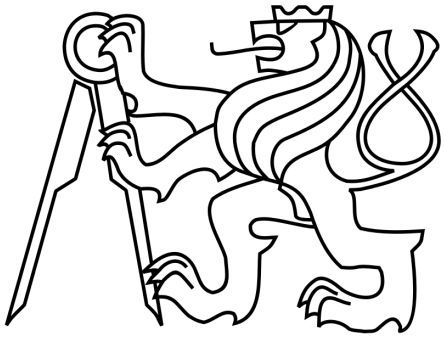
ČESKÉ VYSOKÉ UČENIE TECHNICKÉ V PRAHE

FAKULTA ELEKTROTECHNICKÁ



Programovo riadený priamy číslicový syntetizátor

Program Controlled Direct Digital Synthesizer

Diplomová práca

Marek Antoška

Vedúci práce: Ing. Michal Brejcha Ph.D.

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra elektroenergetiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Marek Antoška**

Studijní program: Elektrotechnika, energetika a management
Obor: Elektroenergetika

Název tématu: **Programově řízený přímý číslicový syntetizátor**

Pokyny pro vypracování:

1. Seznamte se se základním principem přímé číslicové syntézy kmitočtu (DDS).
2. Navrhněte obvodové řešení s obvodem DDS. Vyberte vhodný procesor pro řízení obvodu a navrhněte periferní obvody pro ovládání, zobrazování a komunikaci s PC.
3. Navrhněte, vyrobte, osaďte a oživte desku plošného spoje.
4. Vytvořte řídící program pro zvolený mikroprocesor a ověřte správnou funkci obvodu.
5. Vytvořte ovládací program pro PC, který usnadní možnosti nastavení daného obvodu - např. rozmítání frekvence.
6. Měřením ověřte funkce a výstupní průběhy obvodu.

Seznam odborné literatury:

[1] A Technical Tutorial on Digital Signal Synthesis. Analog Devices [online]. 1999 [cit. 2015-10-26]. Dostupné z: http://www.ieee.li/pdf/essay/dds.pdf
[2] CMOS, 125 MHz Complete DDS Synthesize, AD9850: Datasheet. Analog Devices, 2004. Dostupné z: http://www.analog.com/media/en/technical-documentation/data-sheets/AD9850.pdf
[3] VOBECKÝ, Jan a Vít ZÁHLAVA. Elektronika: součástky a obvody, principy a příklady. 3., rozš. vyd. Praha: Grada Publishing, 2005, 220 s. ISBN 80-247-1241-5.

Vedoucí: Ing. Michal Brejcha, Ph.D.

Platnost zadání: do konce zimního semestru 2017/2018

L.S.

doc. Ing. Zdeněk Müller, Ph.D.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 18. 4. 2016

Čestné prehlásenie

Prehlasujem, že som predloženú prácu vypracoval samostatne a že som uviedol všetky použité informačné zdroje v súlade s Metodickým pokynom o dodržiavaní etických princípov pri príprave vysokoškolských záverečných prác.

V Prahe, 20. mája 2016 _____

Abstrakt

Diplomová práca sa zaoberá návrhom a realizáciou sínusového generátora priamej číslicovej syntézy kmitočtu ako po hardvérovej stránke, tak aj po stránke programovej. Tematicky je rozdelená na 5 častí. V prvej časti je popísaný stručný princíp priamej číslicovej syntézy. V druhej časti je obsiahnutý vlastný návrh a konštrukcia hardvérovej časti syntetizátora. Tretia časť je venovaná programovému vybaveniu riadiaceho mikroprocesora. Štvrtá časť sa zaoberá realizáciou ovládacej aplikácie pre osobný počítač. V poslednej, piatej časti sú vykonané merania na skonštruovanom zariadení.

Kľúčové slová

DDS, priama číslicová syntéza kmitočtu, PIC18F252, AD9851, LCD, Bluetooth, USB, ošetrenie zákmitov

Abstract

This thesis discusses the hardware and software of design and construction of direct digital synthesis sine-wave generator. It is divided into 5 parts. In the first part the basic principle of direct digital synthesis is described. The design and realization of custom hardware part of synthesizer are presented in the second part of the thesis. In the third part is presented source code for microprocessor. The source code of PC aplication is in the fourth part. The last part describes realized measuremets on the constructed device.

Keywords

DDS, direct digital synthesis of frequency, PIC18F252, AD9851, LCD, Bluetooth, USB, debouncing

Poďakovanie

Týmto by som chcel poďakovať svojmu vedúcemu diplomovej práce Ing. Michalovi Brejchovi Ph. D. za jeho čas, cenné rady a pripomienky.

Obsah

Zoznam obrázkov

Zoznam tabuliek

1. Úvod

Generátory kmitočtu patria medzi najzákladnejšiu výbavu pri vývoji alebo diagnostike elektrotechnických zariadení.

Tradičné analógové generátory sínusového signálu, založené predovšetkým na využití vlastností LC alebo RC obvodov, dosahujú dobré parametre v oblasti skreslenia. Ich nevýhodou je však obtiažna preladiteľnosť, nízka presnosť výstupného kmitočtu, amplitúdy a opakovateľnosti nastavenia. V praktickej realizácii sú často vybavené prepínaním frekvenčných rozsahov, čo znižuje užívateľský komfort a nedovoľuje rozmietanie v širokom spektre.

Moderný vývoj rieši väčšinu nedostatkov analógových generátorov prechodom k digitálnym metódam generovania signálu. Jednou z týchto metód je priama číslicová syntéza kmitočtu, ktorá je využitá aj v tejto práci.

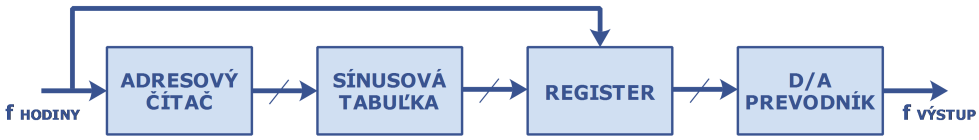
V nasledujúcom texte je stručne vysvetlený princíp priamej číslicovej syntézy a s jej využitím kompletne navrhnutý a zrealizovaný frekvenčný syntetizátor.

1. Priama číslicová syntéza

Priama číslicová syntéza (DDS) je výkonná metóda produkcie analógového signálu, predovšetkým sínusového. Svoje využitie nájde najmä v aplikáciach vyžadujúcich vysokú presnosť, rýchlosť a frekvenčnú preladiteľnosť, ako sú, napríklad, rádiové prijímače a vysielače, signálne generátory či obvodové analyzátory.

1.1 Základný princíp

Za najjednoduchšiu formu DDS môžeme považovať zapojenie na obrázku 1.1. Pozostáva z adresového čítača, programovateľnej pamäte ROM a digitálno-analógového prevodníka [1].



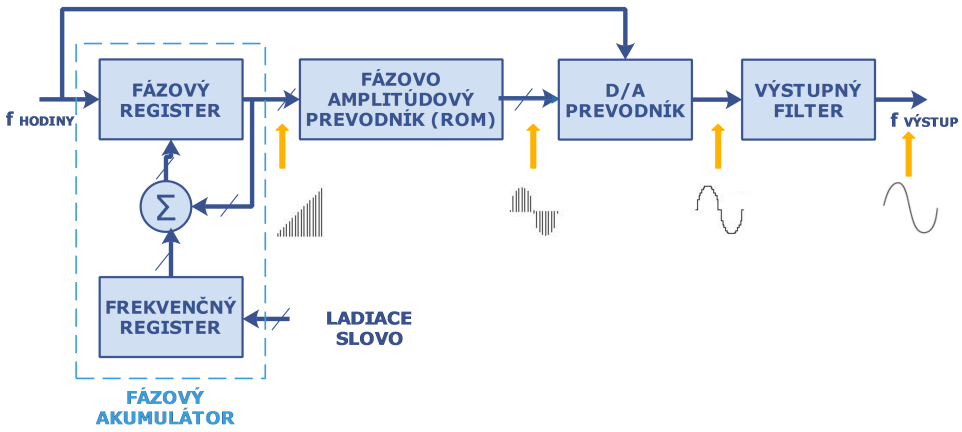
Obr. 1.1: Elementárny princíp priamej číslicovej syntézy

Referenčný hodinový signál f_{HODINY} v každom svojom cykle inkrementuje adresový čítač. Ten postupne prechádza všetky adresy pamäte ROM, na ktorých sú uložené odpovedajúce číselné hodnoty funkcie sínus. Tieto hodnoty sú následne v D/A prevodníku prevedené na analógový signál.

Základným problémom pri takto navrhnutej konfigurácii DDS je, že výstupná frekvencia závisí len na vstupnej hodinovej frekvencii alebo na počte krokov naprogramovaných v pamäti ROM. Tým pádom ladenie pripadá do úvahy len zmenou vstupnej hodinovej frekvencie alebo zmenou programu v pamäti RAM. Ani jedna z týchto možností však nie je ideálna, lebo neumožňuje vysokorýchlostné preladovanie.

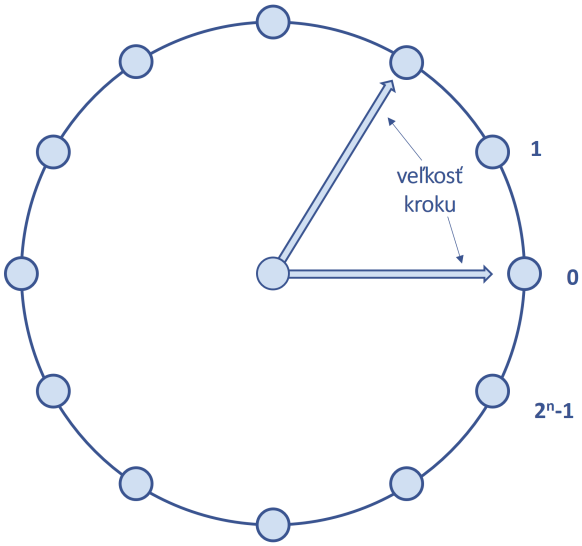
1.2 Princíp praktickej DDS

Hoci je predchádzajúci koncept DDS jednoduchý a funkčný, jeho ladenie je problematické, a rýchlejšie preladovanie je prakticky nemožné. Požiadavku rýchleho preladovania možno vyriešiť nahradením adresového čítača fázovým akumulátorom [1]. Schéma popísaného zapojenia je na obrázku 1.2.



Obr. 1.2: Princíp praktickej priamej číslicovej syntézy

Funkciu fázového akumulátoru si je možné predstaviť na "fázovej kružnici" zobrazenej na obrázku 1.3 [2]. Po obvode tejto kružnice sa pohybuje koncový bod rotujúceho vektora. Každý bod na obvode kruhu odpovedá bodu na generovanej sínusoide. Jedna otáčka vektora o 360 stupňov pri konštantnej rýchlosti predstavuje jednu periódu sínusového signálu. Počet bodov na obvode fázového kruhu je daný rozlíšením n fázového akumulátoru a pre najbežnejšie rozlíšenia je uvedený v tabuľke 1.1.



Obr. 1.3: Fázová kružnica

n	počet
8	256
12	4 096
16	65 535
20	1 048 576
24	16 777 216
28	268 435 456
32	4 294 967 296
48	281 474 976 710 656

Tab. 1.1: Počet fázových bodov v závislosti od rozlíšenia

Fázový akumulátor pri každom hodinovom cykle pripočítava ku svojej aktuálnej hodnote hodnotu ladiaceho slova uloženého vo frekvenčnom registri. Hodnota frekvenčného registra, nazývaná aj ladiace slovo, je nastavovaná externe a predstavuje veľkosť fázového kroku, respektíve určuje koľko krokov sa má vo fázovom kolese preskočiť. Čim je počet preskočených krokov vyšší, tým skôr dôjde ku pretečeniu akumulátora a tým skôr sa aj ukončí sínusový cyklus.

Napríklad pri rozlíšení fázového registra n = 32 bitov, počiatočnej hodnote fázové registra rovnej 0 a hodnoty uloženej vo frekvenčnom registri rovnej 1 by došlo k pretečeniu fázového akumulátora po 2^{32} (okolo 4 miliardy) cykloch hodinového signálu (pričítala by sa jednotka 2^{32} krát) a celý cyklus by sa opakoval znovu. Ak by pri zachovaní predchádzajúcich podmienok bola vo frekvenčnom registri napríklad uložená hodnota 2, k pretečeniu fázového akumulátora by došlo dvakrát rýchlejšie, a teda aj výstupná frekvencia by bola dvakrát vyššia.

Výstup fázového akumulátora je lineárny, presnejšie pílovitý, preto sa pre potreby generovania sínusového signálu, používa prevodná tabuľka fázovo amplitúdového prevodníku, pre funkciu sínus, ktorej výstup je pomocou D/A prevodníku prevedený na analógový sínusový signál.

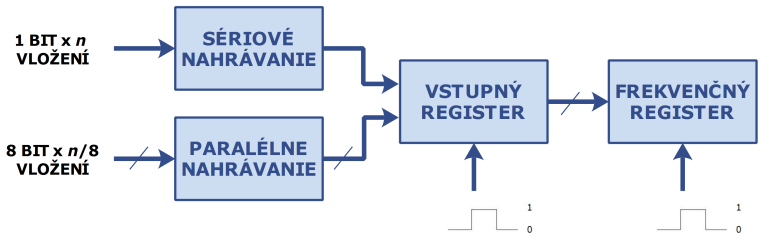
Niektoré DDS architektúry využívajú symetrickosť sínusovej funkcie a v prevodnej tabuľke ukladajú len rozsah od 0 do Pi/4 [3].

Závislosť medzi binárnou hodnotou ladiaceho slova *LS* vo frekvenčnom registri, rozlíšením fázového akumulátoru n v bitoch, a vstupnou hodinovou frekvenciou f_{HODINY} popisuje základná ladiaca rovnica DDS architektúry [4]:

$$f_{VÝSTUP} = \frac{LS * f_{HODINY}}{2^n}$$

kde, $f_{VÝSTUP}$ je frekvencia výstupného DDS signálu.

V praxi sa hodnota ladiaceho slova *LS* nahráva pomocou sériového, 1-bitového alebo paralelného, 1-bytové (8-bitového) rozhrania do vstupného registra odkiaľ sa zapisuje do frekvenčného registra [5].



Obr. 1.4: Postup nahrávania ladiaceho slova

Nahrávací postup závisí aj od použitého rozlíšenia *n* fázového akumulátora. Napríklad, pri rozlíšení 32 bitov a sériovom nahrávaní, je potrebné do vstupného registra 32 krát zapísať 1 bit. Pri rovnakom rozlíšení a paralelnom nahrávaní je treba zapísať 4 krát 1 byte. Jednotlivé bity/byty sa zapisujú do vstupného registra pomocou napäťového impulzu príslušného externého riadiaceho signálu. Po úspešnej sekvencii zápisov hodnôt do vstupného registra sa jeho obsah pomocou napäťového impulzu iného externého riadiaceho signálu uloží do frekvenčného registra v fázovom akumulátore.

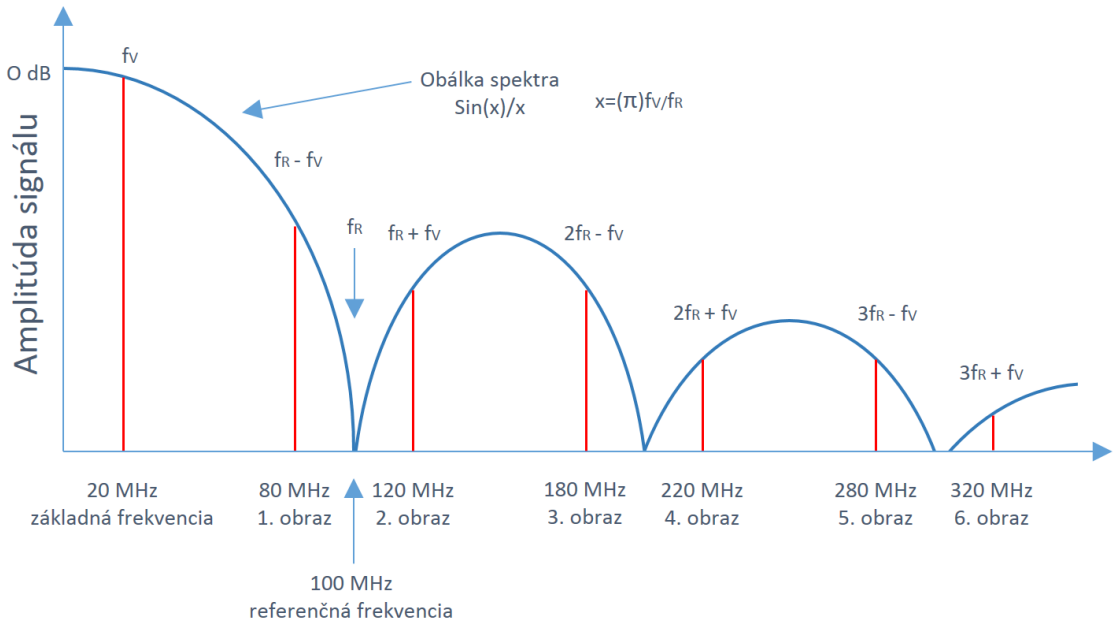
Týmto postupom je možné ladiť výstupnú frekvenciu plynulo, bez výpadku generovania frekvencie pri zápise ladiaceho slova a taktiež je možné meniť frekvenciu s vysokou dynamikou, čo sa dá využiť napríklad pri rozmietaní frekvencie alebo pri viacerých druhoch modulácii.

1.3 Výstupné spektrum

Teoretické výstupné spektrum vzorkovaného signálu D/A prevodníka, spĺňajúce Shannon-Nyquist-Kotělnikov teorém, obsahuje okrem základnej frekvencie aj obrazy, ktoré sa vyskytujú v násobkoch referenčnej frekvencie ± vybraná výstupná frekvencia [7].

Grafická reprezentácia výstupného spektra pre príklad referenčnej frekvencia 100 MHZ a základnej (výstupnej) frekvencia nastavenej na 20 MHz ja zobrazená na obrázku 1.5. Z obrázka je vidieť, že prvý obraz sa vyskytuje na frekvencii f_R-f_V, t.j. 80 MHz a má relatívne vysokú úroveň. Ďalšie obrazy sú vždy slabšie a vyskytnú sa na frekvenciách 120 MHz, 180 MHz, 220 MHz, 280MHz , 320 MHz atď.

Pre získanie sínusového signálu s čistým spektrom je nevyhnutné za D/A prevodník zaradiť filter, ktorého úlohou je potlačiť nežiadúce produkty prevodu. Takýmto filtrom môže byť, napríklad, dolnopriepustný filter, ktorý sa obvykle navrhuje na hodnotu 40% referenčnej frekvencie [5], čo znamená, že prepustí výstupné frekvencie len v rozsahu nula až 40% referenčnej frekvencie.



Obr. 1.5: Výstupné spektrum D/A prevodníka

2. Návrh a konštrukcia syntetizátora

Táto kapitola sa zoberá popisom návrhu a konštrukcie hardvérovej časti syntetizátora. Sú v nej rozobrané všetky súčasti syntetizátora, vrátane ich charakteristiky, voľby, až po zapojenie, poprípade realizáciu konkrétneho modulu.

2.1 Koncepcia syntetizátora

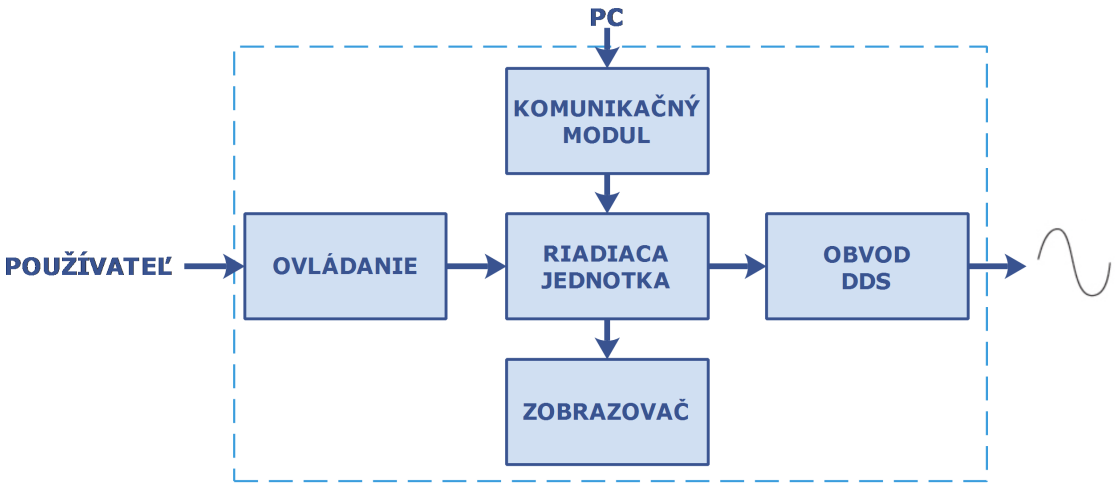
2.1.1 Požiadavky

Návrh a konštrukcia programovo riadeného DDS syntetizátora boli realizované v súlade s nasledujúcimi požiadavkami:

- frekvenčný rozsah aspoň 0-50 MHz
- minimálny frekvenčný krok 1 Hz
- užívateľské ovládanie ovládacími prvkami
- zobrazovanie aktuálnej frekvencie
- komunikácia s PC pre možnosť nastavenie frekvencie z PC
- frekvencia zobrazená

2.1.2 Bloková schéma

Vzhľadom na vstupné požiadavky tvorí navrhnutý DDS syntetizátor päť funkčných súčastí: DDS obvod, ovládanie, zobrazovač, komunikačný modul a riadiaca jednotka. Vzájomné vzťahy týchto súčastí sú znázornené na obrázku 2.1.



Obr. 2.1: Koncepcia programovo riadeného DDS syntetizátora

Riadenie DDS obvodu je navrhnuté dvomi nezávislými spôsobmi s možnosťou ich vzájomnej kombinácie:

1. Ručné ovládanie pomocou mechanických prvkov
2. Počítačové riadenie pomocou príslušného programu

Komunikácia s počítačom je realizovaná prostredníctvom komunikačného modulu, ktorého úlohou je previesť komunikačné rozhranie poskytované mikroprocesorom na rozhranie, ktorým sú bežne vybavené osobné počítače. Jadrom návrhu je riadiaca jednotka, ktorej funkciu plní mikroprocesor. Úlohy riadiacej jednotky sú nasledujúce:

1. Vhodne reagovať na používateľskú manipuláciu s ovládacími prvkami.
2. Prijímať počítačové riadiace signály pomocou komunikačného modulu.
3. Vypočítavať a zasielať ladiace slovo do DDS obvodu.
4. Zobrazovať aktuálnu frekvenciu a veľkosť kroku na zobrazovači

Samotnú priamu číslicovú syntézu kmitočtu zabezpečuje DDS obvod. Je ladený ladiacim slovom, ktoré mu vypočítava a zasiela mikroprocesor, na základe ktorého generuje signál sínusového priebehu.

Jednotlivé súčasti sú podrobne popísané v nasledujúcich častiach tejto kapitoly.

2.2 DDS obvod

Výrobou DDS obvodov sa zaoberá len veľmi málo firiem. Jednou z nich je Analog Devices, ktorá ako jediná poskytuje široký sortiment DDS obvodov, líšiacich sa predovšetkým frekvenčným rozsahom, rozlíšením a cenou. Príklad parametrov rôznych druhov DDS obvodov je v tabuľke 2.1.

Typ	Hodinová frekvencia	Rozlíšenie	Cena [6]
AD9850	125 MHz	32 bitov	530 Kč
AD9851	180 MHz	32 bitov	600 Kč
AD9854	300 MHz	42 bitov	830 Kč
AD9914	1 GHz	32 bitov	4 600 Kč
AD9834	75 MHz	28 bitov	250 Kč

Tab. 2.1: Príklad parametrov rôznych DDS obvodov

Výber vhodného obvodu bol realizovaný z ohľadom na fakt, že maximálna výstupná frekvencia predstavuje asi 40% z hodinovej frekvencie [3], čo pri rešpektovaní požiadavky na výstupnú frekvenciu 50 MHz, predstavuje minimálnu hodinovú frekvenciu 125 MHz, pričom platí, že čím je väčší odstup oboch frekvencií, tým ma sínusový signál menšie skreslenie. Ďalším kritériom výberu bola cena obvodu. Na základe týchto dvoch parametrov bol zvolený obvod AD9851.

2.2.1 DDS obvod AD9851

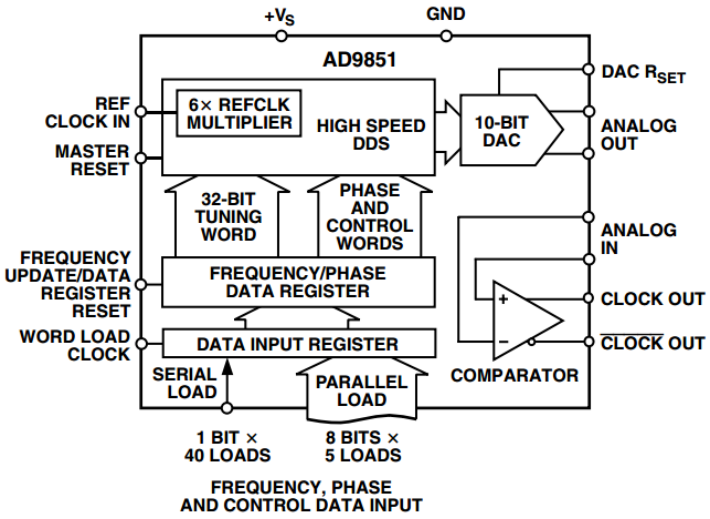
Obvod AD9851 firmy Analog Devices, využíva pokročilú technológiu DDS s vysokorýchlostným 10 bitovým D/A prevodníkom, čím sa stáva plne programovateľným frekvenčným syntetizátorom. Pri použití presného hodinového signálu generuje stabilnú frekvenciu a fázu výstupného analógového sínusového signálu [7].

Vysokorýchlostné DDS jadro poskytuje 32-bitové frekvenčné ladiace slovo, ktoré je pri použití maximálneho možného referenčného kmitočtu 180 MHz, možné prelaďovať s krokom 40 miliHerzov. AD9851 obsahuje unikátnu šesťnásobnú násobičku referenčného kmitočtu, ktorá odstraňuje potrebu vysokorýchlostného referenčného osciliátora (referenčný kmitočet 180 MHz možno dosiahnuť s 30 MHz oscilátorom). Zariadenie tiež poskytuje 5-bitové ovládanie fázy, ktoré umožňuje fázový zdvih v krokoch 180°, 90°, 45°, 22.5°, 11.25°, alebo ich kombináciách [7].

Ladiace slovo, skladajúce sa z frekvenčného ladiaceho slova a ladiaceho slova riadenia a fázy, je nahrávané asynchrónne paralelným alebo sériovým spôsobom [7].

Obvod AD9851 využíva technológiu CMOS, čo mu zaručuje nízku spotrebu, 555 mW pri napájaní 5V a maximálnom možnom referenčnom kmitočte 180 MHz [7].

Dostupný je len v 28-pinovom púzdre SSOP, určenom pre povrchovú montáž [7].



Obr. 2.1: Bloková schéma DDS obvodu AD9851 [7]

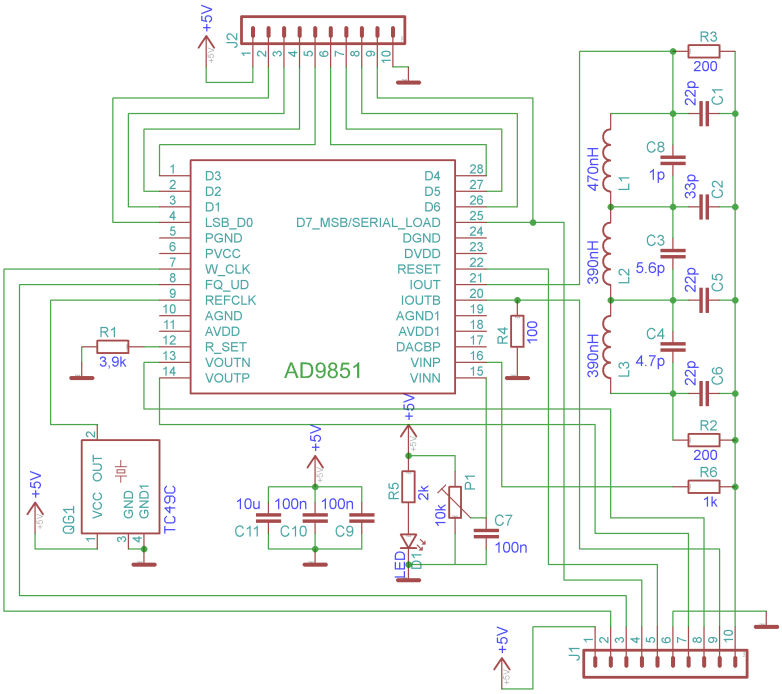
2.2.2 Modul s obvodom AD9851

K správnej funkcii je nutné k samotnému obvodu AD9851 pripojiť zdroj referenčného kmitočtu, výstupný filter a ďalšie pasívne súčiastky. Tieto súčiastky sa navrhujú vzhľadom k uvažovanému výstupnému frekvenčnému rozsahu zariadenia. Existuje však aj možnosť zakúpiť už hotový modul, so všetkými potrebnými súčiastkami už osadenými. Takýto modul je voľbou súčiastok prispôsobený na najvyšší možný frekvenčný rozsah obvodu AD9851.

Obr. 2.3: Modul s obvodom AD9851

Základné parametre modulu:

- kryštál 30MHz (s násobičkou 180 MHz)
- 70 MHz dolnopriepustný filter
- napájanie +5 V
- možnosť paralelného alebo sériového ladenia



Obr. 2.4: Schéma zapojenia modulu s obvodom AD9851

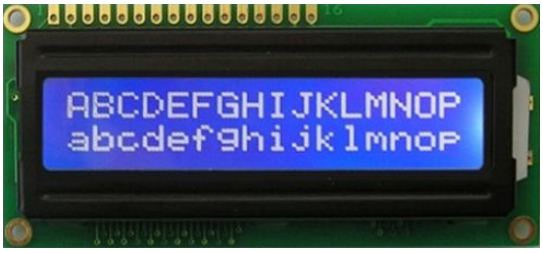
2.3 Zobrazovač

Zobrazovačom sa rozumie zariadenie, umožňujúce jednoduchým spôsobom zobrazovať vybrané informácie. Zobrazovač môže byť takmer čokoľvek, čo dokáže opticky zmeniť svoj charakter a podať tým výpovednú informáciu o stave sledovanej veličiny. Zobrazovačmi môžu byť napríklad displeje, svetelné indikátory, stupnice, mechanické návesti, atď.

V navrhovanom syntetizátore sú sledovanými veličinami nastavená frekvencia a nastavený krok. Vzhľadom k poskytovanému frekvenčnému rozsahu zvoleného DDS modulu - 70 000 000 Hz s minimálnym krokom 1 Hz, bol zvolený ako zobrazovač znakový displej LCD (liquid crystal display) s označením 16x02 .

2.3.1 LCD displej 16x02

Označenie „16x02“ znamená, že displej má 2 riadky a v každom z nich zobrazuje 16 znakov. Znaky môžu byť zobrazované v pixelovej matici 5x8 alebo 8x10. Má dva registre: príkazový a dátový. V príkazovom registry sú uložené príkazy pre displej. Príkazom je inštrukcia daná LCD displeju na vykonanie predurčenej činnosti, ako napríklad inicializácia, vymazanie obsahu, nastavenie kurzora atď. Dátový register ukladá dáta, ktoré sú displejom zobrazené. Dáta do dátového registra sú reprezentované ASCII hodnotou zobrazeného znaku. [8]



Obr. 2.5: LCD displej 16x02

LCD displeje majú paralelné rozhranie, ktoré pozostáva z nasledujúcich pinov:

- **Register select (RS) pin** určuje, do ktorého registra budú zapisované dáta. Logickou úrovňou tohto pinu je možné zvoliť ako dátový tak aj riadiaci register.
- **Read/Write (R/W) pin** určuje režim čítania alebo zápisu
- **Enable pin** umožňuje zápis do registrov
- **8 dátových pinov (D0 -D7)** predstavuje hodnoty zapisované do/z registrov pri zápise, poprípade čítaní.
- **Pin kontrastu (Vo).** Kontrast určený napätím v rozsahu 0 - 5 V
- **Piny napájania displeja** +5 V a Zem
- **Piny napájania podsvietenia** A+ a A-

Proces ovládania displeja spočíva v ukladaní dát do dátového registra a ich následného potvrdenia v riadiacom registri. Presný popis komunikácie s displejom však nebude v tejto kapitole popísaný, z dôvodu jeho nevyužitia pri programovaní. Komunikáciu s displejom zabezpečuje programová knižnica integrovaná vo vývojom prostredí MikroC a bude popísaná v tretej kapitole.

2.4 Ovládanie

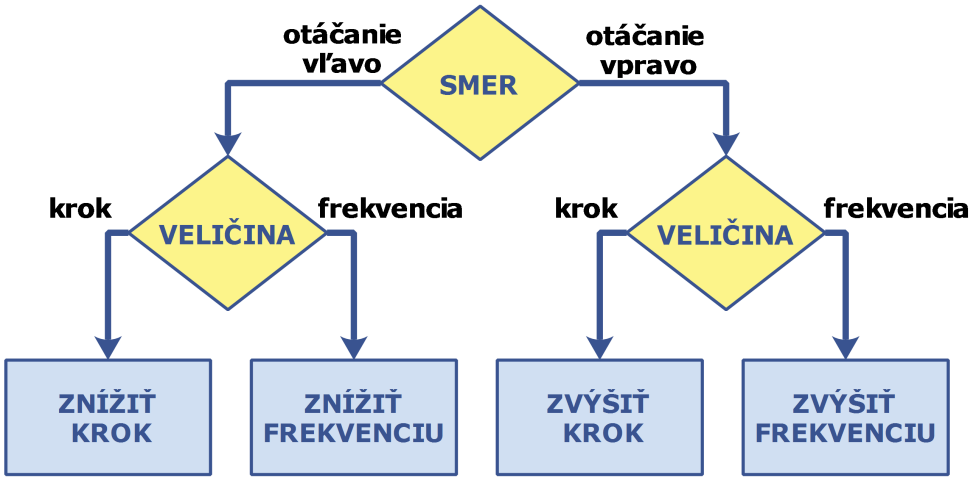
Ovládanie syntetizátora je v princípe možné riešiť mechanickými ovládacími prvkami, ako sú tlačítka, potenciometre, otočné prepínače, rotačné kódery, maticové klávesnice, atď., alebo pomocou externých riadiacich zariadení, napríklad počítača či smartfónu. Externé riadenie a komunikácia budú náplňou podkapitoly 2.5.

Návrh mechanických ovládacích prvkov bol zameraný na ich minimalizáciu, pri zachovaní potrebných požiadaviek na ovládanie, ktorými sú zvyšovanie a znižovanie frekvencie o nastavený ladiaci krok a samotné nastavovanie tohto kroku. Prehľad požadovaných nastavení je v nasledujúcej tabuľke:

č.	Funkcia
1	zvýšiť frekvenciu o ladiaci krok
2	znížiť frekvenciu o ladiaci krok
3	zvýšiť ladiaci krok
4	znížiť ladiaci krok

Tab. 2.2: Požadované funkcie ovládacích prvkov

Pri takto definovaných požiadavkách, je k ovládaniu ideálne použiť jeden rotačný kóder so zabudovaným tlačítkom. Rotácia kódera predstavuje zvyšovanie alebo znižovanie, kým stlačenie tlačítka prepína medzi zvyšovanou/znižovanou veličinou, teda frekvenciou a krokom. Navrhnutý princíp lepšie znázorňuje nasledujúci obrázok:



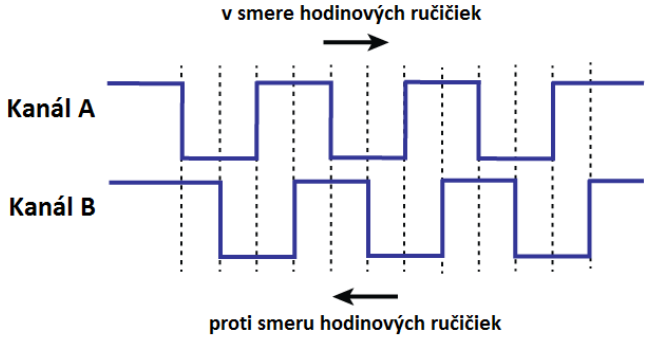
Obr. 2.6: Princíp funkcie rotačného kódera

2.4.1 Rotačný kóder

Rotačný kóder je elektro-mechanická súčiastka, slúžiaca k prevodu pohybu hriadeľa na digitálny kód. Svoje využitie nachádza v mnohých zariadeniach, predovšetkým ako digitálna náhrada potenciometru, s tým rozdielom, že je s ním možné otáčať donekonečna a to v ktoromkoľvek smere. Rotačné kódery sa podľa fyzikálneho princípu delia na typy mechanické, magnetické alebo optické. Z hľadiska funkcie sa rozdeľujú do dvoch kategórií: absolútne a inkrementačné. Z výstupu absolútneho kódera je možné zistiť presnú polohu hriadeľa v ľubovoľnom čase, zatiaľčo výstup inkrementačného kódera poskytuje informáciu len o pohybe samotnom a vyžaduje externé spracovanie. Napriek tomu je inkrementačný kóder najrozšírenejší zo všetkých rotačných kóderov, a to vzhľadom k svojej nízkej cene a schopnosti poskytovať ľahko interpretovateľný signál. preto sa bude ďalší popis zaoberať len ním.

2.4.1.1 Inkrementačný rotačný kóder

Inkrementačné kódery generujú dva výstupné signály, pomenované kanál A a kanál B, pričom sú vzájomne fázovo posunuté o 90 stupňov. Výstupná sekvencia stavov týchto kanálov určuje smer otáčania hriadeľa.



Obr. 2.7: Hodnoty kanálov A a B počas otáčania hriadeľom

Prvou možnosťou vyhodnocovania smeru otáčania je predstaviť si oba kanály ako tvoria dvoj-bitové číslo a toto číslo čítať. Napríklad pri otáčaní v smere hodinových ručičiek sa cyklicky striedajú hodnoty 2-0-1-3 atď. Pri opačnom smere otáčania sa tieto hodnoty menia v opačnom poradí, t.j. 3-1-0-2 atď.

hodnota	Kanál A	Kanál B
2	0	1
0	0	0
1	1	0
3	1	1

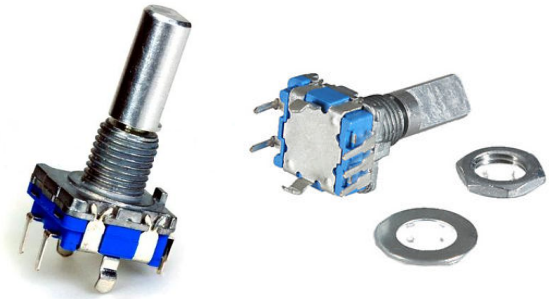
hodnota	Kanál A	Kanál B
3	1	1
1	1	0
0	0	0
2	0	1

Tab. 2.3: Postupnosť hodnôt inkrmentačeho kódera. Vľavo pre smer otáčania v smere hodinových ručičiek, v pravo proti smeru.

Druhá metóda je založená na detekcii nábežnej/zbežnej hrany jedného z výstupných kanálov a kontrola aktuálneho stavu druhého kanálu pri tejto hrane. Nevýhodou tejto metódy je, že zníži rozlíšenie kódera na polovicu, lebo v skutočnosti je detekovaná iba každá druhá hrana signálu. Táto nevýhoda môže byť odstránená detekovaním aj opačnej hrany signálu, vyžaduje však využitie dvoch prerušení mikroprocesora, čo môže byť v niektorých prípadoch problémom. Obyčajne však nie je dôležité vysoké rozlíšenie kódera, a preto nie je nutné túto nevýhodu odstraňovať.

2.4.1.1 Zvolený rotačný kóder

Mnou zvolený rotačný kóder EC-11 je inkrementačný mechanického typu so vstavaným tlačítkom a je vyobrazený na obrázku 2.8.

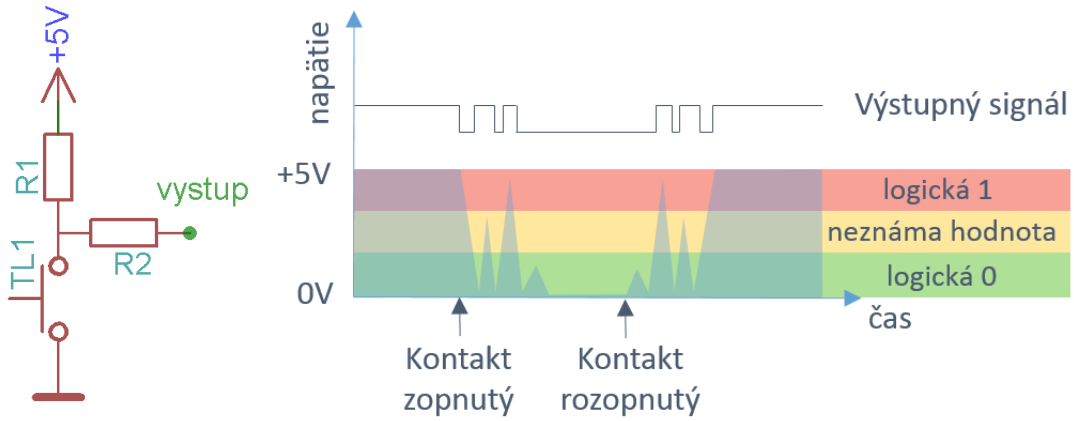


Obr. 2.8: Mechanický inkrementačný rotačný kóder EC-11

Tri piny na jednej strane slúžia ako dátové pre rotačný kóder. Stredný pin je spoločný a je pripojený na zem. Pravý a ľavý pin sú kanály A a B. V zásade je jedno, ktorý je použitý ako A alebo B, oba sú hardvérovo identické. Dva piny na druhej strane kódera sú kontakty zabudovaného tlačítka. Tlačítko sa zopne po kolmom zatlačením hriadeľa.

2.4.2 Potlačenie zákmitov

Mechanické rotačné kódery obsahujú rozpínateľné vodivé kontakty, ktoré podobne ako pri spínačoch, často generujú rušivé spínaco-rozpínacie prechody, nazývané aj zákmity. Dôvodom sú mechanicko-fyzikálne príčiny, z ktorých najpodstatnejší je fakt, že vodivé kontakty sú pružné a pri ich vzájomnom kontakte môžu niekoľkokrát od seba odskočiť pred tým, ako nastane ich stabilný kontakt. Inou možnou príčinou môžu byt znečistené kontakty spínača či už atmosférickými vplyvmi alebo v dôsledku ich predchádzajúceho opálenia. Priebeh zákmitu je zobrazený na obrázku 2.9.



Obr. 2.9: Priebeh nepotlačeného zákmitu

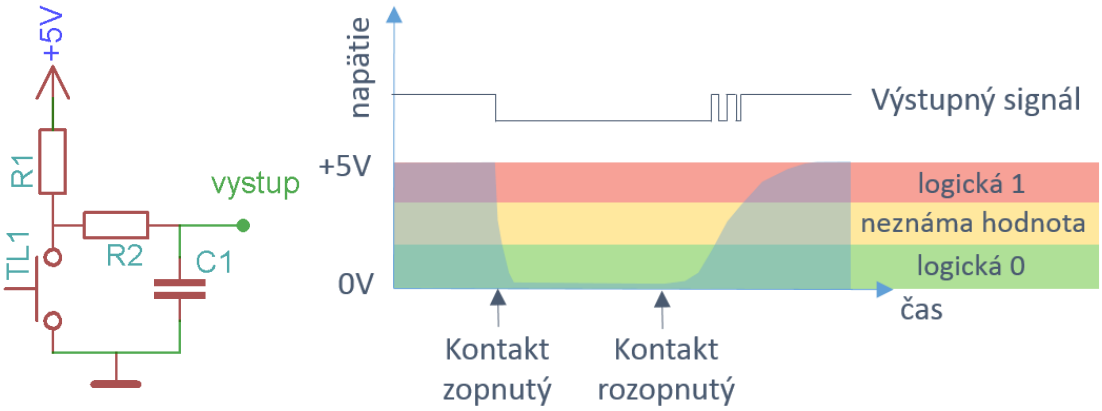
Tieto zákmity môžu byť vyhodnotené ako niekoľkonásobné zopnutie a rozopnutie vo veľmi krátkom čase a môže dochádzať k nesprávnemu vyhodnocovaniu programom. Preto je nevyhnutné zákmity potláčať. Existujú dve bežné metódy ako to dosiahnuť:

1. Úpravou spínacieho obvodu
2. Programovým algoritmom

Obidve sú často používané, a niekedy sú dokonca aj kombinované pre dosiahnutie čo najviac spoľahlivého riešenia.

2.4.2.1 Úprava spínacieho obvodu

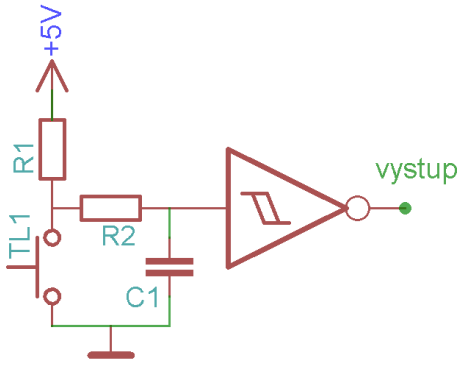
Princíp úpravy spínacieho obvodu je založený na doplnení obvodu kondenzátorom. Kondenzátor bráni rýchlym napäťovým zmenám výstupného signálu. Šírka pulzov, ktoré budú odfiltrované, respektíve prepustené závisí od kapacity kondenzátora C1 a rezistor ov R1 a R2 a to tak, že s narastajúcimi hodnotami C1, R1 a R2 bude spínací obvod viac odolný voči zákmitom, avšak bude narastať čas, potrebný na určenie správneho výstupu [9]. Preto sa ich hodnota volí v závislosti od konkrétnej aplikácie, väčšinou metódou pokus omyl. Pre všeobecné použitie sú vhodné hodnoty napríklad R1=10kΩ, R2=100Ω, C1=100nF [10].



Obr. 2.10: Ošetrenie zákmitu úpravou spínacieho obvodu kondenzátorom

V porovnaní s obvodom bez kondenzátora má výstupné napätie oveľa priaznivejší priebeh. Napriek tomu, je v dobe po rozopnutí kontaktu, hodnota výstupného signálu, relatívne dlhý časový úsek v pásme s neznámou logickou hodnotou (medzi 1,5 až 3,5 V) [11]. Práve kvôli neznámej hodnote môžu aj naďalej vznikať zákmity, hoci už nie také dlhé ako v obvode bez kondenzátora.

Tento posledný nedostatok je riešiteľný ešte zaradením Schmittovho preklápacieho obvodu, ktorý je známy tým, že zmena jeho výstupu pri náraste napätia vstupu nastane pri väčšom napätí než zmena výstupu pri poklese napätia vstupu. Týmto sa spoľahlivo preskočí nutnosť určovania výstupnej hodnoty počas doby trvania zóny s neznámou hodnotou a výstupná hodnota je určovaná až po dosiahnutí bezpečne vyššej alebo nižšej hodnoty.



Obr. 2.11: Ošetrenie zákmitov kondenzátorom a Smittovým preklápacím obvodom

2.4.2.2 Programové riešenie

Existuje viacero programových algoritmov potláčania zákmitov, od úplne jednoduchých až po veľmi sofistikované. Ich hlavnou myšlienkou je skontrolovať vstupnú hodnotu niekoľkokrát pred tým, ako dospejú k jednoznačnému rozhodnutiu, či sa jedná o zmenu stavu tlačidla alebo len zákmit. Líšia sa v dostupných prostriedkoch konkrétneho mikroprocesora a náročnosťou hlavného programu mikroprocesoru.

Najjednoduchšou programovou technikou potláčania zákmitov je čítať stav tlačidla opakovane po nastavenej dobe. Táto doba závisí od konkrétneho ovládacieho prvku. Napríklad u tlačidla musí byť táto doba kratšia ako u prepínača, ale môže byť dlhšia ako u rotačného kódera.

```
if(PORTD.F0 == 0)          //Pokiaľ je tlačidlo stlačené
{
   Delay_ms(50);           //Maximálna doba trvania zákmitov
   if(PORTD.F0 == 0)       //Pokiaľ je tlačidlo stále stlačené
   {
     stlacene = True;      //Vnútorná premená nastavená na True
     Delay_ms(1000);       //Rozostup medzi opetovným stlačením

   }
}
```

Obr. 2.12: Jednoduchý algoritmus na potláčanie zákmitov

Predchádzajúci algoritmus je veľmi jednoduchý a postačuje pri niektorých aplikáciach. Pri časovo-kritických aplikáciach je však nevhodný, kvôli vnoreným pauzám, čím sa celý hlavný program zdržuje a nemôže včas reagovať na vzniknuté situácie.

Iným, dokonalejším algoritmom, vhodným aj pre náročnejšie aplikácie, je kontrolovanie stavu tlačítka v pravidelných intervaloch a určenie, či je nepretržite stlačené počas všetkých týchto intervalov. Za pravidelnú kontrolu stavu tlačidla zodpovedá časovač mikroprocesora, ktorý po nastavenom čase, napríklad 10ms, vždy volá kontrolovaciu funkciu. Počet volaní kontrolovacej funkcie závisí od konkrétneho ovládacieho prvku, napríklad 10 volaní. Kontrolovacia funkcia zistí, či bolo stlačené tlačidlo, a pokiaľ áno zvýši hodnotu čítača o jedna. Pokiaľ je celý čas tlačidlo stlačené vždy rastie hodnota čítača o jedna až do nastavenej hodnoty 10, kedy je možné povedať, že tlačidlo je naozaj celý čas stlačené. V prípade, že aspoň raz dôjde ku rozopnutiu tlačidla, hodnota čítača sa vynuluje. Telo kontrolovacej funkcie je uvedené v nasledujúcom kóde.

```
void KontrolovaciaFunkcia(){  //Metóda volaná každých napr. 10ms
    if(PORTD.F0 == 0)         //Pokiaľ je tlačidlo stlačené
    {
       citac++;               //Zväčši čítač o jedna
       if(citac==10){         //Po 10 cykloch tlačidlo stále stlačené
       stalacene=True;        //Vnútorná premená nastavená na True
       }
    }
    else{                     //Aspoň raz došlo k zmene stavu tlačidla
       citac=0;               //Čitač sa reštartuje
       stlacene=False;        //Vnútorná premená nastavená na False
    }
}
```

Obr. 2.13: Univerzálnejší algoritmus potláčanie zákmitov

2.5 Komunikácia s PC

Návrh spôsobu komunikácie syntetizátora s PC bol realizovaný s ohľadom na možnosť využitia štandardného vybavenia osobných počítačov. Takým vybavením je napríklad sériový port, paralelný port, USB, Bluetooth alebo Ethernet.

Najjednoduchším riešením by bolo využitie sériového portu so štandardom RS-232. Ten umožňuje priamu komunikáciu s UART rozhraním mikroprocesora s nutnosťou len napäťovo prispôsobiť komunikačné signály. UART reprezentuje logickú jednotku ako +5V, logickú nulu ako 0V, zatiaľčo RS-232 má definovanú logickú jednotku ako -12V a logickú nulu ako +12V. Sériové a aj paralelné porty však postupne miznú z ponuky výrobcov v prospech USB a taktiež sa bežne nenachádzajú na prenosných počítačoch. Naopak, najzložitejšie a najdrahšie riešenie je komunikácia cez Ethernet.

Preto bola pre komunikáciu s PC zvolená technológiu USB, ktorá je dostupná takmer na každom PC, nie je drahá a existujú spôsoby ako naň previesť signál UART, s ktorým pracuje mikroprocesor.

Vzhľadom k budúcemu využitiu syntetizátora ako súčasť anténneho analyzátora, kedy je predpoklad, že sa s ním bude pracovať v teréne, bolo ako druhá komunikačná technológia zvolená Bluetooth. S použitím Bluetooth odpadne nutnosť používať prepájací kábel a taktiež je možné nepoužiť k ovládaniu počítač, ale napríklad smartfón. Obidve technológie USB aj Bluetooth sú pre syntetizátor realizované vo forme komunikačného modulu, t.j. samostatnej dosky plošného spoja, pripojiteľnej do základnej dosky s možnosťou ich súčastného použitia.

2.5.1 USB

2.5.1.1 Základná charakteristika

USB - univerzálna sériová zbernica je štandard vyvinutý v 90-tych rokoch, definujúci konektory, káble a komunikačné protokoly. Nahradzuje skôr používané spôsoby pripojenia (sériový port, paralelný port, game port, PS/2 atď) počítača k bežným druhom periférnych zariadení, ako napríklad tlačiarne, klávesnice, myši, modemy, fotoaparáty, prenosné prehrávače, diskové jednotky a ďalšie. V súčastnosti je zo všetkých typov počítačových zberníc rozšírená najviac.

USB architektúra pozostáva z jedného zariadenia typu master, nazývaným HOST a niekoľkých, maximálne 127 periférnych zariadení typu slave. Akúkoľvek komunikáciu začína vždy HOST, a naopak, žiadne zariadenie nemôže vysielať samo o sebe, ale musí čakať na výzvu vysielať svoje dáta [12]. To znamená, že zariadenia nemôžu komunikovať priamo medzi sebou.

Prepájací kábel, dokáže okrem prenosu dát, aj poskytovať napájacie napätie pripojenému zariadeniu. Tým odpadá nutnosť používať samostatný napájací kábel. Niektoré zariadenie dokonca využívajú USB zbernicu len ako zdroj napätia.

USB podporuje 4 dátové rýchlosti:

Režim	Najvyššia rýchlosť	Dostupná od
Low Speed	1.5 Mbit/s	USB 1.0
Full Speed	12 Mbit/s	USB 1.0
High Speed	480 Mbit/s	USB 2.0
SuperSpeed	5 Gbit/s	USB 3.0

Tab. 2.4: Dátové rýchlosti USB [13]

Existujú dva možné spôsoby, ako pripojiť mikroprocesor k počítaču cez USB port:

1. Programová implementácia USB do mikroprocesora.
2. Použitie prevodníka medzi USB a existujúcim komunikačným rozhraním mikroprocesora.

Pre návrh komunikácie syntetizátora s počítačom je využitá druhá možnosť.

5.2.1.2 Obvod FT232R

Jeden z najznámejších prevodníkov medzi UART a USB je obvod FT232RX,pričom posledné X je označenie púzdra. Pre syntetizátor bolo zvolené 28 pinové SSOP púzdro, pri ktorom má obvod označenie FT232RL.

Obr. 2.14: Obvod FT232RL [14]

Základné vlastnosti obvodu FT232R [15] sú:

- Kompletný prevodník medzi UART a USB v jednom čipe, bez nutnosti pripájania ďalších súčiastok,
- USB 2.0 (kompatibilný s USB 1.0),
- možnosť plne hardvérového riadenia prenosu - signály RTS, CTS, DTR, DSR, DCD, RI,
- možnosť práce s 5 V aj 3,3 V logikou,
- možnosť napájať priamo z USB,
- ovládače výrobcu pre všetky bežné OS,
- nízka cena

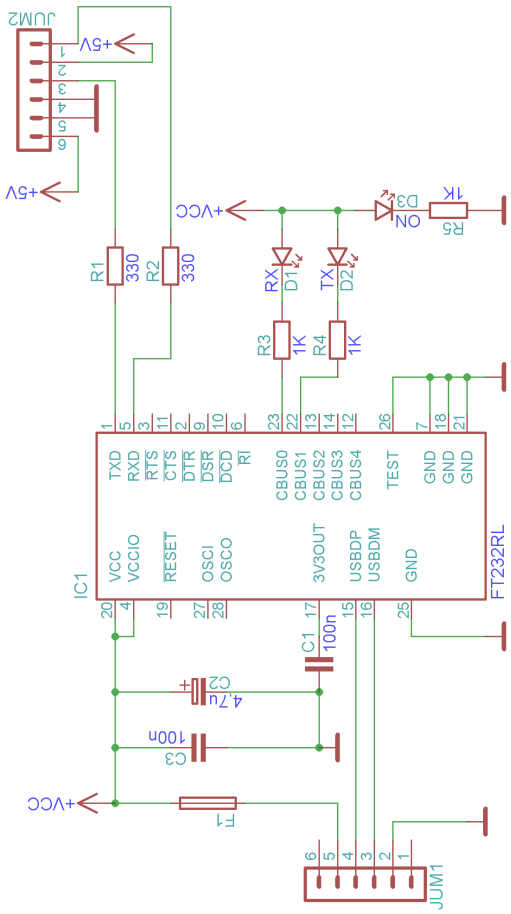
2.5.1.3 Zapojenie USB modulu

Funkčné zapojenie obvodu FT232 bolo realizované formou modulu, z dôvodu možnosti budúcej výmeny tohto modulu iným modulom. Samotné zapojenie USB modulu je vzhľadom ku komplexnosti obvodu FT232R, ktorý už ku svojej funkcii nevyžaduje žiadne ďalšie súčiastky, veľmi jednoduché. Napriek tomu obsahuje tento modul okrem samotného obvodu a pripojiteľných konektorov ku základnej doske aj ďalšie súčiastky. Sú nimi napríklad rezistory, s hodnotou 330Ω, zapojené v každej vysielaco-prijímacej linke na strane UART, z dôvodu ochrany obidvoch komunikujúcich zariadení. Ďalším ochranným prvkom je poistka, ktorá ochráni USB port pred prípadným vnútorným skratom v DDS syntetizátore. Prípadné zákmity napájacieho napätia sú potlačené dvojicou kondenzátorov s hodnotami 4,7 µF a 100 nF, pripojenými paralelne k napájaciemu napätiu. Obvod FT232R umožňuje sledovať komunikačný stav pomocou logických hodnôt na príslušných výstupných pinoch. Pokiaľ sa na tieto piny pripojí LED dióda, môže zobrazovať príslušnú operáciu. LED dióda na pine 22 indikuje prijímané dáta, zatiaľčo LED dióda na pine 23 indikuje odosielané dáta. Posledná LED dióda indikuje prítomnosť napájacieho napätia. Všetky tri ledky majú predradený odpor 1kΩ.

Obvod FT232R je možné zapojiť rôzne pre rôzne aplikácie, preto sú pre účel komunikačného modulu využité iba niektoré jeho piny. Zoznam použitých pinov a ich význam je v nasledujúcej tabuľke.

Pin	Funkcia	Názov	Popis
1	výstup	TXD	Vysielanie UART
4	napájanie	VCCIO	Definovanie nap. hladiny UART +5V
5	vstup	RXD	Príjem UART
7	napájanie	GND	Zem
15	vstup/výstup	USBDP	USB +Dáta
16	vstup/výstup	USBDM	USB -Dáta
17	výstup	3V3OUT	Zdroj 3,3 V
18	napájanie	GND	Zem
20	napájanie	VCC	Napájanie obvodu +5 V
21	napájanie	GND	Zem
22	výstup	CBUS1	LED prijímané dáta
23	výstup	CBUS0	LED vysielané dáta
25	napájanie	GND	Zem
26	vstup	TEST	Pri netestovacom režime uzemnené

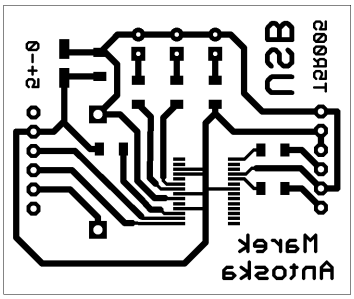
Tab. 2.5: Tabuľka zapojených pinov Bluetooth modulu BTM-222



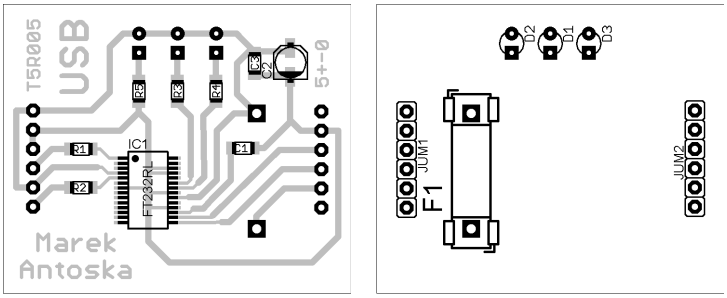
Obr. 2.15: Schéma zapojenia USB modulu

2.5.1.4 Doska plošného spoja USB modulu

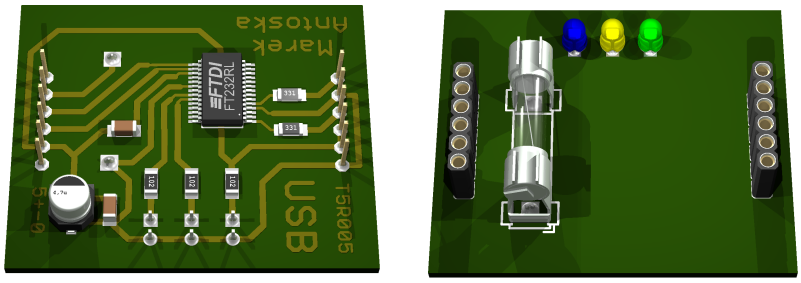
Rozmery dosky sú 45x38 mm. Je určená k samonosnému pripojeniu k základnej doske a tiež umožňuje pripojiť k sebe Bluetooth modul. V blízkosti obvodu sa vodivé plošné spoje zužujú, aby sa prispôsobili veľkosti púzdra obvodu, zatiaľčo všade inde sú širšie z dôvodu menšieho rizika prerušenia spoja pri výrobe dosky. Doska bola navrhnutá, vyrobená, osadená a oživená svojpomocne. Nasledujúce obrázky zachytávajú podobu dosky plošného spoja USB modulu.



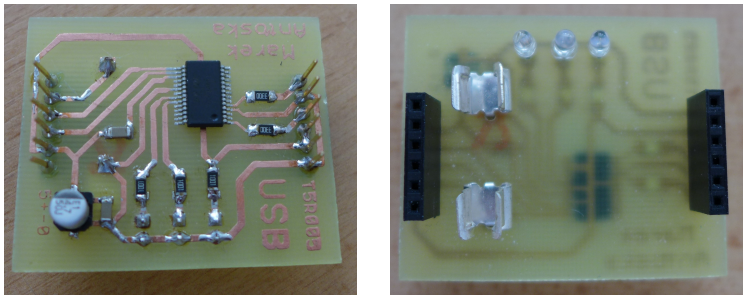
Obr. 2.16: Doska plošných spojov USB modulu.



Obr. 2.17: Rozmiestnenie súčiastok USB modulu.



Obr. 2.18: Vizualizácia USB modulu.



Obr. 2.19: Fotografia USB modulu

2.5.2 Bluetooth

2.5.2.1 Základná charakteristika

Bluetooth je štandardizovaný protokol pre vysielanie a prijímanie dát v pásme 2,4 GHz. Je bezpečný a vhodný pre pre bezdrôtovú komunikáciu medzi zariadeniami na krátke vzdialenosti.

Siete Bluetooth používajú master-slave model, pre určenie kedy má ktoré zariadenie vysielať dáta. V tomto modeli zariadenie typu master môže byť pripojené až ku siedmym rôznym zariadeniam typu slave [16].

Jednotlivé zariadenia majú jedinečnú 48 bitovú adresu, z čoho vrchných 24 bitov označuje výrobcu zariadenia. Adresa je obvykle prezentovaná ako 12 hexadecimálnych číslic. Pre jednoduchšiu identifikáciu zariadenia je ho možné pomenovať vlastným menom, a toto meno je užívateľovi zobrazované namiesto adresy [16].

Vytvorenie spojenia medzi dvomi zariadeniami je viackrokový proces [16]:

1. **Hľadanie** – Pokiaľ Bluetooth zariadenie nevie nič o svojom okolí, pokúša sa vyhľadať ostatné zariadenia. Neustále vysiela požiadavky na spojenie, zatiaľ čo ostatné zariadenia môžu v prípade záujmu na tieto požiadavky zareagovať.
2. **Pripájanie** – Pripájanie je proces tvorby spojenia medzi dvomi zariadeniami. Zariadenie, ktoré zachytilo požiadavku na spojenie, zašle tomuto zariadeniu svoju adresu, poprípade svoje meno.
3. **Prepojenie** – Po úspešnej výmene adries možno obe zariadenia považovať za prepojené. Počas trvania prepojenia môže zariadenia medzi sebou komunikovať, alebo počas nečinnosti môžu prejsť do režimu spánku.

Samotné prepojenie, kedy už dokážu zariadenia spolu komunikovať, ešte nedovoľuje vzájomnú výmenu dát. Tá je možná až po spárovaní zariadení. Párovanie je jednorázový prihlasovací proces a zvyčajne ho musí potvrdiť užívateľ. Spárované zariadenia zdieľajú svoje adresy, mená, profily a tajný kľúč, ktorý im v budúcnosti umožní spárovať sa bez potvrdenia užívateľom [16].

Spárované zariadenia si už môžu medzi sebou vymieňať dáta. Spôsob ich výmeny je určený použitým Bluetooth profilom, čo je dodatočný protokol, ktorý jasne definuje aké dáta sú vysielané. Napríklad tlačiarne využívajú profil BPP, zatiaľ čo hands-free zariadenia využívajú profil HFP. Aby boli obidve komunikujúce zariadenia kompatibilné, musia obe podporovať rovnaký profil [17].

Pri dátovej komunikácii, nahradzujúcej sériovú linku je najlepšie využiteľný profil SPP (Serial Port Profile). SPP profil je najzákladnejším Bluetooth profilom a umožňuje zariadeniam medzi sebou komunikovať ako keby boli ich sériové porty fyzický prepojené [17].

Výkon Bluetooth zariadenia je určený jeho triedou. Existujú tri triedy: Class 1, Class 2 a Class 3. Tabuľka č. 2.6 porovnáva výkony a teoretické maximálne dosahy (v nezastavanom priestore) [18].

Trieda	Maximálny výkon [12]	Maximálny výkon [dBm]	Maximálny dosah
Class 1	100 mW	20 dBm	100 metrov
Class 2	2,5 mW	4 dBm	10 metrov
Class 3	1 mW	0 dBm	1 meter

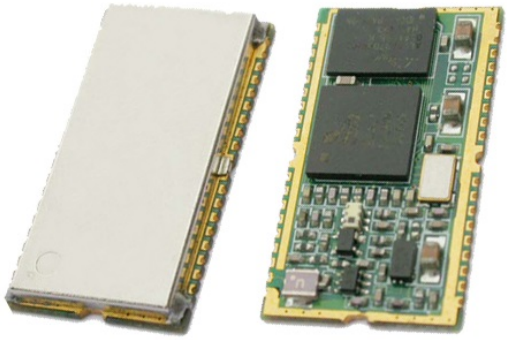
Tab. 2.6: Prehľad výkonových tried Bluetooth zariadení

2.5.2.2 Bluetooth modul BTM-222

Na trhu existuje veľké množstvo Bluetooth modulov podporujúcih profil SPP, líšiacich sa predovšetkým v ich cene a maximálnom dosahu. Medzi najznámejšie patria:

- HC-05
- HC-06 RS232 TTL
- BlueSMiRF
- JY-MCU
- Bluefruit EZ-Link
- BTM-222

Mojim kritériom výberu bola cena a lokálna dostupnosť. Preto som pre komunikáciu syntetizátora s okolím zvolil modul BTM-222, ktorého cena je približne 13 EUR. Je zobrazený na obrázku 2.19, kde na pravej strane je možné vidieť jeho vnútornú štruktúru, zatiaľčo na ľavej strane je zobrazený s tieniacim krytom a takto je aj predávaný.



Obr. 2.20: Bluetooth modul BTM-222 [19]

Základné parametre [20]:

- Výstupný výkon: 100mW (Class 1)
- Napájanie: 3,0-3,6V
- Nízka spotreba
- Rozhranie: USB, UART, SPI, PCM
- Rozmery: 28,2 x 15,0 x 2,8 mm
- Konfigurácia: AT príkazy
- Cena: 13 EUR

2.5.2.3 Zapojenie Bluetooth modulu

Rovnako ako v prípade USB modulu, je aj Bluetooth modul zapojený na samostatnej doske plošných spojov (module). Zvolený Bluetooth modul BTM-222 pracuje na napäťovej hladine 3,3 volta, pričom všetky ostatné obvody syntetizátora pracujú s 5 voltovou napäťovou hladinou. Aby nedošlo k jeho poškodeniu, bolo nutné vymyslieť ako tieto napäťové hladiny medzi sebou prevádzať. Riešenie pozostávalo z dvoch krokov:

1. Zaistiť Bluetooth modulu správne napájacie napätie. To sa podarilo použitím napäťového stabilizátora LF33, ktorého výstupné napätie je 3,3 V.

2. Zaistiť správny napäťový prevod prijímaného signálu. Bluetooth modul priíma po sériovej linke signál, ktorého logická jednotka predstavuje 3,3 V. Keďže bude použitý v zapojení, kde tento signál bude vysielať mikroprocesor, ktorého logická jednotka je reprezentovaná ako 5 V, nie je možné tieto signály prepojiť priamo ako tomu bolo v prípade zapojenia obvodu FT232, ale je potrebný jeho prevod. Ten sa docielil použitím dvojice tranzistorov BC847 v darlingtonovom zapojení, kde nižšie napätie je spínané vyšším.

3. Zaistiť správny napäťový prevod vysielaného signálu. Na prvý pohľad táto podmienka nie je nevyhnutná, lebo mikroprocesor by dokázal prijímať aj signál s nižším napätím. Problém by však mohol nastať počas programovania procesora, kedy sa nedá jednoznačne určiť jeho správanie a mohol by napríklad na malý moment nastaviť jeho prijímací pin na výstupný a pustiť doň 5V, čo by nebolo pre Bluetooth modul vhodné. Preto je použitý úplné rovnaký spôsob prevodu napätia ako v predchádzajúcom prípade, no tentokrát je vyššie napätie spínané nižším.

Bluetooth modul BTM-222 umožňuje sledovať svoj aktuálny stav pomocou logických signálov na jeho troch určených pinoch. Týmito pinmi sú PIO(5), PIO(6) a PIO(7) a dá sa ku ním napríklad cez vhodný rezistor pripojiť LED dióda. V nasledujúcej tabuľke je popis farieb a významu jednotlivých indikačných lediek.

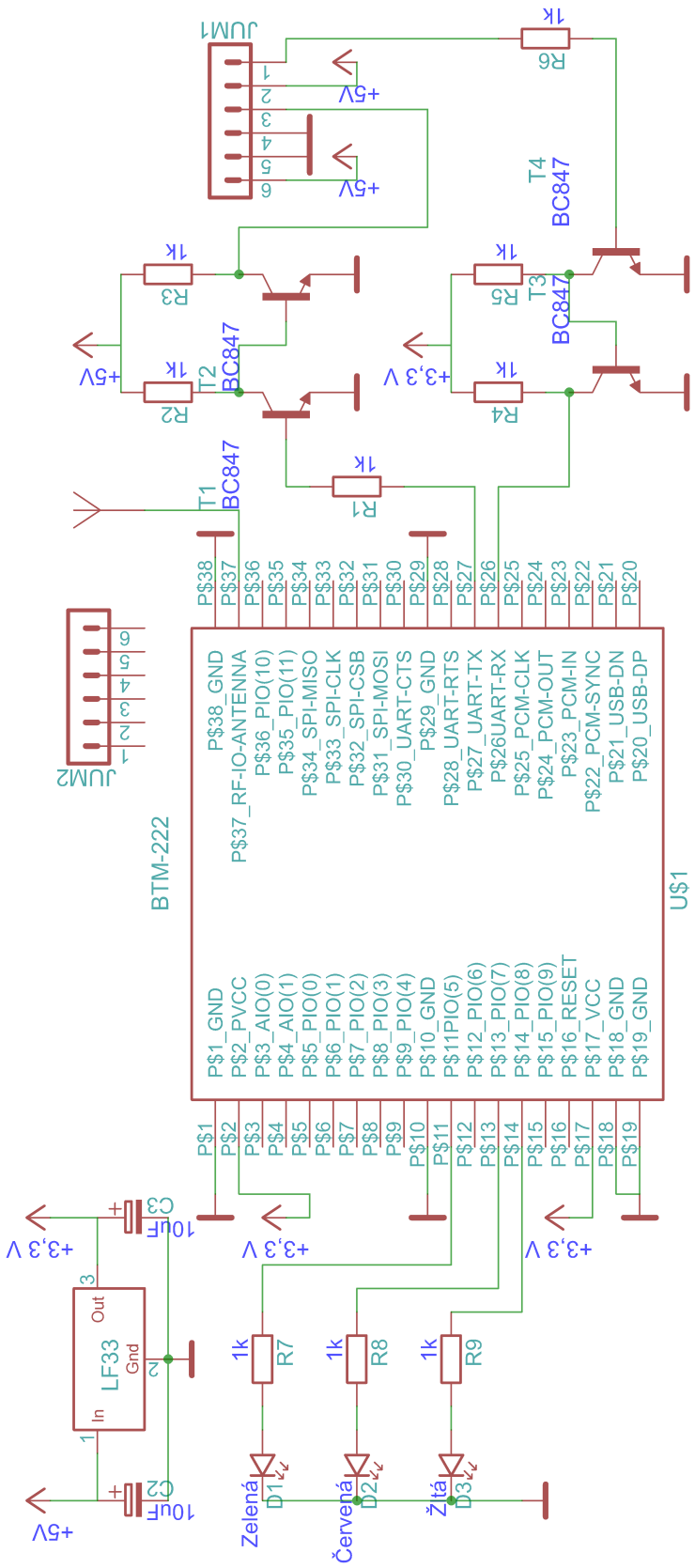
Farba LED	**Pin**	**Význam indikácie**
Zelená	PIO(5)	Komunikácia
Červená	PIO(7)	Pripojenie k inému zariadenie
Žltá	PIO(8)	Správna funkcia Bluetooth modulu

Tab. 2.7: Tabuľka indikácie stavu Bluetooth modulu BTM-222 pomocou LED diód

Vzhľadom k univerzálnemu využitiu modulu BTM-222 sú zapojené iba niektoré jeho piny. Zoznam použitých pinov a ich význam je v nasledujúcej tabuľke.

Pin	Funkcia	Názov	Popis
1	napájanie	GND	Zem
2	napájanie	PVCC	Napájanie koncového stupňa 3,3 V
10	napájanie	GND	Zem
11	výstup	PIO(5)	Zelená LED
13	výstup	PIO(7)	Červená LED
14	výstup	PIO(8)	Žltá LED
17	napájanie	VCC	Napájanie logiky obvodu 3,3V
18	napájanie	GND	Zem
19	napájanie	GND	Zem
26	vstup	UART-RX	Vstup UART 3,3V
27	výstup	UART-TX	Výstup UART 3,3V
29	napájanie	GND	Zem
37	anténa	RF-ANT	Anténa
38	napájanie	GND	Zem

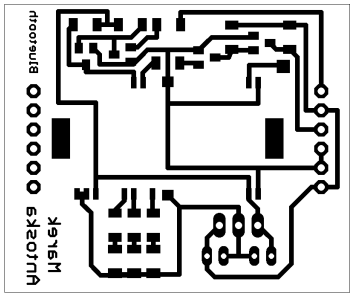
Tab. 2.8: Tabuľka zapojených pinov Bluetooth modulu BTM-222



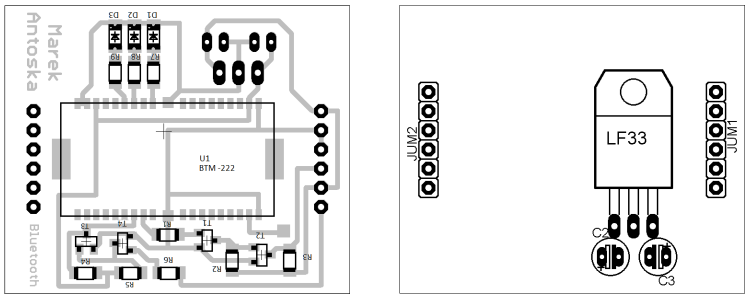
Obr. 2.21: Schéma zapojenia Bluetooth modulu BTM-222

2.5.2.4 Doska plošného spoja BT dosky

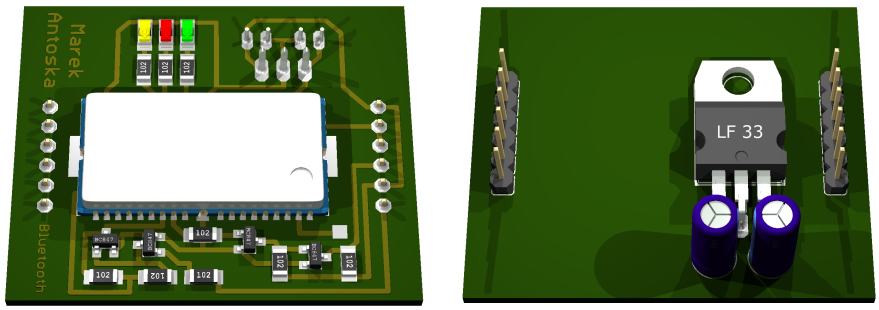
Rozmery dosky sú identické s USB modulom, t.j. 45x38 mm. a je možné ju zapájať priamo do základnej dosky alebo do USB modulu. Doska bola navrhnutá, vyrobené, osadená a oživená svojpomocne. Nasledujúce obrázky zachytávajú podobu dosky plošného spoja USB dosky.



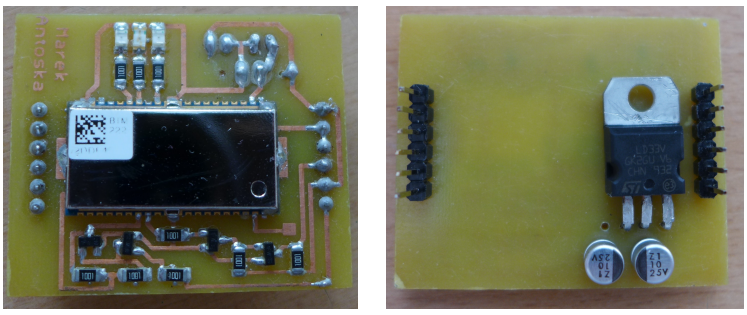
Obr. 2.22: Doska plošných spojov BT dosky



Obr. 2.23: Rozmiestnenie súčiastok BT dosky



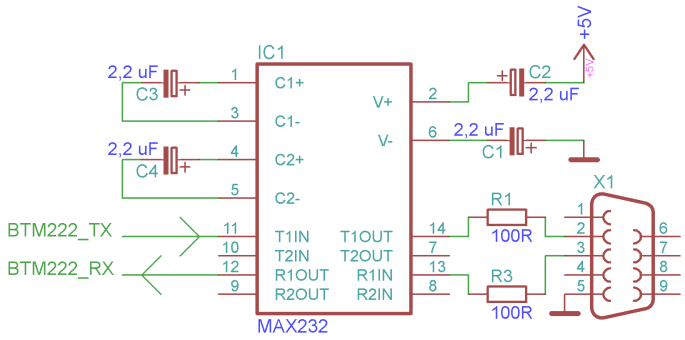
Obr. 2.24: Vizualizácia BT dosky



Obr. 2.25: Fotografia BT dosky

2.5.2.5 Konfigurácia Bluetooth modulu

Bluetooth modul je potrebné pred prvým použitím nakonfigurovať podľa požiadavok konkrétnej aplikácie. Konfigurácia sa vykonáva cez UART rozhranie modulu pomocou AT príkazov . K tomuto účelu je najvhodnejšie pripojiť Bluetooth modul k počítačovému sériovému portu cez príslušný prevodník, napríklad MAX232, a následne zasielať AT príkazy cez terminál z prostredia operačného systému. Jedno z možných prepojení modulu s počítačom, aké bolo použité pri konfigurácii modulu BTM-222, je na obrázku 2.24.



Obr. 2.26: Prepojenie BT modulu so sériovým portom počítača

Parametre sériovej komunikácie pri konfigurácii sú továrensky prednastavené na nasledujúce hodnoty hodnoty:

- **Baud rate:** 19 200 bps
- **Dátových bitov:** 8
- **Parita:** žiadna
- **Stop bit:** 1
- **Riadenie toku:** žiadne

AT príkazy sú textové reťazce začínajúce vždy písmenami AT, nasledované ďalším písmenom, predstavujúcim konkrétnu nastavovanú vlastnosť. Presný zoznam dostupných nastavení je uvedený v datasheete modulu.

Pri konfigurácii Bluetooth modulu boli použité tieto AT príkazy:

- **ATN** - Umožňuje pomenovať zariadenie ľahkozapametateľným názvom, ktorý sa bude zobrazovať v zozname dostupných zariadení. Povolené znaky sú A-Z, a-z, 0-9, medzera a pomlčka,pričom maximálny počet znakov je 16. Príkaz sa zadáva vo formáte ATN=xxxxxxxxxxxxxxxx

- **ATP** - Určuje PIN kód zariadenia. PIN kód možno nastaviť(ATP=xxxx), prečítať(ATP?) alebo zakázať (ATP0)

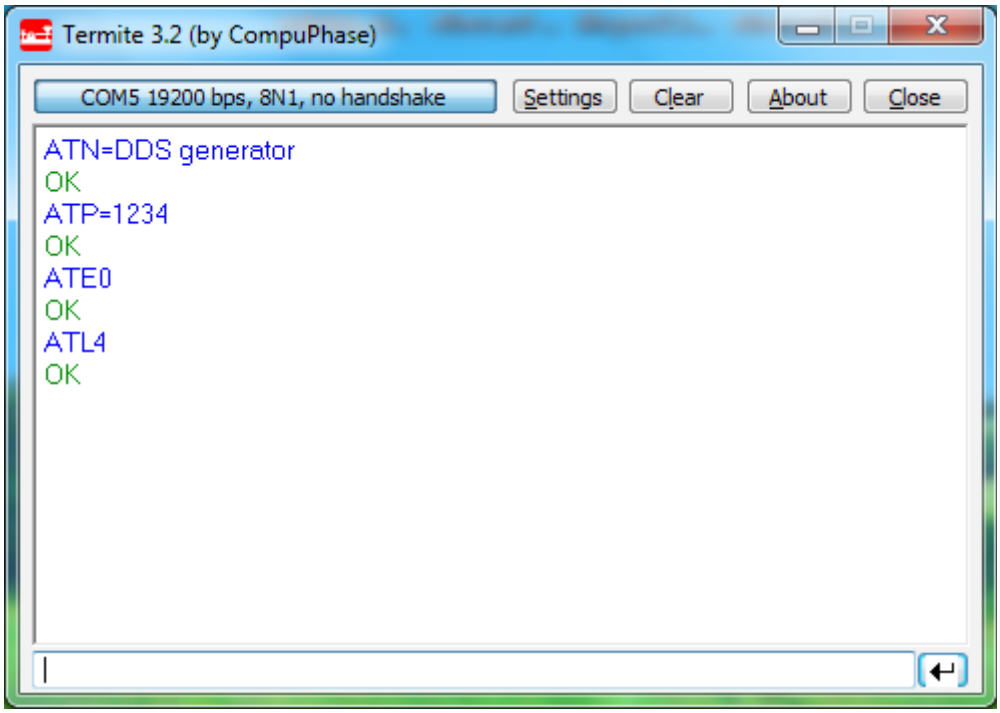
- **ATE** - Špecifikuje, či zariadenie bude na strane UART preposielať komunikáciu z prijímacej linky na vysielaciu(ATE1), v opačnom prípade (ATE0). Je možné zistiť aktuálne nastavenie(ATE?)

- **ATL** - Špecifikuje dátovú rýchlosť. Napríklad ATL2 nastavuje rýchlosť 19 200 Bd/s, ATL4 nastavuje rýchlosť 57 600 Bd/s a ATL? vracia aktuálnu nastavenú rýchlosť. Ostatné hodnoty príkazu ATL sú uvedené v datasheete [20].

  Hodnoty, ktoré boli prenastavené:

- ATN=DDS generator
- ATP=1234
- ATE0
- ATL4

Ďalšie možnosti nastavovania sú napríklad spôsob koumukácie (počet dátových bitov, parita, počet stop-bitov, riadenie toku), rola (master/slave), viditeľnosť, atď. Tieto nastavanie však neboli na modul aplikované, z dôvodu ich vhodného továrneho nastavenia.



Obr. 2.27: Záznam komunikácie cez terminál

2.6. Riadiaci mikroprocesor

2.6.1 Požiadavky na mikroprocesor

Po dôkladnom výbere všetkých súčasti DDS syntetizátora, už bolo jasné, aké požiadavky budú kladené na výber použitého mikroprocesora. Požiadavky jednotlivých súčastí sú zhrnuté v nasledujúcej tabuľke:

DDS obvod	• 4 výstupné porty • 5 V
Ovládanie	• 1 externé prerušenie • 2 vstupné porty
LCD displej	• 6 výstupných portov • 5 V
Komunikačné moduly	• 1 UART • 5 V alebo 3,3 V

Tab. 2.9: Požiadavky zvolených súčastí na použitý mikroprocesor

Zvolený mikroprocesor preto musel disponovať minimálne desiatimi výstupnými portami, jedným externým prerušením, dvomi vstupnými portami a jedným sériovým portom UART. Pracovné napätia procesora bolo možné voliť 5 V alebo 3,3 V. Vzhľadom k tomu, že všetky navrhnuté obvody, okrem Bluetooth modulu, pracujú s 5 V, bolo najvýhodnejšie voliť ako pracovné napätie procesora tiež 5V, aby nebolo nutné všetky tieto obvody vybavovať napäťovými prevodníkmi, a tak musí mať napäťový prevodník iba Bluetooth modul.

Ďalším kritériom výberu bola možnosť programovať mikroprocesor už zapojený v obvode, bez nutnosti jeho prekladania do programátora pred každým programovaním . K tomu slúži technológia ICSP (In-Circuit Serial Programming) a disponujú ňou kvalitnejšie mikroprocesory.

Taktiež dôležitým parametrom je veľkosť mikroprocesorovej pamäte. Je ťažké dopredu odhadnúť jej potrebnú veľkosť, najmä pri programovaní v jazyku C, kedy je jednoduchý príkaz preložený prekladačom ako veľký ťažkoodhadnuteľný počet inštrukcií, častokrát väčší, ako je v skutočnosti nutné, preto bol volený model procesora s relatívne veľkou pamäťou v porovnaní s ostatnými modelmi, konkrétne 32 kB.

Všetky uvedené požiadavky splňovalo viacero mikroprocesorov rôznych značiek, (PIC, ATMEL, STM), preto boli rozhodujúce faktory výberu lokálna dostupnosť a nízka cena.

Z dostupných možností bol zvolený mikroprocesor PIC18F252, ktorého vlastnosti sú popísané v nasledujúcej časti práce.

2.6.2 PIC18F252

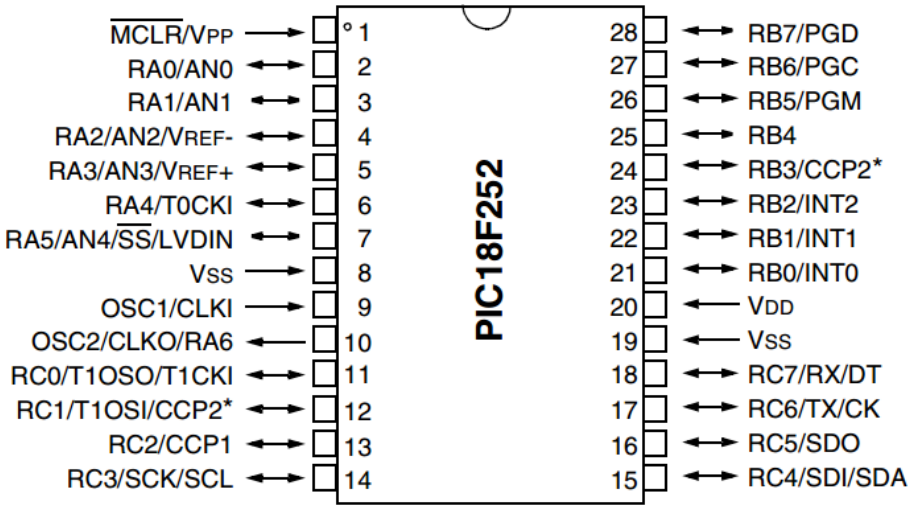
PIC18F252 je univerzálny 8-bitový jednočipový mikrokontrolér s rýchlosťou 10 MIPS (10 miliónov inštrukcii za sekundu) nachádzajúci sa v 28-pinovom púzdre DIP (pre montáž do dier) alebo SOIC (pre povrchovú montáž) [21]. Pre aplikáciu syntetizátora bol zvolený typ v prevedení DIP.

V nasledujúcej tabuľke sú uvedené základné vlastnosti tohto mikroprocesora [21].

Rýchlosť CPU	• 10 MIPS
Typ programovej pamäte	• FLASH
Veľkosť programovej pamäte	• 32 kB
Veľkosť RAM	• 1536 B
Veľkosť EEPROM	• 256 B
Komunikačné rozhrania	• 1 x UART • 1 x USART • 1 x SPI • 1 x I2C
Časovače	• 1 x 8 bit • 3 x 16 bit
AD prevodníky	• 5 kanálov x 10 bitov •
Rozsah napájacieho napätia	• 2-5,5 V

Tab. 2.10: Základné vlastnosti mikroprocesora PIC18F252

U mikroprocesorov je bežné, že jeden pin môže mať viac funkcii. Tak je tomu aj v prípade procesora PIC18F252. Prehľad funkcii jednotlivých pinov je na obrázku 2.26.



Obr. 2.28: Popis pinov PIC18F252 [21]

2.7 Konštrukcia základnej dosky

Základná doska navrhnutého DDS syntetizátora prepája jednotlivé funkčné súčastí s riadiacim procesorom. Pevnou súčasťou základnej dosky je riadiaci mikroprocesor so zdrojom taktovacích impulzov, obvody potlačenia zákmitov mechanického rotačného kódera s tlačítkom a stabilizovaný zdroj napätia +5V. Všetky ostatné obvody sa k základnej doske pripájajú pomocou konektorov. Konkrétne ide o:

1. konektory na zapojenie DDS obvodu s AD9851,
2. konektory na zapojenie komunikačného modulu (USB, Bluetooth, alebo obidva súčasne),
3. konektory na vyvedenie sínusového signálu z DDS obvodu,
4. konektory na pripojenie LCD displeja,
5. konektor na prepojenie rotačného kódera s tlačítkom,
6. konektor určený k programovaniu mikroprocesora PIC,
7. konektor na pripojenie USB konektoru,
8. konektor s nevyužitými portami mikroprocesora pre prípadné budúce využitie,
9. konektory pre prípadné budúce zapojenie nového modulu.

2.7.1 Schéma zapojenia

Jadrom základnej dosky je mikroprocesor PIC18F252, pripojený ku kladnému napájaciemu napätiu +5V pinom číslo 20 a k zemi pinmi 8 a 19.

Zdroj taktovacích impulzov predstavuje kryštáľ s hodnotou 14 MHz pripojený k vývodom 11 a 12 a podľa doporučenia výrobcu je každý jeho vývod premostený so zemou kondenzátorom s hodnotou 22pF.

Procesor disponuje technológiou ICSP, umožňujúcou programovať procesor osadený v obvode. Pre možnosť využitia tejto technológie, je základná doska vybavená konektorom JUM1, cez ktorý je možné ku nej priamo pripojiť programátor, pričom rozloženie signálov v tomto konektore presne odpovedá rozloženiu signálov v používanom programátore.

Pomocou konektorov JUM2 a JUM3 sa ku základnej doske pripája LCD displej. JUM2 slúži na dátovu komunikáciu s LCD displejom, zatiaľčo JUM3 na napájanie. Napája sa logická časť, kde je pripojených +5V nepriamo, ďalej je napájané podsvietenie cez odpor 330Ω a nakoniec je potrebné napájať vstup, určujúci kontrast displeja, napätím v rozsahu 0-5 V. Toto napätie sa neurčuje výpočtom, ale sa získava z potenciometra a nastavuje sa až pri zhotovení zariadenia tak, aby aby vyhovoval konkrétnej aplikácii.

Piny mikroprocesora 21, 22 a 23 sú určené k čítaniu stavov rotačného kódera. Z dôvodu ošetrenia zákmitov kódera, je v obvode zaradený pre každý mechanický kontakt jeden keramický kondenzátor s hodnotou 100 nF a Schmittov preklápací obvod. Schmittov preklápací obvod tvorí logický obvod 7414, napájaný cez pin 7 (GND) a 14 (+5 V). Mechanické kontakty kódera sa pripájajú k základnej doske cez konektor JUM4 a spínajú voči zemi.

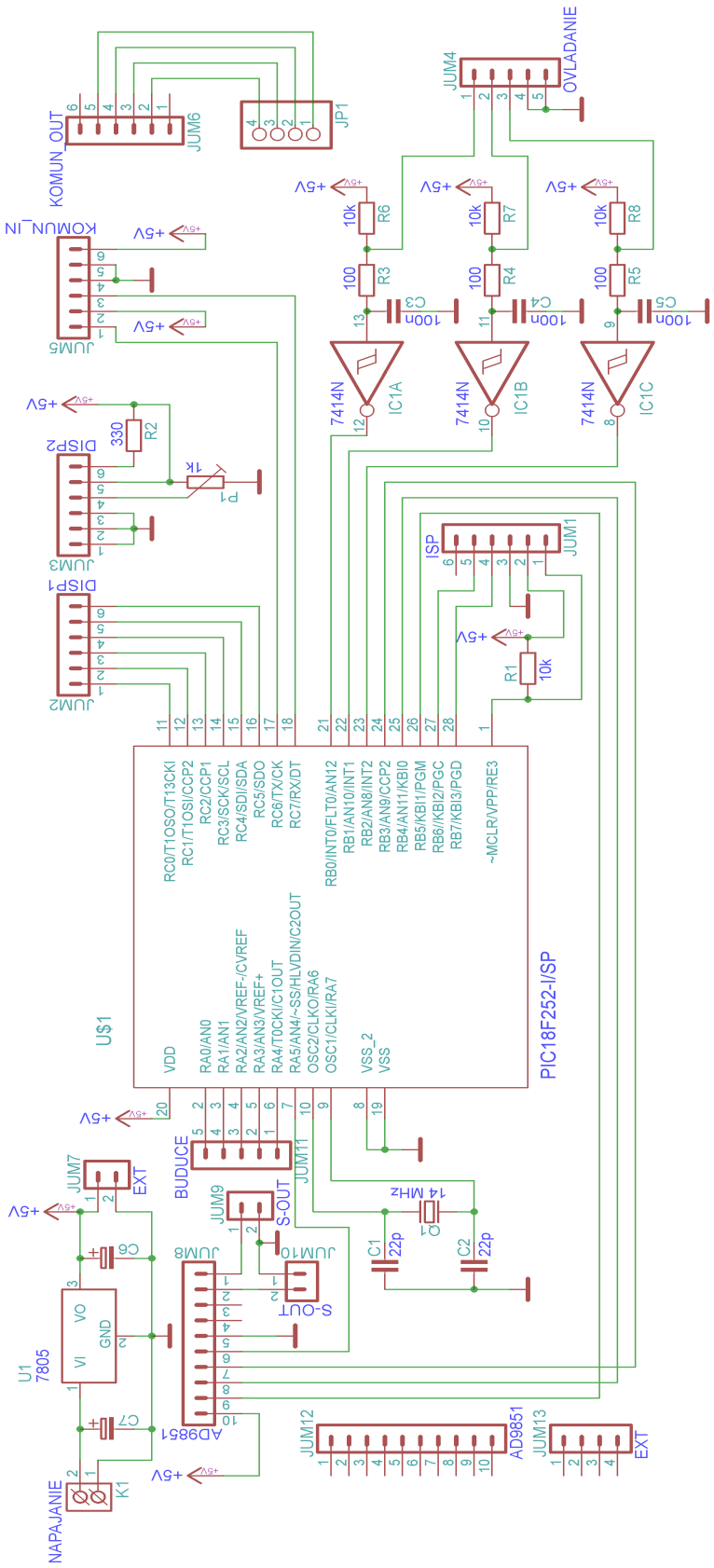
Komunikačné moduly sa k základnej doske pripájajú cez konektory JUM5 a JUM6. Obidva moduly, Bluetooth aj USB, využívajú konektor JUM5 ako vstup a výstup pre UART komunikáciu s mikroprocesorom. Konektor JUM6 sa používa len v prípade USB modulu a to na vyvedenie USB portu na panel syntetizátora pomocou ďalšieho konektoru PJ1.

DDS modul je vybavený dvomi radami konektorov JUM8 a JUM12 vybavenými desiatimi pinmi. K tomu sa museli prispôsobiť konektory na základnej doske. Konektor JUM 8 slúži na napájanie, príjem ladiaceho slova a vyvedenie výstupného sínusového signálu do základnej dosky, odkiaľ je inými konektormi, JUM9 a JUM10, vyvedené na BNC konektory na panely syntetizátora. Konektor JUM12 nie je v tejto aplikácii využitý a má len mechanicko-oporný význam.

Na mikroprocesore zvýšilo 5 nevyužitých pinov, s ich možným budúcim využitím buď ako vstupné, výstupné, alebo analógové a preto boli vyvedené na konektor JUM11

V prípade budúcej potreby, bolo na zvláštny konektor JUM7 vyvedené aj napájanie +5 V a zem, a taktiež bol umiestnený aj konektor JUM13 pre mechanické upevnenie prípadného rozširujúceho modulu. .

Základná doska je napájaná jednosmerným napätím v rozsahu 9 až 12 V cez svorkovnicu K1. Toto napätie je následne upravené stabilizátorom napätia 7805 na hodnotu +5 V. Stabilizátor je doplnený vyhladzovacími kondenzátormi na vstupe aj výstupe podľa doporučenia výrobcu.



Obr. 2.29: Schéma zapojenia základnej dosky

2.7.2 Zapojenie konektorov

Základná doska obsahuje spolu 13 konektorov typu JUMPER. Ich zapojenie je v niektorých prípadoch predurčené zapojením pripájaných modulov, poprípade programátora, ostatné však boli realizované s ohľadom na jednoduchšiu realizáciu dosky plošného spoja (DPS), alebo náhodne.

V nasledujúcich tabuľkách je presné zapojenie a význam jednotlivých kontaktov každého jedného použitého konektora.

JUM1 - Slúži k pripojeniu mikroprocesorového programátora, jeho zapojenie záviselo od použitého programátora, po naprogramovaní sa programátor odpojí a konektor je ostáva nezapojený

Pin	Funkcia	Označenie	Popis
1	vstup	MCLR	Hlavný reset
2	napájanie	+5V	+5 V
3	napájanie	GND	Zem
4	výstup	PGC	Hodinový bit programátora
5	vstup/výstup	PGD	Dátový bit programátora
6	žiadna	NC	Nezapojený

Tab. 2.11: Zapojenie konektora JUM1

JUM2 - Slúži k pripojeniu dátových signálov LCD displeja. Konkrétne rozloženie jednotlivých signálov bolo navrhnuté s ohľadom na jednoduchšiu realizáciu DPS. Prepojenie so samotným displejom je realizované jednožilovými prepojovacími vodičmi.

Pin	Funkcia	Označenie	Popis
1	výstup	D7	Dátový bit LCD displeja
2	výstup	D6	Dátový bit LCD displeja
3	výstup	D5	Dátový bit LCD displeja
4	výstup	D4	Dátový bit LCD displeja
5	výstup	RS	Riadiaci dátový bit REGISTER SELECT
6	výstup	E	Riadiaci dátový bit ENABLE

Tab. 2.12: Zapojenie konektora JUM2

JUM3 - Slúži k pripojeniu napájacích vodičov ku LCD displeju. Pre konkrétne rozloženie a prepojenie s displejom platí to isté čo pre JUM2.

Pin	Funkcia	Označenie	Popis
1	napájanie	GND	Zem
2	napájanie	GND	Zem

3	napájanie	GND	Zem
4	napájanie	+KONTR	Nastavenie kontrastu LCD displeja
5	napájanie	+5V	+5 V
6	výstup	+PODSV	Podsvietenie LCD displeja

Tab. 2.13: Zapojenie konektora JUM3

JUM4 - K tomuto konektoru sa pripájajú kanály A a B rotačného kódera, a vstavané tlačítko kódera

Pin	Funkcia	Označenie	Popis
1	vstup	RKA	Kanál A rotačného kódera
2	vstup	RKB	Kanál B rotačného kódera
3	vstup	RKT	Tlačítko rotačného kódera
4	napájanie	GND	Zem
5	napájanie	GND	Zem

Tab. 2.14: Zapojenie konektora JUM4

JUM5 - Slúži na prepojenie UART portu mikroprocesora s UART portom komunikačných modulov. Je možné k nemu pripojiť samastatne Bluetooth modul, alebo samostatne USB modul, alebo obidba súčasne.

Pin	Funkcia	Označenie	Popis
1	výstup	RX	Prijímací bit komunikačného modulu
2	napájanie	+5V	+5 V
3	vstup	TX	Vysielací bit komunikačného modulu
4	napájanie	GND	Zem
5	napájanie	GND	Zem
6	napájanie	+5V	+5 V

Tab. 2.15: Zapojenie konektora JUM5

JUM6 - Má význam len pri používaní USB modulu, kedy odvádza/privádza USB komunikáciu do USB modulu

Pin	Funkcia	Označenie	Popis
1	žiadna	NC	Nezapojený
2	napájanie	+5V	+5 V
3	vstup/výstup	D-	Data -
4	vstup/výstup	D+	Data +
5	napájanie	GND	Zem
6	žiadna	NC	Nezapojený

Tab. 2.16: Zapojenie konektora JUM6

JUM7 - Je konektor pre prípadné budúce použitie a je naň vyvedené napájacie napätie.

Pin	Funkcia	Označenie	Popis
1	napájanie	+5V	+5 V
2	napájanie	GND	Zem

Tab. 2.17: Zapojenie konektora JUM7

JUM8 - Na tento konektor sa pripája DDS modul. Rozloženie signálov na konektore bolo dané zapojením DDS modulu.

Pin	Funkcia	Označenie	Popis
1	vstup	OUTB	Filtrovaná výstupná frekvencia
2	vstup	OUTA	Nefiltrovaná výstupná frekvencia
3	žiadna	NC	Nezapojený
4	žiadna	NC	Nezapojený
5	napájanie	GND	Zem
6	výstup	RST	Reset
7	výstup	D7	Dátovy bit sériového ladenia
8	výstup	FQUD	Aktualizácia frekvencie
9	výstup	WCLK	Hodinový signál nahrávania bitov
10	napájanie	+5V	+5 V

Tab. 2.18: Zapojenie konektora JUM8

JUM9 - K tomuto konektoru sa pripája výstupný BNC konektor a slúži na vyvedenie signálu prefiltrovaného filtrom na DDS module

Pin	Funkcia	Označenie	Popis
1	výstup	OUTB	Filtrovaná výstupná frekvencia
2	napájanie	GND	Zem

Tab. 2.19: Zapojenie konektora JUM9

JUM10 - Jeho význam je rovnaký ako JUM9, s tým rozdielom, že na BNC konektor vyvádza z DDS modulu nefiltrovaný sínusový signál, ktorý bude možné do budúcna fitrovať prípadným dokonalejším filtrom ako tým, čo je vstavaný v DDS module.

Pin	Funkcia	Označenie	Popis
1	výstup	OUTA	Nefiltrovaná výstupná frekvencia
2	napájanie	GND	Zem

Tab. 2.20: Zapojenie konektora JUM10

JUM11 - Slúži k vyvedeniu nevyužitých pinov mikroprocesora, pre možné budúce využitie.

Pin	Funkcia	Označenie	Popis
1	vstup	RKA	Kanál A rotačného kódera
2	vstup	RKB	Kanál B rotačného kódera
3	vstup	RKT	Tlačítko rotačného kódera
4	napájanie	GND	Zem
5	napájanie	GND	Zem

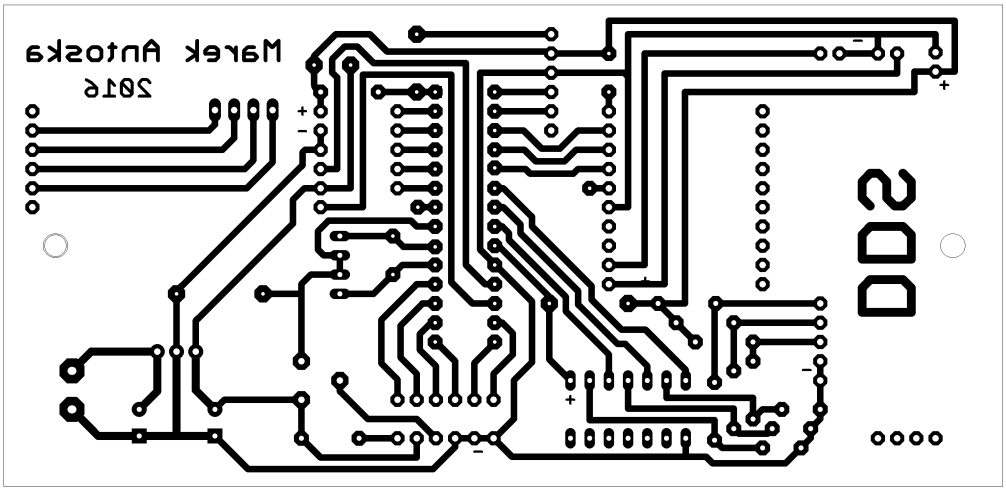
Tab. 2.21: Zapojenie konektora JUM11

JUM12 - Je v prípade sériového ladenie DDS obvodu kompletne nezapojený, slúži len ako mechanické upevnenie DDS modulu

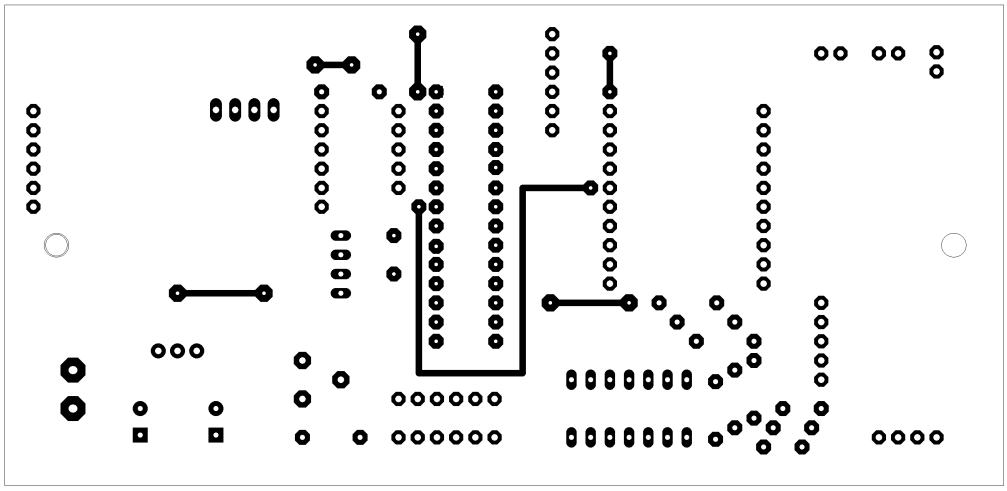
JUM13 - Je určený pri prípadné budúce mechanické upevnenie rozširujúceho modulu.

2.7.3 Doska plošných spojov

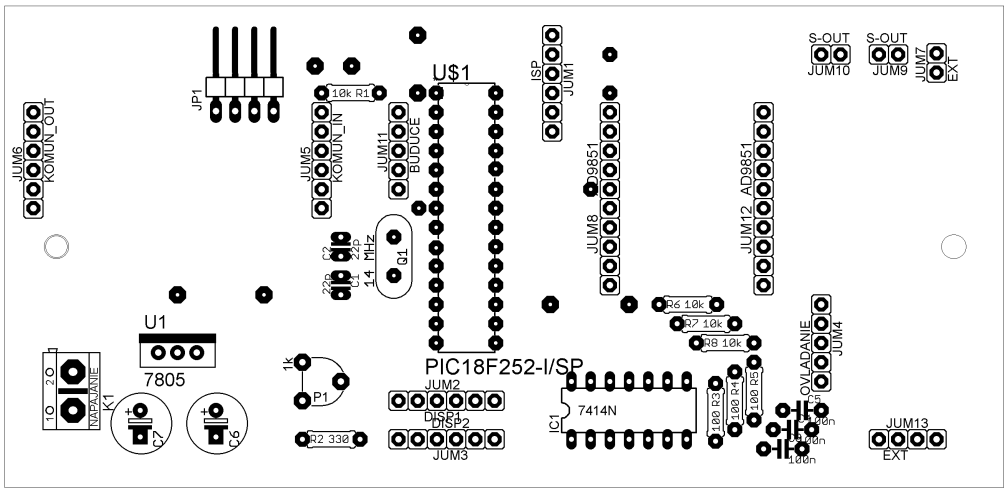
Rozmery dosky sú 132x64 mm a boli volené s ohľadom na rozmery prístrojovej krabičky, v ktorej sa bude doska nachádzať. Na obidvoch koncoch dosky je vyvŕtaná jedna diera pre mechanické pripevnenie dosky o krabičku. Šírka vodivých spojov je 0,8 mm a je jednotná v celej doske. Mikroprocesor a obvod 7414 sú osadené v obvodových päticiach, kvôli bezpečnej montáži obvodov a jednoduchšej prípadnej výmene. Doska bola navrhnutá, vyrobená, osadená a oživená svojpomocne. Nasledujúce obrázky zachytávajú podobu dosky plošného spoja základnej dosky.



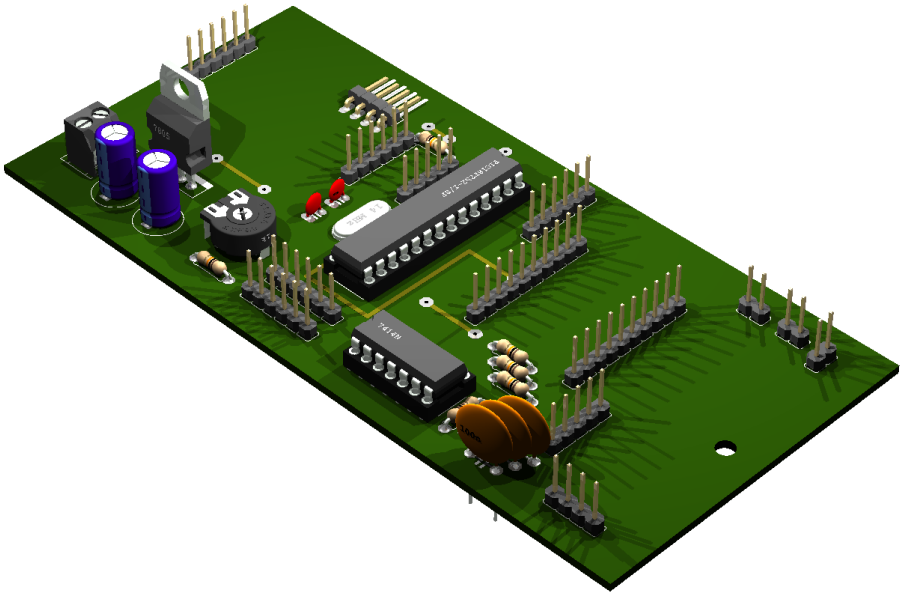
Obr. 2.30: Doska plošných spojov, mierka 1:1 (strana spojov)



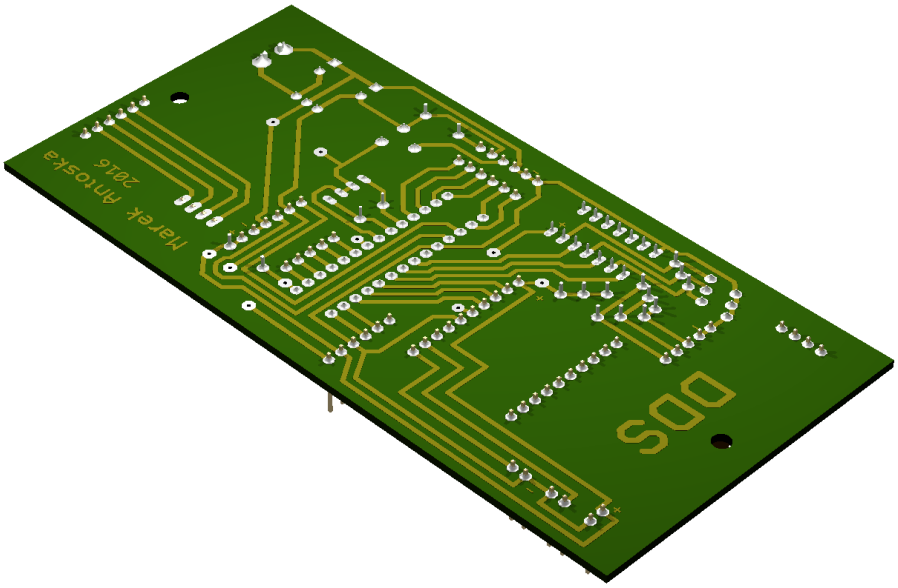
Obr. 2.31: Doska plošných spojov, mierka 1:1 (strana súčiastok)



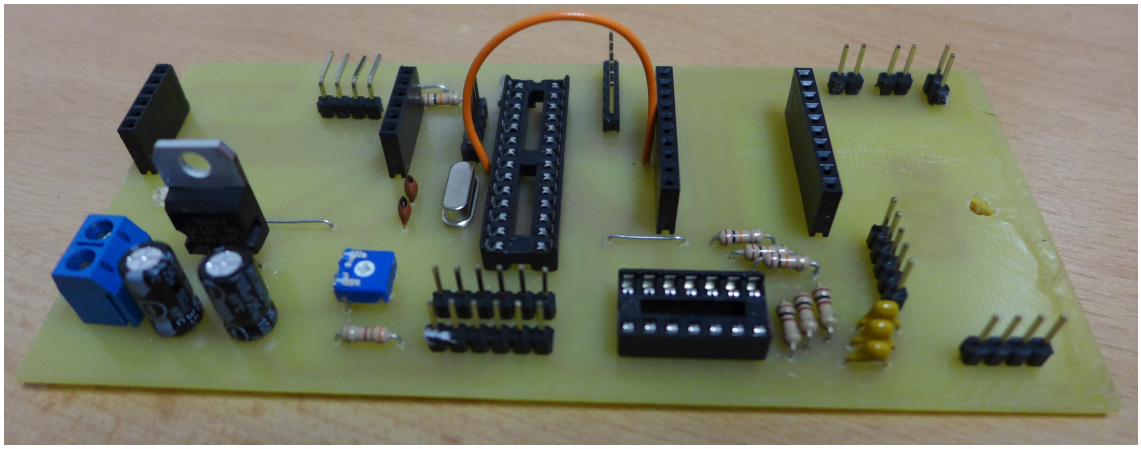
Obr. 2.32: Rozmiestnenie súčiastok,, mierka 1:1



Obr. 2.33: Vizualizácia finálneho vzhľadu (vrch)



Obr. 2.34: Vizualizácia finálneho vzhľadu (spodok)



Obr. 2.35: Fotografia osadenej základnej dosky

3. Riadiaci program mikroprocesora

Programová časť syntetizátora bola vyvíjaná v jazyku C, vo vývojovom prostredí mikroC. Jedná sa o plnohodnotné vývojové prostredie s možnosťou vývoja až pre 7 rôznych mikroprocesorových architektúr: PIC, dsPIC, PIC32, AVR, 8051, ARM, FT900. Poskytuje mnoho hardvérových aj softvérových knižníc, voliteľnú úroveň optimalizácie kódu a prídavné nástroje pre uľahčenie vývoja [22].

Samotný program mikroprocesora (Príloha B) je zložený z dvoch logických častí: z hlavného programu a obsluhy prerušenia.

3.1 Hlavný program

Po pripojení napájania k mikroprocesoru sa začína vykonávať hlavný program, ktorého vývojový diagram je na obrázku 3.1. Hlavný program začína inicializáciou mikroprocesora, kde sú definované funkcie jednotlivých pinov mikroprocesora a nakonfigurované príslušné registre podľa požadovanej funkcie, konkrétne registre nastavujúce vstupnú alebo výstupnú funkciu konkrétnych pinov a registre nastavujúce prerušenie. Hodnoty nastavovaných registrov sú v tabuľkách 3.1 a 3.2.

Register	Názov	Hod.	Funkcia
RCON	IPEN	0	Zakázaná priorita prerušení
INTCON2	INTEDG0	1	Prerušenie INT0 s nábežnou hranou
INTCON	INT0IE	1	Prerušenie od pinu INT0 povolené
INTCON	INT0IF	0	Reset príznaku prerušenia INT0
INTCON	PEIE	1	Externé prerušenia povolené
INTCON	GIE	1	Prerušenia globálne povolené

Tab. 3.1: Nastavenie registrov prerušenia

Register	Názov	Hod.	Funkcia
TRISB	F0	1	pin RB0 je vstup, Rotačný kóder - Kanál A
TRISB	F1	1	pin RB1 je vstup, Rotačný kóder - Kanál B
TRISB	F2	1	pin RB2 je vstup, Rotačný kóder - Tlačítko
TRISB	F3	0	pin RB3 je výstup, DDS_D7
TRISB	F4	0	pin RB4 je výstup, DDS_FQUD
TRISBB	F5	0	pin RB5 je výstup, DDS_WCLK
TRISA	F5	0	pin RA5 je výstup, DDS_RESET

Tab. 3.2: Nastavenie registrov funkcie pinov

Po nastavení registrov dôjde k inicializácii LCD displeja, DDS obvodu a sériového portu.

K ovládaniu LCD displeja je využitá knižnica, ktorú ponúkalo vývojové prostredie. Súčasťou tejto knižnice bol aj inicializačný príkaz, ktorý stačilo pre inicializáciu displeja len zavolať.

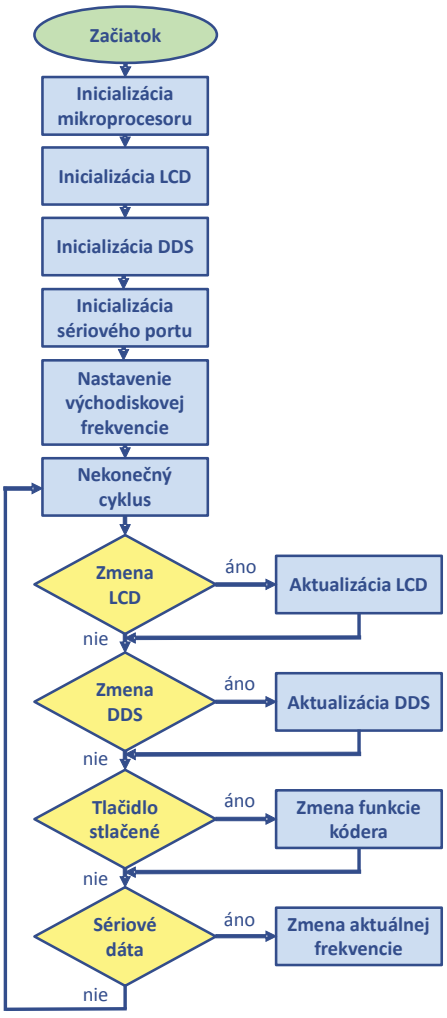
DDS obvod je štandardne nastavený pre paralelný prenos dát. V syntetizátore je však navrhnutá sériová komunikácia, preto je ju potrebné pred použitím nastaviť. Toto nastavenie sa vykonáva logickými impulzmi ovládacích signálov, v tomto poradí RESET, WCLK a FQUD [21].

Knižnica dostupná vo vývojovom prostredí bola použitá aj pre sériovú komunikáciu. Tej bola nastavená rýchlosť 57600 kBd.

Zvyšok programu prebieha v „nekonečnej slučke“, kde sa neustále kontrolujú tieto činnosti:

1. kontrola, či nedošlo ku zmene nastavenej frekvencie,
2. kontrola, či nedošlo ku zmene obsahu LCD displeja,
3. kontrola, či nedošlo ku stlačeniu tlačítka na rotačnom kóderi,
4. kontrola, či sú dostupné ladiace dáta so sériovej linky,

ktoré sú v prípade výskytu obslúžené. Situácie 1 a 2 nastávajú počas prerušenia, kde sa nastaví príznak ich výskytu a obslúžené sú až v hlavnom programe. Dôvod takéhoto riešenia je taký, aby nedochádzalo k medzivláknovému kríženiu pri volaní funkcií.



Obr. 3.1: Vývojový diagram hlavného programu

Ak bol počas prerušenia nastavený príznak zmeny LCD, je volaná funkcia na výpis dát na displej. V nej sa vždy na začiatku celý displej vymaže a vypne sa kurzor. V ďalšej časti sa rieši správny formát zobrazovaných údajov. V prvom riadku je zobrazovaná nastavená frekvencia, ktorej výpis závisí od veľkosti frekvencie. Frekvencia rovná alebo vyššia ako 1 MHz, je vypísana v tvare 1,000 000 MHz, t.j. oddeľovač miliónov je čiarka a oddeľovač tisícov medzera, pričom na honci čísla je vypísaná jednotka „MHz“. Nižšia frekvencia, ale vyššia alebo rovná ako 1 kHz má ako oddeľovač tisícov použitú čiarku a vypísanú jednotku „kHz“, pričom úvodné nuly nie sú zobrazované. Frekvencia nižšia ako 1 kHz má zobrazenú jednotku „Hz“ a taktiež nie sú zobrazované úvodné nuly. V druhom riadku je zobrazovaný aktuálny nastavený krok, ktorý sa používa pri zvyšovaní alebo znižovaní frekvencie. Hodnoty zobrazených krokov sú: 1 Hz, 10 Hz, 100 Hz, 1 kHz, 10 kHz, 100 kHz, 1 MHz a 10 MHz.

V prípade zistenia príznaku zmeny DDS sa volá funkcia na zápis ladiaceho slova do DDS obvodu. V tejto funkcii je najskôr ladiace slovo zostavené a následne je zapísané do DDS obvodu.

Ladiace slovo má 40 bitov a skladá sa z dvoch častí: 32-bitového frekvenčného ladiaceho slova a 8-bitového fázového a riadiaceho ladiaceho slova. Frekvenčné ladiace slovo (bity 0-31) je jedinečné pre každú frekvenciu, preto je vždy vypočítané na začiatku funkcie pre zápis do DDS obvodu, podľa nasledujúceho vzťahu,

$$LS_{frekvencie} = \frac{2^{32} * f_{vystup}}{f_{clk}}$$

kde f_{clk} je frekvencia použitého kryštálu a f_{vystup} predstavuje nastavenú frekvenciu. Nastavená frekvencia je vnútorná premenná programu a jej hodnota je tiež zobrazovaná na displeji. Modul s DDS obvodom používa kryštáľ s hodnotou 30 MHz, pričom DDS obvod má programovo vypínateľnú násobičku kmitočtu šiestimi, preto je možné využiť ako hodinový signál frekvenciu 30MHz alebo 180 MHz. V tomto návrhu bolo počítané len s frekvenciou 180 MHz. Výpočet ladiaceho slova vyžaduje operácie s 32-bitovými číslami, čo sa ukázalo ako problém pre kompilátor, preto bolo zvolené náhradné riešenie. Podiel $2^{32}/180$ miliónov bol uložený do desatinnej premennej ako konštanta rovná 23,860929 a tou bola násobená nastavená frekvencia. Počet desatinných miest tejto konštanty bol zvolený s ohľadom na to, aby nebol spôsobený rozdiel medzi nastavenou a výstupnou frekvenciou pri rozlíšení syntetizátora 1 Hz.

Fázové a riadiace ladiace slovo (bity 32-39) sa ďalej delí na 5-bitov umožňujúcich nastaviť fázu výstupného signálu (bity 35-39) a 3 riadiace bity (bity 32-34) . Nastavenie fázy je výhodné napríklad v prípade využitia viacerých DDS modulov, s možnosťou ich vzájomného fázového posunu, napríklad pri kvadratúrnej modulácii, preto v našom prípade fáza výstupného signálu nie je nastavovaná. Zostávajúce 3 riadiace bity slúžia na aktiváciu násobičky vstupnej hodinovej frekvencie (bit 32), prechod obvodu do/z režimu spánku (bit 34) a posledný bit je kontrolný a vždy musí mať hodnotu 0 (bit 33). Preto celé frekvenčné a riadiace slovo môže byť pre jednoduchosť vždy konštantné, s binárnou hodnotou 00000001.

Frekvencia																																Fáza a riadenie							
0/1 0	0/1 1	0/1 2	0/1 3	0/1 4	0/1 5	0/1 6	0/1 7	0/1 8	0/1 9	0/1 10	0/1 11	0/1 12	0/1 13	0/1 14	0/1 15	0/1 16	0/1 17	0/1 18	0/1 19	0/1 20	0/1 21	0/1 22	0/1 23	0/1 24	0/1 25	0/1 26	0/1 27	0/1 28	0/1 29	0/1 30	0/1 31	1 32	0 33	0 34	0 35	0 36	0 37	0 38	0 39

Tab. 3.3: Tvar ladiaceho slova

V druhej časti tejto funkcie je zapísané ladiace slovo do DDS modulu. Najskôr je zapísané frekvenčné ladiace slovo, potom fázové a riadiace. Keďže ide o sériový prenos, bola zavedená 32-bitová maskovacia premenná s hodnotou 1. Hodnota masky a hodnota frekvenčného ladiaceho slova je porovnaná logickou funkciou AND, výsledkom čoho je hodnota frekvenčného ladiaceho slova na pozícii nultého bitu. Táto hodnota je privedená do dátového vstupu DDS obvodu D7 a následne doň zapísaná signálom WCLK. V ďalšom kroku dochádza k bitovému posunu hodnoty masky o jeden bit doľava. Celý tento proces sa opakuje 32 krát, čím sa zavŕši zápis frekvenčného ladiaceho slova. Pokračuje rovnaký spôsob zápisu fázového a riadiaceho, uloženého ako konštanta, s tým rozdielom, že sa zavádza 8-bitová maska a proces zápisu sa opakuje 8 krát. Po zápise všetkých 40 bitov je prenos celého ladiaceho slova potvrdený signálom FQUD.

Ďalšou činnosťou hlavného programu v nekonečnej slučke je kontrolovať stlačenie tlačidla. Po zistení jeho stlačenia dochádza ku negácii príznaku „zmena frekvencie“. V závislosti od tohto príznaku je v prerušení možné zistiť, či dôjde pri rotácii kódera k zmene frekvencie alebo kroku. Aby nedochádzalo ku opakovanému vyhodnoteniu raz-zatlačneného tlačidla, dostáva sa program po každom zatlačení tlačidla do slučky, čakajúcej na jeho uvoľnenie.

Poslednou úlohou hlavného programu je čítať dáta zo sériového portu UART. Za platný reťazec dát je považovaná skupina bytov s maximálnou dĺžkou 9 bytov a zakončená znakom „*“. Formát prijatých dát odpovedá požadovanej frekvencii, preto môžu byť tieto dáta priamo prevedené z textovej formy do celočíselnej a následne je nimi nahradená aktuálna frekvencia DDS syntetizátora. Nakoniec sa volajú funkcie na prepísanie obsahu displeja a zápisu do DDS obvodu.

3.2 Prerušenie

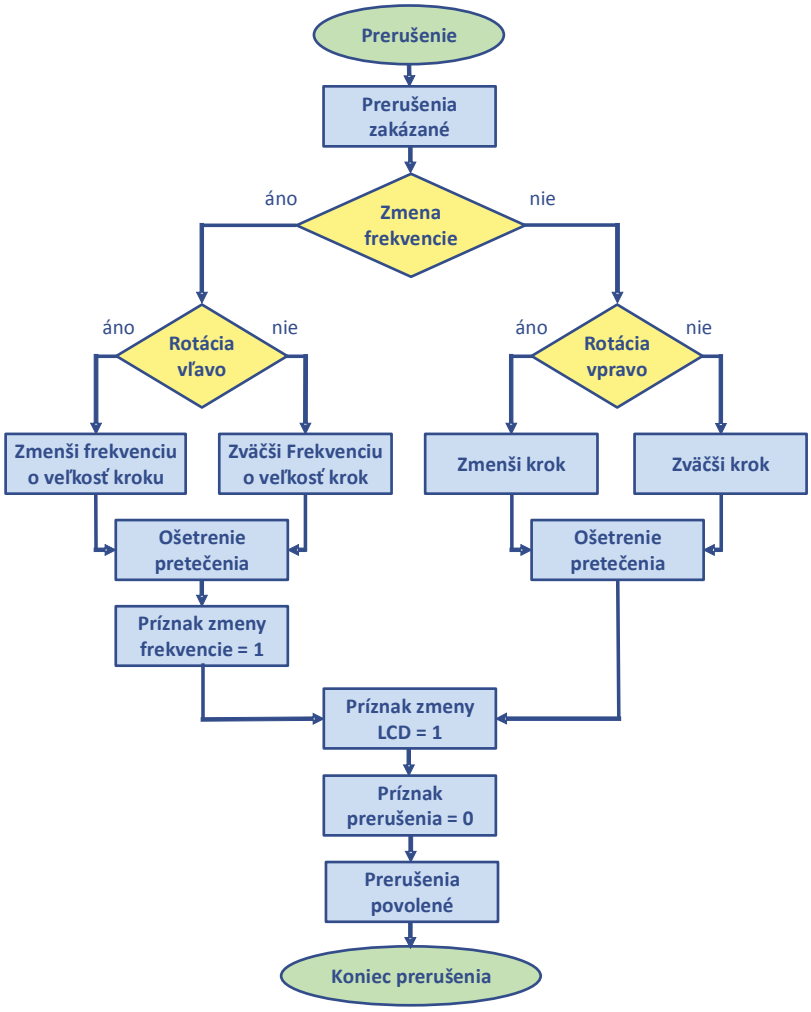
Jediným zdrojom prerušenia môže byť rotačný kóder, ktorého výstupný kanál A je pripojený na mikorprocesorový pin externého prerušenia INT0. Ak dôjde k prerušeniu, hneď na začiatku celej procedúry sú zakázané globálne všetky prerušenia zápisom do príslušného registra. Ďalej sa kontroluje zdroj prerušenia. Táto kontrola v našom prípade nie je nevyhnutná (iný zdroj prerušenia sa neuvažuje), slúži však na sprehľadnenie kódu.

Funkcia rotačného kódera je závislá od nastavenia príznaku „zmena frekvencie“, ktorý sa nastavuje, poprípade resetuje stlačením tlačítka rotačného kódera. V prípade, že je tento príznak nastavený, rotačný kóder bude meniť frekvenciu. V tomto okamihu dochádza ku kontrole kanálu B rotačného kódera, ktorého logická hodnota je určená smerom rotácie hriadeľa. Pri otáčaní kódera proti smeru hodinových ručičiek dôjde

ku zníženiu nastavenej frekvencie o nastavenú veľkosť kroku. Naopak, pri otáčaní hriadeľa v smere hodinových ručičiek sa bude frekvencia zvyšovať o nastavenú veľkosť kroku. V tejto časti kódu je aj ošetrenie, aby nebolo možné prekročiť minimálnu a maximálnu možnú hodnotu frekvencie. Minimálnou nastavenou frekvenciou je 0 Hz a maximálnou 70 MHz. Po zmene frekvencie sa nastaví príznak zmeny frekvencie, ktorý bude obslúžený v hlavnom programe.

V prípade že príznak „zmena frekvencie“ nie je nastavený, rotácia kódera spôsobuje zmenu kroku. Krok je rotačným kóderom nastavovaný v intervale 1-8. Platí, že pri prekročení hodnoty 8 sa jeho hodnota vracia na hodnotu 1 a to isté platí aj obrátene. Z hodnoty kroku je vypočítaná veľkosť kroku, ktorá odpovedá rádu hodnoty kroku. Napríklad 1 odpovedá 1 Hz, 6 odpovedá 100 kHz, atď. Práve o túto veličinu je potom v nasledujúcom prerušení v prípade nastaveného príznaku „zmena frekvencie“, zvyšovaná alebo znižovaná frekvencia.

Po spracovaní funkcie rotačného kódera sa nastaví príznak zmeny LCD, ktorý bude obslúžený v hlavnom programe. Nakoniec sa resetuje príznak externého prerušenia a povolia sa všetky prerušenia.



Obr. 3.2: Vývojový diagram prerušenia

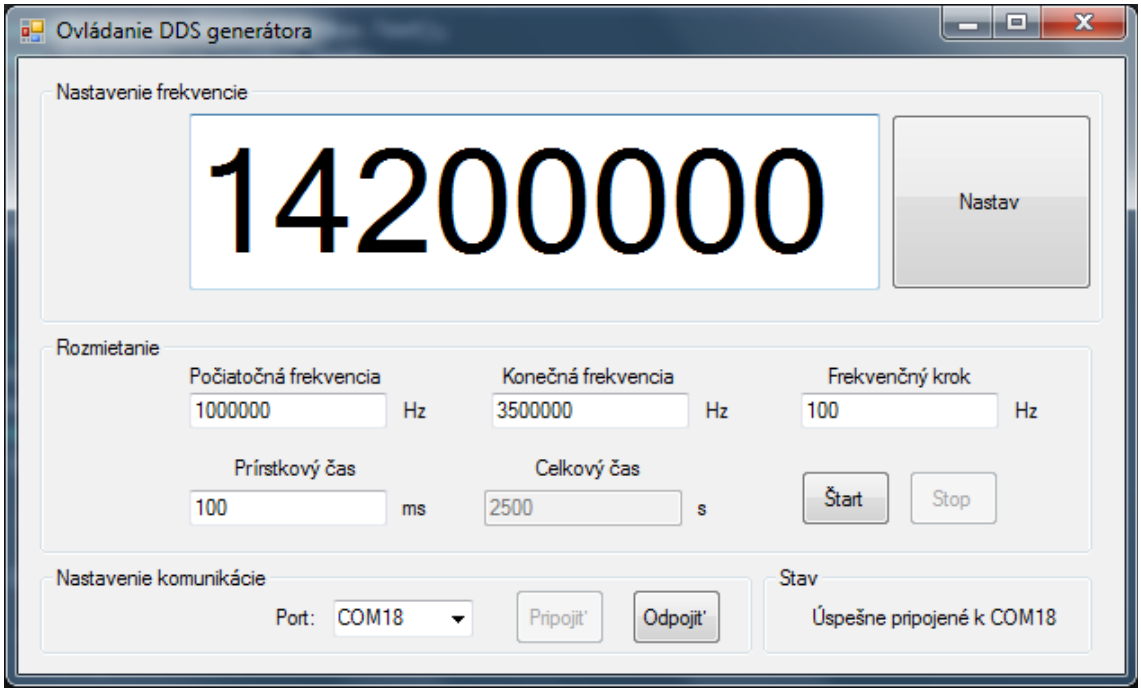
4. Ovládacia aplikácia pre PC

Ovládací aplikácia umožňuje ovládať DDS syntetizátor pomocou osobného počítača. Umožňuje priame nastavovanie zadanej frekvencie a rozmietanie frekvencie. Prepojenie s počítačom môže byť realizované portom USB alebo Bluetooth, ktorý sa pre program javí ako fyzický sériový port COMx, pričom x je číslo z intervalu 0-255.

Aplikácia bola napísaná v programovacom jazyku C#, za použitia knižnice WinForms, vo vývojovom prostredí Visual Studio 2015, ktorý je pre študentov k dispozícii bezplatne.

4.1 Grafická časť

Grafická podoba ovládacej aplikácie je znázornená na obrázku 4.1.



Obr. 4.1: Grafická podoba ovládacej aplikácie

Najhlavnejšou časťou aplikácie je zadávacie pole pre frekvenciu v jednotkách Hertzoch. Táto frekvencia je pomocou tlačidla „Nastav“ odoslaná do DDS syntetizátora.

Druhou funkciou aplikácie je možnosť rozmietania frekvencie. Parametrov rozmietania sa týka päť textových polí, pričom užívateľ zadáva žiadané hodnoty do štyroch z nich: „Počiatočná frekvencia“, „Konečná frekvencia“, „Frekvenčný krok“ a „Prírastkový čas“. V prvých troch textových poliach je zadávaná jednotka Hertz, jednotka v štvrtom poli je milisekunda. Hodnotu piateho textového poľa „Celkový čas“ vypočítava aplikácia a udáva celkovú dobu trvania jednej periódy rozmietania. Pre zahájenie rozmietania slúži tlačidlo „Štart“ a zastavuje sa tlačidlom „Stop“

Pred samotným použitím aplikácie je nevyhnutné pripojiť sa k príslušnému portu COMx, dostupnom v rozbaľovacej ponuke. Bez pripojenia k príslušnému portu nie je možné aplikáciu používať, lebo všetky ovládacie prvky sú deaktivované. Pripojenie sa vykonáva výberom príslušného portu z rozbaľovacej ponuky a stlačením tlačidla „Pripojiť“ a odpája sa tlačidlom „Odpojiť“.

V pravom dolnom rohu aplikácie sa nachádza textové pole, ktoré informuje užívateľa o aktuálnom stava aplikácie, napríklad úspešnosť pripojenia k portu COMx, posledná nastavená frekvencia, chybné hodnoty textových polí, atď. Zápis programu grafickej časti aplikácie sa nachádza v prílohe B.

4.2 Programová časť

Programová časť aplikácie zabezpečuje správnu funkciu grafických ovládacích prvkov a tiež zabezpečuje komunikáciu so sériovým portom. Jej úlohou je reagovať na stlačenie tlačidiel, čítanie a zapisovanie textových polí, kontrola prípustnosti zadaných hodnôt a ošetrovanie výnimiek, ktoré môžu nastať počas behu programu.

V nasledujúcej tabuľke je uvedený zoznam pomenovania a významu ovládacích prvkov s ktorými bude program pracovať.

Typ	Názov	Popis
Form	Form1	Okno so všetkými ovládacími prvkami
textBox	nastavTextBox	Zadávacie pole pre nastavenie frekvencie
Button	nastavButton	Potvrdenie a odoslanie zadanej frekvencie
textBox	pociatokTextBox	Počiatočná frekvencia rozmietania
textBox	koniecTextBox	Konečná frekvencia rozmietania
textBox	krokTextBox	Krok frekvencie pri rozmietaní
textBox	prirastokTextBox	Časový prírastok frekvencie rozmietania
textBox	celkovyTextBox	Vypočítaný celkový čas rozmietania
Button	rozmStartButton	Spustenie rozmietania
Button	rozmStopButton	Zastavenie rozmietania
ComboBox	serialSelectComboBox	Ponuka dostupných COMx portov
Button	serialStartButton	Pripojenie k vybranému COMx portu
Button	serialStopButton	Odpojenie od vybraného COMx portu
Label	stavLabel	Zobrazuje aktuálny stav aplikácie

Tab. 4.1: Zoznam ovládacích prvkov, s ktorými pracuje program

Všetky uvedené ovládacie prvky sú objekty, ktoré majú širokú škálu vlastností a metód a pri vývoji aplikácie pre syntetizátor bolo niekoľko z týchto vlastnosti a metód využitých.

Celý program je zapísaný v rámci triedy `Form1`, ktorá dedí od triedy `Form`. Na začiatku triedy sú deklarované dve statické premenné. Jedna je

port typu *SerialPort*, ktorá musí byť viditeľná pre celú aplikáciu a bude v sebe uchovávať parametre a prístup k zvolenému sériovému portu. Druhá deklarovaná statická premenná je typu *BackgroundWorker* a je navyše aj inicializovaná novým objektom *backgroundWorker* typu *BackgroundWorker*. Táto premenná bude musieť byť tiež viditeľná pre celý program, aby mohlo kedykoľvek dôjsť k zastaveniu *BackgroundWorker*. Funkcia *BackgroundWorker* bude vysvetlená neskôr.

V konštruktore triedy *Form1* sa volá metóda *InitializeComponent()*, ktorá automaticky definuje všetky ovládacie prvky, viditeľné na formulári, podľa grafického návrhu aplikácie. Nasledujúcou úlohou konštruktora *Form1* je priradiť k udalosti *backgroundWorker.DoWork* delegáta *Rozmietanie* Inými slovami, v prípade aktivovania *backgroundWorker* dôjde k volaniu metódy *Rozmietanie*.

Nasledujúcou činnosťou po inicializácii komponent, je volanie metódy *Form1_Load*, ktorá sa spúšťa automaticky po spustení okna. Táto metóda volá ďalšiu metódu *NaplnenieComboBoxu()*.

Metóda *NaplnenieComboBoxu()* sa stará o získanie zoznamu aktuálne dostupných portov metódou *SerialPort.GetPortNames()* a tento zoznam vloží ako položky do rozbaľovacej ponuky *serialSelectComboBox.DataSource*.

Aby bolo možné získavať zoznam dostupných COMx portov aj počas behu aplikácie, čo môže byť požadované v prípade pripojenia syntetizátora k počítaču až po spustení aplikácie, bola využitá metóda rozbaľovacej ponuky *serialSelectComboBox_Click*, ktorej jedinou úlohou je po kliknutí na rozbaľovaciu ponuku volať metódu *NaplnenieComboBoxu()*.

Akonáhle sa vykonajú predchádzajúce činností, program nevykonáva žiadnu činnosť a čaká na prípadnú udalosť, napríklad stlačenie nejakého tlačidla alebo ukončenie aplikácie.

Po spustení aplikácie je nutné pripojiť sa k nejakému portu COMx, preto majú všetky ostatné ovládacie tlačidlá okrem tlačidla *serialStartButton* („Pripojiť“) nastavenú vlastnosť *Enable=false*. Jedinou možnosťou je teda vybrať správny port v rozbaľovacej ponuke *serialSelectComboBox* a kliknúť na tlačidlo *serialStartButton*.

Po kliknutí na tlačidlo *serialStartButton* dochádza ku kontrole, či bol zvolený port COMx v rozbaľovacej ponuke *serialSelectComboBox* . Ak nebol, užívateľ uvidí vyskakovacie okno so správou „Port nebol zvolený“ a taká istá správa sa objaví aj v popise aktuálneho stavu aplikácie v *stavLabel.text*.

V prípade, že port COMx bol zvolený, dôjde k volaniu metódy *Connect(port)* s jedným parametrom typu *SerialPort*. V tomto parametri je prenesený zvolený port a je vykonaný pokus o pripojenie sa k nemu s nastavenou dátovou rýchlosťou 57 600 kBd. Pokiaľ sa pokus o pripojenie nepodarí, je zachytená výnimka. Zachytenie výnimky predíde pádu aplikácie a v rámci jej obsluhy je skutočnosť o nepodarenom pokuse vypísaná do *stavLabel.text*.

Po úspešnom pripojení ku zvolenému portu COMx sa nastaví vlastnosť *serialStartButton.Enable=false*, zatiaľčo u všetkých ostatných tlačidiel sa táto vlastnosť nastaví na *True*. Tým sa aktivovali všetky tlačidlá a deaktivovalo sa len pripájacie tlačidlo. Nakoniec sa zapíše hláška úspešného pripojenia do *stavLabel.text*.

Žiadaná frekvencia sa vpisuje do textového poľa *nastavTextBox*. Odoslanie frekvencie do syntetizátora sa vykonáva stlačením klávesy Enter

alebo kliknutím na tlačidlo *nastavButton* („Nastav“). Obidve udalosti volajú metódu *nastavButton_Click()*, v ktorej dôjde najskôr ku konverzii textového poľa do číselnej premennej *frekvencia*. V prípade, že je táto premenná v rozsahu 0 až 70 000 000, opätovne sa prekonvertuje do textovej podoby a na jej koniec je doplnený znak „*“, na ktorý reaguje program mikroprocesora. Takto doplnený textový reťazec je odoslaný po sériovej linke syntetizátoru. Úspešné vykonanie zápisu je oznámené v *stavLabel.text* hodnotou zapísanej frekvencie. Ak je zadaná hodnota mimo požadovaný rozsah, do *stavLabel.text* bude zapísané „Hodnota mimo rozsah“. Ak zadanú hodnotu nie je možné prekonvertovať do číselnej premennej, do *stavLabel.text* bude zapísané „Neplatné znaky“. Ak nastane iná chyba bude do v *stavLabel.text* zapísané „Neznáma chyba“.

Časť programu rozmietanie obsahuje 4 vstupné textové polia: *pociatokTextBox*, *koniecTextBox*, *krokTextBox*, *prirastokTextBox*. Počas vypĺňania týchto textových polí je po každom opustení daného poľa dynamicky vypĺňané piate, výstupné textové pole *celkovyTextBox*, kde je aplikáciou vypočítavaný celkový čas aktuálne nastaveného rozmietania.

Po kliknutí na tlačidlo *rozmStartButton* sa najskôr vykoná kontrola prípustnosti zadaných hodnôt v metóde *KontrolaRozmietania()* typu *bool*. V tejto metóde sú prekonvertované textové hodnoty zo vstupných polí do číselnej podoby. Ak nie je možné nejaké zo vstupných polí prekonvertovať, metóda vracia hodnotu *False*. V opačnom prípade sa skontroluje číselný rozsah platných hodnôt. Platnou hodnotou poľa *pociatokTextBox* je 0 až 69 999 999, inak je zapísané do *stavlabel.text* „Počiatočná frekvencia mimo rozsah“ a vrátená hodnota metódy *false*. Pre pole *koniecTextBox* platí rozsah 1 až 70 000 000 a zároveň musí byť toto číslo väčšie ako v *pociatokTextBox*, inak je zapísané do *stavlabel.text* „Konečná frekvencia mimo rozsah“ a vrátená hodnota metódy *false*. Textové pole *krokTextBox* musí byť väčší alebo rovný ako 0 a zároveň maximálne tak veľký ako rozdiel *koniecTextBox* a *pociatokTextBox*, inak je zapísané do *stavlabel.text* „Nesprávna hodnota kroku“ a vrátená hodnota metódy *false*. Posledným kontrolovaným rozsahom je *prirastokTextBox*, kde je z dôvodu maximálnej možnej rýchlosti zmeny frekvencie, danej syntetizátorom, kontrolovaná minimálna hodnota rovná 100 ms, v opačnom prípade je zapísané do *stavlabel.text* „Najmenší prírastok je 100 ms“ a vrátená hodnota metódy *false*. Pri úspešnom výsledku kontroly všetkých vstupných polí vráti metóda hodnotu *true*. Rozmietanie môže začať len pri úspešnom výsledku kontroly.

Rozmietanie prebieha periodicky v nekonečnom cykle, kedy sa v podcykle odošle na sériový port najskôr počiatočná frekvencia, potom je program pozastavený na nastavenú dobu časového prírastku, následne je zvýšená hodnota počiatočnej frekvencie o nastavený krok a celý podcyklus sa opakuje až po dosiahnutie konečnej frekvencie, kedy sa rozmieta znovu od začiatku. Takéto správanie by samozrejme viedlo k zamrznutiu celej aplikácie, preto je táto činnosť vykonávaná na pozadí (v inom vlákne). Vykonávať nejakú činnosť na pozadí umožňuje objekt triedy *BackgroundWorker*. Naším objektom tejto triedy je backgroundworker. Objekt *backgroundWorker* sa spustí metódou *RunWorkerAsync()* a po jeho spustení je v ňom vykonávaná metóda na ktorú ukazuje delegát jej udalosti *DoWork*. V našom prípade bol tento delegát priradený v konštruktore triedy *Form1* a teda ukazuje na metódu *Rozmietanie()*. V tejto metóde prebieha

hore popísaný rozmietací cyklus. Objekt *backroundWorker* má nastavenú vlastnosť *WorkerSupportsCancellation=true*, čím je ho možné zastaviť z iného vlákna. Poslednými činnosťami po stlačení *rozmStartButton* je aktualizácia s*tavlabel.text* na „Rozmietanie spustené“, deaktívácia tlačidla *rozmStartButton* a aktivácia *rozmStopButton*.

Tlačidlo *rozmStopButton* zastavuje *backgroundWorker*, deaktivuje *rozmStopButton* a aktivuje *rozmStartButton*.

Tlačidlo *serialStopButton*, v prípade, že beží *backgroundWorker*, zastaví ho, odpojí aplikáciu od portu COMx a deaktivuje všetky tlačítka okrem *serialStartButton,* ktoré naopak, aktivuje.

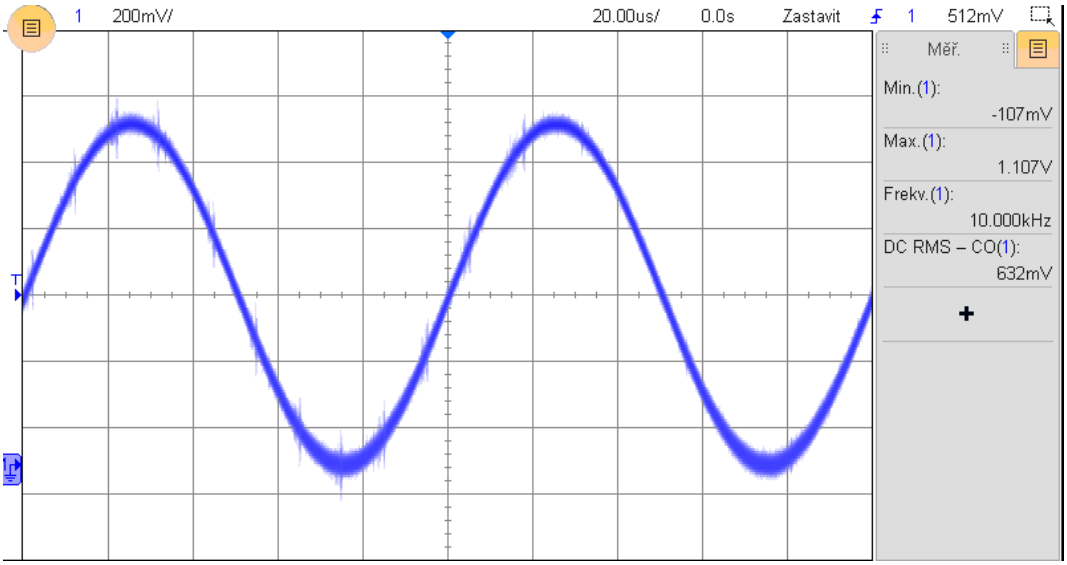
Užívateľské ukončenie aplikácia automaticky zavolá tesne pred jej ukončením metódu *Form1_FormClosing*, kde sa predíde výnimkám ukončením *backgroundWorker* a odpojeniu od portu COMx, v prípade, že už tak nebolo učinené užívateľom.

5. Meranie

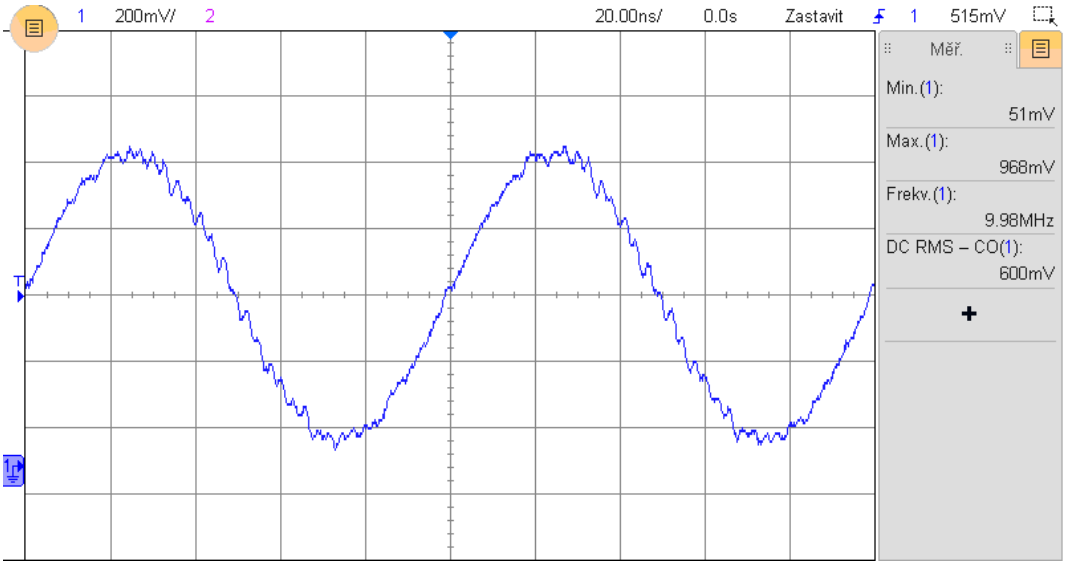
Meranie malo za cieľ overiť správnosť návrhu a funkcie syntetizátora s ohľadom na kvalitatívne parametre výstupného signálu. Boli zmerané nasledujúce parametre:

1. prítomnosť sínusového priebehu
2. presnosť nastavenej frekvencie
3. prítomnosť harmonických zložiek
4. frekvenčná charakteristika filtrovaného signálu

Prítomnosť sínusového priebehu bola dokázaná pripojením syntetizátora ku osciloskopu KEYSIGHT MSO-X-3102T pre dve rôzne nastavené frekvencie.



Obr. 5.1: Sínusový priebeh pre výstupnú frekvenciu 10 kHz



Obr. 5.2: Sínusový priebeh pre výstupnú frekvenciu 10 MHz

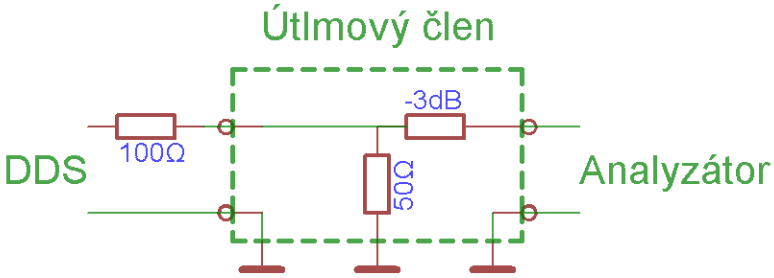
Na obrázku 5.1, zobrazujúcom výstupnú frekvenciu 10 kHz, je sínusový priebeh, vzhľadom k vysokej vstupnej hodinovej frekvencii DDS obvodu, bez viditeľných deformácií. Deformácie sínusového priebehu však vznikali so zvyšujúcou sa frekvenciou. Na obrázku 5.2, získanom pre frekvenciu 10 MHz, sú už viditeľné väčšie nepravidelnosti oproti ideálnemu priebehu. Tento jav je pri DDS prirodzený a je spôsobený zvyšovaním pomeru výstupnej a vzorkovacej frekvencie.

Orientačné zistenie presnosti nastavenej frekvencie bolo odmerané presným frekvenčným čítačom HP-5385A. Rozdiel medzi nastavenou a nameranou frekvenciou bol spôsobený zaokruhľovaním ladiacej konštanty v programe mikroprocesora.

DDS generátor [Hz]	Presný čítač [Hz]
10	9,97
100	100,00
1 000	999,96
10 000	9 999,99
100 000	99 999,99
1 000 000	999 999,98
10 000 000	10 000 000,95

Tab. 5.1: Nastavené a namerané frekvencie

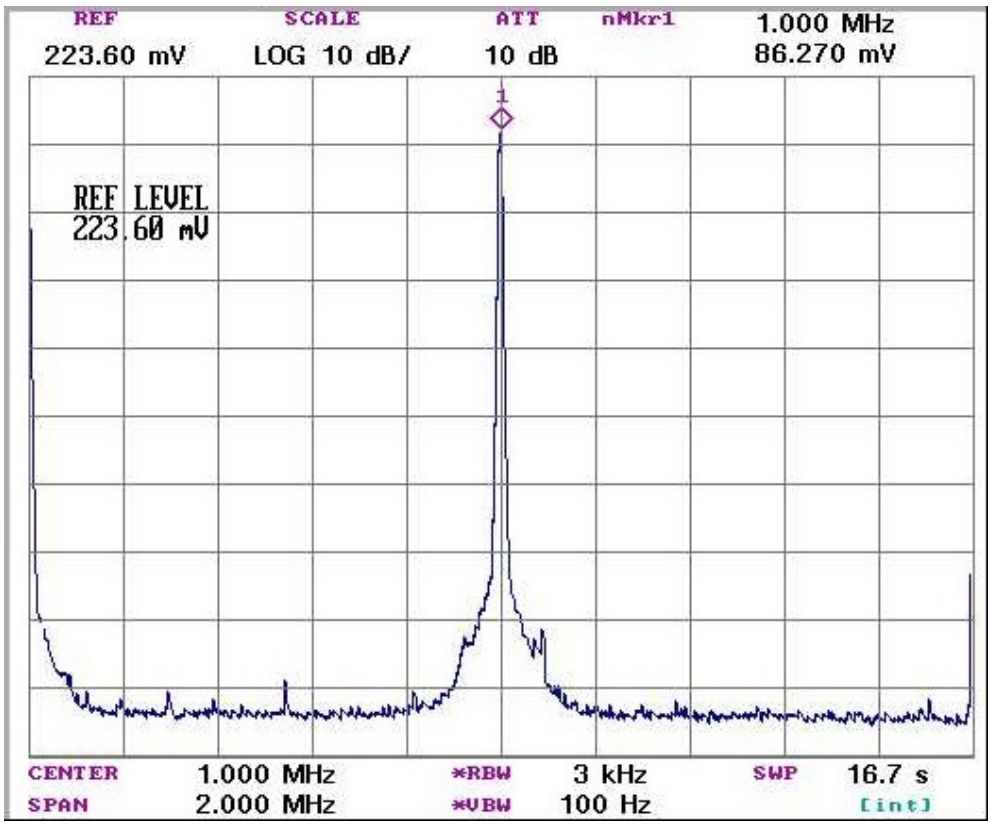
Meranie frekvenčného spektra syntetizátora bolo realizované pomocou spektrálneho analyzátora s označením 2399B 9kHZ to 3GHz Spectrum Analyzer. Z dôvodu nízkeho maximálneho vstupného napätia analyzátora bol medzi syntetizátor a analyzátor zaradený útlmový článok s hodnotou -3 dB.



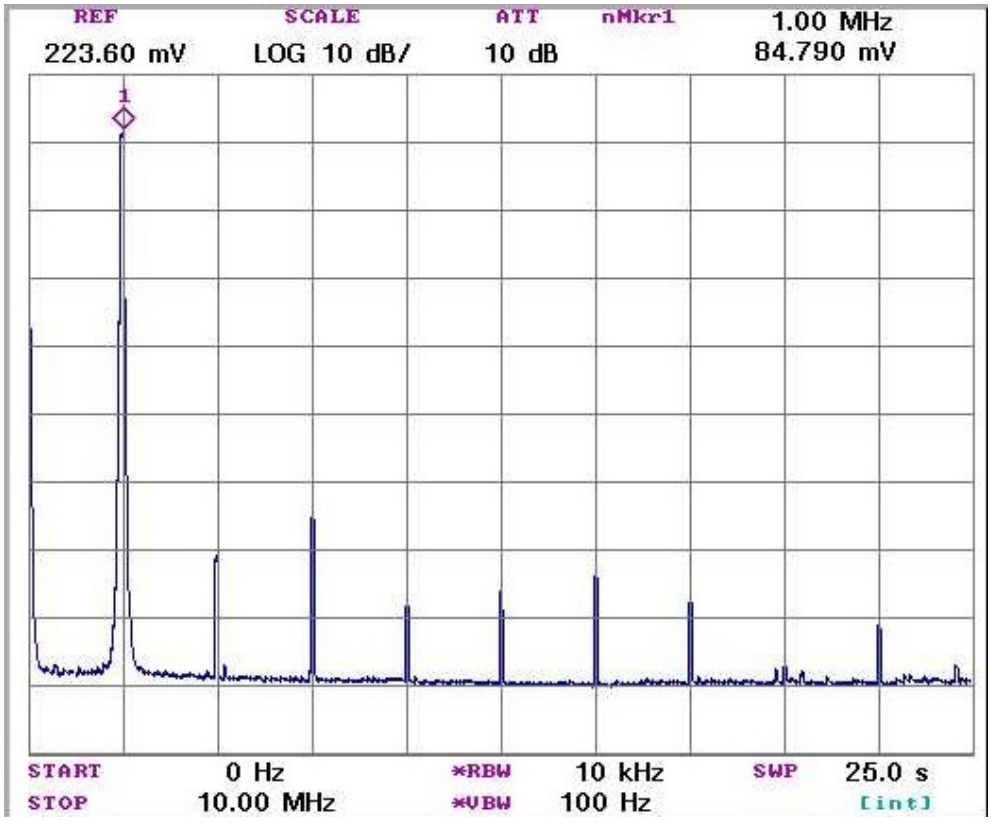
Obr. 5.3: Schéma pripojenia syntetizátora ku analyzátoru

Táto úprava mala vplyv na analyzátorom zobrazovanú hodnotu napätia. Hodnotu napätia na strane DDS je možné vypočítať podľa nasledujúceho vzťahu:

$$U_{DDS} = \frac{U_{AN}}{10^{\frac{-3}{20}}} \cdot \frac{150}{50}$$



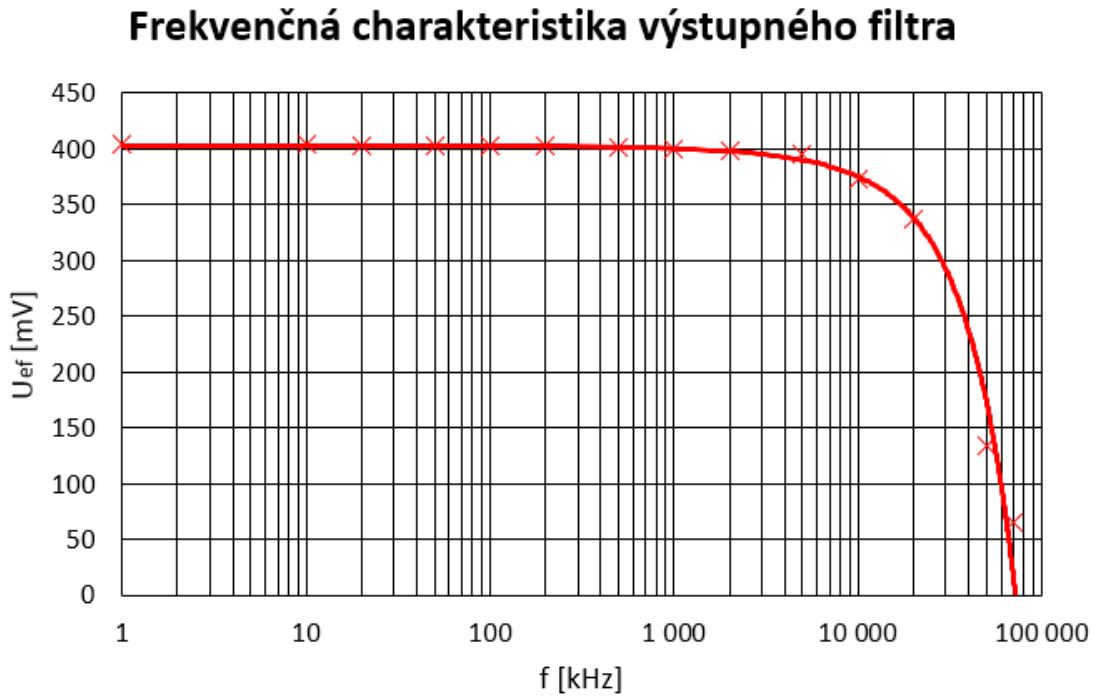
Obr. 5.4: Frekvenčné spektrum v okolí nastavenej frekvencie 1 MHz



Obr. 5.5: Vyššie harmonické nastavenej frekvencie 1 MHz

Z nameraných grafov je vidieď, že pri frekvencii 1 MHz bola najsilnejšia tretia harmonická základnej frekvencie, t.j. 3 MHz a jej útlm oproti základnej frekvencii bol 9,5 dB.

Výstupom DDS modulu sú dva signály: jeden je priamy výstup z D/A prevodníka DDS obvodu a druhý je fázovo posunutý o 180° a filtrovaný dolnopriepustným LC filtrom osadeným na DDS module. Na nasledujúcom obrázku je zmeraná výstupná frekvenčná charakterictika tohto filtra.



Obr. 5.6: Frekvenčná charakteristika výstupného filtra

Z nameraného priebehu možno usúdiť, že limitná frekvencia filtru pri zaťažení 50Ω je približne 10 MHz, čo nezodpovedá deklarovanej frekvencii 70 MHz. Dôvodom by mohol byť fakt, že hodnota limitnej frekvencie, deklarovaná výrobcom platí pre nezaťažený filter.

Záver

Cieľ mojej diplomovej práce, navrhnúť a skonštruovať programovo riadený syntetizátor priamej číslicovej syntézy vrátane riadiaceho programu pre mikroprocesor a ovládacej aplikácie pre PC som úspešne splnil.

Výsledkom je generátor sínusového signálu v rozsahu 1 Hz až 70 MHz s minimálnym ladiacim krokom 1 Hz. Generátor je štandardne ovládaný pomocou jedného rotačného kódera, slúžiaceho na ladenie frekvencie a nastavovanie ladiaceho kroku. Nastavená frekvencia a krok sú zobrazované na LCD displeji. Výstupom sú dva signály (s/bez výstupného filtra) vyvedené pomocou BNC konektorov. Syntetizátor je možné pomocou USB konektora alebo technológie Bluetooth pripojiť k PC a ovládať ho aplikáciou navrhnutou v tejto práci. Táto aplikácia umožňuje okrem nastavenia frekvencie aj jej rozmietanie.

Vnútorná štruktúra syntetizátora je navrhnutá tak, aby bolo v budúcnosti možné rozšíriť jeho funkciu napríklad modulom na spracovanie prijatého signálu, čím bude možné celé zariadenie využívať ako obvodový alebo anténny analalyzátor (wobler).

Zdroje

[1] *Fundamentals of Direct Digital Synthesis (DDS)* [online]. ANALOG DEVICES, 2009 [cit. 2016-03-2]. Dostupné z: http://www.analog.com/media/en/training-seminars/tutorials/MT-085.pdf

[2] *All About Direct Digital Synthesis* [online]. ANALOG DEVICES, 2004 [cit. 2016-03-2]. Dostupné z: http://www.analog.com/library/analogDialogue/archives/38-08/dds.pdf

[3] *Direct Digital Synthesis: A Tool for Periodic Wave Generation (Part 1)* [online]. LIONEL CORDESSES, 2004 [cit. 2016-03-2]. Dostupné z: http://lionel.cordesses.free.fr/gpages/DDS1.pdf

[4] *DDS design* [online]. EDN network, 2004 [cit. 2016-03-2]. Dostupné z: http://www.edn.com/design/test-and-measurement/4332832/DDS-design

[5] *A Technical Tutorial on Digital Signal Synthesis [online]. ANALOG DEVICES, 1999 [cit. 2016-03-2]. Dostupné z: https://www.ieee.li/pdf/essay/dds.pdf*

[6] *Analog devices* [online]. A Premier Farnell Company, 2016 [cit. 2016-03-2]. Dostupné z: http://cz.farnell.com/analog-devices?searchRef=SearchLookAhead

[7] *CMOS 180 MHz DDS/DAC Synthesizer* [online]. ANALOG DEVICES, 2004 [cit. 2016-03-2]. Dostupné z: http://www.analog.com/media/en/technical-documentation/data-sheets/AD9851.pdf

[8] *16 x 2 Character LCD* [online]. Vishay, 2002 [cit. 2016-03-2]. Dostupné z: http://www.engineersgarage.com/sites/default/files/LCD%2016x2.pdf

[9] *A Guide to Debouncing* [online]. The University of Utah, 2008 [cit. 2016-03-2]. Dostupné z: http://www.eng.utah.edu/~cs5780/debouncing.pdf

[10] *De-bouncing circuits [online]. IKAlogic, 2007 [cit. 2016-03-2]. Dostupné z: https://www.ikalogic.com/de-bouncing-circuits/*

[11] *Switch Debouncing* [online]. electroSome, 2014 [cit. 2016-03-2]. Dostupné z: https://electrosome.com/switch-debouncing/

[12] *USB Documentation* [online]. Cornell University: Christopher D. Leary and Devrin Talen, 2007 [cit. 2016-03-2]. Dostupné z: https://electrosome.com/switch-debouncing/

[13] *USB 3.0 / 3.1 Speed & Drive Benchmark* [online]. EeverythingUSB, 2012 [cit. 2016-03-2]. Dostupné z: http://www.everythingusb.com/speed.html

[14] *USB to UART Bridge - FT232RL* [online]. SparkFun, 2016 [cit. 2016-03-2]. Dostupné z: https://www.sparkfun.com/products/650

[15] *FT232R USB UART: Datasheet* [online]. Future Technology Devices International Ltd., 2015 [cit. 2016-03-2]. Dostupné z: http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf

[16] *Bluetooth Basics* [online]. SparkFun, 2015 [cit. 2016-03-2]. Dostupné z: https://learn.sparkfun.com/tutorials/bluetooth-basics

[17] *Profiles Overview* [online]. Bluetooth SIG, 2010 [cit. 2016-03-2]. Dostupné z: https://developer.bluetooth.org/TechnologyOverview/Pages/Profiles.aspx

[18] *Dispelling Common Bluetooth Misconceptions* [online]. SANS Technology Institute, 2013 [cit. 2016-03-2]. Dostupné z: http://www.sans.edu/research/security-laboratory/article/bluetooth

[19] *Class1 BC04-ext Module BTM-222: Datasheet* [online]. Rayson, 2005 [cit. 2016-03-2]. Dostupné z: http://www.micropik.com/PDF/btm222.pdf

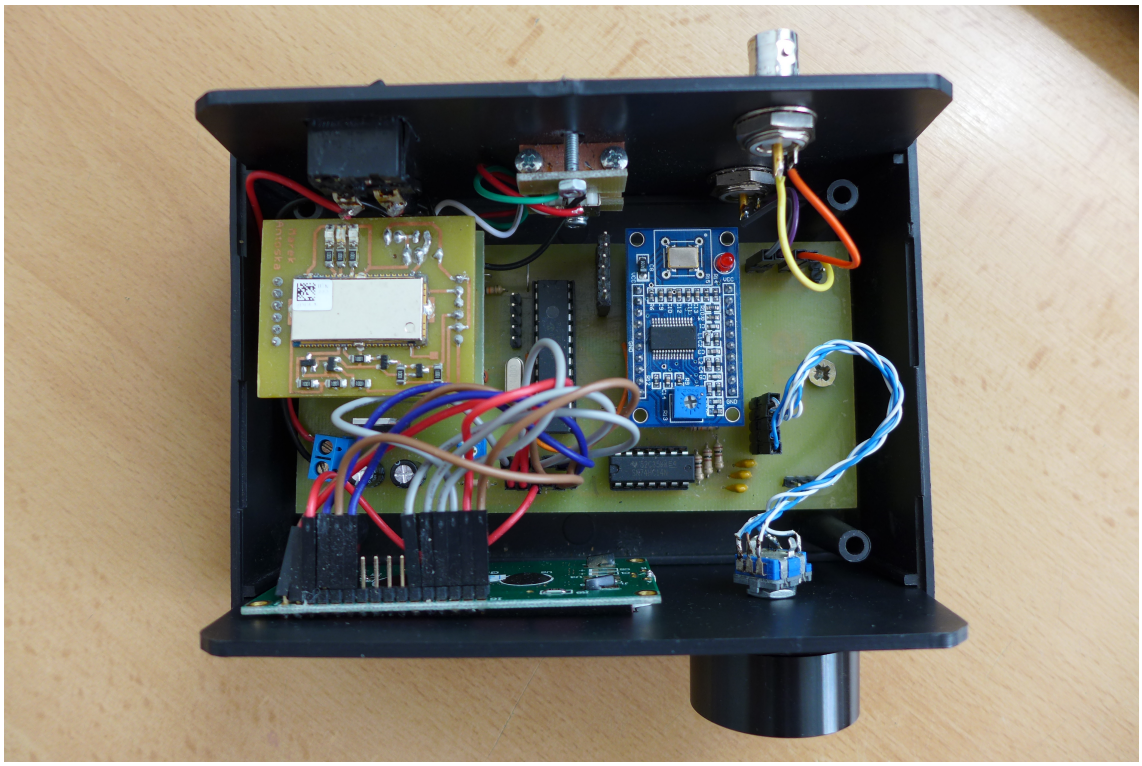
[20] *Bluetooth® Dongle/ Module: Produkty* [online]. 2016 [cit. 2016-03-2]. Dostupné z: http://www.bluetooth-products.com.tw/btm220.html

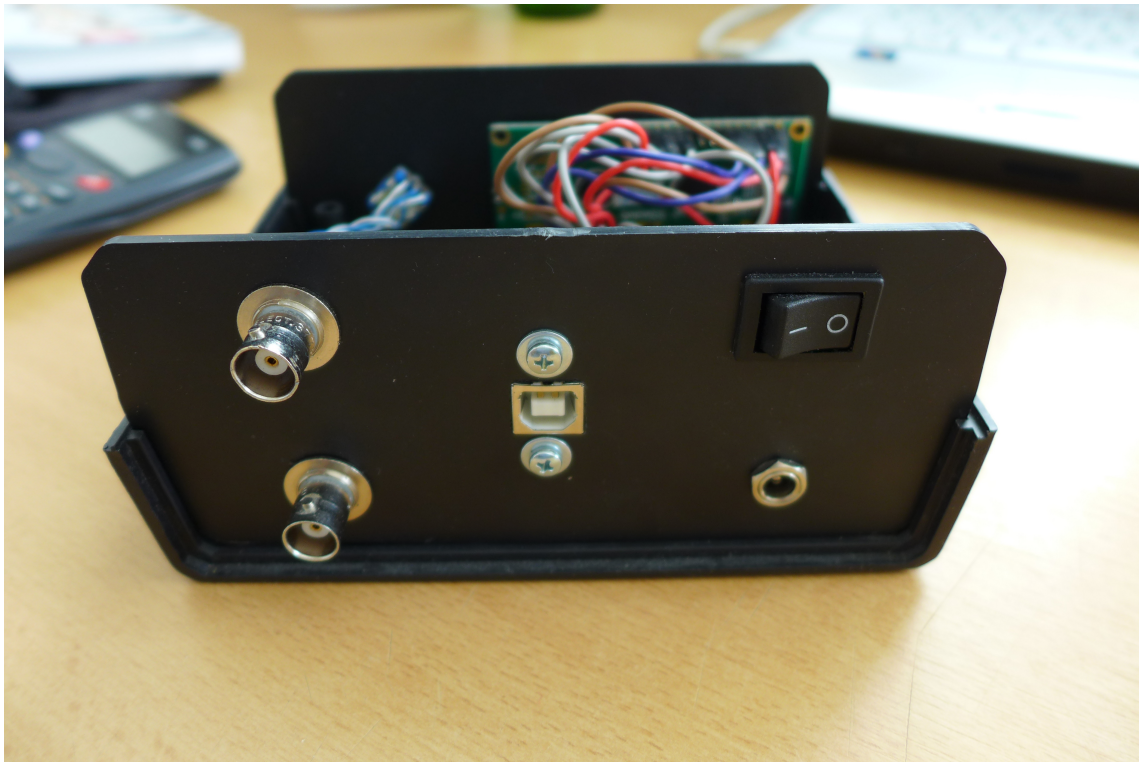
[21] *PIC18FXX2 Datasheet: High-Performance, Enhanced Flash Microcontrollers with 10-Bit A/D* [online]. Microchip, 2006 [cit. 2016-03-2]. Dostupné z: http://ww1.microchip.com/downloads/en/DeviceDoc/39564c.pdf

[22] *MikroC: Produkty* [online]. MikroElektronika, 2016 [cit. 2016-03-2]. Dostupné z: http://www.mikroe.com/mikroc/

Zoznam príloh

Príloha A






20,140 440 MHz
10 MHz

Príloha B

```
#define True 1              //definovanie boolovských názvov
#define False 0

#define DDS_WCLK PORTB.RB5      //definovanie pinov pre DDS modul
#define DDS_FQUD PORTB.RB4
#define DDS_D7 PORTB.RB3
#define DDS_RESET PORTA.RA5

#define ROT_KODER_B PORTB.RB1     //definovanie pinov pre rotačný kóder, kanál A
#define ROT_KODER_TL PORTB.RB2  //je pripojený na prerušnie INT0 (PORTB.RB0)

sbit LCD_RS at RC4_bit;         //určenie pinov pre LCD display
sbit LCD_EN at RC5_bit;         //tento zápis vyžaduje LCD knižnica
sbit LCD_D7 at RC0_bit;
sbit LCD_D6 at RC1_bit;
sbit LCD_D5 at RC2_bit;
sbit LCD_D4 at RC3_bit;

sbit LCD_RS_Direction at TRISC4_bit;    //určenie smeru pinov pre LCD display
sbit LCD_EN_Direction at TRISC5_bit;
sbit LCD_D7_Direction at TRISC0_bit;
sbit LCD_D6_Direction at TRISC1_bit;
sbit LCD_D5_Direction at TRISC2_bit;
sbit LCD_D4_Direction at TRISC3_bit;


bit zmenaLCD;                   //príznak zmeny obsahu displej
bit zmenaFreq;                  //príznak zmeny frekvencie
bit nastavenieFrekvencie;       //príznak menenie frekvencie/kroku v rot.kódery
long nastavenaFreq =440;        //nastavenie počiatočnej frekvencie (0-70000000)
short krok=1;                   //nastavenie počiatočného kroku (1-8)
long velkostKroku=1;            //súčtový prírastok frekvencie
short fazaRiadenie =1;          //00000001 nastavuje násobenie frekvencie DDS 6x
char nastavenaFreq_str[12];     //znakové pole pre zobrazenie frekvencie na LCD
char Output[10];                //Pole prijatých znakov od sériovej linky

void zapisLCD(){

 LongToStr(nastavenaFreq,nastavenaFreq_str);

 Lcd_Cmd(_LCD_CLEAR);
 Lcd_Cmd(_LCD_CURSOR_OFF);

 if(nastavenaFreq>999999){
   Lcd_Chr(1,2,nastavenaFreq_str[3]);
   Lcd_Chr_Cp(nastavenaFreq_str[4]);
   Lcd_Chr_Cp(',');
 }
 if(nastavenaFreq>999){
   Lcd_Chr(1,5,nastavenaFreq_str[5]);
   Lcd_Chr_Cp(nastavenaFreq_str[6]);
   Lcd_Chr_Cp(nastavenaFreq_str[7]);
   if(nastavenaFreq>999999){
     Lcd_Chr_Cp(' ');
   }
   else{
     Lcd_Chr_Cp(',');
   }
 }
   Lcd_Chr(1,9,nastavenaFreq_str[8]);
   Lcd_Chr_Cp(nastavenaFreq_str[9]);
```

```
    Lcd_Chr_Cp(nastavenaFreq_str[10]);
    Lcd_Chr_Cp(' ');
  if(nastavenaFreq>999999){
    Lcd_Chr_Cp('M');
    Lcd_Chr_Cp('H');
    Lcd_Chr_Cp('z');
  }

    else if(nastavenaFreq>999){
    Lcd_Chr_Cp('k');
    Lcd_Chr_Cp('H');
    Lcd_Chr_Cp('z');
  }

    else{
    Lcd_Chr_Cp('H');
    Lcd_Chr_Cp('z');
  }

  if(krok==1){
    Lcd_Out(2, 1, "1 Hz");
  }
  else if(krok==2){
    Lcd_Out(2, 1, "10 Hz");
  }
  else if(krok==3){
    Lcd_Out(2, 1, "100 Hz");
  }
  else if(krok==4){
    Lcd_Out(2, 1, "1 kHz");
  }
  else if(krok==5){
    Lcd_Out(2, 1, "10 kHz");
  }
  else if(krok==6){
    Lcd_Out(2, 1, "100 kHz");
  }
  else if(krok==7){
    Lcd_Out(2, 1, "1 MHz");
  }
  else if(krok==8){
    Lcd_Out(2, 1, "10 MHz");
  }
}
 void VypocetKroku(){
  if(krok==1){
    velkostKroku=1;
  }
  else if(krok==2){
    velkostKroku=10;
  }
  else if(krok==3){
    velkostKroku=100;
  }
  else if(krok==4){
    velkostKroku=1000;;
  }
  else if(krok==5){
    velkostKroku=10000;
  }
  else if(krok==6){
    velkostKroku=100000;
  }
  else if(krok==7){
    velkostKroku=1000000;
```

```
  }
  else if(krok==8){
    velkostKroku=10000000;
  }
}

void zapisDDS(){

   unsigned long ladiaceSlovoA;
   unsigned long ladiaceSlovoB;
   unsigned long ladiaceSlovo;
   unsigned long maska32b;
   short maska8b;
   int i;

   ladiaceSlovo=0;
   //1Hz = 23.86092942LS
   //treba vykonať ladiaceSlovo=nastavenaFreq*23.86092942
   ladiaceSlovoA=23*nastavenaFreq+8*nastavenaFreq/10+6*nastavenaFreq/100;
   ladiaceSlovoB=9*nastavenaFreq/10000+29*nastavenaFreq/1000000;
   ladiaceSlovo=ladiaceSlovoA+ladiaceSlovoB;

   maska32b = 0b00000000000000000000000000000001;

   for (i=0; i<32; i++){
     if ((ladiaceSlovo & maska32b)>0){
       DDS_D7=1;
     }
     else{
       DDS_D7=0;
     }
     Delay_us(500);
     DDS_WCLK=1;
     Delay_us(500);
     DDS_WCLK=0;
     maska32b = maska32b << 1 ;
     Delay_us(500);
   }
   maska8b = 0b00000001;
   for (i=0; i<8; i++){
     if ((fazaRiadenie & maska8b)>0){
       DDS_D7=1;
     }
     else{
       DDS_D7=0;
     }
     Delay_us(500);
     DDS_WCLK=1;
     Delay_us(500);
     DDS_WCLK=0;
     maska8b = maska8b << 1 ;
     Delay_us(500);
   }
   Delay_us(500);
   DDS_FQUD=1;
   Delay_us(500);
   DDS_FQUD=0;

}

void inicializaciaDDS(){ //sériová komunikácia sa inicializuje kladným pulzom
   DDS_RESET=1;         //signálov v tomto poradí RESET,WCLK a FQUD
   Delay_us(500);
   DDS_RESET=0;
   Delay_us(500);
```

```
  DDS_WCLK=1;
  Delay_us(500);
  DDS_WCLK=0;
  Delay_us(500);

  DDS_FQUD=1;
  Delay_us(500);
  DDS_FQUD=0;
  Delay_us(500);
 }


void interrupt() {
    INTCON.GIE=0;
    if (INTCON.INT0IF) {
      if (nastavenieFrekvencie){
        if (ROT_KODER_B == 1){
          nastavenaFreq=nastavenaFreq+velkostKroku;
            if(nastavenaFreq>70000000){
              nastavenaFreq=70000000;
            }
          }
        else{
          nastavenaFreq=nastavenaFreq-velkostKroku;
            if(nastavenaFreq<0){
              nastavenaFreq=nastavenaFreq+velkostKroku;
            }
        }
            zmenaFreq=True;
      }
      else{
        if (ROT_KODER_B == 1){
          krok=krok+1;
          if(krok>8){
            krok=1;
          }
        }
        else{
          krok=krok-1;
            if(krok<1){
              krok=8;
            }
        }
      }
      zmenaLCD=True;
      INTCON.INT0IF = 0;
    }
  INTCON.GIE=1;
}

void main(){
  RCON.IPEN=0;                    //Zakázaná priorita prerušení
  INTCON2.INTEDG0=1;              //Prerušenie INT0 s nábežnou hranou
  INTCON.INT0IE=1;               //Prerušenie od pinu INT0 povolené
  INTCON.INT0IF=0;               //Reset príznaku prerušenia INT0
  INTCON.PEIE=1;                 //Externé prerušenia povolené
  INTCON.GIE=1;                  //Prerušenia globálne povolené

  TRISB.F0 = 1;                 //pin RB0 je vstup, RotKodA
  TRISB.F1 = 1;                 //pin RB1 je vstup, RotKodB
  TRISB.F2 = 1;                 //pin RB2 je vstup, RotKodTl

  TRISB.F3 = 0;                 //pin RB3 je výstup, DDS_D7
  TRISB.F4 = 0;                 //pin RB4 je výstup, DDS_FQUD
```

```
 TRISB.F5 = 0;                //pin RB5 je výstup, DDS_WCLK
 TRISA.F5 = 0;                //pin RA5 je výstup, DDS_RESET

 inicializaciaDDS();
 Lcd_Init();
 UART1_Init(57600);
 Delay_ms(100);

 zmenaLCD=True;
 zmenaFreq=True;
 nastavenieFrekvencie=True;       //na začiatku sa pri točení kóderom bude
                                  //meniť frekvencia, nie krok

 while(1) {       //Hlavný cyklus
  if(zmenaLCD == True){         //V prípade, že sa zmenil obsah LCD displeja
    zapisLCD();                 //aktualizuje sa
    zmenaLCD = False;
  }
  if(zmenaFreq == True){        //V prípade, že sa zmenila frekvencia
    vypocetKroku();             //aktualizuje sa
    zapisDDS();
    zmenaFreq = False;
  }

  if(ROT_KODER_TL == 1){
    if(nastavenieFrekvencie==True){
      nastavenieFrekvencie=False;
    }
    else{
      nastavenieFrekvencie=True;
    }
    while(ROT_KODER_TL == 1){    //po dobu držania tlačítka sa nič nedeje
    }
  }
  if (UART1_Data_Ready()) {              // Pokiaľ sú prijaté nejaké dáta
   UART1_Read_Text(Output,"*", 9);  // Ak sa skupina maximálne 9 znakov končí *
   nastavenaFreq=atol(Output);          // nahraď týmito znakmi nastavenú frekvenciu
   vypocetKroku();
   zapisDDS();
   zapisLCD();
  }
 }
}
```

Príloha C

```
namespace DDS
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            this.nastavTextBox = new System.Windows.Forms.TextBox();
            this.nastavButton = new System.Windows.Forms.Button();
            this.groupBox2 = new System.Windows.Forms.GroupBox();
            this.rozmStartButton = new System.Windows.Forms.Button();
            this.rozmStopButton = new System.Windows.Forms.Button();
            this.label11 = new System.Windows.Forms.Label();
            this.label9 = new System.Windows.Forms.Label();
            this.label8 = new System.Windows.Forms.Label();
            this.label7 = new System.Windows.Forms.Label();
            this.label6 = new System.Windows.Forms.Label();
            this.celkovyTextBox = new System.Windows.Forms.TextBox();
            this.prirastokTextBox = new System.Windows.Forms.TextBox();
            this.label5 = new System.Windows.Forms.Label();
            this.label4 = new System.Windows.Forms.Label();
            this.label3 = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.label1 = new System.Windows.Forms.Label();
            this.krokTextBox = new System.Windows.Forms.TextBox();
            this.koniecTextBox = new System.Windows.Forms.TextBox();
            this.pociatokTextBox = new System.Windows.Forms.TextBox();
            this.groupBox3 = new System.Windows.Forms.GroupBox();
            this.serialStartButton = new System.Windows.Forms.Button();
            this.label10 = new System.Windows.Forms.Label();
            this.serialStopButton = new System.Windows.Forms.Button();
            this.serialSelectComboBox = new System.Windows.Forms.ComboBox();
            this.groupBox4 = new System.Windows.Forms.GroupBox();
            this.stavLabel = new System.Windows.Forms.Label();
            this.groupBox1.SuspendLayout();
            this.groupBox2.SuspendLayout();
```

```
            this.groupBox3.SuspendLayout();
            this.groupBox4.SuspendLayout();
            this.SuspendLayout();
            //
            // groupBox1
            //
            this.groupBox1.Controls.Add(this.nastavTextBox);
            this.groupBox1.Controls.Add(this.nastavButton);
            this.groupBox1.Location = new System.Drawing.Point(12, 12);
            this.groupBox1.Name = "groupBox1";
            this.groupBox1.Size = new System.Drawing.Size(591, 136);
            this.groupBox1.TabIndex = 0;
            this.groupBox1.TabStop = false;
            this.groupBox1.Text = "Nastavenie frekvencie";
            //
            // nastavTextBox
            //
            this.nastavTextBox.Enabled = false;
            this.nastavTextBox.Font = new System.Drawing.Font("Microsoft Sans
Serif", 60F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(238)));
            this.nastavTextBox.Location = new System.Drawing.Point(83, 19);
            this.nastavTextBox.MaxLength = 8;
            this.nastavTextBox.Name = "nastavTextBox";
            this.nastavTextBox.Size = new System.Drawing.Size(384, 98);
            this.nastavTextBox.TabIndex = 1;
            this.nastavTextBox.Text = "14200000";
            this.nastavTextBox.KeyDown += new
System.Windows.Forms.KeyEventHandler(this.nastavTextBox_KeyDown);
            //
            // nastavButton
            //
            this.nastavButton.Enabled = false;
            this.nastavButton.Location = new System.Drawing.Point(473, 19);
            this.nastavButton.Name = "nastavButton";
            this.nastavButton.Size = new System.Drawing.Size(112, 98);
            this.nastavButton.TabIndex = 0;
            this.nastavButton.Text = "Nastav";
            this.nastavButton.UseVisualStyleBackColor = true;
            this.nastavButton.Click += new
System.EventHandler(this.nastavButton_Click);
            //
            // groupBox2
            //
            this.groupBox2.Controls.Add(this.rozmStartButton);
            this.groupBox2.Controls.Add(this.rozmStopButton);
            this.groupBox2.Controls.Add(this.label11);
            this.groupBox2.Controls.Add(this.label9);
            this.groupBox2.Controls.Add(this.label8);
            this.groupBox2.Controls.Add(this.label7);
            this.groupBox2.Controls.Add(this.label6);
            this.groupBox2.Controls.Add(this.celkovyTextBox);
            this.groupBox2.Controls.Add(this.prirastokTextBox);
            this.groupBox2.Controls.Add(this.label5);
            this.groupBox2.Controls.Add(this.label4);
            this.groupBox2.Controls.Add(this.label3);
            this.groupBox2.Controls.Add(this.label2);
            this.groupBox2.Controls.Add(this.label1);
            this.groupBox2.Controls.Add(this.krokTextBox);
            this.groupBox2.Controls.Add(this.koniecTextBox);
            this.groupBox2.Controls.Add(this.pociatokTextBox);
            this.groupBox2.Location = new System.Drawing.Point(12, 154);
            this.groupBox2.Name = "groupBox2";
            this.groupBox2.Size = new System.Drawing.Size(585, 122);
            this.groupBox2.TabIndex = 1;
```

```
            this.groupBox2.TabStop = false;
            this.groupBox2.Text = "Rozmietanie";
            //
            // rozmStartButton
            //
            this.rozmStartButton.Enabled = false;
            this.rozmStartButton.Location = new System.Drawing.Point(423, 75);
            this.rozmStartButton.Name = "rozmStartButton";
            this.rozmStartButton.Size = new System.Drawing.Size(50, 31);
            this.rozmStartButton.TabIndex = 17;
            this.rozmStartButton.Text = "Štart";
            this.rozmStartButton.UseVisualStyleBackColor = true;
            this.rozmStartButton.Click += new
System.EventHandler(this.rozmStartButton_Click);
            //
            // rozmStopButton
            //
            this.rozmStopButton.Enabled = false;
            this.rozmStopButton.Location = new System.Drawing.Point(483, 75);
            this.rozmStopButton.Name = "rozmStopButton";
            this.rozmStopButton.Size = new System.Drawing.Size(50, 31);
            this.rozmStopButton.TabIndex = 16;
            this.rozmStopButton.Text = "Stop";
            this.rozmStopButton.UseVisualStyleBackColor = true;
            this.rozmStopButton.Click += new
System.EventHandler(this.rozmStopButton_Click);
            //
            // label11
            //
            this.label11.AutoSize = true;
            this.label11.Location = new System.Drawing.Point(199, 89);
            this.label11.Name = "label11";
            this.label11.Size = new System.Drawing.Size(20, 13);
            this.label11.TabIndex = 15;
            this.label11.Text = "ms";
            //
            // label9
            //
            this.label9.AutoSize = true;
            this.label9.Location = new System.Drawing.Point(539, 35);
            this.label9.Name = "label9";
            this.label9.Size = new System.Drawing.Size(20, 13);
            this.label9.TabIndex = 13;
            this.label9.Text = "Hz";
            //
            // label8
            //
            this.label8.AutoSize = true;
            this.label8.Location = new System.Drawing.Point(363, 89);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(12, 13);
            this.label8.TabIndex = 12;
            this.label8.Text = "s";
            //
            // label7
            //
            this.label7.AutoSize = true;
            this.label7.Location = new System.Drawing.Point(367, 35);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(20, 13);
            this.label7.TabIndex = 11;
            this.label7.Text = "Hz";
            //
            // label6
            //
```

```
            this.label6.AutoSize = true;
            this.label6.Location = new System.Drawing.Point(199, 35);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(20, 13);
            this.label6.TabIndex = 10;
            this.label6.Text = "Hz";
            //
            // celkovyTextBox
            //
            this.celkovyTextBox.Enabled = false;
            this.celkovyTextBox.Location = new System.Drawing.Point(247, 86);
            this.celkovyTextBox.Name = "celkovyTextBox";
            this.celkovyTextBox.Size = new System.Drawing.Size(110, 20);
            this.celkovyTextBox.TabIndex = 9;
            //
            // prirastokTextBox
            //
            this.prirastokTextBox.Location = new System.Drawing.Point(83, 86);
            this.prirastokTextBox.Name = "prirastokTextBox";
            this.prirastokTextBox.Size = new System.Drawing.Size(110, 20);
            this.prirastokTextBox.TabIndex = 8;
            this.prirastokTextBox.Text = "100";
            this.prirastokTextBox.Leave += new
System.EventHandler(this.prirastokTextBox_Leave);
            //
            // label5
            //
            this.label5.AutoSize = true;
            this.label5.Location = new System.Drawing.Point(273, 67);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(65, 13);
            this.label5.TabIndex = 7;
            this.label5.Text = "Celkový čas";
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.Location = new System.Drawing.Point(105, 67);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(75, 13);
            this.label4.TabIndex = 6;
            this.label4.Text = "Prírstkový čas";
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Location = new System.Drawing.Point(435, 16);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(87, 13);
            this.label3.TabIndex = 5;
            this.label3.Text = "Frekvenčný krok";
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(254, 16);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(103, 13);
            this.label2.TabIndex = 4;
            this.label2.Text = "Konečná frekvencia";
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(80, 16);
```

```
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(114, 13);
            this.label1.TabIndex = 3;
            this.label1.Text = "Počiatočná frekvencia";
            //
            // krokTextBox
            //
            this.krokTextBox.Location = new System.Drawing.Point(423, 32);
            this.krokTextBox.Name = "krokTextBox";
            this.krokTextBox.Size = new System.Drawing.Size(110, 20);
            this.krokTextBox.TabIndex = 2;
            this.krokTextBox.Text = "100";
            this.krokTextBox.Leave += new
System.EventHandler(this.krokTextBox_Leave);
            //
            // koniecTextBox
            //
            this.koniecTextBox.Location = new System.Drawing.Point(251, 32);
            this.koniecTextBox.Name = "koniecTextBox";
            this.koniecTextBox.Size = new System.Drawing.Size(110, 20);
            this.koniecTextBox.TabIndex = 1;
            this.koniecTextBox.Text = "3500000";
            this.koniecTextBox.Leave += new
System.EventHandler(this.koniecTextBox_Leave);
            //
            // pociatokTextBox
            //
            this.pociatokTextBox.Location = new System.Drawing.Point(83, 32);
            this.pociatokTextBox.Name = "pociatokTextBox";
            this.pociatokTextBox.Size = new System.Drawing.Size(110, 20);
            this.pociatokTextBox.TabIndex = 0;
            this.pociatokTextBox.Text = "1000000";
            this.pociatokTextBox.Leave += new
System.EventHandler(this.pociatokTextBox_Leave);
            //
            // groupBox3
            //
            this.groupBox3.Controls.Add(this.serialStartButton);
            this.groupBox3.Controls.Add(this.label10);
            this.groupBox3.Controls.Add(this.serialStopButton);
            this.groupBox3.Controls.Add(this.serialSelectComboBox);
            this.groupBox3.Location = new System.Drawing.Point(12, 282);
            this.groupBox3.Name = "groupBox3";
            this.groupBox3.Size = new System.Drawing.Size(396, 50);
            this.groupBox3.TabIndex = 2;
            this.groupBox3.TabStop = false;
            this.groupBox3.Text = "Nastavenie komunikácie";
            //
            // serialStartButton
            //
            this.serialStartButton.Location = new System.Drawing.Point(264, 13);
            this.serialStartButton.Name = "serialStartButton";
            this.serialStartButton.Size = new System.Drawing.Size(50, 31);
            this.serialStartButton.TabIndex = 19;
            this.serialStartButton.Text = "Pripojiť";
            this.serialStartButton.UseVisualStyleBackColor = true;
            this.serialStartButton.Click += new
System.EventHandler(this.serialStartButton_Click);
            //
            // label10
            //
            this.label10.AutoSize = true;
            this.label10.Location = new System.Drawing.Point(128, 22);
            this.label10.Name = "label10";
            this.label10.Size = new System.Drawing.Size(29, 13);
```

```
            this.label10.TabIndex = 18;
            this.label10.Text = "Port:";
            //
            // serialStopButton
            //
            this.serialStopButton.Enabled = false;
            this.serialStopButton.Location = new System.Drawing.Point(329, 13);
            this.serialStopButton.Name = "serialStopButton";
            this.serialStopButton.Size = new System.Drawing.Size(50, 31);
            this.serialStopButton.TabIndex = 18;
            this.serialStopButton.Text = "Odpojiť";
            this.serialStopButton.UseVisualStyleBackColor = true;
            this.serialStopButton.Click += new
System.EventHandler(this.serialStopButton_Click);
            //
            // serialSelectComboBox
            //
            this.serialSelectComboBox.FormattingEnabled = true;
            this.serialSelectComboBox.Location = new System.Drawing.Point(163,
19);
            this.serialSelectComboBox.Name = "serialSelectComboBox";
            this.serialSelectComboBox.Size = new System.Drawing.Size(78, 21);
            this.serialSelectComboBox.TabIndex = 0;
            this.serialSelectComboBox.Click += new
System.EventHandler(this.serialSelectComboBox_Click);
            //
            // groupBox4
            //
            this.groupBox4.Controls.Add(this.stavLabel);
            this.groupBox4.Location = new System.Drawing.Point(414, 282);
            this.groupBox4.Name = "groupBox4";
            this.groupBox4.Size = new System.Drawing.Size(183, 50);
            this.groupBox4.TabIndex = 3;
            this.groupBox4.TabStop = false;
            this.groupBox4.Text = "Stav";
            //
            // stavLabel
            //
            this.stavLabel.AutoSize = true;
            this.stavLabel.Location = new System.Drawing.Point(24, 22);
            this.stavLabel.Name = "stavLabel";
            this.stavLabel.Size = new System.Drawing.Size(64, 13);
            this.stavLabel.TabIndex = 0;
            this.stavLabel.Text = "Nepripojené";
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(615, 342);
            this.Controls.Add(this.groupBox4);
            this.Controls.Add(this.groupBox3);
            this.Controls.Add(this.groupBox2);
            this.Controls.Add(this.groupBox1);
            this.Name = "Form1";
            this.Text = "Ovládanie DDS generátora";
            this.FormClosing += new
System.Windows.Forms.FormClosingEventHandler(this.Form1_FormClosing);
            this.Load += new System.EventHandler(this.Form1_Load);
            this.groupBox1.ResumeLayout(false);
            this.groupBox1.PerformLayout();
            this.groupBox2.ResumeLayout(false);
            this.groupBox2.PerformLayout();
            this.groupBox3.ResumeLayout(false);
            this.groupBox3.PerformLayout();
```

```
            this.groupBox4.ResumeLayout(false);
            this.groupBox4.PerformLayout();
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.TextBox nastavTextBox;
        private System.Windows.Forms.Button nastavButton;
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.TextBox celkovyTextBox;
        private System.Windows.Forms.TextBox prirastokTextBox;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox krokTextBox;
        private System.Windows.Forms.TextBox koniecTextBox;
        private System.Windows.Forms.TextBox pociatokTextBox;
        private System.Windows.Forms.Label label11;
        private System.Windows.Forms.Label label9;
        private System.Windows.Forms.Label label8;
        private System.Windows.Forms.Label label7;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.Button rozmStartButton;
        private System.Windows.Forms.Button rozmStopButton;
        private System.Windows.Forms.GroupBox groupBox3;
        private System.Windows.Forms.Button serialStartButton;
        private System.Windows.Forms.Label label10;
        private System.Windows.Forms.Button serialStopButton;
        private System.Windows.Forms.ComboBox serialSelectComboBox;
        private System.Windows.Forms.GroupBox groupBox4;
        private System.Windows.Forms.Label stavLabel;
    }
}
```

Príloha D

```
using System;
using System.ComponentModel;
using System.IO.Ports;
using System.Threading;
using System.Windows.Forms;

namespace DDS
{
    public partial class Form1 : Form
    {
        static SerialPort port;
        static BackgroundWorker backgroundWorker=new BackgroundWorker();


        public Form1()
        {
            InitializeComponent();
            backgroundWorker.DoWork += Rozmietanie;
        }


        private void serialStartButton_Click(object sender, EventArgs e)
        {
            if (serialSelectComboBox.SelectedIndex > -1)
            {
                Connect(serialSelectComboBox.SelectedItem.ToString());
                if (port.IsOpen)
                {
                    serialStartButton.Enabled = false;
                    serialStopButton.Enabled = true;
                    nastavButton.Enabled = true;
                    rozmStartButton.Enabled = true;
                    nastavTextBox.Enabled = true;
                    stavLabel.Text = "Úspešne pripojené k " +
serialSelectComboBox.SelectedItem;
                }
            }
            else
            {
                MessageBox.Show("Port nebol zvolený!");
                stavLabel.Text = "Port nebol zvolený!";
            }
        }

        private void Connect(string portName)
        {
            port = new SerialPort(portName);
            if (!port.IsOpen)
            {
                port.BaudRate = 57600;

                try
                {
                    port.Open();
                }
                catch
                {
                    stavLabel.Text = "Nepodarilo sa pripojiť";
                }
              }
        }
```

```
        private void serialStopButton_Click(object sender, EventArgs e)
        {
            if (backgroundWorker.IsBusy)
            {
                backgroundWorker.CancelAsync();
            }

            if (port.IsOpen)
            {
                port.Close();
                serialStartButton.Enabled = true;
                serialStopButton.Enabled = false;
                nastavButton.Enabled = false;
                rozmStartButton.Enabled = false;
                rozmStopButton.Enabled = false;
                nastavTextBox.Enabled = false;
                stavLabel.Text = "Port odpojený";
            }

        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (backgroundWorker.IsBusy)
            {
                backgroundWorker.CancelAsync();
            }
            if (port != null)
            {
                if (port.IsOpen)
                {
                    port.Close();
                }
            }

        }

        private void Form1_Load(object sender, EventArgs e)
        {
            NaplnenieComboBoxu();
        }

        private void NaplnenieComboBoxu()
        {
            var ports = SerialPort.GetPortNames();
            serialSelectComboBox.DataSource = ports;
        }

        private void nastavButton_Click(object sender, EventArgs e)
        {
            int frekvencia;

            try
            {
                frekvencia = Convert.ToInt32(nastavTextBox.Text);
                if (frekvencia < 70000000 && frekvencia > 0)
                {
                    String sprava = Convert.ToString(frekvencia) + "*";
                    port.Write(sprava);
                    stavLabel.Text = "Nastavené " + Convert.ToString(frekvencia)
+ " Hz";
                }
                else
                {
```

```
                stavLabel.Text = "Hodnota mimo rozsah";
            }
        }
        catch (FormatException)
        {
            stavLabel.Text = "Neplatné znaky";
        }
        catch
        {
            stavLabel.Text = "Neznáma chyba";
        }

    }

    private void nastavTextBox_KeyDown(object sender, KeyEventArgs e)
    {
        if (e.KeyCode == Keys.Enter)
        {
            nastavButton_Click(this, new EventArgs());
        }
    }

    private bool KontrolaRozmietania()
    {

        try
        {
            int pociatocna = Convert.ToInt32(pociatokTextBox.Text);
            int konecna = Convert.ToInt32(koniecTextBox.Text);
            int krok = Convert.ToInt32(krokTextBox.Text);
            int delta = Convert.ToInt32(prirastokTextBox.Text);

            if (!(pociatocna < 70000000 && pociatocna > -1))
            {
                stavLabel.Text = "Počiatočná frekv. mimo rozsah";
                return false;
            }

            if (!(konecna < 70000001 && konecna > 0 && konecna>pociatocna))
            {
                stavLabel.Text = "Konečná frekv. mimo rozsah";
                return false;
            }
            if (!(krok > -1 && krok<(konecna-pociatocna)))
            {
                stavLabel.Text = "Nesprávna hodnota kroku";
                return false;
            }
            if (!(delta > 99))
            {
                stavLabel.Text = "Najmenší prírastok je 100 ms!";
                return false;
            }
            return true;

        }
        catch
        {
            stavLabel.Text = "Neplatné znaky";
            return false;
        }

    }

    private void pociatokTextBox_Leave(object sender, EventArgs e)
```

```
        {
            VypocetCelkovehoCasu();
        }

        private void koniecTextBox_Leave(object sender, EventArgs e)
        {
            VypocetCelkovehoCasu();
        }

        private void krokTextBox_Leave(object sender, EventArgs e)
        {
            VypocetCelkovehoCasu();
        }

        private void prirastokTextBox_Leave(object sender, EventArgs e)
        {
            VypocetCelkovehoCasu();
        }

        private void VypocetCelkovehoCasu()
        {
            try
            {
                int pociatocna = Convert.ToInt32(pociatokTextBox.Text);
                int konecna = Convert.ToInt32(koniecTextBox.Text);
                int krok = Convert.ToInt32(krokTextBox.Text);
                int delta = Convert.ToInt32(prirastokTextBox.Text);
                celkovyTextBox.Text = String.Format("{0:0.##}", (konecna -
pociatocna) / krok * delta / 1000);
            }
            catch
            {

            }
        }

        private void Rozmietanie (object sender,
System.ComponentModel.DoWorkEventArgs e)
        {

                int pociatocna = Convert.ToInt32(pociatokTextBox.Text);
                int konecna = Convert.ToInt32(koniecTextBox.Text);
                int krok = Convert.ToInt32(krokTextBox.Text);
                int delta = Convert.ToInt32(prirastokTextBox.Text);

                while(true)
                {
                    int aktualna = pociatocna;
                    while (aktualna <= konecna)
                    {
                        try
                        {
                            String sprava = Convert.ToString(aktualna) + "*";
                            port.Write(sprava);
                            aktualna = aktualna + krok;
                            Thread.Sleep(delta);
                            if (delta > 300)
                            {
                                stavLabel.Text = aktualna.ToString();
                            }
                            if (backgroundWorker.CancellationPending)
                            {
                                e.Cancel = true;
                                return;
                            }
```

```
                    }
                    catch
                    {
                    }


                }
            }

        }

        private void rozmStartButton_Click(object sender, EventArgs e)
        {
            if (KontrolaRozmietania())
            {
                backgroundWorker.RunWorkerAsync();
                backgroundWorker.WorkerSupportsCancellation = true;
                stavLabel.Text = "Rozmietanie spustené";
                rozmStartButton.Enabled = false;
                rozmStopButton.Enabled = true;
            }
        }

        private void rozmStopButton_Click(object sender, EventArgs e)
        {
            if (backgroundWorker.IsBusy)
            {
                backgroundWorker.CancelAsync();
            }
            rozmStartButton.Enabled = true;
            rozmStopButton.Enabled = false;

        }

        private void serialSelectComboBox_Click(object sender, EventArgs e)
        {
            NaplnenieComboBoxu();
        }

    }
}
```