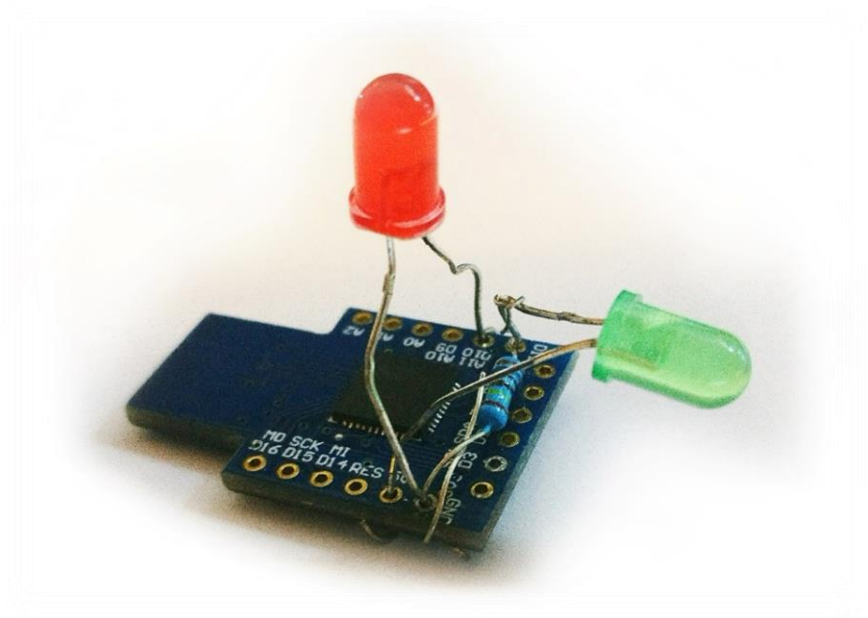
29.2 DC Characteristics

Table 29-1. DC Characteristic, TA = -40°C to 85°C, VCC = 2.7V to 5.5V (unless otherwise noted)

Symbol	Parameter	Condition	Min.(5)	Typ.	Max.(5)	Units
V_{IL}	Input Low Voltage, Except XTAL1 and Reset pin	V_{CC} = 2.7V - 5.5V	-0.5		$0.2V_{CC}$-0.1V(1) (LVTTL)	
V_{IH}	Input High Voltage, Except XTAL1 and RESET pins	V_{CC} = 2.7V - 5.5V	$0.2V_{CC}$+0.9V(2) (LVTTL)		V_{CC} + 0.5	V

Slika 4: Izvleček iz dokumentacije čipa 32U4 [4]

Iz prejšnjega grafa je razvidno, da je brez ojačenja branje napetosti neposredno s fotodiode z našim laserjem uporabno le na razdaljah, manjših od 5 metrov. Ker je to vezje zelo enostavno, smo ga zgradili kar neposredno na mikrokrmilniški plošči SS Micro, ki se lahko napaja neposredno iz USB priključka. Da smo odstranili šum, kije nastal, ko je bil laser izklopljen in je bil priključek v t.i. *floating* stanju, smo dodali še ozemljitveni upor, za lažjo kalibracijo pa smo dodali še LED diodo, ki se je vklopila, ko je bil signal dovolj močen.



Slika 5: Sprejemnik, nameščen na ploščo SS Micro (manjša različica Arduino Leonardo)

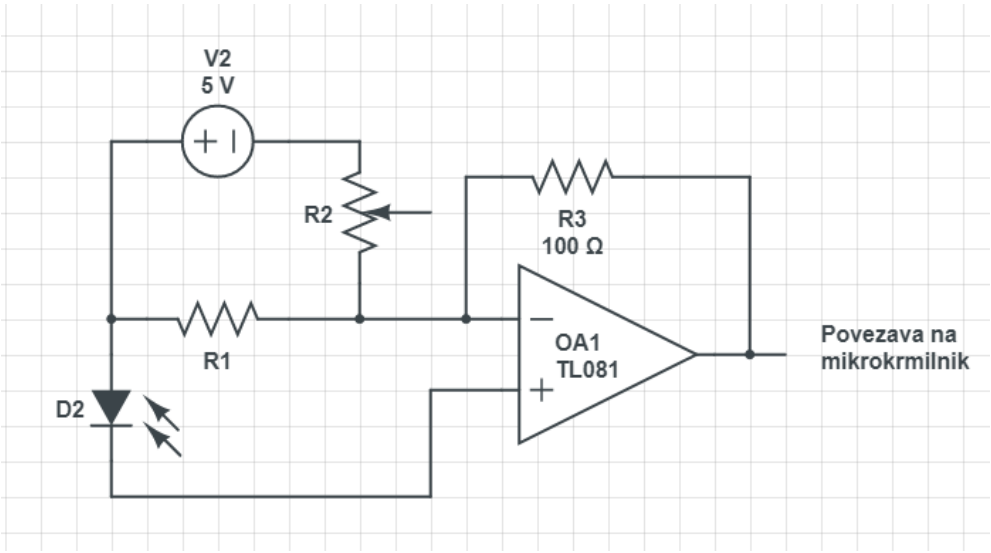
To je dovolj za nekaj preizkusov v nadzorovanem okolju, v praksi pa takšno vezje ne bi delovalo. Napetost bi z večjo razdaljo precej hitro padla pod minimalno V_{IH}, prav tako pa bi lahko zelo močna sončna svetloba dvignila napetost brez laserja nad zgornji prag V_{IL}.

Za rešitev tega problema smo načrtovali dva različna vezja. V obeh je glavna komponenta operacijski ojačevalec (ang. *op-amp*), v našem primeru v obliki integriranega vezja *TL081CN*.

---

[4] http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7766-8-bit-AVR-ATmega16U4-32U4_Datasheet.pdf (stran 383)

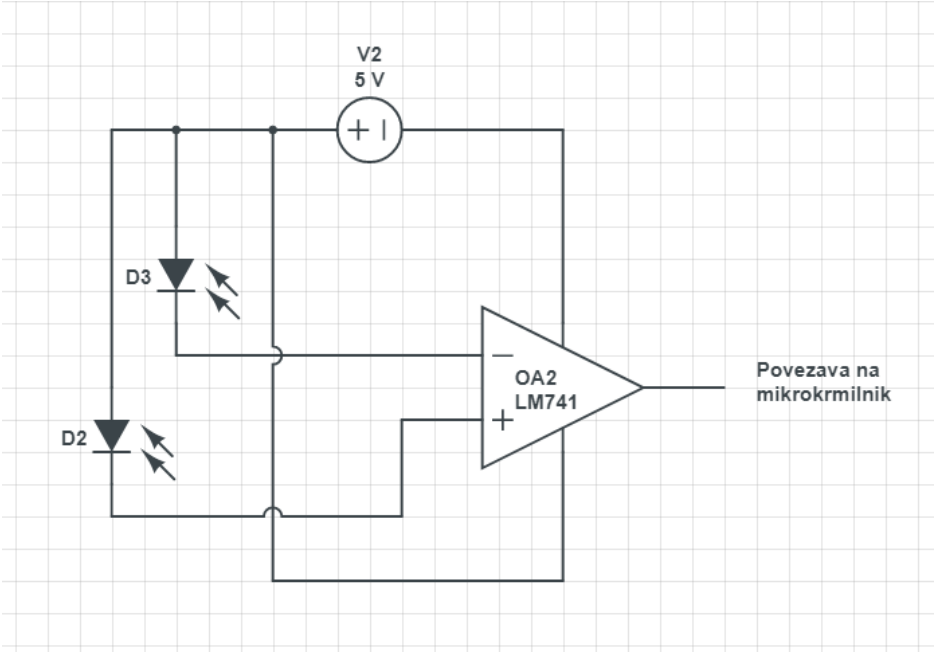
V prvem poskusu smo *op-amp* uporabili v standardni konfiguraciji kot ojačevalnik napetosti. To smo naredili s povezavo izhoda integriranega vezja nazaj na vhod preko upora. Po enačbi $V_O / V_I = A_V = 1 + R_2 / R_1$ (*Av* v tem primeru predstavlja faktor ojačanja) in z malo eksperimentiranja smo določili vrednosti uporov tako, da je ob vklopljenem laserju izhod vedno nad minimalno napetostjo V_{IH}. Ko smo izhod ojačevalnika povezali na digitalni vhod mikrokrmilnika, smo tako morali dobiti vrednost *HIGH*, ko je bil laser vklopljen in *LOW*, ko je bil izklopljen. Za lažje nastavljanje faktorja ojačevanja ne le med izdelavo ampak tudi med uporabo v primeru, da se nivo svetlobe preveč spremeni, smo namesto enega izmed uporov vstavili potenciometer.



Slika 6: Shema sprejemnika z nastavljivim pragom

Takšen pristop je sicer boljši od direktne povezave, še vedno pa ima svoje slabosti. Največje je definitivno to, da je v prag treba nastavljati ročno, prav tako pa je precej občutljiv na spremembe v svetlobi okolice. V enem poskusu smo v zelo močni sončni svetlobi povezavo začasno izgubili zaradi oblaka. To smo sicer rešili z improviziranim senčnikom nad fotodiodo, vseeno pa smo morali prag s potenciometrom občasno spreminjati.

Problem spreminjanja svetlobe v okolici bi bilo v teoriji možno rešiti z uporabo komparatorja in še ene fotodiode. Na referenčni vhod komparatorja bi povezali še eno, identično, fotodiodo, ki bi bila od glavne oddaljena dovolj, da žarek laserja na njo ne bi imel vpliva. Na glavni vhod bi priklopili glavno fotodiodo, na napajalne priključke pa bi priklopili logične napetosti *LOW* in *HIGH*. Ko bi laser posvetil na glavno fotodiodo, bi se vhodna napetost komparatorja dvignila na referenčno, zato bi komparator izhod dvignil na zgornjo napajalno napetost (5V). Izhod komparatorja bi priključili na digitalni vhod mikrokrmilnika ter tako dobili dosti bolj zanesljiv signal kot v prejšnjih primerih.



Slika 7: Shema sprejemnika z avtomatskim pragom (laser sveti na D2)

Tega vezja pa žal nismo uspeli preizkusiti, saj nismo imeli operacijskega ojačevalca z dostopnimi napajalnimi priključki.

4.3. Protokol

Do se daj smo uspeli med seboj preko laserskega žarka povezati dva mikrokrmilnika. Naslednji korak pa je, da preko te povezave prenesemo neke podatke.

Po dolgem iskanju smo presenečeni ugotovili, da na spletu nismo našli nobene programske knjižnice, ki bi omogočala serijski prenos podatkov preko enostavno digitalne povezave. Našli smo sicer kar nekaj knjižnic, ki implementirajo TTL Serial povezavo, a te delujejo le po pravi žici, saj TTL Serial sprejemnik ne le bere RX povezave, ampak jo najprej dvigne na 5V, nato pa čaka, da jo oddajnik zniža na GND (več o tem kasneje).

Zato nam ni preostalo drugega, kot da primeren serijski protokol razvijemo in implementiramo sami.

Če bi bil naš cilj razviti dober in zanesljiv protokol, bi načrtovanje začeli z raziskovanjem obstoječih protokolov kot so RS-232. Tak protokol bi najverjetneje podatke pošiljal v kosih (t.i. okvirjih, ang. frames), vsakega z ustrezno glavo in nogo. Imel bi mehanizme za zaznavanje in odpravo napak (paritetni biti in ukaz za zahtevo ponovnega pošiljanja), ohranjanje vrstnega reda, razlikovanje med kontrolnimi biti in podatki (ang. escape codes), (samodejno dogovarjanje za parametre povezave…

Preden pa se lotimo razvoja popolnoma novega, pravega, serijskega protokola, pa je najbolje, da začnemo s še bolj osnovno povezavo ter tako ugotovimo omejitve naše strojne opreme, preden porabimo ogromno časa na nečem, kar morda sploh ni možno.

Zato smo začeli z najbolj osnovno povezavo: prenos podatkov bit po bit, v eno smer, brez popravljanja napak.

Za lažje usmerjanje laserja smo se že na začetku odločili, da mora v nedejavnem stanju laser biti vklopljen. To pomeni, da moramo podatke kodirati v izklope laserja. Z uporabo začetnih in končnih bitov bi lahko za prenos uporabili vsako spremembo (povezavi bi bili sinhronizirani), ker pa delamo z asinhrono povezavo, nam ne preostane drugega kot, da podatke kodiramo v čas izklopa laserja.

Celotna izvorna koda programa, ki smo ga napisali, je na voljo v prilogi, tukaj pa bomo le površinsko razložili kako deluje.

Vsak procesorski cikel mikrokrmilnik naredi dve stvari: preveri, če je na serijskem vmesniku z računalnikom na voljo nov podatek ter preveri, če se je stanje laserskega sprejemnika spremenilo.

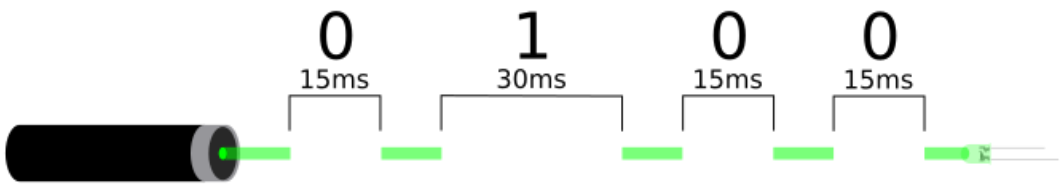
Če na serijskem vhodu čakajo novi podatki, mikrokrmilnik z njega prebere prvi bajt in ga shrani v pomnilniku. S pomočjo bitne maske prebere vsak bit tega bajta posebej, ter ga pošlje po laserju. Glede na to, ali je bit `1` ali `0`, se laser izklopi za določen čas (`TIME_0` in `TIME_1`). Po vsakem poslanem bitu, se laser ponovno vklopi vsaj za čas `TIME_BETWEEN`.

Po koncu oddajanja, pa preveri še, če so na voljo kakšni podatki za sprejem. Podatke sprejema s pomočjo 2 spremenjlivk: bajta, ki ga sproti sestavlja in bitne masko, ki določa pozicijo naslednjega bita. Ob vsaki spremembi stanja laserskega žarka, se pokliče sprejemna rutina.

V primeru padca nivoja (`1`→ `0`), rutina shrani trenuten čas izvajanja v spremenljivko.

V primeru dviga nivoja (`0`→ `1`), pa rutina prebere shranjen čas in izračuna, koliko časa je bila linija izklopljena. Glede na ta čas in glede na pragove, definirane v nekaj konstantah, se odloči, ali gre za `1` ali `0`. Če gre za `0`, bit preskoči, saj je bajt inicializiran na `00000000`. Če gre za `1`, pa s pomočjo bitne operacije OR bit, določen z bitno masko, spremeni v `1`.

Po zapisu, rutina preveri, če je prejet bit bil zadnji v bitu. Če ni, enostavno premakne bitno masko na naslednjo pozicijo in nadaljuje z izvajanjem normalno. Če pa je bil bit zadnji, rutina do konca sestavljen bajt pošlje po serijski povezavi na računalnik, nato pa ponastavi shranjen bajt in bitno masko ter nadaljuje z izvajanjem.



Slika 8: Prikaz prenosa bitov po laserju

Med razvojem smo uporabljali vrednosti 15 in 30 milisekund, kar je dovolj počasi, da lahko vidimo kaj se dogaja in lažje odpravljamo napake. Kmalu pa smo ugotovili, da dosti hitreje sploh ne gre. Pulzi pod 10 milisekund so že bili nestabilni, kar pa je še vedno komaj 50bps v eno smer. Predvidevamo, da je to v glavnem omejitev našega laserja oziroma njegovega napajanja. Že s prostim očesom smo opazili, kar je potrdil tudi 240fps video posnetek, da laser potrebuje nekaj milisekund, da pride do polne moči, dokler ni pri polni moči, pa ga sprejemnik ne zazna. Zakasnitev pri laserski diodi ne bi smela biti tako velika, torej večji del zakasnitve prihaja od napajalnika. Uporabljali smo namreč 3.3V napajanje direktno s plošče

mikrokrmilnika. Ta uporablja regulator napetosti, ki je zelo slabe kvalitete in je namenjen le za komponente, ki potrebujejo relativno malo toka. Vklop laserja povzroči velik dvig porabe toka, ki ga regulator ne more takoj priskrbeti. To dalje povzroči padec napetosti po celotnem sistemu, kar je vidno tudi iz zmanjšanja svetlosti ostalih LED diod na plošči.

Rešitev tega bi bila uporaba zunanjega vira napetosti. Ker pa pri roki nismo imeli nobenega ustreznega napajalnika, tega nismo uspeli preizkusiti.

4.4. Fizična serijska povezava

V prejšnjem poskusu smo svoj protokol implementirali v programski opremi in podatke z računalnikom izmenjevali preko serijske povezavo. Prenos podatkov je tako omejevala ne samo hitrost mikrokrmilnika, ampak tudi hitrost serijske povezave in pretvornika, ki ju povezuje.

Zato smo dobili idejo, da bi poskusili laserski vmesnik za serijsko povezavo implementirati v celoti v strojni opremi, torej s posebnim vezjem, ki bi ga priključili direktno na serijski priključek (ali USB-Serial adapter) računalnika.

Tukaj je pomembno razlikovati med RS-232 in TTL Serial povezavo. Prva je pogosta na matičinih ploščah starejših računalnikov v obliki DB-25 priključka. Ta deluje z logičnimi nivoji `0` = +3V do +15V in `1` = −15V do −3V. Uporaba negativnih napetosti, še posebej za nedejavni signal, predstavlja veliko nevarnost za mnoga vezja, še posebej standardne mikrokrmilnike, saj delujejo le na TTL (*Transistor-Transistor Logic*) nivojih (GND do +5V ali +3,3V), vezje pa naredi zelo zapleteno.

Ker pa večina novejših računalnikov nima več vgrajenih RS-232 priključkov, bi v vsakem primeru morali uporabiti nek (npr. USB) pretvornik. Zato smo si delo olajšali in raje izbrali TTL Serial povezavo. Ta, za razliko od RS-232 deluje le na TTL nivojih. To je tudi povezava, ki jo uporabljajo mikrokrmilniki Arduino, ki imajo na ploščah vgrajene USB adpterje za programiranje preko računalnika. Če izklopimo procesor mikrokrmilnika tako, da RESET pin povežemo na GND, lahko ploščo uporabimo kot enostaven USB-Serial adapter.

Ko smo z načrtovanjem vezja za prenos TTL Serial signala po laserju že začeli, pa smo med iskanjem po spletu naleteli na projekt[5], ki je storil prav to. Tam opisano vezje smo želeli preizkusiti sami, a nismo imeli vseh potrebnih komponent. Integrirano vezje MAX232A, ki je osrednji element opisanega vezja je namreč še vedno na poti s Kitajske.

[5] http://www.qsl.net/n9zia/wireless/laser/laser.htm

4.5. Omrežna povezava

S serijsko povezavo vzpostavljeno med dvema računalnikoma, nam preostane le še vzpostavitev IP omrežja preko te povezave. Na srečo, protokoli in programska oprema za to že obstajajo.

Izmed teh sta najbolj znana dva: *Serial Line Internet Protocol (SLIP)* in *Point-to-Point Protocol (PPP)*. Oba sta nastala v devetdesetih letih z namenom povezave domačih računalnikov na internet preko serijskih povezav, ki so bile takrat dosti pogostejše in cenejše od takrat čisto novih tehnologij, kot je na primer Ethernet. Osredotočili se bomo na protokol *PPP*, saj je malo novejši in bolje podprt na modernih sistemih.

Kot je razvidno že iz imena, je *PPP* protokol, ki je namenjen povezavi dveh točk. Ko je *PPP* povezava enkrat vzpostavljena, lahko preko nje transparentno prenašamo kakršne koli podatke. Operacijski sistem jo vidi kot čisto navadno mrežno povezavo, ne glede na to, kakšna je fizična povezava. V našem primeru, smo za vzpostavitev *PPP* povezave uporabili implementacijo, ki je vključena v Linux jedro.

Ker je naš namen le čisto enostavna *Point-to-Point* povezava (brez naprednih funkcij, ki so specifične različnim fizičnim povezavam), bomo *pppd* (Linux orodje za vzpostavljanje *PPP* povezav) uporabili v t.i. "*Null modem"* konfiguraciji. V tej konfiguraciji programu *pppd* določimo le napravo (posebno Unix datoteko), s katero lahko sprejema in pošilja podatke, ter hitrost (*"baud rate"*), s katero naj deluje.

Ta naprava je naša plošča Arduino, ki jo računalnik poveže v datoteko */dev/ttyUSB0* (Arduino Nano) ali */dev/ACM0* (Arduino Leonardo). V primeru večih naprav enakega tipa, se številka na koncu poveča.

Za lažje odpravljanje napak, smo pred uporabo laserja najprej povezavo vzpostavili preko direktne serijske povezave. S spodnjim ukazom na obeh napravah naredili novo omrežno povezavo, ki je dobila oznako *ppp0*.

```
pppd -detach crtscts lock noauth [lokalni IP]:[oddaljen IP] /dev/[naprava] [baudrate]
```

Ko smo potrdili, da povezava deluje, smo žici zamenjali z laserskim sprejemnikom in oddajnikom in tako dobili omrežno povezavo.

Z dodajo parametra `proxyarp` na napravi z internetno povezavo in parametra `defaultroute` na napravi brez, smo preko laserjev dobili tudi dostop do interneta.

5. ZAKLJUČEK

Pred začetkom pisanja smo si zadali cilj – vsako metodo teoretično raziskati ter tudi implementirati v praksi. Teoretični del cilja smo opravili, pri praktičnem pa smo imeli nekaj težav.

Že v začetku smo na primer vedeli, da bomo za končni preizkus radijske povezave potrebovali radioamatersko licenco. To se nam na začetku ni zdela težava, saj smo licenco med letom tako ali tako želeli pridobiti. Zaradi pomanjkanja časa, pa nam to ni uspelo, zato je ta del raziskovalne naloge ostal zgolj teoretičen.

Ker smo toliko časa porabili na radijski povezavi, pa nam je zmanjkalo časa tudi za podrobnejšo raziskavo prenosa podatkov preko laserja. Osnovno povezavo smo sicer uspeli vzpostaviti, z malo več časa pa bi nam najverjetneje uspelo povezavo implementirati v strojni opremi in tako doseči dosti višje hitrosti.

Če čas dovoli, bomo raziskovanje še nadaljevali. Vso nadaljnjo delo na tem področju bomo po koncu ocenjevanja objavili na spodnji povezavi:

http://rebrand.ly/IPPrekoVsega2018

DRUŽBENA ODGOVORNOST

Komunikacija je v sodobnem svetu izrednega pomena. Brez prenosa informacij si več ne znamo predstavljati življenja. Vse, od osebnega življenja do gospodarstva, zdravstva, politike… je odvisno od prenosa informacij in znanja. Težko si je zamisliti, kako bi sodobni svet deloval, če bi iz kakršnegakoli razloga prišlo do prekinitve prenosa informacij. Čeprav je naše zaupanje v sodobno tehnologijo neskončno, je pomembno razmišljati o alternativnih načinih prenosa informacij. Sodobna tehnologija je namreč odvisna od številnih dejavnikov in deluje le v idealnih pogojih. Že stara ljudstva so se posluževala načinov prenosa, ki smo jih pozabili, ali pa niso več praktični. Pomembno je torej odkriti, kateri alternativni načini odgovarjajo sodobnemu svetu, a bi hkrati delovali tudi v izrednih razmerah. Najina raziskovalna naloga je namenjena prav temu – raziskovanje možnosti prenosa informacij v razmerah, ki onemogočajo uporabo sodobne tehnologije.

ZAHVALA

Posebej bi se radi zahvalili profesorju elektrotehnike, ki nam je pomagal pri načrtovanju vezja za sprejem laserja ter prijatelju, ki nam je v zadnjem trenutku posodil laserski kazalnik po tem, ko smo našega po pomoti priklopili na 5V namesto na 3V napajanje.

Viri in literatura

- Spletni portal LinuxHam, (http://www.linux-ax25.org/, 12.2.2018)
- *"AX25 Soundmodem"*, Dr. George Smart, M1GEO (https://www.george-smart.co.uk/aprs/ax25_soundmodem/, 12.2.2018)
- *"Notes on Soundmodem"*, Spletni portal HacDC (https://wiki.hacdc.org/index.php/Notes_on_Soundmodem, 12.2.2018)
- *"RS-232 Laser Transciever"*, Electronics Australia za oktober 1997 (http://www.qsl.net/n9zia/wireless/laser/laser.htm, 12.2.2018)
- *"Baofeng UV – Technical section"*, Miklor (http://www.miklor.com/COM/UV_Technical.php, 12.2.2018)
- *"ATmega32U4 Datasheet"*, Atmel (http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7766-8-bit-AVR-ATmega16U4-32U4_Datasheet.pdf, 12.2.2018)

Priloga 1: Izvorna koda za prenos podatkov preko laserja

```
#include "Arduino.h"

// Hitrost povezave z računalnikom
const int SERIAL_BAUD = 9600;

// Vsi časi so v milisekundah!
// Oddajanje
const int TIME_0 = 15;
const int TIME_1 = 30;
const int TIME_BETWEEN = 10;

// Sprejemanje
const int TIME_0_MAX = 25; // Vse manj kot to je 0
const int TIME_1_MIN = 25; // Vse več kot to je 1

// Zakasnitev po sprejemu
const int TIME_DEBOUNCE = 5; // Obvezno več kot TIME_BETWEEN
// Čas nedejavnosti, po katerem se sprejemnik resetira
const int TIME_TIMEOUT = 100;

// Številke pinov
const int RX_PIN = 11; // Sprejemnik (fotodioda)
const int TX_PIN = 2; // Oddajnik (laser)

// Bajt, ki ga prejemamo
byte received = 0b00000000;
// Bitna maska (pozicija naslednjega bita)
byte rcvMask = 0b00000001;

bool lastState;
unsigned long lastTime = 0;

// To se izvede ob zagonu mikrokrmilnika
void setup() {
   // Povezava na računalnik
   Serial.begin(SERIAL_BAUD);

   // Nastavitev pinov
   pinMode(TX_PIN, OUTPUT);

   // Vedno začni v stanju HIGH
   digitalWrite(TX_PIN, HIGH);
   lastState = digitalRead(RX_PIN);
}

// To se ponavlja neskončno
void loop() {
   if (Serial.available()) {
       byte input = Serial.read();
       transmit(input);
   }

   // Naslednje bi lahko bolje rešili s prekinitvami, a je bilo vse že
povezano ko smo ugotovili, da izbran pin ne podpira zunanjih prekinitev

   // Preberi vhod
   bool lineState = digitalRead(RX_PIN);
```

```
57
58      if (lineState != lastState) {
59          // Če se je stanje spremenilo
60
61          receive(lineState);
62
63          // Počakaj, da se senzor stabilizira
64          delay(TIME_DEBOUNCE);
65      }
66      // Shrani zadnje stanje
67      lastState = lineState;
68  }
69
70  // Pošlje bajt po liniji
71  void transmit(byte data) {
72      // Za vsak bit v bajtu, lsb -> msb
73      for (byte mask = 0b00000001; mask > 0; mask <<= 1) {
74          if (data & mask) {
75              // Bit je 1
76              send1();
77          } else {
78              // Bit je 0
79              send0();
80          }
81          delay(TIME_BETWEEN);
82      }
83  }
84
85  // Sprejme naslednji bit
86  void receive(bool lineState) {
87      // Čas od zadnje spremembe
88      long time = millis() - lastTime;
89
90      if (lastState == HIGH && lineState == LOW) {
91          // Linija se je spustila (falling)
92
93          // Shrani trenutni čas
94          lastTime = millis();
95
96      } else if (lastState == LOW && lineState == HIGH) {
97          // Linija se je dvignila (rising)
98
99          // Če povezava izgine predolgo, resetiraj vse
100         if (time > TIME_TIMEOUT) {
101             flush();
102             return;
103         }
104
105         if (time < TIME_0_MAX) {
106             // Short pulse
107
108             // Zapiši bit 0
109             pushBit(0);
110         } else if (time > TIME_1_MIN) {
111             // Long pulse
112
113             // Zapiši bit 1
```

```
            pushBit(1);
        }
    }
}

// Pošlje signal za 0
void send0() {
    digitalWrite(TX_PIN, LOW);
    delay(TIME_0);
    digitalWrite(TX_PIN, HIGH);
}

// Pošlje signal za 1
void send1() {
    digitalWrite(TX_PIN, LOW);
    delay(TIME_1);
    digitalWrite(TX_PIN, HIGH);
}

// Zapiši bit v prejemni bajt
void pushBit(unsigned short int bit) {

    if (bit == 1) {
        // Če je bit 1, ga zapiši na pozicijo v maski
        received |= rcvMask;
    }

    if (rcvMask == 0b10000000) {
        // Zadnji bit je bil prejet

        // Pošlji bajt na računalnik
        Serial.write(received);

        // Resetiraj sprejet bajt in masko
        flush();
    } else {
        // Bit še ni zadnji

        // Premakni bitno masko v levo (naslednja pozicija)
        rcvMask <<= 1;
    }
}

// Izprazni prejet bajt in se pripravi na novo sprejemanje
void flush() {
    // Izprazni prejet bajt
    received = 0b00000000;

    // Ponastavi bitno masko
    rcvMask = 0b00000001;
}
```