



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У
НОВОМ САДУ



Стефан Станивук

**Реализација и даљинско управљање
аудио системом на вишепроцесорској
платформи**

ДИПЛОМСКИ РАД
- Основне академске студије -

Нови Сад, 2013



КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:		
Идентификациони број, ИБР:		
Тип документације, ТД:	Монографска документација	
Тип записа, ТЗ:	Текстуални штампани материјал	
Врста рада, ВР:	Завршни (Bachelor) рад	
Аутор, АУ:	Стефан Станивук	
Ментор, МН:	др Јелена Ковачевић	
Наслов рада, НР:	Реализација и даљинско управљање аудио системом на вишепроцесорској платформи.	
Језик публикације, ЈП:	Српски / латиница	
Језик извода, ЈИ:	Српски	
Земља публиковања, ЗП:	Република Србија	
Уже географско подручје, УГП:	Војводина	
Година, ГО:	2013	
Издавач, ИЗ:	Ауторски репринт	
Место и адреса, МА:	Нови Сад; трг Доситеја Обрадовића 6	
Физички опис рада, ФО: (поглавља/страна/ цитата/табела/слика/графика/прилога)	7/31/0/4/15/0/0	
Научна област, НО:	Електротехника и рачунарство	
Научна дисциплина, НД:	Рачунарска техника	
Предметна одредница/Кључне речи, ПО:	Аудио систем, XMOS, xCORE, микроконтролер, ехо ефекат	
УДК		
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, ВН:		
Извод, ИЗ:	У раду је приказано једно решење аудио система заснованог на xCORE архитектури, компаније XMOS. Дати микроконтролер је повезан са Cirrus Logic развојном плочом преко I2S серијске спрете на којој се врши дигитално-аналогна конверзија. Микроконтролер са Twitter сервера добавља поруке које садрже ноте, производи музичке тонове и прослеђује их до аудио излаза.	
Датум прихватања теме, ДП:		
Датум одбране, ДО:		
Чланови комисије, КО:	Председник: др Илија Башичевић Члан: мр Жељко Лукач Члан, ментор: др Јелена Ковачевић	Потпис ментора



KEY WORDS DOCUMENTATION

Accession number, ANO:			
Identification number, INO:			
Document type, DT:	Monographic publication		
Type of record, TR:	Textual printed material		
Contents code, CC:	Bachelor Thesis		
Author, AU:	Stefan Stanivuk		
Mentor, MN:	Jelena Kovacevic, PhD		
Title, TI:	Implementation and remote control of audio system on multiprocessor platform.		
Language of text, LT:	Serbian		
Language of abstract, LA:	Serbian		
Country of publication, CP:	Republic of Serbia		
Locality of publication, LP:	Vojvodina		
Publication year, PY:	2013		
Publisher, PB:	Author's reprint		
Publication place, PP:	Novi Sad, Dositeja Obradovica sq. 6		
Physical description, PD: (chapters/pages/ref./tables/pictures/graphs/applications)	7/31/0/4/15/0/0		
Scientific field, SF:	Electrical Engineering		
Scientific discipline, SD:	Computer Engineering, Engineering of Computer Based Systems		
Subject/Key words, S/KW:	Audio system, XMOS, xCORE, microcontroller, echo effect		
UC			
Holding data, HD:	The Library of Faculty of Technical Sciences, Novi Sad, Serbia		
Note, N:			
Abstract, AB:	This paper describes one solution to the audio system based on xCORE architecture of XMOS company. Microcontroller is connected to the Cirrus Logic development board via I2S serial interface where digital to analog conversion is done. Microcontroller from Twitter servers downloads „tweets” that contain musical notes, generates musical tones and sends them to the audio output.		
Accepted by the Scientific Board on, ASB:			
Defended on, DE:			
Defended Board, DB:	President:	Ilija Basicovic, PhD	
	Member:	Zeljko Lukac, MSc	Menthor's sign
	Member, Mentor:	Jelena Kovacevic, PhD	

Zahvalnost

Zahvaljujem se svim dobrim i poštenim ljudima, koji odolevaju pritiscima teškog vremena.

SADRŽAJ

1.	Uvod.....	1
2.	Teorijske osnove.....	3
2.1	xCORE arhitektura.....	3
2.2	I2S serijska sprega.....	5
2.3	I2C serijska sprega	7
2.4	Twitter API	9
2.5	Muzički ton	9
3.	Koncept rešenja	11
3.1	Stvaranje muzičkih tonova.....	15
3.2	Eho efekat	17
4.	Programsko rešenje.....	19
4.1	Modul app_twsynthesizer.....	19
4.2	Modul module_i2c_master.....	23
4.3	Modul module_i2s_master	25
4.4	Modul module_wifi_tiwisl	25
4.5	Modul module_spi_master	26
5.	Ispitivanja i rezultati	28
6.	Zaključak.....	30
7.	Literatura	31

SPISAK SLIKA

Slika 2.1 Opšti prikaz sistema	3
Slika 2.2 Arhitektura jedne xCORE pločice	4
Slika 2.3 Primer prostih I2S sprega	5
Slika 2.4 Audio signali na linijama I2S sprega	6
Slika 2.5 Dijagram I2C komunikacije sa uspešno (gore) i neuspešno (dole) prenesenim podacima	8
Slika 2.6 Osam osnovnih tonova	9
Slika 3.1 <i>Slice Kit</i> razvojna ploča	11
Slika 3.2 <i>WiFi Slice</i>	11
Slika 3.3 <i>GPIO Slice</i>	12
Slika 3.4 XTAG (JTAG) adapter	12
Slika 3.5 CDB470xx razvojna ploča	12
Slika 3.6 Dijagram redosleda jedne sesije	14
Slika 3.7 Dijagram redosleda: Sesija isključivanja tona	15
Slika 3.8 Dijagram redosleda: Uključivanje echo efekta	18
Slika 5.1 Izgled xSCOPE alata	29

SPISAK TABELA

Tabela 3.1 Oznake nota/pauze.....	13
Tabela 3.2 Dužine trajanja nota/pauza.....	14
Tabela 3.3 Učestanosti tonova.....	16
Tabela 5.1 Razlika između audio odbiraka sa PC računara i xCORE mikrokontrolera	29

SKRAĆENICE

MCU	- <i>Microcontroller</i> , mikrokontroler
RISC	- <i>Reduced Instruction Set Computing</i> , procesor sa smanjenim skupom instrukcija
DSP	- <i>Digital Signal Processing</i> , digitalna obrada signala ili <i>Digital Signal Processor</i> , digitalni signal procesor
MIPS	- <i>Milion Instructions Per Second</i> , milion instrukcija u sekundi
I2S	- <i>Integrated Interchip Sound</i> , serijska komunikaciona sprega
I2C	- <i>Inter-Integrated Circuit</i> , serijska komunikaciona sprega
TDM	- <i>Time-Division Multiplexing</i> , prenos zasnovan na vremenskom multipleksiranju
CMOS	- <i>Complementary Metal–Oxide–Semiconductor</i> , generacija integrisanih kola
ACK	- <i>Acknowledgement</i> , potvrda o uspešnom prijemu podataka
NACK	- <i>NotAcknowledge</i> , potvrda o neuspešno primljenim podacima
API	- <i>Application Programming Interface</i> , aplikativna sprega
HTTP	- <i>HyperText Transfer Protocol</i> , protokol za prenos HTML stranica
JSON	- <i>Java Script Object Notation</i> , tekstualno bazirana notacija za razmenu podataka
TCP	- <i>Transmission Control Protocol</i> , protokol upravljanja prenosom podataka
JTAG	- <i>Joint Test Action Group</i> , standard za ispitivanje i otklanjanje grešaka namenskih sistema
SPI	- <i>Serial Peripheral Interface</i> , serijski sprežni sistem za periferije

GPIO

- *General-purpose input/output*, izvodi (pinovi) na integrisanim kolima koji se po potrebi mogu podešiti da budu ulazni, izlazni ili ulazno/izlazni

1. Uvod

U ovom radu je realizovan audio sistem zasnovan na xCORE višejezgarnom mikrokontroleru kompanije XMOS. Cilj zadatka je da se ispitaju mogućnosti fizičke arhitekture i da se pokažu ubrzanja višejezgarnog xCORE mikrokontrolera pri paralelnom izvršavanju operacija. Zahtevi predstavljeni u ovom radu su sledeći: omogućiti preuzimanje muzičkih nota preko poruka (engl. *tweet*) društvene mreže *Twitter*, omogućiti proizvodnju muzičkih tonova na osnovu preuzetih nota, preko I2S sprege preneti proizvedene audio odbirke do audio izlaza (u ovom slučaju je to *Cirrus Logic* CDB470xx razvojna ploča sa DSP-om CS470xx), pokretanje operativnog sistema *Cirrus Logic* razvojne ploče treba da inicira xCORE mikrokontroler, realizovati echo efekat, preko dostupnih tastera omogućiti uključivanje/isključivanje echo efekta kao i uključivanje/isključivanje tona audio izlaza (potrebne komande preneti preko I2C sprege).

Danas živimo u svetu informacionih tehnologija, s računarima velikih mogućnosti i ogromne moći obrade podataka. Bez računara se ne može zamisliti skoro ni jedna aktivnost u ljudskom životu. Neki su napravljeni da budu što moćniji bez obzira na cenu, da bi služili u zahtevnim naučnim i vojnim istraživanjima, a neki da služe za kućnu i poslovnu upotrebu, slabije moći obrade ali zato i jeftiniji. Druga kategorija računara je slabo prepoznatljiva, obično zato što krajnji korisnici nekad nisu ni svesni njihovog postojanja. To je tip računara koji je ugrađen u krajnji proizvod da bi njime upravljao. Ova vrsta proizvoda se naziva ugrađeni sistem (engl. *embedded system*), a ovi mali računari mikrokontroleri (engl. *microcontroller – MCU*)[1], pored kojih postoji i veliki broj raznih namenskih procesora.

Mikrokontroleri se danas koriste u svim sferama ljudske delatnosti, od potrošačke elektronike do automobilske i vojne industrije. Procenjuje se da preko 50% prodatih računarskih integrisanih kola čine upravo mikrokontroleri, i to zbog mogućnosti široke primene, lakog programiranja i jeftine izrade.[2]

Rad je organizovan u sedam poglavlja. U drugom poglavlju su opisane teorijske osnove: xCORE arhitekture, I2S i I2C serijskih sprega, *Twitter API*-ja i muzičkih tonova. U trećem poglavlju je predstavljen koncept rešavanja problema navedenih u zahtevima rada. U četvrtom poglavlju su izneti detalji programske realizacije na ciljnoj platformi. U petom poglavlju opisan je postupak ispitivanja i verifikacije datog rešenja. U šestom poglavlju je ukratko opisano šta je urađeno i šta se može dodatno unaprediti. Sedmo, poslednje poglavlje, sadrži spisak korišćene literature.

2. Teorijske osnove

U ovom poglavlju su izložene teorijske osnove gradivnih elemenata ovog rada, prikazane na slici 2.1: xCORE arhitektura, I2S serijska sprega, I2C serijska sprega, *Twitter API*, muzički ton.



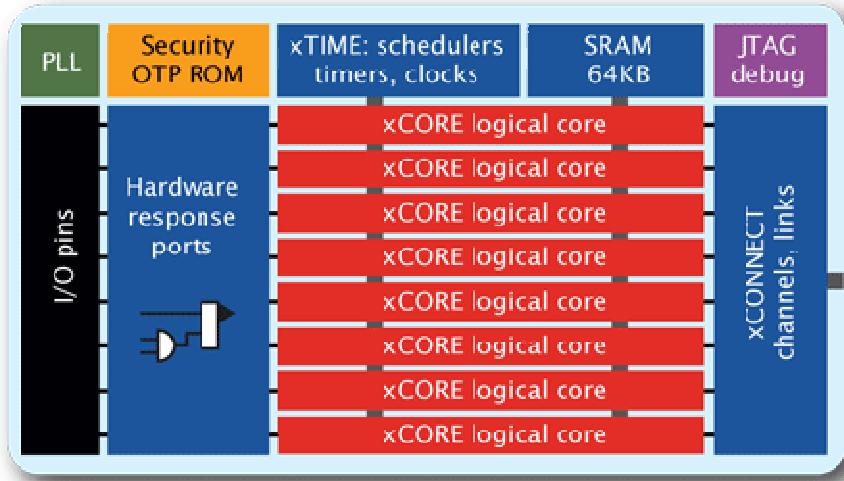
Slika 2.1 Opšti prikaz sistema

2.1 xCORE arhitektura

xCORE predstavlja novu klasu mikrokontrolera koja ima više procesorskih jezgara, izuzetno prilagodljiv ulaz/izlaz i jedinstven deterministički odziv, što ga čini izuzetno lakin za korišćenje. Koristeći xCORE se lako mogu projektovati novi ugrađeni elektronski sistemi koji će podržati sve želje krajnjeg korisnika sa tačno predvidivim odzivom.

Za razliku od konvencionalnih mikrokontrolera, xCORE višejezgarni mikrokontroler je sposoban da izvršava više zadataka u realnom vremenu, i to istovremeno. xCORE višejezgarni mikrokontroler sadrži 32-bitna jezgra koja podržavaju programski jezik visokog nivoa i koja izvršavaju program koristeći RISC instrukcije. Svako logičko procesorsko jezgro može izvršavati razne matematičke i DSP operacije, upravljati automatom stanja ili upravljati ulazom/izlazom.

xCORE višejezgarni mikrokontroler se sastoji iz više logičkih procesorskih jezgara (engl. *logical processor core*) koja su raspoređena po određenom broju pločica (engl. *tile*). Uređaji su trenutno dostupni sa 4, 6, 8, 10, 12, 16 ili 32 logička jezgra na 1, 2 ili 4 pločice. xCORE mikrokontroler obezbeđuje 500 MIPS procesne moći (na uređaju radne učestanosti 500 MHz) čineći ga tako mnogo moćnijim od konvencionalnih mikrokontrolera. Logička procesorska jezgra koja se nalaze na istoj pločici, dele memoriju i druge resurse te pločice, ali svako jezgro ima svoje odvojene registrarske informacije. Svako logičko jezgro dobija na raspolaganje garantovani deo procesne moći (do 125 MIPS na uređajima od 500 MHz), što je kontrolisano jedinstvenom tehnologijom, nazvanom xTIME. Vremenski odsečci se računaju na potpuno transparentan način – logička procesorska jezgra izgledaju kao odvojeni, paralelni procesori, sposobni za izvršavanje više zadataka u realnom vremenu, istovremeno. Logička procesorska jezgra komuniciraju međusobno preko kanala (engl. *channel*).



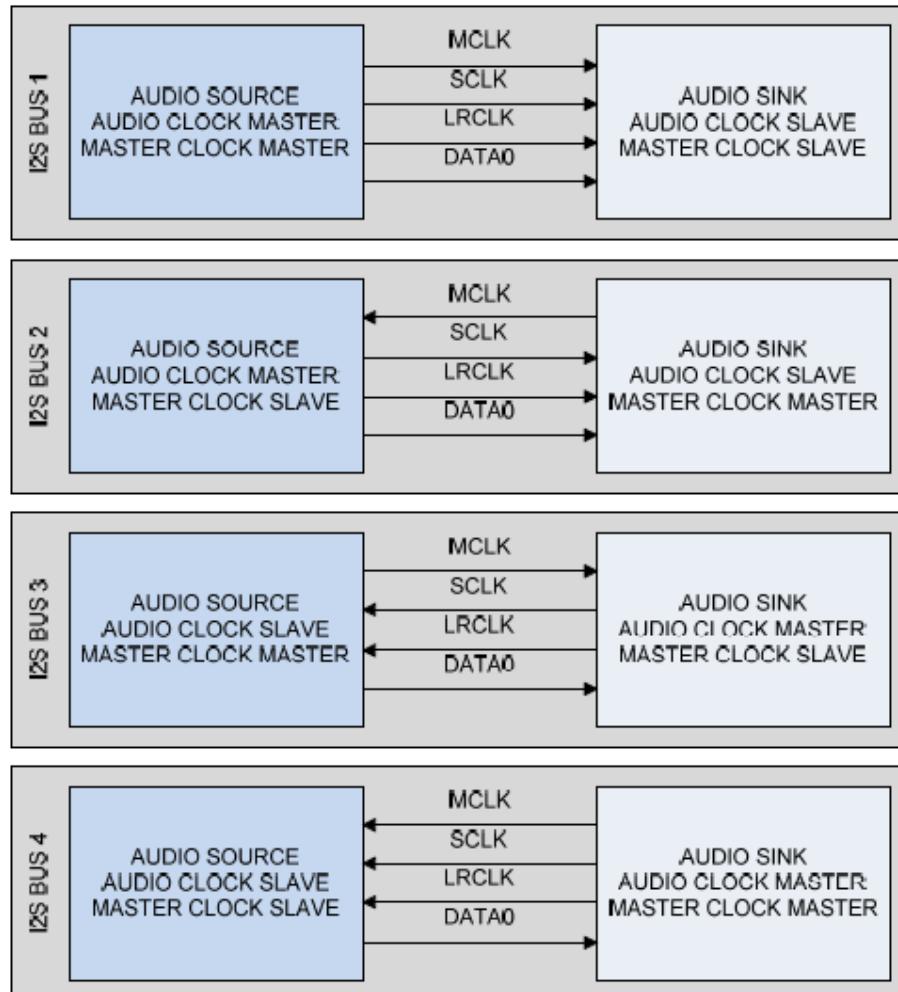
Slika 2.2 Arhitektura jedne xCORE pločice

xCORE višejezgarni mikrokontroleri se mogu programirati kako u asembleru tako i u C/C++ programskom jeziku. Da bi iskoristio sve prednosti xCORE arhitekture, XMOS je napravio jedno proširenje C programskog jezika i nazvao ga XC, koje podržava paralelne zadatke i kontrolu odziva.[3]

2.2 I2S serijska sprega

I2S protokol je standard koji je razvio Philips za digitalan prenos audio signala čime je omogućeno povezivanje integrisanih kola različitih proizvođača. U osnovi, I2S je TDM (engl. *Time Division Multiplexing*) serijski tok sa dva aktivna kanala. TDM je metoda prenosa više od jednog kanala preko jedne fizičke veze.

Tok audio podataka je isključivo u jednom smeru, ali signale takta može da vodi izvor (engl. *source*), odredište (engl. *destination, sink*) ili čak treći učesnik na magistrali (kontroler). Može da postoji i više odredišta, pod uslovom da se zadovolje električni zahtevi vodova i učesnika na magistrali. Svaka linija podataka tipično prenosi dva diskretna audio kanala, jedan (levi) u jednoj poluperiodi LR takta a drugi (desni) u drugoj.



Slika 2.3 Primer prostih I2S sprega

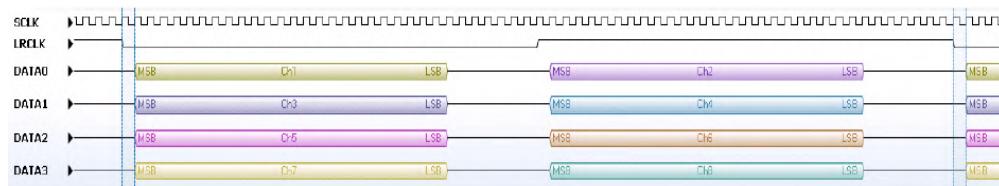
U većini primena koriste se tri linije takta i n linija podataka (gde je n veće ili jednako od polovine broja diskretnih audio kanala):

- Master takt (MCLK, Master Clock).
- Takt za izbor kanala (LRCLK, Left-Right Clock; WCLK, Word Clock).
- Takt bita (SCLK, Serial Clock; BCLK, Bit Clock).
- Linije podataka (DATAn).

MCLK je glavni takt u sistemu. Na osnovu njega se izvode skoro svi ostali taktovi, što obezbeđuje da sve komponente sistema budu sinhronizovane. Tipične učestanosti za master takt su 12,288 MHz, 24,576 MHz i 49,152 MHz.

SCLK je audio takt na čijim se aktivnim ivicama uzorkuju pojedinačni biti linija podataka. Tipične učestanosti su 3,072 MHz, 6,144 MHz i 12,288 MHz, najčešće 64x učestanost odabiranja (48 kHz, 96 kHz, 192 kHz). Da bi sprega bila kompatibilna sa uređajima audio preciznosti 24 i 32 bita po odbirku, a pošto se po jednoj liniji podataka I2S sprege prenose dva audio kanala, broj bita prenesenih u jedinici vremena koja odgovara jednom odbirku je 64.

LRCLK je audio takt koji odgovara učestanosti odabiranja, a služi da odredišni uređaji imaju informaciju o tome kom audio kanalu pripadaju biti koji se trenutno prenose na linijama podataka. Ako je učestanost odabiranja 48 kHz, LRCLK ima 48000 perioda a 96000 poluperioda u sekundi. Ovo je dovoljno da se prenese 48000 odbiraka za oba kanala na liniji podataka, ako se jedan kanal prenosi u nižoj a drugi u višoj poluperiodi.



Slika 2.4 Audio signali na linijama I2S sprege

Propisani parametri su:

- Odbirci se šalju od bita najveće do bita najmanje važnosti.
- Bit najveće važnosti se šalje jedan period SCLK takta nakon promene LRCLK takta (slika 2.3).
- Biti na linijama podataka mogu da budu poravnati bilo na rastuću, bilo na opadajuću ivicu SCLK takta.
- Biti na linijama podataka se uzorkuju isključivo na rastuću ivicu SCLK takta.
- Kada je LRCLK na niskom naponskom nivou, prenosi se prvi (levi) kanal.

- LRCLK može da se menja bilo na opadajuću, bilo na rastuću ivicu SCLK takta, ali mora da se uzorkuje na rastuću.
- LRCLK ne mora da bude simetričan.

Varijacije koje se najčešće sreću su po pitanju pozicije audio podataka u LRCLK okviru:

- Left-justified (podaci poravnati na levo).
- Right-justified (podaci poravnati na desno).
- I2S (podaci kasne za jedan SCLK ciklus, kao po standardu).

2.3 I2C serijska sprega

I2C je serijska komunikaciona sprega koja podržava sprezanje više vodećih (engl. *master*) i više pratećih (engl. *slave*) uređaja u jedan komunikacioni sistem. Ova sprega je definisana od strane firme Philips sa ciljem sprezanja perifernih uređaja sa centralnim procesorima, niskom brzinom komunikacije – tipično 100 kbps.

Koristeći svega dve linije komunikacije, I2C omogućava jednostavno povezivanje na fizičkom nivou velikog broja uređaja, do 128 u osnovnoj verziji I2C standarda.

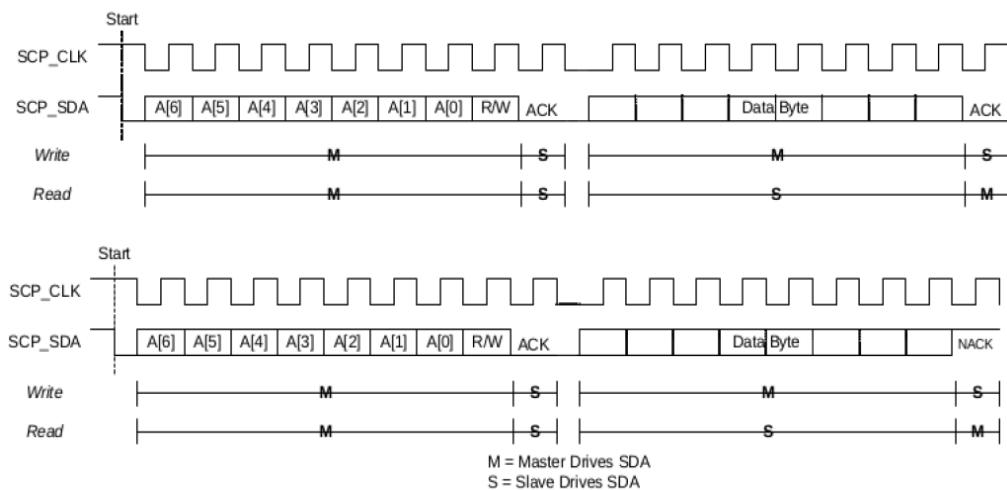
Ovo jednostavno sprezanje računarskog sistema je osnovna prednost I2C magistrale, te se I2C prvenstveno koristi za inicijalizaciju, podešavanje računarskog sistema, te dijagnostiku i prikupljanje informacija o stanju sistema kao i za akviziciju podataka gde se ne zahteva velika brzina komunikacije.

Komunikacija se na fizičkom sloju obavlja koristeći svega dve linije komunikacije: SDA (Serial Data) i SCL (Serial Clock). Koristeći izlaze sa otvorenim kolektorom (*open drain* za CMOS integrisana kola) više uređaja može istovremeno pristupati linijama komunikacije. Kolizija, momenat kada dva uređaja definišu različita logička stanja na linijama sprege, je u I2C spregi iskorišćena kao koristan događaj za definisanje posebnih stanja u komunikaciji, kao npr. definisanje ACK ili NACK signala.

U sprezi učestvuju dva tipa uređaja: vodeći uređaji i prateći uređaji. Vodeći je onaj uređaj koji definiše takt komunikacije i koji proziva adrese pratećih uređaja. Prateći je onaj koji prima takt komunikacije i koji sluša prozvane adrese.

Za komunikaciju se koristi 7-bitno ili 10-bitno adresiranje, pri čemu je 10-bitno adresiranje izuzetak, prisutno samo u specifičnim implementacijama I2C spregi.

I2C komunikacija se sastoji od sekvene događaja: Start marker, prozivanje adrese pratećeg uređaja, R/W (Read/Write) bit, ACK/NACK bit, prenos podataka, ACK/NACK bit:



Slika 2.5 Dijagram I2C komunikacije sa uspešno (gore) i neuspešno (dole) prenesenim podacima

Start marker je događaj definisan opadajućom ivicom SDA signala, dok je SCL (SCP_CLK na Slika 2.4) u logičkom stanju 1. Start marker signalizira ostalim vodećim uređajima (ako su prisutni u sprezi) da je sprega zauzeta za komunikaciju.

Stop marker je definisan rastućom ivicom SDA signala, dok je SCL signal u stanju logičke 1. Stop marker signalizira ostalim vodećim uređajima (ukoliko su prisutni u sprezi) da je sprega slobodna za novu komunikaciju. Start i Stop markeri se uvek stvaraju od strane vodećeg uređaja (master).

Nakon start markera sledi adresa. Svi uređaji koji su povezani jednom I2C spregom moraju imati jedinstvenu adresu u toj sprezi.

R/W bit definiše režim pristupa pratećem uređaju. Stanje logičke 1 ovog bita definiše režim pristupa za čitanje podataka, i obratno za upis.

ACK/NACK je bit kojim prateći uređaj signalizira uspešno prepoznatu adresu i dekodovane informacije. Stanje logičke 0 označava ACK (uspešan prijem), obratno NACK (neuspescan prijem). Veličina prenesenih podataka koji slede definisanu adresu i R/W bit nije ograničena.

Ukoliko je aktivovan režim pristupa za čitanje, vodeći uređaj je dužan da definiše ACK/NACK bit posle polja podataka. Tj. onaj koji prima podatke definiše da li su podaci uspešno primljeni. ACK/NACK bit koji sledi adresno polje je uvek definisan od strane prozvanog pratećeg uređaja. Ukoliko prozvana adresa nije validna, ACK/NACK bit ostaje na vrednosti logičke 1, tj. , NACK.[4]

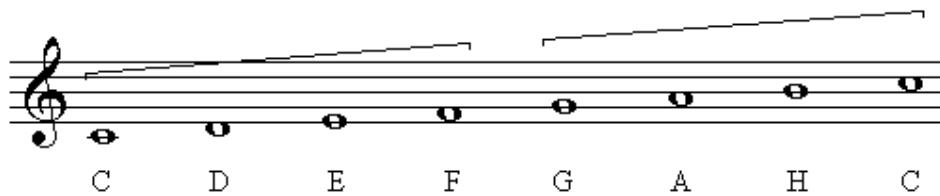
2.4 Twitter API

Twitter, kao društvena mreža koja okuplja veliki broj ljudi, je postala izuzetno zanimljiva i programerima, koji žele da povežu svoje aplikacije i veb sajtove sa *Twitter*-om iz raznih razloga. Da bi razne funkcionalnosti i usluge *Twitter*-a bile lako dostupne programerima, napravljen je *Twitter API*. Ovaj API predstavlja spregu između aplikacije (klijenta) i same društvene mreže (servera). *Twitter API* koristi HTTP protokol za komunikaciju sa klijentima, što znači da radi na principu zahteva (engl. *request*) i odgovora (engl. *response*) nakon uspostavljene TCP veze. Klijent posle zahteva, dobija odgovor u JSON formatu.[5]

2.5 Muzički ton

Ton nastaje pravilnim, ravnomernim treperenjem zvučnog izvora. Ton je zvuk koji je određen svojom učestanošću (broj treperenja u sekundi), amplitudom (veličina treperenja), trajanjem (koliko dugo izvor tona treperi) i bojom (zavisi od vrste materijala i oblika zvučnog izvora).

Ljudsko uho je u proseku sposobno da čuje zvukove u rasponu od 20 do 20000 Hz dok se u muzici najčešće koriste tonovi sa osnovom 30 – 8000 Hz, što ostavlja prostora za preko sto opštekorušenih tonova u klasičnoj teoriji muzike. Konvencija je da se ovi tonovi dele u oktave, od kojih svaka sadrži po 12, za čije se imenovanje koristi sedam imena: do, re, mi, fa, sol, la, si čije su opšteprihvaćene oznake: C, D, E, F, G, A, H, odnosno C, D, E, F, G, A, B.



Slika 2.6 Osam osnovnih tonova

Svako od ovih imena je dodeljeno egzaktnom tonu u oktavi. Time je već pokriveno sedam od dvanaest tonova koje sadrži jedna oktava. Ostali tonovi se mogu označiti na više načina, naznačavanjem da je neki od prethodno navedenih tonova viši ili niži za pola tona (koristi se izraz polustepen) nego što to njegovo ime naznačava.

Tonovi se ciklično ponavljaju po oktavama, tj. (jedna oktava) do, re, mi, fa, sol, la, si, (sledeća oktava) do, re, mi, fa, sol, la, si, itd.

Učestanosti muzičkih tonova slede princip po kome je odnos učestanosti između dva istoimena tona susednih oktava 1:2 odnosno 2:1, u zavisnosti od toga koji je viši a koji niži.[6]

3. Koncept rešenja

Rešenje je realizovano povezivanjem dve razvojne ploče:

- *Slice Kit* razvojna ploča kompanije XMOS (sadrži 2 pločice sa po 8 logičkih jezgara na svakoj, slika 3.1), sa dodacima za: bežičnu vezu (slika 3.2), proširenje sa dodatnim pinovima i tasterima (slika 3.3) i otklanjanje grešaka (XTAG adapter, slika 3.4) - kao glavni mikrokontroler u sistemu.
- CDB470xx razvojna ploča kompanije *Cirrus Logic* sa DSP-om CS470xx - kao audio izlaz (slika 3.5).



Slika 3.1 *Slice Kit* razvojna ploča



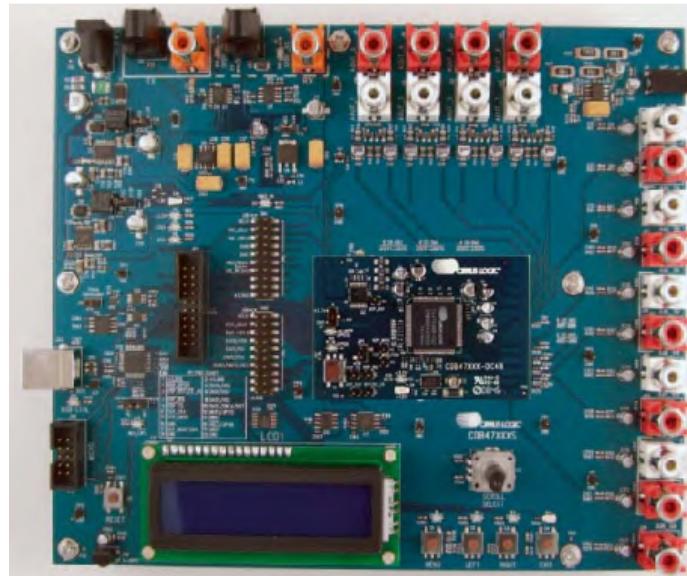
Slika 3.2 *WiFi Slice*



Slika 3.3 GPIO Slice



Slika 3.4 XTAG (JTAG) adapter



Slika 3.5 CDB470xx razvojna ploča

Razvojne ploče treba da komuniciraju preko I2S sprege, za prenos audio odbiraka, kao i preko I2C sprege, za uključivanje i isključivanje tona. Kao što je gore već objašnjeno, I2S sprega može da se koristi u više režima. Ovde je korišćen režim gde, vodeći uređaj u smislu celokupnog sistema (u ovom slučaju je to xCORE mikrokontroler), zahteva master takt od pratećeg uređaja (u ovom slučaju je to CS470xx DSP). xCORE mikrokontroler na osnovu primljenog master takta proizvodi sve ostale taktove vezane za I2S spregu. xCORE kao glavni mikrokontroler u sistemu predstavlja vodeći uređaj u radu I2C sprege, dok CS470xx DSP koji prima komande preko I2C sprege predstavlja prateći uređaj.

xCORE mikrokontroler na početku sesije treba da inicira pokretanje operativnog sistema na CS470xx DSP-u (postavljanjem odgovarajuće vrednosti na *HS* pinove za vreme trajanja *RESET* signala), a zatim da se poveže sa dostupnom pristupnom tačkom (engl. *Access Point*) uz pomoć *WiFi slice* dodatka (slika 3.2). Posle pokretanja sistema DSP-a, DSP prenosi master takt do xCORE-a, i time omogućuje stvaranje svih ostalih I2S taktova. Preko pristupne tačke, xCORE ima pristup svetskoj mreži, pa samim tim i *Twitter* serveru. Sa serverom pravi TCP vezu radi prenosa korisničkih poruka (engl. *tweets*). Posle inicijalizacije sistema, xCORE pravi HTTP zahtev (gde zahteva korisničke poruke sa određenom oznakom - engl. *hashtag*) i šalje ga *Twitter* serveru, koji na osnovu primljenog zahteva šalje nazad odgovor u obliku JSON poruke. Prozivanje *Twitter* servera se ponavlja ciklično posle određenog vremena.

xCORE kad primi JSON poruku, treba da je parsira da bi pronašao relevantne podatke, a relevantni podaci su korisničke poruke koje sadrže oznaku #XMOSTwSynth i to samo one koje su nove, tj. koje nisu pre bile svirane.

Vreme potrebno za stvaranje muzičkih tonova i sviranje može dovesti do zatvaranja TCP veze zbog neaktivnosti. Zbog toga je uvedena tehnika duplih bafera (engl. *double buffering*) koja omogućuje istovremeno izvršavanje niti vezanih za stvaranje muzičkih tonova i sviranje (prenos audio odbiraka preko I2S sprege do CS470xx DSP-a) i niti za prihvatanje JSON poruka i njihovo parsiranje. Jedan memorijski niz se koristi za iščitavanje nota i stvaranje muzičkih tonova a drugi za skladištenje isparsiranih nota. Izmena nizova se obavlja kada je sadržaj niza koji služi za čitanje nota iščitan odnosno odsviran.

Tonovi su označeni kao na slici 2.5, s tim da je drugo C označeno sa *m* a pauza sa *p*:

Note + pauza	Oznake nota/pauza, za korišćenje u poruci
Do	c
Re	d
Mi	e
Fa	f
Sol	g
La	a
Si	h
Do	m
Pauza	p

Tabela 3.1 Oznake nota/pauze

Podržane dužine trajanja nota/pauza su:

Dužina trajanja nota/pauza	Oznake dužina trajanja nota/pauza, za korišćenje u poruci
cela nota/pauza	1
1/2	2
1/4	4
1/8	8
1/16	6

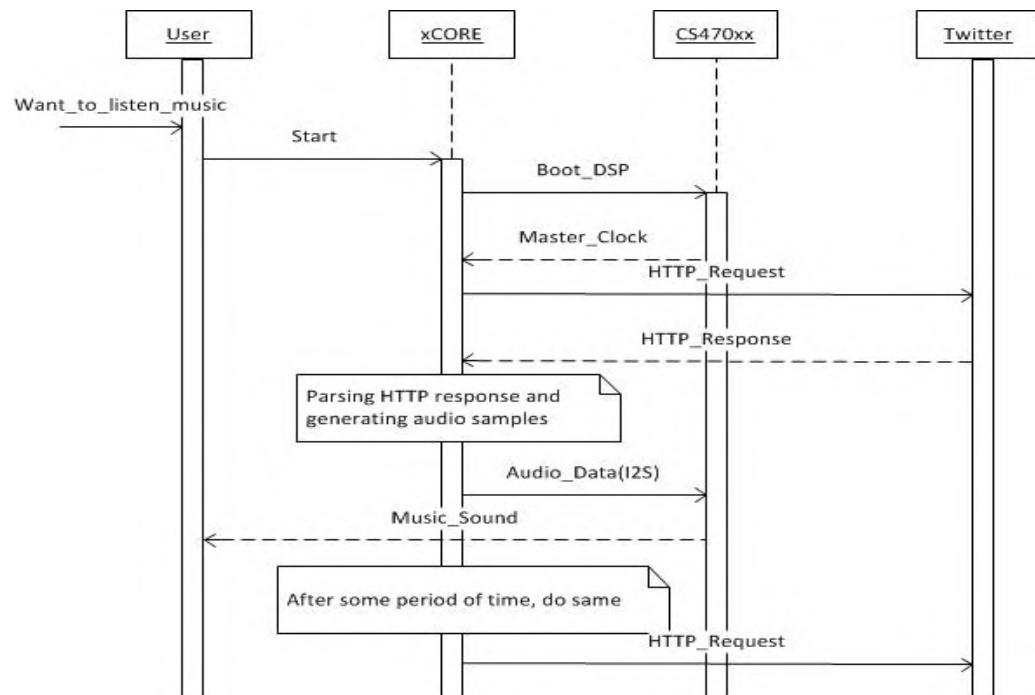
Tabela 3.2 Dužine trajanja nota/pauza

Iz gornjih tabela sledi notacija za pisanje muzike preko poruke: nota_trajanje_pauza_trajanjeNota_trajanje..., primer: d2p2e4p2h2p1m6...

Niti međusobno komuniciraju preko kanala, koji predstavljaju resurse svake pločice na xCORE mikrokontroleru i koji predstavljaju bezbedan način za prenos podataka između procesa.

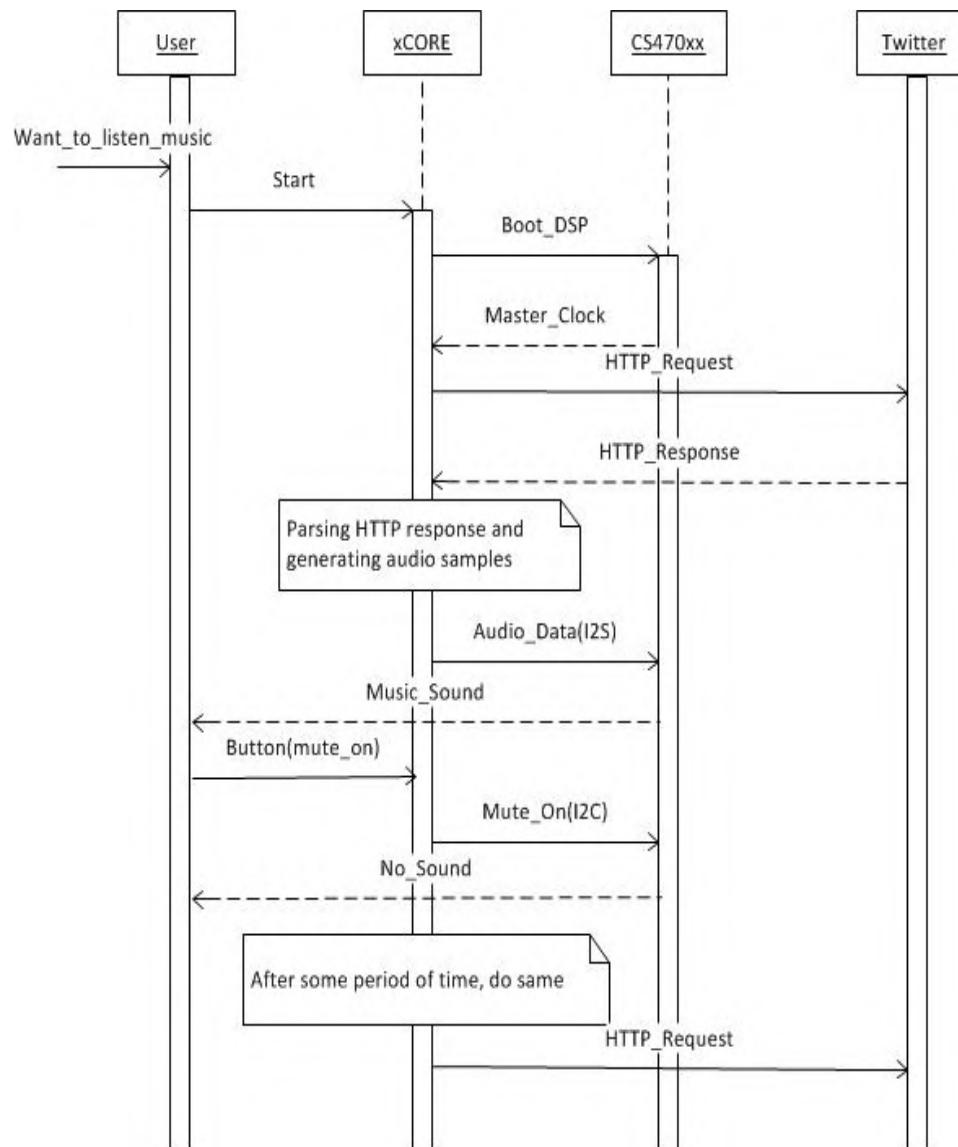
Potrebne veze su ostvarene pinovima sa *GPIO slice* dodatka (slika 3.3), na kome se nalaze i potrebni tasteri, dok je veza između xCORE mikrokontrolera, tj. *Slice Kit* razvojne ploče i računara ostvarena uz pomoć XTAG adaptera (slika 3.4).

Sledi prikaz dijagrama redosleda događaja jedne sesije:



Slika 3.6 Dijagram redosleda jedne sesije

Sledi prikaz dijagrama redosleda jedne sesije sa upotreborom tastera za isključivanje tona:



Slika 3.7 Dijagram redosleda: Sesija isključivanja tona

3.1 Stvaranje muzičkih tonova

Kao što je gore objašnjeno, ton predstavlja zvuk određene učestanosti, amplitude, trajanja i boje. Za ovaj projekat potrebno je napraviti osam osnovnih tonova (slika 2.5), a za učestanosti tonova, izabrana je osnovna oktava[7]:

Tonovi	Učestanosti (Hz)
Do	262
Re	294
Mi	331
Fa	349
Sol	393
La	441
Si	495
Do	556

Tabela 3.3 Učestanosti tonova

Stvaranje tonova (audio odbiraka) gore navedenih učestanosti za učestanosti odabiranja od 48 kHz i 44,1 kHz je urađeno pomoću sinusne funkcije na sledeći način:

- Perioda sinusnog talasa je dužine 2π i može se predstaviti pomoću 4 kvadranta. Talas u prirodi je kontinualan, što nam za digitalne sisteme ne odgovara pa samim tim taj talas odnosno njegovu periodu moramo predstaviti konačnim brojem tačaka. Broj tačaka treba izabrati tako da se zadovolji kvalitet signala a da se pri tome zauzme što manje memorije. Obzirom da ugrađeni sistemi ne raspolažu velikom memorijom kao arhitekture opšte namene, izabrano je 2048 tačaka. Svaki kvadrant je određen uz pomoć 512 tačaka ($\pi = 1024$, $2\pi = 2048$).
- Dovoljno je izračunati samo prvi kvadrant sinusne funkcije. Na osnovu njega možemo dobiti vrednosti za ostala tri kvadranta. Prvi kvadrant ima vrednost od 0 do $\pi/2$, i kad taj opseg podelimo na traženih 512 tačaka, dobijamo korak od 0.0030679. Za izračunavanje 512 tačaka prvog kvadranta sinusne funkcije korišćena je C funkcija `sin()`, biblioteke `math.h`.
- Svaka vrednost `sin()` funkcije je pomnožena sa $2^{15}-1$, kako bi se iz aritmetike pokretnog zareza (engl. *floating point*), koju ne podržava xCORE arhitektura, prešlo u aritmetiku nepokretnog zareza (engl. *fixed point*). Određeno je da audio odbirci budu 16-bitni.

- Pošto je sad izračunat niz *sineWave[]* koji sadrži vrednosti prvog kvadranta, vrednosti ostalih kvadrantata se mogu dobiti na sledeći način: kako je prvi kvadrant *sineWave[x]*, drugi, treći i četvrti će biti *sineWave[1024 - x]*, *-sineWave[x - 1024]*, *-sineWave[2048 - x]*, respektivno, za promenljivu *x* koja će predstavljati *signal_freq * sample_index * 2048 / sample_rate*:

```
int sine(int x) {
    x = x & 2047;
    switch (x >> 9) {
        case 0: return sineWave[x];
        case 1: return sineWave[1024-x];
        case 2: return -sineWave[x-1024];
        case 3: return -sineWave[2048-x];
    }
    return 0;
}
```

- Kako se ne bi trošili resursi ugrađenog sistema na računanje *x* promenljive iz gornjeg primera i da bi se ubrzao ceo proces stvaranja audio signala u realnom vremenu, napravljenе su tabele talasa (engl. *wave tables*) za svaki ton (statički napravljeni nizovi koji su popunjeni prethodno izračunatim vrednostima), koje sadrže onoliko vrednosti koliko određeni ton ima tačaka u svojoj jednoj periodi za datu učestanost odabiranja (ove tabele zahtevaju više memorije ali zato direktno dobijamo vrednosti audio odbiraka, bez zahtevnog računanja).

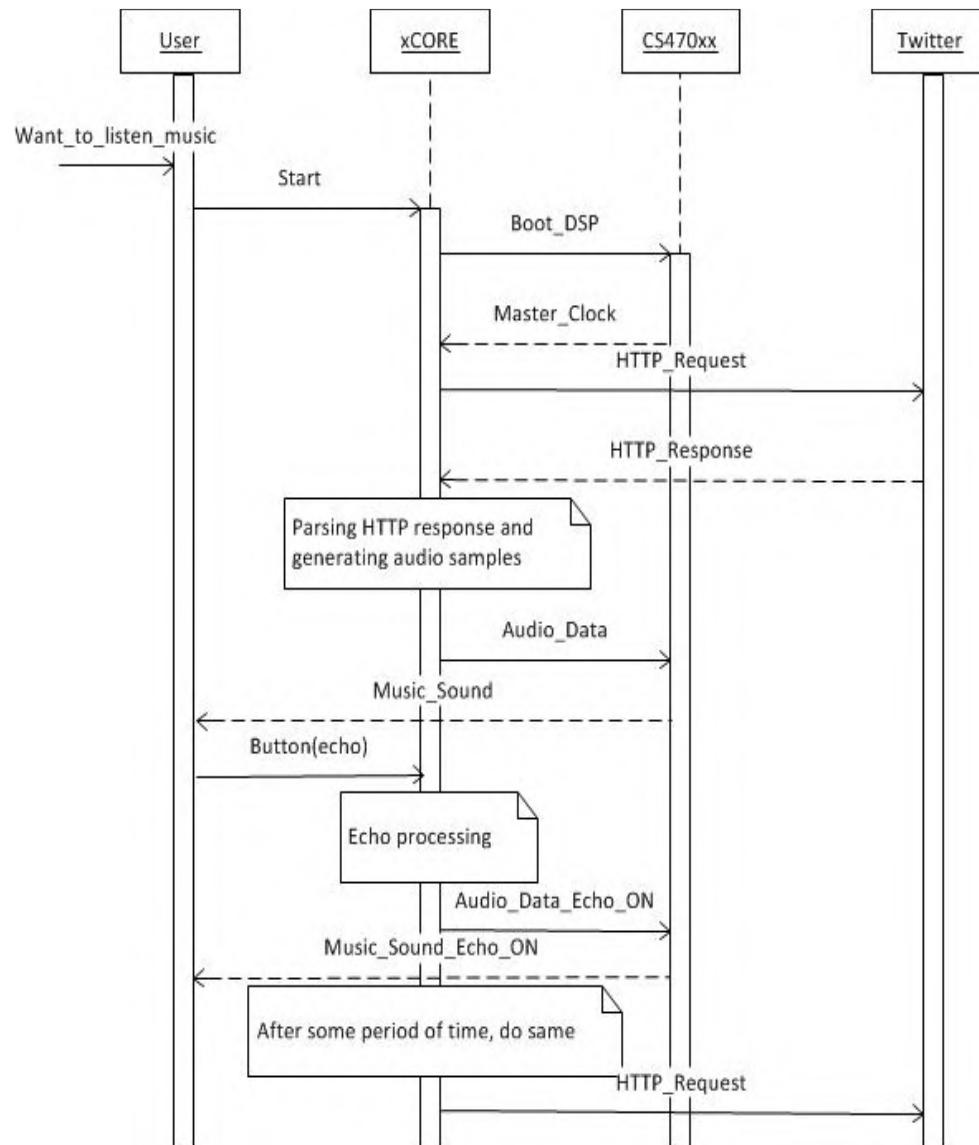
3.2 Echo efekat

Echo predstavlja refleksiju zvuka, koja do slušaoca dolazi sa zakašnjnjem u odnosu na direktni zvuk (zvuk koji nije reflektovan).[8] Samim tim za realizaciju echo efekta potrebna je linija za kašnjenje (engl. *delay line*). Linija za kašnjenje će biti predstavljena kao jedan niz audio odbiraka, čija dužina će biti određena željenim kašnjenjem.

Niz će imati dva pokazivača: jedan će pokazivati na početak niza i služiti kao indeks za čitanje a drugi će pokazivati na kraj i služiti kao indeks pri upisu u niz. Dužina niza, odnosno razlika između pokazivača predstavlja kašnjenje.

Na kraju, da bi se dobio echo efekat, potrebno je još sabrati trenutni audio odbirak (koji će se upisati u niz na mesto koje pokazuje pokazivač za upis) i audio odbirak na koji pokazuje pokazivač za čitanje (zakašnjeni odbirak), umanjen (u ovom slučaju) za 1/2 svoje vrednosti kako ne bi previše narušio vrednost trenutnog audio odbirka i da bi se sprečilo prekoračenje opsega i dobio signal echo efekta bez izobličenja.

Dijagram redosleda jedne sesije pri korišćenju tastera za uključivanje echo efekta prikazan je na slici 3.8.



Slika 3.8 Dijagram redosleda: Uključivanje echo efekta

4. Programsko rešenje

Programsko rešenje ovog rada je podeljeno na sledećih 5 funkcionalnih modula:

- app_twsynthesizer
- module_i2c_master
- module_i2s_master
- module_wifi_tiwisl
- module_spi_master

U narednim odeljcima će biti predstavljen svaki modul, kao i najznačajnije funkcije modula.

4.1 Modul app_twsynthesizer

Ovaj modul predstavlja funkcionalnu srž projekta. Čine ga sledeće osnovne funkcionalne jedinice:

- **audio_io.xc** – koristi se za inicijalizaciju I2S i I2C sprege kao i za komunikaciju preko I2C.

```
void audio_io(streaming chanend c_i2s_data);
```

Funkcija koja inicijalizuje I2S i I2C module.

Parametri:

1. `c_i2s_data` – komunikacioni kanal za prenos I2S podataka

Povratna vrednost: nema

```
void audio_mute_off();
```

Funkcija koja šalje komandu za uključivanje tona preko I2C sprege.

Parametri: nema

Povratna vrednost: nema

```
void audio_mute_on();
```

Funkcija koja šalje komandu za isključivanje tona preko I2C sprege.

Parametri: nema

Povratna vrednost: nema

- **boot_dsp.xc** – koristi se za pokretanje operativnog sistema (engl. *boot*) ciljnog uređaja (ovde je to CS470xx DSP).

```
void boot_dsp();
```

Funkcija koja preko svojih pinova inicira pokretanje operativnog sistema ciljnog uređaja u željenom modu.

Parametri: nema

Povratna vrednost: nema

- **signal_generator.xc** – koristi se za stvaranje audio signala na osnovu pristiglih nota.

```
void generate_signal(chanend c_note, streaming chanend c_sample);
```

Funkcija koja na osnovu primljenih nota proizvodi audio odbirke datog tona i iste prosleđuje ka jedinici za dodatnu obradu (potencijalna dodatna obrada je echo efekat).

Parametri:

1. *c_note* – komunikacioni kanal za prenos nota
2. *c_sample* – komunikacioni kanal za prenos proizvedenih audio odbiraka

Povratna vrednost: nema

- **dsp_effects.xc** – koristi se za dodatnu obradu audio odbiraka (realizovan je echo efekat, ali je cela funkcionalna jedinica struktuirana tako da omogući laku realizaciju novih efekata).

```
void dsp_effects(streaming chanend c_i2s_data, streaming chanend
                 c_sample, chanend c_effect);
```

Funkcija koja vrši dodatnu obradu nad audio odbircima (ako to korisnik zahteva), dodajući echo efekat.

Parametri:

1. `c_i2s_data` – komunikacioni kanal za prenos audio odbiraka do I2S modula
2. `c_sample` – komunikacioni kanal za primanje proizvedenih audio odbiraka
3. `c_effect` – komunikacioni kanal za primanje identifikacionog broja željenog efekta (omogućava laku proširivost funkcionalne jedinice u vidu više realizovanih efekata)

Povratna vrednost: nema

- **button_control.xc** – koristi se za kontrolu pritisnutih tastera na GPIO dodatku (slika 3.3).

```
void button_control(chanend c_effect);
```

Funkcija koja “nadgleda” tastere, i koja nakon pritiska određenog tastera reaguje u skladu sa projektnom logikom (uključuje/isključuje echo efekat ili ton).

Parametri:

1. `c_effect` – komunikacioni kanal za prenos identifikacionog broja željenog efekta (na osnovu pritisnutog tastera)

Povratna vrednost: nema

- **music_transceiver.xc** – koristi se za prijem isparsiranih nota iz HTTP odgovora i za prenos istih do generatora audio signala.

```
void music_transceiver(chanend c_addr, chanend c_note);
```

Funkcija koja iz primljenog niza nota, prosleđuje note do generatora audio signala.

Parametri:

1. `c_addr` – komunikacioni kanal za prijem adrese niza koji sadrži note
2. `c_note` – komunikacioni kanal za prenos nota iz datog niza

Povratna vrednost: nema

- **`xtw_reader.xc`** – koristi se za inicijalizaciju `module_wifi_tiwisl` modula i za primanje događaja (engl. *events*) od istog modula kao i za iniciranje veze prema pristupnoj tački i željenom serveru.

```
void xtw_reader(chanend c_tcp, chanend c_addr);
```

Funkcija koja inicijalizuje `module_wifi_tiwisl` modul, prima događaje od toga modula i svrshishodno reaguje, kao i što inicira vezu ka željenoj pristupnoj tački i serveru.

Parametri:

1. `c_tcp` – kanal za komunikaciju sa `module_wifi_tiwisl` modulom
2. `c_addr` – komunikacioni kanal za prenos adrese niza nota koji je trenutno u upotrebi (postoje dva zbog tehnike duplih bafera)

Povratna vrednost: nema

- **`parser.c`** – koristi se za potrebna parsiranja primljenih HTTP odgovora.

```
int http_header_parser(char data[]);
```

Funkcija koja parsira zaglavla HTTP odgovora.

Parametri:

1. `data[]` – niz koji sadrži zaglavje HTTP odgovora

Povratna vrednost: veličina celog HTTP odgovora (dužina zaglavla + dužina sadržaja)

```
void parse_tweet_time(REFERENCE_PARAM(tw_time_t, tw_time));
```

Funkcija koja parsira HTTP odgovor u potrazi za vremenom stvaranja određene poruke (da bi se znalo da li je poruka već odsvirana).

Parametri:

1. `tw_time` – struktura, čija polja predstavljaju vreme stvaranja date poruke:

```
typedef struct tw_time_t
{
    short year;
```

```

        short month;
        short day;
        int seconds;
    } tw_time_t;
}

```

Povratna vrednost: nema

```
void parse_tweet(char data[], char tw_buffer[]);
```

Funkcija koja parsira HTTP odgovor u potrazi za sadržajem primljenih poruka.

Parametri:

1. `data[]` – niz koji sadrži HTTP odgovor
2. `tw_buffer[]` – niz koji sadrži isparsirane poruke (note)

Povratna vrednost: nema

```
int parse_http_response(char data[], char tw_buffer[]);
```

Glavna funkcija za parsiranje HTTP odgovora, u zavisnosti od potrebe poziva ostale funkcije za parsiranje.

Parametri:

1. `data[]` – niz koji sadrži HTTP odgovor
2. `tw_buffer[]` – niz za skladištenje isparsiranih poruka (nota)

Povratna vrednost: 1 – sviraj isparsirane note

0 – ne sviraj

4.2 Modul module_i2c_master

Ovaj modul realizuje funkcionalnost I2C sprege. Sadrži sledeću funkcionalnu jedinicu:

- **i2c-mm.xc** – realizuje funkcionalnost I2C serijske sprege.

```
void i2c_master_init(REFERENCE_PARAM(struct r_i2c,i2c_master));
```

Funkcija koja inicijalizuje prolaze (engl. *port*) I2C uređaja.

Parametri:

1. `i2c_master` – struktura koja sadrži potrebne I2C resurse koji predstavljaju prolaze za takt i za podatke, kao i broj sistemskih taktova potrebnih za jedan I2C takt:

```

typedef struct r_i2c
{
    port scl;
    port sda;
    unsigned int clockTicks;
} r_i2c;

```

Povratna vrednost: nema

```

int i2c_master_rx(int device, unsigned char data[], int nbytes,
                   REFERENCE_PARAM(struct r_i2c, i2c));

```

Funkcija koja prima podatke adresiranog I2C uređaja.

Parametri:

1. device – adresa uređaja
2. data[] – memorijski niz za prihvati podataka
3. nbytes – broj bajtova za prihvati
4. i2c – struktura koja sadrži I2C resurse

Povratna vrednost: 0 – neuspešno primanje podataka

1 – uspešno primanje podataka

```

int i2c_master_write_reg(int device, int reg_addr,
                         unsigned char data[],
                         int nbytes,
                         REFERENCE_PARAM(struct r_i2c, i2c_master));

```

Funkcija za pisanje (slanje) podataka u registre I2C uređaja.

Parametri:

1. device – adresa uređaja
2. reg_addr – adresa registra u koji se želi pisati
3. data[] – memorijski niz koji sadrži podatke za pisanje u registar
4. nbytes – broj bajtova za pisanje (slanje) u registar
5. i2c_master – struktura koja sadrži I2C resurse

Povratna vrednost: 0 – neuspešno poslati podaci

1 – uspešno poslati podaci

4.3 Modul module_i2s_master

Ovaj modul realizuje funkcionalnost I2S sprege. Sadrži sledeću funkcionalnu jedinicu:

- **i2s_master.xc** – realizuje funkcionalnost I2S serijske sprege.

```
void i2s_master(r_i2s &r_i2s, streaming chanend c_data, unsigned
                mclk_bclk_div);
```

Funkcija realizuje I2S serijsku spregu.

Parametri:

1. **r_i2s** – struktura koja sadrži potrebne resurse I2S sprege (takt blokove i prolaze):

```
typedef struct i2s_resources
{
    clock cb1;
    clock cb2;
    in port mck;
    out buffered port:32 bck;
    out buffered port:32 wck;
    out buffered port:32 dout[NUM_PORTS_DAC];
} r_i2s;
```

2. **c_data** – kanal za prenos audio odbiraka

3. **mclk_bclk_div** – odnos master takta i bit takta

Povratna vrednost: nema

4.4 Modul module_wifi_tiwisl

Ovaj modul realizuje funkcionalnost bežične komunikacije sa Internetom, na način da predstavlja programsку spregu između aplikacije i WiFi dodatka (slika 3.2).

Osnovna funkcionalna jedinica je:

- **wifi_tiwisl_fsm.xc** – programska sprega sa WiFi dodatkom.

```
void wifi_tiwisl_fsm(chanend c_xtcp,
                      REFERENCE_PARAM(spi_master_interface,tiwisl_spi),
                      REFERENCE_PARAM(wifi_tiwisl_ctrl_ports_t,
                                      tiwisl_ctrl));
```

Funkcija koja predstavlja spregu između *app_twsynthesizer* modula i WiFi dodatka. Realizovana je tako da predstavlja automat sa konačnim brojem stanja, u zavisnosti od kojih izvršava određene radnje.

Parametri:

1. *c_xtcp* – kanal za komunikaciju sa *app_twsynthesizer* modulom
2. *tiwisl_spi* – struktura koja sadrži potrebne resurse za SPI spregu (takt blokove i prolaze)
3. *tiwisl_ctrl* – struktura koja sadrži kontrolne prolaze za komunikaciju sa WiFi dodatkom (slika 3.2)

Povratna vrednost: nema

4.5 Modul **module_spi_master**

Ovaj modul realizuje funkcionalnost SPI sprege za komunikaciju sa WiFi dodatkom (slika 3.2).

Modul je sačinjen od sledeće funkcionalne jedinice:

- **spi_master.xc** – realizuje funkcionalnost SPI serijske sprege.

```
void spi_master_init(spi_master_interface &spi_if, int spi_clock_div);
```

Funkcija koja inicijalizuje potrebne prolaze i takt blokove.

Parametri:

1. *spi_if* – struktura koja sadrži potrebne SPI resurse (takt blokove i prolaze):

```
typedef struct spi_master_interface
{
    clock blk1;
    clock blk2;
    out buffered port:8 mosi;
    out buffered port:8 sclk;
    in buffered port:8 miso;
} spi_master_interface;
```

2. *spi_clock_div* – delilac sistemske učestanosti (određuje vrednost sclk takta)

Povratna vrednost: nema

```
void spi_master_in_buffer(spi_master_interface &spi_if, unsigned char
                           buffer[], int num_bytes);
```

Funkcija za primanje određene količine podataka.

Parametri:

1. `spi_if` - struktura koja predstavlja resurse SPI sprege
2. `buffer[]` - memorijski niz za prihvatanje podataka
3. `num_bytes` - broj bajtova za prihvat

Povratna vrednost: nema

```
void    spi_master_out_buffer(spi_master_interface    &spi_if,    const
                                unsigned char buffer[], int num_bytes);
```

Funkcija za slanje određene količine podataka.

Parametri:

1. `spi_if` - struktura koja predstavlja resurse SPI sprege
2. `buffer[]` - memorijski niz koji sadrži podatke za slanje
3. `num_bytes` - broj bajtova za slanje

Povratna vrednost: nema

5. Ispitivanja i rezultati

U cilju ispitivanja i verifikacije realizovanog rešenja izvršene su dve vrste ispitivanja: ispitivanje primanja i parsiranja traženih poruka kao i ispitivanje audio izlaza.

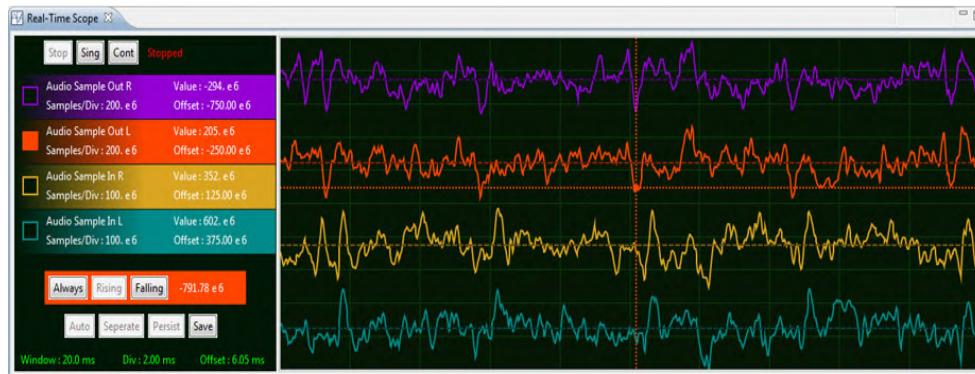
U okviru prvog ispitivanja sistema, slate su poruke, sa relevantnom oznakom i notama (naravno, pored navedenog sadržaja, poruke mogu sadržati i dodatni tekst po želji korisnika), sa više *Twitter* naloga. Po primanju tih poruka, praćeno je ponašanje sistema u toku izdvajanja bitnog sadržaja iz primljenih poruka i stvaranja tonova na osnovu primljenih nota i dužina trajanja nota. Sistem je uspevao da izdvoji relevantan sadržaj iz primljenih poruka i da na osnovu nota i dužina trajanja nota uspešno proizvodi muzičke tonove. Ostali sadržaj poruka kao što su korisničke poruke proizvoljne sadrzine sistem nije uzimao u obzir, što je pri projektovanju i zamišljeno. Takođe pri cikličnom prozivanju *Twitter* servera, već odsvirane note nisu bile ponovo svirane, što je takođe bilo i zamišljeno pri projektovanju.

Ispitivanje audio izlaza je urađeno na sledeći način. Proizvođač muzičkih tonova je prvo realizovan u vidu C programa, pomoću *Microsoft Visual Studio* razvojnog okruženja. Taj program je koristio aritmetiku pokretnog zareza i reč dužine 32 bita za predstavljanje audio odbiraka. Kako je ranije rečeno, xCORE arhitektura, kao predstavnik ugrađenog sistema, ne podržava aritmetiku pokretnog zareza i ne sadrži veliku količinu memorije za skladištenje. Zbog toga je dati program izmenjen u cilju korišćenja celobrojne aritmetike i audio odbiraka dužine 16 bita. Izlazi datog programa su zapisivani u datoteke. Taj program je posle prilagođen xCORE arhitekturi i prebačen na ciljnu platformu. Audio odbirci, sa ciljne platforme, su preko I2S sprege slati na spoljnu zvučnu karticu koja je zapisivala date audio odbirke. Na kraju su audio izlazi C programa i programa ugrađenog u xCORE mikrokontroler upoređeni a rezultat je bio očekivan, tj. nije bilo odstupanja, tabela 5.1:

Tonovi	Razlika
Do	Nema razlike
Re	Nema razlike
Mi	Nema razlike
Fa	Nema razlike
Sol	Nema razlike
La	Nema razlike
Si	Nema razlike
Do	Nema razlike

Tabela 5.1 Razlika između audio odbiraka sa PC računara i xCORE mikrokontrolera

Za dodatna ispitivanja signala taktova i vrednosti audio odbiraka koristili su se osciloskop i xSCOPE. xSCOPE je programski osciloskop kompanije XMOS, projektovan radi mogućnosti prikupljanja i praćenja podataka u realnom vremenu (slika 5.1).



Slika 5.1 Izgled xSCOPE alata

6. Zaključak

U ovom radu je prikazano jedno rešenje traženog audio sistema zasnovanog na xCORE višejezgarnom mikrokontroleru kompanije XMOS. Cilj zadatka je bio da se ispitaju mogućnosti fizičke arhitekture i da se pokaže ubrzanje višejezgarnog xCORE mikrokontrolera pri paralelnom izvršavanju, u ovom slučaju proizvodnje audio odbiraka i dobavljanja potrebnih podataka (muzičkih nota). Pomenuta paralelizacija je uspešno realizovana na datoј platformi, a dobijeno ubrzanje je išlo u korist izvršavanju operacija u realnom vremenu.

Na osnovu zadatih zahteva realizovane su sledeće funkcionalnosti. xCORE mikrokontroler (u sastavu *Slice Kit* razvojne ploče), kao glavni uređaj u sistemu, je povezan sa *Cirrus Logic* CDB470xx razvojnom pločom, koja predstavlja audio izlaz, preko I2S i I2C serijskih sprega. Mikrokontroler na osnovu primljenih HTTP odgovora, posle poslatih HTTP zahteva, proizvodi muzičke tonove i omogućuje dodatnu obradu, za šta je realizovan echo efekat koji se uključuje/isključuje pritiskom na taster GPIO dodatka (slika 3.3). Realizovana je i kontrola uključivanja/isključivanja tona preko tastera.

Ispitivanja i verifikacija su obuhvatila ispitivanje sistema kako u delu za pribavljanje i parsiranje muzičkih nota tako i u delu proizvodnje audio odbiraka. Audio izlaz je uspešno ispitana upoređivanjem vrednosti audio odbiraka ciljne platforme sa referentnim vrednostima (proizvedenim na PC računaru).

Ovaj rad se može proširiti uvođenjem povisilica i snizilica u notni sistem, kao i uvođenjem novih audio efekata.

7. Literatura

- [1] Tim Wilmhurst: *Designing embedded systems with PIC microcontrollers*, 2010
- [2] Jivan S. Parab, Vinod G. Shelake, Rajanish K. Kamat, Gourish M. Naik: *Exploring C for microcontrollers*, 2007
- [3] XMOS Multicore Microcontrollers, <http://www.xmos.com/>, jul 2013
- [4] V. Kovačević, M. Temerinac, M. Popović, N. Teslić: *Arhitekture i algoritmi DSP-a 1*, 2004
- [5] Twitter Developers, <https://dev.twitter.com/docs>, jul 2013
- [6] Marko Tajčević: *Osnovna teorija muzike*, 1997
- [7] Frequencies of musical notes – Physics, <http://www.phy.mtu.edu/~suits/notefreqs.html>, avgust 2013
- [8] The Physics Classroom, <http://www.physicsclassroom.com/mmedia/waves/er.cfm>, avgust 2013