VŠB - TECHNICKÁ UNIVERZITA OSTRAVA

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2011 **Radim Štěrba**

VŠB - TECHNICKÁ UNIVERZITA OSTRAVA

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

KATEDRA MĚŘICÍ A ŘÍDICÍ TECHNIKY

Mobilní aplikace pro vizualizaci externího signálu na PDA zařízení v reálném čase (digitální osciloskop)

Mobile Application for Visualization of External Signal on PDA device in Real Time (Digital Oscilloscope)

Ostrava, 2011 Radim Štěrba

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal, a nejsou mi známy žádné okolnosti, které by mohly vést k pochybnostem o mé práci.

Radim Štěrba

Datum odevzdání bakalářské práce: 6. 5. 2011

Poděkování

Chtěl bych poděkovat vedoucímu mé bakalářské práce panu Ing. Ondřeji Krejcarovi, Ph.D. za poskytnuté materiály, rady a konzultace, které mi v průběhu práce poskytl. Rovněž bych velmi rád poděkoval panu Ing. Jakubovi Jirkovi za velice cenné rady a velice cenné konzultace v oblasti programování mobilních aplikací.

Abstrakt:

Cílem bakalářské práce je vytvořit mobilní aplikaci pro vizualizaci externího signálu na PDA zařízení v reálném čase.

V textu práce je uveden postup při vývoji aplikace, nahrávání audio signálu, zpracování a nakonec následné vykreslení na displej PDA zařízení. Celá aplikace běží ve vláknech, jejichž použití je v textu rovněž popsáno. Součástí řešení je krátký úvod do problematiky, vysvětlení pojmů jako PDA, .NET Compact Framework nebo Windows Mobile. Dále tato práce řeší tvorbu uživatelských komponent v programovacím jazyce C#, konkrétně řeší vývoj komponenty vykreslující grafy. Na závěr je provedeno testování aplikace pro různé frekvence vstupního audio signálu a zhodnocení dosažených výsledků.

Výsledkem této práce je funkční komponenta pro vykreslování grafů pro platformu .NET Compact Framework. Velkou výhodou takto vytvořené komponenty je využitelnost v dalších projektech a aplikacích. Hlavním výsledkem celého snažení je aplikace digitálního osciloskopu, která je schopná věrně vizualizovat v reálném čase externí audio signál až do frekvence 3000Hz.

Klíčová slova:

.NET Compact Framework, C#, Windows Mobile, PDA, digitální osciloskop

Abstract:

The aim of this work is to create mobile application for visualization of the external signal on the PDA in real time.

In the thesis the procedure of application development is described, recording an audio signal, its processing and subsequent render on the PDA screen at the end. The whole application runs in threads whose application is also described in the text. Part of the solution is also a short introduction to this issue and explanations of the terms such as PDA, NET Compact Framework or Windows Mobile. Furthermore, this work deals with custom components creation in C#, specifically the development of component which draws the graphs. At the end, the application was tested for different frequency of the input audio signal and the results were analysed.

The result of this work is a operational component for drawing the graphs for NET Compact Framework. The great advantage of this component is its utility in other projects and applications. And the most significant result of the effort is the application of a digital oscilloscope that is able to visualize the external audio signal in a real time and accurately up to the frequency of 3000Hz.

Key words:

.NET Compact Framework, C#, Windows Mobile, PDA, digital oscilloscope

Seznam použitých obrázků:

Obsah:

1. Úvod

Bakalářská práce je zaměřena na problematiku vizualizace externího signálu za pomoci PDA zařízení s operačním systémem Windows Mobile na platformě .NET Compact Framework. Jedná se tedy o vývoj jednoduchého digitálního osciloskopu.

Začátek je věnován obecné části práce, kdy obsahem druhé kapitoly s názvem Úvod do problematiky je obecný popis PDA, osciloskopu, operačního systému Windows Mobile a platformy .NET Compact Framework. V kapitole se dále zabývám programovacím jazykem C# a přináším stručný popis vývojového prostředí Microsoft Visual Studia.

Další kapitoly se zabývají teoretickou částí práce. Popsal jsem v nich tvorbu aplikací pro Windows Mobile a zpracovávání audio signálu v C#. V praktické části jsem se zaměřil na tvorbu uživatelských komponent v jazyce C# pro Windows Mobile a vytvoření samotné aplikace pro vizualizaci externího audio signálu. Závěr je věnován testování aplikace a popsání dalších možností vývoje aplikace.

Dnešní PDA zařízení disponují poměrně výkonným hardwarem, mají rychlé procesory s taktovací frekvencí okolo 1GHz a operační paměti v řádech stovek MB. Dalšími přednostmi jsou velké displeje a velmi dobrá konektivita. Mojí motivací pro tuto práci byla aplikace VIRTINS Pocket Oscilloscope. Náhled této aplikace je na obrázku 1. Chtěl jsem vytvořit aplikaci, která bude schopna vizualizovat externí audio signál, a na které bude možné stavět při dalším vývoji jednoduchého, kapesního osciloskopu.

Obrázek 1: Náhled aplikace VIRTINS Pocket Oscilloscope. [1]

V kapitole 2, s názvem Úvod do problematiky, jsem se zabýval obecným popisem pojmů, které jsou spojeny s touto prací. Jsou to pojmy jako PDA, osciloskop, platforma :NET Compact Framework nebo třeba programovací jazyk C#. Návrh aplikace pro vizualizaci měřeného signálu, tedy kapitola 3, přináší blokové schéma aplikace s popisem, uvedl jsem zde princip snímání zvuku a jeho další zpracování. Ve čtvrté kapitole jsem popsal vlastní komponentu pro vykreslování grafů, popsal jsem použití vláken v aplikaci a uživatelské rozhraní. Další, v pořadí pátá kapitola, popisuje testování aplikace a společně s šestou kapitolou přináší dosažené výsledky. Popsal jsem zde, jak probíhalo testování a také do jaké nejvyšší frekvence je aplikace schopná věrně vizualizovat externí audio signál. V závěru jsem shrnul celou práci, jak aplikace vznikala, jaké byly problémy při testování nebo jaké jsou možnosti dalšího vývoje a využití aplikace.

2. Úvod do problematiky

Cílem druhé kapitoly je seznámení s pojmy. Je zde napsáno několik málo slov o PDA, Windows Mobile a osciloskopu. Dále je zde popsán programovací jazyk C# a platforma .NET Compact Framework. U jazyka C# je napsáno, k čemu ho lze použít, stejně jako něco málo k historii jazyka. Je zde uveden krátký popis vývojového prostředí Microsoft Visual Studio a nakonec je popsána tvorba aplikací pro Windows Mobile.

2.1. PDA (Personal Digital Assistent)

Mobilní elektronika nám přináší uživatelský komfort a ulehčuje nám život. PDA, mobilní telefony a různé komunikátory jsou dnes normální součástí našich životů. V důsledku velkého technického vývoje jsou PDA v dnešní době velmi výkonné. Disponují výkonnými procesory s vysokou taktovací frekvencí, velkou operační pamětí v řádu stovek Mega Bytů a také mají velké barevné displeje s vysokým rozlišením. Po komunikační stránce jsou PDA vybavena nejčastěji rozhraním USB. Z bezdrátové komunikace zvládají komunikovat pomocí IrDa, Bluetooth, WiFi, GMS, GPRS, UMTS.

Dnešní PDA dokážou přehrávat video nebo hudbu, umožňují surfování na internetu, umí určit polohu a navigovat pomocí integrovaného GPS přijímače. Díky výkonným procesorům a velkým operačním pamětem na nich lze spouštět i velmi náročné aplikace. Ovšem ne vždy tomu tak bylo. Původně měly PDA za úkol pomáhat s plánováním úloh, organizováním času a správou kontaktů. Kromě multimediálního obsahu lze na PDA využívat kancelářských aplikací (Microsoft Word, Microsoft Excel a další). [2]

Po softwarové stránce jsou PDA v našich zeměpisných šířkách nejčastěji postaveny na platformě Pocket PC a vybaveny operačním systémem Windows Mobile. Ovšem není to jediný možný operační systém. Lze se setkat s operačním systémem Palm OS. A v poslední době je na velkém vzestupu také operační systém Android společnosti Google. Operační systém Android je postavený na Linuxu. Obrázek 2 nám ilustruje, jak může takové PDA vypadat.

Obrázek 2: Ukázka PDA zařízení. [3]

2.2. Osciloskop

Co je to vlastně osciloskop? Osciloskop je v zásadě univerzální měřicí přístroj. S jeho pomocí lze zobrazit měřený signál a následně měřit amplitudu, frekvenci, periodu, a v neposlední řadě třeba fázi signálu. Osciloskopy se dají rozdělit na analogové a digitální. Digitální osciloskop měří signál přivedený na vstup pouze v daných okamžicích. Pomocí A/D převodníku se signál upraví do digitální podoby, uloží v paměti a následně zobrazí na displeji. Osciloskopy bývají většinou vícekanálové. Lze pak porovnávat více měřených signálů. Toho lze s výhodou využít například u zesilovačů, kdy se porovnává signál na vstupu a signál na výstupu zesilovače. Porovnáním těchto dvou průběhů lze zjistit, jaké má zesilovač zesílení, zkreslení. Obecně lze dvoukanálový osciloskop využít na měření vstupního a výstupního signálu libovolného reálného dvojbranu.

Moderní digitální osciloskopy jsou velké, těžké a závislé na zdroji elektrické energie. Jsou to přístroje, které jsou přizpůsobeny pro práci v laboratoři. Ovšem pro práci v terénu je zapotřebí mít vhodné nástroje, které jsou pokud možno co nejvíce komplexní.

Právě spojením PDA a osciloskopu vzniká takovýto komplexní nástroj. PDA vydrží na jedno nabití relativně dlouho, naměřené údaje lze okamžitě zapisovat do připravené tabulky, případně lze pracovat s různými datasheety a manuály.

2.3. Windows Mobile

Windows Mobile je operační systém firmy Microsoft, založený na Windows CE. Operační systém je určen pro mobilní zařízení, jako jsou mobilní telefony, kapesní počítače. Vzhled Windows Mobile je podobný jako u klasických Windows pro desktop.

Verze Windows Mobile:
Windows Mobile 2003
Windows Mobile 5.0
Windows Mobile 6.0
Windows Phone 7.0

Platforma Windows Phone 7.0 je nástupce starší platformy Windows Mobile. Tato platforma byla vydaná na konci roku 2010. Náhled, jak vypadá Windows Mobile 5.0, ukazuje obrázek 3 na straně 5.

Windows Phone 7.0 je úplně jiný, než předchozí verze Windows Mobile. Windows Phone 7.0 je oproti předcházejícím verzím velice inovativní jak v designu, tak v ovládání. Pro ovládání se již nepoužívá stylus, ale prst nebo prsty. Windows Phone 7.0 podporuje multitouch. Nové Windows Phone 7.0 už nemají klasické tlačítko start, ani tlačítko s křížkem v pravém horním rohu. Bez použití stylusu by se tyto tlačítka ovládala velice špatně, a tak bylo ovládání zcela změněno. Systém nepodporuje technologii Adobe Flash a tato technologie je nahrazena technologií Microsoft Windows Silverlight. Velkou novinkou je také to, že systému chybí

multitasking. Všechny aplikace půjde instalovat pouze z marketplace společnosti Microsoft. Pro práci samozřejmě zůstávají nástroje jako Microsoft Word, Excel nebo Powerpoint.

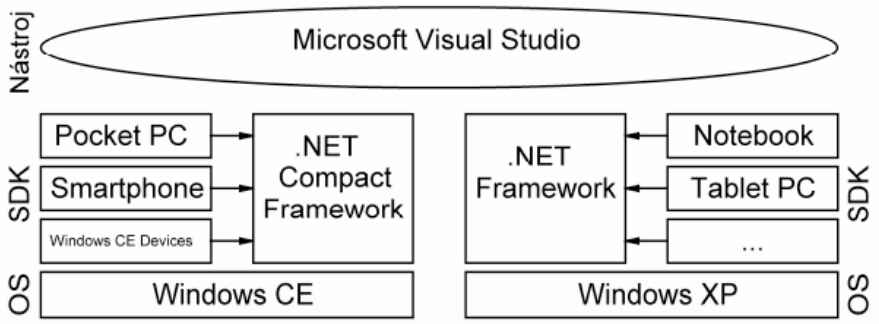
Obrázek 3: Ukázka Windows Mobile 5.0. [4]

2.4. .NET Compact Framework

.NET Compact Framework je technologická platforma, vycházející z platformy .NET Framework. [5] .NET Compact Framework je podmnožinou .NET Frameworku. To znamená, že .NET Comapct Framework neobsahuje všechny funkce základního .NET Frameworku. Obsahuje pouze ty funkce, u kterých lze dodržet kompatibilitu v rámci možností mobilního zařízení. Jedná se především o zobrazovací možnosti zařízení. .NET Compact Framework je verze určená pro běh na mobilních zařízeních (PDA, mobilní telefony), které používají Windows Mobile. Tato technologie je využita v set-top boxech a podporuje ji i herní konzole Xbox 360. První verze .NET Compact Frameworku 1.0 byla vydána v druhé polovině roku 2002. Poslední dostupná verze je pak z roku 2009 a jedná se o verzi 3.7. :NET Compact Framework používá některé shodné knihovny tříd jako klasický .NET Framework a také pár knihoven designovaných speciálně pro mobilní zařízení.

Aplikace pro .NET Compact Framework lze vyvíjet s pomocí nástroje Microsoft Visual Studio. Poslední a zároveň nejnovější verze Microsoft Visual Studia je verze 2010. Aplikace lze psát v programovacích jazycích C# nebo Visual Basic .NET. Tato práce je věnována aplikaci napsané v programovacím jazyce C#, a proto se bude práce dále zabývat už jen tímto jazykem.

Z pohledu architektury můžeme situaci .NET Compact Frameworku vyjádřit následujícím obrázkem. Obrázek 4 představuje přehled nástrojů a možností pro vývoj aplikací.



Obrázek 4: Přehled nástrojů a možností pro vývoj aplikací. [6]

Typy .NET Frameworků:

Microsoft .NET Framework: je nejrozšířenější platforma pro osobní počítače s operačním systémem Microsoft Windows od verze Windows 98.

Microsoft .NET Compact Framework: je platforma určená pro kapesní počítače a mobilní telefony s operačním systémem Windows Mobile.

Microsoft .NET Micro Framework: je platforma určená pro embedded zařízení, s ještě menší výpočetní kapacitou a většími omezeními, než kapesní počítače. [7]

2.5. Programovací jazyk C#

Jazyk C# [8] je vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý firmou Microsoft zároveň s platformou .NET Framework. Jazyk C# čerpá syntaxi z jazyka C a je založen na jazycích C++ a Java. Mohlo by se zdát, že jazyk C# má svého jasného předka v jazyku C++, ale právě naopak má C# blíže k jazyku Java, než k C++. V jazyce C# zanikly některé nepopulární záležitosti, jako byly pointery. Pointery nahradily v novém jazyce odkazy. C# je možné využít k tvorbě formulářových aplikací ve Windows, softwaru pro mobilní zařízení (PDA, mobilní telefony), webových služeb, webových aplikací nebo k tvorbě databázových programů.

Programátorům ulehčuje život hlavně Garbage Collector. Garbage Collector je část běhového prostředí programovacího jazyka nebo přídavná knihovna podporovaná kompilátorem. Má za úkol automaticky určit, která část paměti programu se nepoužívá, a připravit ji pro další použití.

V jazyce C# neexistují žádné globální proměnné a metody. Stejně tak neexistuje vícenásobná dědičnost. To znamená, že každá třída může být potomkem pouze jedné třídy. C# je case sensitive, je tedy třeba dávat pozor na psaní velkých a malých písmen. Jazyk rozlišuje velké a malé písmena.

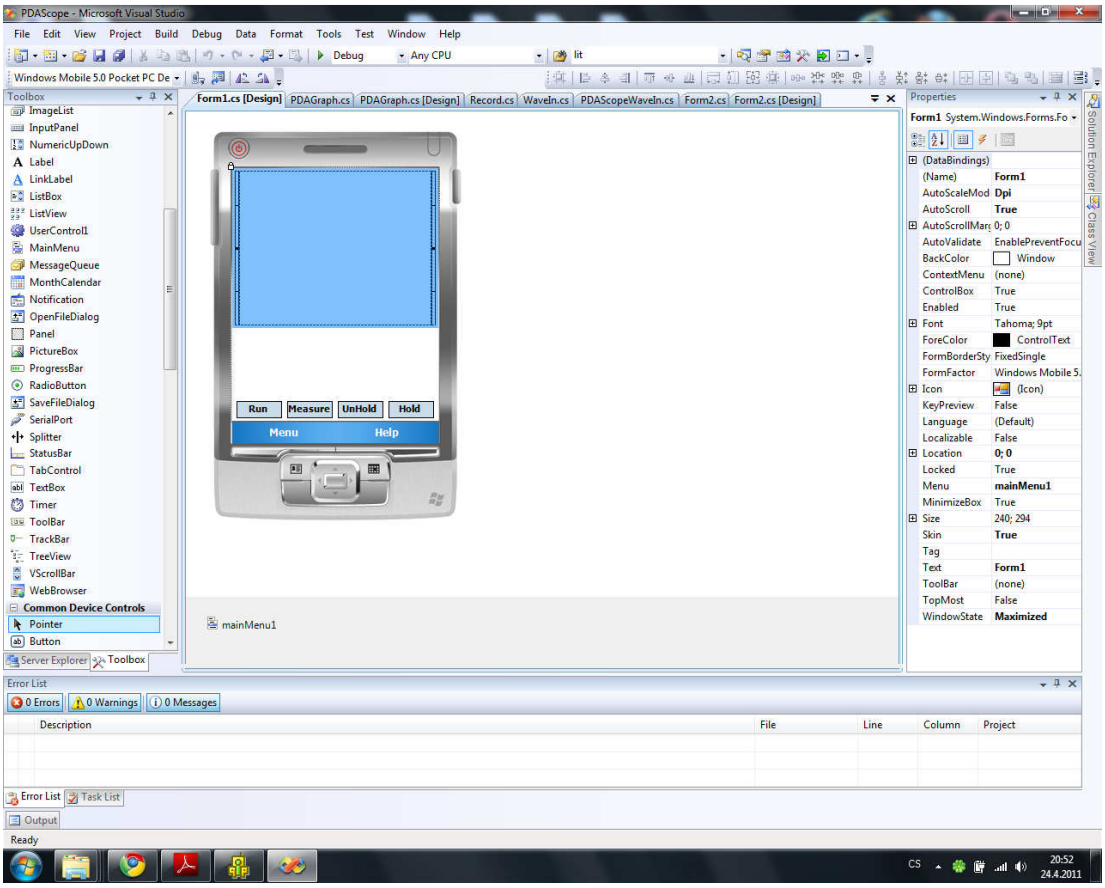
2.6. Microsoft Visual Studio

Jedná se o vývojové prostředí od firmy Microsoft. Vývojové prostředí může být použito pro návrh a vývoj aplikací jak pro stolní počítače, tak pro mobilní zařízení. Jednoduše lze vyvíjet konzolové aplikace, aplikace s grafickým rozhraním nebo třeba webové aplikace. Od verze 2008 je k dispozici ve Visual Studiu technologie LINQ (Language Integrated Query). Tato technologie umožňuje vývojářům jednoduchý a rychlý přístup k datům a práci s nimi. Od stejné verze 2008 je rovněž možné vybrat mezi verzemi .NET, pro které budeme vyvíjet aplikaci. Popisu prostředí Microsoft Visual Studia se tato práce věnovat nebude, není to totiž její hlavní náplní.

2.7. Tvorba aplikací pro Windows Mobile

Postup při vývoji aplikací pro mobilní zařízení s využitím .NET Compact Framework je až na malé odlišnosti stejný jako při vývoji desktopových aplikací pro .NET Framework. Rozdíl je v tom, že je k dispozici méně programátorských možností a taky velikost formuláře je o dost menší. Je tedy třeba zvolit sofistikovaný způsob rozvržení prvků ve formuláři, to znamená více šetřit místem. Displej kapesního počítače totiž není ani zdaleka tak velký, jako monitor stolního počítače nebo displej notebooku. Při tvorbě klasické Windows Forms aplikace jsou k dispozici obvyklé prvky: Button, CheckBox, Label, ListBox, RadioButton, TextBox a další.

Náhled, jak vypadá vývojové prostředí Microsoft Visual Studia 2008, pro programování mobilních aplikací ukazuje obrázek 5 na straně 8. Nacházejí se zde úplně stejné nabídky, jako při programování aplikací pro desktopy. Na levé straně je nabídka Toolbox, kde jsou k dispozici všechny komponenty. Na pravé straně je nabídka Properties, kde se mění jednotlivé vlastnosti komponent a také je zde nabídka Solution Explorer. Úplně dole je pak Error List, kde nás vývojové prostředí informuje o případných chybách v kódu.



Obrázek 5: Náhled Microsoft Visual Studia 2008.

3. Návrh aplikace pro vizualizaci měřeného signálu

Obsahem této kapitoly je již konkrétní představa o aplikaci. Kapitola tedy řeší blokové schéma, popis blokového schématu a popis jednotlivých částí aplikace. Je zde popsáno snímání externího audio signálu a jeho zpracování na dvourozměrné pole bodů x, y. V rámci kapitoly o snímání audio signálu je objasněn princip vzorkování.

3.1. Blokové schéma, popis aplikace

Na obrázku 6 je vidět blokové schéma aplikace. První blok představuje snímání signálu, to znamená, že je zde řešena práce s hardwarem, v tomto případě se zvukovou kartou a vestavěným mikrofonem. Pro záznam zvuku se používá mikrofon, jenž převádí zvuk na elektrický signál. O zpracování audio signálu se stará algoritmus, který spolupracuje s třídami Microsoft pro zpracování zvuku. Vykreslení pak zajišťuje komponenta, která je popsána v páté kapitole s názvem implementace navrženého řešení.

Obrázek 6: Blokové schéma aplikace PDAScope.

První blok blokového schématu představuje část programu, která se stará o snímání vnějšího signálu z vestavěného mikrofonu. O komunikaci s mikrofonem se stará třída WaveIn.cs. V tomto prvním bloku je externí analogový signál převeden na diskrétní signál, který nabývá hodnot 0 až 255. Úroveň analogového signálu tedy reprezentují diskrétní hodnoty 0 až 255, přičemž analogovou nulu (na výstupu z mikrofonu není žádný signál) reprezentuje diskrétní hodnota 128. Diskrétní hodnoty 0 až 127 reprezentují záporné výchylky a diskrétní hodnoty 129 až 255 kladné výchylky analogového signálu.

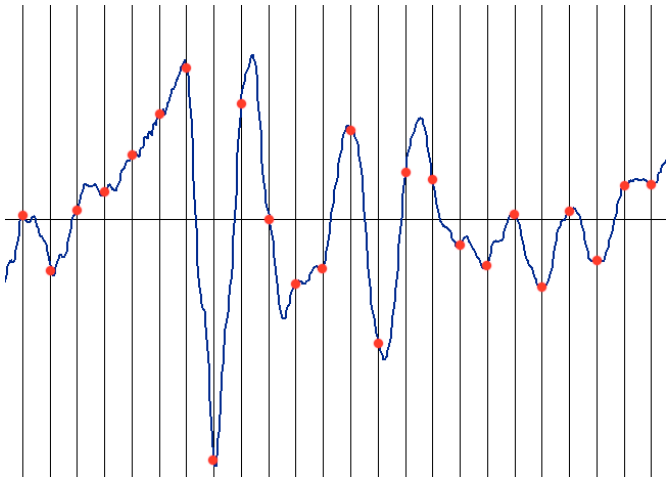
Blok zpracování audio signálu představují třídy PDAScopeWaveIn.cs a Record.cs. Třída PDAScopeWaveIn.cs vytváří ze surových dat pole vzorků. Ve Form1.cs se vytvoří kopie pole, se kterou se pak dále pracuje. Tato kopie slouží jako data pro vykreslování a další výpočty.

V poslední části dochází k vykreslování nasnímaného průběhu na obrazovku PDA. Vykreslování zajišťuje uživatelská komponenta, která je popsána dále v textu. Komponenta obsahuje pro vykreslování dvě metody AddGraph a DeleteGraph. Při překreslení se jako parametr předá metodě AddGraph pole vzorků, a to je pak vykresleno. Vykreslený průběh se nechá na chvíli vykreslený na displeji (20ms) a poté je volána metoda DeleteGraph. Tato metoda vykreslený průběh smaže. Toto se neustále s velkou rychlostí překreslování cyklicky opakuje.

3.2. Snímání analogového signálu

Mikrofon je k PDA připojen pomocí zvukové karty. Vstupní signál je samozřejmě analogový a je třeba jej převést na digitální pomocí A/D převodníku. Tento převod se uskutečňuje pomocí vzorkování (sampling). Vzorkování signálu je proces, kdy dochází k diskretizaci signálu v jeho časové oblasti. V každém časovém okamžiku je zaznamenána aktuální hodnota signálu (vzorek). Na obrázku 7 jsou tyto vzorky znázorněny červenými kolečky. Z obrázku je patrné, že vzorkováním ztratíme z původního signálu mnoho detailů. I při zvýšení vzorkovací frekvence dostáváme pouze jen množinu diskrétních bodů. Vzorkováním se zabývá Shannonův teorém, který říká, že přesná rekonstrukce spojitého, frekvenčně omezeného, signálu z jeho vzorků je možná tehdy, pokud byl vzorkován frekvencí alespoň dvakrát vyšší, než je maximální frekvence rekonstruovaného signálu. V praxi se tedy vzorkovací frekvence volí dvakrát větší plus ještě malá rezerva než je maximální požadovaná přenášená frekvence. [9]

Člověk je schopen vnímat frekvence od 20Hz do 20kHz. Vzorkovací frekvence by tedy měla být nejméně 40kHz. V praxi se obvykle volí vzorkovací frekvence 44100Hz. V aplikaci PDAScope je nastavena kvalita nahrávání na 8 bitů na vzorek a vzorkovací frekvence je nastavena na 22050Hz. Toto nastavení je provedeno ve třídě WaveIn.cs a lze ho změnit. Je třeba ovšem respektovat hardwarové možnosti PDA zařízení, jako výpočetní výkon a zobrazovací schopnosti displeje.



Obrázek 7: Ukázka vzorkování signálu. [10]

Co se týká mikrofonů, existuje mnoho typů mikrofonů. Každý typ je založen na jiném fyzikálním principu a má proto i jiné vlastnosti. V současné době jsou nejpoužívanějším typem mikrofony elektrostatické, též někdy nazývané jako kondenzátorové. Tento typ se vyznačuje velkou vstupní impedancí, vyrovnanou vstupní charakteristikou, vysokou citlivostí. Elektrostatické mikrofony jsou považovány za jedny z nejkvalitnějších a používají se často pro profesionální záznam. V kapesních počítačích ovšem takové velice kvalitní mikrofony pro

profesionální záznam zvuku nejsou. Ovšem vestavěné mikrofony kapesních počítačů jsou dostatečně kvalitní například pro snímání mluveného slova.

Kondenzátorový mikrofon pracuje tak, že akustické kmity rozechvívají membránu, která je jednou z elektrod kondenzátoru, připojeného do elektrického obvodu. V rytmu změny polohy membrány se mění kapacita kondenzátoru. Tato změna se převádí na elektrický signál buď tak, že je vlastní mikrofonní vložka napájena z velmi měkkého zdroje polarizačního napětí a napětí na ní je snímáno předzesilovačem s velkou vstupní impedancí, nebo je kapacita vložky použita jako součást vysokofrekvenčního oscilátoru, rozlaďovaného změnou kapacity, a v následujících obvodech je demodulován nízkofrekvenční signál. [11]

3.3. Zpracovávání analogového signálu

Pro komunikaci s hardwarovým zařízením, v tomto případě s vestavěným mikrofonem, je zapotřebí použít třídy Microsoftu. To zajistíme příkazem: using Wave; ve zdrojovém kódu aplikace. V případě aplikace PDAScope potřebujeme zvuk nahrávat a tudíž použijeme třídu WaveIn.cs. [12] Kromě třídy WaveIn.cs se v aplikaci požívají i uživatelsky definované třídy Record.cs a PDAScopeWaveIn.cs.

Třída WaveIn.cs, která zajišťuje komunikaci s vestavěným mikrofonem, je upravena pro účely aplikace PDAScope. Je zde doplněna metoda GetRawData(), která poskytuje surová data. Jedná se o hodnoty v rozsahu 0 až 255, které představují nahraný analogový signál.

Kód metody GetRawData():

```
/// <summary>
/// Get raw data to let the user defined class process them.
/// This method is only accesible to classed that derive from WaveIn class.
/// </summary>
/// <returns>Buffers with recorded data.</returns>
public Wave.WAVEHDR[] GetRawData()
{
    if (!m_inited)
        return new Wave.WAVEHDR[1];
    if (m_recording)
        Stop();
    return m_whdr;
}
```

Třída PDAScopeWaveIn.cs uživatelská třída napsaná pro speciálně pro aplikaci digitálního osciloskopu. Třída PDAScopeWaveIn.cs dědí ze třídy WaveIn.cs. Výpis kódu třídy PDAScopeWaveIn.cs je uveden níže. Tato třída zpracovává surová data a předává je dále třídě Record.cs na další zpracování.

Kód třídy PDAScopeWaveIn.cs:

```
class PDAScopeWaveIn : WaveIn
    {
        /// <summary>
        /// PDAScopeWaveIn inherited from class WaveIn. Processes the recorded data
        /// and transmits to the class Record, to the next processing.
        /// </summary>
        /// <returns>field of byte</returns>
        public byte[] getRawData()
        {
            Wave.WAVEHDR[] m_whdr = m_file.GetRawData();
            byte[] readData;

            readData = new byte[m_whdr[0].dwBytesRecorded];
            Marshal.Copy(m_whdr[0].lpData, readData, 0, readData.Length);

            return readData;
        }
    }
```

4. Implementace navrženého řešení

Aplikace pro vizualizaci externího signálu byla napsána v jazyce C# pro operační systém Windows Mobile. Veškeré testování probíhalo na zařízení Fujitsu-Siemens N560, kde je nainstalován operační systém Windows Mobile 5.0. Rozlišení displeje tohoto zařízení je 640x480 bodů.

Aplikace obsahuje celkem 3 formuláře (okna). V prvním formuláři probíhá vizualizace signálu, druhý formulář obsahuje informace a nápovědu k programu a třetí slouží pro nastavení. Celá aplikace je rozdělena do dvou vláken, jedno vlákno realizuje záznam externího audio signálu a druhé pak vizualizaci, tzn. vykreslování na displej PDA zařízení.

4.1. Popis vlastní komponenty vykreslující grafy

Při každém vykreslení vizualizovaného signálu se do komponenty posílají informace o barvě křivky, o tloušťce křivky a hlavně se posílají informace o tom, jak má křivka vypadat. Tyto informace představuje dvourozměrné pole, které obsahuje souřadnice bodů x, y, ze kterých je průběh složen.

Pro vykreslování naměřených průběhů je využita vlastní komponenta s názvem PDAGraph. Komponenta byla napsána speciálně pro aplikaci digitálního osciloskopu. Komponenta je napsána v jazyce C# a obsahuje 6 základních metod pro práci s komponentou. Všechny metody jsou popsány níže.

Komponenta je napsána tak, aby se přizpůsobila velikostem displeje každého PDA zařízení. Je tedy nutné vždy znát hodnoty šířky a výšky panelu, do kterého se vykresluje průběh. Tyto rozměry lze zjistit pomocí dvou metod getPanelWeight a getPanelHeight. Metody vždy vrátí celočíselnou hodnotu velikosti v pixelech.

Vykreslení dvou vertikálních měřítek na pravé a levé straně okna probíhá při spuštění aplikace. O vykreslení se starají dvě metody panel2_Paint a panel3_Paint. Obě metody jsou popsány níže.

Metoda AddGraph

AddGraph je metoda, která přímo vykresluje grafy. AddGraph má 3 argumenty: pole bodů, šířka čáry a barva čáry. Prvním argumentem je pole bodů, to znamená souřadnice x, y jednotlivých bodů průběhu. Druhý argument představuje šířku čáry, datový typ je Integer, zadává se tedy celé číslo. Poslední argument je barva čáry, datový typ je Color. Metoda se volá: *PDA_Graph.AddGraph(points, 3, Color.Blue);*. Points je pole bodů, 3 představuje šířku průběhu (šířka průběhu bude 3 body) a Color.Blue je barva průběhu (v tomto případě modrá). Metoda AddGraph se volá při každém překreslení obrazovky společně s metodou DeleteGraph. Překreslení probíhá takovým způsobem, že se jako první zavolá metoda AddGraph, která vykreslí průběh. Pak se vlákno, které realizuje vykreslování, uspí na 20 ms, aby bylo možné

vykreslený průběh vidět, a hned se smaže pomocí metody DeleteGrafph. Toto se neustále opakuje a je tím dosaženo kontinuálního vykreslování.

Kód metody AddGraph:

```
/// <summary>
/// Very importatnt method. This method draws the chart in the panel. Due to
/// continuous redrawing of the screen, the method AddGraph is called cyclically
/// together with the method DeleteGraph.
/// </summary>
/// <param name="points">field points [x,y]</param>
/// <param name="graph_width">width of the waveform</param>
/// <param name="graph_color">color of the waveform</param>
public void AddGraph(Point[] points, int graph_width, Color graph_color)
{
    Graphics g = panel1.CreateGraphics();
    pencil.Color = graph_color;
    pencil.Width = graph_width;

    g.DrawLines(pencil, points);
    g.Dispose();
}
```

Metoda DeleteGraph

Tato metoda maže celý obsah grafu. Smaže vykreslený průběh. Metoda se volá při každém překreslení. Metoda nemá žádné argumenty, a proto stačí zavolat: *PDA_Graph.DeleteGraph();*.

Kód metody DeleteGraph:

```
/// <summary>
/// This is a method, that deletes the entire contents of the panel.
/// </summary>
public void DeleteGraph()
{
    Graphics g = panel1.CreateGraphics();
    g.Clear(Color.Transparent);
    g.Dispose();
}
```

Metoda panel2_Paint

Metoda panel2_Paint vykreslí vertikální měřítko na levé straně komponenty. Vertikální měřítko slouží pro lepší orientaci. Tato metoda se volá hned při startu, vertikální měřítko je proto vykresleno hned při spuštění aplikace.

Kód metody panel2_Paint:

```
/// <summary>
/// This method renders a left side vertical scale (level). Vertical
/// scale is rendered at startup application.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void panel2_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;

    Pen pen = new Pen(Color.Black, 3);
    Pen pen2 = new Pen(Color.Black, 1);

    g.DrawLine(pen, 0, panel2.Height / 2, panel2.Width, panel2.Height / 2);

    g.DrawLine(pen2, panel2.Width - 1, 0, panel2.Width - 1, panel2.Height - 1);
    g.DrawLine(pen2, 0, 0, 0, panel2.Height);

    for (int i = panel2.Height / 2 + 50; i < panel2.Height; i = i + 50)
    {
        g.DrawLine(pen2, 0, i, panel2.Width, i);
    }
    for (int i = panel2.Height / 2 - 50; i > 0; i = i - 50)
    {
        g.DrawLine(pen2, 0, i, panel2.Width, i);
    }
}
```

Metoda panel3_Paint

Jedná se o obdobnou metodu, jakou je panel2_Paint. Metoda panel3_Paint vykresluje vertikální měřítko na pravé straně komponenty.

Kód metody panel3_Paint:

```
/// <summary>
/// This method renders a right side vertical scale (level). Vertical
/// scale is rendered at startup application.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void panel3_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;

    Pen pen = new Pen(Color.Black, 3);
    Pen pen2 = new Pen(Color.Black, 1);

    g.DrawLine(pen, 0, panel2.Height / 2, panel2.Width, panel2.Height / 2);

    g.DrawLine(pen2, panel2.Width - 1, 0, panel2.Width - 1, panel2.Height - 1);
    g.DrawLine(pen2, 0, 0, 0, panel2.Height);

    for (int i = panel2.Height / 2 + 50; i < panel2.Height; i = i + 50)
    {
        g.DrawLine(pen2, 0, i, panel2.Width, i);
    }
    for (int i = panel2.Height / 2 - 50; i > 0; i = i - 50)
    {
        g.DrawLine(pen2, 0, i, panel2.Width, i);
    }
}
```

Metoda getPanelWidth

Jedná se o metodu, která vrací šířku panelu, do kterého se vykresluje průběh. Každé PDA má jinou velikost displeje a hlavně rozlišení. A proto je nutné vracet velikost komponenty, do které se provádí vykreslování. Vrací se jak šířka, tak výška. Obě metody getPanelWeight i getPanelHeight jsou bezparametrové a vrací vždy celé číslo jako informaci o rozměru panelu v pixelech.

Zdrojový kód metody getPanelWeight:

```
/// <summary>
/// Each display is different, to identify the size, this method returns the
/// width of the panel.
/// </summary>
/// <returns>panel width</returns>
public int getPanelWidth()
{
    return this.panel1.Width;
}
```

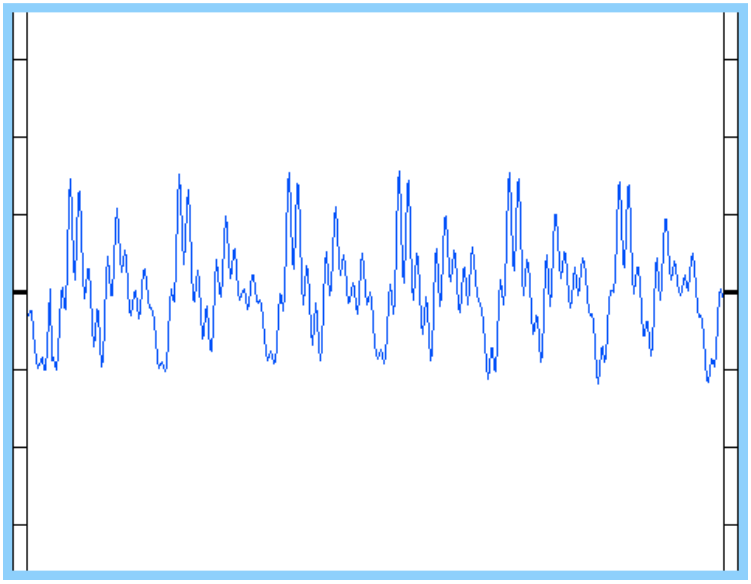
Metoda getPanelHight

Metoda getPanelHeight slouží pro vrácení hodnoty výšky panelu, do kterého se vykresluje snímaný signál. Jak bylo popsáno o pár řádků výše každé PDA zařízení má jinou velikost displeje a je tedy nutné dynamicky nastavovat parametry pro vykreslování.

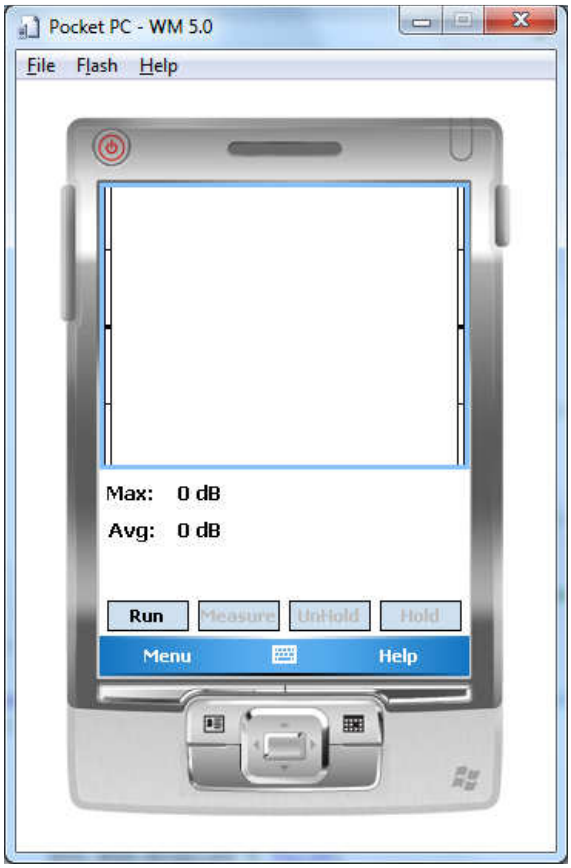
Kód metody getPanelHeight:

```
/// <summary>
/// Each display is different, to identify the size, this method returns the
/// height of the panel.
/// </summary>
/// <returns>panel height</returns>
public int getPanelHeight()
{
    return this.panel1.Height;
}
```

Náhled výše popisované komponenty je na obrázku 8. Nejdůležitější je bílá plocha, kde se vykreslují snímané průběhy. Barvy, kterými lze průběh vykreslovat, lze měnit v nastavení. Implicitně je nastavena modrá barva. Na stranách se nachází vertikální měřítko. Na obrázku 9 je ukázka náhledu celé aplikace spuštěné v emulátoru.



Obrázek 8: Náhled komponenty s vykresleným průběhem.



Obrázek 9: Náhled aplikace spuštěné v emulátoru.

4.2. Uživatelské rozhraní

Celá aplikace obsahuje 3 obrazovky (formuláře). Po spuštění aplikace se zobrazí první obrazovka, kde probíhá samotná vizualizace externího audio signálu a měření. Odtud se lze přepínat na další dvě obrazovky. Jedná se o obrazovku nastavení a obrazovku s nápovědou a informacemi k aplikaci.

Obrazovka 1:

Tato první obrazovka se zobrazí při spuštění programu. Celou horní polovinu zabírá komponenta, která vizualizuje externí audio signál. Na stranách této komponenty jsou pro lepší orientaci zobrazena vertikální měřítka. Pod touto komponentou se zobrazují základní naměřené hodnoty. Jedná se o hodnoty maxima, minima a průměrné hodnoty. Ve spodní části se pak nacházejí ovládací tlačítka a nabídky Menu a Help.

Pro přepnutí do nastavení (obrazovka 2) slouží nabídka Menu > Settings. Zde lze provést příslušné nastavení. Pro získání informací o aplikaci a nápovědě (obrazovka 3) slouží nabídka Help > Information / Help.

Obrazovka 2:

Zde lze provést nastavení aplikace. Jediné nastavené, které lze provést, je změna barvy vykreslovaného průběhu. Je možno si zvolit z pěti barev: žlutá, červená, zelená, modrá, černá. Přednastavenou barvou je barva modrá. Je to z toho důvodu, že hezky kontrastuje s bílým pozadím. Pro navrácení na hlavní obrazovku slouží nabídka Back v dolní liště.

Obrazovka 3:

Obrazovka 3 (Information / Help) slouží pro získání informací k aplikaci. Je zde k dispozici také nápověda, jak aplikaci ovládat a jak s ní pracovat. Pro navrácení na hlavní obrazovku slouží nabídka Back v dolní liště.

4.3. Popis funkčnosti

Při spuštění aplikace se zobrazí první obrazovka. Pro spuštění vizualizace je nutné kliknout na tlačítko Run, které se nachází ve spodní části obrazovky. Druhá možnost je využít nabídky Menu, kde se nachází stejnojmenná položka Run, která taktéž spustí vizualizaci. Nyní lze sledovat vizualizovaný externí audio signál na displeji PDA. Pro další práci s aplikací je nutné kliknout na tlačítko Hold. Tlačítko Hold zastaví vizualizaci a na displeji zůstane zobrazen průběh, který odpovídal času stisknutí tlačítka Hold. V tuto chvíli je možné provést základní měření kliknutím na tlačítko Measure. Při kliknutí na tlačítko Measure se v dolní polovině obrazovky zobrazí naměřené hodnoty maxima a průměrné hodnoty v decibelech. Tyto údaje se vztahují k právě zobrazenému vzorku dat. Pro opětovné spuštění vizualizace signálu je zapotřebí stisknout tlačítko UnHold. Kdykoliv za běhu aplikace lze provádět změny v nastavení (Settings) a procházet nápovědu (Information / Help).
Tlačítko **Run**: Kliknutím na tlačítko Run se spustí vizualizace externího audio signálu. Po kliknutí tlačítko zšedne, aby už nebylo možné na něj klikat.

Tlačítko **Measure**: Toto tlačítko slouží pro provedení základních měření. Tlačítko je aktivní až po stisku tlačítka Hold. Je to z toho důvodu, že měření se provádí z právě zobrazeného vzorku signálu. Kliknutím na tlačítko se na displeji zobrazí hodnoty maxima a průměrné hodnoty v decibelech.

Tlačítko **Hold**: Slouží pro zastavení vizualizace. Na displeji PDA zůstane vykreslený průběh signálu, který odpovídal času stisknutí tohoto tlačítka.

Tlačítko **UnHold**: Plní opačnou funkci než tlačítko Hold. Při kliknutí se opět začne vizualizovat externí audio signál na displeji PDA.

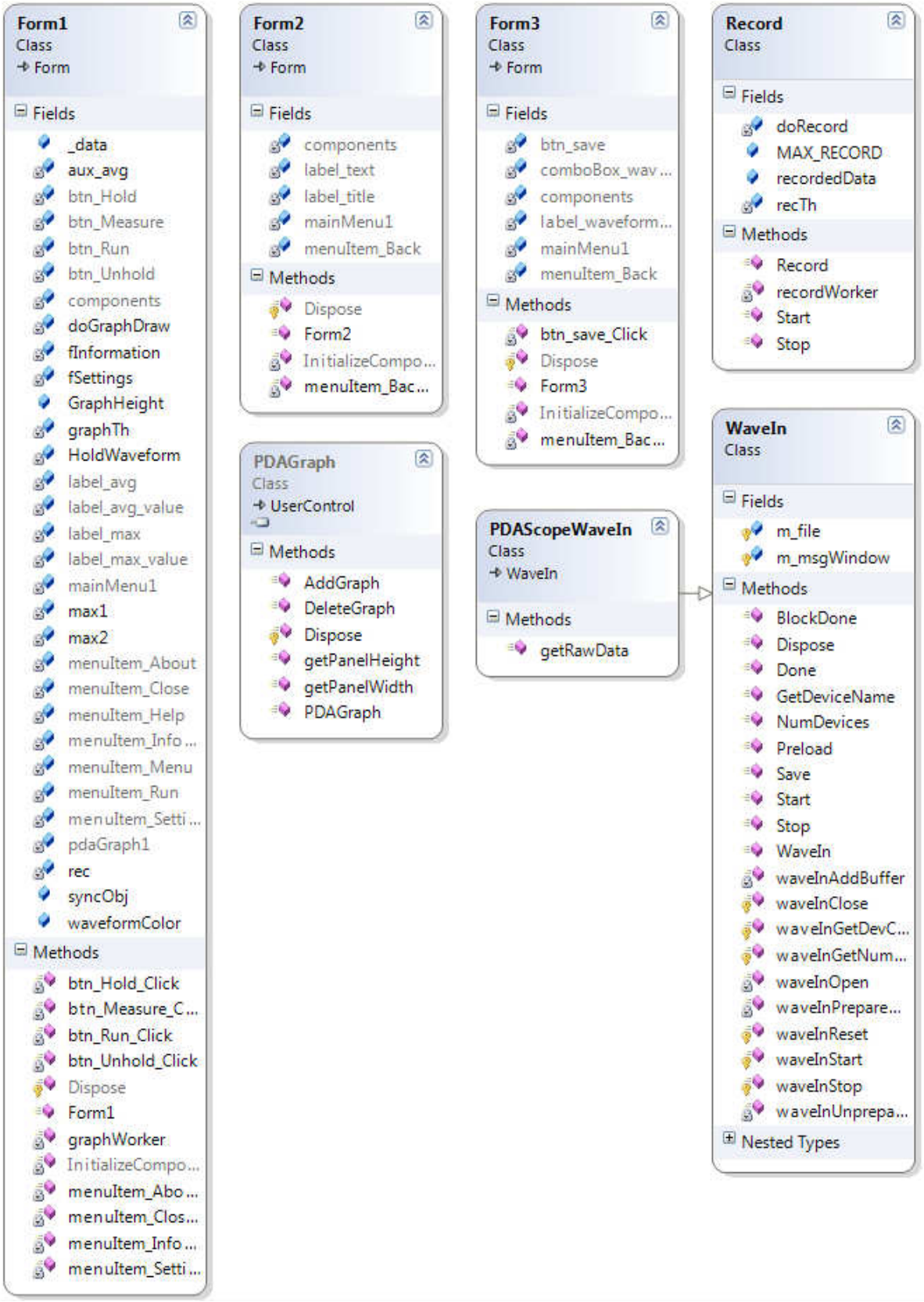
4.4. Aplikace vláken

Aplikace je napsána tak, že využívá vlákna. Konkrétně jsou v aplikaci 2 vlákna, která se starají o záznam externího audio signálu a vykreslování průběhu na obrazovku PDA.

Ve třídě Form1.cs je definován synchronizační objekt syncObj typu EventWaitHandle, který zajišťuje synchronizaci vláken. K přepínání kontext vláken dochází manuálně, je to dáno nastavením EventResetMode.ManualReset. Jako první se realizuje nahrávání, které zajišťuje vlákno s názvem recTh ve třídě Record.cs. Třída Record.cs obsahuje důležitou konstantu MAX_RECORD, která určuje čas, po který bude nahráván zvuk z vestavěného mikrofonu. Konstanta MAX_RECORD je nastavena na 30 ms. Po inicializaci nahrávání se zkontroluje dostupnost hardwarového zařízení, tedy vestavěného mikrofonu. Je-li vestavěný mikrofon k dispozici, začne nahrávání. Po dobu nahrávání, doba je dána konstantou MAX_RECORD, je vlákno recTh uspáno a následně se nahrávání ukončí. Po kontrole délky pole, kde jsou uložena data se pomocí metody Form1.syncObj.Set(); přepne kontext na vlákno graphTh. Vlákno graphTh běží ve třídě Form1.cs a vždy čeká až vlákno recTh ukončí nahrávání. Čekání na signál od vlákna recTh zajišťuje metoda Form1.syncObj.WaitOne();. Jako první jsou data překopírována do lokálního pole pomocí zámku (lock). Je to proto, aby nemohlo dojít k dvojímu přístupu do paměti, a aby byla aplikace thread-safe. Dále se pracuje už jen s lokálním polem dat. Poté se provedou výpočty spojené s měřením maxima a průměrné hodnoty, následně samotné vykreslení vzorku dat na displej PDA příslušnou barvou. Průběh zůstane na displeji vykreslen určitou dobu, která je zvolena jako MAX_RECORD – 10. Průběh zůstane vykreslen tedy 20ms. Následně se průběh smaže a kontext se pomocí metody Form1.syncObj.Reset(); předá zpět vláknu recTh a celá situace se opakuje znovu.

4.5. UML diagram aplikace PDAScope

UML, Unified Modeling Language, je v softwarovém inženýrství grafický jazyk pro vizualizaci, specifikaci, navrhování a dokumentaci programových systémů. [13] UML diagram aplikace PDAScope je na obrázku 10 na straně 21. Jsou zde zobrazeny jednotlivé třídy, které obsahuje projekt aplikace. U každé třídy jsou uvedeny všechny proměnné a metody, které daná třída obsahuje. V diagramu je hezky naznačená i dědičnost. Šipkou je naznačeno, že třída PDAScopeWaveIn.cs dědí ze třídy WaveIn.cs. Stejně jako všechny třídy je v UML diagramu uvedena i komponenta PDAGraph.



Obrázek 10: UML diagram tříd znázorňující třídy použité v aplikaci PDAScope.

5. Testování aplikace

Funkčnost aplikace byla testována v emulátoru Microsoft Visual Studia 2008 a hlavně na reálném zařízení Fujitsu-Siemens N560. Emulátor se nemusí vždy chovat úplně stejně jako reálné zařízení. Navíc se v této aplikaci přistupuje k hardwaru, je proto vždy lepší k testování využít reálné zařízení PDA.

Výrobce	Fujitsu-Siemens
Operační systém	Windows Mobile 5; Premium Edition
Typ procesoru	Intel PXA270; XScale
Frekvence procesoru	624 MHz
Paměť ROM	64 MB
Paměť RAM	128 MB
Displej	3,5 "; TFT; 65536 barev
Rozlišení displeje	640 x 480
Slot na paměťové karty	ano; SD
Mikrofon	ano; mono
Reproduktor	ano; mono
USB	ano; USB 1.1
Wi-Fi	ano
Bluetooth	ano; Bluetooth 1.2
IrDa	ano
GSM	ne
GPS	ano
Počet kanálů GPS	20
GPS chip	SiRF Star III
Baterie	Li-ion; 1200 mAh
Hmotnost	160 g
Rozměry	116 x 71 x 14 mm

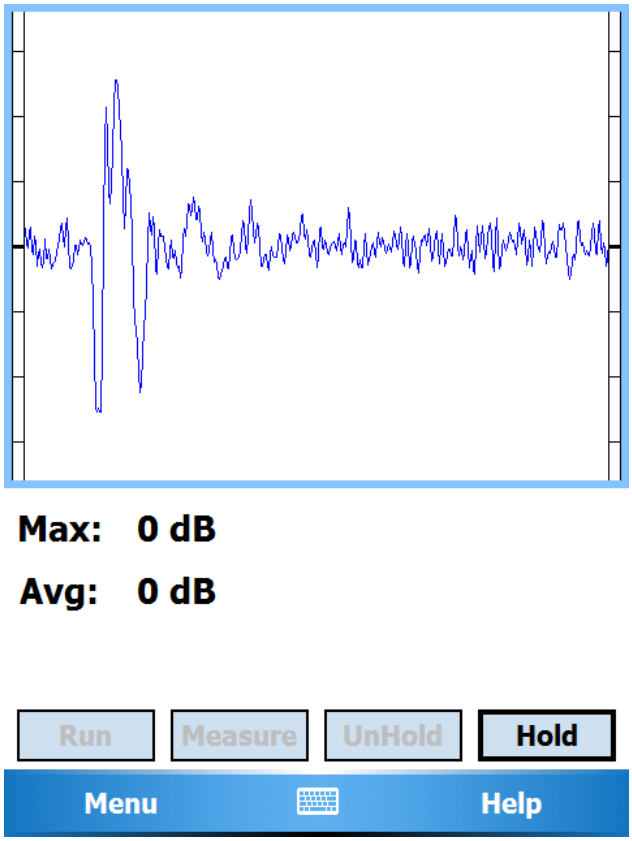
Tabulka 1: Parametry PDA Fujitsu-Siemens N560.

Tabulka 1 zobrazuje parametry PDA zařízení, na kterém bylo prováděno testování aplikace PDAScope. Zařízení je již několik let staré, a proto ho nelze srovnávat s dnešními nejmodernější PDA. Nejvíce limitující parametr je asi velmi malá operační paměť RAM.

První fáze testování spočívá ve vizualizaci předem nedefinovaných signálu, jako například lidské řeči. Pomocí tohoto testu bylo ověřeno, že aplikace PDAScope opravdu nějakým způsobem vizualizuje externí audio signál. Způsob testování ilustruje obrázek 11 na straně 23.

Obrázek 11: Princip první fáze testování aplikace PDAScope.

Na obrázku níže je printscreen obrazovky PDA při testování. Signál, vykreslený na obrázku 12, odpovídá průběhu lidské řeči.

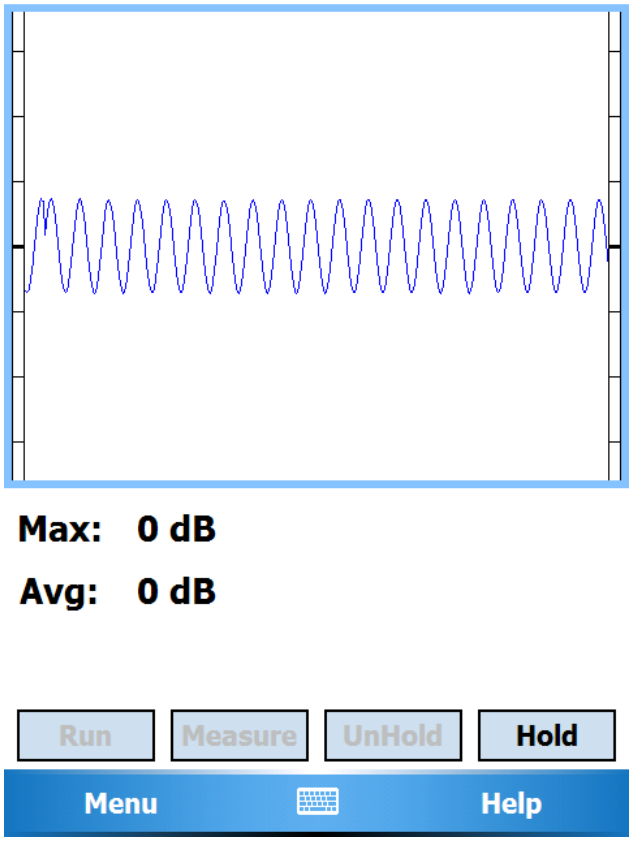


Obrázek 12: Printscreen obrazovky při první fázi testování.

Druhá fáze testování byla zaměřena na testování signálů, u kterých víme, jaký mají průběh v čase. Jedná se například o funkci sinus. K testování byl použit mobilní telefon. Z mobilního telefonu byl puštěn zvukový soubor ve formátu mp3, který představoval sinusový průběh o dané frekvenci. V tomto případě byla vybrána frekvence 1000Hz. Mobilní telefon byl tedy použit jako zdroj signálu (signálový generátor).

Aplikace na PDA vizualizovala sinusový průběh s drobnou chybou vždy na levém kraji displeje, tzn. na začátku každého snímacího cyklu. Za celou dobu vývoje aplikace a testování nebylo zjištěno, proč se u aplikace vyskytuje toto zkreslení. Zkreslení průběhu je vidět na obrázku 14, který představuje sinusový průběh o frekvenci 1000Hz.

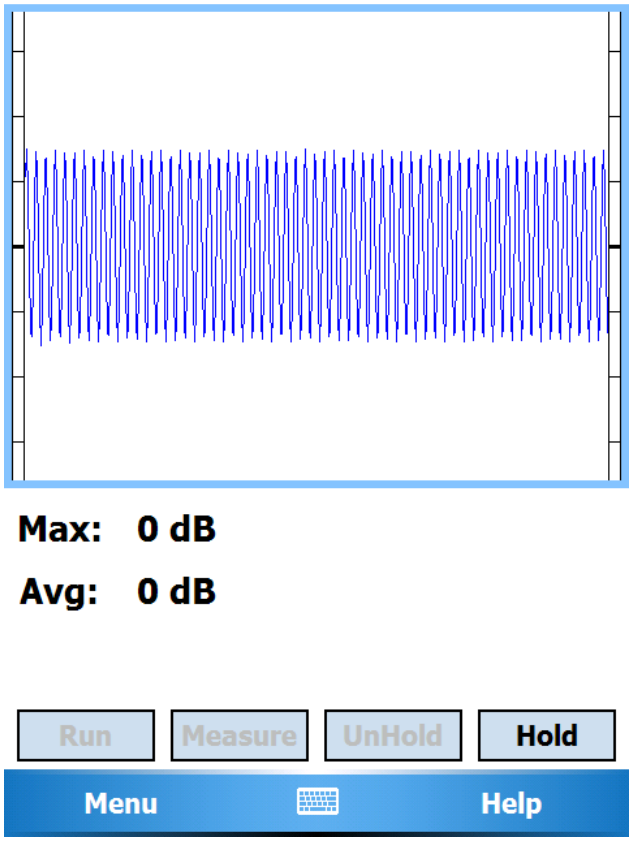
Obrázek 13: Princip druhé fáze testování aplikace PDAScope.



Obrázek 14: Printscreen obrazovky při druhé fázi testování.

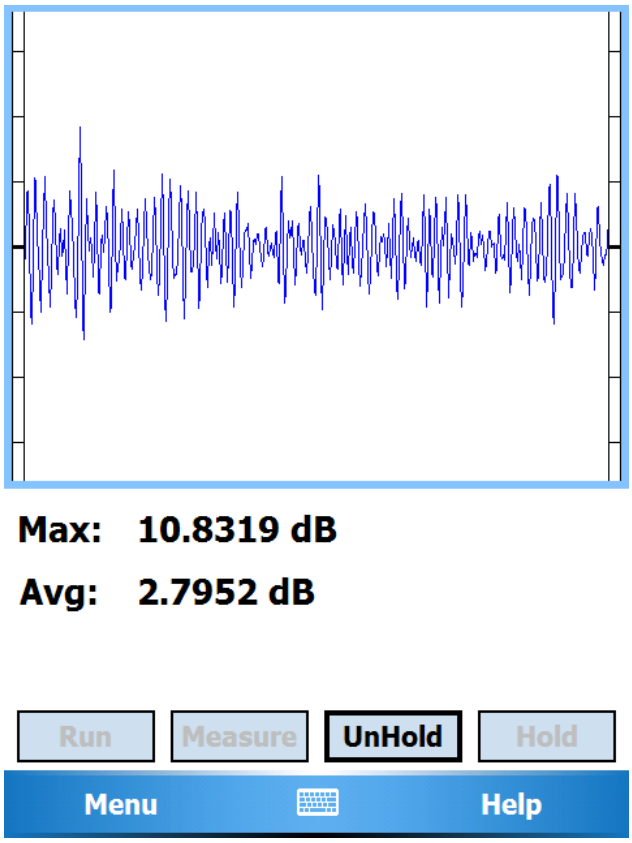
Aby bylo možné zjistit do jaké frekvence sinusového signálu je aplikace schopna věrně vizualizovat, bylo odzkoušeno několik dalších frekvencí. Testování probíhalo postupným navyšováním frekvence vždy o 1000Hz. Celkově byly otestovány sinusové signály o frekvencích 1000Hz, 2000Hz, 3000Hz, 4000Hz a 5000Hz. Ovšem již při frekvenci 4000Hz byl

signál téměř nečitelný. Z testu tedy plyne, že aplikace je schopna vizualizovat signály do frekvence zhruba 3000Hz. Při vizualizaci tak vysoké frekvence už zkreslení popsané v této kapitole není identifikovatelné. Printscreen obrazovky, kdy je vizualizován sinusový signál o frekvenci 3000Hz, ukazuje níže obrázek 15.



Obrázek 15: Vizualizace signálu s frekvencí 3000Hz.

Obrázek 16 na straně 26 zobrazuje poslední test, který spočíval v otestování měření, tedy tlačítka Measure. Při kliknutí na toto tlačítko se vypočtou hodnoty maxima a průměrné hodnoty ze zobrazeného signálu v decibelech.

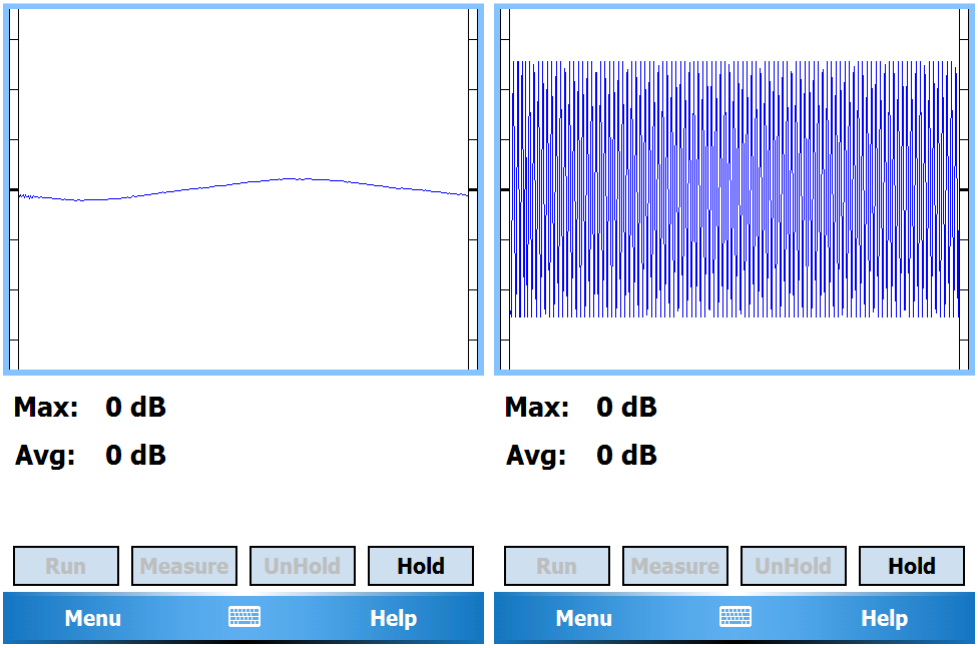


Obrázek 16: Ukázka měření pomocí aplikace PDAScope.

6. Diskuze dosažených výsledků

Aplikace PDAScope tak, jak byla napsána v rámci této práce, je plně funkční. Je schopná věrně vizualizovat externí audio signál do frekvence zhruba 3000Hz. Omezení frekvence je dáno mimo jiné hardwarovými možnostmi zařízení PDA (vestavěným mikrofonem, A/D převodníkem, zvukovou kartou, zobrazovacím zařízením). Aplikace obsahuje jednoduché GUI, které je realizováno v anglickém jazyce. Aplikace obsahuje možnost změny barvy vykreslovaného průběhu a také nápovědu. Ovládání je velice jednoduché a intuitivní.

Obrázek 17 poskytuje srovnání vizualizace velmi nízké a velmi vysoké frekvence externího audio signálu. Printscreen, který je v levé části obrázku 17, zobrazuje aplikaci, při vizualizaci externího audio signálu s frekvencí 50Hz. Printscreen na pravé straně obrázku 17 pak zobrazuje aplikaci při vizualizaci externího audio signálu o frekvenci 5000Hz. Vyšší frekvence než 5000Hz, ani nižší frekvence než 50Hz testovány nebyly. Z obrázku 17 je patrné, že aplikace tak, jak je napsána, již takto nízké, ani takto vysoké frekvence vizualizovat nedokáže.



Obrázek 17: Porovnání vizualizace velmi nízkých a velmi vysokých frekvencí.

Co se týká dalšího vývoje aplikace, nabízí se celá řada dalších možností pro vylepšení. Ať se již jedná o vylepšení grafického prostředí, obsažených funkcí nebo efektivnějšího zdrojového kódu. V oblasti měření fyzikálních veličin lze doprogramovat další funkce pro měření například fekvence signálu, periody signálu a dalších. Další možností je přidaní funkce printscreen, která sejme obraz displeje a uloží jej pro další zpracování. Lze také doprogramovat funkci ukládání naměřených dat a exportu v podobě tabulky nebo třeba zvukového souboru.

Velmi zajímavým námětem pro další vývoj je také možnost využití sluchátkového výstupu PDA zařízení. Využít sluchátkový výstup jako vstup pro přivedení měřeného signálu. Protože sluchátkový výstup je stereo (levý a pravý kanál), je tudíž možné přivést 2 signály na zvukovou kartu PDA zařízení a tyto signály dále vizualizovat nebo je měřit.

Možnosti využití aplikace PDAScope jsou ve vizualizaci a následné analýze zvukových signálů. Základ aplikace lze třeba využít na hlídání intenzity okolního hluku, kdy při překročení stanoveného limitu lze generovat na výstupu patřičné akční zásahy.

7. Závěr

Touto prací jsem realizoval zadání bakalářské práce. Na začátku práce ve druhé kapitole jsem rozebral teoretickou část problematiky. Přiblížil jsem pojmy jako kapesní počítač, osciloskop, programovací jazyk C# nebo .NET Compact Framework.

V praktické části jsem vytvořil plně funkční aplikaci podle zadání této bakalářské práce. Jako první vznikla komponenta pro vykreslování grafů s názvem PDAGraph. Následně pak vznikala celá aplikace. Postupoval jsem úplně od nejjednoduššího. V první fázi jsem se naučil nahrávat audio signál přes vestavěný mikrofon předem stanovené délky. Pak přišla práce s třídami WaveIn.cs a PDAScopeWaveIn.cs, kdy už jsem byl schopen dostat na výstupu pole s odpovídajícími hodnotami. Následovala implementace komponenty pro vykreslování grafů. V poslední fázi vývoje jsem celou aplikaci rozhodil do vláken a vytvořil speciální třídu Record.cs, která se stará o záznam audio signálu.

Vývoj probíhal na PDA zařízení Fujitsu Siemens N560 s operačním systémem Windows Mobile 5.0. Tento systém, stejně jako zařízení PDA jsou už poměrně staré a na dnešní dobu ne až tak výkonné. Pro vývoj mobilních aplikací na platformě .NET Compact Framework bych doporučoval použít systém Windows Mobile 6.5 a PDA zařízení s vyšším hardwarovým výkonem. Ušetří se tím docela dost času. Stejně jako použitím výkonného stolního počítače, kde běží vývojové prostředí.

Během psaní této práce jsem si osvojil objektově orientované programování, které jsem do té doby neovládal na takové úrovni. Myslím si, že společně s touto prací vznikla hezká aplikace, která přináší spoustu možností jak pro další vývoj, tak pro následné reálné využití.

Seznam literatury:

[1] Náhled aplikace VIRTINS Pocket Oscilloscope; http://www.virtins.com/page2.shtml#Pocket Oscilloscope (datum citace 02. 05. 2011)

[2] Personal Digital Assistant; http://cs.wikipedia.org/wiki/Personal_Digital_Assistant (datum citace 02. 05. 2011)

[3] Ukázka PDA zařízení; http://blogs.totalpda.co.uk/2009/06/work-and-play-with-the-fujitsu-siemens-pocket-loox-n560/ (datum citace 02. 05. 2011)

[4] Ukázka Windows Mobile 5.0; http://pocketpccentral.net/wm5brief.htm (datum citace 02. 05. 2011)

[5] LACKO, L. *Programujeme mobilní aplikace ve Visual Studiu .NET*. 1. vyd. Brno: Computer Press, 2004. 479 s. ISBN 80-251-0176-2.

[6] LACKO, L. *Programujeme mobilní aplikace ve Visual Studiu .NET*. 1. vyd. Brno: Computer Press, 2004. 479 s. ISBN 80-251-0176-2.

[7] .NET; http://cs.wikipedia.org/wiki/.NET (datum citace 02. 05. 2011)

[8] KAČMÁŘ, D. *Programujeme .NET aplikace ve Visual Studiu .NET*. 1. vyd. Praha: Computer Press, 2001. 335 s. ISBN 80-251-0176-2.

[9] Vzorkování; http://cs.wikipedia.org/wiki/Vzorkov%C3%A1n%C3%AD (datum citace 02. 05. 2011)

[10] Ukázka vzorkování signálu; http://upload.wikimedia.org/wikipedia/commons/a/a0/Vzorkov%C3%A1n%C3%AD.png (datum citace 02.05.2011)

[11] Mikrofon; http://cs.wikipedia.org/wiki/Mikrofon (datum citace 02. 05. 2011)

[12] Recording and Playing Sound with the Waveform Audio Interface; http://msdn.microsoft.com/en-us/library/aa446573.aspx (datum citace 03. 05. 2011)

[13] Unified Modeling Language; http://cs.wikipedia.org/wiki/Unified_Modeling_Language (datum citace 05. 05. 2011)

Seznam příloh:

1. Extended abstract
2. Zrojové kódy k aplikaci PDAScope