Mladi za napredek Maribora 2015

32. srečanje

# Inovacijski predlog

# Pametna soba

Računalništvo (strojna oprema, programska oprema, informatika)


Avtor: ZORAN URAN

Mentor: KARINA JUDER, SUZANA URAN

Šola: OŠ LEONA ŠTUKLJA MARIBOR


2015, Maribor

Mladi za napredek Maribora 2015

32. srečanje

# Inovacijski predlog

# Pametna soba

Računalništvo(strojna oprema, programska oprema, informatika)

2015, Maribor

# Kazalo

# Povzetek

V inovacijskem predlogu bom predstavil pametno sobo. To bo soba, ki bo pripravljena na ukaze z mojega telefona (preko bluetootha) in bo vsak ukaz, ki ga bom sprogramiral, tudi izvedla. Pametna soba bo imela hkrati raven avtonomije, saj bo sama nastavljala primerno osvetljenost sobe, zalivanje rastlin… . Tako se bom rešil tečnih opravil, ki jih moram opravljati vsak dan. V inovacijskem predlogu bom uporabljal naslednje metode:

- spoznavanje delovanja naprav,
- načrtovanje in gradnja naprav,
- preizkus delovanja,
- razčlenjevanje vzrokov napak in odprava napak v delovanju naprave,
- vgradnja v mojo sobo.

Če bo delovanje uspešno, bo morda ta na novo izumljeni sistem za pametno sobo prišel v vsako stanovanje.

Današnje pametne hiše delujejo na osnovi žičnega vodila. V mojem inovacijskem predlogu predlagam izvedbo pametne sobe na osnovi brezžične bluetooth komunikacije, kar omogoča izvedbo pametne sobe brez večje gradbene prenove.

# Zahvala

Rad se bi zahvali mentorici (moji mami) za spodbudo, dobre nasvete in pomoč pri izdelavi naprav. Hkrati se bi rad zahvalil somentorci za lektoriranje pisnega izdelka. In še bi se rad zahvali koordinatorki  za vso pomoč pri sestavi inovacijskega predloga.

# Uvod

»Tooooooooo!!! Uspelo mi je, moj bluetooth robot končno vozi,« je bila moja prva reakcija, ko je robot naredil prvi gib, katerega sem ukazal iz telefona. To je bila moja prva praktična uporaba bluetooth modula (in s tem bletooth komunikacije). In to je tlakovalo pot do tega inovacijskega predloga.

Moja soba je namreč do nedavnega imela standardno opremo. Tudi električna napeljava v njej je bila takšna. Stikalo za vklop luči se je nahajalo le pri vratih v mojo sobo. Ampak postelja se nahaja ravno ob steni nasproti vrat. Zato je vsako moje odpravljanje spat zelo nevarno, kajti po izklopu luči pri vratih moram po temi do svoje postelje. (Nastavim si dosti »min«.) Po uspehu z bluetooth robotom me je nekega dne, ob odpravljanju spat, prešinila ideja, da bi lahko izklopil oziroma vklopil luč v sobi s pomočjo bluetootha kar iz postelje. Pomislil sem tudi na dedka, ki je sledil razvoju tehnologije, vendar ga je izdalo zdravje in je veliko časa preživel v postelji. Tudi njemu bi bluetooth luč olajšala življenje.

Cilj inovacijskega predloga je izboljšati kvaliteto vsakdanjega življenja, pomagati invalidom in starejšim ljudem ter jim s tem vrniti vsaj del avtonomije. Inovacija te naloge je pametna soba z bluetooth komunikacijo. Prednost sobe z bluetooth komunikacijo je enostavna vgradnja in možnost krmiljenja naprav na daljavo. Uspeh pri tem inovacijskem predlogu bi bil moj prvi korak v to smer. Za cilj sem si zastavil delujoč osnutek bluetooth komunikacijske mreže z dvema delujočima napravama v svoji sobi. Po krajšem razmisleku sem izvedbi bluetooth luči, izbrane iz predhodno navedenih razlogov, dodal še izvedbo bluetooth žaluzij, saj se nahaja krmilna palica žaluzij za mojo pisalno mizo in jo težko dosežem.

Pred začetkom inovacijskega predloga sem že znal povezati telefon z bluetooth slave modulom in znal to povezavo uporabiti za pošiljanje ukazov. Vendar bom za pametno sobo moral še korak naprej in narediti uporabo bluetooth komunikacijskega posrednika lažjo. Ta posrednik je mikroprocesor, ki bo prejel ukaz in ga posredoval ostalim. Tako se uporabniku ne bo treba povezovati, pošiljati naprej množice ukazov in nato končati bluetooth komunikacijo. Saj imamo v primerjavi z elektroniko ljudje reflekse »crknjenega konja«. Avtonomni bluetooth sistem se namreč lahko sam poveže in preveri določene podatke (ali je dovolj toplo,…).

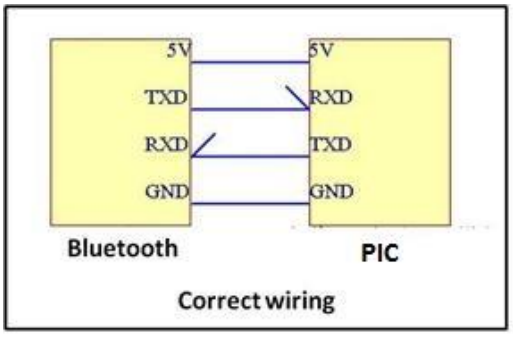
# Metodologija dela

## 1 Zbiranje podatkov

Za dosego zastavljenih ciljev sem moral spoznati Blutooth komunikacijo.

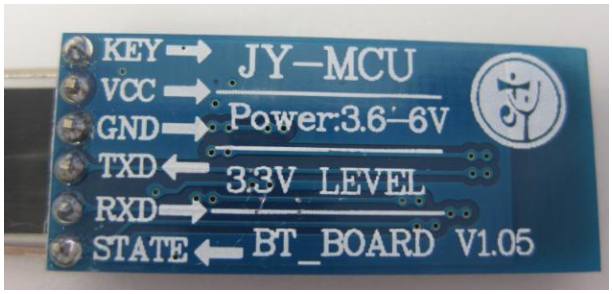
Najprej sem spoznal delovanje bluetooth komunikacije. Kratka zgodovino in opis delovanja najdemo v prilogi.

Iz povedanega v prilogi lahko vidimo, da je Bluetooth protokol zelo zapleten. Ampak osnutek vsega pomembnega je, da bluetooth protokol deli bluetooth module na gospodarje in na sužnje. Na srečo so za izvedbo bluetooth komunikacije izdelali integrirano vezje, ki izvaja Bluetooth komunikacijo samodejno. Naša naloga pa je, da temu integriranemu vezju posredujemo s pomočjo mnogo enostavnejše serijske žične komunikacije podatke, ki jih želimo prenesti s pomočjo Bluetooth komunikacije. Bluetooth modul z Rx in Tx signali povežemo z mikroprocesorjem, kot je prikazano na sliki 1 (Rx – read, Tx - talk).




Slika 1

Bluetooth suženj modul ločimo od bluetooth gospodar modula po številu nogic. Bluetooth suženj ima le štiri nogice (napajanje Vcc, GND, in Rx ter Tx). Bluetooth gospodar modul pa ima šest nogic (napajanje Vcc, GND, Rx, Tx, Key in State), kot je prikazano na sliki 2.




Slika 2

Bluetooth gospodar modul lahko preklaplja svojo vlogo, lahko je gospodar ali suženj v komunikacijski mreži. Bluetooth gospodar modul pozna t.i. AT ukazni način. V AT ukazni način lahko pridemo na dva načina. Pri prvem načinu je key nogica ob vklopu napajanja priključena na napajalno napetost in modul po inicializaciji komunicira z mikrokrmilnikom s hitrostjo 38400 baudov. Pri drugem načinu na key nogico priklopimo napajalno napetost po inicializaciji modula, zato modul komunicira z mikrokrmilnikom s hitrostjo 9600 baudov. Po vstopu v AT ukazni način bluettoth modul čaka AT ukaze, npr. AT+INQ. Z AT ukazi povemo gospodar modulu, kaj naj naredi oziroma kam se naj poveže. Npr. lahko zahtevamo podatke o MAC naslovu bluetootha, menjavo vloge med gospodarjem in sužnjem, zahtevamo poizvedovanje o drugih bluetooth napravah v bližini, itd. (Preglednica uporabljenih AT ukazov je v prilogi). Ko se bluetooth gospodar modul poveže z bluetooth suženj modulom, izklopimo AT ukazni način in lahko začnemo pošiljati podatke drugemu bluetooth modulu.

Pri svojem delu z bluetoothom sem na mojem telefonu z Android operacijskim sistemom uporabljal prosto dostopno Amarino aplikacijo. Le-ta omogoča iskanje bluetooth naprav preko telefona, povezavo z bluetooth napravami in pošiljanje podatkov bluetooth napravam. Aplikacija pošilja podatke v obliki Ascii znakov. (Glej Ascii tabelo v prilogi.)

Postopek povezovanja z bluetooth modulom:

- Na pametnem telefonu odpremo Amarino aplikacijo.
- Pritisnemo na gumb add BT device in počakamo, da najde nekaj bluetooth naprav.
- Izberemo tisto BT napravo, s katero se hočemo povezati.
- Vpišemo varnostno PIN kodo (ki je 1234), ki jo lahko v AT ukaznem načinu spremenimo.
- Bluetooth napravi se »pairata« in na ekranu se pokaže bluetooth naprava, s katero smo se ravno pairali.
- S pritiskom na connect gumb vzpostavimo povezavo.
- Nato pritisnemo na Monitoring gumb.
- Odpre se ukazna vrstica, v katero vpišemo ukaz, ki ga pošljemo bluetooth modulu.

# 2 Načrtovanje

Po končanem zbiranju podatkov je na vrsti načrtovanje. V pametni sobi je potrebno zasnovati Bluetooth komunikacijsko mrežo med napravami in predelati naprave za krmiljenje z bluetooth komunikacijo.

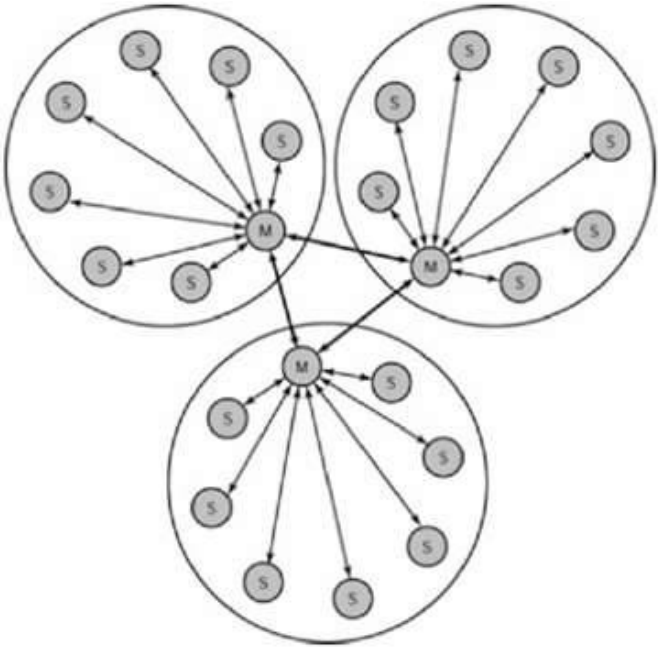
**Cilj:** Bluetooth komunikacijska mreža z dvema delujočima napravama (luč, žaluzije) in pametnim telefonom.

## 2.1 Bluetooth komunikacijska mreža

Bluetooth gospodar lahko komunicira z do sedmimi sužnji v piconet mreži. Mobilni telefon je bluetooth gospodar. S pomočjo mobilnega telefona (bluetooth gospodarja) torej lahko komuniciramo s sedmimi bluetooth napravami (sužnji) v pametni sobi. Torej bi lahko v moji pametni sobi bluetooth piconet mreža zadovoljila moje zahteve: krmiljenje luči in žaluzij s pomočjo mobilnega telefona. Vendar se za takšno izvedbo nisem odločil. Moje izkušnje z bluetooth komunikacijo med mobilnim telefonom (gospodar) in robotom (suženj) so pokazale, da varnost uporabe bluetooth komunikacije pri povezovanju s drugimi napravami zahteva veliko vpisov v mobilni telefon. Zato je takšna izvedba zahtevna za uporabo in tečna. Zastavil sem si nalogo, da zgradim pametno sobo z uporabniško prijaznejšo izvedbo bluetooth mreže. Da bi dosegel svoj cilj, potrebujem v svoji pametni sobi bluetooth komunikacijski vmesnik, ki bo vmesnik med mobilnim telefonom in napravami v moji sobi. Komunikacijski vmesnik bo omogočal, da bom pošiljal z mobilnega telefona ukaze večim napravam v moji sobi z enim samim povezovanjem mobilnega telefona v bluetooth mrežo.

Primer: želim več svetlobe v moji sobi. S telefon se povežem na vmesnik in mu ukažem, naj dvigne žaluzije. Ker pa zunaj ni več svetlo, moram vklopiti še luč.

Komunikacijski vmesnik mora igrati vlogo gospodarja in sužnja v moji bluetooth mreži pametne sobe. S tem se moja piconet mreža spremeni v scatternet mrežo, ki je prikazana na sliki 3.

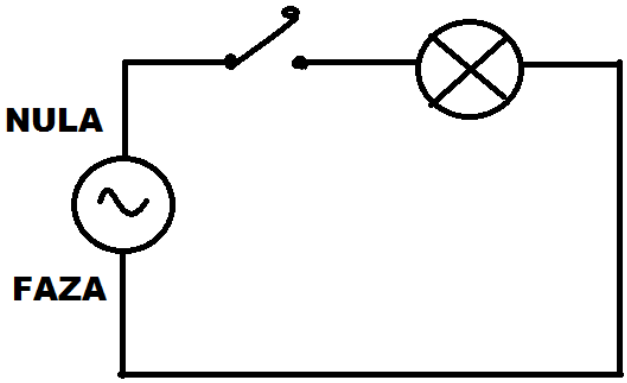



Slika 3

## 2.2 Bluetooth luč

Električno napeljavo za luč v moji sobi je potrebno predelati tako, da bom luč lahko vklopil in izklopil s pomočjo bluetootha in s pomočjo stikala pri vratih moje sobe, pri čemer naj bo za predelavo potrebnih čim manj gradbenih del. Hkrati želim s pomočjo bluetootha prebrati, ali je luč vklopljena.

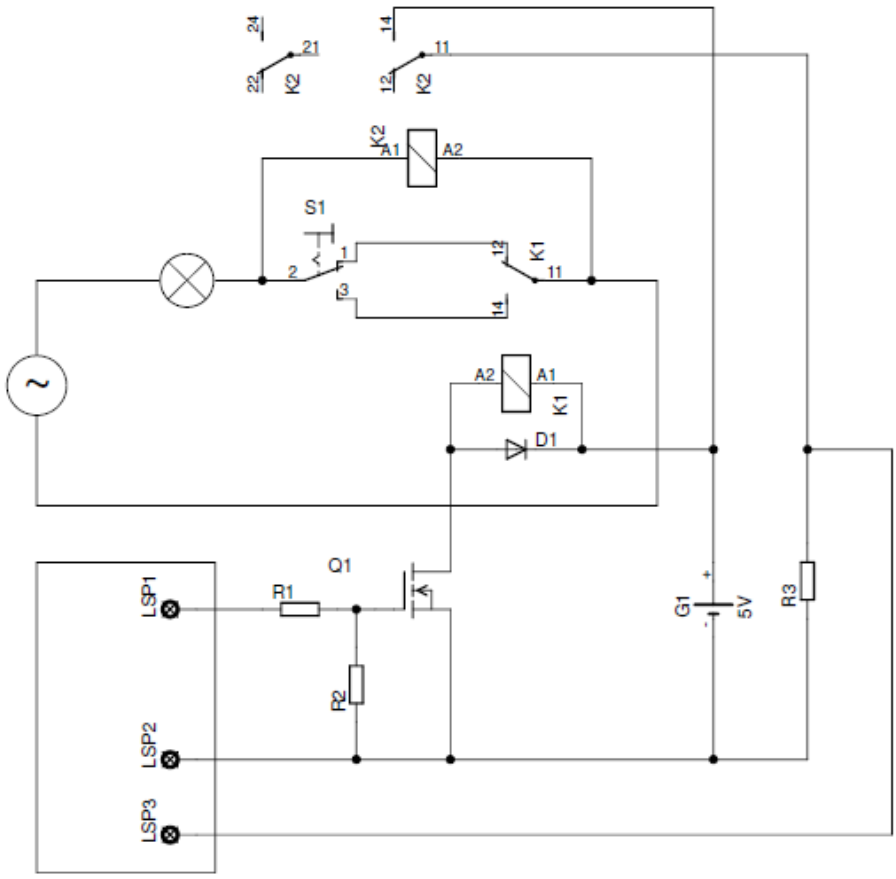
Doslej sem lahko luč v svoji sobi vklapljal le s stikalom pri vratih moje sobe. Električna napeljava v moji sobi pred predelavo je prikazana na sliki 4.




Slika 4

Če želim luč v svoji sobi vklapljati ročno, kot doslej, in s pomočjo bluetootha, potrebujem dve menjalni stikali. Prvo menjalno stikalo bo nadomestilo dosedanje navadno stikalo pri vratih. Drugo menjalno stikalo pa bo omogočilo vklop/izklop luči z bluetoothom. To drugo menjalno stikalo bom izvedel z elektromehanskim relejem in ga vgradil v steno ob vratih pod dosedanjim stikalom. Za ugotavljanje ali je luč v moji sobi vklopljena, potrebujem še en dodaten rele, ki ga bom vgradil skupaj s prvim relejem.

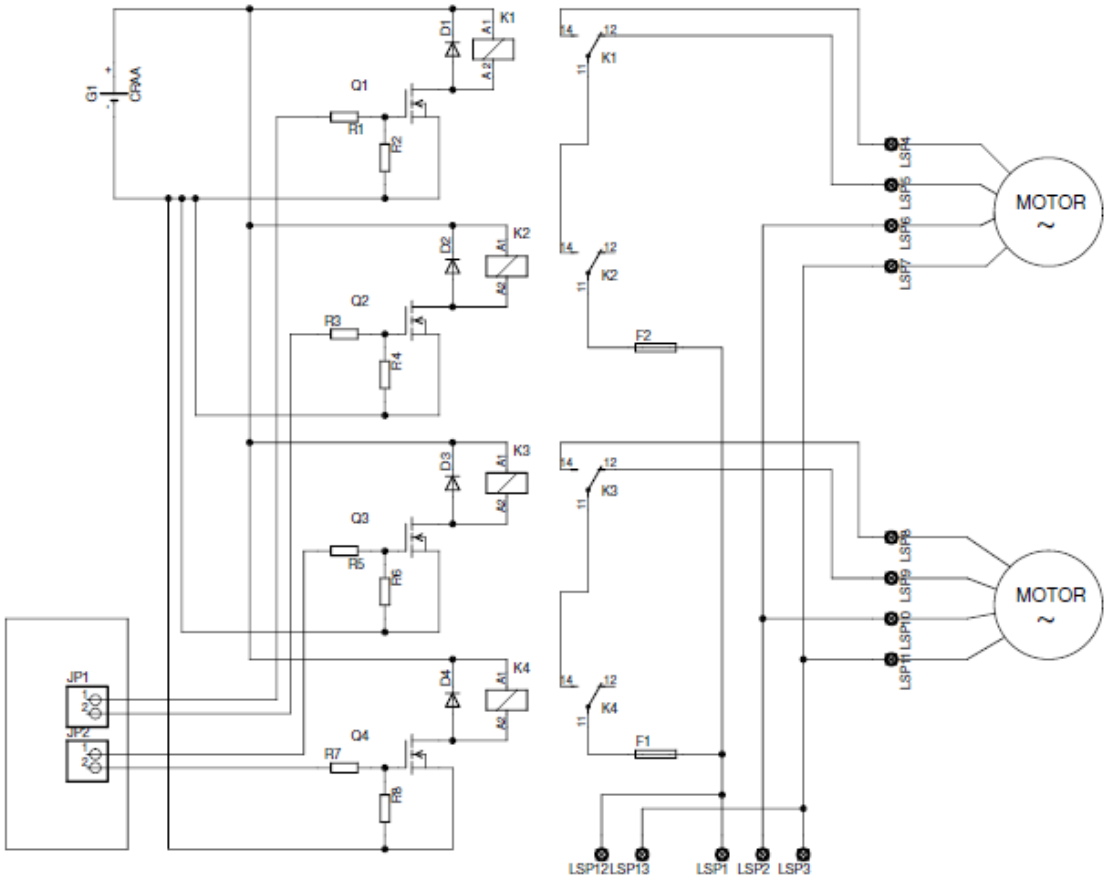
Električna napeljava za bluetooth luč v moji sobi je prikazana na sliki 5.




Slika 5

## 2.3 Bluetooth žaluzije

Na oknu moje sobe imam žaluzijo z motornim pogonom. To žaluzijo želim premikati s pomočjo bluetootha. Zato moram, tako kot pri luči, uporabiti releje. Motor žaluzij ima dva priključka za priklop fazne napetosti (L1,L2). Če priklopim napetost na L1, se žaluzije dvigujejo. Če pa priklopim napetost na L2, se žaluzije spuščajo. Vezje za bluetooth žaluzije je prikazano na sliki 6.




Slika 6

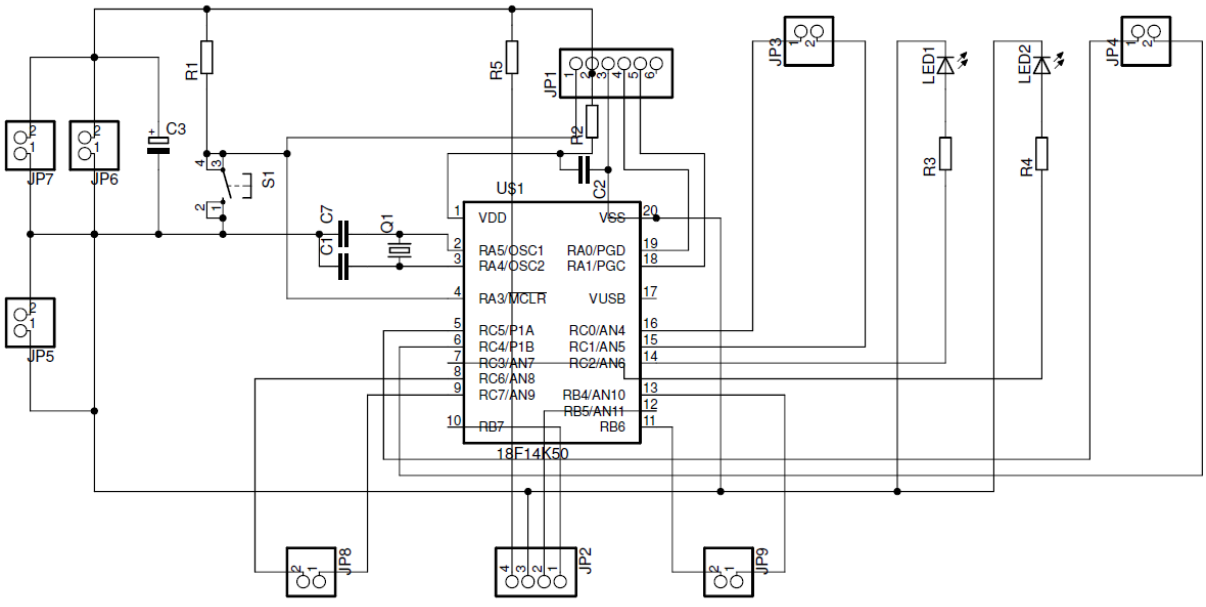
## 2.4 Načrtovanje mikroprocesorskih vezij

Načrte za mikroprocesorska vezja sem že risal v programu Eagle, zato jih bom tudi sedaj. Najprej sem risal vezje za bluetooth suženj napravo. Pri risanju načrta sem moral upoštevati pogoje:

- vsi elementi morajo imeti 5V napajanje in biti prav priključeni,
- mikroprocesor mora imeti uro 4 MHz,
- mikroprocesor mora imeti reset tipko,
- mikroprocesor mora biti povezan z modulom bluetooth suženj,
- mikroprocesor mora imeti dve signalizacijski LED,
- mikroprocesor mora imeti osem sponk za digitalne vhode/izhode za krmiljenje relejev,
- v vezju mora biti čim manj mostičkov.

Za to ploščico sem izbral mikrokrmilnik Pic18f14k22, ker je majhen, ima ena serijska vrata in dovolj digitalnih vhodov/izhodov, kljub temu da ima samo 20 nogic.

Vezje suženj bluetooth naprave je prikazano na sliki 7.

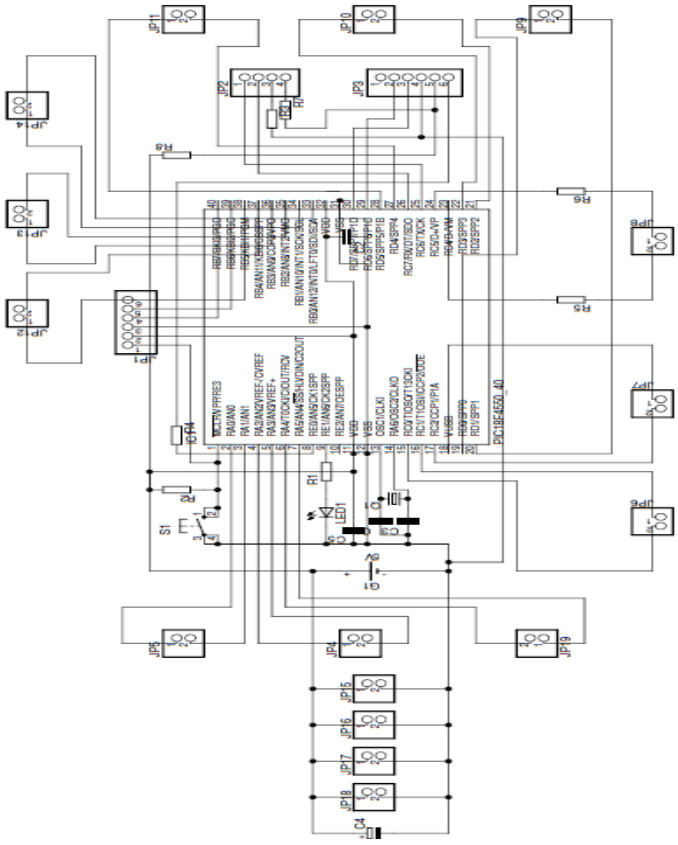



Slika 7

Naslednja v nizu načrtovanj je bila ploščica komunikacijskega vmesnika, za katero sem si zastavil naslednje pogoje:

- vsi elementi morajo imeti 5V napajanje in biti prav priključeni,
- mikroprocesor mora imeti uro 4 MHz,
- mikroprocesor mora imeti reset tipko,
- mikroprocesor mora biti povezan z modulom bluetooth suženj in z modulom bluetooth gospodar, torej mora imeti dvojna serijska vrata,
- mikroprocesor mora imeti eno signalizacijsko LED,
- mikroprocesor mora imeti dvanajst sponk za digitalne vhode/izhode za krmiljenje relejev,
- v vezju morajo biti 4 dodatne sponke za napajanje drugih vezij,
- v vezju mora biti čim manj mostičkov.

Po približno tednu razmišljanja sem izbral Pic18f45k22 za »glavno centralo« (komunikacijski vmesnik), ker ima dvojna serijska vrata in lahko nanj nalagam programe s samonalagalnikom.



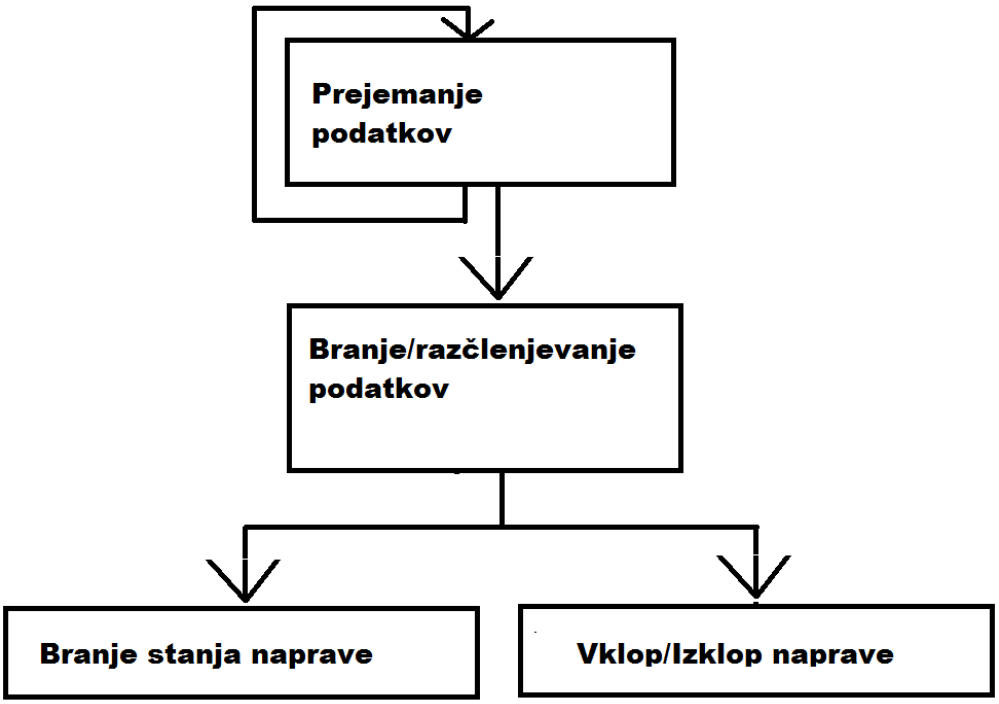
Slika 8

## 2.5 Načrtovanje programa

Zahteve za program:

Izvedba ukazov: vklop/izklop luči (LUC1,LUC0), branje stanja luči (LUC?), spust/dvig okenskih žaluzij (ZALUAD010,ZALUAD001), spust/dvig balkonskih žaluzij (ZALUAL010,ZALUAL001)

Diagram poteka programa:




Slika 9

# 3 Izdelava

Po koncu načrtovanja sem se lotil izdelave elektronskih vezij, programiranja in vgradnje.

## 3.1 Izdelava elektronskega vezja (jedkanje,vrtanje)

Proces izdelovanja ploščice se začne s tiskanjem že nastalega načrta na poseben svileni papir (v tem primeru na svileni papir reklam). Med tiskanjem vzamemo v roke vitroplast (pobakrena na eni strani) in jo pobrusim z gladkim brusnim papirjem, da odstranim zamaščen baker (pri tem tudi pomaga detergent).

Slika 10

Pred kratkim natisnjen papir obrežemo na velikost vitroplasta in ga spustimo skozi laminator. Barva na papirju se stopi in zalepi na pobakreni del vitroplasta.


Slika 11

Skozi laminator vitroplast spuščamo približno 15 minut.

Slika 12

Nato pa vitroplast (z zdaj prilepljenim papirjem) potopimo v mešanico 20% solne kisline in vodikovega peroksida (razmerje 10 : 1).


Slika 13

Baker takoj reagira s kislino in se začne raztapljati.

Slika 14

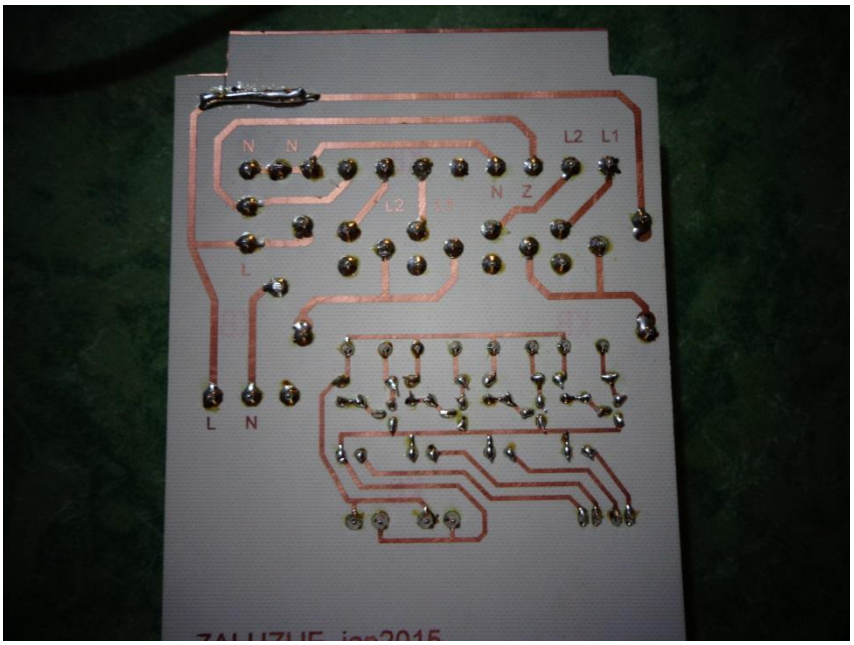
Kislino mešamo, dokler ne ostane samo načrt (in baker pod njem).


Slika 15

Po končanem delu vzamemo vitroplast iz kisline, ga operemo z vodo in z acetonom počistimo odvečno barvo. Na koncu še samo zvrtamo luknje. Po potrebi popravim kakšno napako, sicer je elektronsko vezje končano.

### 3.2 Spajkanje

Proces spajkanja se začne z zbiranjem potrebnih električnih (elektronskih) elementov. V mojem primeru sem za prejemniško vezje potreboval pic18f14k50, 40 pinsko podnožje, 5 sponk, 2 10nF kondenzatorja, 1 100 µF kondenzator, 2 10 µF kondenzatorja, 1 4MHz quartz, 1 LED, 1 1kΩ in 1 10kΩ upor. Ko so ti elementi zbrani, segrejemo spajkalnik in pripravimo orodje. V luknje vstavimo elektronske elemente (najnižji k najvišjim) in jih sproti spajkamo. To nadaljujemo, dokler niso vsi elementi prispajkani.




Slika 16

Ko končamo, preverimo svoje delo, npr. če ima mikroprocesor napajanje.

### 3.3 Programiranje

S končanimi in preizkušenimi mikroprocesorskimi ploščicami sem se osredotočil na programski del inovacijskega predloga. Programiral sem v programskem jeziku C. Osnova programiranja v tem jeziku je vpisovanje funkcij matematičnih peracij in »krmilnih« (if,While,else) stavkov. V naslednjem primeru sem uporabljal knjižnico funkcij za serijsko komunikacijo (npr. UART_init(9600)).

Primer programa:

```
while (Lastbyte2 != ACK)            // Tu preveri če zadnji byte ki ga je prebral ni ascii znak ACK
{
 while (UART1_Data_Ready() == 1)  // Tu preveri če je naslednji byte pripravljen na branje
{
 Lastbyte2 = UART1_Read();  // Tukaj prebere UART port in ga shrani v spremenljivko Lastbyte2
 buffer2[buffercount2] = Lastbyte2;  // Vrednost Lastbyte2 prepiše v buffer  z številko  buffercount2
 buffercount2 = buffercount2 + 1; //  buffercount2 se prišteje 1
}
}
```

Slika 17

Na sliki 14 vidimo del programa za branje niza znakov s serijskih vrat. Program se ponavlja in shranjuje prejete znake v vmesnik z imenom buffer2, dokler ne prebere ascii znak ACK (Acnowledge) z desetiško vrednostjo 19.

Naslednji primer (slika 14) prikazuje izsek programa za branje podatkov, razčlenitev podatkov in tudi izvedbo naloge (vklop ali izklop luči).

```
if (Prvibyte  == 0x41)  //  če je prvibyte  enak A
 {
  if (buffer[0] == 0x4C)   // če je buffer[0] enak L
  {
   if (buffer[1] == 0x55)   // če je buffer[1] enak U
    {
     if (buffer[2] == 0x31) // če je buffer[2] enak 1
      {
       if (PORTC.B5 == 1) // če je luč vklopljena
        {
          while (UART1_Tx_Idle == 0); //ali so serijska vrata prosta za pošijanje podatkov
          UART1_Write(0x4F);  // Pošlji ascii znak O
          while (UART1_Tx_Idle == 0); //ali so serijska vrata prosta za pošiljanje  podatkov
          UART1_Write(0x4B);  // pošlji ascii znak K
          while (UART1_Tx_Idle == 0); // ali so serijska  vrata prosta za pošiljanje  podatkov
          UART1_Write(19);  // pošlji asci znak ACK
         if (stanje == 0) // če luč ni vklopljena
         {
         PORTC = 0b00011000;  // vklopi luč
         stanje = stanje + 1; // spremenljivko  stanje povečaj za 1
         }
         else  // če je luč vklopljena
         {
         PORTC = 0b00000000;  // izklopi  luč
         stanje = stanje - 1; // stanje pomanjšaj  za 1
         }
        }
      }
    }
  }
}
```

Slika 18

## 3.4 Vgradnja

Zadnja metoda dela, ki sem jo uporabil, je vgradnja. To ni nič posebnega, saj gre samo za vgradnjo nekaj elektronskih ploščic na ali v steno.

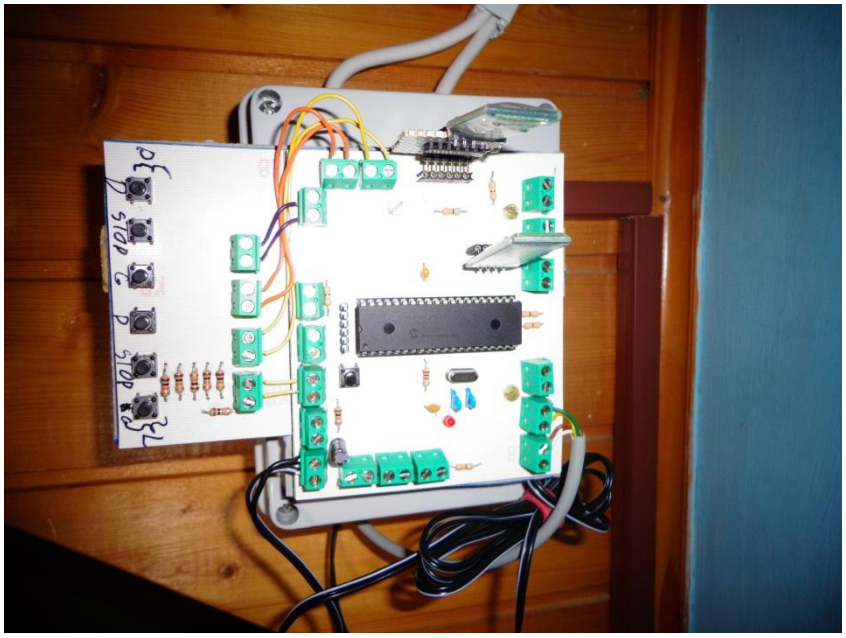
### *3.4.1 Predelava luči*

Slika 19

Slika 20

### *3.4.2 Predelava žaluzij*

Slika 21



Slika 22

# 4 Preizkušanje in odprava napak

Preizkušanja in odpravljanja napak se lotimo sistematično in po delih (če vse naenkrat odpove, sploh ne veš, kje začeti).

## 4.1 Preizkus tiskanih vezij

Ko uspešno prispajkamo vse elemente na tisk, je čas za preizkus. Preden kaj pregori, moramo delovanje tiska preizkusiti. To pomeni, da najprej preverimo z inštrumentom, če imajo tvsi elementi primerno napajanje. Če nimajo, je potrebno odkriti vzrok in ga odpraviti (ponavadi je to kakšna povezava preveč in zato pride do kratkega stika). Nato vstavimo elektronske elemente, naložimo testni program za utripanje LED diod in preverimo delovanje LED diod. Nato preverimo še vse preostale signalne povezave.

## 4.2 Preizkus Programa

S preverjenim tiskom se lotimo glavnega programa. Program pa preverimo tako, da ga naložimo na mikroprocesor in preizkusimo po delih. Če ne deluje kot bi moral, dodamo signalizacijo v program (npr. ko mikroprocesor prejme ascii znak A, posveti z rdečo LED) in tako je takoj očitno, kaj deluje in kaj ne. In potem je zelo preprosto najti napako ter jo odpraviti.

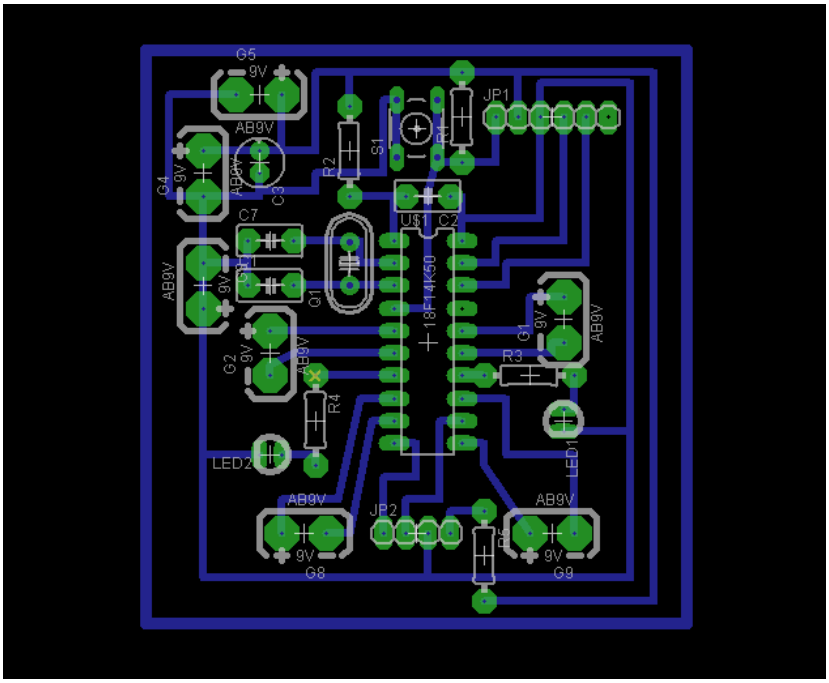
## 4.3 Odprava napak pri vgradnji v sobo

Sledi preverjanje za vgradnjo v sobo. Tukaj preverimo, če so tiski pravih dimenzij za vgradnjo, ali so še kakšne druge ovire (npr. trdožični kabli). Potem je čas za popoln test, kjer se preizkusi delovanje bluetooth naprave v celoti. Tako lahko najdemo tudi kakšne nenavadne napake (katerih še super računalnik ne bi mogel predvideti) npr. zatikanje relejev, ko so priklopljeni na žaluzije.

# Rezultati

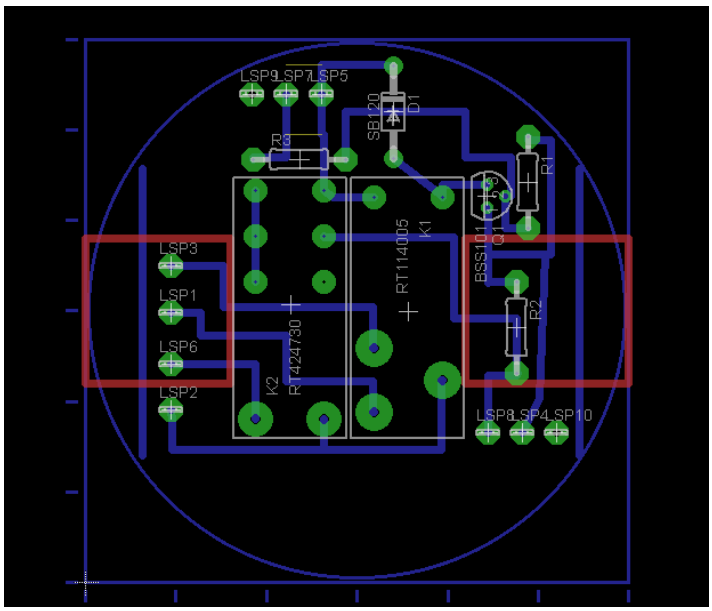
Seveda sploh ne bi napisal inovacijskega predloga, če ne bi imel rezultatov. Rezultat tega inovacijskega predloga je zasnova pametne sobe, kjer s pomočjo bluetooth mreže in mobilnega telefona vklopim in izklopim luč ter dvigam in spuščam žaluzije. Zaradi vmesnika je krmiljenje naprav v pametni sobi uporabniško prijazno.  Rezultati so prikazani na slikah 27 do 32.

Slika 27 prikazuje tisk suženjske ploščice. Ta je odgovorna za sprejemanje ukazov in preklapljanje relejev. V sredini ploščice je Pic18f14k22, ki dodeli napajanje kateri koli od nogic in tako vklopi ali izklopi releje.




Slike 23

Na sliki 28 vidimo relejsko ploščico (to je vgrajeno v steno). To je drugo menjalno stikalo moje luči. Rdeča območja predstavljaj veliko oviro, torej so lahko tam samo upori in diode.

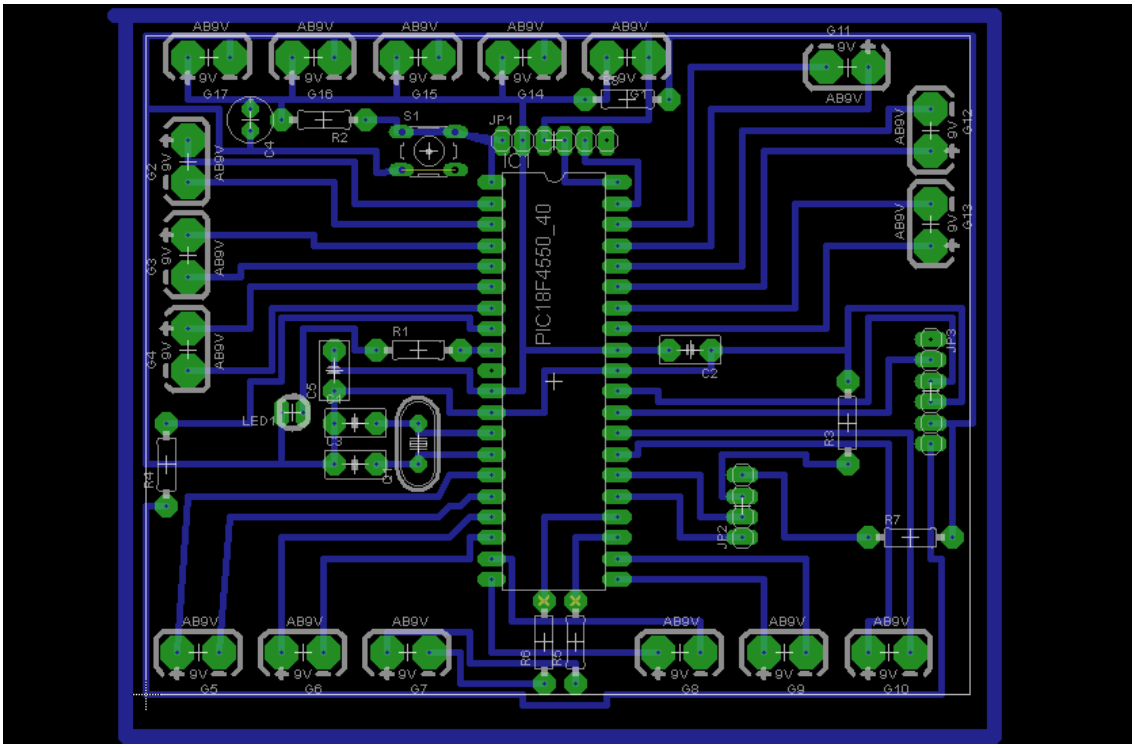



Slika 24

Na sliki 29 vidimo končani izdelek (končano stikalo). Z očitnim mikroprocesorjem prilepljen na bližnjo omaro. Pri tem novem stikalu lahko ali preitisnemo na tipko ali pa vtipkamo v telefon, ali LU1, ali LU0 za vklop ali izklop (z LU? beremo stanje).

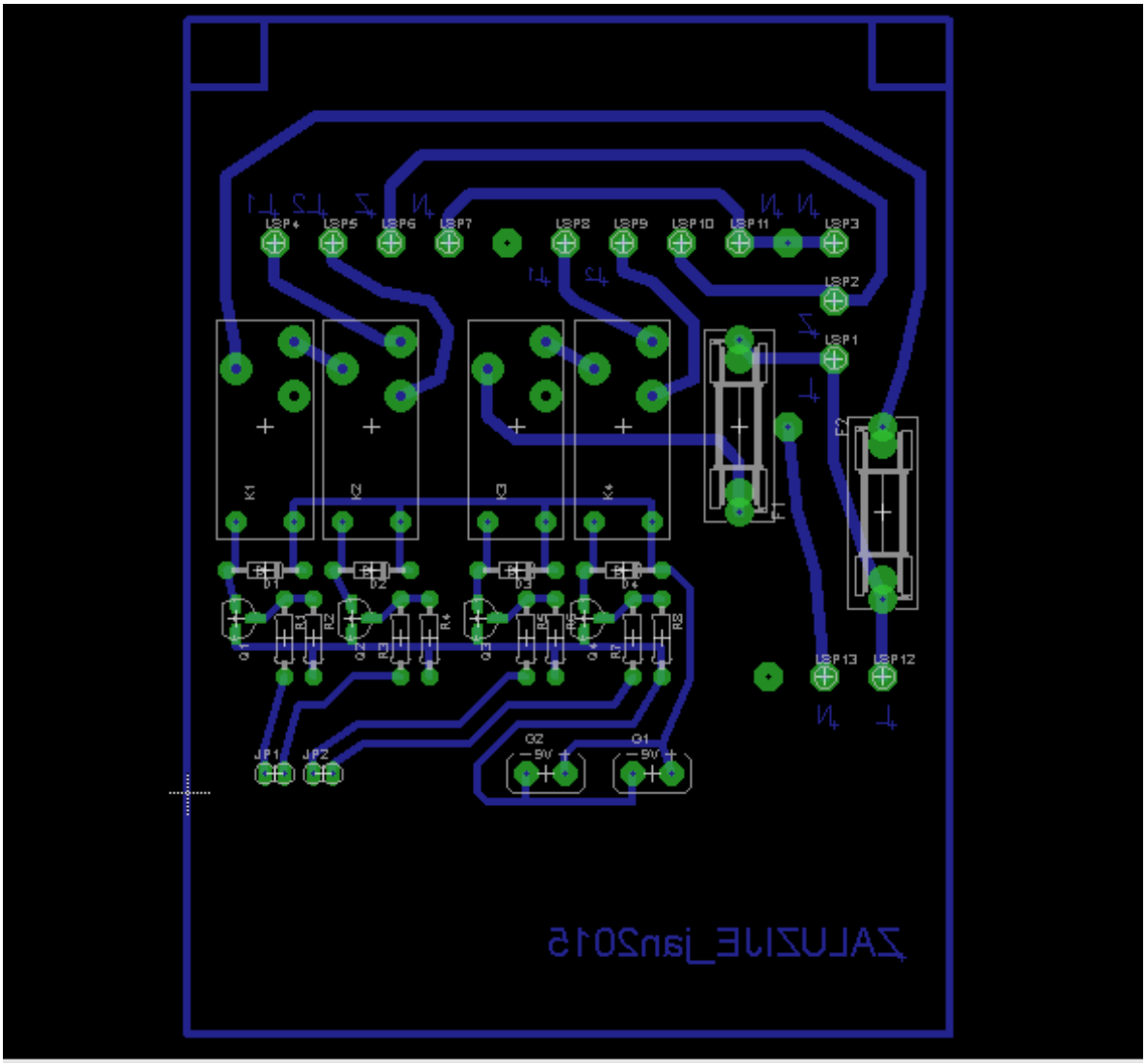
Slika 25

Na sliki 30 vidimo vmesniki (vmesniško ploščico). Njena naloga je prejemanje podatkov poslanih iz telefona nato jih malce spremeni in pošlje. Ta ploščica krmili tudi žaluzije na isti način kot suženj ploščica luč (več relejev).



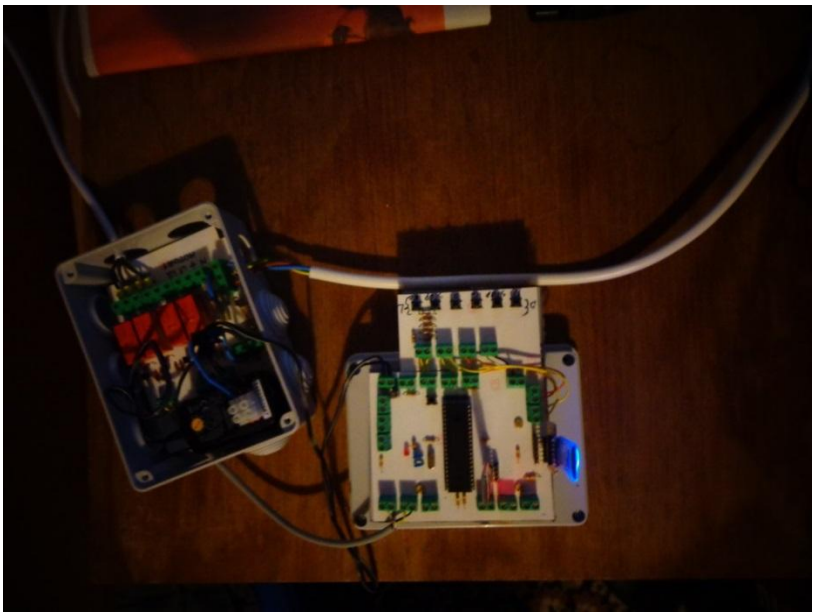

Slika 26

Na sliki vidimo relejsko ploščico za žaluzije. V notranjosti so priterjene tudi varovalke za dodatno varnost.



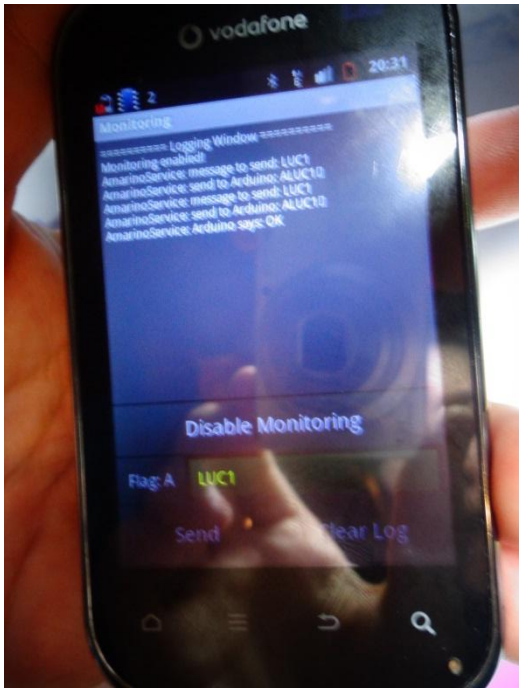

Slika 27

Na sliki 32 vidimo dokončan izdelek v preiskusni fazi. Priključena je samo ena žaluzija.



Slika 28

Na sliki 33 vidimo Amarino aplikacijo odprto na telefonu. Z vpisanim poveljem LUC1 ,kar je povelje za vmesnik, naj pošlje LU1 suženjski ploščici in ta bo vklopila luč.




Slika 29

# Razprava/Interpretacija rezultatov

Uspelo mi je zgraditi pametno sobo na osnovi bluetooth komunikacijske mreže z dvema delujočima napravama. Kot že rečeno bo moja soba poslej zame udobnejša, saj bom lahko ugašal in prižigal luč v iz postelje, konec pa je tudi stegovanja čez mizo pri premikanju žaluzij.

Pametna soba pa ne omogoča udobnejšega bivanja samo meni, ampak ima tudi številne prednosti za starejše slabo gibljive ljudi, ali ljudi priklenjene na posteljo in invalide, saj jim omogoča z vklapljanjem naprav na daljavo, nastavljanjem temperature, svetlobe, ipd. več avtonomije. Resnim bolnikom (srčni bolniki, sladkorni bolniki) pa omogoča tudi stalni zdravstveni nadzor, saj je lahko npr. merilnik srčnega utripa bolnika s pomočjo bluetootha neprekinjeno povezan z zdravniškim nadzornim središčem, tako da lahko le-ti v primeru potrebe hitro priskočijo na pomoč.

Stanovanja/hiše s pametno električno napeljavo si postopoma utirajo pot v naše življenje. V Evropi je 60 % vseh izvedenih napeljav pametnih električnih napeljav. V Sloveniji na tem področju precej zaostajamo. Trenutno so pametne električne napeljave dražje od navadnih, zato jih najdemo le v novih stanovanjskih objektih višjega cenovnega razreda (Eko srebrna hiša Ljubljana, TPC Livada Izola). Žal imajo tisti, ki pametno sobo najbolj potrebujejo – starejši ljudje in invalidi, običajna stanovanja brez pametne električne napeljave. Le-te so jim zaradi cene pogosto nedosegljive. Zaradi brezžične bluetooth komunikacije je izvedba pametne sobe v tem inovacijskem predlogu cenovno ugodna, izvedljiva tudi brez večjih gradbenih del in zelo modularna (omogoča postopno vgradnjo večjega števila naprav), na primer: nočem imeti luči na bluetooth, zato jo lahko po milji volji izpustim.  Zato menim, da bi bila dosegljiva tudi starejšim in invalidom.

# Zaključek

V zaključku lahko potrdim, da je bila ideja vredna truda. Moja soba je poslej bolj uporabniško prijazna, saj si lahko, ko grem spat, luč izklopim kar iz postelje, pa tudi dviganje in spuščanje žaluzij je sedaj dostopnejše in ugodnejše (brez stegovanja čez mizo). Uspeh tukaj pomeni prvi korak v popolno uporaben sistem pametne hiše, ki je cenovno ugoden in zato dostopen vsem. Ta sistem bo pomagal starejšim ljudem in invalidom vrniti avtonomnost. Sedaj imam odprto pot za nadaljnje izboljšave: s priklopom ustreznih bluetooth senzorjev lahko izvedem samodejno uravnavanje svetlobe in temperature v sobi. Seveda lahko sobo tudi robotiziram.

# Družbena odgovornost

Mislim, da je bilo moje delo družbeno odgovorno. Delal sem trdo in zavzeto. Upošteval sem varnost in primerno zaščitil električni del. Moj inovacijski predlog bo (morda) izboljšal življenjski standard in pomagal invalidom in starejšim ljudem vrniti avtonomijo. V našem svetu vse več časa preživimo v službi, zakaj bi morali priti domov in skrbeti še za hišo!? In zaradi dostopnosti sistemov pametnih hiš bi lahko tudi (če bi hoteli) privarčevali.

# Viri:


- http://blog.zakkemble.co.uk/getting-bluetooth-modules-talking-to-each-other/
- http://en.wikipedia.org/wiki/MAC_address
- http://en.wikipedia.org/wiki/Radio_frequency
- https://learn.sparkfun.com/tutorials/bluetooth-basics/bluetooth-profiles
- https://learn.sparkfun.com/tutorials/bluetooth-basics/how-bluetooth-works
- http://phillipecantin.blogspot.com/2014/01/hc-05-bluetooth-link-of-2-arduino.html
- S.Monk: Arduino+Android projects for the evil genius, McGraw-Hill,New york,Chicago,London,Madrid,2012
- http://sl.wikipedia.org/wiki/Radio
- http://www.amarino-toolkit.net/
- http://www.instructables.com/id/BlueTooth-Link-with-auto-detect-connect/?ALLSTEPS
- http://www.informit.com/articles/article.aspx?p=21324&seqNum=4
- http://www.ni.com/tutorial/3541/en/

# Viri slike in tabele:

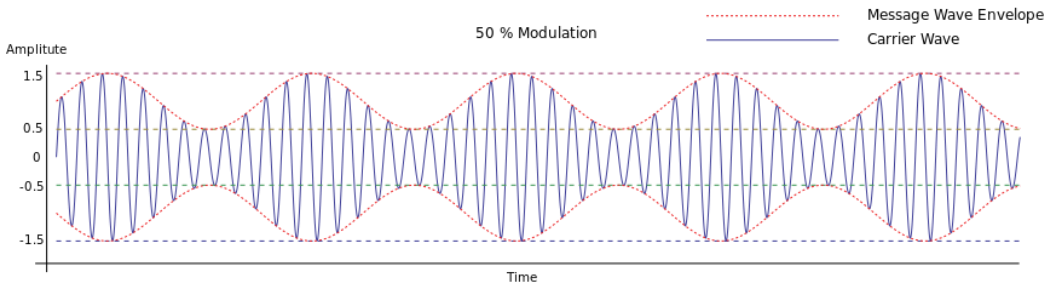
- Slika 1: http://2.bp.blogspot.com/-jvixH9FQ9TI/UiO5ObAtm5I/AAAAAAAAAGc/FCYy5O_48nw/s1600/connection.jpg
- Slika 2: http://mcuoneclipse.files.wordpress.com/2013/06/jy-mcu-bt_board-v1-05-bottom-side.png?w=584&h=275
- Slika 3: http://www.seminarski-diplomski.co.rs/INFORMATIKA/pictures/Topologija%20scatternet-a.JPG
- Slika 4: Avtor
- Slika 5: Avtor
- Slika 6: Avtor
- Slika 7: Avtor
- Slika 8: Avtor
- Slika 9: Avtor
- Slika 10: Avtor
- Slika 11: Avtor
- Slika 12: Avtor
- Slika 13: Avtor
- Slika 14: Avtor
- Slika 15: Avtor
- Slika 16: Avtor
- Slika 17: Avtor
- Slika 18: Avtor
- Slika 19: Avtor
- Slika 20: Avtor
- Slika 21: Avtor
- Slika 22: Avtor
- Slika 23: Avtor
- Slika 24: Avtor
- Slika 25: Avtor
- Slika 26: Avtor
- Slika 27: Avtor
- Slika 28: Avtor
- Slika 29: Avtor

- Slika 30: http://upload.wikimedia.org/wikipedia/commons/thumb/b/b0/Amplitude_Modulated_Wave-hm-64.svg/922px-Amplitude_Modulated_Wave-hm-64.svg.png
- Slika 31: http://upload.wikimedia.org/wikipedia/commons/thumb/a/ae/AM_spectrum.svg/600px-AM_spectrum.svg.png
- Slika 32: http://en.wikipedia.org/wiki/Gaussian_filter
- Slika 33: http://www.eetimes.com/document.asp?doc_id=1277573
- Slika 34: http://www.asciitable.com/index/asciifull.gif
- Tabela 1: Avtor

# Priloge

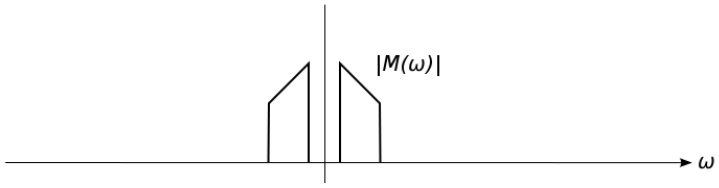
## Kratka zgodovina brezžične komunikacije/Spoznavanje bluetooth komunikacije

Bluetooth spada med brezžične komunikacije. Osnova brezžične komunikacije je elektromagnetno valovanje z visokimi frekvencami. Prvi začetki brezžične komunikacije segajo v leto 1901, ko je Marconi prvič uspešno izvedel brezžični prenos podatkov (morsove kode) iz Velike Britanije v Kanado. Leta 1906 je bil izveden prvi prenos zvoka po radijskih signalih. Za brezžično komunikacijo potrebujemo oddajnik, ki pošilja nosilni val, in sprejemnik, naravnan na frekvenco nosilnega vala. Podatki, ki jih pošiljamo, so naloženi na nosilni val. Frekvence nosilnega vala pri radijskem prenosu podatkov so od 4 kHz do 40 GHz (1cm do 1km). Prvi radijski prenosi podatkov so bili izvedeni tako, da so podatki spreminjali amplitudo nosilnega vala. Primer:




Slika 30

Na sliki 1 vidimo, kako je na nosilni val (modra barva) naložen podatek z enostavnim potekom - sinusni signal (rdeča barva). Od človeških glasov ima glas u potek, ki je najbližji sinusnemu poteku. Kadar brezžično prenašamo človeški govor, ki je sestavljen iz množice frekvenc, tedaj potrebujemo levo in desno ob nosilni frekvenci še frekvenčni pas človeškega govora (slika 2).

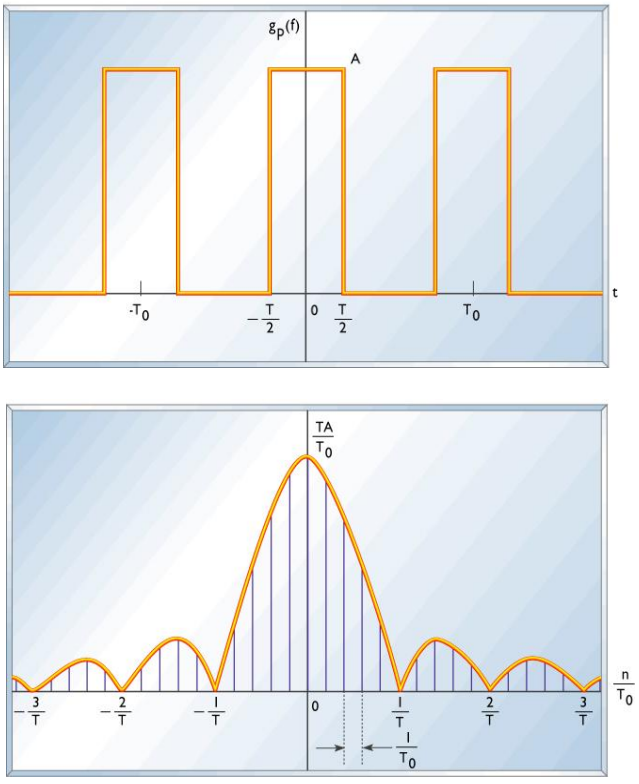



Slika 31

Bluetooth je, za razliko od radia, televizije in mobilnih telefonov, namenjen brezžičnemu prenosu podatkov na kratkih razdaljah, to je do največ 100 m. V prvi vrsti naj bi zamenjal žični prenos podatkov med računalnikom in napravami, npr. tiskalniki, …

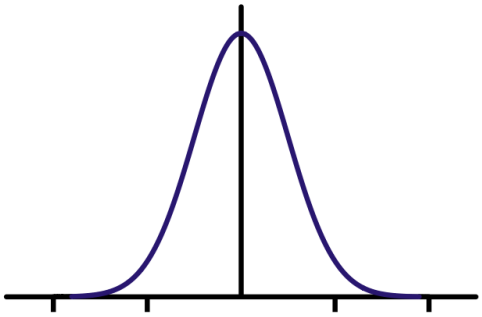
Stalen problem brezžičnega prenosa podatkov so motnje. Da bi bil Bluetooth prenos podatkov čim bolj odporen na motnje, Bluetooth naključno spreminja frekvenco svojega nosilnega vala 1600 krat na sekundo. To pomeni, da Bluetooth vsakih 625 mikrosekund spremeni frekvenco nosilnega signala. Bluetooth uporablja za nosilni val frekvence od 2400 MHz (2,4 GHz) do 2483,5 MHz (2,4835 GHz). Zadnja različica Bluetooth standarda (Bluetooth 4) določa za prenos podatkov pasovno širino 2 MHz, zato se nahaja na območju od 2400 MHz do 2483,5 MHz 40 različnih nosilnih frekvenc. Znotraj nabora teh 40 nosilnih frekvenc Bluetooth naključno spreminja svojo nosilno frekvenco.

Bluetooth standard določa prenos digitalnih podatkov, t.j. niz bitov podatkov, ki imajo le vrednosti 0 in 1. Te digitalne podatke prenašamo tako, da jih naložimo na nosilni val. Niz podatkov si lahko predstavljamo kot vlak pulzov. Vlak pulzov ima neskončno število frekvenc (slika 3, zgoraj vlak pulzov, spodaj slika frekvenc pulznega signala).




Slika 32

Ker imajo podatki v Bluebooth standardu omejeno pasovno širino 1 ali 2 MHz, podatke filtriramo (npr. z Gausovim filtrom, 8DSPK, ali ..), da jim omejimo pasovno širino, preden podatke naložimo na nosilni val. Gausov filter spremeni obliko pulzov, v obliko prikazano na sliki 4.



Slika 33

Za nas zelo pomembna značilnost Bluetooth protokola je, da deli naprave, ki med seboj komunicirajo na gospodarje in sužnje. En gospodar lahko komunicira z do 7 sužnji v mreži imenovani piconet. Vse naprave v piconet mreži uporabljajo isto uro za prenos podatkov, ki jo določa gospodar. Tudi vse spremembe frekvence nosilnega vala določa gospodar, zato lahko le gospodar sproži prenos podatkov, medtem ko sužnji le čakajo na povezavo z gospodarjem. Prenos podatkov na osnovi Bluetooth komunikacijskega protokola se izvede v podatkovnih paketih. Zato se podatki, ki jih želimo prenašati, razdelijo na pakete. Paket je določen s časom trajanja potrebnega za prenos podatkov. Osnovna časovna enota za pakete je 625 mikrosekund. Te časovne enote so oštevilčene. Gospodar oddaja podatke ob lihih časovnih enotah in sprejema podatke ob sodih časovnih enotah. Suženj pa ravno obratno oddaja podatke ob sodih časovnih enotah in sprejema ob lihih časovnih enotah.

Več piconet mrež se lahko poveže scatternet mrežo.

## Ascii tabela

| Dec | Hx | Oct | Char | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 000 | NUL (null) | 32 | 20 | 040 |   | Space | 64 | 40 | 100 | @ | @ | 96 | 60 | 140 | ` | ` |
| 1 | 1 | 001 | SOH (start of heading) | 33 | 21 | 041 | ! | ! | 65 | 41 | 101 | A | A | 97 | 61 | 141 | a | a |
| 2 | 2 | 002 | STX (start of text) | 34 | 22 | 042 | " | " | 66 | 42 | 102 | B | B | 98 | 62 | 142 | b | b |
| 3 | 3 | 003 | ETX (end of text) | 35 | 23 | 043 | # | # | 67 | 43 | 103 | C | C | 99 | 63 | 143 | c | c |
| 4 | 4 | 004 | EOT (end of transmission) | 36 | 24 | 044 | $ | $ | 68 | 44 | 104 | D | D | 100 | 64 | 144 | d | d |
| 5 | 5 | 005 | ENQ (enquiry) | 37 | 25 | 045 | % | % | 69 | 45 | 105 | E | E | 101 | 65 | 145 | e | e |
| 6 | 6 | 006 | ACK (acknowledge) | 38 | 26 | 046 | & | & | 70 | 46 | 106 | F | F | 102 | 66 | 146 | f | f |
| 7 | 7 | 007 | BEL (bell) | 39 | 27 | 047 | ' | ' | 71 | 47 | 107 | G | G | 103 | 67 | 147 | g | g |
| 8 | 8 | 010 | BS (backspace) | 40 | 28 | 050 | ( | ( | 72 | 48 | 110 | H | H | 104 | 68 | 150 | h | h |
| 9 | 9 | 011 | TAB (horizontal tab) | 41 | 29 | 051 | ) | ) | 73 | 49 | 111 | I | I | 105 | 69 | 151 | i | i |
| 10 | A | 012 | LF (NL line feed, new line) | 42 | 2A | 052 | * | * | 74 | 4A | 112 | J | J | 106 | 6A | 152 | j | j |
| 11 | B | 013 | VT (vertical tab) | 43 | 2B | 053 | + | + | 75 | 4B | 113 | K | K | 107 | 6B | 153 | k | k |
| 12 | C | 014 | FF (NP form feed, new page) | 44 | 2C | 054 | , | , | 76 | 4C | 114 | L | L | 108 | 6C | 154 | l | l |
| 13 | D | 015 | CR (carriage return) | 45 | 2D | 055 | - | - | 77 | 4D | 115 | M | M | 109 | 6D | 155 | m | m |
| 14 | E | 016 | SO (shift out) | 46 | 2E | 056 | . | . | 78 | 4E | 116 | N | N | 110 | 6E | 156 | n | n |
| 15 | F | 017 | SI (shift in) | 47 | 2F | 057 | / | / | 79 | 4F | 117 | O | O | 111 | 6F | 157 | o | o |
| 16 | 10 | 020 | DLE (data link escape) | 48 | 30 | 060 | 0 | 0 | 80 | 50 | 120 | P | P | 112 | 70 | 160 | p | p |
| 17 | 11 | 021 | DC1 (device control 1) | 49 | 31 | 061 | 1 | 1 | 81 | 51 | 121 | Q | Q | 113 | 71 | 161 | q | q |
| 18 | 12 | 022 | DC2 (device control 2) | 50 | 32 | 062 | 2 | 2 | 82 | 52 | 122 | R | R | 114 | 72 | 162 | r | r |
| 19 | 13 | 023 | DC3 (device control 3) | 51 | 33 | 063 | 3 | 3 | 83 | 53 | 123 | S | S | 115 | 73 | 163 | s | s |
| 20 | 14 | 024 | DC4 (device control 4) | 52 | 34 | 064 | 4 | 4 | 84 | 54 | 124 | T | T | 116 | 74 | 164 | t | t |
| 21 | 15 | 025 | NAK (negative acknowledge) | 53 | 35 | 065 | 5 | 5 | 85 | 55 | 125 | U | U | 117 | 75 | 165 | u | u |
| 22 | 16 | 026 | SYN (synchronous idle) | 54 | 36 | 066 | 6 | 6 | 86 | 56 | 126 | V | V | 118 | 76 | 166 | v | v |
| 23 | 17 | 027 | ETB (end of trans. block) | 55 | 37 | 067 | 7 | 7 | 87 | 57 | 127 | W | W | 119 | 77 | 167 | w | w |
| 24 | 18 | 030 | CAN (cancel) | 56 | 38 | 070 | 8 | 8 | 88 | 58 | 130 | X | X | 120 | 78 | 170 | x | x |
| 25 | 19 | 031 | EM (end of medium) | 57 | 39 | 071 | 9 | 9 | 89 | 59 | 131 | Y | Y | 121 | 79 | 171 | y | y |
| 26 | 1A | 032 | SUB (substitute) | 58 | 3A | 072 | : | : | 90 | 5A | 132 | Z | Z | 122 | 7A | 172 | z | z |
| 27 | 1B | 033 | ESC (escape) | 59 | 3B | 073 | ; | ; | 91 | 5B | 133 | [ | [ | 123 | 7B | 173 | { | { |
| 28 | 1C | 034 | FS (file separator) | 60 | 3C | 074 | < | < | 92 | 5C | 134 | \ | \ | 124 | 7C | 174 | | | \| |
| 29 | 1D | 035 | GS (group separator) | 61 | 3D | 075 | = | = | 93 | 5D | 135 | ] | ] | 125 | 7D | 175 | } | } |
| 30 | 1E | 036 | RS (record separator) | 62 | 3E | 076 | > | > | 94 | 5E | 136 | ^ | ^ | 126 | 7E | 176 | ~ | ~ |
| 31 | 1F | 037 | US (unit separator) | 63 | 3F | 077 | ? | ? | 95 | 5F | 137 | _ | _ | 127 | 7F | 177 | &#127; | DEL |

Source: www.LookupTables.com

Slika 34

## Tabela uporabljenih bluetooth ukazov

| | |
|---|---|
| AT | Preiskus delovanaj seriske komunikacije. |
| AT+INQ | Preveri, če so v dosegu druge bluetooth naprave |
| AT+INIT | Inicializira »bluetooth profil« in tako omogoča povezavo z drugimi bluetooth napravami |
| AT+PAIR=<naslov naprave>; <dani čas> | Prva povezava. Pri katerem si bluetooth modula izmenjata podatke, da se bosta lažje povezala naslednjič |
| AT+LINK=<naslov naprave> | Vzpostavi povezavo med dvema bluetooth napravama |
| AT+DISC | Prekine povezavo med dvema bluetooth napravama |

Tabela 1

## Program suženjske ploščice

```
char Prvibyte = 0;
char Potrdilo = 0;
char Prebranibyte = 0;
char ACK = 19;
char buffer[5];
char bufferCount = 0;
char Vecod0 = 0;
char stanje = 0;

void main()
{
 // --- inicializacija ---
 ANSEL = 0;
 ANSELH = 0;
 TRISC = 0b00100000;
 TRISB = 0b00000000;
 UART1_Init(9600);
 while (bufferCount < 5)
 {
  buffer[bufferCount] = 0;
  bufferCount = bufferCount + 1;
 }
 bufferCount = 0;
 PORTC = 0b00000100;
 delay_ms(1000);
 PORTC = 0b00000000;
while (UART1_Tx_Idle == 0)
 {
 }
 UART1_Write(0x4F);
 while (UART1_Tx_Idle == 0)
 {
 }
 UART1_Write(0x4B);
 while (UART1_Tx_Idle == 0)
 {
 }
 UART1_Write(19);
  while(1)
  {
    while (UART1_Data_Ready() == 1)
    {
        Prebranibyte = UART1_Read();
        Prvibyte = Prebranibyte;
        while (Prebranibyte != ACK)
        {
         if (UART1_Data_Ready() == 1)
         {
          Prebranibyte = UART1_Read();
          buffer[bufferCount] = Prebranibyte;
          bufferCount = bufferCount + 1;
```

```
  }
 }
 Potrdilo = Potrdilo + 1;
 bufferCount = 0;
}
if (Potrdilo == 1)
{

 if (Prvibyte == 0x41)
 {
  if (buffer[0] == 0x4C)
  {
   if (buffer[1] == 0x55)
    {
     if (buffer[2] == 0x31)
     {
      if (PORTC.B5 == 1)
       {
         while (UART1_Tx_Idle == 0)
         {
         }
          UART1_Write(0x4F);
          while (UART1_Tx_Idle == 0)
         {
         }
          UART1_Write(0x4B);
          while (UART1_Tx_Idle == 0)
         {
         }
          UART1_Write(19);
        if (stanje == 0)
        {
        PORTC = 0b00011000;
        stanje = stanje + 1;
        }
        else
        {
        PORTC = 0b00000000;
        stanje = stanje - 1;
        }
       }
    }
  }
 }
 }
 if (Prvibyte == 0x41)
 {

  if (buffer[0] == 0x4C)
  {

   if (buffer[1] == 0x55)
    {
     if (buffer[2] == 0x30)
      {
```

```
    if (PORTC.B5 == 0)
     {
       while (UART1_Tx_Idle == 0)
       {
       }
        UART1_Write(0x4F);
        while (UART1_Tx_Idle == 0)
       {
       }
        UART1_Write(0x4B);
        while (UART1_Tx_Idle == 0)
       {
       }
        UART1_Write(19);
      if (stanje == 0)
      {
      PORTC = 0b00011000;
      stanje = stanje + 1;
      }
      else
      {
      PORTC = 0b00000000;
      stanje = stanje - 1;
      }
    }
   }
  }
}
}
if (Prvibyte == 0x41)
{

 if (buffer[0] == 0x4C)
 {

  if (buffer[1] == 0x55)
   {
    if (buffer[2] == 0x3F)
    {
     if (PORTC.B5 == 0)
      {
       while (UART1_Tx_Idle == 0)
       {
       }
        UART1_Write(0x56);
        while (UART1_Tx_Idle == 0)
       {
       }
        UART1_Write(0x4B);
        while (UART1_Tx_Idle == 0)
       {
       }
        UART1_Write(19);
      }
      else
```

```
            {
             while (UART1_Tx_Idle == 0)
             {
             }
              UART1_Write(0x49);
              while (UART1_Tx_Idle == 0)
             {
             }
              UART1_Write(0x5A);
              while (UART1_Tx_Idle == 0)
             {
             }
              UART1_Write(19);
             }
           }
          }
         }
        }

       Potrdilo = Potrdilo - 1;

        while (bufferCount < 5)
        {
         buffer[bufferCount] = 0;
         bufferCount = bufferCount + 1;
        }
        bufferCount = 0;


   }
}
}
```

## Program za vmesniško ploščice

```
char eenter = 0x0D;
char denter = 0x0A;
char plus = 0x2B;
char AA = 0x41;
char BB = 0x42;
char CC = 0x43;
char DD = 0x44;
char EE = 0x45;
char FF = 0x46;
char GG = 0x47;
char HH = 0x48;
char II = 0x49;
char JJ = 0x4A;
char KK = 0x4B;
char LL = 0x4C;
char MM = 0x4D;
char NN = 0x4E;
char OO = 0x4F;
char PP = 0x50;
char RR = 0x52;
char SSS = 0x53;
char TT = 0x54;
char UU = 0x55;
char VV = 0x56;
char ZZ = 0x5A;
char YY = 0x59;
char QQ = 0x51;
char WW = 0x57;
char XX = 0x58;
char nic = 0x30;
char ena = 0x31;
char dva = 0x32;
char tri = 0x33;
char stiri = 0x34;
char pet = 0x35;
char sest = 0x36;
char sedem = 0x37;
char osem = 0x38;
char devet = 0x39;
char ook = 0x28;
char ozk = 0x29;
char dvop = 0x3A;
char enacaj = 0x3D;
char vejica = 0x2C;
buffer[100];
buffer2[100];
char zanka = 0;
int cas = 0;
char Lastbyte = 0;
char Lastbyte2 = 0;
char buffercount = 0;
```

```
char buffercount2 = 0;
char Potrdilo = 0;
char ACK = 19;
char pogoj = 0;
char pogoj2 = 0;
char pogoj3 = 0;
char pogoj4 = 0;
int stevilo1D = 0;
int stevilo2D = 0;
int stevilo3D = 0;
int steviloD = 0;
int stevilosD = 0;
int stevilo1L = 0;
int stevilo2L = 0;
int stevilo3L = 0;
int steviloL = 0;
int stevilosL = 0;
char enotaD = 0;
char enotaL = 0;
char stanje = 0;
void tipke()
{
 if (pogoj == 0)
 {
while (PORTB.B0 == 0)
{
 PORTC.B1 = 1;
 delay_ms(100);
 PORTC.B0 = 1;

 PORTE = 0b11111111;
 delay_ms(1000);
 PORTE = 0b00000001;
 pogoj2 = 1;
 pogoj = 1;
}
}
 if (pogoj == 0)
 {
while (PORTB.B2 == 0)
{
 PORTC.B1 = 0;
 delay_ms(100);
 PORTC.B0 = 1;

 PORTE = 0b11111111;
 delay_ms(1000);
 PORTE = 0b00000001;
 pogoj2 = 1;
 pogoj = 1;
}
}
if (pogoj4 == 0)
{
while (PORTB.B3 == 0)
```

```
{
 PORTC.B3 = 1;
 delay_ms(100);
 PORTC.B2 = 1;

 PORTE = 0b11111111;
 delay_ms(1000);
 PORTE = 0b00000001;
 pogoj3 = 1;
 pogoj4 = 1;
}
}
if (pogoj4 == 0)
{
while (PORTB.B5 == 0)
{
 PORTC.B3 = 0;
 delay_ms(100);
 PORTC.B2 = 1;

 PORTE = 0b11111111;
 delay_ms(1000);
 PORTE = 0b00000001;
 pogoj3 = 1;
 pogoj4 = 1;
}
}
while (PORTB.B4 == 0)
{
 PORTC.B2 = 0;
 PORTE = 0b11111111;
 delay_ms(1000);
 PORTE = 0b00000001;
 pogoj4 = 0;
}
while (PORTB.B1 == 0)
{
 PORTC.B0 = 0;
 PORTE = 0b11111111;
 delay_ms(1000);
 PORTE = 0b00000001;
 pogoj = 0;
}
}
void zakasnitevSD()
{
if (steviloD == 1)
{
delay_ms(1000);
}
if (steviloD == 2)
{
delay_ms(2000);
}
if (steviloD == 3)
```

```
{
delay_ms(3000);
}
if (steviloD == 4)
{
delay_ms(4000);
}
if (steviloD == 5)
{
delay_ms(5000);
}
if (steviloD == 6)
{
delay_ms(6000);
}
if (steviloD == 7)
{
delay_ms(7000);
}
if (steviloD == 8)
{
delay_ms(8000);
}
if (steviloD == 9)
{
delay_ms(9000);
}
if (steviloD == 10)
{
delay_ms(10000);
}
if (steviloD == 11)
{
delay_ms(11000);
}
if (steviloD == 12)
{
delay_ms(12000);
}
if (steviloD == 12)
{
delay_ms(13000);
}
if (steviloD == 14)
{
delay_ms(14000);
}
if (steviloD == 15)
{
delay_ms(15000);
}
if (steviloD == 16)
{
delay_ms(16000);
}
```

```
}
void zakasnitevSL()
{
if (steviloL == 1)
{
delay_ms(1000);
}
if (steviloL == 2)
{
delay_ms(2000);
}
if (steviloL == 3)
{
delay_ms(3000);
}
if (steviloL == 4)
{
delay_ms(4000);
}
if (steviloL == 5)
{
delay_ms(5000);
}
if (steviloL == 6)
{
delay_ms(6000);
}
if (steviloL == 7)
{
delay_ms(7000);
}
if (steviloL == 8)
{
delay_ms(8000);
}
if (steviloL == 9)
{
delay_ms(9000);
}
if (steviloL == 10)
{
delay_ms(10000);
}
if (steviloL == 11)
{
delay_ms(11000);
}
if (steviloL == 12)
{
delay_ms(12000);
}
if (steviloL == 12)
{
delay_ms(13000);
}
```

```
if (steviloL == 14)
{
delay_ms(14000);
}
if (steviloL == 15)
{
delay_ms(15000);
}
if (steviloL == 16)
{
delay_ms(16000);
}
}
void zakasnitevED ()
{
if (enotaD == 1)
{
delay_ms(1000);
}
if (enotaD == 2)
{
delay_ms(2000);
}
if (enotaD == 3)
{
delay_ms(3000);
}
if (enotaD == 4)
{
delay_ms(4000);
}
if (enotaD == 5)
{
delay_ms(5000);
}
if (enotaD == 6)
{
delay_ms(6000);
}
if (enotaD == 7)
{
delay_ms(7000);
}
if (enotaD == 8)
{
delay_ms(8000);
}
if (enotaD == 9)
{
delay_ms(9000);
}
if (enotaD == 10)
{
delay_ms(10000);
}
```

```
if (enotaD == 11)
{
delay_ms(11000);
}
if (enotaD == 12)
{
delay_ms(12000);
}
if (enotaD == 13)
{
delay_ms(13000);
}
if (enotaD == 14)
{
delay_ms(14000);
}
if (enotaD == 15)
{
delay_ms(15000);
}
if (enotaD == 16)
{
delay_ms(16000);
}
}
void zakasnitevEL ()
{
if (enotaL == 1)
{
delay_ms(1000);
}
if (enotaL == 2)
{
delay_ms(2000);
}
if (enotaL == 3)
{
delay_ms(3000);
}
if (enotaL == 4)
{
delay_ms(4000);
}
if (enotaL == 5)
{
delay_ms(5000);
}
if (enotaL == 6)
{
delay_ms(6000);
}
if (enotaL == 7)
{
delay_ms(7000);
}
```

```
if (enotaL == 8)
{
delay_ms(8000);
}
if (enotaL == 9)
{
delay_ms(9000);
}
if (enotaL == 10)
{
delay_ms(10000);
}
if (enotaL == 11)
{
delay_ms(11000);
}
if (enotaL == 12)
{
delay_ms(12000);
}
if (enotaL == 13)
{
delay_ms(13000);
}
if (enotaL == 14)
{
delay_ms(14000);
}
if (enotaL == 15)
{
delay_ms(15000);
}
if (enotaL == 16)
{
delay_ms(16000);
}
}
void master1()
{
while (zanka == 0)
 {
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(AA);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(TT);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(eenter);
 while (UART2_Tx_Idle() == 0)
 {
```

```
}
UART2_write(denter);
buffercount = 0;
while (Lastbyte != 0x0A)
{
while (UART2_Data_Ready() == 1)
{
Lastbyte = UART2_Read();
buffer[buffercount] = Lastbyte;
buffercount = buffercount + 1;
}
}
Lastbyte = 0;
if (buffer[0] == OO)
 {
  if (buffer[1] == KK)
  {
   zanka = 1;
  }
 }
}
zanka = 0;
while(zanka == 0)
{
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(AA);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(TT);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(plus);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(II);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(NN);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(II);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(TT);
while (UART2_Tx_Idle() == 0)
{
}
```

```
UART2_write(eenter);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(denter);
buffercount = 0;
while (Lastbyte != 0x0A)
{
while (UART2_Data_Ready() == 1)
{
Lastbyte = UART2_Read();
buffer[buffercount] = Lastbyte;
buffercount = buffercount + 1;
}
}
Lastbyte = 0;
if (buffer[0] == OO)
 {
  if (buffer[1] == KK)
   {
    zanka = 1;
   }
 }
if (buffer[0] == EE)
 {
   if (buffer[1] == RR)
    {
      if (buffer[2] == RR)
      {
        if (buffer[3] == OO)
        {
          if (buffer[4] == RR)
          {
           if (buffer[5] == dvop)
           {
            if (buffer[6] == ook)
            {
              if (buffer[7] == nic)
              {
                if (buffer[8] == ozk)
                {
                }
              }
              if (buffer[7] == ena)
              {
                if (buffer[8] == sedem)
                {
                 if (buffer[9] == ozk)
                 {
                  zanka = 1;
                 }
                }
              }
            }
           }
```

```
          }
         }
       }
     }
   }
 if (buffer[0] == FF)
  {
    if (buffer[1] == AA)
     {
       if (buffer[2] == II)
        {
         if (buffer[3] == LL)
         {
         }
        }
     }
}
}
zanka = 0;
while(zanka == 0)
 {
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(AA);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(TT);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(plus);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(LL);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(II);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(NN);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(KK);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(enacaj);
 while (UART2_Tx_Idle() == 0)
 {
```

```
}
UART2_write(dva);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(nic);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(ena);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(stiri);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(vejica);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(dva);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(vejica);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(ena);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(sedem);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(ena);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(sedem);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(pet);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(devet);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(eenter);
while (UART2_Tx_Idle() == 0)
```

```
{
}
UART2_write(denter);
Lastbyte = 0;
buffercount = 0;
while (Lastbyte != 0x0A)
{
  while (UART2_Data_Ready() == 1)
  {
    Lastbyte = UART2_Read();
    buffer[buffercount] = Lastbyte;
    buffercount = buffercount + 1;
  }
}
Lastbyte = 0;
buffercount = 0;
if (buffer[0] == OO)
 {
  if (buffer[1] == KK)
   {
    zanka = 1;
   }
 }
if (buffer[0] == EE)
 {
   if (buffer[1] == RR)
    {
      if (buffer[2] == RR)
      {
        if (buffer[3] == OO)
        {
          if (buffer[4] == RR)
          {
           if (buffer[5] == dvop)
            {
             if (buffer[6] == ook)
             {
              if (buffer[7] == nic)
               {
                if (buffer[8] == ozk)
                 {
                 }
               }
             }
            }
          }
        }
      }
    }
 }
if (buffer[0] == FF)
 {
   if (buffer[1] == AA)
    {
      if (buffer[2] == II)
```

```
      {
       if (buffer[3] == LL)
       {
       }
      }
    }
 }
 delay_ms(1000);
 }
PORTE = 0b00000010;
delay_ms(1000);
PORTE = 0b00000000;
delay_ms(1000);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(AA);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(LL);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(UU);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(nic);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(ACK);
 delay_ms(1000);
 PORTE = 0b00000001;
 delay_ms(1000);
zanka = 0;
 while(zanka == 0)
 {
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(AA);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(TT);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(plus);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(DD);
```

```
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(II);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(SSS);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(CC);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(eenter);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(denter);
buffercount = 0;
while (Lastbyte != 0x0A)
{
while (UART2_Data_Ready() == 1)
{
Lastbyte = UART2_Read();
buffer[buffercount] = Lastbyte;
buffercount = buffercount + 1;
}
}
Lastbyte = 0;
 if (buffer[0] == EE)
 {
   if (buffer[1] == RR)
    {
      if (buffer[2] == RR)
      {
        if (buffer[3] == OO)
         {
           if (buffer[4] == RR)
           {
            if (buffer[5] == dvop)
             {
              if (buffer[6] == ook)
               {
                if (buffer[7] == nic)
                 {
                  if (buffer[8] == ozk)
                   {
                   }
                 }
               }
             }
           }
         }
```

```
     }
    }
  }
 buffercount = 0;
 while (Lastbyte != 0x0A)
 {
 while (UART2_Data_Ready() == 1)
 {
 Lastbyte = UART2_Read();
 buffer[buffercount] = Lastbyte;
 buffercount = buffercount + 1;
 }
 }
 Lastbyte = 0;
 if (buffer[0] == OO)
  {
   if (buffer[1] == KK)
    {
     zanka = 1;
    }
  }
}
}
void master2()
{
 while (zanka == 0)
 {
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(AA);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(TT);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(eenter);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(denter);
 buffercount = 0;
 while (Lastbyte != 0x0A)
 {
 while (UART2_Data_Ready() == 1)
 {
 Lastbyte = UART2_Read();
 buffer[buffercount] = Lastbyte;
 buffercount = buffercount + 1;
 }
 }
 Lastbyte = 0;
 if (buffer[0] == OO)
```

```
 {
  if (buffer[1] == KK)
  {
   zanka = 1;
  }
 }
}
zanka = 0;
while(zanka == 0)
{
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(AA);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(TT);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(plus);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(II);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(NN);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(II);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(TT);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(eenter);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(denter);
buffercount = 0;
while (Lastbyte != 0x0A)
{
while (UART2_Data_Ready() == 1)
{
Lastbyte = UART2_Read();
buffer[buffercount] = Lastbyte;
buffercount = buffercount + 1;
}
```

```
}
Lastbyte = 0;
if (buffer[0] == OO)
 {
  if (buffer[1] == KK)
   {
    zanka = 1;
   }
 }
if (buffer[0] == EE)
 {
   if (buffer[1] == RR)
    {
      if (buffer[2] == RR)
      {
        if (buffer[3] == OO)
        {
          if (buffer[4] == RR)
          {
           if (buffer[5] == dvop)
           {
            if (buffer[6] == ook)
            {
             if (buffer[7] == nic)
             {
              if (buffer[8] == ozk)
              {
              }
             }
             if (buffer[7] == ena)
             {
               if (buffer[8] == sedem)
               {
                if (buffer[9] == ozk)
                {
                 zanka = 1;
                }
               }
             }
            }
           }
          }
        }
      }
    }
 }
if (buffer[0] == FF)
 {
   if (buffer[1] == AA)
    {
      if (buffer[2] == II)
       {
        if (buffer[3] == LL)
        {
        }
```

```
      }
    }
 }
}
zanka = 0;
while(zanka == 0)
 {
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(AA);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(TT);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(plus);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(LL);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(II);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(NN);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(KK);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(enacaj);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(dva);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(nic);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(ena);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(stiri);
```

```
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(vejica);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(dva);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(vejica);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(ena);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(sedem);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(ena);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(sedem);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(pet);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(devet);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(eenter);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(denter);
Lastbyte = 0;
buffercount = 0;
while (Lastbyte != 0x0A)
{
  while (UART2_Data_Ready() == 1)
  {
    Lastbyte = UART2_Read();
    buffer[buffercount] = Lastbyte;
    buffercount = buffercount + 1;
  }
}
```

```
 Lastbyte = 0;
 buffercount = 0;
if (buffer[0] == OO)
 {
  if (buffer[1] == KK)
   {
    zanka = 1;
   }
 }
if (buffer[0] == EE)
 {
   if (buffer[1] == RR)
    {
      if (buffer[2] == RR)
      {
        if (buffer[3] == OO)
        {
          if (buffer[4] == RR)
          {
           if (buffer[5] == dvop)
           {
            if (buffer[6] == ook)
            {
             if (buffer[7] == nic)
             {
              if (buffer[8] == ozk)
              {
              }
             }
            }
           }
          }
        }
      }
    }
 }
if (buffer[0] == FF)
 {
   if (buffer[1] == AA)
    {
      if (buffer[2] == II)
       {
        if (buffer[3] == LL)
       {
       }
       }
    }
}
 delay_ms(1000);
 }
PORTE = 0b00000010;
delay_ms(1000);
PORTE = 0b00000000;
delay_ms(1000);
 while (UART2_Tx_Idle() == 0)
```

```
{
}
UART2_write(AA);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(LL);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(UU);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(ena);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(ACK);
delay_ms(1000);
PORTE = 0b00000001;
delay_ms(1000);
zanka = 0;
while(zanka == 0)
{
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(AA);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(TT);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(plus);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(DD);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(II);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(SSS);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(CC);
while (UART2_Tx_Idle() == 0)
{
```

```
}
UART2_write(eenter);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(denter);
buffercount = 0;
while (Lastbyte != 0x0A)
{
while (UART2_Data_Ready() == 1)
{
Lastbyte = UART2_Read();
buffer[buffercount] = Lastbyte;
buffercount = buffercount + 1;
}
}
Lastbyte = 0;
 if (buffer[0] == EE)
 {
   if (buffer[1] == RR)
    {
      if (buffer[2] == RR)
      {
        if (buffer[3] == OO)
         {
           if (buffer[4] == RR)
           {
            if (buffer[5] == dvop)
             {
              if (buffer[6] == ook)
               {
                if (buffer[7] == nic)
                 {
                  if (buffer[8] == ozk)
                   {
                   }
                 }
               }
             }
           }
         }
      }
    }
 }
buffercount = 0;
while (Lastbyte != 0x0A)
{
while (UART2_Data_Ready() == 1)
{
Lastbyte = UART2_Read();
buffer[buffercount] = Lastbyte;
buffercount = buffercount + 1;
}
}
Lastbyte = 0;
```

```
 if (buffer[0] == OO)
  {
   if (buffer[1] == KK)
    {
     zanka = 1;
    }
  }
}
}
void master3()
{
////////////////////////////////////////////////////////////////////////////////
 while (zanka == 0)
 {
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(AA);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(TT);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(eenter);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(denter);
 buffercount = 0;
 while (Lastbyte != 0x0A)
 {
 while (UART2_Data_Ready() == 1)
 {
 Lastbyte = UART2_Read();
 buffer[buffercount] = Lastbyte;
 buffercount = buffercount + 1;
 }
 }
 Lastbyte = 0;
 if (buffer[0] == OO)
  {
   if (buffer[1] == KK)
   {
    zanka = 1;
   }
  }
 }
 zanka = 0;
////////////////////////////////////////////////////////////////////////////////
 while(zanka == 0)
 {
 while (UART2_Tx_Idle() == 0)
 {
```

```
}
UART2_write(AA);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(TT);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(plus);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(II);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(NN);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(II);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(TT);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(eenter);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(denter);
buffercount = 0;
while (Lastbyte != 0x0A)
{
while (UART2_Data_Ready() == 1)
{
Lastbyte = UART2_Read();
buffer[buffercount] = Lastbyte;
buffercount = buffercount + 1;
}
}
Lastbyte = 0;
if (buffer[0] == OO)
 {
  if (buffer[1] == KK)
   {
    zanka = 1;
   }
 }
if (buffer[0] == EE)
 {
   if (buffer[1] == RR)
```

```
   {
    if (buffer[2] == RR)
    {
     if (buffer[3] == OO)
      {
       if (buffer[4] == RR)
       {
        if (buffer[5] == dvop)
         {
          if (buffer[6] == ook)
           {
            if (buffer[7] == nic)
             {
              if (buffer[8] == ozk)
               {
               }
             }
             if (buffer[7] == ena)
              {
               if (buffer[8] == sedem)
               {
                if (buffer[9] == ozk)
                {
                 zanka = 1;
                }
               }
              }
           }
         }
       }
      }
    }
   }
 }
 if (buffer[0] == FF)
 {
  if (buffer[1] == AA)
   {
    if (buffer[2] == II)
     {
      if (buffer[3] == LL)
      {
      }
     }
   }
 }
}
zanka = 0;
////////////////////////////////////////////////////////////////////////////////////
 while(zanka == 0)
 {
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(AA);
```

```
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(TT);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(plus);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(LL);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(II);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(NN);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(KK);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(enacaj);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(dva);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(nic);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(ena);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(stiri);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(vejica);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(dva);
while (UART2_Tx_Idle() == 0)
{
}
```

```
UART2_write(vejica);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(ena);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(sedem);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(ena);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(sedem);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(pet);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(devet);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(eenter);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(denter);
Lastbyte = 0;
buffercount = 0;
while (Lastbyte != 0x0A)
{
  while (UART2_Data_Ready() == 1)
  {
    Lastbyte = UART2_Read();
    buffer[buffercount] = Lastbyte;
    buffercount = buffercount + 1;
  }
}
Lastbyte = 0;
buffercount = 0;
if (buffer[0] == OO)
 {
  if (buffer[1] == KK)
   {
    zanka = 1;
   }
 }
if (buffer[0] == EE)
 {
```

```
  if (buffer[1] == RR)
  {
    if (buffer[2] == RR)
    {
      if (buffer[3] == OO)
      {
        if (buffer[4] == RR)
        {
         if (buffer[5] == dvop)
         {
          if (buffer[6] == ook)
          {
           if (buffer[7] == nic)
           {
            if (buffer[8] == ozk)
            {
            }
           }
          }
         }
        }
      }
    }
  }
 }
if (buffer[0] == FF)
 {
  if (buffer[1] == AA)
  {
    if (buffer[2] == II)
    {
      if (buffer[3] == LL)
     {
     }
    }
  }
}
 delay_ms(1000);
 }
//////////////////////////////////////////////////////////////////////////
PORTE = 0b00000010;
delay_ms(1000);
PORTE = 0b00000000;
delay_ms(1000);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(AA);
 while (UART2_Tx_Idle() == 0)
 {
 }
 UART2_write(LL);
 while (UART2_Tx_Idle() == 0)
 {
 }
```

```
UART2_write(UU);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(0x3F);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(ACK);
buffercount = 0;
while (Lastbyte != ACK)
{
while (UART2_Data_Ready() == 1)
{
Lastbyte = UART2_Read();
buffer[buffercount] = Lastbyte;
buffercount = buffercount + 1;
}
}
Lastbyte = 0;
if (buffer[0] == VV)
 {
  if (buffer[1] == KK)
   {
   UART1_write(VV);
   while (UART1_Tx_Idle() == 0)
   {
   }
   UART1_write(KK);
   while (UART1_Tx_Idle() == 0)
   {
   }
   UART1_write(LL);
   while (UART1_Tx_Idle() == 0)
   {
   }
   UART1_write(OO);
   while (UART1_Tx_Idle() == 0)
   {
   }
   UART1_write(PP);
   while (UART1_Tx_Idle() == 0)
   {
   }
   UART1_write(LL);
   while (UART1_Tx_Idle() == 0)
   {
   }
   UART1_write(JJ);
   while (UART1_Tx_Idle() == 0)
   {
   }
   UART1_write(EE);
   while (UART1_Tx_Idle() == 0)
   {
```

```
  }
  UART1_write(NN);
  while (UART1_Tx_Idle() == 0)
  {
  }
  UART1_write(OO);
  while (UART1_Tx_Idle() == 0)
  {
  }
  UART1_write(ACK);
  }
 }
if (buffer[0] == II)
 {
  if (buffer[1] == ZZ)
  {
  UART1_write(II);
  while (UART1_Tx_Idle() == 0)
  {
  }
  UART1_write(ZZ);
  while (UART1_Tx_Idle() == 0)
  {
  }
  UART1_write(KK);
  while (UART1_Tx_Idle() == 0)
  {
  }
  UART1_write(LL);
  while (UART1_Tx_Idle() == 0)
  {
  }
  UART1_write(OO);
  while (UART1_Tx_Idle() == 0)
  {
  }
  UART1_write(PP);
  while (UART1_Tx_Idle() == 0)
  {
  }
  UART1_write(LL);
  while (UART1_Tx_Idle() == 0)
  {
  }
  UART1_write(JJ);
  while (UART1_Tx_Idle() == 0)
  {
  }
  UART1_write(EE);
  while (UART1_Tx_Idle() == 0)
  {
  }
  UART1_write(NN);
  while (UART1_Tx_Idle() == 0)
  {
```

```
  }
  UART1_write(OO);
  while (UART1_Tx_Idle() == 0)
  {
  }
  UART1_write(ACK);
  }
 }
delay_ms(1000);
PORTE = 0b00000001;
delay_ms(1000);
zanka = 0;
while(zanka == 0)
{
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(AA);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(TT);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(plus);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(DD);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(II);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(SSS);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(CC);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(eenter);
while (UART2_Tx_Idle() == 0)
{
}
UART2_write(denter);
buffercount = 0;
while (Lastbyte != 0x0A)
{
while (UART2_Data_Ready() == 1)
{
```

```
Lastbyte = UART2_Read();
buffer[buffercount] = Lastbyte;
buffercount = buffercount + 1;
}
}
Lastbyte = 0;
 if (buffer[0] == EE)
 {
  if (buffer[1] == RR)
   {
    if (buffer[2] == RR)
    {
     if (buffer[3] == OO)
      {
       if (buffer[4] == RR)
       {
        if (buffer[5] == dvop)
         {
          if (buffer[6] == ook)
           {
            if (buffer[7] == nic)
             {
              if (buffer[8] == ozk)
               {
               }
             }
           }
         }
       }
    }
   }
 }
buffercount = 0;
while (Lastbyte != 0x0A)
{
while (UART2_Data_Ready() == 1)
{
Lastbyte = UART2_Read();
buffer[buffercount] = Lastbyte;
buffercount = buffercount + 1;
}
}
Lastbyte = 0;
if (buffer[0] == OO)
 {
  if (buffer[1] == KK)
   {
    zanka = 1;
   }
 }
}
}

void main()
```

```
{
ANSELE = 0;
TRISE = 0;
ANSELD = 0;
TRISD = 0;
ANSELC = 0;
TRISC = 0;
ANSELB = 0;
TRISB = 0b11111111;
UART1_init(9600);
UART2_init(38400);
PORTC = 0b00000000;
PORTE = 0b00000010;
delay_ms(3000);
PORTE = 0b00000000;
delay_ms(3000);
PORTE = 0b00000001;
delay_ms(1);
 while (bufferCount < 100)
 {
  buffer[bufferCount] = 0;
  bufferCount = bufferCount + 1;
 }
 bufferCount = 0;
 while (bufferCount2 < 100)
 {
  buffer2[bufferCount2] = 0;
  bufferCount2 = bufferCount2 + 1;
 }
 bufferCount2 = 0;

 while(1)
 {

  while (Lastbyte2 != ACK)
  {
   tipke();
   while (UART1_Data_Ready() == 1)
  {
   Lastbyte2 = UART1_Read();
   buffer2[buffercount2] = Lastbyte2;
   buffercount2 = buffercount2 + 1;
  }
  }


        if(buffer2[0] == AA)
         {
          if(buffer2[1] == LL)
           {
            if(buffer2[2] == UU)
             {
              if(buffer2[3] == CC)
               {
                if(buffer2[4] == ena)
```

```
        {
         if(buffer2[5] == ACK)
          {
        PORTE = 0b11111111;
        delay_ms(1000);
        PORTE = 0b00000001;
         while (UART1_Tx_Idle() == 0)
          {
          }
          UART1_write(OO);
         while (UART1_Tx_Idle() == 0)
          {
          }
          UART1_write(KK);
          while (UART1_Tx_Idle() == 0)
          {
          }
          UART1_write(ACK);
          master2();
          }
         }
        }
    }
  }
 }
if(buffer2[0] == AA)
 {
  if(buffer2[1] == LL)
  {
   if(buffer2[2] == UU)
    {
     if(buffer2[3] == CC)
      {
       if(buffer2[4] == nic)
       {
        if(buffer2[5] == ACK)
          {
           PORTE = 0b11111111;
           delay_ms(1000);
           PORTE = 0b00000001;
           delay_ms(1000);
           PORTE = 0b11111111;
           delay_ms(1000);
           PORTE = 0b00000001;

         while (UART1_Tx_Idle() == 0)
          {
          }
          UART1_write(OO);
         while (UART1_Tx_Idle() == 0)
          {
          }
          UART1_write(KK);
          while (UART1_Tx_Idle() == 0)
          {
```

```
          }
          UART1_write(ACK);
          master1();
          }
        }
      }
    }
  }
 }
if(buffer2[0] == AA)
 {
  if(buffer2[1] == LL)
   {
    if(buffer2[2] == UU)
    {
     if(buffer2[3] == CC)
      {
       if(buffer2[4] == 0x3F)
        {
         if(buffer2[5] == ACK)
          {
           PORTE = 0b11111111;
           delay_ms(1000);
           PORTE = 0b00000001;
           delay_ms(1000);
           PORTE = 0b11111111;
           delay_ms(1000);
           PORTE = 0b00000001;
           delay_ms(1000);
           PORTE = 0b11111111;
           delay_ms(1000);
           PORTE = 0b00000001;

          while (UART1_Tx_Idle() == 0)
          {
          }
          UART1_write(OO);
          while (UART1_Tx_Idle() == 0)
          {
          }
          UART1_write(KK);
          while (UART1_Tx_Idle() == 0)
          {
          }
          UART1_write(ACK);
          master3();
          }
        }
      }
    }
   }
  }

if(buffer2[0] == BB)
```

```
{
 if(buffer2[1] == ZZ)
  {
   if(buffer2[2] == AA)
    {
     if(buffer2[3] == LL)
      {
       if(buffer2[4] == UU)
        {
         if(buffer2[5] == MM)
          {
          if(buffer2[6] == nic)
           {
           if(buffer2[7] == ACK)
           {

          while (UART1_Tx_Idle() == 0)
           {
           }
           UART1_write(OO);
          while (UART1_Tx_Idle() == 0)
           {
           }
           UART1_write(KK);
           while (UART1_Tx_Idle() == 0)
           {
           }
           UART1_write(ACK);
           pogoj = 1;
           }
         }
        }
    }
  }
 }
}
}
}
if(buffer2[0] == BB)
{
 if(buffer2[1] == ZZ)
  {
   if(buffer2[2] == AA)
    {
     if(buffer2[3] == LL)
      {
       if(buffer2[4] == UU)
        {
         if(buffer2[5] == MM)
          {
           if(buffer2[6] == ena)
           {
           if(buffer2[7] == ACK)
           {

          while (UART1_Tx_Idle() == 0)
```

```
          {
          }
          UART1_write(OO);
          while (UART1_Tx_Idle() == 0)
          {
          }
          UART1_write(KK);
          while (UART1_Tx_Idle() == 0)
          {
          }
          UART1_write(ACK);
          pogoj = 0;
          }
        }
      }
    }
  }
 }
 }
 }
if(buffer2[0] == CC)
 {
  if(buffer2[1] == ZZ)
   {
    if(buffer2[2] == AA)
     {
      if(buffer2[3] == LL)
       {
        if(buffer2[4] == UU)
        {
         if(buffer2[5] == AA)
         {
          if(buffer2[6] == DD)
           {
          if(buffer2[10] == ACK)
          {
         while (UART1_Tx_Idle() == 0)
          {
          }
          UART1_write(OO);
         while (UART1_Tx_Idle() == 0)
          {
          }
          UART1_write(KK);
          while (UART1_Tx_Idle() == 0)
          {
          }
          UART1_write(ACK);
          PORTC = 0b00000000;
          stevilo1D = buffer2[7] - 48;
          stevilo2D = buffer2[8] - 48;
          stevilo3D = buffer2[9] - 48;
          stevilo1D = stevilo1D * 100;
          stevilo2D = stevilo2D * 10;
          steviloD = stevilo1D + stevilo2D + stevilo3D;
```

```
          if(pogoj2 == 1)
          {
           PORTC.B1 = 0;
           delay_ms(100);
           PORTC.B0 = 1;
           delay_ms(10000);
           PORTC = 0b00000000;
           delay_ms(1000);
           PORTC.B1 = 1;
           delay_ms(100);
           PORTC.B0 = 1;
           zakasnitevSD();
           PORTC = 0b00000000;
           delay_ms(1000);
           stevilosD = steviloD;
           pogoj2 = 0;
          }
           if (pogoj2 == 0)
           {
             if (steviloD < stevilosD)
               {
               enotaD = stevilosD - steviloD;
               stevilosD = steviloD;
               PORTC.B1 = 1;
               delay_ms(100);
               PORTC.B0 = 1;

               zakasnitevED();
                }
               if (steviloD > stevilosD)
               {
               enotaD = steviloD - stevilosD;
               stevilosD = steviloD;
               PORTC.B1 = 0;
               delay_ms(100);
               PORTC.B0 = 1;

                zakasnitevED();
                }
            PORTC = 0b00000000;
            delay_ms(1000);
           }

          }
        }
       }
     }
   }
  }
 }
 }
if(buffer2[0] == CC)
 {
  if(buffer2[1] == ZZ)
   {
```

```
if(buffer2[2] == AA)
 {
  if(buffer2[3] == LL)
   {
    if(buffer2[4] == UU)
     {
      if(buffer2[5] == AA)
       {
        if(buffer2[6] == LL)
         {
        if(buffer2[10] == ACK)
        {
       while (UART1_Tx_Idle() == 0)
        {
        }
        UART1_write(OO);
       while (UART1_Tx_Idle() == 0)
        {
        }
        UART1_write(KK);
        while (UART1_Tx_Idle() == 0)
        {
        }
        UART1_write(ACK);
        PORTC = 0b00000000;
        stevilo1L = buffer2[7] - 48;
        stevilo2L = buffer2[8] - 48;
        stevilo3L = buffer2[9] - 48;
        stevilo1L = stevilo1L * 100;
        stevilo2L = stevilo2L * 10;
        steviloL = stevilo1L + stevilo2L + stevilo3L;
        if(pogoj3 == 1)
        {
         PORTC.B3 = 0;
         delay_ms(100);
         PORTC.B2 = 1;
         delay_ms(10000);
         PORTC = 0b00000000;
         delay_ms(1000);
         PORTC.B3 = 1;
         delay_ms(100);
         PORTC.B2 = 1;
         zakasnitevSL();
         PORTC = 0b00000000;
         delay_ms(1000);
         stevilosL = steviloL;
         pogoj3 = 0;
        }
         if (pogoj2 == 0)
         {
           if (steviloL < stevilosL)
             {
             enotaL = stevilosL - steviloL;
             stevilosL = steviloL;
             PORTC.B3 = 0;
```

```
                    delay_ms(100);
                    PORTC.B2 = 1;

                    zakasnitevEL();
                     }
                    if (steviloL > stevilosL)
                    {
                    enotaL = steviloL - stevilosL;
                    stevilosL = steviloL;
                    PORTC.B3 = 1;
                    delay_ms(100);
                    PORTC.B2 = 1;
                     zakasnitevEL();
                     }
                   PORTC = 0b00000000;
                   delay_ms(1000);
                  }

                 }
               }
              }
             }
            }
           }
          }
          }
       if (Lastbyte2 == ACK)   // ali je prišel ACK ?
       {
         while (bufferCount2 < 100)
         {
          buffer2[bufferCount2] = 0;
          bufferCount2 = bufferCount2 + 1;
         }
         bufferCount2 = 0;
         Lastbyte2 = 0;
      // konec - ali je prisel nov znak ?
   } // konec neskončne zanke
}
}
```