UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY
A INFORMATIKY

DIPLOMOVÁ PRÁCE

2011 Martin Lauterbach

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Sběr dat z frekvenčně modulovaného radaru a jejich 3D zobrazení

Autor práce: Martin Lauterbach

Vedoucí práce: Ing. Richard Capalini, CSc.

Diplomová práce

2011

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2010/2011**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Martin LAUTERBACH**
Osobní číslo:	**I09403**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Komunikační a řídicí technologie**
Název tématu:	**Sběr dat z frekvenčně modulovaného radaru a jejich 3D zobrazení**
Zadávající katedra:	**Katedra elektrotechniky**

Zásady pro vypracování:

Cílem práce je vytvořit program pro sběr dat z frekvenčně modulovaného radaru umístěného v tzv. demonstrátoru [1] v reálném čase a jejich zobrazení v trojrozměrném prostoru [X, Y, τ] na PC. X a Y jsou souřadnice polohy radaru v demonstrátoru a τ je zpoždění signálu odraženého od cíle. Pomocí odrazů od přesně definovaného cíle bude dále vytvořena trojrozměrná směrová charakteristika radaru. Protože systém sám vytváří mnoho vlastních odrazů signálu, je třeba vytvořit vhodnou kalibrační proceduru pro jejich odstranění. Osnova práce: Popis signálu frekvenčně modulovaného radaru. Popis demonstrátoru a systému sběru dat z frekvenčně modulovaného radaru. Výběr vhodné kalibrační procedury pro odstranění vlastních odrazů systému. Vytvoření trojrozměrné směrové charakteristiky radaru z naměřených dat v demonstrátoru. Zobrazení naměřených dat před aplikací a po aplikaci kalibrační procedury.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] Capalini, R.: Další možnosti využití UWB senzoru. STEINEL Technik, 2010.

[2] Kohl, Ch. - Krause, M. - Majerhofer, Ch. - Wöstmann, J. - Wiggenhauser, H.: 3D-Visualisation of NDT-Data using Data Fusion Technique. NDT-CE 2003.

[3] Lualdi, M. - Zanzi, L. - Binda, L.: Acquisition and processing requirements for high quality 3D reconstruktions from GPR investigations. NDT-CE 2003

[4] Lambot, S. - Van den Bosch, I. - Slob, C.: Frequency domain GPR signal forvard and inverse modelling for identifying the subsurface dielectric properties.

Vedoucí diplomové práce: **Ing. Richard Capalini, CSc.**
Steinel Technik Pardubice

Datum zadání diplomové práce: **3. listopadu 2010**

Termín odevzdání diplomové práce: **20. května 2011**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Zdeněk Němec, Ph.D.
vedoucí katedry

V Pardubicích dne 12. listopadu 2010

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Kolíně dne 28. srpna 2011

Martin Lauterbach

Poděkování:

Tímto bych rád poděkoval vedoucímu práce panu Ing. Richardu Capalini, CSc. za odborné vedení a rady v průběhu práce.

Dále bych chtěl poděkovat firmě Steinel Technik, kde jsem diplomovou práci realizoval.

Velký dík bych chtěl vyjádřit mé rodině a všem, kteří mě podporovali po celou dobu studia.

Souhrn

Tato diplomová práce se zabývá sběrem dat z frekvenčně modulovaného radaru. Shromážděná data jsou zde použita k ověření kalibračních technik a k sestavení trojrozměrné směrové charakteristiky. Dále je tento diagram zobrazen v trojrozměrném prostoru s využitím vlastního softwaru. Jsou zde nastíněny techniky zobrazazování ve 3D s využitím hardwarové akcelerace.

Klíčová slova

radar, FMCW, OpenGL, blízká zóna, vyzařování antény, sníh, GPR, UWB

Abstract

This master's thesis deals with data acquisition from frequency-modulated radar. Those data are used to verify the calibration techniques and to build three-dimensional directional characteristics of the radar. The diagram is shown in 3D space using own software. At the last, there are outlined techniques of 3D visualisation with use of hardware acceleration.

Keywords

radar, FMCW, OpenGL, near field, antenna propagation, snow, GPR, UWB

Obsah

Seznam obrázků

Seznam zkratek

API	*Application Programming Interface*, aplikační programové rozhraní
CAN	*Controller Area Network*, sběrnice pro automobilní průmysl
DFT	*Discrete Fourier Transform*, diskrétní Fourierova transformace
DDS	*Direct Digital Synthesis*, digitální syntéza kmitočtu
DSP	*Digital Signal Processor*, digitální signálový procesor
FFT	*Fast Fourier Transform*, rychlá Fourierova transformace
FMCW	*Frequency Modulated Continuous-wave*, systém s lineární frekvenční modulací spojité vlny
GPR	*Ground-Penetrating Radar*, radar sloužící k průzkumu prostoru pod zemským povrchem
GUI	*Graphical User Interface*, grafické uživatelské rozhraní
IP	*Internet Protocol*, datový protokol používaný pro přenos dat přes paketové sítě
LFM	*Linear Frequency Waveform*, lineární frekvenční modulace
PLL	*Phase-Locked Loop*, fázový závěs
SFW	*Stef Frequency Waveform*, vlna se změnou frekvence po krocích
SIMD	*Single Instruction, Multiple Data*, architektura procesoru s paralelním zpracováním dat během jedné instrukce
UART	*Universal Asynchronous Receiver/Transmitter*, asynchronní seriové rozhraní
UDP	*User Datagram Protocol*, protokol transportní vrstvy
USB	*Universal Serial Bus*, univerzální sériová sběrnice
UWB	*Ultra-Wide band*, systém s velkou šířkou pásma

Kapitola 1

Úvod

Firma Steinel Technik již delší dobu pracuje na projektu frekvenčně modulovaného radaru pro měření hloubky sněhu. Tento projekt je svým rozsahem ve firmě ojedinělý, ztvárňuje podporu výzkumu a nasazení nových technologií.

Požadavkem zákazníka zde bylo vytvořit zařízení, které by se namontovalo na sněžnou rolbu a poskytovalo rolbaři informace o hloubce sněhu pod rolbou. Toto zařízení musí splňovat náročné požadavky v provozu a spadá do kategorie zařízení pro průmysl. Jsou zde kladeny požadavky na práci ve velkém rozsahu teplot a odolnost proti výkyvům napájení. Důležitým požadavkem je i velikost a celé zařízení se musí vejít do prostoru krychle o hraně přibližně čtyřicet centimetrů.

Uvedený radar je formou GPR. Rozdíl od běžných GPR je v tom, že sníh může dosahovat velké vlhkosti a tím s rostoucí frekvencí značného útlumu šířícího se signálu.

Tato diplomová práce byla vypsaná ve stádiu, kdy již radar pracuje a je možné pro GPR běžným způsobem pořizovat snímky prostoru. Ty jsou však pro obsluhu rolby nepřijatelné a požadavek je pouze na číselnou hodnotu hloubky sněhu. Tento systém číselné informace funguje v příhodných podmínkách, ale pokud se ve sněhu vyskytnou nespojitosti v podobě vzduchových trhlin nebo kamenů, stane se, že jsou pro radar dominantním cílem ve srovnání s rozhraním sníh-sníh nebo sníh-podklad. V těchto případech je měření zkresleno a rolbař dostává špatnou informaci.

Úkolem práce je zajistit kalibrační údaje, aby bylo možné nespojitosti z výpočtů vyloučit. Jedná se především o sestavení charakteristiky antény a s tím spojené operace.

Oblast zobrazení anténní charakteristiky ve 3D by měla být přípravou na další využití radaru jako scanneru třídimenzionálního prostoru pod zemským povrchem.

Kapitola 2

Radar pro měření hloubky sněhu

Požadavek na vývoj radaru pro měření hloubky sněhu vznikl z více důvodů. První požadovanou funkcí, která má velký ekonomický význam, je samotné měření tloušťky sněhové vrstvy. Na základě přesných znalostí o vrstvě sněhové pokrývky lze na lyžařských svazích efektivněji uplatňovat nákladné umělé zasněžování.

Dalším požadavkem je odhalení trhlin, nehomogenit a průrev na ledovci. Tím lze předcházet lavinovému nebezpečí.

Řešení problému měření hloubky sněhu můžeme definovat jako měření vzdálenosti dvou rozhraní různých materiálů [9]. První rozhraní je rozhraní vzduch-sníh, druhé rozhraní je sníh-zem.

Měřicí zařízení musí nejen obě vrstvy rozlišit, ale musí také umět pronikat vrstvou sněhu o tloušťce až 3 m.

Pro měření hloubky sněhu byl vyroben dále popsaný radar s frekvenční modulací vlny. Použitý radar je formou GPR, tedy radaru sloužícího k podpovrchovému průzkumu.

2.1 Radary k podpovrchovému průzkumu

Pro radary sloužící k průzkumu pod povrchem, ať už země nebo jiným, se používá označení GPR (ground penetrating radar).

Jedná se o radary využívající nedestruktivního průzkumu pomocí elektromagnetických vln v mikrovlném pásmu.

Využívá se zde odrazu od rozhraní s různou dielektrickou konstantou.

Dosah radaru je ovlivněn útlumem v prostředí, které je prozkoumáváno. Na tento útlum má zásadní vliv vodivost zmiňovaného prostředí.

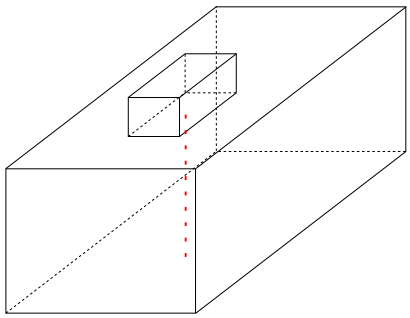
2.1.1 GPR zobrazení

Ve světě GPR je běžných několik typů zobrazení signálu. Tato zobrazení nesou označení A, B a C. Jejich charakter je následující.

Typ A

Zobrazení typu A představuje zobrazení amplitudy signálu v závislosti na jeho frekvenci. Toto zobrazení lze sestavit z jediného odběhu FMCW radaru Fourierovou transformací signálu.

Na ose X je zpoždění τ, na ose Y amplituda signálu.



Obrázek 2.1: Řez A

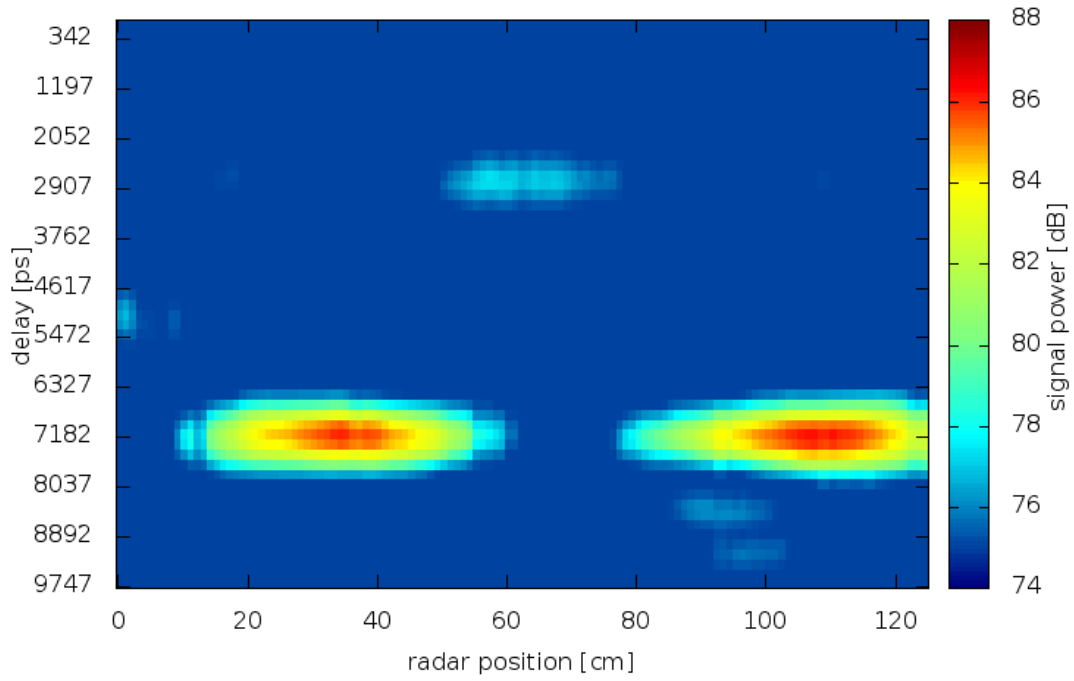
Typ B

Zobrazení typu B obsahuje řadu zobrazení typu A. Zobrazení je nad množinou dvou proměnných. Jednou proměnou je frekvence odpovídající zpoždění τ. Druhou je po-

loha radaru. Toto je nejobvyklejší zobrazení pro GPR. K jeho vytvoření je zapotřebí větší množství odběhů.

Na jedné ose je poloha radaru a na druhé zpoždění τ. Barva představuje amplitudu signálu.

Zkušená obsluha radaru je schopna v tomto zobrazení vyčíst potřebné informace.

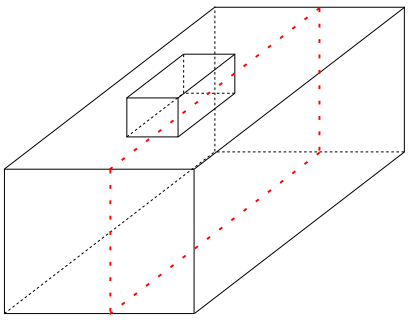


Obrázek 2.2: Zobrazení typu B

Typ C

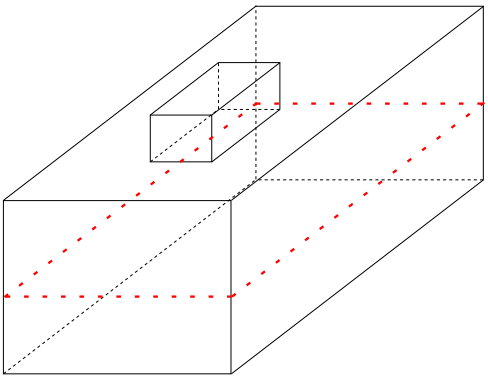
Zobrazení typu C lze považovat za řez prostorem v rovině rovnoběžné s povrchem, nad kterým probíhá snímání.

V našem případě je vytvářeno pohybem radaru ve dvou osách, ale je možné toto zobrazení vytvořit s využitím anténní řady, která zajistí snímání v rovině.

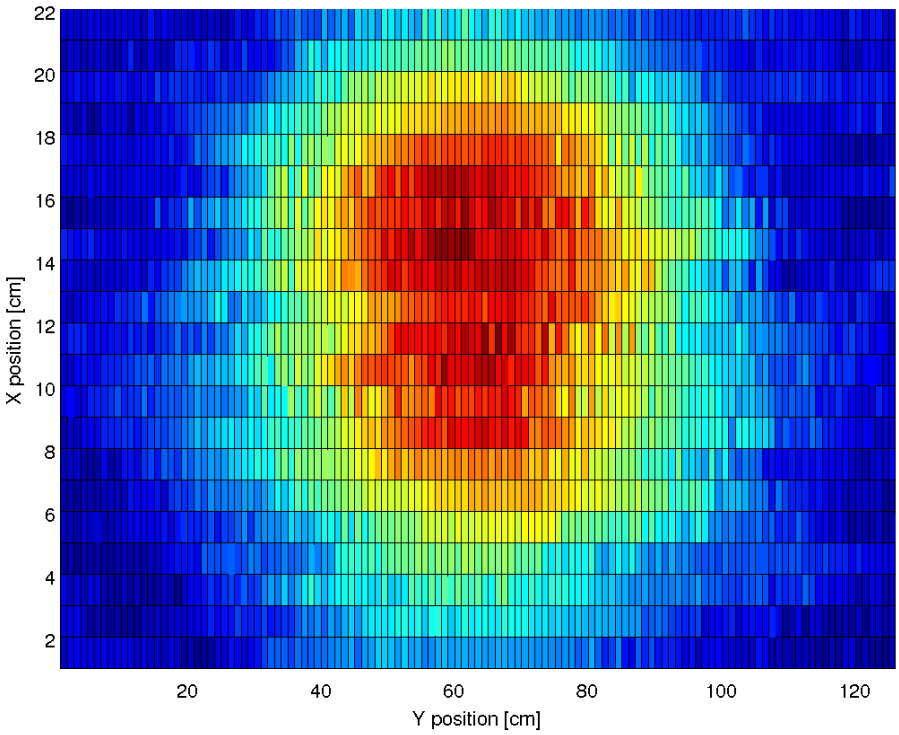


Obrázek 2.3: Řez B

V celém spektrogramu je konstantní τ. Na osách je poloha radaru a barva představuje amplitudu signálu.



Obrázek 2.4: Řez C



Obrázek 2.5: Zobrazení C, kalibrační koule uprostřed demonstrátoru

2.2 Konstrukce našeho radaru

Radar pro měření hloubky sněhu pracuje s šířkou pásma 730 MHz začínající na frekvenci 230 MHz. Takové parametry umožňují systém charakterizovat jako UWB, protože šířka pásma je více jak 300%.

Pro zajištění plné koherence je vysokofrekvenční signál generovaný s pomocí DDS a vše vychází z jediného velmi přesného oscilátoru.

Složité výpočty řeší signálový procesor s dostatkem operační paměti.

Fungování radaru musí být zajištěno v nepříznivých podmínkách. Je umístěn naspodu rolby pod motorem. Shora tedy působí vysoké teploty a pod radarem je mokrý sníh. To klade vysoké nároky na konstrukci, aby odolávala vlhku a velkým rozdílům teplot. Zároveň musí radar odolávat i extrémním vibracím.

Funkce elektronických obvodů v radarech je velmi závislá na teplotě. Radar je z těchto důvodů vyhříván přibližně na teplotu 55°C. To je teplota, kterou lze v náročných podmínkách stabilizovat vyhříváním.

Elektronické uspořádání radaru je patrné z obrázku 2.6. Vybrané bloky jsou popsány dále.

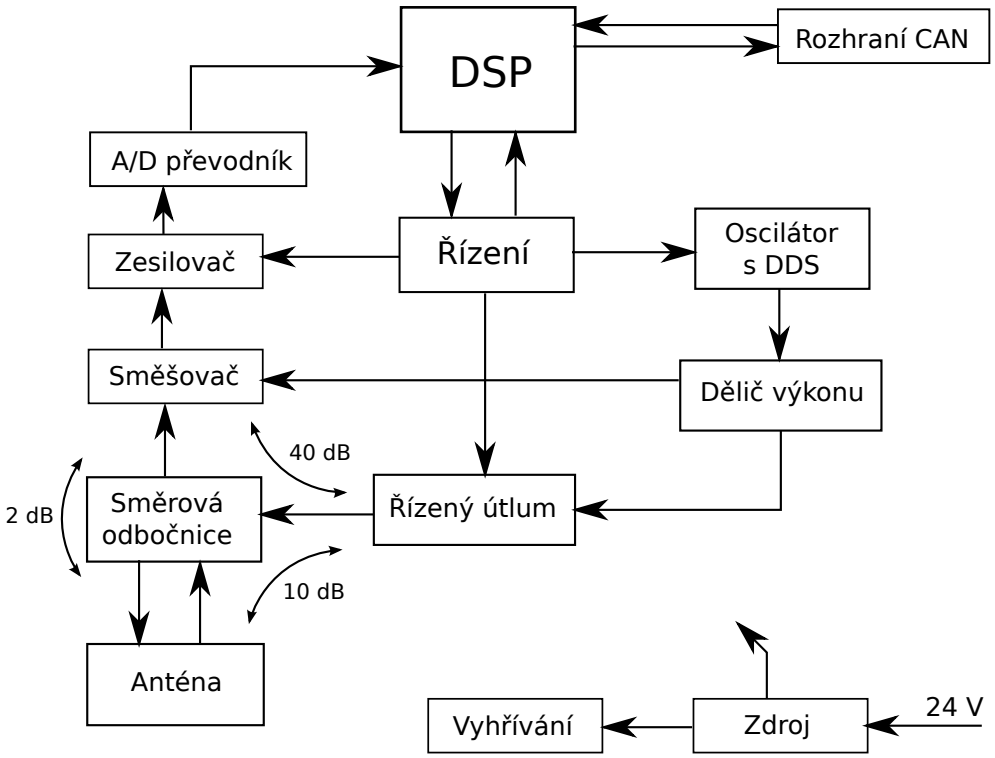
2.2.1 Signálový procesor

Výpočet hloubky sněhu klade velké požadavky na výpočetní výkon. Je třeba řešit rozsáhlé soustavy lineárních rovnic. Počet hledaných proměnných jsou desítky až stovky.

Aby bylo možné řešit tak velké soustavy rovnic, musí procesor poskytovat možnost pro připojení velké paměti.

Uvedeným požadavkům vyhověl procesor Analog Devices ADSP-21369. Jeho parametry jsou následující:

- výpočetní výkon špičkově až 2,4 GFLOPS s využitím architektury SIMD,
- taktovací kmitočet 400 MHz,
- vnitřní paměť 2Mbits SRAM,



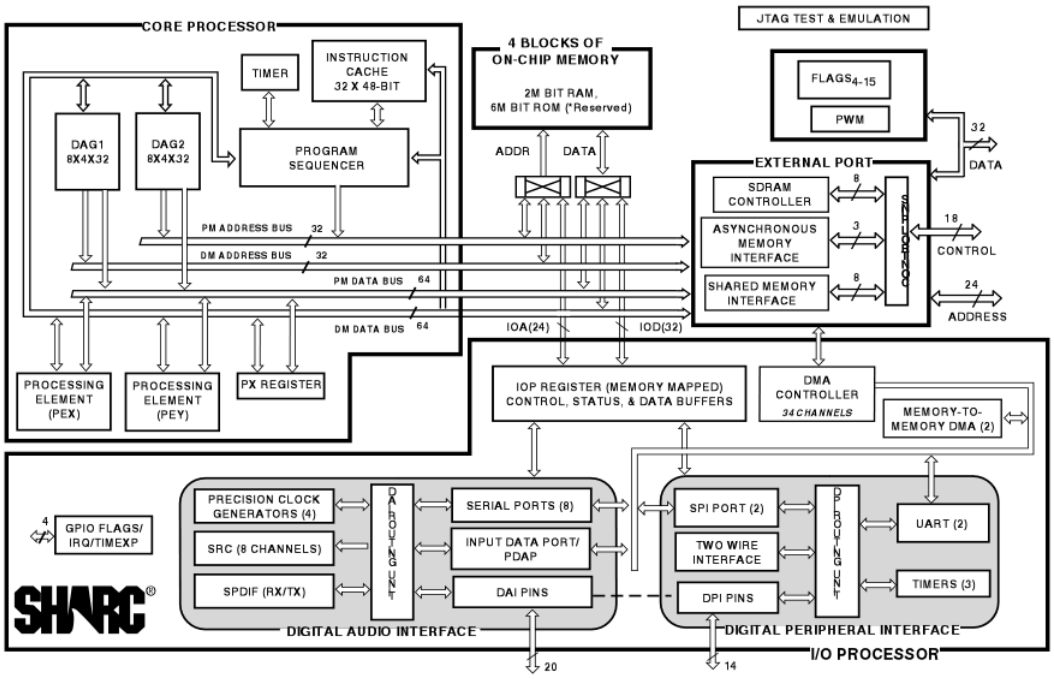
Obrázek 2.6: Blokové schéma radaru

- 32 bitová externí sběrnice pro připojení pamětí,
- 3 časovače, 2 porty SPI, 2 porty UART, 2 porty I2C.

Blokové schéma signálového procesoru je na obrázku 2.7.

2.2.2 Řízení

Aby bylo zajištěno plně synchronní řízení, byl tento úkol svěřen mimo procesor programovatelnému obvodu CPLD. K CPLD je připojen precizní oscilátor, který zajišťuje vzorkování signálu v přesně daných intervalech, aby nebyla ovlivněna fáze signálu.



Obrázek 2.7: Blokové schéma procesoru ADSP-21369 [7]

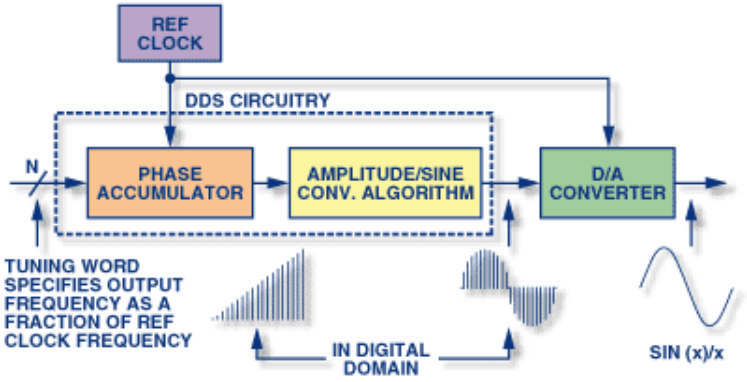
2.2.3 DDS

Použití přímé digitální syntézy kmitočtu je řešením k zajištění plné koherence systému a umožňuje pořízení vzorku vždy ve stejné fázi signálu. Vlivem použití DDS se kmitočet nosné mění nespojitě po krocích.

Na obrázku 2.8 je znázorněn princip fungování DDS. Ladícím slovem N je nastaven krok, s kterým se mění hodnota ve fázovém akumulátoru a tím i frekvence výstupního harmonického signálu. Tato hodnota představuje okamžitou fázi. Převod čísla z fázového akumulátoru může být vyřešen jako tabulka hodnot funkce sinus, kde fázový akumulátor vybírá adresu s číslem pro D/A převodník.

2.2.4 Rozhraní CAN

O zprostředkování komunikace po sběrnici CAN se stará samostatný mikrokontrolér připojený k signálovému procesoru sběrnicí SPI. Důvodem pro tuto separaci je, že použitý signálový procesor nedisponuje rozhraním CAN, zároveň není vytížen komunikací a může se plně věnovat výpočtům.

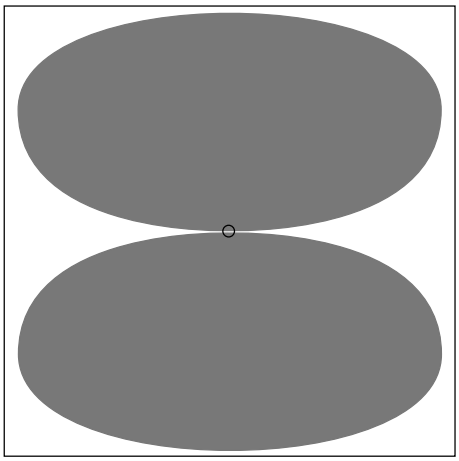


Obrázek 2.8: Blokové schéma DDS [8]

Do CAN mikrokontroléru je implementován protokol sloužící k přenesení velkých bloků dat. Tato komunikace není pro sběrnici běžná, ale zde bylo nutné tento přenos zajistit kvůli experimentům.

2.2.5 Anténa

Anténa radaru je velice specifická. Její zisk je záporný. Jedná se o plošný dipól zhotovený jako tištěný spoj obklopený útlumovou hmotou. Pro představu je načrtnuta na obrázku 2.9.



Obrázek 2.9: Anténa

2.2.6 Napájení

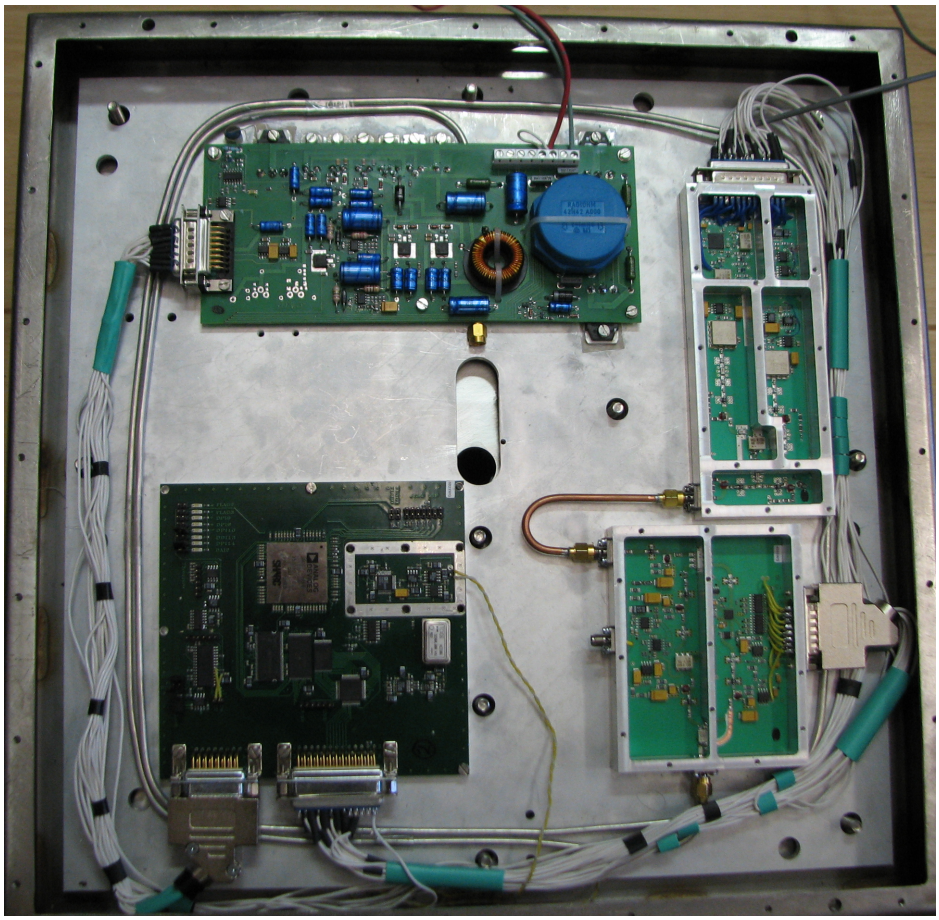
Jelikož rolba pracuje v náročných podmínkách, odpovídají tomu i nároky kladené na napájecí zdroj.

Vstupem zdroje je palubní napětí 24 V. Výstupů zdroje je několik s různými úrovněmi napětí.

Zdroj musí odolávat výkyvům napájecího napětí a zároveň musí snést i zkrat na kterémkoli výstupu. Při zkratu musí zdroj vypnout a jakmile zkrat pomine, musí se znovu uvést do činnosti.

2.2.7 Mechanické uspořádání

Konstrukce radaru je vytvořena z 2.5 mm silného nerezového ocelového plechu. Velikost odpovídá kvádru s rozměry 410 mm x 410 mm x 200 mm. V horní části je umístěna elektronika, pod ní anténa. Pohled na elektroniku je viditelný na obrázku 2.10. Spodní část radaru pod anténou je vyplněna útlumovou hmotou, aby se zúžil anténní svazek.



Obrázek 2.10: Pohled na elektroniku radaru, nahoře je viditelný zdroj, dole procesorová deska, vpravo mikrovlnná část

Kapitola 3

Frekvenčně modulovaný radar

Koncepce frekvenčně modulovaného radaru je shrnuta ve zprávách [9] a [6] odkud vychází i následující text.

Každé radarové měření je v principu založené na měření časového zpoždění τ přijatého signálu, který se odrazí od cíle, vzhledem k signálu, který je k cíli vyslán.

V použitém systému probíhají všechna měření vysíláním lineárně frekvenčně modulované vlny. Tato vlna se odrazí od cíle (rozhraní dvou různých dielektrik) a část energie se tak vrátí zpět do antény radaru. Vlivem frekvenčního rozmítání je již v tu chvíli vysílaná frekvence posunuta.

Na směšovač je přiváděn zároveň vysílaný a přijímaný signál. Vzniká zde tak mimo jiné rozdílová frekvence f_r přímo úměrná časovému zpoždění τ.

3.1 Model signálu

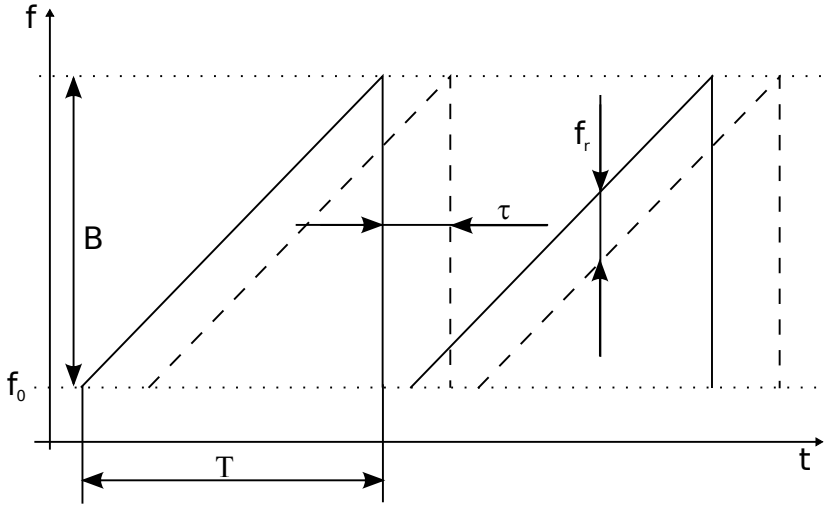
Signál s lineární frekvenční modulací mění kmitočet lineárně v závislosti na čase. Průběh znázorňuje obrázek 3.1 a matematicky ho popisují rovnice 3.1 a 3.2.

$$f(t) = f_0 + \frac{B}{T}t \quad \text{pro} \quad t \in \langle t_0; t_0 + T) \tag{3.1}$$

$f(t)$ ... frekvence vysílané vlny $[Hz]$

f_0 ... nejnižší vysílaná frekvence $[Hz]$

B ... šířka pásma $[Hz]$



Obrázek 3.1: Vysílaný a přijímaný signál FMCW radaru

T ... perioda rozmítání $[s]$

t ... čas $[s]$

$$\phi(t) = 2\pi \cdot \int f(t)dt = 2\pi \left(f_o t + \frac{B}{2T} t^2 \right) \tag{3.2}$$

$\phi(t)$ ... fáze vysílané vlny $[rad]$

Takto definovaný signál je v intervalu $\langle t_0; t_0 + T)$ popsán vztahem 3.3. Mimo uvedený interval je možné signál považovat rovný nule.

$$\begin{aligned} u_c(t) = \cos\left[2\pi\left(f_o t + \tfrac{B}{2T} t^2\right)\right] \quad &\text{pro} \quad t \in \langle t_0; t_0 + T) \\ u_c(t) = 0 \quad &\text{pro} \quad t \text{ jiná} \end{aligned} \tag{3.3}$$

$u_c(t)$ ... průběh signálu $[-]$

Signál přicházející na směšovač jako místní oscilátor je dán vztahem 3.4.

$$u_L(t) = |g_L(t)| \cdot \cos\left[2\pi\left(f_o t + \frac{B}{2T}t^2\right) + \phi_L(t)\right] \quad \text{pro} \quad t \in \langle t_0; t_0 + T) \tag{3.4}$$

$u_L(t)$ ... signál z místního oscilátoru $[-]$
$g_L(t)$ ... přenos mezi oscilátorem a směšovačem $[-]$
$\phi_L(t)$ ... fázový posun mezi oscilátorem a směšovačem $[rad]$

Přenos mezi oscilátorem a vstupem směšovače pro místní oscilátor je dán přenosovou funkcí $g_L = |g_L(t)|e^{j\phi_L(t)}$.

Toto vyjádření je možné pouze proto, že u LFM radaru existuje jednoznačný vztah mezi kmitočtem a časem. Obecně je nutno přenos popsat konvolučním integrálem.

Přenos mezi vysílačem, cílem a přijímačem je popsán časovým zpožděním τ a komplexní přenosovou funkcí $g_{TR}(t,\tau) = |g_{TR}(t,\tau)|e^{j\phi_{TR}(t,\tau)}$. Signál na vstupu směšovače se pak vyjádří vztahem 3.5.

$$\begin{aligned} &u_{TR}(t,\tau) = |g_{TR}(t,\tau)| \cdot \cos\left[2\pi\left(\left(f_o - \frac{B\tau}{T}\right)t + \frac{B}{2T}t^2 - f_0\tau + \frac{B\tau^2}{2T}\right) + \phi_{TR}(t)\right] \\ &\text{pro } t \in \langle t_0 + \tau; t_0 + T + \tau) \end{aligned} \tag{3.5}$$

$u_{TR}(t,\tau)$ ... přijatý signál $[-]$
τ ... časové zpoždění $[s]$

Signál přijatý a signál z místního oscilátoru se smísí v balančním směšovači. Na jeho výstupu se po filtraci získá rozdílový signál 3.6.

$$\begin{aligned} &u(t,\tau) = \tfrac{1}{2} \cdot |g_{TR}(t,\tau)| \cdot |g_L(t)| \cdot \cos\left[2\pi\left(\frac{B\tau}{T}t + f_0\tau - \frac{B\tau^2}{2T}\right) + \phi_L t - \phi_{TR}(t)\right] \\ &\text{pro } t \in \langle t_0 + \tau; t_0 + T) \end{aligned} \tag{3.6}$$

$u_(t,\tau)$ ... rozdílový signál ze směšovače $[-]$
$g_{TR}(t,\tau)$ ... přenosová funkce prostředí před anténou $[-]$

$\phi_{TR}(t,\tau)$ ... fázový posun v prostředí před anténou $[rad]$

Vztah 3.6 lze přepsat do tvaru 3.7.

$$\begin{aligned} u(t,\tau) &= |g(t,\tau|\cos\left(2\pi\tfrac{B\tau}{T}t+\phi(t,\tau)\right) \\ |g(t,\tau)| &= \tfrac{1}{2}|g_{TR}(t,\tau)\cdot g_L(t)| \\ \phi(t,\tau) &= 2\pi\left(f_0\tau-\tfrac{B\tau^2}{2T}\right)+\phi_L(t)-\phi_{TR}(t,\tau)\approx \\ &\approx 2\pi f_0\tau+\phi_L(t)-\phi_{TR}(t,\tau) \quad pro\ \ f_0\tau\gg\tfrac{B\tau^2}{2T} \end{aligned} \tag{3.7}$$

Filtrovaný rozdílový signál ze směšovače se ovzorkuje $M = 256$ vzorky během doby snímání T. Snímání začíná u každého odběhu přesně ve stanoveném čase t_0, kdy začíná i vzorkování signálu. Vzorkovací perioda je $T_v = 4\ \mu s$. Ta odpovídá vzorkovacímu kmitočtu $f_s = 250\ kHz$. Získá se tak posloupnost vzorků 3.8. Z důvodu aproximace ve frekvenční oblasti se 256 vzorků doplní nulami na $N = 2^r$ vzorků (r je přirozené).

$$\begin{aligned} u(m,\tau) &= |g(m,\tau|\cos\left(2\pi\tfrac{B\tau}{M}m+\phi(m,\tau)\right) \\ u(m,\tau) &= 0 \ \text{ pro } \ m=256,257,...,2^r \end{aligned} \tag{3.8}$$

$u(m,\tau)$ ... vzorkovaný signál $[-]$

m ... pořadí vzorku $[-]$

M ... počet vzorků na jeden odběh $[-]$

Vzorkování rozdílového signálu musí splňovat Nyquistovo vzorkovací kritérium. Z něj vyplývá vztah 3.9.

$$\tfrac{B\tau_{max}}{T}=\tfrac{f_v}{2} \tag{3.9}$$

τ_{max} ... maximální jednoznačně měřitelné zpoždění $[s]$

Úpravou tohoto vztahu se získá podmínka 3.10.

$$\tau_{max}\leq\tfrac{M}{2B}=175,34\ ns \tag{3.10}$$

Pro další zpracování je lepší pracovat s komplexním analytickým signálem w dle vztahu 3.11.

$$\begin{aligned} w[m,\tau] &= u[m,\tau] + jv[m,\tau] = \\ &= |g[m,\tau]| \cdot \left[\cos\left(2\pi\tfrac{B\tau}{M}m + \phi(m,\tau)\right) + j\sin\left(2\pi\tfrac{B\tau}{M}m + \phi(m,\tau)\right)\right] = \\ &= g[m,\tau] \cdot e^{j2\pi\frac{B\tau}{M}m} \end{aligned} \tag{3.11}$$

$w[m,\tau]$ ... posloupnost vzorků komplexního analytického signálu pro jeden cíl $[-]$

Diskrétní Fourierovou transformací 3.12 se posloupnost 3.11 převede do kmitočtové oblasti.

$$\begin{aligned} &DFT\{w[m]\} = W[k] = \textstyle\sum_{m=0}^{N-1} w[m]e^{-j\frac{2\pi}{N}mk} = U[k] + jV[k] \\ &U[k] = \textstyle\sum_{m=0}^{N-1}\left(u[m]\cos\left(\frac{2\pi}{N}mk\right) + v[m]\sin\left(\frac{2\pi}{N}mk\right)\right) \\ &V[k] = \textstyle\sum_{m=0}^{N-1}\left(v[m]\cos\left(\frac{2\pi}{N}mk\right) - u[m]\sin\left(\frac{2\pi}{N}mk\right)\right) \end{aligned} \tag{3.12}$$

$W[k]$ ... posloupnost komplexních vzorků spektra $[-]$

$U[k]$ ... reálná část diskrétního spektra signálu $[-]$

$V[k]$ ... imaginární část diskrétního spektra signálu $[-]$

k ... pořadí prvku ve spektru $[-]$

DFT ze vztahu 3.12 lze vyjádřit i pomocí diskrétní periodické konvoluce 3.13

$$\begin{aligned} &W[k] = \textstyle\sum_{\kappa=0}^{N-1} G[\kappa]J[k-\kappa] \\ &\text{kde } G[\kappa] = DFT\{g[m]\} \text{ a } J[k] = DFT\{e^{j2\pi\frac{B\tau}{M}m}\} \end{aligned} \tag{3.13}$$

Lze dokázat, že funkce $|J[k]|$ má maximum, je-li splněna rovnost 3.14.

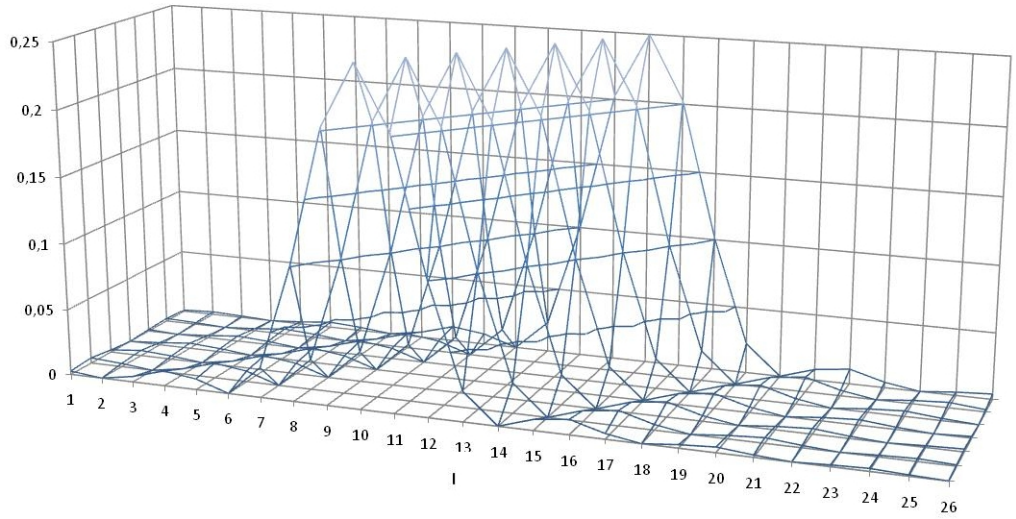
$$\tfrac{B\tau}{M} = \tfrac{k}{N} \tag{3.14}$$

Z rovnice 3.14 plyne, že zpoždění τ lze určit pouze v diskrétních hodnotách podle 3.15.

$$\tau_l = l \cdot \Delta\tau = l\tfrac{M}{N}\tfrac{1}{B} \tag{3.15}$$

Dosadí-li se 3.15 do 3.11, vyjde pro analytický signál diskrétní vztah 3.16.

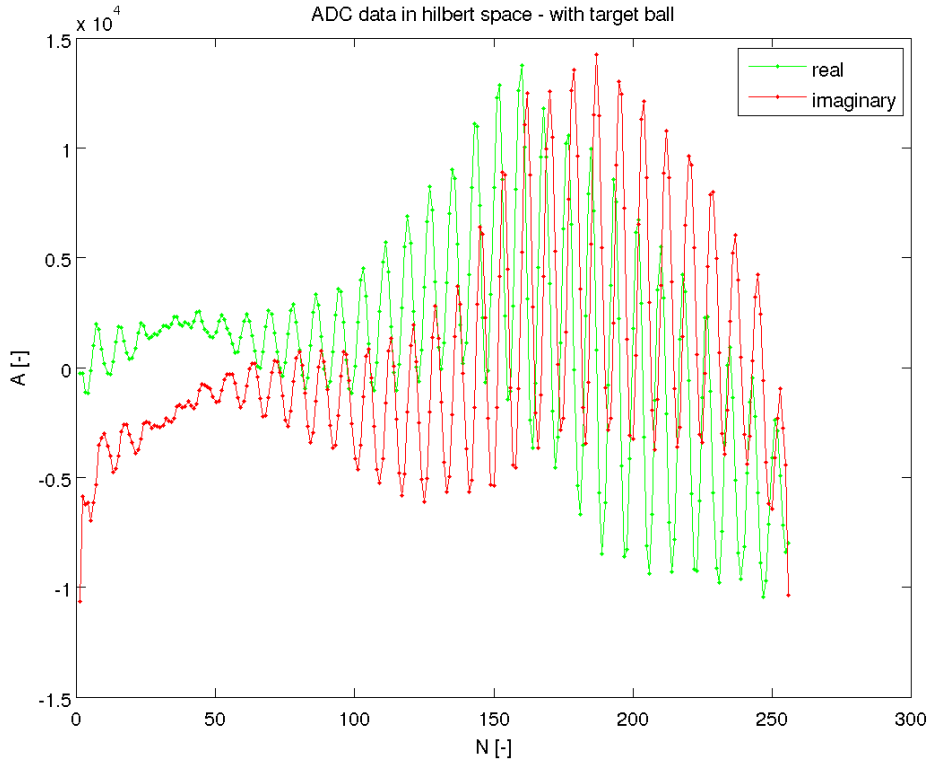
$$w[m,l] = g[m,l] \cdot e^{j\frac{2\pi}{N}lm} \tag{3.16}$$



Obrázek 3.2: Zobrazení funkcí $|J[k]|$ pro sadu několika po sobě jdoucích l

Celkový signál je tvořen příspěvkem od všech cílů podle vztahu 3.17.

$$w[m] = \sum_{l=1}^{L} g[m,l] \cdot e^{j\frac{2\pi}{N}lm} \tag{3.17}$$



Obrázek 3.3: Komplexní analytický signál vytvořený Hilbertovou transformací z reálného signálu na výstupu směšovače

Kapitola 4

Experimentální zařízení

Pro zajištění potřebných měření spojených s úkolem této diplomové práce vzniklo laboratorní zařízení, které by mělo nahradit pro počáteční experimenty jízdu rolby po sněhu. Toto zařízení by mělo rovněž umožnit měření anténního diagramu radaru v blízké zóně a mělo by sloužit k vyzkoušení radaru jako 3D scanneru.

Zařízení umožňuje pohyb radaru ve dvou osách nad prostorem, kde je umístěn cíl. Tento prostor je možné vyplnit krabicemi s expandovaným perlitem a tím simulovat suchý sníh. Krátkou jízdu je možné simulovat pohybem radaru v jedné z os. Jedna osa pohybu radaru je vybavena ultrazvukovým měřičem vzdálenosti, aby bylo možné k jednotlivým měřením přiřadit polohu.

4.1 Konstrukční parametry

Celá konstrukce je zhotovena ze dřeva a tím je pro radar prakticky neviditelná. Pohyb radaru je z ekonomických důvodů zajištěn jen lidskou silou. Rozměry experimentálního zařízení (interně nazývaného demonstrátor) jsou uvedeny na obrázku 4.2.

Obrázek 4.1: Experimentální zařízení

4.2 Náhrada sněhu

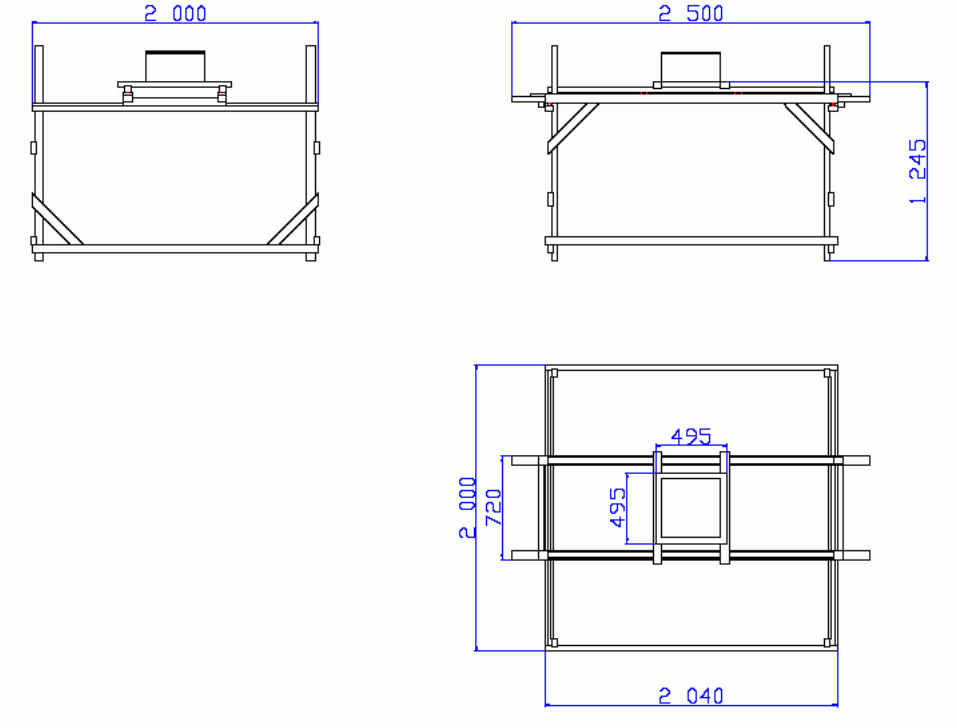
V laboratorních podmínkách je obtížné, až nereálné, pracovat se skutečným sněhem. Pro pokusné účely bylo nutné nahradit sníh látkou s podobnými dielektrickými vlastnostmi. Za tuto látku byl zvolen expandovaný perlit, látka používaná ve stavebnictví.

Důležitým parametrem je relativní permitivita, která je závislá na řadě vlivů jako jsou teplota, tlak a frekvence.

4.2.1 Sníh

Sníh představuje směs ledu, vzduchu a vody. Dříve provedenými experimenty se ukázalo, že hodnota permitivity sněhu odpovídá jeho měrné hmotnosti. S rostoucí hustotou roste permitivita.

Pokud se zvyšuje podíl vody ve sněhu, dochází rovněž k růstu permitivity, ale zároveň se zvyšuje vodivost prostředí a narůstá útlum pro signál, který se prostředím šíří.



Obrázek 4.2: Experimentální zařízení

4.2.2 Expandovaný perlit

Expandovaný perlit podle [3] je lehká, zrnitá, pórovitá hmota bílé nebo šedobílé barvy vyráběna tepelným zpracováním ze surového perlitu. Perlit je v podstatě amorfní křemičitan hlinitý sopečného původu. Patří ke kyselým vulkanickým sklům obdobně jako obsidián, smolek a pemza, od kterých se liší obsahem chemicky vázané vody. Tepelným zpracováním (expandací) při teplotách 900-1300°C vznikne produkt ve formě drobných dutých kuliček různých velikostí. Při expandaci se objem perlitu zvětšuje 5 až 10 krát.

Permitivita skla, jako hlavní pevné složky perlitu, je velmi blízká ledu. Budeme-li sníh považovat za zcela suchý, je druhou složkou už jen vzduch. Na základě předchozích tvrzení je možné perlitem sníh pro experimenty nahradit.

Tabulka 4.1: Dielektrická konstanta některých látek [4]

látka	ϵ_r
Vzduch	1,00059
Voda	81
Led	3,15
Sklo	3 - 19
Dřevo	2,04 - 7,3
Polystyrén pěnový	1,03
Vakuum	1

4.3 Kalibrační cíl

Přesně definovaný kalibrační cíl byl zhotoven z aranžovací polystyrenové koule polepené hliníkovou folií. Takto zhotovená vodivá koule poskytuje cíl s definovanými odraznými vlastnostmi, které jsou ve všech směrech stejné.

Obrázek 4.3: Kalibrační cíle

4.3.1 Efektivní odrazná plocha

Koule představuje nesměrový odražeč, u kterého efektivní odrazná plocha závisí v podstatě na poměru poloměru koule r k vlnové délce λ [14]. Pro vodivé koule o poměru $r/\lambda < 0.13$ platí 4.1.

$$S_e = 9\left(2\pi\frac{r}{\lambda}\right)^4 \pi r^2 \tag{4.1}$$

Pro větší poměry r/λ má efektivní odrazná plocha oscilační charakter a pro velmi velké poměry se stává efektivní odrazná plocha na zmíněném poměru nezávislou a je rovna přibližně ploše hlavního kruhu koule, tj. pro $r/\lambda > 1$ platí vztah 4.2

$$S_e = \pi r^2 \tag{4.2}$$

Pro pokusy byly zhotoveny dvě koule. Jedna o průměru 12 cm a druhá o průměru 6 cm.

Kapitola 5

Softwarové prostředky

Veškeré činnosti související s touto diplomovou prací byly prováděny pod operačním systémem Linux. Výjimku tvoří jen přenos po sběrnici CAN, kdy byl potřeba analyzátor sběrnice dodávaný s ovladačem pouze pro operační systém Windows.

Tato kapitola shrnuje vybrané knihovny a nástroje, které je možné nejen v rámci Linuxu využívat a usnadnit si tak programování.

5.1 Knihovna FFTW

FFTW [1] je knihovna funkcí jazyka C pro optimalizované výpočty diskrétní Fourierovy transformace (DFT) v prostoru jedné i více proměnných. Výpočty lze provádět na množině komplexních i reálných čísel s různým počtem vstupních prvků. Další možnosti jsou výpočty diskrétní kosinové (DCT) a sinové transformace (DST).

Výkon použitých algoritmů je optimalizován pro řadu platforem, předčí většinu volně dostupných ekvivalentů a je srovnatelný s komerčními knihovnami pro výpočet FFT.

5.1.1 Plánování a spuštění výpočtu

Práce knihovních funkcí je rozdělena do dvou fází. První fází je nastavení výpočtu. Toto nastavení se provede funkcí `fftw_plan` a jedná se o výběr vhodného algoritmu pro daný problém a inicializaci parametrů.

Před voláním funkce musí být v paměti alokována vstupní a výstupní pole. Podle volané funkce lze nastavit zachování hodnot ve vstupním poli nebo využití tohoto prostoru k výpočtu.

Zde takzvané plánování slouží k urychlení samotného výpočtu a zásadní vliv má na rychlost při opakovaných výpočtech FFT . Dále stačí jen předložit vstupní data a spustit výpočet funkcí `fftw_execute` s parametrem, kterým je dříve vytvořený plán.

5.1.2 Komplexní proměnné v jazyce C

Většinou při výpočtech rychlých Fourierových transformací je vstup nebo výstup v komplexním oboru a je zapotřebí zvolit vhodný datový typ pro práci s komplexními čísly.

Existují dvě možnosti jak to zajistit. Možností je buď vytvoření datového typu: `typedef double fftw_complex[2];` a nebo využít specifikaci jazyka C99 a novější, kde je již podpora pro komplexní čísla zajištěna. Ve druhém případě je nutné vložit hlavičkový soubor `complex.h` dříve než `fftw3.h`.

5.2 Nástroj Gnuplot

Gnuplot [2] je všestranný multiplatformní nástroj ovládaný z příkazového řádku. Byl stvořen, aby poskytl vědcům a studentům interaktivní zobrazení matematických funkcí a dat. Lze ho použít i jako součást jiných aplikací, kde usnadňuje vykreslování grafů.

S pomocí Gnuplot je možné vykreslovat 2D a 3D grafy. Umožňuje kreslení bodů, čar, plošných geometrických útvarů, vektorových polí, prostorových ploch a řadu dalších speciálních grafů.

Gnuplot umožňuje rozličné formy výstupu: počínaje zobrazením na obrazovce s interaktivním použitím myši a klávesnice, přes přímý výstup na plotter nebo tiskárnu, konče nespočetným množtvím souborových formátů (eps, png, jpeg, svg, pdf, LaTeX, postscript atd.).

5.2.1 Propojení s C

Gnuplot lze použít z vlastního programu v jazyce C dvěma způsoby. Jedním způsobem je vygenerovat skript pro GNU plot a ten následně zobrazí druhý vygenerovaný soubor s grafem. Druhým způsobem je spuštění programu skrz rouru a tak zobrazit graf. Zde je použit druhý způsob uvedený dále v listingu.

5.2.2 Barevná mapa

Pro grafy zobrazující amplitudu v závislosti na dvou proměnných je důležité vhodné přiřazení barvy amplitudě. Zde použitý graf `pm3d` umožňuje uživatelské přiřazení vhodné palety ze souboru.

V tomto souboru jsou v řádcích uvedeny přiřazení barvy některým úrovním amplitudy. Mezi těmito úrovněmi přecházejí barevné složky lineárně a tak je v tomto rozsahu barevný odstín rovněž definován.

První hodnota v řádku představuje relativní úroveň amplitudy. Nejnižší amplituda je na prvním řádku a nejvyšší na posledním. Další tři hodnoty v řádku jsou intenzity barevných složek RGB, každá v rozsahu $< 0; 1 >$.

Uvedeným způsobem lze dosáhnout vhodného a přehledného zobrazení. Příklad palety, která je zde použita, je v listingu `paleta.dat`.

`paleta.dat`

```
8       0       0       0.5
22      0       1       1
31      0.5     1       0.5
45      1       1       0
54      1       0       0
60      0.5     0       0
```

5.2.3 Vytvoření grafu v C

Dále je uveden listing s příkladem vykreslení grafu s pomocí Gnuplot.

Na prvním řádku je otevření roury, tedy spuštění programu Gnuplot a přesměrování standardního vstupu na náš program. Ve druhém a třetím řádku jsou uvedeny dva možné formáty výstupu grafu. Čtvrtý řádek určuje výstupní soubor. Dále se nastavuje typ grafu, paleta, popisky a rozsahy os. Nakonec je graf vykreslen a roura uzavřena.

Roura z C do Gnuplot

```
_gnuplot = popen("gnuplot", "w");
//   fprintf(_gnuplot,"set term png size 800,600\n");
fprintf(_gnuplot,"set terminal postscript eps enhanced\n");
fprintf(_gnuplot,"set output \"%s.eps\"\n",soubor);
fprintf(_gnuplot,"set pm3d map\n");
fprintf(_gnuplot,"set palette file \"paleta.dat\"\n");
//fprintf(_gnuplot,"set logscale z\n");
fprintf(_gnuplot,"set yrange [-180:-123]\n");
fprintf(_gnuplot,"set ylabel \"delay {/Symbol t} (frequency lines)\"\n");
fprintf(_gnuplot,"set xlabel \"radar position [cm]\"\n");
fprintf(_gnuplot,"set cblabel \"signal power [dB]\"\n");
fprintf(_gnuplot,"set xrange [0:%d]\n",POCET_ODBEHU-1);
fprintf(_gnuplot,"set zrange [%f:%f]\n",PLOT_MIN,PLOT_MAX);
//fprintf(_gnuplot,"set cbrange [0:35000]\n");
fprintf(_gnuplot,"splot \"%s.dat\" with pm3d\n",soubor);
pclose(_gnuplot);
```

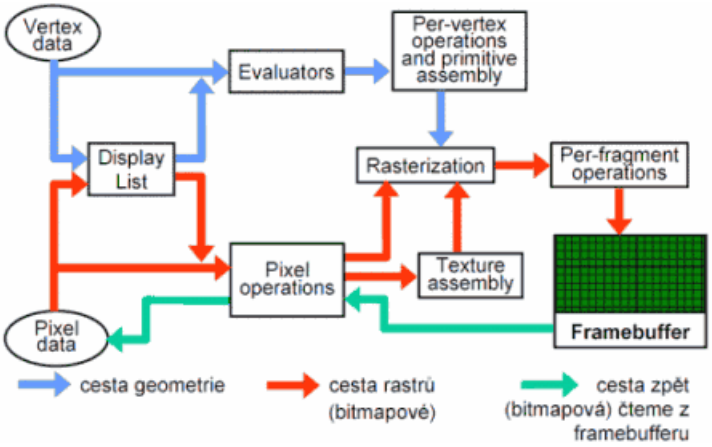
5.3 OpenGL

OpenGL je knihovna navržená firmou SGI jako programové aplikační rozhraní pro jejich grafické akcelerátory.

Z programátorského hlediska se OpenGL chová jako stavový automat [11]. To znamená, že během zadávání příkazů pro vykreslování lze průběžně měnit vlastnosti vykreslovaných primitiv (barva, průhlednost) nebo celé scény (volba způsobu vykreslování, transformace) a toto nastavení zůstane zachováno do té doby, než ho explicitně změníme. Výhoda tohoto přístupu spočívá především v tom, že funkce pro vykreslování mají menší počet parametrů a že jedním příkazem lze globálně změnit způsob vykreslení celé scény, například volbu drátového zobrazení modelu (wireframe model) nebo zobrazení pomocí vyplněných polygonů (filled model). Vykreslování scény se provádí procedurálně – voláním funkcí OpenGL se vykreslí výsledný rastrový obrázek. Výsledkem volání těchto funkcí je rastrový obrázek uložený v tzv. framebufferu, kde je každému pixelu přiřazena barva, hloubka, alfa složka popř. i další atributy. Z framebufferu lze získat pouze barevnou informaci a tu je možné následně zobrazit na obrazovce.

5.3.1 Vykreslovací řetězec

Při vykreslování 2D obrazců či celých 3D scén se do vykreslovacího řetězce (rendering pipeline) OpenGL nahrávají dva odlišné typy dat: údaje o vrcholech a rastrová data. Tato data jsou průběžně vykreslovacím řetězcem zpracovávána, přičemž se na výstupu (ve framebufferu) vytváří rastrová podoba zadané scény. Jednotlivé moduly vykreslovacího řetězce jsou zobrazeny na obrázku 5.1.



Obrázek 5.1: Zobrazovací řetězec OpenGL [12]

5.3.2 Základní geometrické útvary

Veškeré geometrické objekty se v OpenGL vytvářejí z několika základních tvarů. Jak už bylo psáno, chová se OpenGL jako stavový automat. I zde to znamená, že při vkládání těchto primitiv dojde k přechodu do stavu, kdy se zadávají body útvaru. Tento stav se vyvolá funkcí `glBegin` a ukončí `glEnd`.

Dále je uvedeno několik příkladů základních geometrických útvarů.

Bod

Nejjednodušším prvkem je bod. Body se v terminologii OpenGL nazývají vertexy.

Způsobem uvedeným v následujícím listingu jsou do scény vloženy 3 samostatné body. Body jsou zadávány v trojrozměrném kartézském systému, v ukázce s čísly v plovoucí řádové čárce.

Vložení bodů

```
glBegin(GL_POINTS);
  glVertex3f(1.0,1.4,1.8);
  glVertex3f(5.0,3.2,-0.8);
  glVertex3f(1.1,1.2,1.9);
glEnd();
```

Trojúhelník

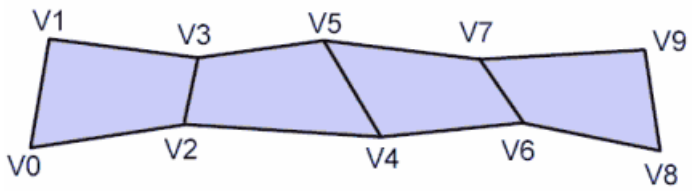
Základním plošným útvarem je trojúhelník. Umožňuje vykreslení plochy.

Body se vkládají do bloku `glBegin(GL_TRIANGLES)`. Jejich počet v tomto bloku musí být dělitelný třemi.

Pás čtyřúhelníků

Tento útvar je využíván dále při zobrazení anténního diagramu.

Objekt začíná `glBegin(GL_QUAD_STRIP)` dále jsou vkládány body v pořadí uvedeném na obrázku 5.2. Musí být vloženy nejméně 4 body. Počet bodů musí být sudý.



Obrázek 5.2: Pás čtyřúhelníků [13]

5.4 Mesa 3D

Mesa 3D je knihovna s otevřeným zdrojovým kódem implementující část API OpenGL bez potřeby hardwarového grafického akcelerátoru. S touto knihovnou je možné provozovat programy využívající techniky OpenGL bez hardwarové akcelerace.

Pokud by měl být vytvořen 3D scanner a nebyl by k dispozici procesor s níže popsaným typem hardwarového urychlení grafiky, byla by tato knihovna možnou al-

ternativou. Odpadla by tak nutnost zvládnou složité algoritmy 3D zobrazování. Softwarové výpočty by však měly zásadní dopad na výkon a vytížení už tak zaměstnaného procesoru.

5.5 OpenGL ES

OpenGL ES představuje standard hardwarové podpory akcelerovaného vykreslování grafiky pro embedded zařízení. V dnešní době prožívající rozmach v mobilních telefonech a internetových tabletech. OpenGL ES je součástí mnoha aplikačních procesorů v těchto zařízeních. V případě inovací, ke kterým vývoj sněžného radaru směřuje, by bylo vhodné takový procesor využít, pokud by měl být radar použit jako 3D scanner s přímým zobrazením do trojrozměrného prostoru.

5.5.1 Procesory OMAP

Jedním z potenciálně vhodných procesorů implementujících API OpenGL ES by mohl být procesor Texas Instruments OMAP (Open Multimedia Application Platform). Jedná se o procesor ARM se signálovým koprocesorem.

Nejmodernější dostupnou variantou je v současné době model OMAP4430. Jedná se o dvoujádrový procesor ARM Cortex A9 vybavený signálovým koprocesorem TMS320 a především podsystémem pro akceleraci 2D a 3D grafiky.

Výhodou proti stávajícímu procesoru by mělo být daleko větší množství dostupné operační paměti a hlavně mnohonásobný výpočetní výkon.

Nevýhodou uvedeného signálového koprocesoru je možnost pouze výpočtů v pevné řádové čárce. Při vhodném návrhu softwaru by to nemělo být překážkou. Navíc jádro ARM Cortex A9 obsahuje jednotku pro výpočty v plovoucí řádové čárce a mohlo by nahradit samotný signálový procesor v těchto výpočtech.

Samozřejmostí je zde běh nejen Linuxu, ale i sofistikovaných operačních systémů pro embedded systémy (RTOS DSP/BIOS). Linux by mohl být vhodnou volbou alespoň na jednom jádře, protože v něm lze využívat knihovny popsané v této kapi-

tole. Vytvoření softwaru pro 3D scanner by tak bylo snadnější, zvlášť pokud by bylo vyžadováno grafické uživatelské rozhraní.

5.6 Knihovna QT

Knihovna QT poskytuje nástroje usnadňující vývoj aplikací včetně grafického uživatelského prostředí.

S použitím QT je možné vytvořit aplikaci s grafickým uživatelským prostředím doslova na pár řádcích, jak ukazuje následující listing.

Aplikace s QT

```
#include <QApplication>
#include <QTextEdit>

int main(int argv, char **args)
{
  QApplication app(argv, args);

  QTextEdit textEdit;
  textEdit.show();

  return app.exec();
}
```

5.6.1 Signály a sloty

Signály a sloty jsou v QT mocným nástrojem. Umožňují komunikaci mezi objekty GUI (widgety).

Pokud jeden widget změní svůj stav (třeba uživatelským zásafem), vyvolá patřičný signál. Ostatní widgety tento signál akceptují, pokud ho mají připojen do slotu. Uvedené propojení se vytváří metodou `connect`.

Dříve byl tento princip implementovaný formou callback metody objektu. Tato metoda je vyvolána pokud má být objekt informován o akci z jiného objektu. Nevýhodou je chybějící typová kontrola callback metody, protože je znám jen její ukazatel. Zároveň musí volaná meto znát ukazatel na metodu, ze které byla vyvolána.

5.6.2 QT Designer

QT Designer je software sloužící k návrhu grafického uživatelského prostředí využívajícího knihovnu QT. Umožňuje vytváření GUI ve viditelné formě z jednotlivých komponent (formuláře, tlačítka, popisky atd.).

Výstupem programu je soubor `*.ui`, který je možné snadno překonvertovat do `*.H` a `*.cpp` souborů a začlenit je do vlastního programu.

Jednotlivé prvky GUI se při návrhu v QT Designeru standardně propojují pomocí signálů a slotů. Lze tak ovlivňovat chování uživatelského prostředí.

5.6.3 Licence

Knihovnu QT je možné využívat se dvěma formami licence.

První formou je svobodné využívání za splnění podmínek licence GNU LGPL verze 2.1. Zkráceně: je možné neomezené použití, pokud bude umožněn přístup ke zdrojovému kódu aplikace.

Druhou formou je takzvané licencování QT Commercial. Tato zpoplatněná forma použití nevyžaduje šíření kódu vytvořené aplikace. Hodí se pro aplikace určené k prodeji.

Volba licencování v případě QT v embedded systémech je složitou otázkou. Patrně se zde nevyplatí komerční verze, protože aplikace bude zřejmě nefunkční pokud nepoběží na cílovém systému.

5.6.4 Okenní správce KDE

Zkráceně KDE, plným jménem K Desktop Environment, je prostředí okenního správce pro operační systém Linux a unixové systémy. Toto prostředí je dnes plně postaveno na knihovně QT.

Součástí prostředí je mnoho aplikací a tím je utvořen jednotný vzhled a funkcionalita systému.

5.6.5 QT Embedded

Knihovna QT nenalézá uplatnění jen v prostředí desktopu, je ji možné použít v nejrůznějších zařízeních, kde je potřeba vytvořit grafické uživatelské rozhraní.

K těmto účelů slouží tzv. QT Embedded, což je verze knihovny optimalizovaná pro zařízení s malou pamětí a omezenými prostředky. Na druhou stranu je zde podpora pro dotykové obrazovky a není vyžadován okenní systém X11. K vykreslování na display stačí přístup k framebufferu.

Touto formou je postaven systém Maemo na mobilních telefonech Nokia.

Kapitola 6

Sběr dat

Shromáždění dat měření spojených s polohou radaru je jedním z hlavních úkolů této práce. K dalším výpočtům je nutné zajistit několik měření v jednotlivých bodech plochy, ve které je možné s radarem na demonstrátoru pohybovat.

Pro sestavení směrové charakteristiky radaru je nutné provést měření v celé ploše dvakrát. Jedno měření je bez definovaného cíle a druhé je s cílem. Odečtením těchto dvou měření dosáhneme toho, že získáme signál, kde je pouze kalibrační cíl. Cíl je při těchto měřeních umístěn na polystyrenových deskách. Rozhraní vzduch-polystyren je pro radar prakticky neviditelné.

Další skupina měření probíhá nad prostorem vyplněným krabicemi s expandovaným perlitem. V jedné krabici je umístěna opět vodivá koule jako definovaný cíl. Toto měření simuluje sněhovou pokrývku.

6.1 Rozhraní CAN

Standard sběrnice CAN pochází od firmy Robert Bosch GmbH a byl vyvinut pro zajištění komunikace řídicích jednotek ve vozidlech.

Ve sněžné rolbě je tato sběrnice využívána a poskytuje spojení mezi radarem a zobrazovací jednotkou v rolbařově kabině.

Ve finální verzi zařízení by se měla přenášet po sběrnici jen informace o hloubce sněhu. Ve vývojové verzi je použita k veškeré datové komunikaci s radarem. Po

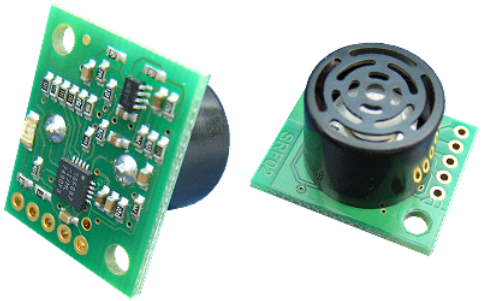
sběrnici je možné vysílat jednotlivé naměřené odběhy, sledovat teplotu a napájení radaru. Je také možné ovládat signálový procesor a aktualizovat jeho software.

V laboratoři je CAN rozhraní radaru přímo připojeno na CAN analyzátor firmy Sontheim Industrie Elektronik GmbH, který zprostředkovává komunikaci s PC skrz port USB.

6.2 Měření vzdálenosti ultrazvukem

Z počátku probíhalo manuální nastavovaní polohy radaru v obou osách pohybu. To se ukázalo jako nevhodné, protože měření trvalo příliš dlouho.

Demonstrátor byl z těchto důvodů dovybaven snímáním polohy radaru v ose Y čidlem znázorněným na obrázku 6.1. Čidlo disponuje pouze rozhraním UART a k počítači se připojuje pomocí univerzálního převodníku UART-USB.

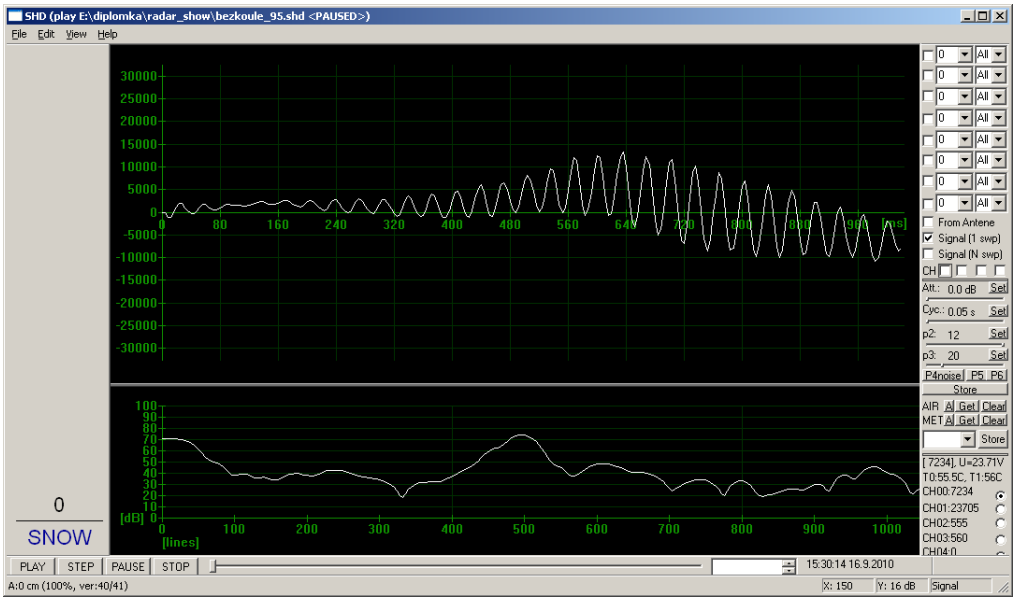


Obrázek 6.1: Ultrazvukový měřič vzdálenosti [5]

6.3 Ovládací SW

Při předchozím vývoji radaru byl současně s radarem vytvářen software pro PC, který zobrazuje naměřené průběhy. Při testech v laboratoři nahrazuje signálový procesor ve výpočtech a slouží k ovládání radaru. Tento software byl využit i pro získávání dat. Pro tyto účely byl rozšířen o modul určení polohy v ose Y demonstrátoru a modul, který vytváří datový proud v IP síti. Tímto se zajistila abstrakce od rozhraní CAN, především od softwarového ovladače Windows.

Tento software tak přijímá data s jednotlivými odběhy z radaru a zároveň informaci o poloze z ultrazvukového měřiče vzdálenosti. Vytvoří UDP datagram s tímto obsahem a odešle ho na jiný počítač v síti, kde probíhá sběr dat a jejich zpracování.



Obrázek 6.2: Původní software

6.4 Datový proud přes UDP/IP

Protokol UDP představuje jeden z protokolů transportní vrstvy ze skupiny síťových protokolů. U přenosu pomocí datagramů UDP není zajištěno jejich doručení. Tento způsob přenosu se využívá především pro multimediální informace. Zde byl tento typ přenosu zvolen kvůli jednoduché implementaci.

Vysílací strana sestaví datagram, odešle ho a o víc se nestará. Přijímací strana jen čeká a sbírá data.

Délka jednoho UDP datagramu může být až 65536 bytů. To je dostatečný prostor pro jeden odběh, který je tvořen 256 čísly s délkou 16 bitů, tedy 512 bytů.

Tabulka 6.1: UDP datagram

+	bity 0 - 15	bity 16 - 31
0	zdrojový port	cílový port
32	délka	kontrolní součet
.	data	

6.4.1 Obsah UDP datagramu

V každém datagramu je jeden odběh radaru a pomocné informace, pro které je vyčleněno prvních 8 bytů. Celý datový blok obsahuje 520 bytů. Strukturu dat znázorňuje tabulka 6.2.

Tabulka 6.2: Datová část UDP datagramu

byte	funkce
0 - 1	poloha radaru na ose Y
2 - 7	rezervováno
8 - 519	vzorky z radaru

6.5 Rozšíření původní aplikace

Původní aplikaci bylo nutné rozšířit o část, která by odesílala naměřená data po síti do jiného PC. Tato komunikace je, jak bylo napsáno výše, zajištěna protokolem UDP. Za tímto účelem byla vytvořena třída v jazyce C++, která se stará o odesílání bloků dat po síti.

6.5.1 UDP klient

UDP klient je program který odesílá UDP datagramy s využitím protokolu nižší vrstvy IP na rozhraní, kde naslouchá UDP server. Kód, který byl implementován do původního ovládacího softwaru radaru, je uveden v následujících dvou výpisech.

Implementace je provedena v podobě třídy jazyka C++. V konstruktoru této třídy se nastaví adresát UDP datagramů. Dále jsou jen vkládána data metodou `UdpSend`. Tato data se bezprostředně po zavolání metody odešlou.

udpdata.H

```
#ifndef _UDPDATA_H_
#define _UDPDATA_H_

#ifdef BUILD_DLL
/* DLL export */
#define EXPORT __declspec(dllexport)
#else
/* EXE import */
#define EXPORT __declspec(dllimport)
#endif

EXPORT class UdpData
{
  private:
    WSADATA wsaData;
    SOCKET SendSocket;
    sockaddr_in RecvAddr;
  public:
    EXPORT UdpData(String addr, int port);
    EXPORT ~UdpData();
    EXPORT void UdpSend(unsigned char *data);
};

#endif
```

udpdata.cpp

```
#include "udpdata.H"
#include <iostream>
#include <windows.h>
#include <winsock2.h>
using namespace std;

EXPORT UdpData::UdpData(String addr, int port)
{
  //---------------------------------------------
  // Initialize Winsock
  WSAStartup(MAKEWORD(2,2), &wsaData);

  //---------------------------------------------
  // Create a socket for sending data
  SendSocket = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);

  //---------------------------------------------
  // Set up the RecvAddr structure with the IP address of
```

```
  // the receiver (in this example case "192.168.1.1")
  // and the specified port number.
  RecvAddr.sin_family = AF_INET;
  RecvAddr.sin_port = htons(port);
  RecvAddr.sin_addr.s_addr = inet_addr(addr); //htonl(INADDR_BROADCAST);
}

EXPORT UdpData::~UdpData()
{
  closesocket(SendSocket);
  WSACleanup();
}

EXPORT void UdpData::UdpSend(unsigned char *data)
{
  sendto(SendSocket,
    (const char *)data,
    2048,
    0,
    (SOCKADDR *) &RecvAddr,
    sizeof(RecvAddr));
}
```

6.6 Záznam dat

Na protistraně síťové komunikace naslouchá UDP server a příchozí data zaznamenává do souboru. Soubory jsou textové, aby bylo možné snadno ověřit data v nich zapsaná a především, aby je bylo možné snadno importovat do jiných aplikací bez obtěžující práce s nějakým typem binárně kódovaného souboru.

Začátek záznamu se zahájí se spuštěním popisovaného programu. Pro každý přesun v ose Y je nutné spustit program s parametrem, kterým je jméno souboru, kam se data zaznamenají. Vždy se takto zaznamená jeden přesun v ose Y. Průběh záznamu s vyznačením zaznamenaných bodů je možné sledovat v textové konzoli. Až jsou všechny body zaznamenány program lze ukončit signálem `Ctrl+C`.

6.6.1 UDP server

UDP server je program, který naslouchá na zvoleném portu UDP a čeká na příchozí datagramy.

Kód implementující UDP server je uveden v následujícím výpise. Funkce `recvfrom` je zde blokující a čeká na příchod datagramu. Zde je kód jen pro přiblížení funkce. Ve skutečném programu pro záznam dat dojde k dekódování datagramu a uložení dat z něj. Zde cyklus pokračuje v nekonečné smyčce dokud není z vnějšku nastaven příznak ukončení `flag_konec`.

UDP server

```
if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
        perror("socket");
          exit(1);
}
my_addr.sin_family = AF_INET;                  // host byte order
my_addr.sin_addr.s_addr = INADDR_ANY; // automatically fill with my IP
my_addr.sin_port = htons(MYPORT);          // short, network byte order
memset(&(my_addr.sin_zero), '\0', 8); // zero the rest of the struct
memset(obsazeni,'.',POCET_ODBEHU);

if (bind(sockfd, (struct sockaddr *)&my_addr,
          sizeof(struct sockaddr)) == -1) {
          perror("bind");
           exit(1);
}

while(!flag_konec){
  if ((numbytes = recvfrom(sockfd, buf, MAXBUFLEN-1 , 0,
   (struct sockaddr *)&their_addr, &addr_len)) == -1) {
    perror("recvfrom");
    exit(1);
  }
}
```

6.6.2 Formát datových souborů

Do datových souborů jsou zaznamenány jednotlivé odběhy. Data jsou v textovém formátu. Každý odběh připadá na jeden řádek. První hodnota je poloha naměřená ultrazvukem a další hodnoty na řádku jsou vzorky z daného měření.

Kapitola 7

Zobrazení GPR řezů z naměřených dat

V průběhu experimentů bylo nutné zajistit různé typy zobrazení naměřených dat, aby bylo možné ověřit fungování radaru.

Jedná se o zobrazení signálu od jediného frekvenčního zdvihu (odběhu) až po spektrogram s konstantním τ získaný měřením na celé ploše demonstrátoru (řez C).

Způsoby jednotlivých zobrazení jsou popsány dříve.

7.1 Zobrazení a výpočty spekter v jazyce C

Pro vybrané úkoly bylo sestaveno několik jednoduchých programů fungujících v příkazovém řádku. Jednotlivé datové soubory jsou programům předávány jako parametry při spuštění. Tyto soubory jsou dříve vytvořeny programem pro sběr dat.

Celý datový soubor pro jeden řez typu B je načten do paměti. S pomocí knihovny FFTW se v paměti spočítá spektrum jednotlivých odběhů. Toto spektrum je komplexní a před záznamem do souboru pro program Gnuplot, který vytváří grafy, je nutné spočítat velikost komplexních čísel, aby spektrum bylo čistě reálné, amplitudové.

7.1.1 Načtení datového souboru

Načtení celého datového souboru provádí kód z následujícího výpisu.

Načtení datového souboru

```
fd = fopen(argv[1],"r");
if(!fd) {
  printf("File missing or wrong...\n");
  return 2;
}

for(int i=0;i<POCET_ODBEHU;i++)
{
  for(int j=0;j<256;j++)
  {
    odbehy[i][j]=cislo;
  }
}
fclose(fd);
```

7.1.2 Výpočet spektra

Níže je uvedena část kódu, která se stará o výpočet spekter pro všechny odběhy ve spektrogramu. Nejprve se provede plánování výpočtu FFT a dále se jen výpočet spustí pro různá data.

Výpočet spektrogramu

```
//planovani fft
xPlan = fftw_plan_dft_r2c_1d(
    1024, dIn, xOut, FFTW_MEASURE | FFTW_PRESERVE_INPUT);

//vypocet originalniho obrazu
for(int i=0;i<POCET_ODBEHU;i++)
{
  for(int j=0;j<256;j++)
  {
    dIn[j]=(double)odbehy[i][j];
  }
  //vynulovat vstup kvuli interpolaci
  for(int j=256;j<1024;j++)
  {
    dIn[j] = 0.0;
  }
  //pustit fft na odbeh
  fftw_execute(xPlan);
  for(int j=0;j<512;j++)
  {
```

```
      spektrum[i][j] = xOut[j];
    }
  }
```

7.1.3 Funkce pro zobrazení spektrogramu

Vstupem funkce je již předpočítaný řez B. Hodnoty spektra jsou komplexní.

Ve funkci jsou definována makra `PLOT_MIN` a `PLOT_MAX`, která dále určují amplitudové omezení grafu. Prakticky se jedná o prahování, aby byla zobrazena jen důležitá data.

Následuje výpočet velikosti komplexních čísel tvořících spektrogram. K tomuto účelu je využita funkce `cabs`. Aby mohla být použita je třeba využívat specifikaci jazyka C99 a novější, kde je možné provádět výpočty v komplexním oboru.

Funkce končí vytvořením grafu programem Gnuplot, na jehož vstup jsou parametry předány pomocí roury.

Grafický výstup spektrogramu

```
void plot_spektrum_2d(
  double complex data[POCET_ODBEHU][512], char *soubor
)
{
    FILE *_fx, *_gnuplot;
  int _index1,_index2;
  double val;
  char s[40];
 #define PLOT_MIN 70.0
 #define PLOT_MAX 110.0


 sprintf(s,"%s.dat",soubor);
 _fx = fopen(s,"w");
    if(_fx)
    {
      for(_index1=0;_index1<POCET_ODBEHU;_index1++)
      {
        for(_index2=0;_index2<512;_index2++)
        {
            val = 10*log10(cabs(data[_index1][_index2])*
                                cabs(data[_index1][_index2]));
            val = val < PLOT_MIN ? PLOT_MIN : val;
            val = val > PLOT_MAX ? PLOT_MAX : val;
            fprintf(_fx,"%d %d %f\n",_index1,-_index2,val);
```

```
        }
        fprintf(_fx,"\n");
      }
      fclose(_fx);
    }
    else
    {
      printf("Nepodarilo se otevrit soubor pro hlavni vystup\n");
    }

  _gnuplot = popen("gnuplot", "w");
  fprintf(_gnuplot,"set terminal postscript eps enhanced\n");
  fprintf(_gnuplot,"set output \"%s.eps\"\n",soubor);
  fprintf(_gnuplot,"set pm3d map\n");
  fprintf(_gnuplot,"set palette file \"paleta.dat\"\n");
  //fprintf(_gnuplot,"set logscale z\n");
  fprintf(_gnuplot,"set yrange [-180:-123]\n");
  fprintf(_gnuplot,"set ylabel \"delay {/Symbol t} (frequency lines now)\"\n");
  fprintf(_gnuplot,"set xlabel \"radar position [cm]\"\n");
  fprintf(_gnuplot,"set cblabel \"signal power [dB]\"\n");
  fprintf(_gnuplot,"set xrange [0:%d]\n",POCET_ODBEHU-1);
  fprintf(_gnuplot,"set zrange [%f:%f]\n",PLOT_MIN,PLOT_MAX);
  fprintf(_gnuplot,"splot \"%s.dat\" with pm3d\n",soubor);
  pclose(_gnuplot);
}
```

7.1.4 Hromadné zpracování

Jelikož měření bylo prováděno mnoho, bylo nutné zpracování automatizovat. Pro tyto účely byl využit skriptovací programovací jazyk BASH.

Toto dávkové zpracování bylo nejprve použito pro řezy typu A a následně i pro řezy typu B.

Dávkové vytvoření řezů B

Pro vytvoření řezů B pro všechny polohy X (rozestup 5 cm v celé délce osy pohybu) byl využit skript v následujícím výpisu.

Skript na vytvoření řezu B

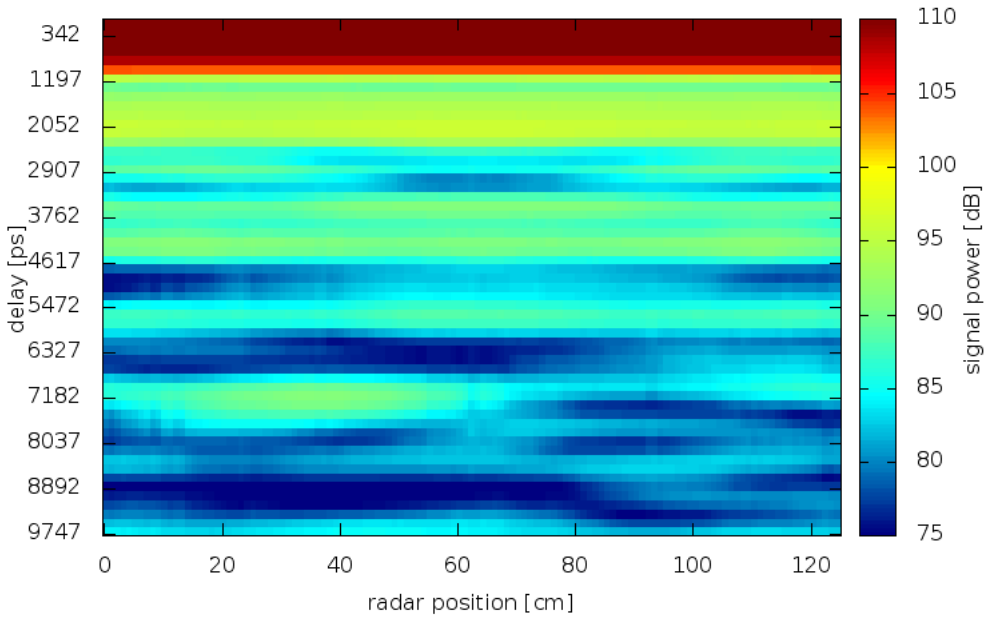
```
#!/bin/sh
for i in $(seq -f%03.0f 0 5 105)
do
  ../proces rec_$i
done
```

7.2 Odstranění vlastních odrazů systému

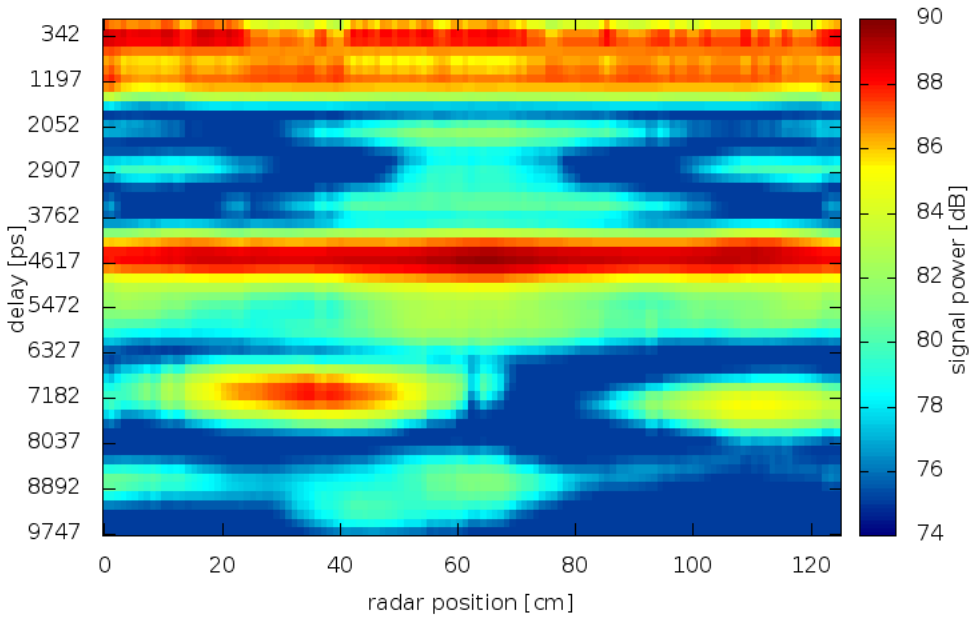
Samotný systém, tedy vedení, anténa a kovová konstrukce radaru, vytváří mnoho vlastních odrazů. Tento rušivý signál prakticky nelze odstranit jinak než jeho změřením a následným odečtením.

Tento signál lze naměřit pokud v okolí radaru nebudou jiné cíle. To lze zajistit namířením radaru do nebe. Tento proces je nutné provést s každým vyrobeným radarem, protože je do jisté míry jedinečný. Tento kalibrační průběh je zaznamenán v paměti radaru. Pro naše měření byl z paměti vyčten a uložen.

Na obrázku 8.4 je spektrogram, kde nebyl odečten průběh obsahující vlastní odrazy. Následuje obrázek 7.2, kde je výrazné zlepšení díky odečteným odrazům.



Obrázek 7.1: Zobrazení typu B, spektrogram surových dat z radaru



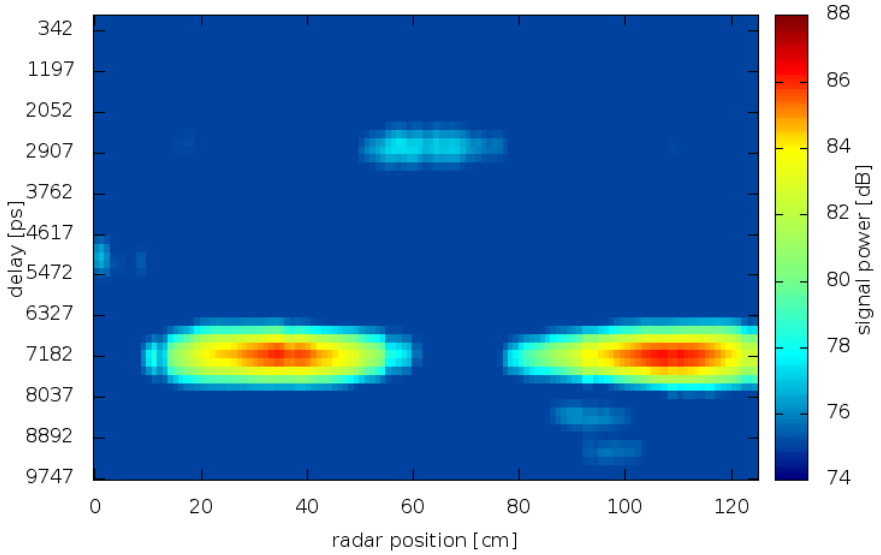
Obrázek 7.2: Zobrazení řezu, kde je od jednotlivých odběhů odečten signál vlastních odrazů radaru

7.3 Separace rovinných rozhraní

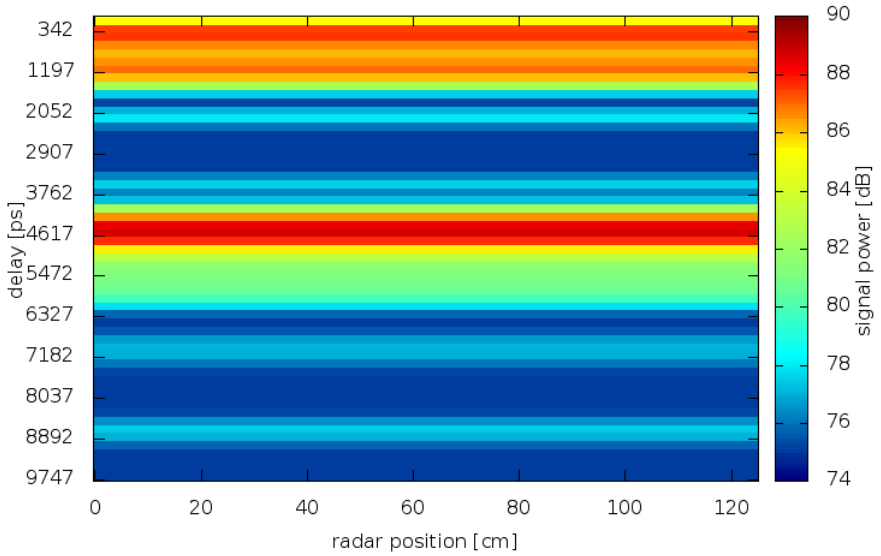
Doposud byla síla vrstvy sněhové pokrývky určována z dat získaných v jednom místě měření. Přesněji řečeno bylo využito více (asi 40) průměrovaných odběhů. Průměrováním došlo sice k potlačení rušení, ale čas, po který se průměrované odběhy sbíraly, byl krátký a rolba ujela zanedbatelnou vzdálenost.

V experimentálním prostředí byla tato vzdálenost prodloužena alespoň na délku osy Y demonstrátoru. Byl tak vypočítán průměr přes celý spektrogram. Tento průměrný spektrogram vidíme na obrázku 7.4. Odečtením průměrného spektrogramu od spektrogramu, který je viditelný na obrázku 7.2, došlo k separaci nespojitostí viditelných v obrázku 7.3.

Metoda průměrování nám v experimentálních podmínkách umožnila separaci nespojitostí ve sněhu od rovinných rozhraní.



Obrázek 7.3: Nespojitosti - nahoře kalibrační koule, dole dva kovové prvky v podlaze pod radarem



Obrázek 7.4: Amplitudové spektrum odečítaného průměrového signálu

Kapitola 8

Sestavení anténního diagramu

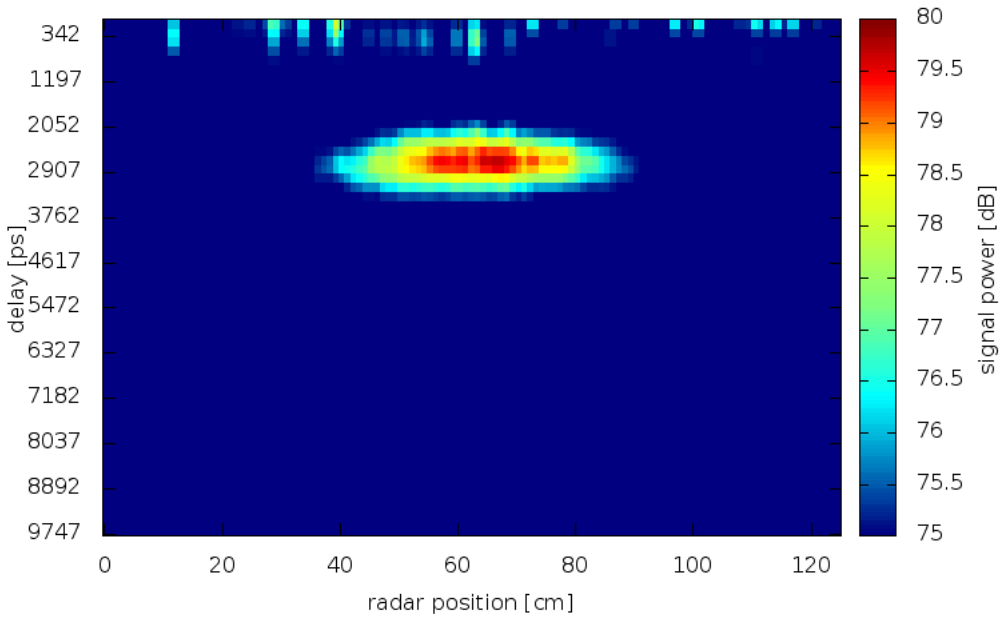
Sestavení anténního diagramu staví na metodách uvedených v [10] a shrnutých v [6].

Jedná se o měření v blízké zóně, která běžně využívají měřené a měrné antény. Při našich měřeních je druhá anténa nahrazena definovaným cílem a měření probíhá jen pomocí měřené antény a odrazu od tohoto cíle. Fourierovými transformacemi nad prostorem tří proměnných a prostorovými geometrickými transformacemi se tato měření přepočítají na anténní diagram.

Nutným vstupem pro tyto výpočty jsou měření, ve kterých figuruje jen jeden přesně definovaný cíl. V praxi nelze takový signál přímo naměřit. Za prvé je obtížné zajistit volný prostor, ve kterém by byl jen určitý cíl, a za druhé samotný systém vytváří mnoho vlastních odrazů.

Vlastní odrazy lze celkem úspěšně potlačit odečtením signálu, který byl získán namířením radaru do nebe a zaznamenáním tohoto signálu. Tato metoda byla využívána dříve k odstranění vlastních odrazů při výpočtech hloubky sněhu. Vlastní odrazy jsou však velmi závislé na teplotě a stárnutí systému.

Separace jednoho cíle byla provedena odečtením veškerého pozadí. Bylo tedy provedeno jedno měření bez cíle a druhé měření, kdy již byl cíl umístěn pod radarem. Tím bylo dosaženo výrazné potlačení všech nežádoucích prvků v signálu a zůstal jen jeden výrazný přesně definovaný cíl. Výsledek je vidět na obrázku 8.1.



Obrázek 8.1: Obraz kalibrační koule získaný odečtením měření prostoru s koulí a bez ní

8.1 Zpřesnění vzdálenosti od antény k cíli

K sestavení anténního diagramu je potřebné měření v síti bodů v rovině a známá poloha kalibračního cíle. Umístíme-li cíl na konkrétní souřadnice v prostoru, nemáme jednak zajištěnu definovanou rádiovou vzdálenost od antény k cíli a za druhé se nám cíl v okrajových polohách radaru ztrácí v šumu.

Bylo nutné sestavit metodu, která by umožňovala vyjádření vzájemné polohy antény radaru a definovaného cíle a zároveň by poskytla informaci o této vzdálenosti v místech, kde nelze jednoznačně určit polohu cíle v signálu.

Řešením je proložení určitelných vzdáleností matematickým modelem. Tento matematický model vzdáleností představuje hyperboloid charakterizovaný rovnicí 8.1.

$$d(x,y) = \sqrt{(x - T_x)^2 + (y - T_y)^2 + T_z^2} \tag{8.1}$$

$d(x,y)$ ... vzdálenost anténa-cíl $[m]$

x, y ... poloha radaru $[m]$

T_x, T_y, T_z ... poloha cíle $[m]$

Aproximace experimentálních dat představuje problém minimalizace kvadratických odchylek mezi experimentálními vzdálenostmi a matematickým modelem. Vycházíme z toho, že máme k dispozici vzdálenost anténa-cíl pro většinu bodů měření a hledáme vrchol hyperboloidu. V tomto vrcholu je umístěn cíl za předpokladu, že anténa je v bodě [0,0,0]. Tímto nezajistíme sice skutečné souřadnice antény a cíle, ale jen jejich vzájemnou vzdálenost, kterou potřebujeme k výpočtům.

8.1.1 Penalizační funkce

Předpis penalizační funkce je vztah 8.2. Hledán je bod $A[a_1, a_2, a_3, a_4, a_5]$ v prostoru s dimenzí 5. Prostor byl rozšířen o parametrizaci souřadnic, což se ukázalo jako vhodné.

$$P = \sum_r \sum_s \left(d_{exp}(r,s) - \sqrt{(a_1 x_r - a_2)^2 + (a_3 y_s - a_4)^2 + a_5^2} \right)^2 \tag{8.2}$$

d_{exp} ... experimentálně určené vzdálenosti anténa-cíl $[m]$

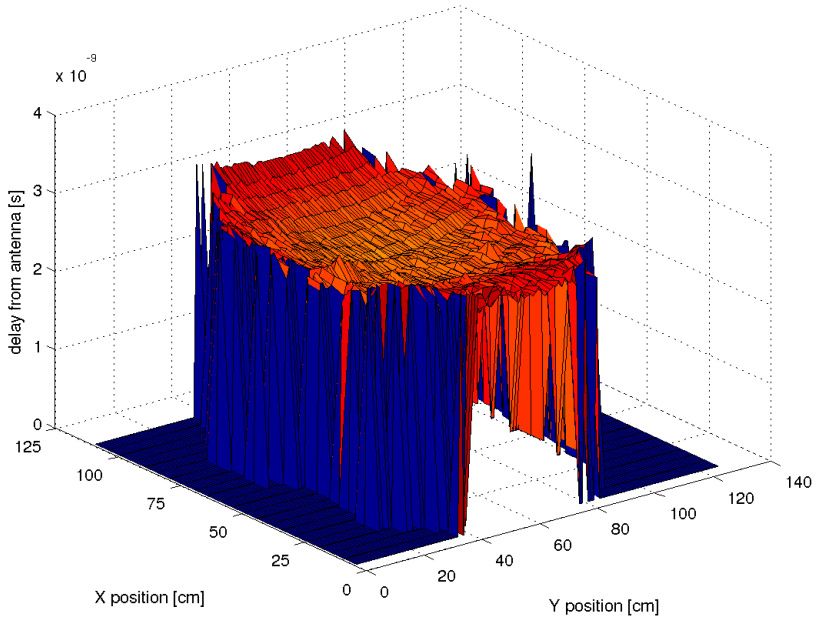
x_r, y_s ... poloha radaru $[m]$

a_1, a_2, a_3, a_4, a_5 ... souřadnice bodu A vyhovující podmínce minima $[-]$

Data $d_{exp}[m]$ jsou vzdálenosti určené z experimentálních dat. Viditelné jsou na obrázku 8.4. Níže v listingu je použito globální pole `hyperboloid(r,s)` obsahující tyto vzdálenosti.

Penalizační funkce

```
function [ penalty ] = hyperfce( a )

%T = [0.55,0.65,0.8]; %vrchol pro overeni

a0 = [1, 0.55, 1, 0.65, 0.8]
R = 22;
S~= 126;

h = zeros(R,S);
x = linspace(0,1.1,R);
y = linspace(0,1.25,S);
```



Obrázek 8.2: Vzdálenosti od antény k cíli v závislosti na poloze radaru

```
%hyperboloid = zeros(R,S);
%for r = 1:R
%  for s = 1:S
%      hyperboloid(r,s) = sqrt((x(r)-T(1)).^2 + (y(s)-T(2)).^2 + T(3).^2);
%  end
%end
%global hyperboloid;

global hyperboloid_aprox;
hyperboloid_aprox = zeros(R,S);
for r = 1:R
  for s~= 1:S
      hyperboloid_aprox(r,s) =
          sqrt((a(1)*x(r)-a(2)).^2 + (a(3)*y(s)-a(4)).^2 + a(5).^2);
  end
end

penalty = 0;
for r = 1:R
    for s~= 1:S
       if(hyperboloid(r,s) ~= 0)
       penalty = penalty + (hyperboloid(r,s) - hyperboloid_aprox(r,s)).^2;
    end
    end
end
```

```
end
```

8.1.2 Určení polohy cíle

Polohu antény lze prohlásit pro všechny odběhy za stejnou a byla určena ze signálu ručně.

Problematické je určení polohy kulového cíle, protože obzvlášť v krajních polohách se ztrácí v šumu.

Hledání bylo omezeno jen na malou oblast, kde by se cíl měl nacházet.

Experimentálně určeným prahováním bylo dosaženo použitelné detekce polohy cíle. V bodech měření, kde nebylo možno spolehlivě určit polohu, je vzdálenost vynulována. Nulové vzdálenosti se nevyužívají při aproximaci a není tak aproximace negativně ovlivněna špatným určením polohy cíle.

Nalezení polohy cíle

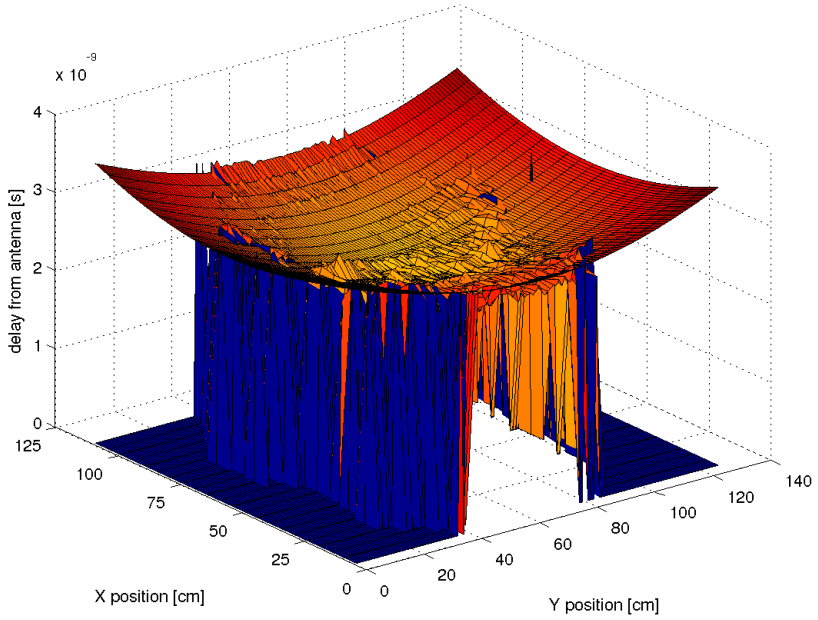
```
nasobeni = 8;
%nalezeni tau anteny v zavislosti na r,s
maxSa = zeros(R,S);
maxSk = zeros(R,S);
%mapa_platnych= zeros(R,S);
for r = 1:R
    for s~= 1:S
        Siga = abs(fft(squeeze(hbezkoule(r,s,:)),1024*nasobeni));
        Sigk = abs(fft(squeeze(w(r,s,:)),1024*nasobeni));
        Sa = Siga(1:(150*nasobeni));
        Sk = Sigk((138*nasobeni):(144*nasobeni));

        for iSk = 1:numel(Sk)
            if(Sk < 8e3) Sk(iSk) = 0;
            end
        end

        maxSa(r,s) = min(find(Sa == max(Sa)));
        maxSk(r,s) = min(find(Sk == max(Sk))) + 138*nasobeni - 1;
    end
end

maxSk = maxSk - 991;  %odecteni koaxialu k antene
maxSk(find(maxSk == min(min(maxSk))))=0;
```

8.1.3 Aproximace hyperboloidu

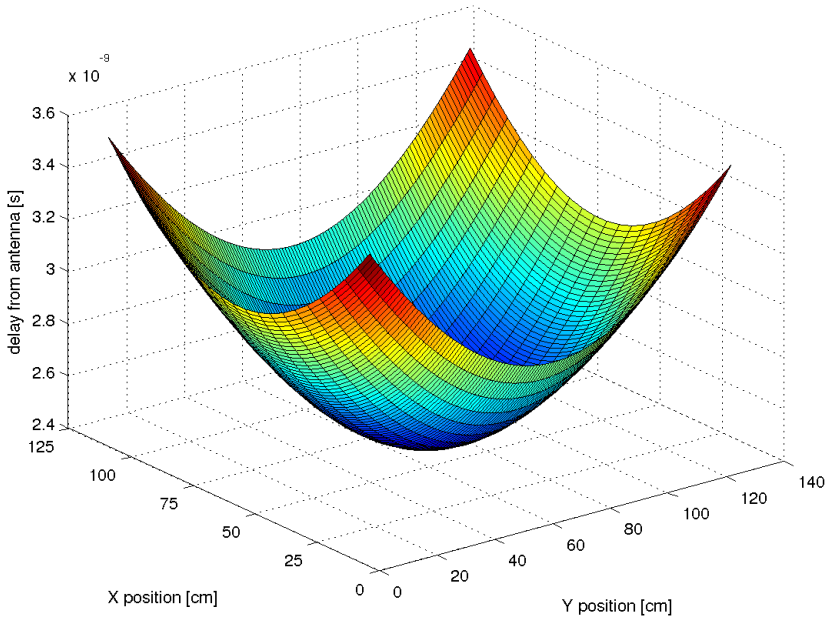
K samotnému nalezení minima kvadratických odchylek byla využita funkce `fminunc` z optimalizačních nástrojů Matlabu. Za počáteční bod hledání byla zvolena ideální poloha cíle, kam byl v souřadném systému demonstrátoru umístěn.

Aproximace hyperboloidu

```
fs = 250e3;
B = 730e6;
T=1e-3;

global hyperboloid;
hyperboloid = ((fs*T*c)/(8192*B*2))*maxSk;  %metry

a0 = [1, 0.55, 1, 0.65, 0.8]; %výchozí bod hledání
a = fminunc(@hyperfce,a0) %minimalizace funkce

x = linspace(0,1.1,R);
y = linspace(0,1.25,S);
hyperboloid_aprox = zeros(R,S);
for r = 1:R
    for s~= 1:S
         hyperboloid_aprox(r,s) = sqrt((a(1)*x(r)-a(2)).^2 +
                                        (a(3)*y(s)-a(4)).^2 + a(5).^2);
    end
end

figure
hyperboloid_aprox = hyperboloid_aprox/c; %prevod na tau (cas z prostoru)
surf(hyperboloid_aprox);
```



Obrázek 8.3: Proložení experimentálního hyperboloidu aproximovaným



Obrázek 8.4: Aproximovaný hyperboloid

8.2 Výpočet

Prvním krokem ve výpočtu je vytvoření komplexního analytického signálu Hilbertovou transformací. Hilbertova transformace měření z prostoru bez definovaného cíle je ve vztahu 8.3.

$$S_{1,r,s} = H\{s_{1,r,s}\} \tag{8.3}$$

s_1 ... odběhy naměřené na demonstrátoru $[-]$

S_1 ... vytvořený komplexní signál ze všech odběhů $[-]$

Vztah 8.4 transformuje měření s definovaným cílem.

$$S_{2,r,s} = H\{s_{2,r,s}\} \tag{8.4}$$

Pro odstranění nežádoucích cílů na pozadí a separaci jen definovaného cíle je nutné obě měření odečíst, vztah 8.5.

$$w = S_1 - S_2 \tag{8.5}$$

w ... komplexní signál se separovaným cílem $[-]$

Vztah 8.6 vypočítává pro každý bod měření zpoždění τ podle dříve nalezené aproximace.

$$\tau_{r,s} = \frac{\sqrt{(a_1 x_r - a_2)^2 + (a_3 y_s - a_4)^2 + a_5^2}}{c} \tag{8.6}$$

$\tau_{r,s}$ ... zpoždění signálu na cestě anténa-cíl $[s]$

S uvažovanou aproximací τ lze nyní vypočítat podle vztahu 8.7 funkci $g(r, s, m)$.

$$g(r, s, m) = w(r, s, m) \cdot e^{-\mathbf{j}\omega_m \tau_{r,s}} \tag{8.7}$$

$g(r, s, m)$ ... přenosová funkce $[-]$

Velikost vlnového vektoru lze vypočítat ze vztahu 8.8.

$$|\overrightarrow{k_m}| = \frac{\omega_m}{c} \tag{8.8}$$

k_m ... vlnové číslo $[m^{-1}]$

Následně je nutné zjistit směrové složky vektoru pro definovaný směr. To umožňuje výpočet 8.9 pro sférické souřadnice.

$$\begin{aligned} k_{x,m} &= |k_m| \cdot \cos\Phi \cdot \cos\Theta \\ k_{y,m} &= |k_m| \cdot \cos\Phi \cdot \sin\Theta \\ k_{z,m} &= |k_m| \cdot \cos\Phi \end{aligned} \tag{8.9}$$

$k_{x,m}, k_{y,m}, k_{z,m}$ ... ortogonální složky vlnového vektoru

Pro jednotlivé frekvence je na základě vztahu 8.10 možné vypočítat směrový diagram.

$$W_{m,\phi,\theta} = \sum_{r,s} g(r,s,m) \cdot e^{\mathbf{j}(k_{x,m}x + k_{y,m}y + 2k_z z)} \tag{8.10}$$

Výpočet diagramu

```
M=256

fstart = 230e6;
fstop = 960e6;

dx = 0.05;
dy = 0.01;

c=3e8;

xt = 0.55;
yt = 0.65;
zt = 0.8;
T = [xt,yt,zt];

R = 22;
S~= 126;

omega_m = 2*pi*linspace(fstart,fstop,M);

%             x   y   m

hbezkoule = zeros(R,S,M);
hskouli = zeros(R,S,M);

for i = 1:(R*S);
    hbezkoule(rec_bezkoule(i,1)/5+1,rec_bezkoule(i,2)+1,:) =
```

```
                                          hilbert(rec_bezkoule(i,3:end));
    hskouli(rec_skouli(i,1)/5+1,rec_skouli(i,2)+1,:) =
                                            hilbert(rec_skouli(i,3:end));
end

w = (hskouli) - (hbezkoule);

a=[0.9256, 0.5132, 0.9442, 0.5994, 0.7340];

tau_rs = zeros(R,S);
for r = 1:R
    for s~= 1:S
        tau_rs(r,s) = sqrt((a(1)*x(r)-a(2)).^2 +
                                       (a(3)*y(s)-a(4)).^2 + a(5).^2)/c;
    end
end

g = zeros(R,S,M);
for r = 1:R
    for s~= 1:S
        for m = 1:M
            g(r,s,m) = w(r,s,m)*exp(-1j*omega_m(m)*tau_rs(r,s));
        end
    end
end

ntheta = 90;
nphi = 45;

W = zeros(256,ntheta,nphi);

k~= omega_m/c;
theta = linspace(-pi,pi,ntheta);
phi = linspace(0,pi/2,nphi);

for m = 1:M
for itheta = 1:ntheta
for iphi = 1:nphi
    W(m,itheta,iphi) = 0; %TODO: m->1

    kx = k(m) * cos(phi(iphi)) * cos(theta(itheta));
    ky = k(m) * cos(phi(iphi)) * sin(theta(itheta));
    kz = k(m) * cos(phi(iphi));

    for r = 1:R
        for s~= 1:5:S
            x = T(1)-(r-1)*0.05;
            y = T(2)-(s-1)*0.01;
```

```
            z~= T(3);

            W(m,itheta,iphi) = W(m,itheta,iphi) +
                               g(r,s,m)*exp(1j*(kx*x + ky*y + 2*kz*z));
        end
    end
end
end
end

for itheta = 1:numel(theta)
    for iphi = 1:numel(phi)
        [x(iphi,itheta),y(iphi,itheta), z(iphi,itheta)] =
     sph2cart(theta(itheta),phi(iphi),abs(squeeze(W(1,itheta,iphi))));
    end
end
```

Vypočítaná funkce $W_{m,\phi,\theta}$ je uložena do souborů pro jednotlivé frekvence m, aby mohla být dále zobrazena za tímto účelem vytvořenou aplikací.

Kapitola 9

Aplikace pro zobrazení 3D diagramu

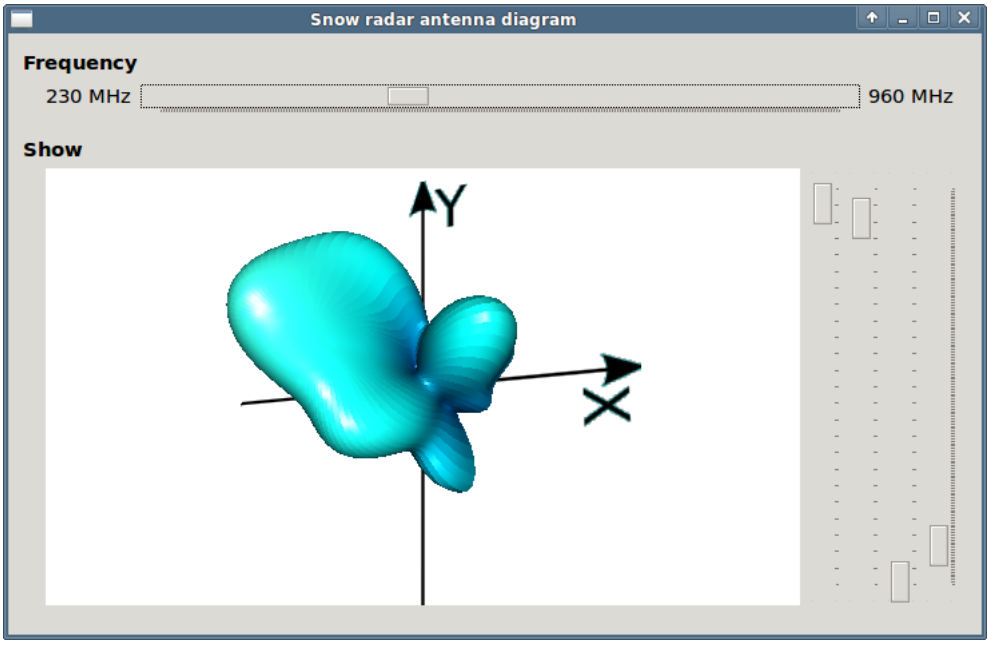
K zobrazování směrových diagramů byl nejprve použit Matlab, ale ukázal se jako nevhodný. Vykreslování trojrozměrných grafů s větším množstvím bodů velmi vytěžovalo prostředky počítače a zabíralo nepřiměřené množství paměti. Bylo také zapotřebí s diagramy manipulovat a zobrazovat plynule pro různé frekvence.

Požadavky na manipulaci Matlab splňuje, ale překreslení diagramu po jakémkoli úkonu trvá nepříjemně dlouho. K přepínání diagramů pro různé frekvence by vyžadovalo vytvoření GUI v Matlabu a za předpokladu pomalého vykreslování diagramů je bezcenné. Nikdo nechce minuty čekat než se mu zobrazí nový diagram.

Za tímto účelem byl vytvořen software využívající hardwarovou podporu zobrazování ve 3D. Uživatelské prostředí bylo vytvořeno s použitím multiplatformní knihovny QT. Zobrazování využívá OpenGL.

Vstupem do tohoto programu jsou soubory vytvořené z výpočtů v Matlabu. Po zapnutí programu jsou načteny informace s digramy v celém frekvenčním rozsahu radaru. Je tak dosaženo velmi rychlého vykreslování diagramů při operacích rotace, přiblížení a přepnutí frekvence. Vykreslování ve srovnání s Matlabem je řádově rychlejší. Z času vykreslení téměř v minutách je dosaženo zkrácení na zlomky vteřin.

Okno vytvořené aplikace je vidět na obrázku 9.1.



Obrázek 9.1: Okno aplikace pro zobrazení diagramu antény ve 3D

9.1 Hlavní vlákno programu

Jako v každém programu v jazyce C, i zde je spuštěna nejprve funkce `main`.

Na začátku této funkce dojde k načtení souborů vygenerovaných při předchozích výpočtech v Matlabu. Paměťový prostor pro tato data je vytvořen staticky.

Po načtení diagramu je normalizována jeho velikost. Normalizace probíhá tak, že je nalezena největší úroveň amplitudy ve sférických souřadnicích a touto úrovní jsou následně poděleny všechny ostatní úrovně amplitud. Dojde tak k tomu, že maximální hodnota amplitudy je nyní rovna jedné. Rozsahu $< 0; 1 >$ odpovídá celé zobrazení i barevná škála, kterou je diagram kolorován.

Následně je zavolán konstruktor třídy `QApplication`. Dále je běh programu již v moci QT.

9.2 Vytvoření GUI

Grafické uživatelské prostředí aplikace je vytvořeno s využitím dříve zmíněné knihovny QT. Je implementováno jako třída dědící od třídy `QWidget`.

Jednotlivé prvky okna jsou dynamicky tvořeny programem a není zde využito žádného nástroje pro konstrukci uživatelského prostředí.

9.3 Třída OpenGL zobrazení

Aby mohlo být využito akcelerace zobrazení s OpenGL, musí být vytvořen prostor, kam bude výstup směřovat. Tento princip je v QT implementován jako třída `QtOpenGL`, od které dědí naše třída `GLWidget`.

V následujícím listingu je část hlavičkového souboru deklarující třídu `GLWidget`.

glwidget.H

```
class GLWidget : public QGLWidget
 {
     Q_OBJECT

 public:
     GLWidget(QWidget *parent = 0);
     ~GLWidget();

     QSize minimumSizeHint() const;
     QSize sizeHint() const;
     GLuint texture[1];// Ukládá texturu

 public slots:
     void setXRotation(int angle);
     void setYRotation(int angle);
     void setZRotation(int angle);
     void setZoom(int ztrans);
     //void chaneData();
     void loadData(int);


 signals:
     void xRotationChanged(int angle);
     void yRotationChanged(int angle);
     void zRotationChanged(int angle);
     void zoomChanged(int ztrans);

 protected:
     void initializeGL();
     void paintGL();
     void resizeGL(int width, int height);
     void mousePressEvent(QMouseEvent *event);
     void mouseMoveEvent(QMouseEvent *event);
```

```
    void wheelEvent(QWheelEvent *event);

private:
    GLuint makeObject();
    void quad(GLdouble x1, GLdouble y1, GLdouble x2, GLdouble y2,
              GLdouble x3, GLdouble y3, GLdouble x4, GLdouble y4);
    void extrude(GLdouble x1, GLdouble y1, GLdouble x2, GLdouble y2);
    void normalizeAngle(int *angle);
    int frekvence;


    GLuint object;
    int xRot;
    int yRot;
    int zRot;
    int zoom;
    QPoint lastPos;
    QColor trolltechGreen;
    QColor trolltechPurple;
    static GLubyte gimp_image[];


};
```

9.4 Obsluha událostí myši

O obsluhu událostí v hlavním okně se stará samotné QT v rámci jednotlivých prvků GUI. Pro OpenGL widget je nutné obsluhu událostí od myši a klávesnice implementovat.

Při stisku levého tlačítka a tažení myší v okně zobrazení diagramu dojde k rotaci diagramu v ose X a Y. Při stisku pravého tlačítka a následného tažení dojde k rotaci v ose X a Z. Diagram je možné přibližovat a oddalovat kolečkem myši.

Aby došlo k patřičnému nastavení ovládacích prvků v hlavním okně, emituje `GLWidget` signály, které jsou napojeny na tyto ovládací prvky.

`GLWidget` je zároveň posluchačem, jsou zde vytvořeny sloty externích signálů. Tímto je zajištěna interakce nastavovacích prvků okna se zobrazením.

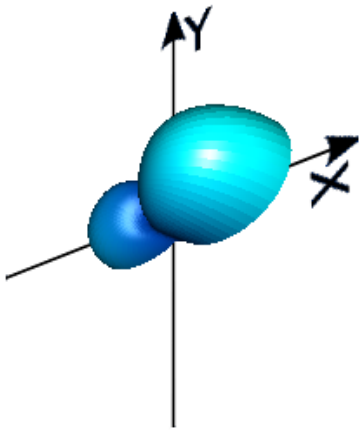
9.5 Modelování diagramu antény

Diagram je v paměti programu tvořen body kartézského souřadného systému, jsou tedy tvořeny souřadnicemi v osách X,Y a Z. Tyto body však pocházejí z kulového systému a podle něj jsou uspořádány.

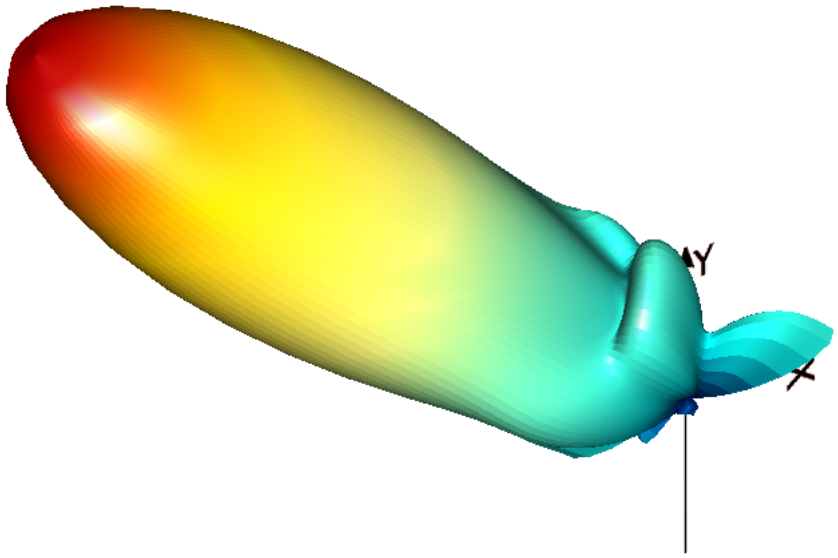
Dle sférického uspořádání probíhá i vykreslení. Diagram je tvořen pásy ze čtyřúhelníků. První čtyřúhelník pásu začíná body s indexem azimutu n a $n + 1$ a pro tyto dvě množiny azimutu jsou vykresleny všechny body s různou elevací. Potom pro všechny indexy azimutu je z pásu složen celý diagram.

Jednotlivé body mají barvu, která jim je přiřazena dle barevné škály. Jednotlivé čtyřúhelníky jsou vykresleny v podobě plochy vyplněné barevným přechodem. Odstín výplně čtyřúhelníku ovlivňují barvy jeho rohových bodů.

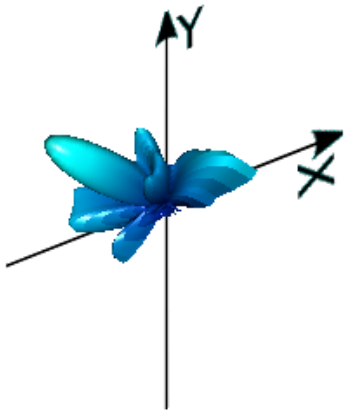
Ukázka vytvořených diagramů pro frekvence ze začátku, z prostředka a konce pásma je na obrázcích 9.2, 9.3 a 9.4.



Obrázek 9.2: Anténní diagram pro frekvenci 230 MHz



Obrázek 9.3: Anténní diagram pro frekvenci 680 MHz



Obrázek 9.4: Anténní diagram pro frekvenci 960 MHz

Kapitola 10

Závěr

V počátečním stádiu práce byl upraven software v signálovém procesoru radaru a ovládací software na PC. Signálovému procesoru zůstala jen úloha řízení radaru a starost o přenos navzorkovaného signálu směrem do PC. V tomto PC s původním upraveným softwarem se spojovala skupina vzorků z jednoho odběhu s informací o poloze radaru a vše se odeslalo po síti do jiného PC, kde probíhal sběr dat a jejich další zpracování.

Pro sběr dat byl vytvořen jednoduchý konzolový program, který umožňuje seskupit odběhy z jednoho přesunu radaru po celé délce osy Y. Tímto programem byly vytvářeny soubory v podstatě představující řezy typu B pro různé polohy v ose X. Program sloužil ke sběru dat ze všech měření.

Pro ověření postupů při zpracování bylo využito Matlabu, ale samotné zpracování velkých objemů dat probíhalo ve vlastním softwaru napsaném v jazyce C. Zpracování se jednak výrazně urychlilo a zároveň se vytvořila cesta pro snadnou implementaci výpočtů přímo do radaru v budoucnu.

Jedním z úkolů této práce je nalezení metody pro odhalení nespojitostí ve sněhu. Pro funkci sněžného radaru jsou jako cíle důležitá jen plošná rozhraní mezi dielektrickými prostředími. Diskontinuity se podařilo nalézt v řezech typu B, které byly pořízeny na demonstrátoru. Byl vypočítán průměr amplitudových spekter přes určitou vzdálenost, kterou rolba ujede. V průměrovém signálu zůstala jen rozhraní a diskontinuity byly potlačeny. V laboratorních podmínkách je tato metoda dostatečná,

ale ve skutečnosti by bylo nutné optimalizovat délku průměrového filtru, protože rozhraní nemusejí být rovnoběžná s pohybem rolby, jako je tomu v laboratoři.

Dalším úkolem bylo sestavení anténního diagramu. K tomuto úkolu byla nutná dvě měření přes celý prostor demonstrátoru. Jedno měření proběhlo s přesně definovaným cílem v prostoru a druhé bez něj. První měření se odečetlo od druhého a výsledek byl takový, jako by byl v prostoru jen osamocený přesně definovaný cíl. Nyní bylo možné při výpočtech využít upravenou metodu pro měření antén v blízké zóně. K tomuto měření se běžně využívá dvou antén. Zde byla použita jen jedna anténa a definovaný odraz signálu.

Na základě výše uvedeného byl vypočítán anténní diagram pro konečný počet frekvencí z pracovního rozsahu radaru. K zobrazení tohoto diagramu byla napsána aplikace využívající hardwarové akcelerace grafiky. S použitím této aplikace je možné pohodlně prohlížet všechny vypočítané diagramy v trojrozměrném prostoru. Aplikace zároveň demonstruje budoucí možnosti zobrazování prozkoumávaného prostoru s využitím trojrozměrných technik.

Výsledky této diplomové práce umožňují další postup při vývoji radaru pro měření hloubky sněhu a především otevírají nové možnosti jeho využití v podobě podpovrchového scanneru s trojrozměrným zobrazením.

Literatura

[1] *ADSP-21369*. [online, cit. 1.8.2011].
URL `http://www.analog.com/en/processors-dsp/sharc/adsp-21369/processors/product.html`

[2] *FFTW homepage*. [online, cit. 21.8.2011].
URL `http://www.fftw.org/`

[3] *Gnuplot homepage*. [online, cit. 21.8.2011].
URL `http://www.gnuplot.info/`

[4] *perlit.cz - Expandovaný perlit*. [online, cit. 26.8.2011].
URL `http://www.perlit.cz/expand_perlit.php`

[5] *Relativní permitivita - fyzikální tabulky*. [online, cit. 26.8.2011].
URL `http://www.converter.cz/tabulky/relativni-permitivita.htm`

[6] *SRF02 Ultrasonic range finder Technical Specification*. [online, cit. 24.8.2011].
URL `http://www.robot-electronics.co.uk/htm/srf02tech.htm`

[7] Capalini, R.: *Další možnosti využití UWB senzoru*. Pardubice: Steinel Technik, 2011.

[8] Murphy, E.: *ADI - Analog Dialogue, DDS*. [online, cit. 1.8.2011].
URL `http://www.analog.com/library/analogdialogue/archives/38-08/dds.html`

[9] Reichrt, V.: *Dílčí zpráva k projektu Sněhový radar*. 2004.

[10] Schejbal, V.: *Šíření vln v blízké a vzdálené zóně antény - 10. seminář*. Univerzita Pardubice.

[11] Tišnovský, P.: *Grafická knihovna OpenGL (1)*. [online, cit. 21.8.2011].
URL `http://www.root.cz/clanky/graficka-knihovna-opengl-1/`

[12] Tišnovský, P.: *Grafická knihovna OpenGL (11): vykreslovací řetězec*. [online, cit. 21.8.2011].
URL `http://www.root.cz/clanky/opengl-11-vykreslovaci-retezec/`

[13] Tišnovský, P.: *Grafická knihovna OpenGL (3) - Základní geometrické prvky*. [online, cit. 21.8.2011].
URL `http://www.root.cz/clanky/opengl-3-zakladni-geometricke-prvky/`

[14] Šebesta, J.: *Radiolokace a radionavigace*. Brno: Vysoké učení technické v Brně, 2004, ISBN 80-214-2482-6.

Obsah CD

Obsah přiloženého CD je následující:

- soubor `dp.pdf` - technická zpráva ve formátu PDF
- adresář `tex` - zdrojové soubory technické zprávy
- adresář `soft_sber` - software pro sběr dat
- adresář `soft_plot` - software pro zobrazení řezů GPR
- adresář `soft_3D` - software pro zobrazení anténního diagramu ve 3D
- adresář `matlab_skript` - skripty v jazyce Matlab