VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2024 Timotej Fojtík

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Audio inženýrství**
specializace Zvuková produkce a nahrávání
Ústav telekomunikací

Student: Timotej Fojtík ***ID:*** 240151
Ročník: 3 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

Aktualizace úlohy demonstrující syntézu periodických signálů do předmětu Analýza signálů a soustav

POKYNY PRO VYPRACOVÁNÍ:

Cílem této práce je aktualizovat úlohu předmětu Analýza signálů a soustav (BPC-ASI), kde je demonstrován proces syntézy periodických signálů pomocí částečných součtů harmonických signálů. Bude navržen a realizován nový přípravek provádějící syntézu. Dále bude aktualizován návod a vypracování dané laboratorní úlohy, přičemž tato úloha bude mít 8 variant tak, aby studenti v 8 skupinách vždy naměřili trochu jiné výsledky.

DOPORUČENÁ LITERATURA:

Podle pokynů vedoucího práce

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 28.5.2024

Vedoucí práce: Ing. Ondřej Krajsa, Ph.D.

doc. Ing. Jiří Schimmel, Ph.D.
předseda rady studijního programu

Abstrakt

Táto záverečná práca sa zameriava na návrh a implementáciu prípravku pre demonštráciu syntézy signálov v rámci výučby Analýzy signálov a sústav. Cieľom prípravku je umožniť študentom experimentovať so syntézou periodických signálov, konkrétne obdĺžnikového, pílového a trojuholníkového, prostredníctvom postupného pridávania vyšších harmonických zložiek.

Prípravok je navrhnutý s dôrazom na jednoduchú obsluhu a širokú dostupnosť súčiastok, aby bol vhodný pre vyučovanie v nasledujúcich rokoch. Implementácia zahŕňa výber mikrokontroléru ATmega328, digitálno-analógový prevodník MCP4921 a rekonštrukčný Butterworthov filter štvrtého radu.

Kľúčové slová

Syntéza signálov, Výučba Analýzy signálov a sústav, Periodické signály, Harmonické zložky, Mikrokontrolér ATmega328, Digitálno-analógový prevodník MCP4921, Rekonštrukčný filter, Napájanie a ochrana pred prepoľovaním

Abstract

This final thesis focuses on the design and implementation of a device for demonstrating signal synthesis within the teaching of Signal Analysis and Systems. The device aims to enable students to experiment with the synthesis of periodic signals, specifically square, sawtooth, and triangular waves, by gradually adding higher harmonic components.

The device is designed with an emphasis on simple operation and the widespread availability of components to make it suitable for teaching in the following years. The implementation includes the selection of the ATmega328 microcontroller, the MCP4921 digital-to-analog converter, and a fourth-order Butterworth reconstruction filter.

Keywords

Signal Synthesis, Teaching Signal Analysis and Systems, Periodic Signals, Harmonic Components, Microcontroller ATmega328, Digital-to-Analog Converter MCP4921, Reconstruction Filter, Power Supply and Overvoltage Protection

Bibliografická citace

FOJTÍK, *Timotej. Aktualizace úlohy demonstrující syntézu periodických signálů do předmětu Analýza signálů a soustav*. Brno, 2024. Dostupné také z: *https://www.vut.cz/studenti/zav-prace/detail/159274*. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Ondřej Krajsa.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	*Timotej Fojtík*
VUT ID studenta:	*240151*
Typ práce:	*Bakalářská práce*
Akademický rok:	*2023/24*
Téma závěrečné práce:	*Aktualizace úlohy demonstrující syntézu periodických signálů do předmětu Analýza signálů a soustav*

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucího závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

.

V Brně dne: 22.05.2024 ----------

podpis autora

Obsah

1. ÚVOD 5

2. TEORETICKÝ ÚVOD 6

2.1 HARMONICKÉ SIGNÁLY 6
2.2 HARMONICKÁ SYNTÉZA 6
2.3 D/A PREVODNÍK 8
2.3.1 R-2R 8
2.3.2 Binárna sieť rezistorov 8
2.3.3 Čipové D/A prevodníky 9
2.4 REKONŠTRUKČNÉ FILTRE 9
2.4.1 Butterworthov filter 10
2.4.2 Chebyshevov filter 10
2.4.3 Besselov filter 10
2.5 KOMUNIKAČNÉ PROTOKOLY 11
2.5.1 I^2C 11
2.5.2 SPI 12

3. NÁVRH PRÍPRAVKU 13

3.1 VYBER SÚČIASTOK 13
3.1.1 Výber mikrokontroleru 13
3.1.2 Výber D/A prevodníku 14
3.1.3 Výber operačného zosilňovača rekonštrukčného filtru a jeho návrh 15
3.1.4 Výber napájania 17
3.2 OVLÁDANIE A UŽÍVATEĽSKÉ ROZHRANIE 18
3.3 PREPROGRAMOVANIE A DIAGNOSTIKA 18
3.4 SCHÉMA 19
3.4.1 Tabuľka komponentov 19
3.4.2 Schéma prípravku 20
3.4.3 Doska plošných spojov 21
3.5 OCHRANA 21
3.6 SOFTVÉR 22
3.6.1 Vývojový diagram 22
3.6.2 Programový kód 23
3.7 TESTOVANIE 25
3.7.1 Generovanie 25
3.7.2 Užívateľské rozhranie 26
3.8 VÝSTUP PRÍPRAVKU 27

4. ZÁVER 28

PRÍLOHA A – ZDROJOVÝ KÓD PRÍPRAVKU 30

1. ÚVOD

Cieľom tohto projektu je obnovenie úlohy pre predmet Analýza signálov a sústav. Táto úloha má demonštrovať syntézu signálov. Pod syntézou rozumieme skladanie periodických signálov z mnohých harmonických vĺn. Amplitúda a fáza týchto vyšších harmonických vĺn vplýva na výsledný tvar signálu a jeho vlastnosti.

Navrhnutý prípravok umožní študentom v praxi pozorovať ako sa postupným zvyšovaním počtu vyšších harmonických vĺn tvoria známe signály ako napríklad obdĺžnikový, pílový alebo trojuholníkový signál.

Zadaný prípravok sa má skladať prioritne z SMD (z anglického Surface Mount Device) súčiastok. Jeho celkový rozmer nemá presiahnuť 10 x 10 cm. Taktiež má byť aktualizované užívateľské rozhranie, pre príjemnejšie ovládanie študentami, ale aj profesormi.

Ďalšia priorita prípravku je široká dostupnosť všetkých použitých súčiastok, a to s ohľadom na nadčasovosť prípravku, keďže bude používaný na vyučovacích hodinách v nasledujúcich rokoch.

Prípravok musí mať dostatočnú ochranu, aby neohrozoval neskúsených študentov a zároveň nebolo nutné o jeho časté opravy. Zadanie taktiež udáva že prípravok bude napájaný zo symetrického DC zdroja ±12 V.

Záverečná práca je členená do troch základných kapitol. Prvá kapitola predstavuje teoretický úvod do problematiky syntézy signálov. V druhej kapitole je pozornosť venovaná návrhu tohoto prípravku. Tretia kapitola obsahuje stručné zhrnutie celého dokumentu.

2. TEORETICKÝ ÚVOD

V tejto kapitole sa bližšie zameriame na syntézy signálov, digital-analog prevodníky a rekonštrukčné filtre.

2.1 Harmonické signály

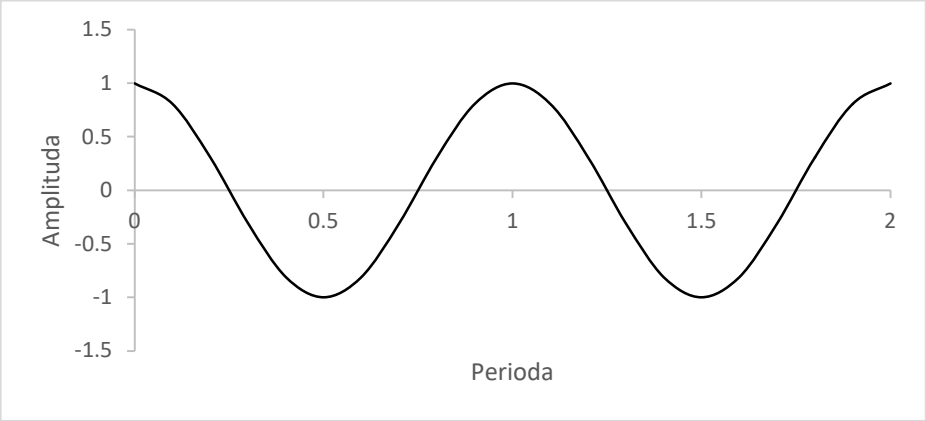
Obsah tejto podkapitoly je inšpirovaný skriptami z predmetu Analýza signálů a soustav od autora menom Prof. Ing. Zdeněk Smékal CSc. [1]

Základný harmonický signál je definovaný rovnicou kosínusu:

$$u(t) = U_m \cdot \cos(\frac{2 \cdot \pi}{T} t + \varphi) \qquad (1.1)$$

Amplitúda U_m je najväčšia výchylka, počiatočná fáza φ je rozdiel medzi počiatkom funkcie a počiatkom súradníc, a T je perióda. Túto periódu môžeme jednoducho previesť na frekvenciu pomocou vzorca $f = \frac{1}{T}$.

Na obrázku 2.1 môžeme vidieť časový priebeh harmonického signálu.



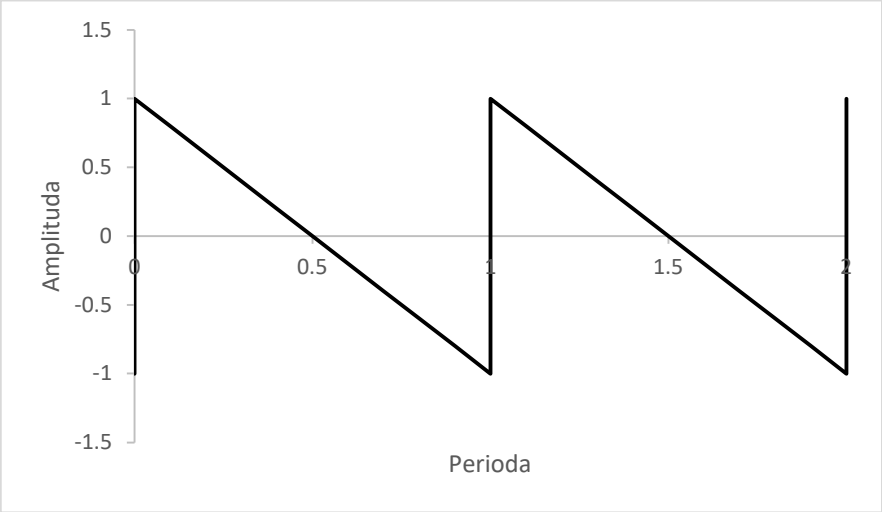
Obrázok 2.1 časový priebeh harmonického signálu.

2.2 Harmonická Syntéza

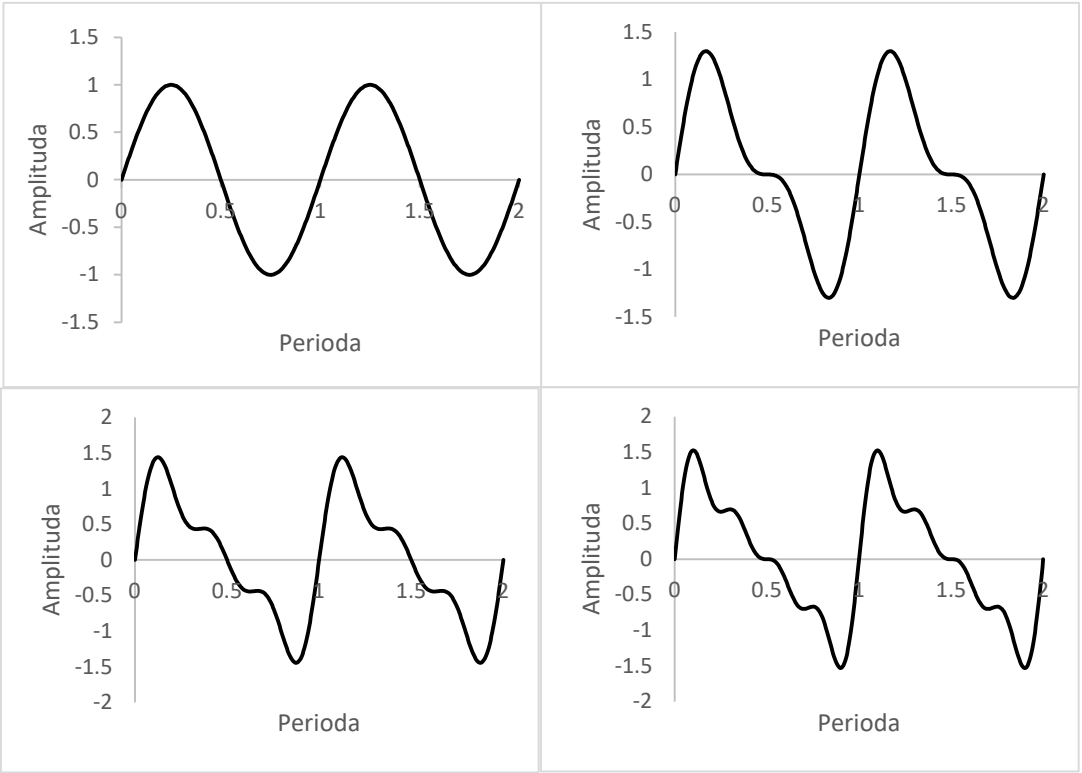
Harmonická syntéza je operácia, pri ktorej sa periodický signál skladá z viacerých harmonických signálov, ktorých frekvencia je daná celočíselnými násobkami jeho fundamentálnej frekvencie. To znamená, že frekvencia vyšších harmonických zložiek je 1 f, 2 f, 3 f, ...

Amplitúda vyšších harmonických zložiek je daná spektrom amplitúd. Ich počiatočná fáza je daná spektrom fáz.

Na obrázku 2.2 je príklad periodického pílového signálu. Na obrázku 2.3 je znázornené, ako sa postupne skladajú jednotlivé harmonické zložky ktoré v nekonečne vytvoria priebeh periodického pílovitého signálu z obrázku 2.2.



Obrázok 2.2 príklad periodického pílového signálu.



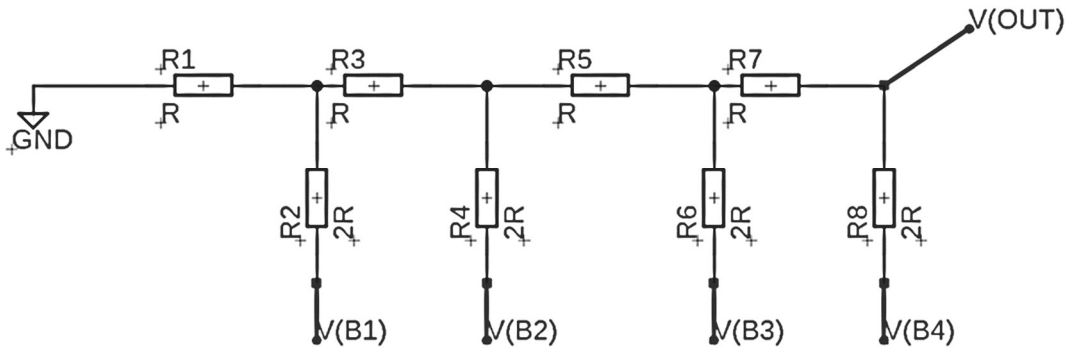
Obrázok 2.3 skladanie harmonických zložiek pílového signálu.

2.3 D/A prevodník

Pri generovaní analógového signálu pomocou digitálneho mokrokontroleru vznikajú viaceré komplikácie. Jedna z nich je nedokonalosť výstupného signálu. Keďže mikrokontroler nevie generovať analógové hodnoty, musíme použiť Digital/Analog prevodník. Tento nám pomôže prideliť digitálnym číslam ich analógovú hodnotu. Táto podkapitola je inšpirovaná literatúrou An overview of principles and types of ADC and DAC [2].

2.3.1 R-2R

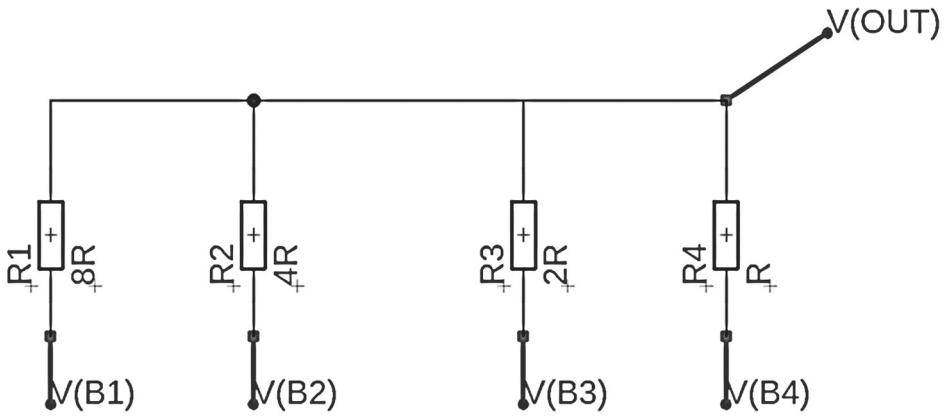
R-2R je najjednoduchší spôsob prevodu digitálneho signálu na analógový. Je založený na kaskádovaní napäťových deličov medzi výstupmi mikrokontroleru a zemou (Obrázok 2.3.1). Hodnoty rezistoru na tomto deliči sú R a 2R, z čoho vyplýva práve názov R-2R.



Obrázok 2.3.1 Zjednodušené schéma R-2R digital-analog prevodníku. Toto schéma obsahuje vstupy na 4 bity.

2.3.2 Binárna sieť rezistorov

Druhý jednoduchý spôsob prevodu digitálneho signálu na analógový je použiť binárnu sieť rezistorov. To znamená pripojiť každý pin výstupu mikrokontroleru na rezistor s hodnotou $2^n \cdot R$. Na obrázku 2.3.2 môžeme vidieť toto jednoduché zapojenie.[2]



Obrázok 2.3.2 Zjednodušené schéma zapojenia binárnej siete rezistorov. Toto schéma obsahuje vstupy na 4 bity.

2.3.3 Čipové D/A prevodníky

Zatiaľ čo niektoré čipové D/A prevodníky obsahujú zmenšené verzie R-2R alebo binárne siete rezistorov, mnohé operujú na iných princípoch.

Sigma-Delta prevodníky pracujú na princípe Sigma-Delta modulácie. Prvým stupňom je delta modulácia. Ide o odčítanie predchádzajúceho výstupu od aktuálneho vstupu, čím sa určí ich rozdiel (delta). Výsledkom je 1-bitový dátový tok, ktorý predstavuje zmenu signálu. 1-bitový signál sa následne prevzorkuje signálom s oveľa vyššou frekvenciou. Prevzorkovaný signál sa integruje v čase a výsledok sa porovnáva s referenčným napätím. Výstup komparátora sa privádza späť do integrátora, čím sa vytvorí slučka spätnej väzby. Vysokofrekvenčný kvantizačný šum sa posúva na vyššie frekvencie a práve táto vlastnosť tvarovania šumu robí delta-sigma D/A prevodníky atraktívnymi pre vysoko presné aplikácie.

2.4 Rekonštrukčné filtre

Vďaka použitiu D/A prevodníku dostaneme na výstupe analógový signál. Tento signál však nie je plynulý, a preto obsahuje aj nežiaduce vysoké frekvencie. Na vyhladenie tohto signálu sa používajú rekonštrukčné filtre.

Pri výbere týchto filtrov máme na výber z dvoch najzakladanejších typov. Prvým typom sú aktívne filtre a druhým typom sú pasívne filtre.

Aktívny filter využíva operačný zosilňovač ktorý funguje ako sledovač napätia, zatiaľ čo pasívny využíva iba pasívne súčiastky. Z tohto dôvodu sa na rekonštrukčné filtre používajú prevažne aktívne filtre.

Základným parametrom každého filtru je jeho medzná frekvencia. Táto sa vypočíta pomocou daného vzorca:

$$f_c = \frac{1}{2\,\pi\,R\,C} \tag{2.4}$$

f_c udáva medzný kmitočet, R je odpor rezistoru a C je kapacita kondenzátoru.

Pri filtri prvého radu udáva táto frekvencia pracovný bod, pri ktorom je pokles výstupu voči vstupu presne o -3 dB.

Táto podkapitola je založená na knihe ACTIVE FILTERS Theory and Design [4]

2.4.1 Butterworthov filter

Najznámejší a najpoužívanejší aktívny filter je Butterworthov filter. Jeho najväčšou prednosťou je jeho obzvlášť plochá charakteristika. To znamená, že je navrhnutý tak, aby jeho frekvenčné spektrum prechodovej oblasti zostalo čo najrovnejšie. Jeho rýchlosť útlmu nie je taká strmá ako iné filtre, avšak jeho jednoduchosť ľahko kompenzuje tento problém, keďže sa dá ľahko kaskádovať.

Napriek tomu, že Butterworthov filter je možné zostrojiť ako filter prvého radu, málokedy sa toto zapojenie používa. S pomocou jedného operačného zosilňovača je možné ho zapojiť ako filter druhého radu, čo nám umožní zlepšiť rýchlosť útlmu z -20 dB/dek na -40 dB/dek.

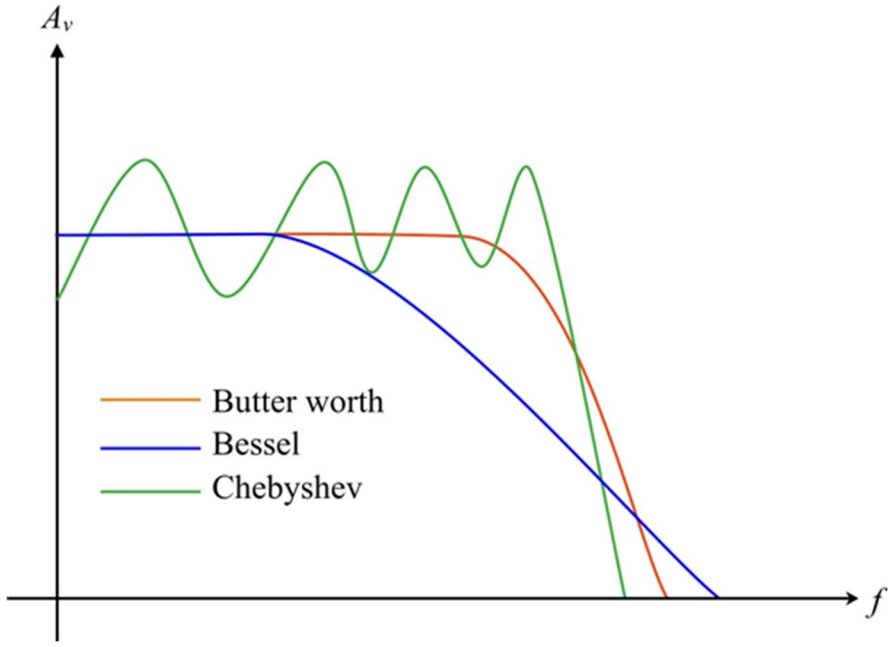
2.4.2 Chebyshevov filter

Na rozdiel od Butterworthovho filtru, má Chebyshevov filter omnoho lepšiu rýchlosť útlmu. Avšak veľká nevýhoda tohto filtru je, že jeho prechodová oblasť je značne zvlnená. Toto sa len zhoršuje pri jeho kaskádovaní. Vďaka tejto skutočnosti sa tento filter využíva len zriedka pre použitie vo sfére audio-techniky.

2.4.3 Besselov filter

Besselov filter má najmenej strmú rýchlosť útlmu avšak jeho prednosťou je obzvlášť lineárna fáza.

Na obrázku 2.4 je znázornený prenos spomenutých filtrov. Môžeme si všimnúť že Butterworthov filter je medzi Besselovým a Chebyschevovým. Jeho strmosť je väčšia ako Besselov ale nemá zvlnenú prenosovú charakteristiku tak ako Chebyschevov.



Obrázok 2.4 Porovnanie základných typov filtrov. Obrázok je prevzatý zo zdroja [5]

2.5 Komunikačné protokoly

2.5.1 I²C

Táto podkapitola je inšpirovaná videom Understanding I2C zo zdroja [7]

Tento protokol využíva iba dva vstupy a to SCL a SDA. SCL (z anglického Serial Clock Line) je využívané na časovanie, zatiaľ čo SDA (z anglického Serial Data Adress) slúži na prenos dát na správnu adresu. Táto adresa je vo väčšine prípadov 7-bitová. Toto umožňuje ovládať až 127 zariadení.

Protokol I²C (taktiež nazývaný I2C) pracuje na hierarchii master-slave. Ako prvé master začne komunikáciu, a následne zašle adresu adresovaného zariadenia. Po začatí komunikácie, master začne zasielanie dáta v 8-bitových úsekoch. Medzi týmito úsekmi zašle slave takzvaný Acknowledgment bit. Tento bit slúži ako indikácia toho že slave prečítal dáta správne a je pripravený na ďalší úsek. V tomto bite sa odohráva aj takzvaný Clock Stretching. Tento proces umožňuje zariadeniu slave mierne natiahnuť časovanie v prípade že nestihol dáta spracovať.

I²C umožňuje komunikáciu o štandardnej rýchlosti 100 kb/s alebo 400 kb/s. Avšak pri použití ATmega328 nebude možné komunikáciu nakonfigurovať na najrýchlejšie nastavenie 3.4 Mb/s.

2.5.2 SPI

Táto podkapitola je inšpirovaná videom Understanding SPI zo zdroja [8]

SPI (z anglického Serial peripheral interface) využíva štvorkanálovú komunikáciu. Využíva na to kanály:

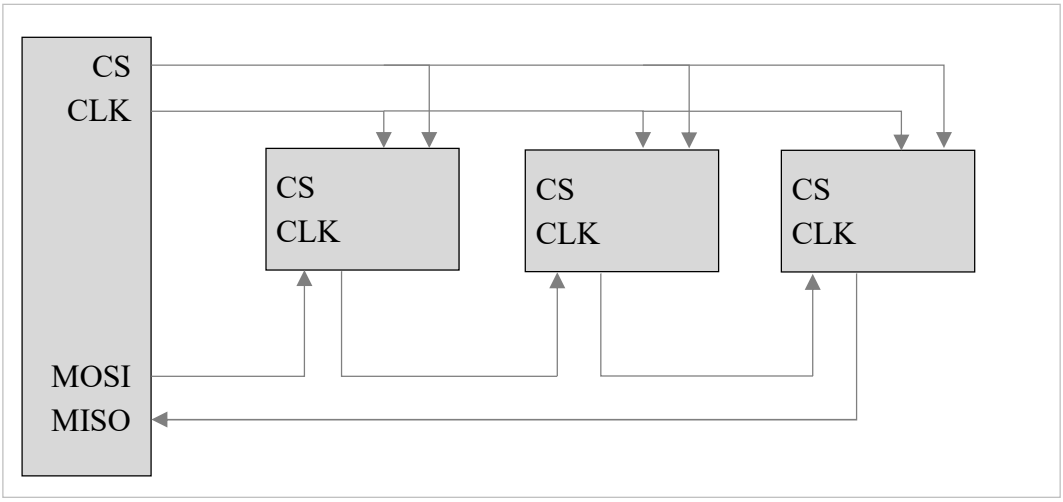
- CLK (clock) – obdĺžnikový signál, slúžiaci na časovanie dát
- CS (chip select) – umožňuje zvolenie správneho zariadenia slave
- MISO (master in, slave out) – komunikácia medzi zariadením slave a master
- MOSI (master out, slave in) – komunikácia medzi zariadením master a slave

Výhodou voči ostatným protokolom je mnohonásobne zvýšená rýchlosť. V prípade použitia ATmega328 s interným časovaním na 20 MHz, dosiahne rýchlosti prenosu približne 5 Mb/s.

Nevýhoda je oveľa zložitejšia implementácia vďaka viacerým štandardizovaným módom. Tieto módy sa venujú čítaniu, respektíve odosielaniu dát. Buď to robia na nábežnej strane (rising edge) alebo upadajúcej strane (falling edge). Zároveň môže časovací signál v kľude nadobúdať hodnotu HIGH alebo LOW.

Ďalšie nevýhody sú spojené s ovládaním viacerých zariadení slave. Pri protokole SPI sa používa samostatný kanál CS na zvolenie zariadenia. Tento kanál má v kľude hodnotu HIGH. Pri komunikácii nadobudne tento kanál hodnotu LOW a započne sa tým komunikácia.

Ďalšia možnosť je použitia Daisy Chain. Toto spočíva v zapojení jedného kanálu CS na všetky zariadenia, ale následne postupne prepojenie MOSI a MISO kanálov ostatných zariadení slave. (obrázok 2.6). Toto umožňuje väčšie množstvo zariadení ako pri tradičnom zapojení. Jedna sa však o niekoľkonásobne zložitejšie zapojenie, ktoré stráca na rýchlosti.

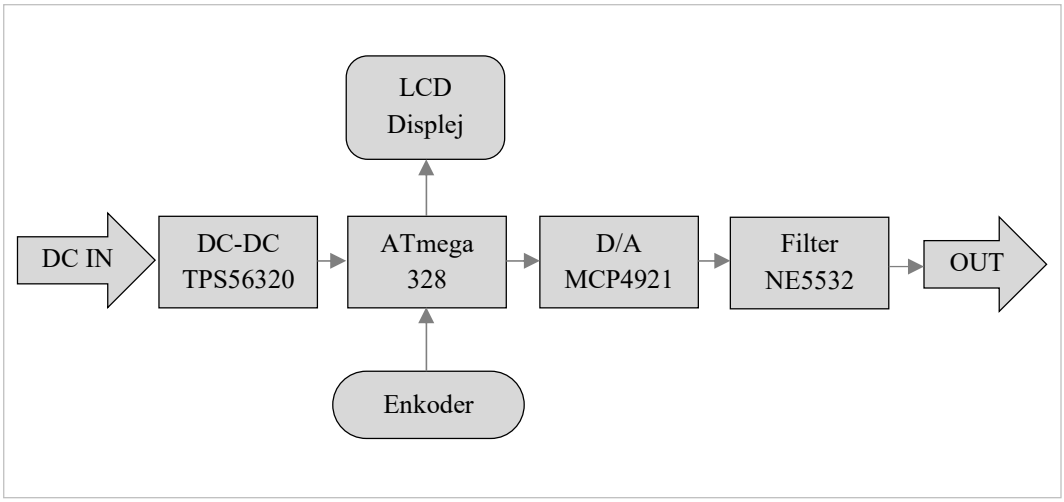


Obrázok 2.6 Blokové schéma znázorňujúce zapojenie Daisy Chain pre SPI

3. NÁVRH PRÍPRAVKU

Táto kapitola bližšie približuje jednotlivé bloky z ktorých sa skladá prípravok. Menovite sa venuje návrhu rekonštrukčného koncového filtru, výberu D/A prevodníku, ovládaniu prípravku a zobrazovaniu nastavení.

Na obrázku 3.0 je zobrazené blokové schéma prípravku od ktorého sa bude ďalej vyvíjať jeho realizácia.



Obrázok 3.0 Blokové schéma prípravku.

Jedným z cieľov je obnovenie užívateľského rozhrania. Ako je vidieť z obrázku, prípravok bude ovládaný jedným enkodérom s tlačidlom a jeho údaje budú zobrazované na displeji. Toto minimalistické ovládanie zjednoduší študentom meranie.

Dôležité je teda aby bol prípravok nielen funkčný, ale aj intuitívny.

3.1 Vyber súčiastok

Všetky súčiastky boli zvolené s ohľadom na udržanie čo najmenších rozmerov prípravku. Z tohto dôvodu boli všetky možné súčiastky zvolene v SMD prevedení. Toto umožnilo vo výsledku dosiahnuť minimálnych rozmerov.

3.1.1 Výber mikrokontroleru

Pri výbere mikrokontroleru bolo hlavnou prioritou, aby bol čo najjednoduchšie programovateľný. Z tohto dôvodu bol zvolený čip ATmega328. Jeho jednoduché programovanie pomocou Arduino IDE a jeho ľahká dostupnosť boli kľúčovými prvkami pri rozhodovaní. ATmega328 zvláda rýchlosť 20 MHz, čo je dostačujúce na

naše účely. Jeho pamäť je 32 kB a disponuje aj 2 kB SRAM, čo vzhľadom na odhadovanú veľkosť programu nebude spôsobovať komplikácie.

Jeho ďalšou výhodou je komunikácia pomocou I^2C, aj SPI protokolu. Toto nám umožní väčší výber pri hľadaní komponentov.

Tento mikrokontroler taktiež disponuje 20 vstupmi. Z nich 6 je pripojených na A/D prevodník s rozlíšením 10 bitov. Zvyšných štrnásť je digitálnych a slúžia aj ako výstupy. Šesť z týchto digitálnych výstupov je možné použiť aj ako PWM (z anglického Pulse Width Modulation). Táto pulzná modulácia umožňuje rýchlo oscilovať medzi 5 V a 0 V, a strieda tohto signálu umožňuje simulovať analógové napätia. Tento spôsob nebude však možné použiť na analógovú syntézu kvôli jeho nízkej frekvencii, a to 490 Hz alebo 980 Hz.

ATmega328 sa vyrába vo viacerých puzdrách, a to TH (through hole), ktoré je vhodné pre jednoduchú výmenu predprogramovaných mikrokontrolerov, a SMD, ktoré je viac permanentne a jeho preprogramovanie je komplikovanejšie. V pôvodnom návrhu bolo zvolené puzdro TH, avšak po konzultáciách bolo zvolené puzdro SMD pre jeho kompaktnejšie prevedenie. V prípade prípravku nebude nutné časté preprogramovanie. Program bude nahraný do mikrokontroleru a už ho nebude nutné ďalej meniť.

Ďalšou možnosťou bolo ATmega2560, ktoré ma niekoľkonásobne viac vstupov, osemkrát väčšiu pamäť úložiska a štvornásobnú pamäť RAM. V našom prípade však nie je nutné používať viacero vstupov a nebude potrebná ani väčšia pamäť. Vzhľadom k tomu, že používajú rovnakú rýchlosť procesora, tento mikrokontroler by nepomohol ani zvýšiť rýchlosť. Preto je použitie tohto mikrokontroleru pre náš prípad zbytočné.

3.1.2 Výber D/A prevodníku

Mikrokontroler ATmega328 disponuje štrnástimi digitálnymi pinmi. Táto skutočnosť nás jemne obmedzuje pri návrhu D/A prevodníku. Keďže budeme pracovať s frekvenciami v počuteľnom spektre, budeme potrebovať minimálne 8-bitové rozhranie, avšak 10 bitov by bolo optimálne. Keby sme zvolili R-2R alebo binárnu sieť rezistorov, tak by sme vyčerpali väčšinu výstupov nášho mikrokontroleru. Pôvodne bol zvolený čipový D/A prevodník s rozhraním I^2C.

MCP4725 prevodník bol zvolený s ohľadom na jeho širokú dostupnosť a množstvo voľne dostupných knižníc pre Arduino. Tento prevodník operuje na binárnej sieti rezistorov a disponuje 12-bitovým rozlíšením, ktoré je viac než dostačujúce pre naše účely. Je napájaný rovnakým napätím ako náš mikrokontroler (5 V DC). Ďalšími výhodami sú jeho malá veľkosť, nízka spotreba prúdu a pamäť EEPROM. Táto pamäť mu umožní uložiť poslednú hodnotu aj po jeho odpojení.

Počas testovania sa však ukázalo, že nebude možné použiť maximálny potenciál tohto prevodníku, a to 3,4 Mb/s. Prípravok teda pracoval na frekvencii len 400 kB/s, čo v praxi umožnilo generovať frekvencie iba v desiatkach Hz. ATmega328 nanešťastie

nepodporuje tuto vyššiu rýchlosť, takže by bolo nutné zmeniť mikrokontroler, ktorý by bol z tohto dôvodu niekoľkonásobne finančne náročnejší. Bol preto zvolený druhý prevodník, a to prevodník s komunikáciou SPI.

MCP4921 je D/A prevodník s 12-bitovým rozlíšením, ktorý disponuje externou napäťovou referenciou, rýchlym prechodovým časom 4,5 µs a vďaka SPI podporuje rýchlosť 5 Mb/s. Toto je viac než 10-krát viac ako pri komunikácii I^2C.

Ďalšie zrýchlenie je možné aj vďaka tomu, že SPI posiela iba 4 bity s konfiguráciou a následne 12 bitov so samotnými dátami pre D/A prevodník. I^2C najskôr zašle 8 bitov s adresovaním na zariadenie slave a až následne 4 bity na konfiguráciu a 12 bitov s dátami. Nielen že prostredníctvom SPI bude vyššia rýchlosť prenosu, ale bude o polovicu viac optimálna. Počas testovania nam toto umožnilo zvýšiť frekvenciu generovaného signálu o vyše 20-násobok!

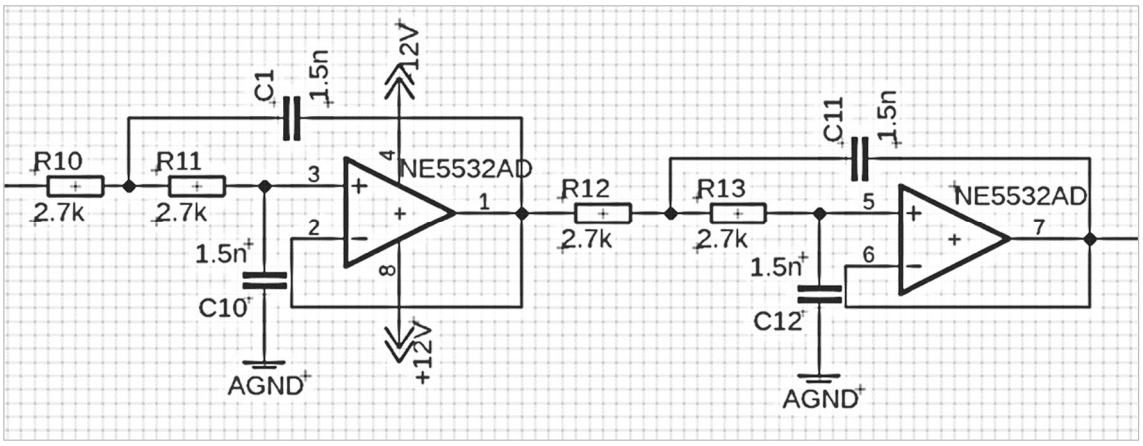
Prechod na protokol SPI však priniesol komplikáciu s komunikáciou s displejom, ktoré sa však nakoniec podarilo prekonať.

3.1.3 Výber operačného zosilňovača rekonštrukčného filtru a jeho návrh

Ako typ filtru bol zvolený Butterworthov aktívny filter. Jeho plochá charakteristika v priepustnom pásme a jeho jednoduchá realizácia boli kľúčovými prvkami pri jeho výbere.

Ako operačný zosilňovač bol zvolený NE5532. Jeho hlavnými prednosťami sú jeho široká dostupnosť, jeho vysoký odstup signálu od šumu a jeho možnosť využitia dvoch kanálov. Maximalne napajacie napatie je ±22 V. Tieto vlastnosti umožnili navrhnúť aktívny Butterworthov filter štvrtého radu.

Obrázok 3.1.3a znázorňuje schéma tohto filtru.



Obrázok 3.1.3a Schéma zapojenia rekonštrukčného Butterworthovho filtru štvrtého radu.

Pri návrhu tohto filtra bola použitá zjednodušená metóda kde majú všetky rezistory a všetky kondenzátory rovnakú hodnotu. Keďže sa jedná o filter štvrtého radu, po použití vzorca 2.4 dostaneme na medznom kmitočte -3 dB pre každý rad. To znamená že na medznom kmitočte by bola redukcia 12 dB, čo je príliš veľa na naše použitie. Preto bol medzný kmitočet zvolený s dostatočnou rezervou, a to 40 kHz.

Hodnota rezistorov bola zvolená na široko dostupnú a to 2.7 kΩ, a následne bola vypočítaná hodnota kondenzátora pomocou vzorca 2.4.

$$C = \frac{1}{2\,\pi\,f_c\,R}$$

$$C = \frac{1}{2\,\pi \cdot 40\,000 \cdot 2\,700}$$

$$C = \frac{1}{216\,000\,000\,\pi}$$

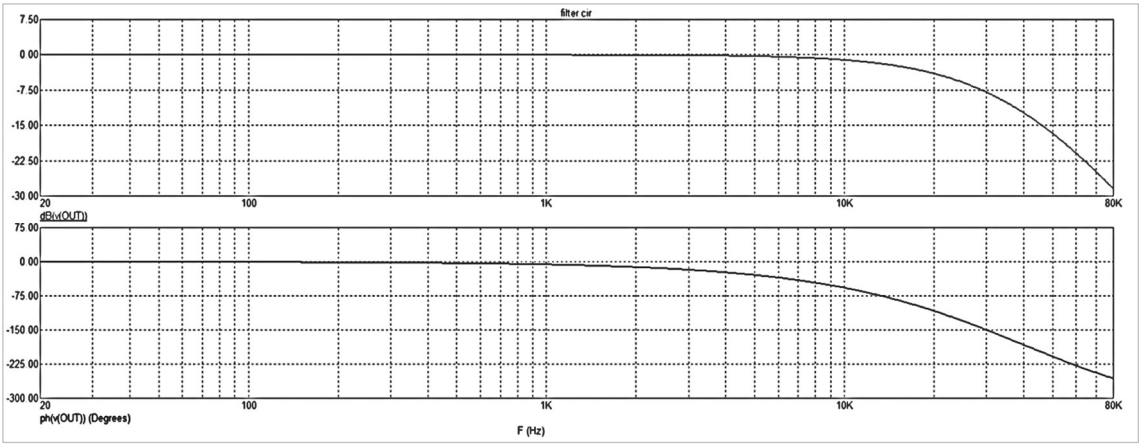
$$C \cong 1.4737 \cdot 10^{-9} \cong 1.47\,nF$$

Kdeže kondenzátor s hodnotou 1.47 nF nie je možné ľahko zakúpiť, bola táto hodnota zaokrúhlená na najbližšiu komerčne dostupnú hodnotu a to 1.5 nF.

Pri vybere kondenzatora boli moznosti vyrazne obmedzene, kedze zo zadania bolo mutne pouzit SMD prevedenia. Z tohto dovodu boli zvolene keramicke kondenzatory. Dve jasne moznosti boli C0G/NP0 a X7R. Pre nase pouzitie boli zvolene C0G/NP0 pre ich lepsiu stabilitu a lepsie operovanie pri vyzsich frekvenciach.

Obrázok 3.1.3b znázorňuje simulovanú charakteristiku tohoto obvodu. Tato simulacia prebehla v programe Micro-Cap 12.2.0.5. Je na nej badateľné, že na kmitočte 20 kHz je pokles približne -4 dB, táto hodnota je viac priaznivá ako -12 dB, ktoré by vznikli keby sme za medznú frekvenciu zvolili 20 kHz.

Taktiež môžeme pozorovať zmenu fázového posunu. Tá v rozmedzí do 20 kHz nepresiahne hodnotu 110 °.



Obrázok 3.1.3b Graf závislosti prenosu a fázy na frekvencii rekonštrukčného filtra.

3.1.4 Výber napájania

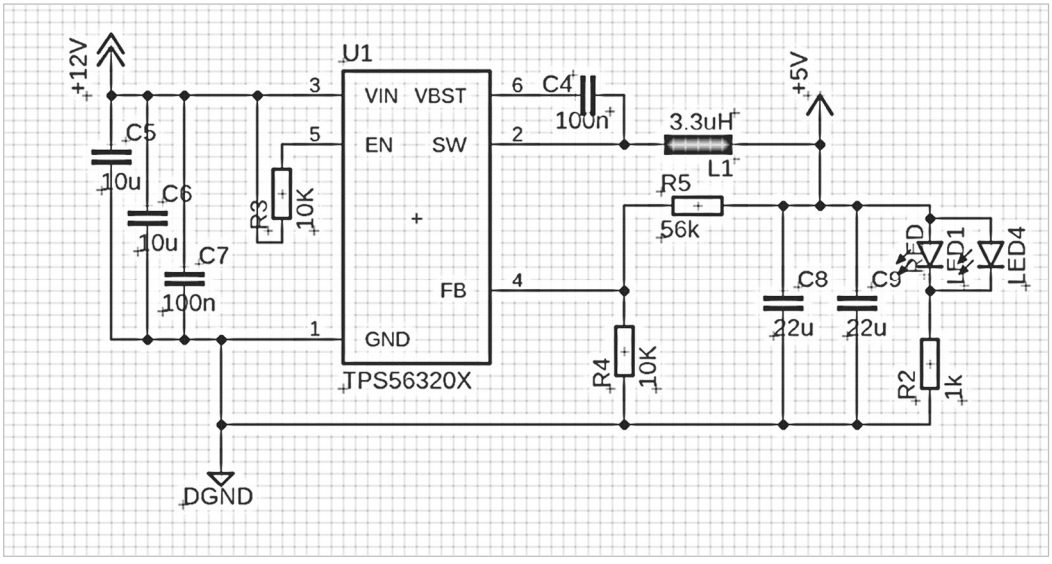
Prípravok bude zo zadania napájaný DC zdrojom zo symetrickými 12 V. Toto napätie je však príliš vysoké na napájanie ATmega328 ale aj D/A prevodníku.

Prvotný návrh bol založený na stabilizátore napätia LM7805, ale keďže sa jedná o stabilizátor napätia, tak nie je stavaný na veľké rozdiely v napätí. Aj keď podľa dokumentácií by mal tento čip zvládnuť znížiť našich 12 V na 5 V, tak by tak robil s veľmi nízkou účinnosťou. Toto by spôsobovalo značné prehrievanie, ktoré by výrazne komplikovalo realizáciu.

Druhou možnosťou bol DC-DC buck konvertor. V tomto prípade je účinnosť oveľa väčšia keďže sa jedná o takzvaný switching regulátor. To znamená, že sa tento konvertor rapídne vypína a zapína. Toto však prináša ďalšie problémy, ako napríklad kolísajúce referenčné napätie. Avšak v našom prípade sa týmto nemusíme veľmi zaoberať, keďže sa frekvencia týchto čipov pohybuje v rámci desiatok kHz. Táto zložka bude preto odfiltrovaná rekonštrukčným filtrom.

Na tento účel bol zvolený TPS563201. Jeho maximálne vstupné napätie je 17 V. To je dostatočná rezerva oproti našim 12 V. Jeho výstup je možné nakonfigurovať medzi 0.76 V a 7 V, čo spĺňa našu požiadavku na 5 V. Frekvencia tohto konvertoru je 580 kHz. Táto hodnota je vysoko nad 20 kHz ktoré bude náš prípravok generovať, takže taktiež nebude spôsobovať problémy. A na záver jeho výstupný prúd je maximálne 3 A, čo bude viac ako dostačujúce pre naše potreby.

Na obrázku 3.1.4 je znázornená schéma zapojenia tohto čipu.



Obrázok 3.1.4 Schéma zapojenia DC-DC buck konvertoru TPS563201

3.2 Ovládanie a užívateľské rozhranie

Ako bolo spomenuté na začiatku kapitoly, celý prípravok bude ovládaný jedným enkodérom s tlačidlom. Jeho potlačením bude môcť užívateľ prepínať medzi nastaviteľnými parametrami, ako napríklad tvar vlny, frekvencia alebo perióda, amplitúda a strieda.

Nastavenie obdĺžnikového signálu o frekvencii 100 Hz, amplitúdou 2 Vpp a striedou 30 %, môže vyzerať napríklad takto:

- Po správnom zapojení prípravku, na displeji začne blikať výber tvaru vlny.
- Následne užívateľ otočí enkodérom na obdĺžnikový signál a stlačením potvrdí.
- Na displeji začne blikať ďalšia položka, a to frekvencia.
- Užívateľ otočí enkodérom dokým nenastaví požadovanú frekvenciu 100 Hz (toto nastavenie bude mať inkrementy v logaritmickom merítku).
- Stlačením potvrdí.
- Následne začne blikať nastavenie amplitúdy, ktoré užívateľ nastaví a opäť stlačením potvrdí.
- Otáčaním enkodéru užívateľ vyberie striedu a opäť potvrdí potlačením.
- Prípravok hneď začne generovať fundament tohto periodického signálu.
- Následným otáčaním bude užívateľ meniť počet vyšších harmonických frekvencií (toto nastavenie bude v logaritmickom merítku).
- Keď bude chcieť užívateľ zmeniť nejaké nastavenie, tak sa jednoducho vráti na začiatok stlačením enkodéru a môže previesť toto nastavenie znova.

Po dokončení cyklu sa vráti nastavenie prípravku na začiatok, takže bude opäť možné nastaviť tvar vlny, avšak nastavené hodnoty zostanú nepozmenené. To znamená že ak potrebuje užívateľ zmeniť iba jednu veličinu, ostatné bude stačiť iba potvrdiť a prejsť ďalej.

Počas procesu nastavovania nebude prípravok generovať žiaden signál. Keby tak bolo, mohlo by dôjsť ku artefaktom pri updatovaní displeja. Toto by mohlo komplikovať meranie alebo prípadne poškodiť meracie prístroje.

Ako displej bude použitý LCD panel s rozlíšením aspoň 128x64 pixelov, a to z dôvodu aby vedel poňať všetky potrebné údaje naraz a bolo z neho očividné aké hodnoty sú nastavené na prípravku.

3.3 Preprogramovanie a diagnostika

Pre nahranie základného programu do mikrokontroleru ATmega328 je použitý FTDI adaptér. Tento adaptér je použitý ako USB komunikácia s mikrokontrolerom. Na základovej doske sú pripravené vývody na pripojenie tohto adaptéru pre jednoduché nahratie programu. V prípade že bude potrebné v programe vykonať zmeny, prípravok

bude možné jednoducho rozobrať a následne len pripojiť FTDI adaptér a nahrať program.

Napriek značným ochranným opatreniam, bude prípravok využívaný najmä študentami. To znamená, že bude často vystavovaný neprofesionálnej manipulácii, čo značne zvyšuje riziko jeho poškodenia. Preto je prípravok vybavený kontrolnými bodmi, na zjednodušenie možných opráv. Každý podstatný vývod z mikrokontroleru, D/A prevodníku, DC-DC buck konvertoru a operačného zosilňovača má sprístupnený kontakt, ktorý umožní jednoduché meranie a hľadanie porúch.

3.4 Schéma

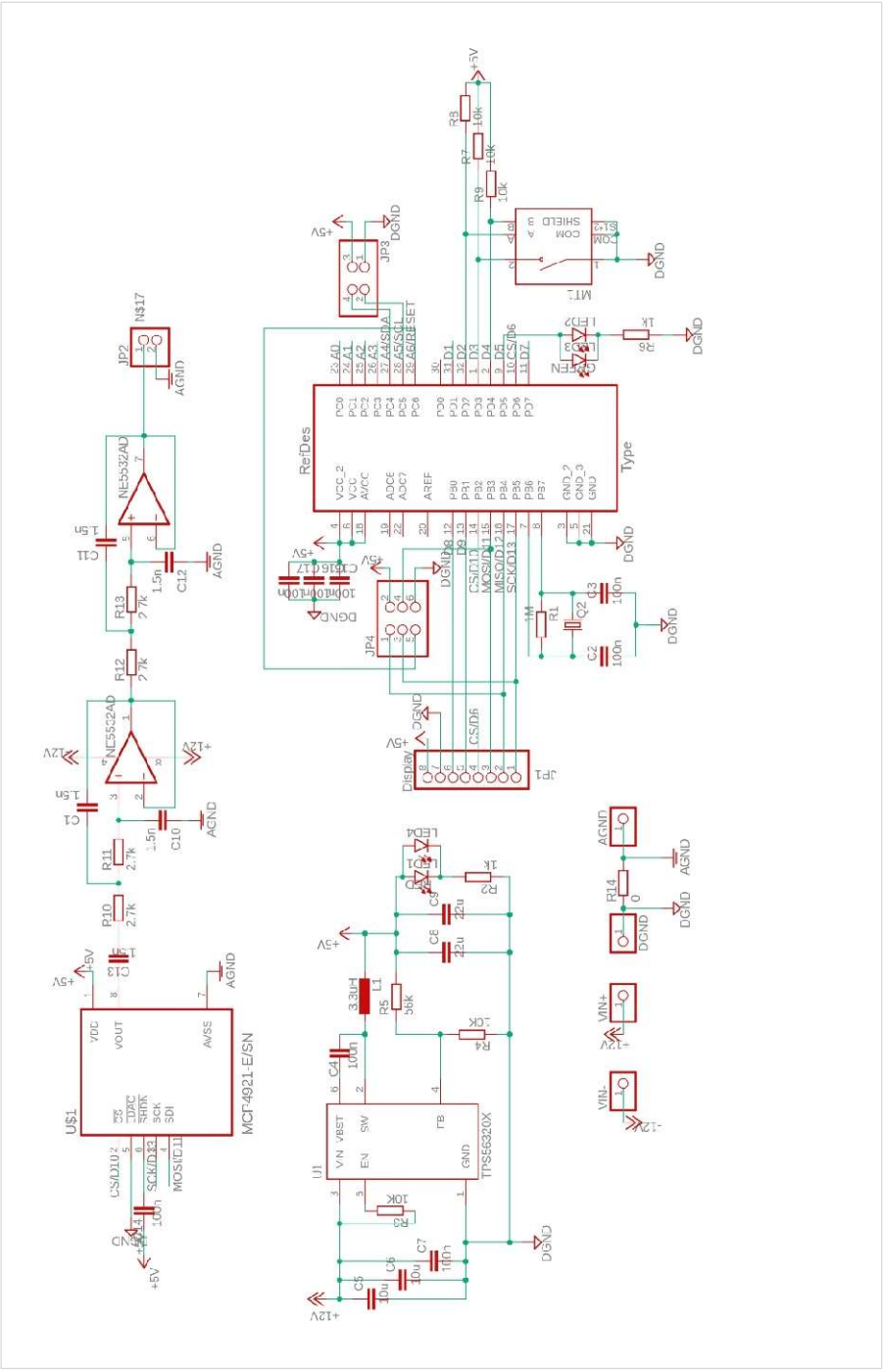
3.4.1 Tabuľka komponentov

Tabuľka 3.4.1 Tabuľka komponentov

Súčiastka	Názov	Puzdro	Hodnota
Rezistor	R1	0603	1 MΩ
Rezistor	R2, R6	0603	1 kΩ
Rezistor	R3, R4, R7, R8, R9,	0603	10 kΩ
Rezistor	R5	0603	56 kΩ
Rezistor	R10, R11, R12, R13	0603	2.7 kΩ
Rezistor	R14	0603	0 Ω
Kondenzátor	C1, C10, C11, C12, C13	0603	1.5 nF
Kondenzátor	C2, C3, C4, C7, C14, C15, C16, C17	0603	100 nF
Kondenzátor	C5, C6,	0603	10 µF
Kondenzátor	C8, C9,	0603	22 µF
Tlmivka	L1	2012	3.3 µH
Mikrokontroler	ATmega328p	TQFP	-
Op-Amp	NE5532	8 SOIC	-
D/A prevodník	MCP4921	8 SOIC	-
DC-DC buck c.	TPS563201	SOT23-6	-
LED	LED 3	0805	Zelená
LED	LED 4	0805	Červena
LED	LED 1	5 mm	Červena
LED	LED 2	5 mm	Zelená
Enkodér	MT1	S0024	24 ink.
Displej	Displej	2,4 palca	128x64p

3.4.2 Schéma prípravku

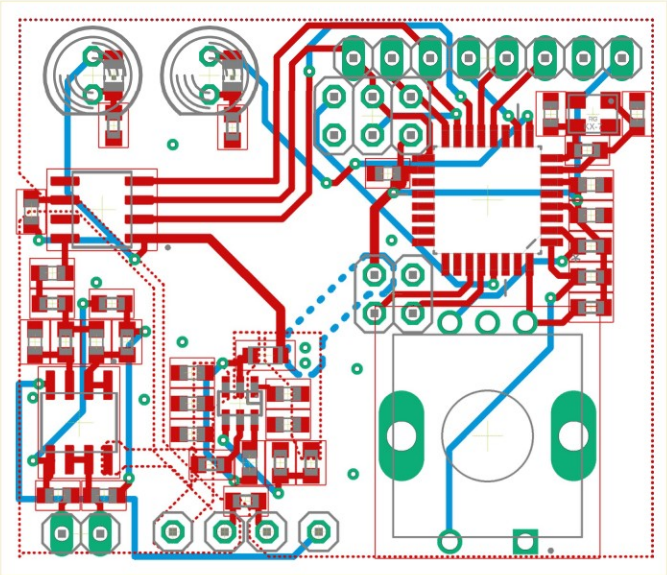
Schéma prípravku bolo navrhnute v programe Eagle verzie 9.6.2.



Obrázok 3.4.2 Schéma prípravku

3.4.3 Doska plošných spojov

Doska plošných spojov bola navrhnutá v programe Eagle verzie 9.6.2.



Obrázok 3.4.3 Doska plošných spojov v pomere 2:1

3.5 Ochrana

Keďže budú s prípravkom pracovať prevažne študenti, je potrebné navrhnúť aj ochranu. Hlavnou prioritou bude v tomto prípade ochrana proti prepoľovaniu, keďže náš DC-DC buck konvertor zvládne ochrániť proti príliš vysokému napätiu.

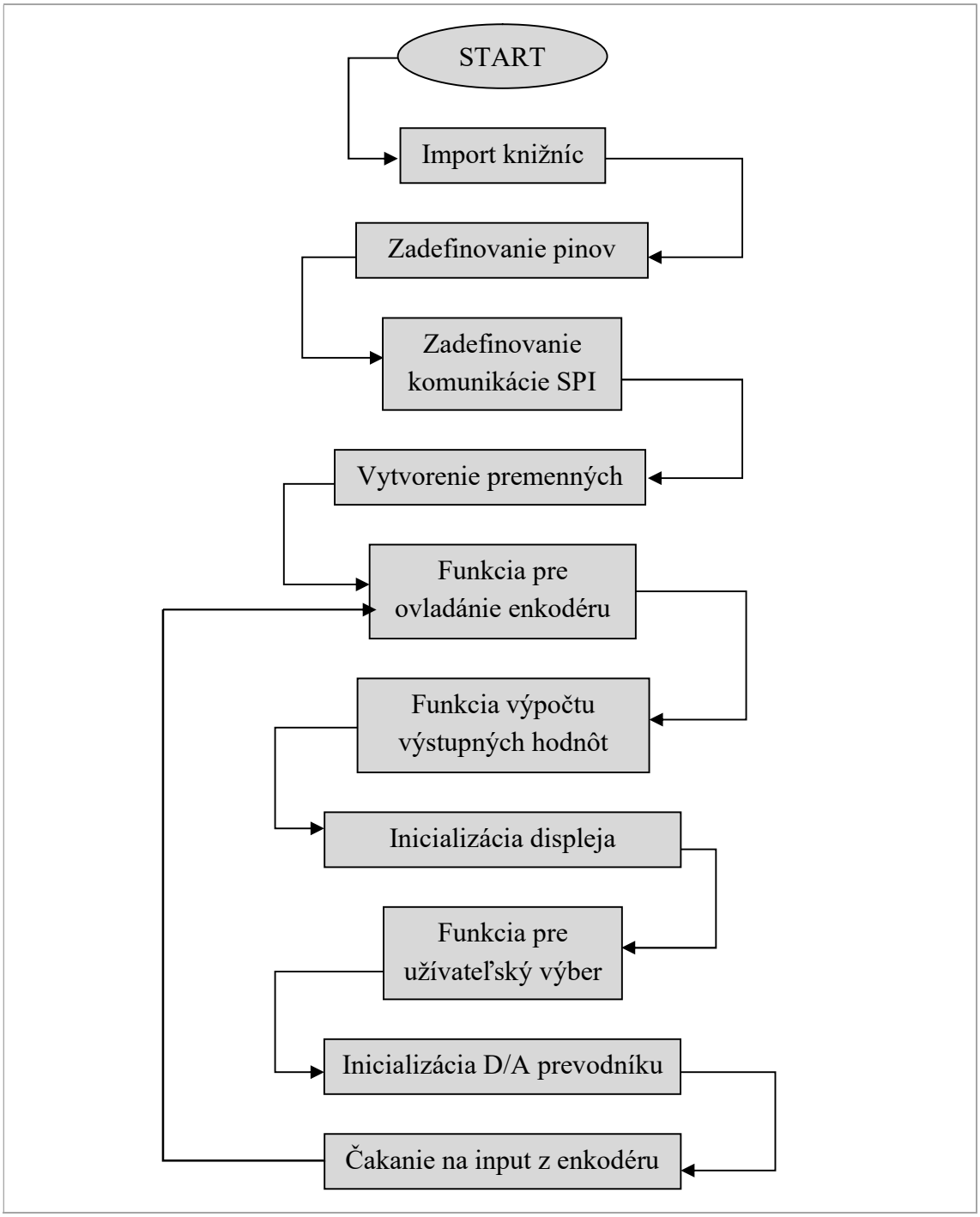
Najjednoduchší spôsob ochrany proti prepoľovaniu je použitie diódy na vstupe, avšak tu vzniká problém s prehrievaním. Keďže úbytok napätia na bežnej dióde je približne 0.8 V, by pri teoretickom prúde 2 A bolo 1.6 W premenených na teplo v tejto dióde. Toto by spôsobovalo prehrievanie a pravdepodobne aj zničenie prípravku. Jedným z riešení je použitie Schottkyho diódy ktorá ma nižší úbytok napätia (okolo 0.5 V), avšak toto nie je stále dostačujúce. Jednoduché riešenie tohto problému je zapojenie diódy paralelne so zdrojom. Toto však spôsobí skrat zdroja pri prepoľovaní. Preto je nutné použiť poistku, ktorá rozopne obvod pri jeho skrate.

Bežne je poistky nutné po každom použití vymeniť. Toto by značne obmedzovalo vyučujúceho ktorý by musel neustále prípravok rozoberať, a taktiež by to bolo pre školu nákladné. Na riešenie tohto problému môžeme použiť napríklad PTC poistku (z anglického Positive Temperature Coefficient). Táto poistka sa pri dosiahnutí vyššej teploty pri skrate rozpojí, a následne po odpojení vychladne a obvod sa znovu uzavrie.

Toto zapojenie nie je vidieť na doske plošných spojov z dôvodu väčších rozmerov súčiastok. Tieto budú zapojené do série pred prípravok, avšak stále skryté v krabičke.

3.6 Softvér

3.6.1 Vývojový diagram



Obrázok 3.6.1 Vývojový diagram kódu prípravku

3.6.2 Programový kód

Programovanie bolo realizovanie v programe Arduino IDE v 2.3.2.

Program začína základným importom funkcií, zadefinovaním použitých pinov a inicializovanie SPI komunikácie.

Ovládanie pomocou enkodéru bolo zrealizovane pomocou funkcie Interrupt. Táto funkcia umožňuje ukončiť všetky procesy a uskutočniť funkciu, ako napríklad otočenie enkodéru, alebo stlačenie tlačidla.

```
void encoder(){
  if (millis() - lastEncoder >100){
    if (digitalRead(ENpinDT)){
      counter --;
    }
    else{
      counter ++;
    }
  functionCount();
  lastEncoder = millis();
  }
}
```

Hlavná funkcia je počítanie hodnôt do množiny, z ktorej sa následne generuje hodnota na výstupe.

```
    for(int k = 1; k <= harm; k++){
      for(int i = 0; i < 256; i++){
        sine_table[i] += 2 / ( k * M_PI ) * sin( k * M_PI * D ) * cos( 2
* M_PI * k * ( i / 256 ) );
      }
    }
```

Ako prvý sa začne vykonávať for loop ktorý sa vykoná toľkokrát, koľko chceme zobraziť vyšších harmonických vĺn. Následne sa začne plniť pole `sine_table` podľa vzorca 3.6.2. Tento vzorec udáva funkciu pre generovanie obdĺžnikového signálu s použitím striedy.

$$x(t) = \sum_{k=1}^{\infty} \left(\frac{2}{k\pi} \sin(\pi k D) \cos(2\pi f k t) \right) \qquad (2.4)$$

f je v našom prípade 1, keďže je nutné zaplniť celé pole, a t sa vzťahuje ku nášmu vnorenému for loopu, ktorý udáva index poľa. Preto $t =$ `i / 256`.

Zobrazovanie na displej je pomocou knižnice Adafruit_GFX.h. Táto umožňuje jednoduché „kreslenie“ na displej pomocou intuitívnych príkazov.

```
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(1, 5);
  display.println("SIGNAL: sin tr saw sq");
  display.setCursor(1, 21);
  display.println("FREQ:   10.00      Hz");
  display.setCursor(1, 37);
  display.println("AMP:    1,00      Vpp");
  display.setCursor(1, 53);
  display.println("STRIDA: 50          %");
  display.display();
```

Generovanie výstupu je založené na knižnici SPI.h.

```
void MCPwrite(byte AB, uint16_t v) {

    v |=0xf000;
    if (!AB)  v &= ~0x8000;

    SPI.beginTransaction(settingsA);
    digitalWrite(CSpinDAC, LOW);
    SPI.transfer((0xff00 & v)>>8 );
    SPI.transfer(0x00ff & v );
    digitalWrite(CSpinDAC, HIGH);
    SPI.endTransaction;
}
```

Hlavná funkcia `loop()` sa opakuje počas celého chodu prípravku. Na jej začiatku sa spustí funkcia na užívateľský výber hodnôt `selectValues()`. Následne sa rozsvieti LED a začne sa samotné generovanie. To je umožnené postupným generovaním jednotlivých hodnôt z tabuľky `sine_table`. Toto sa deje, dokým užívateľ nestlačí tlačidlo enkodéru. Následne sa LED vypne a celý proces sa začne od začiatku.

```
void loop() {
  selectValues();
  digitalWrite(pinLED, HIGH);
  while(!button){
    for (int i = 0; i < DAC_ARRAY_INDICES; i++) {
      MCPwrite( 0, sine_table[i] );
      delayMicroseconds(freqDelays[FREQ])
    }
  }
  digitalWrite(pinLED, LOW);
  button = 0;
}
```

3.7 Testovanie

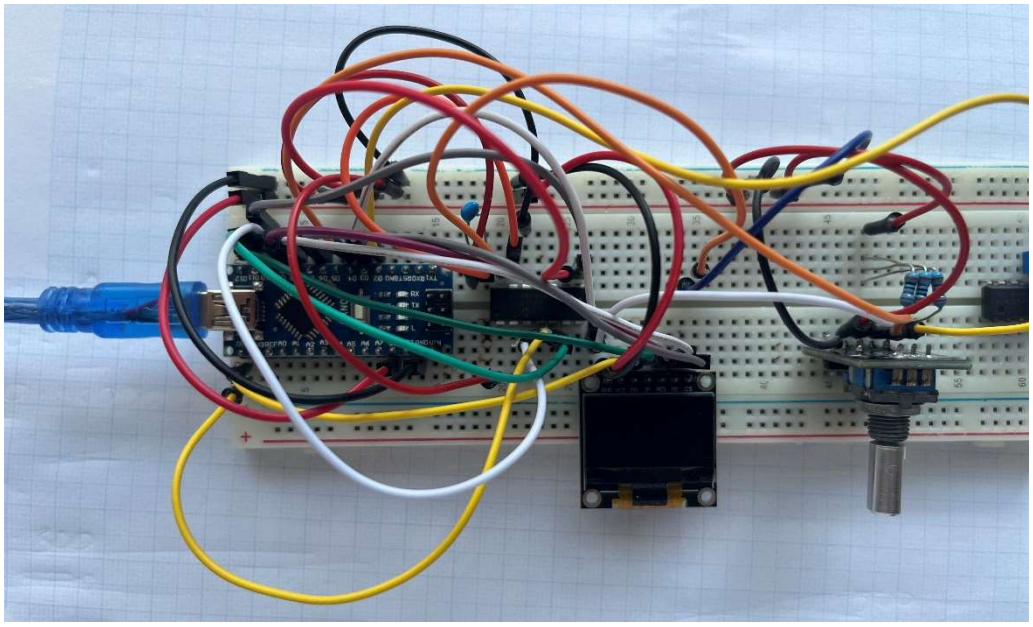
Testovanie prípravku prebehlo z hľadiska časovej efektivity na nepájivé kontaktné pole.

Pre účely testovania bolo použité Arduino Nano, nakoľko jeho mikrokontroler je taktiež ATmega328 a má vlastný DC-DC buck konvertor preto nebolo nutné použiť TPS563201.

Nakoľko bol prípravok testovaný pri „domácich“ podmienkach, a to pomocou externej zvukovej, nebolo nutné používať rekonštrukčný filter. Zvuková karta Focusrite Scarlett Solo II. má zabudovaný vstupný filter na 20 kHz a beztak nemá dostatočnú vzorkovaciu frekvenciu aby na nej bolo možné merať vyššie frekvencie.

3.7.1 Generovanie

Na fotografii 3.7.1 je vidieť prípravok v testovacej fáze.



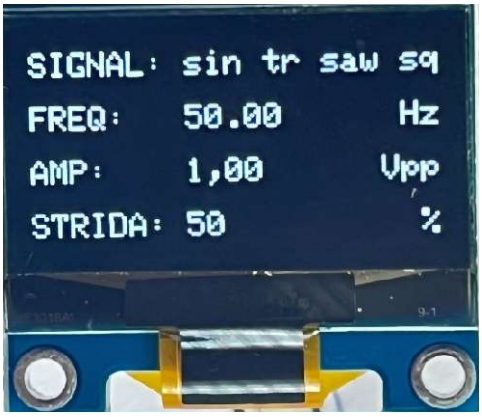
Fotografia 3.7.1 Prípravok vo fáze testovania

Na fotografii je badateľné, že testovaný D/A prevodník nebol MCP4921 ale MCP4922. Nakoľko prvá verzia používala prevodník MCP4725, ktorý komunikoval pomocou I^2C, a MCP4921 nebolo dostupné s rýchlym dátumom dodania, bola zvolená najbližšia dostupná alternatíva a to MCP4922. Jedná sa o rovnaký D/A prevodník s jediným rozdielom, a to tým, že disponuje dvoma kanálmi na rozdiel od jedného.

3.7.2 Užívateľské rozhranie

Fotografia 3.7.2 poukazuje na zobrazenie displeja prípravku. Použitý je OLED displej, nie LCD. Je to z toho dôvodu, že bol rovno k dispozícii z predošlých projektov a bolo možné ho hneď začať testovať. Nakoľko bude displej vo finálnej verzii taktiež ovládaný pomocou SPI, budú nutné iba minimálne zmeny v programovom kóde.

Displej na hotovom prípravku bude taktiež väčší ako displej na ktorom bol prípravok testovaný. Žiaci budú tak jednoznačne vidieť, čo sa s prípravkom odohráva.



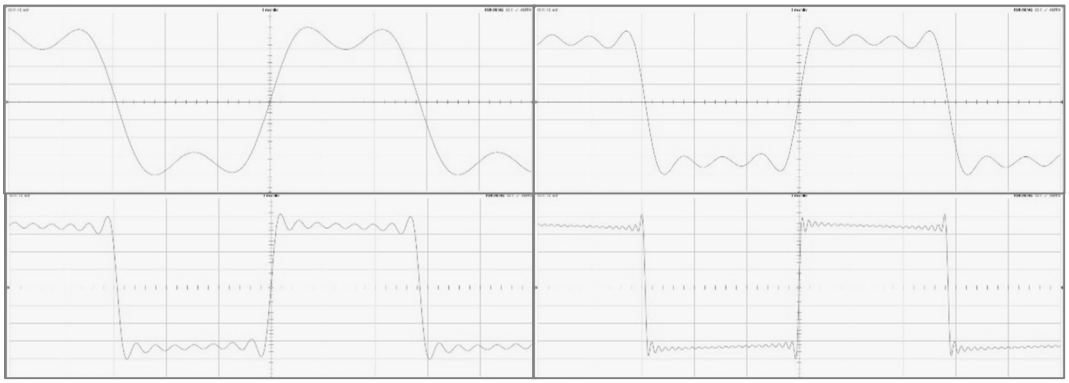
Fotografia 3.7.2 Zobrazenie na displeji

Hodnota, ktorá je v danom čase nastavovaná, má vo svojom okolí kurzor, zobrazený blikajúcim obdĺžnikom v 0.5-sekundových intervaloch. Zvolené hodnoty zostávajú zobrazené na displeji, pokým nie sú znova zmenené. Je tak preto, aby užívateľ vedel rýchlo povedať aké hodnoty sú na prístroji nastavene.

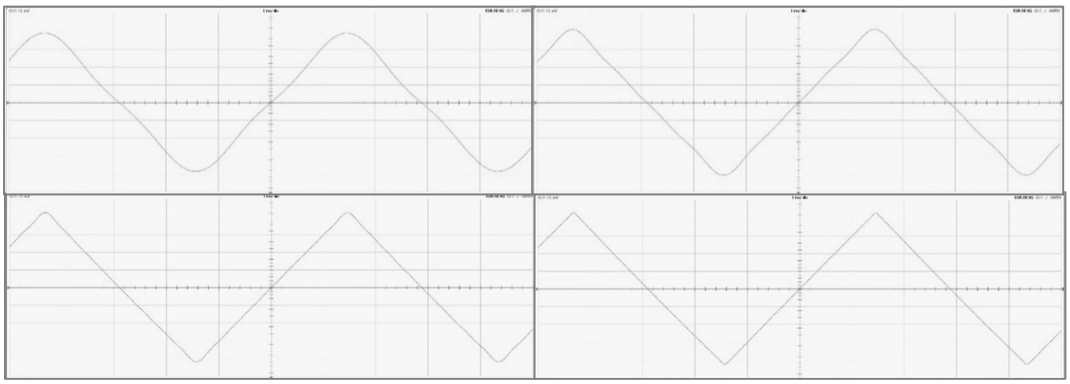
Po nastavení hodnôt sa rozsvieti zelena LED ktorá indikuje proces generovania a na displeji sa zobrazí počet vyšších harmonických vĺn.

3.8 Výstup prípravku

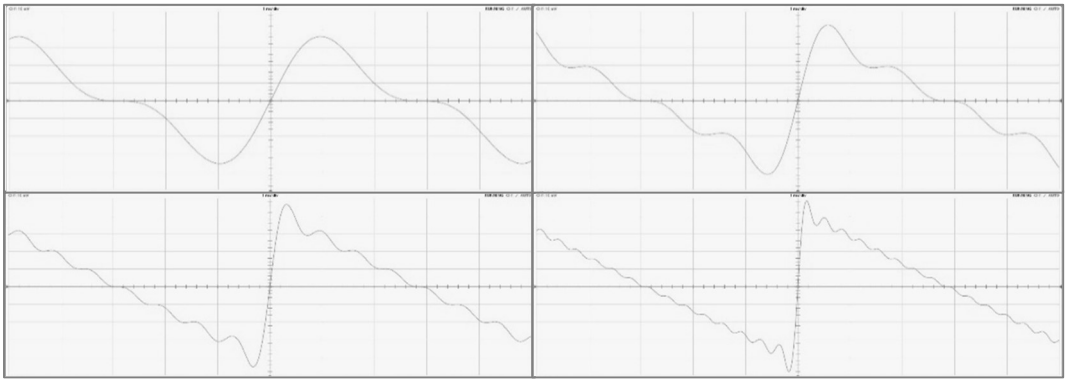
Prípravok umožňuje generovanie obdĺžnikového, trojuholníkového a pílového signálu. Obrázok 3.8.1 znázorňuje generovanie obdĺžnikového signálu, 3.8.2 znázorňuje generovanie trojuholníkového signálu a 3.8.3 generovanie pílového signálu. Jedná sa o vystúp prípravku, sledovaný pomocou zvukovej karty Focusrite Scarlett Solo II, a následne zobrazený v programe REW v5.31.1.



Obrázok 3.8.1 Generovanie obdĺžnikovej vlny so striedou 50 %



Obrázok 3.8.2 Generovanie trojuholníkovej vlny



Obrázok 3.8.3 Generovanie pílovej vlny

4. ZÁVER

Bol vyhotovený prípravok, ktorý dokáže generovať základné periodické signály ako obdĺžnikový, trojuholníkový alebo pílový. Prípravok je založený na mikrokontroleri ATmega328, ktorý komunikuje prostredníctvom SPI protokolu s D/A prevodníkom MCP4921. Výstupný signál je následne odfiltrovaný pomocou výstupného filtru založeného na operačnom zosilňovači NE5532, v prevedení Butterworthoveho aktívneho filtru štvrtého radu.

Napájanie prípravku je zabezpečené pomocou DC-DC buck konvertoru TPS563201. Tento umožňuje premeniť 12 V zo vstupu prípravku na 5 V, ktoré napájajú ATmega328. Vstup je zároveň ochránený PTC poistkami, proti prepoľovaniu.

Prípravok umožňuje študentom v praxi pozorovať, ako sa z vyšších harmonických vĺn, skladajú základné harmonické signály. Je možne nastaviť základný tvar, jeho frekvenciu, amplitúdu alebo aj striedu.

Všetko je umožnené pomocou jednoduchého a intuitívneho ovládania vďaka jednému enkodéru s tlačidlom a vďaka LCD displeju s rozlíšením 128x64, ktorý umožňuje elegantné zobrazovanie hodnôt.

V porovnaní s predošlým prípravkom, je aktuálny vyhotovený z novších komponentov, ktoré umožnili jeho niekoľkonásobne zmenšenie. Taktiež bolo výrazne zjednodušené ovládanie a užívateľské rozhranie. Predošlý prípravok bol ovládaný klávesnicou, ktorá síce umožnila jednoduchšie zadávanie hodnôt, avšak nebola veľmi intuitívna.

Tento prípravok umožní študentom ďalších ročníkov interaktívne analyzovať signály a umožní príjemnejšie splnenie úlohy na predmete Analýzy signálov a sústav.

LITERATÚRA

[1] *SMÉKAL, Zdeněk. Analýza signálů a soustav - BASS. Brno, 2012. Skripta. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ.*

[2] HE, Jiale, et al. An overview of principles and types of ADC and DAC. In: Journal of Physics: Conference Series. IOP Publishing, 2023. p. 012050.

[3] LEENS, Frédéric. An introduction to I 2 C and SPI protocols. *IEEE Instrumentation & Measurement Magazine*, 2009, 12.1: 8-13.

[4] PACTITIS, S. A. Active filters: theory and design. CRC Press, 2018.

[5] ALL ABOUT CIRCUITS. Practical Filter Design Challenges and Considerations for Precision ADCs [online]. 2016 [cit. 2023-12-10]. Dostupné z: https://www.allaboutcircuits.com/industry-articles/practical-filter-design-challenges-and-considerations-for-precision-adcs/

[6] GRAY, R. Oversampled sigma-delta modulation. IEEE Transactions on Communications, 1987, 35.5: 481-489.

[7] Understanding I2C. In: YouTube [online]. 18.04.2023 [cit. 13.12.2023]. Dostupné z: https://www.youtube.com/watch?v=CAvawEcxoPU. Kanál užívateľa Rohde Schwarz.

[8] Understanding SPI. In: YouTube [online]. 12. 4. 2023 [cit. 17.05.2024]. Dostupné z: https://www.youtube.com/watch?v=0nVNwozXsIc. Kanál užívateľa Rohde Schwarz.

PRÍLOHA A – ZDROJOVÝ KÓD PRÍPRAVKU

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

#define MOSIpin 11 // display - D1, DAC - SDI
#define CLKpin 13 // display - D0, DAC - SCK
#define DCpin 8 // display - DC
#define CSpinDAC 10 // DAC - SC and LDAC
#define CSpinDISP 6 // display - CS
#define RESETpin 9 // display - RES

#define ENpinCLK 2 // encoder CLK
#define ENpinDT 4  // encoder DT
#define ENpinSW 3  // encoder SW

#define pinLED 7

#define DAC_RESOLUTION 8
#define DAC_ARRAY_INDICES (pow(2,DAC_RESOLUTION))

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &SPI, DCpin,
RESETpin, CSpinDISP);

SPISettings settingsA(4000000, MSBFIRST, SPI_MODE0);

bool GEN = 0;
int counter = 0;
int button = 0;
int SIGNAL = 0;
int lastCounter = 0;
long lastEncoder = 0;
long lastButton = 0;

int harm = 1;
float D = 0.5;
int freq = 100;
float amp = 2;

const int freqValues[] = {100,150,200,250,300,350,400,450,500};
const int freqDelays[] = {123,74,42,31,21,14,7,3,0};
```

```
float sine_table[256];

void encoder(){
  if (millis() - lastEncoder >100){
    if (digitalRead(ENpinDT)){
      counter --;
      if (harm > 1){
        harm += 10;
      }
    }
    else{
      counter ++;
      harm ++;
    }

  functionCount();

  lastEncoder = millis();
  }
}

void pressButton(){
  if (millis() - lastButton > 200){

    if(button < 2){
      button = button + 1;
    }
    else{
      button = 0;
    }

    harm = 1;

    functionCount();

    lastButton = millis();
  }
}

void functionCount(){
  for (int i = 0; i < 256; i++){
    sine_table[i] = 0;
  }

  if(SIGNAL == 0){ //squere
    for(int k = 1; k <= harm; k++){
```

```
      for (int i = 0; i < 256; i++){
        sine_table[i] +=  2 * pow( k * M_PI , -1 ) * sin( M_PI * k * D )
*  cos( 2 * M_PI * k * ( i * 0.00390625 ) ) ;
      }
    }
  }

  else if(SIGNAL == 1){ //triangle
    for(int k = 1; k <= harm; k++){
      for (int i = 0; i < 256; i++){
        sine_table[i] += 8*pow(M_PI, -2)*pow(-1, k-1 ) * pow(2*k-1, -2 )
* sin(2*M_PI* (2*k-1) * (i*0.00390625) ) ;
      }
    }
  }

  else if(SIGNAL == 2){ //saw
    for(int k = 1; k <= harm; k++){
      for (int i = 0; i < 256; i++){
        sine_table[i] += 2 * pow(M_PI, -1 ) * sin(2*M_PI* k *
(i*0.00390625) ) * pow(k, -1);
      }
    }
  }

  for (int i = 0; i < 256; i++){
    sine_table[i] = sine_table[i] * 1500 + 1800;
  }
}

void selectSignal(){
  while(!button){
    while(counter == 0 && !button){
      display.drawRect(47, 3, 21, 11, WHITE);
      display.display();
      delay(500);
      display.drawRect(47, 3, 21, 11, BLACK);
      display.display();
      delay(500);
    }
    while(counter == 1 && !button){
      display.drawRect(71, 3, 15, 11, WHITE);
      display.display();
      delay(500);
      display.drawRect(71, 3, 15, 11, BLACK);
      display.display();
      delay(500);
```

```
    }
    while(counter == 2 && !button){
      display.drawRect(89, 3, 21, 11, WHITE);
      display.display();
      delay(500);
      display.drawRect(89, 3, 21, 11, BLACK);
      display.display();
      delay(500);
    }
    while(counter == 3 && !button){
      display.drawRect(113, 3, 15, 11, WHITE);
      display.display();
      delay(500);
      display.drawRect(113, 3, 15, 11, BLACK);
      display.display();
      delay(500);
    }
    if (counter > 3){
      counter = 0;
    }
    else if (counter < 0){
      counter = 3;
    }
  }
  SIGNAL = counter;

  button = 0;
  counter = 0;
}


void selectFreq(){
    while(!button){
      display.drawRect(47, 19, 35, 11, WHITE);
      display.display();
      delay(500);
      display.fillRect(47, 19, 35, 11, BLACK);
      display.setCursor(49, 21);
      if (counter > 8){
        counter = 8;
      }
      else if (counter < 0){
        counter = 0;
      }
      display.print(freqValues[counter]);
      display.println(".00");
      display.display();
```

```
      delay(500);
  }
  FREQ = counter;
  counter = 0;
  button = 0;
}

void selectAmp(){
    while(!button){
      display.drawRect(47, 35, 35, 11, WHITE);
      display.display();
      delay(500);
      display.fillRect(47, 35, 35, 11, BLACK);
      display.setCursor(49, 37);
      if (counter > 8){
        counter = 8;
      }
      else if (counter < 0){
        counter = 0;
      }
      display.println((counter*0.5)+1);
      display.display();
      delay(500);
  }
  AMP = (counter*0.5)+1;
  counter = 0;
  button = 0;
}

void selectStrida(){
    while(!button){
      display.drawRect(47, 51, 25, 11, WHITE);
      display.display();
      delay(500);
      display.fillRect(47, 51, 25, 11, BLACK);
      display.setCursor(49, 53);
      if (counter > 8){
        counter = 8;
      }
      else if (counter < 0){
        counter = 0;
      }
      display.println((counter+1)*10);
      display.display();
      delay(500);
  }
  STRIDA = (counter+1)*10;
```

```
  counter = 0;
  button = 0;
}

void selectValues(){
  selectSignal();
  selectFreq();
  selectAmp();
  selectStrida();
}

void MCPwrite(byte AB, uint16_t v) {

    v |=0xf000;
    if (!AB)  v &= ~0x8000;

    SPI.beginTransaction(settingsA);
    digitalWrite(CSpinDAC, LOW);
    SPI.transfer((0xff00 & v)>>8 );
    SPI.transfer(0x00ff & v );
    digitalWrite(CSpinDAC, HIGH);
    SPI.endTransaction;
}

void setup() {
  Serial.begin(115200);

  pinMode(CSpinDAC, OUTPUT);
  pinMode(CSpinDISP, OUTPUT);

  SPI.begin();

  attachInterrupt(digitalPinToInterrupt(ENpinCLK), encoder, LOW);
  attachInterrupt(digitalPinToInterrupt(ENpinSW), pressButton, LOW);

  pinMode(ENpinDT, INPUT);
  pinMode(pinLED, OUTPUT);

  digitalWrite(pinLED, LOW);

  display.begin(SSD1306_SWITCHCAPVCC);

  display.display();

  //display.setRotation(2);

  display.setTextSize(1);
```

```
  display.setTextColor(WHITE);
  display.setCursor(1, 5);
  display.println("SIGNAL: sin tr saw sq");
  display.setCursor(1, 21);
  display.println("FREQ:   50.00      Hz");
  display.setCursor(1, 37);
  display.println("AMP:    1,00      Vpp");
  display.setCursor(1, 53);
  display.println("STRIDA: 50          %");
  display.display();

  functionCount();

}

void loop() {
  selectValues();

  digitalWrite(pinLED, HIGH);
  while(!button){

    for (int i = 0; i < DAC_ARRAY_INDICES; i++) {
      MCPwrite( 0, sine_table[i] );
      delayMicroseconds(freqDelays[FREQ])
    }
  }
  digitalWrite(pinLED, LOW);
  button = 0;

}
```