Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Anton Perc

Modularni MIDI krmilnik

diplomsko delo

univerzitetni študijski program druge stopnje
računalništvo in informatika

Mentor: dr. Dušan Kodek

Ljubljana 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

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

Št. naloge: 01900/2013
Datum: 05.02.2013

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

Kandidat: **ANTON PERC**

Naslov: **MODULARNI MIDI KRMILNIK**
MODULAR MIDI CONTROLLER

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

MIDI standard je orodje, s katerim lahko računalnik uporabimo kot orodje za ustvarjanje in reproduciranje glasbe. V ta namen se uporabljajo različni MIDI krmilniki, ki so sposobni pošiljati MIDI signale. Na osnovi pregleda obstoječih krmilnikov zasnujte rešitev za modularni MIDI krmilnik, ki bo prilagodljiv in uporaben v velikem številu primerov. Izberite ustrezne komponente in izdelajte vso potrebno strojno in programsko opremo. Krmilnik preizkusite in podajte možne načine za izboljšave.

Mentor:
prof. dr. Dušan Kodek

Dekan:
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Anton Perc, z vpisno številko **63050075**, sem avtor diplomskega dela z naslovom:

Modularni MIDI krmilnik

S svojim podpisom zagotavljam, da:

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

V Ljubljani, dne 11. januarja 2011 Podpis avtorja:

Kazalo

Povzetek

Abstract

1 Uvod **1**

1.1 Obstoječe rešitve 3

1.2 Opis končnega izdelka 6

2 Teoretični del **9**

2.1 MIDI 9

2.2 Povezava s krmilnikom 10

2.3 LPC1343 12

2.4 Razvojno okolje 13

3 Strojna oprema **15**

3.1 Osnove 15

3.2 Jedro 17

3.3 Modul 20

4 Programska oprema **25**

4.1 Pregled 25

4.2 Inicializacija 26

4.3 A/D pretvorba 29

4.4 USB 30

4.5 Menjava MIDI Kanalov 33

4.6 Razhroščevalni način 34

5 Sklepne ugotovitve **37**

A Sheme vezij **39**

Povzetek

Namen tega dela je predstaviti realizacijo prototipa MIDI krmilnika s fleksibilno modularno strukturo. V prvih poglavjih je podano teoretično ozadje. Tukaj bomo tudi predstavili že obstojoče rešitve in postavili našo končno rešitev v njihov kontekst. Pri pregledu se bomo predvsem osredotočili na naprave, katerih lastnosti so zanimive iz našega stališča. S tem ozadjem bomo lažje razumeli, tako realizacijo same naprave kot tudi razloge, zakaj se je želja po taki napravi pojavila. V preostalemu delu pa se bomo osredotočili na samo tehnično realizacijo naprave, s stališča programske in strojne opreme. Sama naprava deluje na podlagi NXP LPC1343 mikrokrmilnika in je razdeljena na jedro in na več posamičnih modulov, ki jih lahko razvrščamo po potrebi.

Ključne besede: MIDI, krmilnik, USB, LPC1343, Cortex-M3

Abstract

The purpose of this work is to present a realization of a prototype MIDI controller device with a flexible modular structure. In the first few chapters the theoretical background is presented. Here we will also show devices that already exist and explain how our own solution compares. We will focus on the properties that are interesting for the scope of this work. With the help of theory, it will be easier to understand the inner workings of the device, as well as the motivation behind the need for such a device. The rest of the thesis concerns itself with the actual realization of the prototype, both from the hardware and software perspective. The prototype is based on an LPC1343 microcontroller and is divided into the core and several modules, which can be attached according to the needs of the end user.

Keywords: MIDI, controller, USB, LPC1343, Cortex-M3

Poglavje 1

Uvod

Z razvojem računalništva se je tudi glasba vsaj delno preselila iz svoje analogne domene v digitalno. Računalniki so postali koristno orodje za manipulacijo zvokov, od procesa snemanja glasbe do tudi samega procesa ustvarjanja. Tako so različni programi vsaj delno začeli nadomeščati veliko količino fizičnih naprav, od sekvenčnikov in sintetizatorjev zvoka do različnih avdio efektov.

Standardna računalniška periferija načeloma ni najbolj idealna za manipulacijo s takimi programi, to nalogo navadno opravljajo različni MIDI krmilniki. MIDI krmilnik je tako vsaka naprava, ki je sposobna pošiljati MIDI signale. Ena izmed glavnih razlik med krmilniki je razporeditev kontrolnih elementov po sami napravi. Razporeditev elementov dostikrat vpliva na to, za kakšne namene bo končna naprava uporabljena. Tako je MIDI krmilnik lahko uporabljen za ustvarjanje glasbe, kot preprosta mešalka, ali pa za DJ-anje, ki je proces predvajanja glasbe v živo. Različne računalniške programe se uporablja tudi za nastopanje v živo. Program tu predstavlja inštrument in bi bilo zanimivo, če bi lahko na krmilnik pomislili kot na napravo, ki ima sposobnost fizičnega prilagajanja sami aplikaciji na računalniku.

Sposobnost prilagajanja krmilnika grafičnemu vmesniku ne pride do izraza, ko operiramo s programi, ki imajo že vgrajeno konsistenco v sam uporabniški vmesnik (npr. Ableton Live). Ko pa operiramo s programi, ki sami

po sebi nimajo vnaprej definiranega vmesnika, (npr. PureData, Max/MSP, VST) pa pride sama fleksibilnost bolj do izraza.

1.1 Obstoječe rešitve

Na tržišču je trenutno mnogo MIDI krmilnikov. Kljub temu, da ima vsak svoje posebnosti, je smiselno, da se osredotočimo le na nekaj primerov, ki po svoji konstrukciji dobro predstavijo trenutno situacijo na trgu. Med pregledovanjem različnih ponudb opazimo, da se sami kontrolni elementi in postavitve le-teh med seboj minimalno razlikujejo. Sicer komercialni krmilniki ponujajo zelo veliko dodatnih možnosti. V naših opisih se bomo osredotočili predvsem na kontrolne elemente, ker je njihova razporeditev glavno vodilo pri izdelavi naše rešitve.

1.1.1 Behringer B-Control serija

Behringer je s svojo nizkocenovno B-Control serijo prodrl v trg MIDI krmilnikov. Sama serija je izredno vsestranska, saj je namenjena za uporabo z več različnimi programi, kot so npr. Cubase, Cakewalk, Logic Audio. Sama serija predstavlja glavne kontrolne elemente pri različnih MIDI krmilnikih, vrtljive gumbe, drsnike in stikala. Večina komercialnih krmilnikov se drži podobnih smernic, kar se tiče postavitve elementov. Omogoča tudi uporabo več MIDI kanalov in ima sposobnost komunikacije prek USB vmesnika, kot tudi prek standardnih MIDI vtičnikov [8].

Slika 1.1: Behringer BCR2000

1.1.2 Novation Launchpad in Akai APC serija

Krmilnika Launchpad in Akai APC sta dokaj drugačna, ju pa povezuje pomembna lastnost. Oba sta narejena za uporabo s programsko opremo Ableton Live. Obe napravi sta razviti s pomočjo podjetja Ableton in imata tako visoko integracijo s samo programsko opremo. Naprave lahko vseeno uporabljamo tudi za druge namene, saj se še vedno držijo MIDI standardov. V tem primeru sicer funkcionalnost, kot tudi postavitev samih kontrolnih elementov na napravi, ne pride toliko do izraza [9], [10].

Slika 1.2: Novation Launchpad

Slika 1.3: Akai APC40

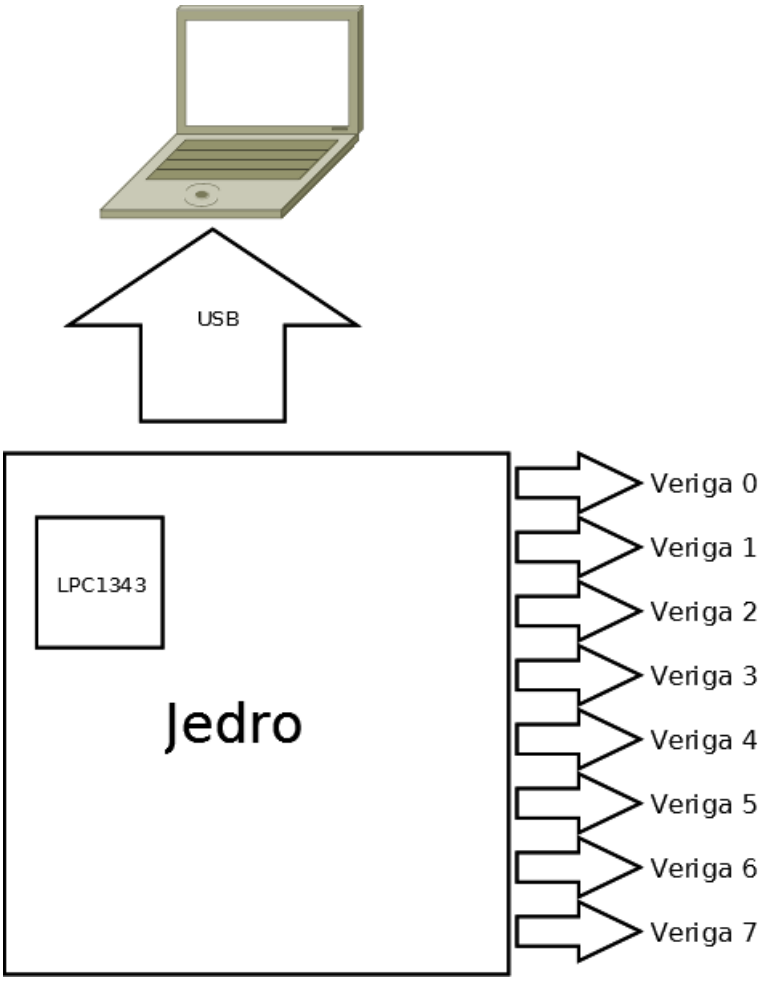
1.1.3 MIDIBox platforma

MIDIBox platforma je prostodostopni projekt. Cilj platforme je zagotoviti preprosto zgradbo MIDI krmilnikov in sicer preko zagotavljanja preprostih dokumentiranih modulov. Sama platforma je namenjena naprednim domačim uporabnikom, ki bi radi sami sestavili svoj MIDI krmilnik, ki bi odražal njihove potrebe [13].

Slika 1.4: Monodeck II MIDI krmilnik, zasnovan na MIDIBox platformi

Primer končne naprave, ki izvira iz MIDIBox platforme, je Monodeck II. Robert Henke, nemški glasbenik, je zaradi zelo specifičnih potreb po MIDI krmilniku sestavil svojega s pomočjo MIDIBox platforme. Sama naprava je nastala zaradi zelo specifičnih potreb po igranju v živo, ki jih trenutni komercialni MIDI krmilniki niso zadovoljili [12].

1.2 Opis končnega izdelka

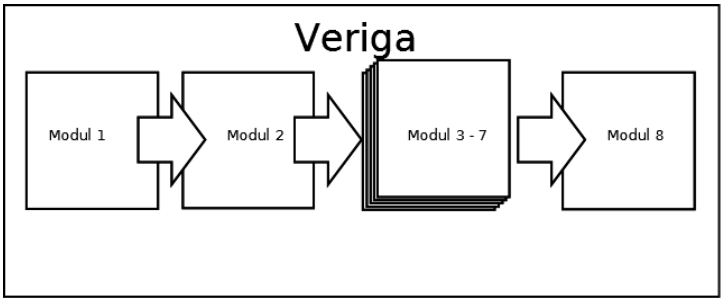


Slika 1.5: Bločna shema končnega izdelka

1.2.1 Zgradba

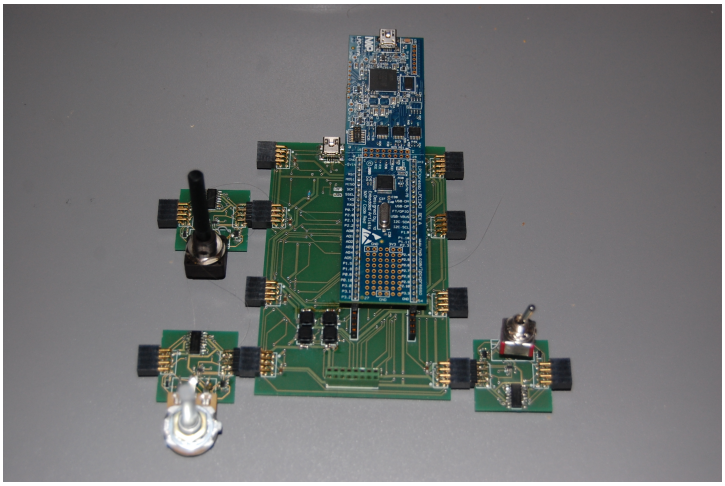
Naš MIDI krmilnik je sestavljen iz enega jedra in do 64 modulov. Na jedru imamo mikrokrmilnik, ki skrbi za komunikacijo preko USB vmesnika. Hkrati pa se na njem nahaja 8 vtičev. Na te vtiče lahko verižimo naše module. Dolžina vsake verige je lahko do 8 modulov. Modul predstavlja en kontrolni element s strojno logiko. Naprava tako nima vnaprej definirane razporeditve

različnih kontrolnih elementov. Veriga je lahko sestavljena iz 8 potenciometrov, 8 gumbov ali druge poljubne kombinacije modulov. Zaradi preprostega načina veriženja teh modulov lahko omogočimo končnemu uporabniku lastno postavitev kontrolnih elementov.



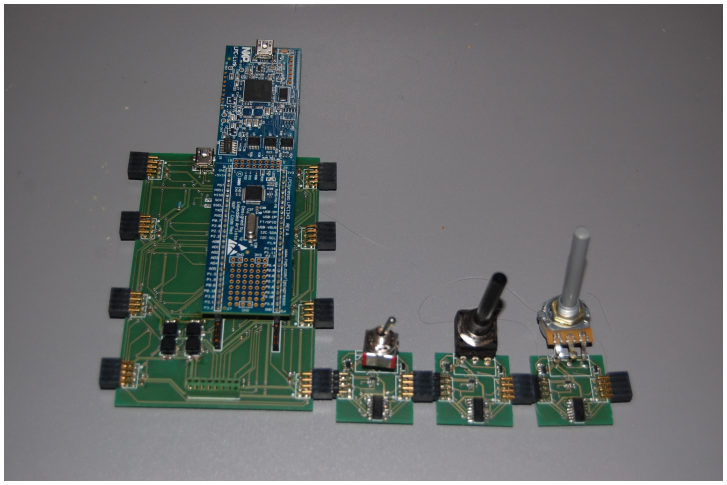
Slika 1.6: Bločna shema verige

Če primerjamo našo napravo z rešitvami, omenjenimi v prejšnem delu, bi lahko rekli, da imamo mrežo 64 elementov, ki je podobna Launchpadu. Hkrati pa imamo modularno sestavo MIDIBox krmilnika. Naši moduli so izredno poenostavljeni v primerjavi s tistimi pri MIDIBox krmilnikih. Cilj naših modulov je, da so med seboj zamenljivi in da lahko menjamo njihovo pozicijo po želji.



Slika 1.7: Končni prototip z razporejenimi moduli

Sam končni prototip predstavlja le vezje in je njegov namen da prikaže glavno funkcionalnost, ki smo jo zastavili s to nalogo.



Slika 1.8: Končni prototip z verigo treh modulov

Poglavje 2

Teoretični del

2.1 MIDI

MIDI (Musical Instrument Digital Interface) se je razvil iz potrebe po avtomatizaciji elektronske glasbene opreme. Z razvojem digitalne glasbene opreme se je pojavila potreba po komunikaciji med različnimi proizvajalci. Tako so leta 1985 objavili prvi MIDI standard, čigar namen je bil standardizirati izmenjavo informacij med različnimi elektronskimi napravami. Kratica MIDI vsebuje "musical", ampak sam standard pravi, da je njegov namen tudi komunikacija z napravami, ki niso direktno vezane na glasbo, npr. sekvenčniki, krmilniki za osvetljavo na koncertih. Standard narekuje fizično implementacijo in sam protokol, ki določa zgradbo paketov.

Del standarda, ki nam je bolj zanimiv, je protokol za komunikacijo med napravami. Sama MIDI sporočila se delijo na dva tipa, in sicer:

- **Sporočila kanalu (channel messages)**

  Sporočila kanalu se nanašajo samo na kanal. Pri MIDI krmilnikih so kanali koristni, da lahko ločimo fizične naprave, da ne pride do kontroliranja istega parametra iz večih naprav. Lahko pa tudi uporabimo tako, da z enim fizičnim elementom kontroliramo več parametrov v (naši) programski opremi. Ta sporočila so v našem primeru bolj zanimiva, saj se nanašajo direktno na parametre, s katerimi bi radi manipulirali.

- **Sistemska sporočila (system messages)**

  Ta sporočila se nanašajo na sam sistem, in posebnost je, da nimajo podatka, na kateri kanal se nanašajo, saj se nanašajo na vse kanale. Ta sporočila se delijo na: splošna, realno-časovna in ekskluzivna.

Predvsem je tukaj pomembno omeniti, da ker tehnično ne realiziramo krmilnika za operiranje z inštrumenti, temveč DAW (Digital Audio Workstation) krmilnika, je potrebno realizirati relativno majhen del standarda, v našem primeru so to le 'Change Control' sporočila. Standard pravi, da so kontrolna sporočila namenjena za spreminjanje tona z drugim kontrolnim elementom kot klaviaturno tipko [3].

2.2 Povezava s krmilnikom

Z razvojem različnih programov, ki služijo kot inštrumenti in tudi oprema za glasbeno produkcijo, se je razvila potreba po pošiljanju le teh signalov na računalnik. Te signale najlažje pošiljamo preko USB vmesnika. Signale pošiljamo po standardu, določenem v "Universal Serial Bus Device Class Definition for MIDI Devices", čigar prva verzija je bila izdana l. 1996.

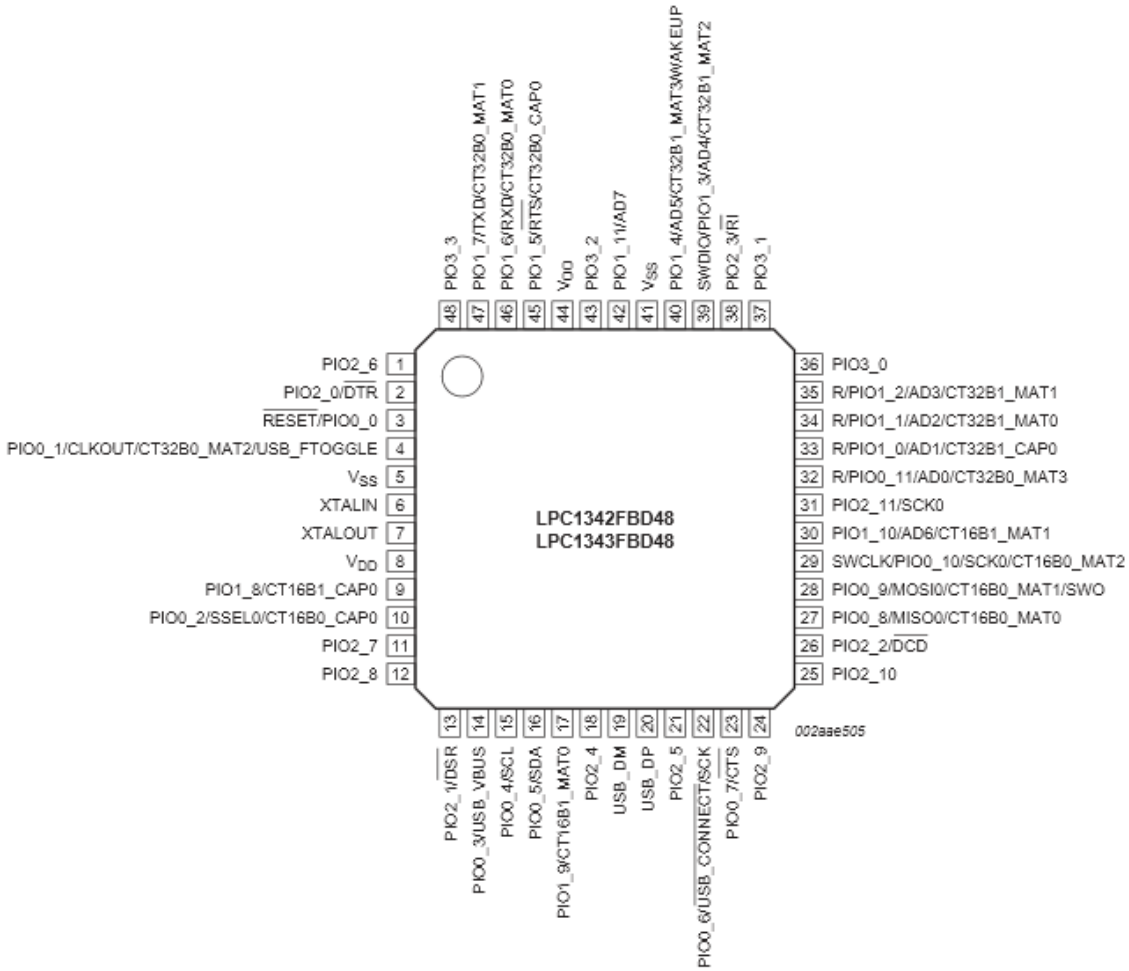
V naši nalogi bomo uporabili ta način komunikacije, saj je prisoten pri večini komercialno-dostopnih USB krmilnikov.

Preden začnemo govoriti o sami strukturi paketov, potrebnih za USB komunikacijo, bi bilo primerno, da najprej omenimo, kako se povezava med napravo in računalnikom vzpostavi. Temu procesu pravimo enumeracija naprave (ang. enumeration). Fizični USB vhodi so vezani na svoje središče (hub), ki ima več vlog. Ena izmed teh vlog je zaznavanje priklopov in izklopov zunanjih naprav. Vsako središče ima sposobnost obveščanja teh dogodkov gostitelju (host) preko prekinitvene končne točke (interrupt endpoint). Ko izvemo za novo napravo, pride do izmenjave informacij. Med izmenjavo gostitelj pošlje serijo povpraševanj (request) središču, ki vzpostavi komunikacijsko pot med gostiteljem in napravo. Gostitelj nato začne proces enumeracije, tako da pošlje standardna USB povpraševanja na končno točko 0

(endpoint 0), katero mora podpirati vsaka USB naprava. Da lahko rečemo, da je enumeracija uspešna, je potrebno, da naprava na vsako povprašanje pošlje nazaj ustrezne informacije. Po uspešni enumeraciji se naloži primerni gonilnik in naprava lahko komunicira z operacijskim sistemom [1].

Vsak USB MIDI paket je dolg 32 bitov ali 4 bajte. Pri USB prenosih so trije bajti z najmanjšo težo identični, kot so definirani po MIDI standardu. Standard pravi, da je MIDI sporočilo lahko krajše kot 3 bajte. V tem primeru neuporabljene bajte zapolnimo z bajti, ki imajo vrednost 0. Bajt z največjo težo (biti 31–24) določa številko kabla (Cable Number) in CIN (Code Index Number). CIN v tem primeru pove, kateri del MIDI standarda predstavljajo zadnji trije bajti [4].

2.3 LPC1343



Slika 2.1: Razpored Nožic LPC1343 mikrokrmilnika

LPC1343 je 32-bitni mikrokrmilnik, osnovan na arhitekturi ARM Cortex M3. Je del serije mikrokrmilnikov LPC, proizvajalca NXP. Namenjen je za uporabo v namenskih aplikacijah, ki potrebujejo visoko stopnjo integracije in nizko porabo [6]. V našem primeru uporabljamo razvojno ploščico OM11048, ki ima poleg procesorja še LPC-LINK, ki služi kot fizični razhroščevalnik. Sam LPC-LINK je razvit tako, da ga je možno uporabljati tudi z drugimi procesorji [5]. Na sliki 2.1 vidimo razporeditev nožic na našem mikrokrmil-

niku. Za našo nalogo so predvsem zanimivi A/D pretvorniki in podpora za USB.

2.4 Razvojno okolje

2.4.1 EAGLE

Pri načrtovanju vezij smo se odločili za uporabo programa EAGLE, ki je uveljavljen program, tako med hobijisti in tudi med ljudmi, ki razvijajo bolj kompleksna vezja. Program omogoča izvoz standardnih datotek, ki jih proizvajalci potrebujejo, da lahko natisnejo naša vezja. Prednost EAGLE programa je tudi, da imamo na voljo veliko izbiro prostodostopnih knjižnic in shematik za posamezne elemente, ki nam olajšajo delo pri načrtovanju. S tem se izognemo porabi časa za prerisovaje načrtov iz podatkov proizvajalca. Program je razdeljen v urejevalnik shem in načrtovalec tiskanih vezij. Ko smo zadovoljni z našo shemo, ki predstavlja logični načrt našega vezja, lahko te podatke prenesemo v načrtovalec tiskanih vezij in tam razporedimo elemente po vezju, preden jih natisnemo.

2.4.2 LPCXpresso IDE

Proizvajalec vezja NXP Semiconductors ima hkrati na voljo tudi razvojno platformo za svoje mikrokrmilnike iz LPC serije, sicer pod imenom LPCXpresso. Razvojno okolje je osnovano na odprtokodni Eclipse platformi, ki je prirejena za razvoj LPC vezij in omogoča hiter razvoj s pomočjo JTAG (Joint Test Action Group) standardiziranih razhroščevalnikov. Preko razvojnega okolja tudi nalagamo programe na sam krmilnik. Vgrajena je podpora za C/C++ in v osnovi podpira celotno linijo LPC mikrokrmilnikov [5].

Poglavje 3

Strojna oprema

3.1 Osnove

Pri produkciji elektronske glasbe je potreba po predefinirani kontrolni shemi dosti manjša. Kljub temu, da obstaja General MIDI standard, ki vnaprej definira inštrumente in njihove pozicije, vezane na MIDI protokol, se pogosto pojavlja potreba po tem, da signale vežemo za lastne potrebe (različni parametri na efektih, lastno predefiniranih inštrumentih...). Iz tega se je porajala ideja, da bi imeli MIDI krmilnik, čigar struktura odraža potrebe uporabnika. MIDI signale lahko kontroliramo na več načinov. Najbolj pogosti kontrolni elementi so:

- **stikala**

  Navadno se uporabljajo za vklop in izklop različnih efektov. MIDI standard narekuje, da se za vrednost OFF pošlje 0, za vrednost ON pa 127. Standard tudi narekuje, če naprava, ki prejme MIDI signal, pričakuje podatek o preklopu, naj vse vrednosti od 0–63 interpretira kot OFF, vrednosti od 64–127 pa kot ON [3].

- **gumbi (buttons, pressure pads, keys)**

  Podobno kot stikala, vendar se tukaj sprožita dva dogodka. Prvi se zgodi ob pritisku gumba in se pošlje NOTE ON podatek, ko gumb

spustimo, pa pošljemo NOTE OFF podatek. Pri določenih gumbih imamo tudi možnost, da ugotovimo, s kakšno hitrostjo je bila tipka pritisnjena. S pomočjo tega podatka dosežemo večjo izraznost zvoka.

- **vrtljivi gumbi (rotary potentiometers, encoders)**

  Kot že omenjeno vrednosti MIDI signalov lahko zavzemajo vredosti od 0 do 127, vrtljivi gumbi zadovoljijo potrebo po natančnem kontroliranju večvrednostnih parametrov.

- **drsniki (slide potentiometers)**

  Drsniki delujejo po istem principu kot vrtljivi gumbi, vendar imajo to prednost, da so vizualno bolj pregledni. V avdioinženiringu so popularni tudi zato, ker lahko relativno preprosto kontroliramo več parametrov hkrati. Uporabljajo se predvsem za kontroliranje zvoka, pogosto imajo drsni potenciometri logaritmični odziv, ker je ta odziv približno podoben našemu ušesu. Pri našemu krmilniku bomo uporabili take z linearnim odzivom, saj želimo enakomerno razdeljene MIDI vrednosti.

Vnaprej ne vemo, s kakšnimi parametri, efekti in inštrumenti, bodo imeli opravka z naši krmilniki. Zato želimo imeti bolj univerzalno rešitev, ki bo imela možnost se prilagajati končnemu uporabniku in ne obratno.

Trenutni komercialni krmilniki imajo razporeditev različnih kontrolnih elementov vnaprej definirano. Tako lahko med upravljanjem s programom pride do situacije, kjer zaradi neintuitivne razporeditve tipk pride do tega, da pozabimo, kaj določen element kontrolira. Ideja je tako, da bi te elemente lahko po lastni potrebi vezali med seboj in jih razporedili glede na naše potrebe.

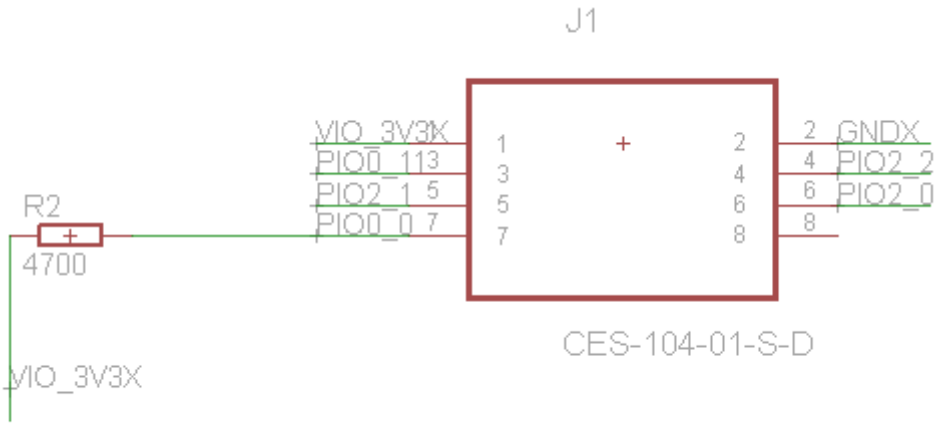
3.2 Jedro

3.2.1 Namen

Osrednji del naprave se nahaja v t.i. jedru, kjer se tudi nahaja naš krmilnik, oziroma v primeru našega prototipa, vtičnik, na katerega priklopimo razvojno ploščico. Služi kot vmesnik med samimi kontrolnimi elementi, ki se nahajajo v različnih modulih, in računalnikom. Glavna vloga je, da hrani naš krmilnik, in tako skrbi za ves programski del projekta.

3.2.2 Zgradba

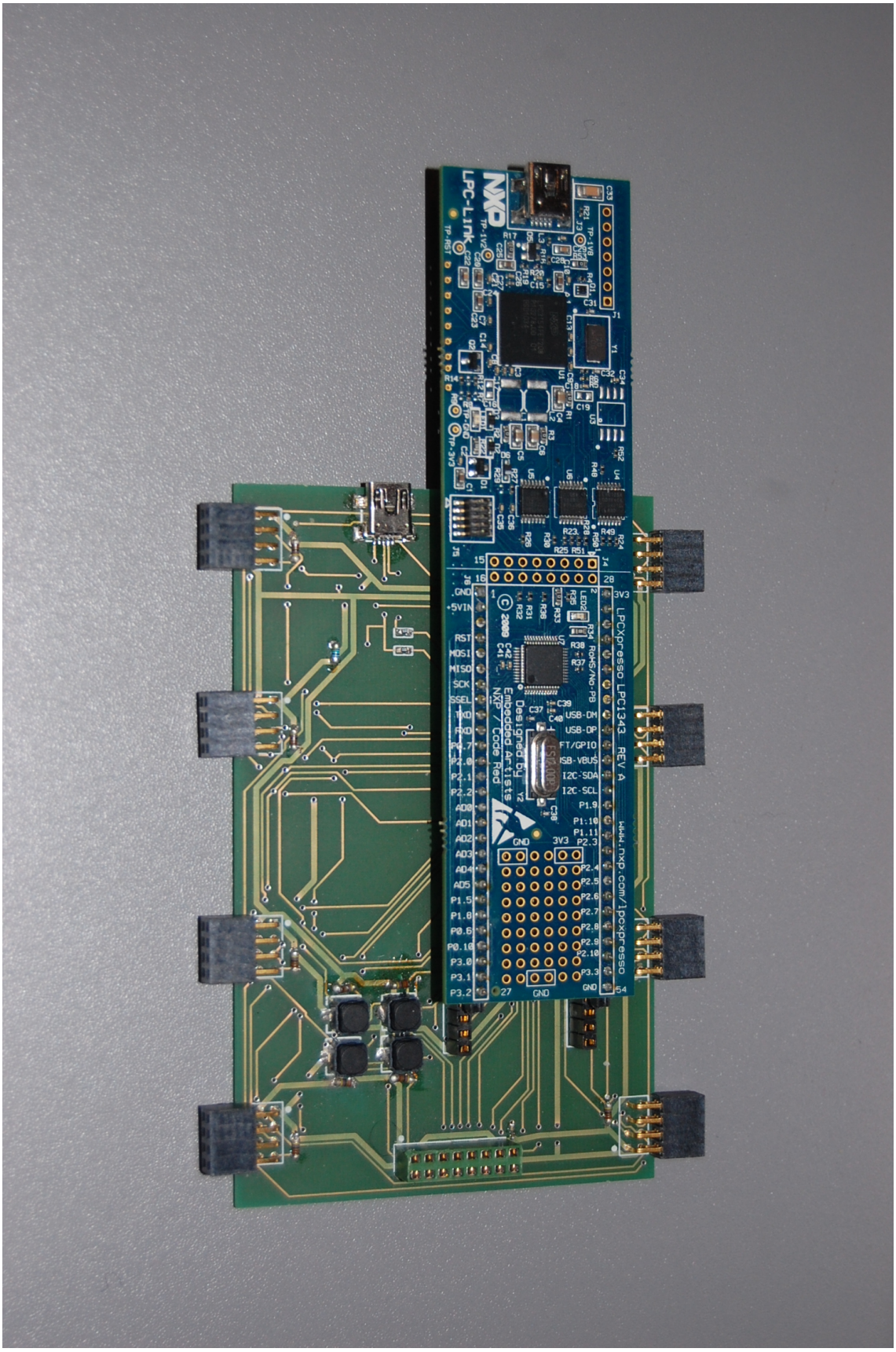
Jedro v osnovi ni preveč kompleksno. Na tiskanem vezju se nahaja vtičnik za razvojno ploščico. Na jedru je tudi 8 8-pinskih pravokotnih vtičnikov, ki služijo za priklop različnih modulov. Vsak vtičnik tako predstavlja eno verigo modulov. Na vsak vtičnik pripeljemo naslednje signale:



Slika 3.1: Izsek vezja za vtičnik na prvi poziciji

- **GND,** V_{cc}

  Napajalni liniji se nahajata na prvih dveh nožicah, in sicer uporabljamo 3.3V napetost, isto kot jo uporablja sam procesor.



Slika 3.2: Jedro z razvojno ploščico

- **Q**

  Na tretji nožici je analogni signal, ki ga pretvorimo na jedru.

- **SEL1, SEL2, SEL3**

  Signali, s katerimi izbiramo, kateri zaporedni modul je aktiven, torej SELECT linije, so na nožicah 4–6.

- **MOD ACTIVE**

  Zanima nas, ali je modul sploh prisoten. V primeru prisotnosti prožimo pretvorbo analognega signala v digitalnega, zato potrebujemo llinijo, na kateri preverimo prisotnost modula. V primeru, da je modul prisoten, je vrednost na liniji 0, v nasprotnem primeru pa je vrednost 1. To se doseže z uporabo dvigovalnih uporov, ki v primeru, da modul ni priklopljen, poskrbijo, da na vhodu ne pride do nedefiniranega stanja. S tem bi namreč lahko prebrali šum iz A/D pretvornikov in tako poslali napačne informacije.

Ker je jedro najbolj fiksni del naprave, se tu nahajajo elementi, ki imajo posebno vlogo, ki se ne veže nujno za upravljanje z MIDI signali.

Na jedru se nahajajo tudi 4 gumbi, katerih namen je menjava MIDI kanalov. Ko je gumb pritisnjen, se da na linijo napetost 3.3V, v ostalem primeru je napetost na liniji 0V. To je realizirano s pomočjo spustnih uporov, ki poskrbijo, da je stanje na liniji vedno definirano.

Na jedru se tudi nahaja mini USB vhod, čigar namen je, da lahko napravo priklopimo preko USB vhoda in tako komuniciramo z njo. Razlog, da potrebujemo še en USB vhod, poleg tistega, ki se nahaja na razvojni ploščici, je to, da nam LCPXpresso ne dopušča možnosti, da uporabljamo LPC-LINK za USB komunikacijo. Tu je vredno omeniti, da v našem primeru za napajanje uporabljamo LPC-LINK in ne USB vhod, tako da v vsakem primeru potrebujemo dva USB vhoda za uporabljanje naprave. Če bi imeli verzijo, ki je primerna za produkcijo, bi za napajanje uporabili USB vhod.

Če bi bilo napajanje preko USB vhoda, bi morali paziti še na nekaj dodatnih stvari. Paziti je treba, da zmanjšamo napetost na 3.3V. USB standard narekuje, da je napajalna napetost (V_{cc}) 5V, medtem ko naš mikroprocesor zahteva 3.3V. Mikrokrmilnik omogoča tudi SoftConnect funkcionalnost, ki jo trenutno lahko zanemarimo. Če bi prišlo do primera, da bi naša naprava potrebovala nekaj časa za inicializacijo, bi potrebovali SoftConnect. Le-ta nam omogoča da se naprava inicializira in šele nato sporoči računalniku svojo prisotnost. Tako se izognemo neprijetnemu primeru predolge inicializacije, kjer se zgodi, da se naprava ne odzove dovolj hitro in je posledično računalnik ne prepozna. USB komunikacija v tem primeru ne bi delovala.

3.3 Modul

3.3.1 Namen

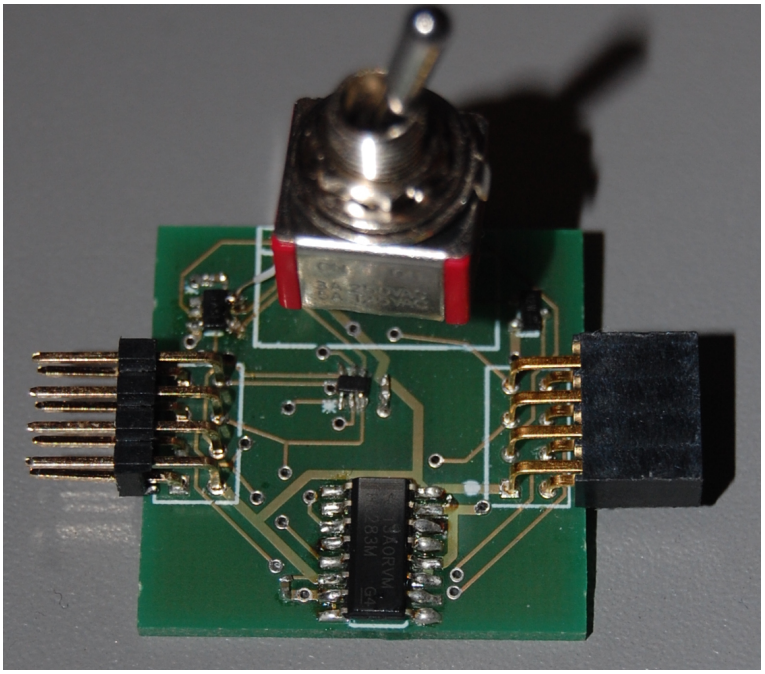
Moduli služijo kot fizični uporabniški vmesnik za upravljanje MIDI signalov. V našem primeru bomo uporabili nekaj standardnih elementov, vendar sama naprava ni omejena na le-te. Primarni namen je bil razviti dokaj univerzalno napravo, zato je pomembno, da dopustimo možnost dodajanja modulov, ki prvotno niso bili predvideni.

Primer modula, ki ga sicer nismo realizirali, je sledilna plošča. To je pravokotna površina, na kateri z dotikom lahko kontroliramo dva parametra hkrati, ki sta določena s koordinatami dotika. V našem primeru bi taka plošča sicer morala biti povezana na dva vtiča hkrati, vendar pa v sami realizaciji to ne bi predstavljalo problemov. Potrebno bi bilo le, da se koordinate na sledilni plošči pravilno pretvorijo v analogni signal, ki se potem pošlje v mikrokrmilnik.

Pri modulu je pomembno narediti konsistenten način komunikacije. Ker vsak modul upravlja z MIDI signali, je vse, kar je treba zagotoviti, da ima modul sposobnost pošiljati vrednost med 0 in 127 (2^7). Pri našem prototipnem vezju smo se sicer odločili, da se omejimo le na stikala in vrtljive gumbe. S tem nam ni treba paziti na dodatne parametre, kot je hitrost

pritiska tipke. Elementi, ki so najbolj pogosti (potenciometri), že sami po sebi pošiljajo analogni signal. Tako je najbolj smiselna odločitev, da je naša podatkovna linija v bistvu analogni signal, ki ga pretvorimo v samem mikrokrmilniku. Naš mikrokrmilnik že vsebuje 8 A/D pretvornikov, tako da ne potrebujemo dodatnih čipov. Sicer so analogni signali bolj občutljivi na šume. Naša resolucija pa je le 7 bitov in nam šum ne bi smel predstavljati neke večje ovire.

Modul ima tudi le enosmerno komunikacijo. Sodobni MIDI krmilniki sicer omogočajo dvosmerno komunikacijo, ker pa je naša želja ohraniti preprostost realizacije, naši moduli podpirajo le enosmerno komunikacijo.



Slika 3.3: Modul s stikalom

3.3.2 Naslavljanje modulov

Preden gremo v podrobnosti same zgradbe modula, je pomembno razložiti, kako sploh naslavljamo različne module. Pomembno je dejstvo, da modul sam ne ve nič o svoji poziciji na verigi. Vse, kar ve, so signali, ki jih dobi na

vhodnih vtičnikih. Tu se osredotočimo na 3 signalne linije, in sicer SEL_1 do SEL_3, saj ti signali določajo, kateri modul je izbran. V primeru, da bi bil ta signal nespremenjen skozi vse module, bi se zgodilo, da bi bili naslovljeni vsi moduli ali pa noben. Da lahko enolično določimo modul, je potrebno najti boljšo rešitev. Tu imamo več možnosti.

Prva možnost je, da ima vsak modul ročno nastavljen naslov, recimo s pomočjo mostičkov. V tem primeru je sicer veliko dela, da nastavimo vse module, in še vseeno moramo biti previdni, da nimamo dva modula vezana v isto verigo z istim naslovom. Naslove bi lahko nastavljali tudi programsko, in sicer tako, da bi se vsak modul ob priklopu registriral z jedrom.

Drugi način, in sicer tak, kot ga pa uporabljamo mi, je pa, da vsak modul spremeni naslov, in pošlje spremenjen naslov naslednjemu modulu. Postopek je v osnovi:

1. Jedro da na signalne linije SEL_1 - SEL_3 željeno pozicijo modula. Ker imamo le 3 naslovne linije, je potrebno poskrbeti, da nimamo več kot 8 zaporedno vezanih modulov.

2. Modul prebere vrednosti in v primeru, da so vrednosti vseh treh signalnih linij v visokem stanju, da na vodilo podatke.

3. Modul inkrementira vrednosti na SEL linijah in jih pošlje na lastni izhod.

Tako se SEL vrednosti pravilno propagirajo skozi vse module. Aktiven je hkrati lahko le eden, seveda ob upoštevanju omejitve, da lahko zaporedno vežemo le 8 modulov. V primeru, da bi dodali še deveti modul, bi se zgodilo, da bi dva modula hkrati postala aktivna, kar bi pa povzročilo nepravilno delovanje vezja.

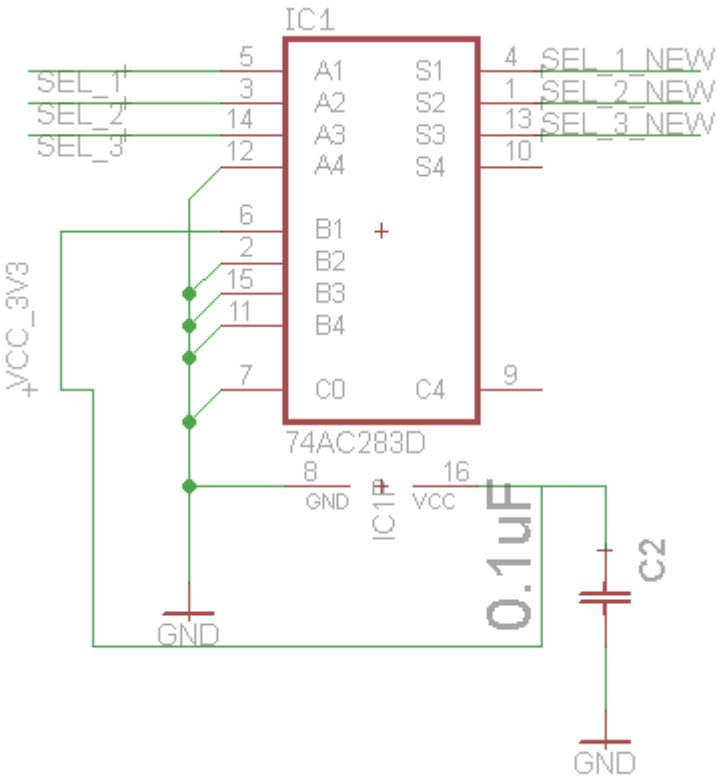
3.3.3 Zgradba

Signali iz jedra pridejo v modul preko vtičnikov, ki smo jih že omenili v prejšnjem delu. Za povezavo med moduli uporabljamo vtičnike pod pravim kotom, saj bi jih drugače težko verižili med seboj.

Signalne linije SEL_1 - SEL_3, se vežejo na NC7SZ11P6X [14]. To je čip, sestavljen iz 6 nožic; tri vhodne, ena izhodna in napajalne nožice. Sicer je 3-vhodno AND vezje. Prvotno je bil v vezju večji čip, ki je sicer opravljal isto vlogo. S časom se je pojavila potreba po manjših čipih, saj je bila površina na tiskanem vezju modula zelo omejena.

Izhodni signal tega vezja je vezan na tranzistor MOSFET-N in na analogno stikalo TS5A3166DBVR [15]. Prvotno je bil tu prisoten le en tranzistor, vendar se izkaže, da tranzistor ni dobra izbira za priklop analognih signalov na povratno podatkovno linijo. Tako se je pojavila potreba po analognem stikalu, ki tega problema nima. Analogno stikalo prejme signal iz AND vezja. V primeru, da je na liniji visoka napetost, analogno stikalo priklopi izhod kontrolnega elementa (npr. potenciometra) na povratno podatkovno linijo Q. MOSFET-N tranzistor pa v temu primeru služi kot signal, ki sporoča prisotnost našega modula, tako da jedro slučajno ne prebere šuma na nožici AD pretvornika.

Signalne linije se pošlje tudi na 4-bitni seštevalnik 74AC283 [16]. Na seštevalnik pripeljemo SEL_1 do SEL_3 linije in jim prištejemo vrednost 1, izhodne linije pa vežemo na izhodni vtič modula, na katerega lahko priklopimo naslednji modul v verigi.



Slika 3.4: Vezava 4-bitnega seštevalnika

Poglavje 4

Programska oprema

4.1 Pregled

Program je razdeljen v več delov. Najprej imamo postopek inicializacije, kjer nastavimo pravilno delovanje mikrokrmilnika. Med samim delovanjem pa skrbimo za pravilno pošiljanje signalov preko USB krmilnika, branje iz A/D pretvornikov in servisiranje prekinitev za menjavo MIDI kanalov. Primerno je, da najprej pogledamo psevdokodo. Tako si lažje predstavljamo osnovno delovanje programa, brez potrebe, da bi takoj razumeli podrobnosti delovanja našega mikrokrmilnika. Psevdokoda v našem primeru predstavlja le najbolj osnovni del programa.

```
initialization;
while 1 do
    foreach Address do
        foreach A/D Converter do
            if Module is present then
                oldValue = oldValues[ADConverter][Address];
                newValue = readCurrentADConverter;
                oldValues[ADConverter][Address] = newValue;
                if |newValue − oldValue| > 0 then
                    sendMidiSignal
                end
            end
        end
    end
end
```

Psevdokoda 1: Osnovni potek programa

Kot lahko vidimo, je program v osnovi le branje vseh A/D pretvornikov na vseh možnih pozicijah. V primeru, da je modul prisoten in da smo prebrali vrednost, ki se razlikuje od prejšne prebrane vrednosti, pošljemo nov MIDI signal.

4.2 Inicializacija

Preden lahko naprava pravilno deluje, je treba poskrbeti, da so stvari pravilno inicializirane. Prva funkcija, ki jo kličemo, je tako:

```
static __INLINE void init()
```

Beseda __INLINE je tukaj le makro, ki pretvori besedo v inline oznako, ki je del C99 standarda. To je v bistvu beseda, ki prevajalniku sporoči, da je klic te

funkcije možno optimizirati. To storimo tako, da nadomestimo klic funkcije s samo kodo v delu programa, kjer se nahaja klic. Klici funkcij so načeloma draga operacija in s tem lahko optimiziramo program. Funkcijo kličemo le enkrat ob inicializaciji, torej ni potrebno, da ima metoda v prevedenem programu svojo logično predstavitev.

Večino časa funkcija nastavlja vhode in izhode različnih nožic, da lahko naše vezje sploh pravilno deluje. To naredimo tako, da za vsaka logična vrata pravilno nastavimo vrednosti v smernem registru (GPIO Data Direction Register). Vsak bit v temu registru pove točno, za katero nožico gre. S temi registri manipuliramo s funkcijo GPIOSetDir(port, position, direction). V tabeli 4.1 vidimo, kako so različne nožice nastavljene. V tej metodi tudi vklopimo prekinitve, ki jih potrebujemo za realizacijo menjave MIDI kanalov.

Drugi del inicializacijskega postopka je pravilna nastavitev naših tabel. Prva tabela mod_table je dvodimenzionalna tabela, ki hrani podatke o zadnji pretvorbi. Pretvorb pred inicializacijo ni in zato nastavimo vse vrednosti na -1. Pozneje v programu primerjamo prejšnjo pretvorbo s trenutno. Pozitivne vrednosti bi tako povzročile potencialno napačno pošiljanje signalov preko USB vmesnika. V dvodimenzionalni tabeli last_step imamo pa lahko le vrednost 1 ali 0. Služi nam kot preprosta prediktorska tabela, v njej pa nastavimo vse vrednosti na 1. Obe tabeli sta velikosti 8 x 8. Vsaka pozicija v tabelah predstavlja pozicijo, na katero lahko postavimo modul.

Zadnji element samega inicializacijskega postopka je omogočanje A/D pretvornikov in USB vmesnika. Za A/D pretvornike je potrebno pravilno izbrati funkcionalnost različnih nožic na našem procesorju. Funkcionalnost izberemo s pomočjo vhodno-izhodnih nastavitvenih (IOCON) registrov. Vsaka nožica ima svoj register, v katerega zapišemo zaželjeno funkcionalnost. V našem primeru izberemo, da se nožica obnaša kot A/D pretvornik.

Za USB komunikacijo spet pravilno nastavimo nožice, ki jih uporabljamo. Vredno je omeniti, da naš krmilnik ne uporablja funkcije SoftConnect.

Št. Nožice	Funkcija	Opis
3	PIO0.0	Namen vhodnih linij je preverjanje prisotnosti modulov
4	PIO0.1	
10	PIO0.2	
15	PIO0.4	
16	PIO0.5	
22	PIO0.6	
23	PIO0.7	
27	PIO0.8	
32	AD0	A/D pretvorniki, ki služijo kot povratne signalne linije
33	AD1	
34	AD2	
35	AD3	
39	AD4	
40	AD5	
30	AD6	
42	AD7	
2	PIO2.0	Naslovne linije
13	PIO2.1	
26	PIO2.2	
36	PIO3.0	Vhodni signali za menjavo MIDI kanalov
37	PIO3.1	
43	PIO3.2	
48	PIO3.3	
14	VBUS	VBUS vhod, potreben za komunikacijo preko USB

Tabela 4.1: Tabela nastavitev funkcij nožic

4.3 A/D pretvorba

LPC1343 vsebuje 8 A/D pretvornikov. A/D pretvorniki lahko delujejo v prekinitvenem načinu. Prekinitveni način deluje tako, da po končani pretvorbi procesor dobi zahtevo po prekinitvi in vstopi v PSP (Prekinitveno servisni program). V PSP-ju preberemo vrednost pretvorbe iz pravega registra. Imamo pa tudi možnost izpraševalnega načina. V tem načinu programsko preverjamo, ali je pretvorba že končana. Naša naprava nima potrebe po prekinitvenem načinu, saj konstantno pretvarjamo vrednosti. To tudi predstavlja glavno nalogo našega mikrokrmilnika. Prekinitveni način bi v našem primeru dodal nivo kompleksnosti, po katerem ni potrebe.

Preden preberemo vrednosti iz vseh A/D pretvornikov, je potrebno pravilno nasloviti pravi zaporedni modul. Sam naslov hranimo v globalni spremenljivki *addr*. Pred vsako pretvorbo kličemo metodo *nextAddress()*. V tej metodi nastavimo vrednosti izhodov na nožicah od PIO0.0 do PIO0.2, ki služijo kot naslovne linije. To storimo tako, da vpišemo vrednost spremenljivke *addr* v podatkovni register za logični port 0. Ko je naslov zapisan, povečamo vrednost *addr* za 1, v primeru pa da je trenutna vrednost spremenljivke 7, pa jo postavimo na 0. Preden nadaljujemo s pretvorbo, vstavimo še nekaj urinih period, da se signal stabilizira skozi module in šele nato začnemo s pretvorbo.

A/D pretvorbo začnemo tako, da kličemo podprogram *readADC(channel)*. Ta podprogram zapiše v kontrolni register AD0CR, kateri kanal smo izbrali, in zapiše, naj se A/D pretvorba začne. Podprogram nato skoči v zanko, v kateri konstantno prebira statusni register AD0STAT, v katerega mikrokrmilnik zapiše, kdaj se je pretvorba zaključila. Potrdilo o končani pretvorbi povzroči, da se zanka zaključi. Register AD0DRn, kjer je n izbrani A/D kanal, hrani vrednost zadnje pretvorbe. To vrednost preberemo, podprogram pa jo vrne glavnemu programu, ki poskrbi za nadaljnjo obdelavo [11].

4.4 USB

4.4.1 Deskriptorji

USB deskriptorji so podatkovne strukture, preko katerih lahko računalnik izve več o sami napravi. Deskriptorji lahko hranijo podatke o celotni napravi ali pa le o posameznih delih naprave. Naši deskriptorji se nahajo v datoteki usbdesc.c. Prvi deskriptor, ki ga pošljemo, je deskriptor naprave (device descriptor), v katerem so osnovni podatki o napravi. V tem deskriptorju bi bilo smiselno omeniti parametra idVendor in idProduct. S pomočjo teh parametrov se operacijski sistem odloči za pravi gonilnik. Načeloma člani USB Implementers' Forum in tisti, ki jim plačajo administrativno ceno, uporabljajo unikatne vrednosti [1]. V našem primeru uporabljamo vendorId/productId par, ki ga podjetje Objective Development Software GmbH daje v prosto uporabo vsem posameznikom in podjetjem [2].

Za potrebe našega prototipa ne potrebujemo kompleksnih deskriptorjev. Pri nas zadostuje deskriptor, ki je podan kot primer v dokumentaciji za USB MIDI [4]. Da imamo veljaven deskriptor, ki opisuje MIDI napravo, je potrebno, da imamo deskriptor za avdio vmesnik, ki vsebuje vsaj en MIDIStreaming vmesnik. Vsebovati mora pa tudi MIDI IN in MIDI OUT vtičnik, ki predstavljata OUT in IN končno točko za USB [1].

4.4.2 MIDI paketi

Kot smo omenili že prej, se pri prenosu MIDI podatkov uporablja MIDI USB standard. V našem primeru uporabljamo zelo majhen del tega. Uporabljamo le ukaz za spremembo kontrol (Control Change). Sprememba kontrol lahko vključuje vse drsnike, gumbe, potenciometre..., kar je idealno za naše potrebe.

Preden pa lahko signal sploh pošljemo, moramo preveriti, da so vsi pogoji izpolnjeni. Imamo tri pogoje, katere moramo izpolniti in sicer:

1. **Modul je prisoten**

   Vsaka veriga modulov ima tudi svojo linijo, preko katere moduli sporočajo prisotnost. V primeru, da je logična vrednost na tej liniji 0, nam to naznanja, da je modul prisoten in da lahko nadaljujemo s pretvorbo.

2. **Vrednost se razlikuje od prejšnje**

   Drugi parameter, ki ga je potrebno upoštevati, je, da smo zaznali razliko med branji iz A/D pretvornikov. V nasprotnem primeru bi naša naprava konstantno pošiljala informacije, ki se niso spremenile, po čemer pa ni potrebe.

3. **Smer spremembe je skladna s smerjo zapisano v prediktorski tabeli**

   Ker lahko pride do manjših odmikov pri prebiranju A/D pretvornikov, lahko pride do problemov, da naprava konstantno pošilja signal, ki skače gor in dol za vrednost 1. To smo rešili tako, da smo uvedli koncept prediktorske tabele. V primeru, da je sprememba v isto smer kot prejšnja sprememba, se pošlje USB paket. Drugače pa spremenimo vrednost tabele, tako da odraža smer spremembe signala. Pri uporabi naprave v samem programu to ne povzroča večjih problemov.

Po izpolnjenih pogojih lahko zgradimo in pošljemo MIDI paket, ki se nahaja v spodnji metodi.

```
__INLINE void sendMIDIMessage(uint32_t adc_n) {
      uint8_t midi_data[] = {0x0B, 0xB0, 0x00, 0x00};
      midi_data[1] |= channel;
      midi_data[2] = adc_n * 8 + addr;
      midi_data[3] = mod_table[addr][adc_n];
      USB_WriteEP(0x81, &midi_data, 4);

}
```

Koda 1: Metoda, odgovorna za pošiljanje MIDI signala

Prvi element v tabeli midi_data je fiksna vrednost 0x0B. Vrednost določa, da se bodo podatki prenašali po kablu z indeksno številko 0. Spodnji štirje biti (0xB) pa določajo, da gre za spremembo kontrole. Prva pozicija se pojavlja le pri USB MIDI prenosih, ne pa pri navadnem MIDI protokolu.

Drugemu elementu nastavimo spodnje štiri bite na vrednost kanala. Ker ne nastavljamo vrednosti celega polja, moramo uporabiti OR operator, ki s pomočjo logične OR operacije nastavi spodnje štiri bite na vrednost kanala. Načeloma pri takih operacijah maskiramo bite, ki jih ne naslavljamo, saj bi lahko povozili zgornje štiri bite. V našem primeru pa vemo, da program lahko nastavlja le vrednosti od 0 do 3, do tega problema ne pride. Zgornji štirje biti pa določajo tip MIDI paketa, v našem primeru je to 'Control Change'.

Tretji element določa številko kontrolnega elementa. Paziti moramo, da enolično določimo vseh 64 možnih pozicij modulov. V nasprotnem primeru, bi se lahko zgodilo, da bi isti parameter lahko kontrolirali z večih pozicij na naši napravi, kar bi povzročilo nezaželjeno obnašanje. Lokacijo preprosto izračunamo s preprosto formulo, ki zagotavlja enolično razporejenost. Hkrati pa tudi poskrbimo, da ima vsak A/D pretvornik zagotovljenih 8 pozicij. V primeru, da bi želeli imeti več kot le 64 modulov, bi bilo potrebno spremeniti faktor množenja na 2^n, kjer je n število naslovnih linij. Vsak MIDI kanal ima

sicer lahko le 128 kontrolnih elementov, vendar bi lahko ta problem obšli s kreativno uporabo kanalov.

Na zadnji element pa zapišemo A/D pretvorbo za to lokacijo, ki je bila zapisana v mod_table tabelo že takoj po tem, ko smo prebrali A/D pretvornik.

Sedaj, ko je paket sestavljen, ga je potrebno še poslati preko USB vmesnika. LPCXpresso ima že obstoječo knjižnico za USB, ki nam omogoča pošiljanje in prejemanje podatkov preko USB vmesnika, zato uporabimo ukaz USB_WriteEP. Prvi parameter tega ukaza določa smer in številko končne točke (endpoint number). V bistvu le bit z največjo težo določa smer, torej sedmi bit. Vrednost tega bita je v našem primeru 1 in pomeni, da pišemo v to končno točko. Razlog, da je potrebno pošiljati smer komunikacije, kljub temu da vemo, da gre za pisanje, leži v implementaciji uporabljene knjižnice. Spodnji štirje biti imajo v našem primeru vrednost 0x1, kar pomeni, da gre za končno točko s številko ena. Ostala dva parametra sta naslov pomnilnika, kjer je paket shranjen, in dolžina paketa. Vsi naši paketi imajo isto lokacijo in isto dolžino, tako da teh parametrov ni potrebno nikoli spreminjati.

4.5 Menjava MIDI Kanalov

Gumbi za menjavo MIDI kanalov so vezani na spodnje štiri nožice na drugem logičnem portu (PIO3.0 - PIO3.3). V prototipu so uporabljeni gumbi, ki spustijo skozi tok, dokler so pritisnjeni navzdol. Naš mikrokrmilnik ob spremembi na vhodu sproži prekinitev. Prekinitve omogočimo tako, da pravilno nastavimo IE (GPIO Interrupt mask register) register, ki ga imajo vsaka logična vrata. Tako v IE register za logična vrata 3 nastavimo spodnje štiri bite na 1. Ti biti predstavljajo vhode, na katere smo pripeljali gumbe.

Sam PSP se nahaja v metodi PIOINT3_IRQHandler(). Razvojno okolje poskrbi, da naša metoda, ki se nahaja v datoteki main.c, prevzame prioriteto nad osnovnimi definicijami. V tem primeru je osnovna definicija za vse PSP neskončna zanka, v katero gre program, če pride do nepričakovane prekinitve. Ob vstopu v naš PSP je njegova prva naloga, da preveri, od kod je prišla

prekinitvena zahteva. Mi vemo, da je prišla iz tretjega logičnega porta, ne vemo pa točno, na kateri signalni nožici je bila zahteva. Ta podatek se nahaja v MIS (Masked Interrupt Status) registru. Preverimo vse štiri možne pozicije, da izvemo, od kod je prišla zahteva za prekinitev. Druga naloga je vezana na samo servisiranje prekinitev. Ko prekinitev servisiramo, je potrebno sporočiti, da je prekinitev bila servisirana. To naredimo tako, da v register IC (Interrupt Clear Register) vpišemo vrednost 1, seveda na pravo pozicijo.

Del PSP, ki je vezan na menjavo kanalov, je preprost. Pri MIDI signalih je kanal le parameter v poslanem USB paketu. Namen našega PSP je torej operiranje s spremenljivko:

```
volatile uint32_t channel;
```

Spremenljivka je tudi označena z besedo volatile, saj s tem naznanimo prevajalniku, da spremenljivko lahko spremeni nek zunanji dejavnik. V našem primeru je to prekinitveno servisni podprogram. V primeru, da spremenljivka ne bi bila pravilno označena, bi lahko prevajalnik zaradi optimizacije narobe prevedel program in naša naprava ne bi pravilno delovala.

4.6 Razhroščevalni način

Proti koncu lahko omenimo še razhroščevalni način. LPCXpresso omogoča t.i. semihosting način delovanja. Pri reševanju težav med razvojem si pogosto pomagamo tako, da izpisujemo željene podatke na zaslon. V našem primeru gre to za printf() ukaze. To lahko naredimo tako, da preko serijskega kabla pošiljamo ukaze. Semihosting način pa deluje drugače, in sicer prevajalnik vstavlja BKPT ukaze. Ko razhroščevalna orodja zaznajo ta ukaz, se izvajanje v procesorju ustavi in izvede zahtevano operacijo. Glavni problem tega načina je, da zelo upočasni izvajanje programa, in je tako koristen le med samim razvojem programa. V našem primeru to rešimo tako, da prevajalniku sporočimo, kateri deli programa so povezani z razhroščevanjem in kateri

ne. Naša rešitev vključuje, da imamo definirano vrednost DBG. S pomočjo prevajalnika preverjamo, ali je vrednost nastavljena na 1. V primeru, da je DBG nastavljen na pravilno vrednost, bo prevajalnik vključil dodatne ukaze, ki bodo v našem primeru izpisovali podatke v konzolo. Druga prednost tega je, da hkrati lahko izklopimo uporabe standardne <*stdio.h*> knjižnice jezika C, kar znatno poveča velikost samega programa, ki ga je potrebno naložiti na mikroprocesor.

Poglavje 5

Sklepne ugotovitve

Sam prototip je uspešen, naprava pošilja potrebne MIDI signale na računalnik, in brez problemov deluje v programu Ableton Live. Samo delo je osredotočeno predvsem na tehnično realizacijo, pri kateri ni bilo večjih problemov. S časom se je pojavilo vprašanje, ali naprava končnemu uporabniku dopušča dovolj fleksibilnosti in ali bi naprava sploh lahko prišla v do status praktične uporabe. Mnenja sem, da bi bilo smiselno dodati še nekaj funkcionalnosti, ki bi bolj izkoristile celotni MIDI protokol. Do teh ugotovitev sem prišel šele v končnih stopnjah dela, ko je bilo vezje že realizirano.

V začetku so bili cilji sicer drugače zastavljeni. Zaradi nepoznavanja samega procesa izdelovanja takega vezja se je prototipno vezje mogoče preveč približevalo željeni končni obliki, vsaj kar se tiče velikosti same naprave. Za prototipno vezjo bi bilo smiselno izdelati večje module, ki bi imeli bolj široko funkcionalnost. V primeru, da je delovanje prototipa uspešno, je lažje izdelati manjšo verzijo tega vezja z manjšimi komponentami. V primeru, da bi bilo delo že od začetka zastavljeno kot prototip, bi lahko dodali še dodatno funkcionalnost, ki bi razširila uporabnost same naprave.

V zaključku sem mnenja, da sicer naprava zadovoljuje začetno idejo, ki je bila zastavljena. Vendar s spoznavanjem procesa načrtovanj vezja, boljšim poznavanjem MIDI protokola in samega mikrokrmilnika, se je ta ideja dokaj spremenila. Sem tudi mnenja, da izbira LPC1343 mikrokrmilnika mogoče ni

najboljša, saj bi za naše potrebe lahko služil bolj preprost mikrokrmilnik.

Dodatek A

Sheme vezij

V dodatku so načrti vezij, zrisani s programom EAGLE. Prva dva načrta sestavljata jedro, tretji načrt pa modul.

vez 1

vez 2

vez 3

Literatura

[1] J. Axelson “USB Complete - Everything You Need to Develop Custom USB Peripherals, Third Edition”, Lakeview Research LLC, 2005.

[2] “Royalty-Free Non-Exclusive Use of USB Product-IDs”, 2009 Dostopno na:
https://secure.my-webspace.at/svn/ledtools/module/usbdrv/USBID-License.txt.

[3] “MIDI 1.0 Detailed Specification, Document Revision 4.2”, The MIDI Manufacturers Association, Los Angeles, 1995.

[4] G. Ashour, B. Brackenridge, O. Tirosh, M. Kent, G. Knapen “Universal Serial Bus Device Class Definition for MIDI Devices”

Dostopno na:
http://www.usb.org/developers/devclass_docs/midi10.pdf .

[5] “LPCXpresso”, Dostopno na: http://ics.nxp.com/lpcxpresso/.

[6] “LPC1311/13/42/43 Product data sheet”, NXP Electronics, 2012

Dostopno na:
http://www.nxp.com/documents/data_sheet/LPC1311_13_42_43.pdf.

[7] “The MIDI Specification”,

Dostopno na:
http://www.oktopus.hu/imgs/MANAGED/Hangtechnikai_tudastar/The_MIDI_Specification.pdf.

[8] "Behringer B-Control Series",

Dostopno na: http://www.behringer.com/EN/Category/USB-MIDI-Controllers.aspx?s=O100.

[9] "Akai APC40",

Dostopno na: http://www.akaipro.com/apc40.

[10] "Novation Launchpad",

Dostopno na: https://www.ableton.com/en/products/controllers/launchpad/.

[11] "LPC1311/13/42/43 User Manual", NXP Electronics, 2012

Dostopno na:
http://www.nxp.com/documents/user_manual/UM10375.pdf.

[12] "Monodeck II", R. Henke

Dostopno na:
http://www.monolake.de/technology/monodeck.html.

[13] "MIDIBox Hardware Platform", T. Klose

Dostopno na:
http://ucapps.de/mbhp.html.

[14] "UHS 3-Input AND Gate", Fairchild Semiconductors

Dostopno na:
http://www.fairchildsemi.com/ds/NC/NC7SZ11.pdf.

[15] "SPST Analog Switch", Texas Instruments

Dostopno na:
www.ti.com/lit/ds/symlink/ts5a3166.pdf.

[16] "High-Speed CMOS Logic 4-Bit Binary Full Adder with Fast Carry", Texas Instruments

Dostopno na:
http://www.ti.com/lit/ds/symlink/cd74hct283.pdf.