

# Odposlouchávání klávesnice analýzou zvukového signálu

Bc. Josef Hrabal

---

Diplomová práce  
2011



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

akademický rok: 2010/2011

## ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Josef HRABAL**

Osobní číslo: **A08803**

Studijní program: **N 3902 Inženýrská informatika**

Studijní obor: **Bezpečnostní technologie, systémy a management**

Téma práce: **Odposlouchávání klávesnice analýzou zvukového signálu**

Zásady pro vypracování:

1. Provedte literární rešerši zabývající se metodami odposlechu klávesnice. Podrobněji se věnujte metodám založeným na zpracování akustického signálu.
2. Vytvořte program, který bude sloužit pro sběr akustických dat. Program bude zaznamenávat zvukový signál z mikrofону a jemu odpovídající stišťené klávesy.
3. Vytvořte program pro rozpoznávání stišťených kláves na základě akustického signálu, který odpovídá stisku klávesy. Program bude schopen pracovat i s podmnožinami kláves (např. numerická klávesnice, šipky)
4. Navrhněte a provedte experimenty sloužící k určení úspěšnosti vytvořeného programu a k jejímu zlepšování.
5. Zhodnoťte úspěšnost rozpoznávání programu naučeného na rozpoznávání stisku kláves konkrétní klávesnice.

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. SALOMON, David. Foundations of Computer Security . 1st edition. Is.I.J : Springer, 2005. 369 s. ISBN 978-1846281938.
2. MCNAMARA, Joel. Secrets of Computer Espionage : Tactics and Countermeasures . 1st edition. Is.I.J : Wiley, 2003. 408 s. ISBN 978-0764537103 .
3. HOUSER , Pavel. Jak odposlouchávat klávesnici [online]. 2009 , 17.03.09 [cit. 2010-01-25]. Dostupný z WWW: <http://scienceworld.cz/aktuality/Jak-odposlouchavat-klavesnici-4768>.
4. ASONOV, D.; AGRAWAL, R. Keyboard acoustic emanations. In Proceedings of 2004 IEEE Symposium on Security and Privacy. Is.I.J : Is.n.J, 2004. s. 3-11. Dostupné z WWW: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1301311>.
5. KIL, David H.; SHIN, Frances B. . Pattern Recognition and Prediction with Applications to Signal Processing. Is.I.J : American Inst. of Physics, 1998. 440 s. ISBN 978-1563964770.
6. TUČKOVÁ, Jana. Vybrané aplikace umělých neuronových sítí při zpracování signálů. Vyd. 1. Praha : České vysoké učení technické v Praze, 2009. 224 s. ISBN 978-80-01-04229-8
7. ZELINKA, Ivan. Umělá inteligence aneb úvod do neuronových sítí, evolučních algoritmů. 1. vyd. Zlín : Univerzita Tomáše Bati, Fakulta technologická, 2005. 127 s. ISBN 8073182777

Vedoucí diplomové práce:

**Ing. Petr Chalupa, Ph.D.**

Ústav řízení procesů

Datum zadání diplomové práce:

**25. února 2011**

Termín odevzdání diplomové práce:

**27. května 2011**

Ve Zlíně dne 25. února 2011

prof. Ing. Vladimír Vašek, CSc.  
*děkan*



doc. RNDr. Vojtěch Křesálek, CSc.  
*veditel ústavu*

## **ABSTRAKT**

Předmětem diplomové práce je provést seznámení s metodami odposlechu klávesnice s důrazem na odposlouchávání klávesnice analýzou zvukového signálu a následnou realizaci počítačového programu k odposlechu konkrétní počítačové klávesnice. Práce v úvodu pojednává o počítačové klávesnici obecně, následně rozebírá možnosti odposlechu, speciálně metodou analýzy zvukového signálu. Realizuje software pro sběr dat z konkrétní klávesnice, jejich následnou analýzu. Závěrem hodnotí úspěšnost odposlouchávání klávesnice metodou analýzy zvukového signálu.

Klíčová slova:

počítačová klávesnice, keylogging, odposlech klávesnice, softwarový odposlech, hardwarový odposlech, analýza zvukového signálu, zabezpečení klávesnice

## **ABSTRACT**

Main topic of this dissertation is to be familiarized with keyboard monitoring methods with emphasis to monitor keyboard by analysis of sound signal. Computer system for monitoring certain computer keyboard is to be implemented. In the beginning there is general analysis of computer keyboards and analysis of possibilities to monitor it, with putting accent on monitoring by sound signal analysis. Implements software for data collection from certain keyboard and analysis of collected data. Success rate of keyboard monitoring by sound signal analysis method is evaluated at the end of dissertation.

Keywords: computer keyboard, keylogging, eavesdropping, software keylogging, hardware keylogging, sound analyzing, keyboard security

## Poděkování

Poděkování patří vedoucímu diplomové práce panu Ing. Petru Chalupovi Ph.D. za odborné vedení, odborné rady a připomínky. Připojuji také poděkování rodině, kolegům v práci a kamarádům za podporu při psaní diplomové práce.

### **Prohlašuji, že:**

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. O vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách) ve znění pozdějších předpisů, bez ohledu na výsledek obhajoby (1);
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk práce bude uložen v archivu Institutu bezpečnostních technologií Fakulty informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen s tím, že na mou diplomovou/bakalářskou práci se plně vztahuje zákon č.121/2000 Sb. O právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů ( autorský zákon) ve znění pozdějších právních předpisů, zejm. §35, odst.3(2);
- beru na vědomí, že podle §60, odst.1 (3) autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu §12, odst.4 autorského zákona;
- beru na vědomí, že podle §60, odst. 2 a 3 mohu užít své dílo – diplomovou práci- nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití),nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

### **Prohlašuji,**

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....  
Podpis diplomanta

- 
- 1) zákon č. 111/1998 Sb. O vysokých školách a o změně a doplnění dalších zákonů ( zákon o vysokých školách) ve znění pozdějších předpisů, § 47 Zveřejňování závěrečných prací
  - 2) Dizertační, diplomové, bakalářské a rigorózní práce odevzdané uchazečem k obhajobě musí být nejméně pět pracovních dnů před konáním obhajoby zveřejněny k nahlížení veřejnosti v místě určeném vnitřním předpisem vysoké školy nebo není-li tak určeno, v místě pracoviště vysoké školy, kde se bude konat obhajoba práce. Každý si může ze zveřejněné práce na vlastní náklady pořizovat výpisy, opisy nebo rozmnoženiny;
  - 3) Platí, že odevzdáním práce autor souhlasí se zveřejněním své práce podle tohoto zákona bez ohledu na výsledek obhajoby.

# OBSAH

<b>ABSTRAKT</b> .....	<b>4</b>
<b>ABSTRACT</b> .....	<b>4</b>
<b>ÚVOD</b> .....	<b>9</b>
<b>1 POČÍTAČOVÁ KLÁVESNICE</b> .....	<b>10</b>
1.1 HISTORIE POČÍTAČOVÉ KLÁVESNICE.....	10
1.2 KONSTRUKCE KLÁVESNICE.....	12
1.2.1 Klávesy.....	12
1.2.2 Druhy kláves.....	14
1.2.3 Typy klávesnic.....	16
1.2.4 Rozložení kláves.....	19
1.2.5 Klávesové spínače.....	21
1.2.6 Další součásti klávesnice.....	25
1.2.7 Druhy připojení a konektorů.....	26
1.3 POPIS FUNKCE KLÁVESNICE.....	29
<b>2 METODY ODPOSLOUCHÁVÁNÍ KLÁVESNICE</b> .....	<b>30</b>
2.1 PRÁVNÍ ASPEKTY ODPOSLOUCHÁVÁNÍ KLÁVESNICE.....	30
2.1.1 Zásah do Ústavních i zákonných práv.....	30
2.1.2 Trestněprávní odpovědnost.....	31
2.1.3 Zákoník práce a kontrola zaměstnanců.....	32
2.2 SOFTWAREOVÝ KEYLOGGING.....	33
2.2.1 Na základě hypervizoru.....	34
2.2.2 Na základě kernelu.....	35
2.2.3 Na základě API.....	36
2.2.4 Sledování dat z formulářů.....	38
2.2.5 Analyzátoři paketů.....	38
2.3 HARDWAROVÝ KEYLOGGING.....	38
2.3.1 Na základě modifikace firmware.....	38
2.3.2 Vložené zařízení.....	39
2.3.3 Bezdrátové odposlouchávání.....	39
2.3.4 Využití elektromagnetického záření.....	39
2.3.5 Překrytí klávesnice.....	44
2.4 KEYLOGGING POMOCÍ ANALÝZY ZVUKOVÉHO SIGNÁLU.....	45
2.4.1 Fourierova transformace.....	45
2.4.2 Neuronové sítě.....	48
2.4.3 Cepstrum.....	51
2.4.4 Technologie rozpoznávání řeči.....	51
2.4.5 Rozbor prací na téma keyloggingu analýzou zvukového signálu.....	54
2.5 DALŠÍ METODY KEYLOGGINGU.....	63
2.6 OCHRANA PROTI ODPOSLECHU.....	64
2.6.1 Ochrana proti softwarovým keyloggerům.....	64

2.6.2	Ochrana proti hardwarovým keyloggerům.....	64
2.6.3	Ochrana proti ostatním keyloggerům.....	64
<b>3</b>	<b>PROGRAM PRO SBĚR A ANALÝZU DAT.....</b>	<b>66</b>
3.1	ZÁKLADNÍ SCHÉMA PROGRAMU.....	66
3.2	PROSTŘEDÍ .NET RUNTIME.....	66
3.3	POSTUP VYTVÁŘENÍ PROGRAMU.....	69
3.3.1	Soubory WAV.....	69
3.3.2	Knihovna DXDSAR.....	71
3.3.3	Program pro sběr dat.....	74
3.3.4	Program pro analýzu dat.....	77
<b>4</b>	<b>EXPERIMENTÁLNÍ OVĚŘENÍ PROGRAMU.....</b>	<b>80</b>
4.1	VÝBĚR EXPERIMENTŮ K OVĚŘENÍ FUNKČNOSTI PROGRAMU.....	80
4.2	REALIZACE EXPERIMENTŮ.....	80
4.2.1	Ověření funkčnosti na naučené klávesnici.....	80
4.2.2	Ověření funkčnosti na nenaučené klávesnici.....	81
4.2.3	Ověření funkčnosti při psaní různými osobami.....	83
4.2.4	Ověření funkčnosti různými mikrofony.....	84
4.2.5	Vliv změny polohy mikrofonu.....	85
4.2.6	Vliv okolního hluku.....	88
4.3	NÁVRHY NA ZLEPŠENÍ PROGRAMU.....	90
4.3.1	Zlepšení rozpoznávání.....	90
4.3.2	Rozšíření možností.....	91
	<b>ZÁVĚR.....</b>	<b>92</b>
	<b>SEZNAM POUŽITÉ LITERATURY.....</b>	<b>93</b>
	<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....</b>	<b>98</b>
	<b>SEZNAM OBRÁZKŮ.....</b>	<b>99</b>
	<b>SEZNAM TABULEK.....</b>	<b>102</b>
	<b>SEZNAM PŘÍLOH.....</b>	<b>102</b>

## ÚVOD

Odposlouchávání klávesnice je činnost, která vede ke zjištění stisknutých kláves na klávesnici bez vědomí osoby, která klávesy stiskne. Může se jednat jak o počítačové klávesnice, tak i o klávesnice mobilních telefonů, klávesnice bankomatů, případně i jiných zařízení.

Odposloucháváním klávesnice je možné zjistit přístupová hesla nebo jiné údaje, případně i text odesílaných zpráv zadávaných na klávesnici a jejich následné využití případně zneužití. Z právního hlediska se jedná většinou o nezákonnou činnost, nicméně existuje i zákonné využití odposlouchávání klávesnice.

Analýzou zvukového signálu je možné odposlouchávat i zařízení, které využívají šifrovaného přenosu dat, které nicméně není u počítačových klávesnic využíváno. Nevýhodou je nutnost naučení odposlouchávacího programu nebo zařízení na příslušnou klávesnici, útočník k ní tedy potřebuje mít fyzický přístup před plánovaným útokem.

Nebezpečnost odposlouchávání klávesnice spočívá i v tom, že se jedná o jednoduchou záškodnickou činnost s nebezpečnými následky. Mállokterý uživatel počítačové, či jiné klávesnice tuší o možnosti, že by klávesnice byla odposlouchávaná a tudíž se této možnosti nijak nebrání.

## 1 POČÍTAČOVÁ KLÁVESNICE

Počítačová klávesnice je jedním ze základních vstupních rozhraní osobního počítače. Bývá doplněno myší a v současné době u některých počítačů nahrazováno dotykovým displejem. Standardní počítačové klávesnice jsou napájeny z počítače a komunikují s ním po sériové lince. Počítačová klávesnice má na vrchní straně tlačítka, zvané klávesy. Ve většině případů stisk klávesy způsobí odeslání jednoho znaku. Některé klávesy slouží pouze jako předvolba. Jiné klávesy fungují jako takzvané mrtvé klávesy, kdy se jich využívá pro psaní znaků, které nejsou na klávesnici obsaženy.

### 1.1 Historie počítačové klávesnice

Počítačová klávesnice je odvozena od klávesnice psacího stroje a dálnopisu.



*Obr. 1: Stolní model dálnopisu T100 firmy Siemens*

Psací stroj slouží k přenosu textu psaného uživatelem na klávesnici na papír. Psací stroj může být mechanický nebo elektrický. První funkční psací stroj byl sestaven a patentován Henry Millem v roce 1714. Rozložení kláves bylo odlišné od současného rozložení a konkrétní řešení se nedochovalo. Dalším významným milníkem v historii psacího stroje byl rok 1874, kdy Christopher Latham Sholes a Carlos Glidden prodali krachující zbrojovce E. Remington & Sons psací stroj s názvem „Typewriter“. První verze uměla pouze verzálky a nebyla úspěšná. Druhá verze už obsahovala přeřazovač (Shift) umožňující psát malá i velká písmena a prodávala se daleko lépe.



*Obr. 2: Jeden z prvních psacích strojů*

Už na začátku vývoje ale Sholes narazil na problém při psaní písmen, která jsou blízko u sebe. Typové páky se zasekávaly a psaní se stávalo náročným. Proto přišel se zajímavou myšlenkou: typové páky s nejčastěji používanými písmeny anglické abecedy umístil dále od sebe a zároveň zvolil takové rozložení, které by pisateli umožnilo rychlý způsob psaní všemy desíti prsty. Vítězství v soutěži v rychlosti a přesnosti psaní dalo přednost tomuto rozložení kláves před všemi ostatními. V roce 1888 bylo toto rozložení kláves přijato v Torontu v Americe na sjezdu odborníků jako standard. Následně se tohle rozložení rozšířilo do celého světa. Jednalo se o rozložení typu „QWERTY“. Z něj byly dále odvozeny další rozložení klávesnice.



*Obr. 3: Český psací stroj Consul*

První elektronická počítačová klávesnice se objevila na počátku 70. let. Klávesové spínače byly jednotlivé a umístěné do děr v kovovém rámu. Stály kolem 80 – 120 dolarů a byly používány jako vstupní zařízení pro mainframe datové terminály. Nejpopulárnější byly spínače s jazýčkovými kontakty (spínač s jazýčkovými kontakty se skládá ze dvou nebo více kovových kontaktků umístěných v hermeticky uzavřeném skleněném obalu). Spínače měly vydržet 100 miliónů úhozů a klávesy měly zdvih 4.75 mm (pro porovnání v současnosti 2.79 mm). V polovině 70. let se objevily levnější spínače (přímo uzavírající elektrický obvod), které byly podstatně levnější nicméně na úkor životnosti. Jedny z prvních byly kapacitní spínače vyvinuté firmou Key Tronic Corporation v roce 1978. Za první modernější klávesnici se dá považovat klávesnice standardu XT od firmy IBM navržené a do prodeje uvedené začátkem 80. let. Jelikož nebyla klávesnice příliš oblíbená přišla firma IBM s modelem klávesnice standardu AT, která se dále vyvíjela až do současné podoby počítačové klávesnice.

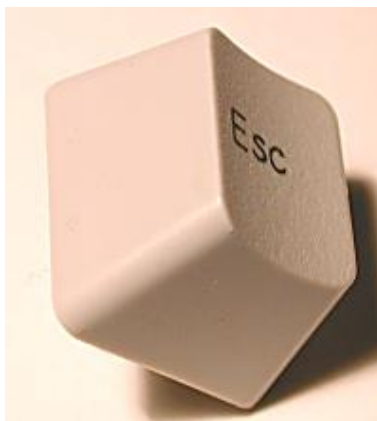


*Obr. 4: Jedna z prvních počítačových klávesnic*

## 1.2 Konstrukce klávesnice

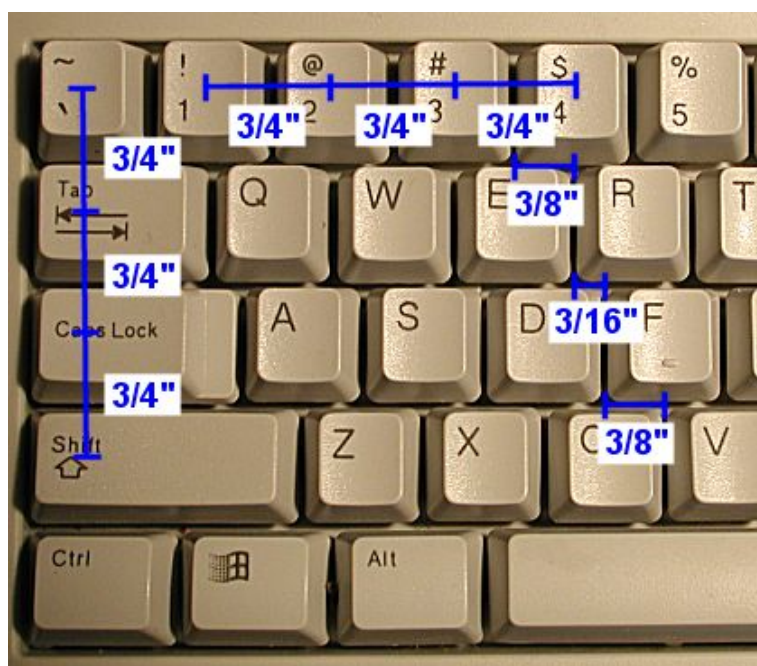
### 1.2.1 Klávesy

Klávesy jsou tou částí klávesnice, která přichází do styku s uživatelovými prsty při psaní na počítačové klávesnici. U nejstarších klávesnic nebyly klávesy odděleny od klávesových spínačů a byly spojeny v jednu část. Oddělitelné klávesy od snímačů mají nicméně výhody v tom, že je možné poškozené nebo opotřebované klávesy vyměnit za nové bez výměny celé klávesnice. Taktéž je možné změnit rozložení kláves vyjmutím a vložením kláves na jiné místo. Stejně tak se i jednodušeji celá klávesnice čistí.



Obr. 5: Klávesa z typické počítačové klávesnice

Téměř všechny počítačové klávesnice používají klávesy o stejné výšce a šířce. Jedná se o čtverec o hraně přibližně 1,25 cm (0,5 palce), který se směrem dolů rozšiřuje na 1,75 cm (11/16 palce). Rozteč kláves je taky víceméně standardní. Jedná se o 1,9 cm (0,75 palce) pro klávesy v jednom řádku. Rozteč jednotlivých řádků je zobrazená na obrázku.



Obr. 6: Mezery mezi klávesami

Tato standardní rozteč je používána u většiny klávesnic z důvodů jednoduššího přechodu uživatelů z jedné klávesnice na druhou. Není dodržována pouze u velmi malých notebooků a podobných zařízení, kde není možné tuto standardní rozteč použít. Jedním z důležitých parametrů kláves je jejich zdvih. Zdvihem se myslí rozdíl mezi stisknutou a nestisknutou klávesou. Zdvih kláves bývá menší typicky u notebooků, kde je potřeba šetřit místem.

### 1.2.2 Druhy kláves

Už od začátku vývoje počítačové klávesnice byly klávesy se shodným nebo podobným významem sdružovány do bloků. Tyto klávesy můžeme rozdělit do několika bloků jak je zobrazeno na obrázku.



Obr. 7: Druhy kláves

Alfanumerické a znaménkové klávesy jsou modré, modifikační klávesy zelené, kurzorové a navigační klávesy žluté, numerická klávesnice červená, funkční klávesy fialové, speciální klávesy azurové, klávesy Windows růžové a další klávesy šedivé.

#### *Alfanumerické a znaménkové klávesy*

Jsou největším a nejpoužívanějším blokem kláves na počítačové klávesnici při zadávání textů. Blok obsahuje písmena anglické abecedy „A“ až „Z“. Normální stisk dává malá písmena, stisk s klávesou „Shift“ dává velká písmena, modifikátor „Caps Lock“ obrací logiku klávesy „Shift“. Dále jsou obsaženy klávesy, které dle rozložení klávesnice píšou číslice, případně speciální znaky, případně další písmena, která nejsou obsažena ve anglické abecedě. Tyto klávesy jsou umístěny nad klávesami s písmeny anglické abecedy. Po stranách bloku alfanumerických a znaménkových kláves jsou umístěny klávesy pro oddělovače. Jedná se o klávesu „Enter“, která nám způsobí nový řádek, klávesu „Tab“, která udělá odskok tabulátoru a klávesu „Space“, která udělá mezeru.

#### *Modifikační klávesy*

Tyto klávesy nemají žádný význam, pokud jsou stisknuté samostatně a modifikují pouze význam kláves, se kterými jsou stisknuty. Jsou umístěny kolem alfanumerických kláves. Jedná se o klávesu „Shift“, jejíž význam byl již zmíněn a kdy píše velká písmena, případně znaky uvedené v horní části popisku na klávesách. Stejně tak může modifikovat význam

funkčních kláves (kdy F6 je jiná klávesová zkratka než Shift + F6). Dále jsou to klávesy „Ctrl“, někdy také „Control“, a „Alt“, které pouze modifikuje význam ostatních kláves. Klávesu „Alt“ lze využít k napsání libovolného znaku pomocí stisku klávesy a vypsání ASCII kódu znaku. Další jsou modifikátory, které se jedním stisknutím zapnou a druhým stisknutím vypnou. Jedná se o „Caps Lock“, který je jakoby permanentní klávesou „Shift“, nicméně pouze u písmen (ať s diakritikou nebo bez), klávesu „Num Lock“, která zapíná numerický blok na klávesnici (pokud je obsažen) do režimu psaní číslic. Jinak klávesy suplují kurzorové klávesy a klávesy pro pohyb v textu. Tento režim numerické klávesnic byl využíván v minulosti, kdy klávesnice neobsahovaly kurzorové klávesy a klávesy pro pohyb v textu. Posledním modifikátorem je klávesa „Scroll Lock“, která mění význam kurzorových kláves a kláves pro pohyb v textu v některých programech.

### ***Kurzorové klávesy a navigační klávesy***

Kurzorové klávesy umožňují pohyb v textu ve čtyřech základních směrech (vlevo, nahoru, dolů, doprava). Některé klávesnice obsahovaly osm kurzorových kláves s mezisměry. Nejčastěji bývají rozloženy do tvaru obráceného písmena „T“. Někdy bývá použité rozložení do diamantu nebo rozložení vedle sebe (většinou u notebooků), které ale ztěžuje použití uživateli zvyklému na klasické rozložení. Klávesa „Insert“, někdy je „Ins“ funguje jako přepínač mezi dvěma režimy psaní. V jednom se znak vkládá na aktuální pozici kurzoru, v opačném se znak přepisuje. Klávesa „Delete“ maže znak z pozice napravo od aktuální pozice, klávesa „Backspace“ vlevo. Klávesy „Page Up“ a „Page Down“, někdy jen „Pg Up“ a „Pg Dn“ jsou používány pro přechod o stránku dopředu nebo dozadu. Klávesy „Home“ a „End“ pro skok na začátek, případně konec řádku.

### ***Numerická klávesnice***

Veškeré klávesy obsažené na numerickém bloku klávesnice najdeme i jinde na klávesnici. Jak klávesy pro psaní číslic, tak klávesy pro základní matematické operace („/“, „\*“, „-“, „+“), tak i ještě jednu klávesu „Enter“. Numerický blok nebývá osazený na klávesnicích, kde je potřeba šetřit místem a numerický blok by bylo obtížné osadit, protože by se tam nevešel. Jedná se hlavně o klávesnice malých notebooků.

### ***Funkční klávesy***

Funkční klávesy nejsou používány pro psaní znaků, ale pouze k přístupu ke speciálním funkcím počítačových programů. U dřívějších klávesnic se jednalo o deset kláves v maticovém rozložení. Nyní se používá dvanáct kláves umístěných v horní části klávesnice v jedné řadě.

### ***Speciální klávesy***

Jedná se o další speciální klávesy, které nelze zařadit do žádné z předcházejících kategorií. První je klávesa „Esc“, někdy „Escape“, která byla používána hlavně ve starších programech ke zrušení akce, případně k vyskočení z programu nebo z některého modulu programu. Klávesa „Print Screen / Sys Request“ způsobí odeslání aktuální obrazovky na tiskárnu nebo do schránky v závislosti na operačním systému. Druhý význam klávesy byl v minulosti pro odeslání systémového požadavku. Tato funkce není ale již v současné době ani používána ani podporována. Klávesa „Pause / Break“ v některých programech pozastaví provádění, případně pozastaví výpis na obrazovku.

### ***Klávesy Windows***

Byly přidány firmou Microsoft pro urychlení přístupu k funkcím operačního systému Microsoft Windows. Klávesa „Win“ umožňuje přístup k nabídce „Start“ a je součástí dalších klávesových zkratk, které zrychlují běžné operace. Klávesa „Menu“ umožňuje přístup ke kontextové nabídce a supluje stisk pravého tlačítka na počítačové myši.

### ***Další klávesy***

Bývají osazeny na takzvaných multimediálních klávesnicích. Jedná se o další klávesy, které neslouží pro přímé zadávání znaků, ale pro spouštění definovaných akcí. Tou bývá ovládání hlasitosti, ovládání přehrávače zvuků nebo videí, případně spuštění nejčastěji internetového prohlížeče, emailového klienta nebo jiného programu.

## **1.2.3 Typy klávesnic**

### ***Klávesnice PC XT 83-key***

Je první počítačovou klávesnicí od firmy IBM z počátku 80. let. Obsahovala 83 kláves, nicméně ne moc dobře rozložených. To byl taky jeden z důvodů, proč nebyla příliš

oblíbená. Klávesy byly špatně rozložené a měly špatnou velikost (speciálně klávesy „Ctrl“, „Shift“, „Caps Lock“ a „Enter“). Klávesnice taktéž neobsahovala samostatné kurzorové klávesy ani žádné LED indikátory. Funkční klávesy byly v bloku nalevo od alfanumerické části, což nebylo také příliš šťastné řešení.



*Obr. 8: Klávesnice PC XT 83-key*

### ***Klávesnice PC AT 84-key***

Na základě stížností od uživatelů vydala firma IBM novou klávesnici s jiným rozložením kláves. Obsahovala 84 kláves podstatně lépe rozložených. Přibyla taktéž klávesa „Sys Request“ a stavové LED indikátory. Co zůstalo je poloha funkčních kláves, nevhodná velikost a umístění kláves „Ctrl“ a „Caps Lock“ a přemístila se klávesa „Escape“. Klávesnice AT taky používá jinou a obousměrnou (klávesnici XT pouze jednosměrnou) signalizaci a není kompatibilní s klávesnicí XT.



*Obr. 9: Klávesnice PC AT 84-key*

### ***Klávesnice „Enhanced“ 101-key***

V roce 1986 k modelu počítače IBM PC/AT Model 339 vydala firma IBM novou klávesnici. Používala stejné rozhraní jako klávesnice AT, měla pouze jiné rozložení kláves. Klávesy „Ctrl“, „Caps Lock“ a „Escape“ se přesunuly do pozice, kde jsme na ně zvyklí nyní. Funkční klávesy se přesunuly nad alfanumerické klávesy a přibýly klávesy „F11“ a „F12“. Největším rozdílem je ale přidání kurzorových kláves s kláves pro pohyb v textu.



Obr. 10: Klávesnice „Enhanced“ 101-key

### ***Klávesnice „Enhanced“ 102-key***

Jediným rozdílem od klávesnice se 101 klávesami je v další klávese a ve tvaru klávesy „Enter“, který připomíná písmeno „L“.

### ***Klávesnice „Windows“ 104-key***

S rozmachem operačního systému Windows (konkrétně verze Windows 95 uvedené v roce 1995) si firma Microsoft prosadila na klávesnici tři další klávesy. Dvě shodné otevírající nabídku „Start“ operačního systému Windows a jednu otevírající kontextovou nabídku. Jinak se klávesnice neliší v rozložení od klávesnic se 101 nebo 102 klávesami.



Obr. 11: Klávesnice "Windows" 104-key

### ***Programovatelné klávesnice***

Obsahují kromě standardních kláves ještě další programovatelné klávesy, které lze využít pro realizaci různých makr pro provedení složitějších akcí na stisk jedné klávesy. Programovatelné klávesnice se nicméně příliš nerozšířily, protože většina uživatelů nepotřebuje měnit význam a polohu standardních kláves a ani nevyužije makra.

### ***Ergonomické klávesnice***

Tyto klávesnice byly navrženy pro zlepšení komfortu při psaní. Klávesnice je rozdělená na dvě poloviny, které mezi sebou svírají úhel mezi 30 a 60 stupni. Bývají často doplněny opěrkami pro zápěstí a případně i dalšími ergonomickými prvky. Někteří uživatelé si je

oblíbili, nicméně většího úspěchu nezaznamenali.



Obr. 12: Ergonomická klávesnice Microsoft

### *Notebookové klávesnice*

Klávesnice u notebooků bývají založené na rozložení klávesnice typu „Windows“. Pouze z důvodu často omezeného prostoru bývá rozložení kláves stísněnější a někdy i chybí numerický blok kláves. Naopak bývají osazeny další funkční klávesou „Fn“, která umožňuje přístup k dalším funkcím.



Obr. 13: Jedna z dřívějších notebookových klávesnic

### **1.2.4 Rozložení kláves**

Rozložení kláves počítačové klávesnice má vliv na její použitelnost. Jedná se o rozmístění kláves, jejich velikost, rozteč a další parametry. Speciálně u alfanumerické části je rozložení kláves důležité, jelikož je klávesnice nejčastěji používána pro vkládání delších textů. Nejčastější rozložení „QWERTY“ bývá modifikováno v některých zemích na „QWERTZ“. Název rozložení vychází z názvu kláves v levé horní části. V 30. letech minulého století pánové August Dvorak and William Dealey studovali možnosti jak zvýšit ergonomii psaní na klávesnici a možnost jak zvýšit rychlost psaní. Začali pracovat na novém rozložení klávesnice. Nejčastěji používané klávesy umístili do základního řádku a

méně častěji používané klávesy dále. Narozdíl od „QWERTY“ rozložení je až 70 % úhozů realizováno v základním řádku.



Obr. 14: Rozložení kláves "Dvorak"

Modré popisky kláves udávají rozdíl proti standardnímu rozložení „QWERTY“. Při psaní anglické abecedy lze prý dosáhnout urychlení až 190 %. Nevýhodou rozložení „Dvorak“ je to, že uživatelé se musí naučit nové rozložení a taktéž, že musí pracovat i s rozložením „QWERTY“. Proto rozložení „Dvorak“ „QWERTY“ nenahradilo. Nicméně v moderních operačních systémech je možné změnit rozložení klávesnice softwarově.

### ***Mezinárodní rozložení klávesnice***

Země, které používají latinku ve většině případů využívají „QWERTY“ rozložení pouze se změněnými diakritickými znaky podle příslušné abecedy. Nicméně i tak existuje francouzské „AZERTY“. Název je opět odvozený od prvních šesti písmen vlevo nahoře. Země, které nepoužívají latinku využívají naprosto rozdílné rozložení znaků klávesnice.

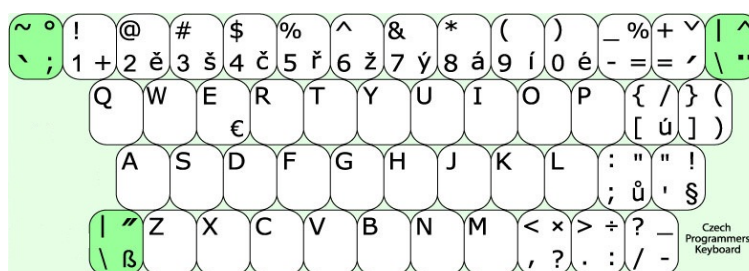


Obr. 15: Ruská klávesnice

### ***Rozložení klávesnice používané v České republice***

Rozložení kláves na české klávesnici vychází z mezinárodní normy ISO/IEC 9995:1997, která uvádí, že rozmístění se provádí podle národních norem nebo zvyklostí. Jako jeden z příkladů uvádí možnost na klávese B01 (na českém standardu znak Y) umístit znaky Z nebo Y nebo W. Je stanoveno ČSN 36 9050:1994 Rozložení znaků na české klávesnici

pro textové a kancelářské systémy, která stanovuje rozložení pro 48 kláves. Upraveno je tak umístění znaků abecedy, číslic a dále +, =, ', ", \$, ), čárky, -, ., ;, ~, %, !, ", (, \_ , : , ? a některých dalších. Umístění znaků vychází z rozložení „QWERTZ“. Rozložení kláves „QWERTZ“ vniklo prohozením umístění znaků Z a Y v německy mluvících zemích již v 19. století. Důvodem k tomu nejspíše byl četnější výskyt Z v německém textu. Vzhledem k tomu, že České země v té době byly součástí Rakouska-Uherska a němčina byla úředním jazykem, ujala se v Českých zemích klávesnice „QWERTZ“. Rozložení kláves „QWERTZ“ je pro psaní českého textu výhodnější než „QWERTY“, neboť i v českém textu se častěji vyskytuje Z než Y. Mnoho českých programátorů ale dává přednost anglickému standardu, který vychází z rozložení „QWERTY“, protože potřebují znaky, které na českém standardu nejsou, případně používají tzv. českou programátorskou klávesnici, nebo českou „QWERTY“ klávesnici, lišící se jen prohozeným Z a Y, protože si již na anglickou klávesnici zvykli.



Obr. 16: České programátorské rozložení

### 1.2.5 Klávesové spínače

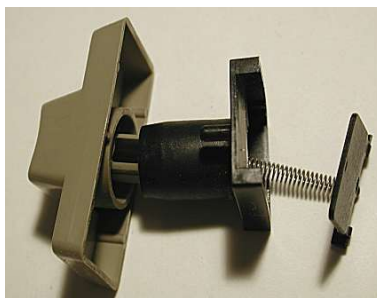
Klávesové spínače se starají o detekci stisknutí klávesy. Klávesové spínače určují vlastnosti klávesnice. Mezi základní vlastnosti klávesových spínačů patří:

- **zdvih** určuje jakou dráhu urazí klávesa než se zaznamená stisk, některé technologie umožňují dosažení velkého i malého zdvihu, jiné možnost volby zdvihu omezují
- **odezva na stisk** určuje odezvu na stisk klávesy, takže uživatel má dobrou nebo špatnou zpětnou vazbu, že klávesu stisknul
- **slyšitelnost** určuje jak je stisk klávesy slyšet a je v korelaci s odezvou na stisk
- **citlivost** určuje sílu, která je potřebná ke stisknutí klávesy, některé spínače potřebují silnější stisk, jiným stačí slabší
- **pocit** je subjektivní a určuje jak klávesnici vnímá uživatel jako celek

- **odolnost** udává výdrž spínačů a určuje i životnost klávesnice jako celku
- **cena** udává jak drahé jsou spínače. Obecně jsou spínače s větší životností, s lepší odezvou na stisk dražší

### *Čistě mechanické spínače*

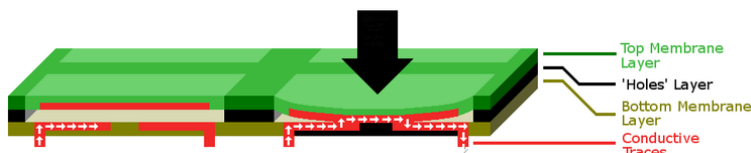
V principu se jedná o nejjednodušší variantu spínačů, kdy se jedná o klasický spínač, který způsobí chvilkové sepnutí dvou kovových kontaktů. K návratu stisknuté klávesy se používá mechanismus složený z pružinky a svorky. Tento mechanismus vydává i zvuk charakteristický pro tuto klávesnici. Další jeho funkcí je i tvorba odporu proti stisku. Výhodou je vysoká trvanlivost. Nevýhodou je odezva na stisk a hlavně cena. V moderních počítačových klávesnicích nebývají už používány.



Obr. 17: Mechanický spínač

### *Membránové spínače*

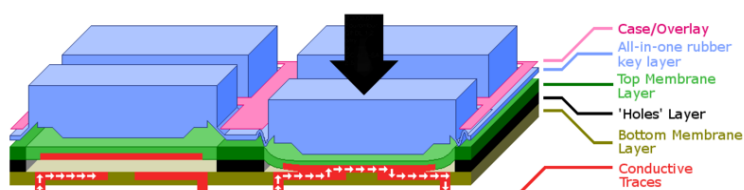
Membránové spínače se skládají z gumové membrány s vodivými prvky, které spojují kontakty. Jejich velkou nevýhodou je malý zdvih a téměř žádná odezva na stisk. Odezva na stisk byla často realizována systémovým reproduktorem. Jejich velkou výhodou je odolnost proti nečistotám a proto se často objevují v průmyslových aplikacích. V současnosti se v počítačových klávesnicích již nepoužívají. Jejich použití je většinou u přístrojů spotřební elektroniky, například mikrovlnných trub nebo kopírek.



Obr. 18: Schéma membránového spínače

### *Mechanické spínače s pěnovými prvky*

Jejich charakteristickým prvkem je pružný pěnový prvek připevněný ke kolíku klávesy. Na spodní části pěnového prvku je vodivá fólie, která spojuje kontakty. Tento typ spínače se již u počítačových klávesnic nepoužívá, jelikož bývá nahrazován ne příliš dražší, ale výhodnější variantou spínače s gumovou membránou. Nevýhodou je špatná odezva na stisk, častá koroze jak na desce s tištěnými spoji, tak i na vodivé fólii. Výhodou je snadné čištění.



Obr. 19: Schéma spínače s pěnovými prvky

### *Mechanické spínače s gumovou membránou*

Mechanické spínače s gumovou membránou vycházejí z membránových spínačů a spínačů s pěnovými prvky. Mechanický spínač je nahrazený gumovou čepičkou s uhlíkovým kontaktem místo návratového mechanismu. Při stisku klávesy uhlíkový kontakt spojí kontakty na desce a po uvolnění se zase vrátí do původního stavu. Mezi výhody patří jednoduchost, není potřeba návratová pružinka, uhlíková část nepodléhá korozi, klávesnice je odolná proti nečistotám a spínače mají velmi solidní výdrž. V současné době se jedná o nejpoužívanější technologii pro počítačové klávesnice.



Obr. 20: Schéma klávesnice se mechanickými spínači s gumovou membránou

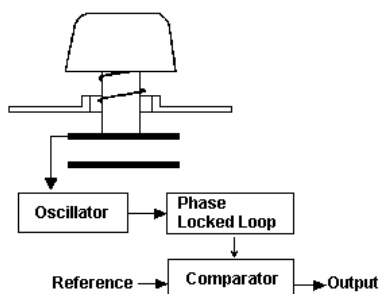
Speciálním typem klávesnic se spínači s gumovou membránou jsou spínače s „nůžkovými“ spínači. Používají se hlavně u klávesnic notebooků, jelikož mají menší zdvih. Taktéž jsou i těžší, nicméně i dražší a obtížněji se i čistí.



Obr. 21: "Nůžkový" snímač

### ***Kapacitní spínače***

Kapacitní spínače využívají změny kapacity při přiblížení pevné elektrody na podložce a pohyblivé na pružné membráně. Jelikož nedochází k mechanickému kontaktu, spínače mají velkou životnost. Taktéž ale samotný kapacitní spínač nemá žádnou odezvu stisku, proto bývá doplňován mechanismem, který tuto odezvu zajistí. Nevýhodou je vysoká cena a proto nejsou příliš používány.



Obr. 22: Schéma klávesnice s kapacitním snímačem

### ***Spínače „Buckling spring“***

Spínač „Buckling spring“ je spínač vyvinutý firmou IBM, kdy malé kladívko spojí elektrický kontakt. Ve starších modelech byl použitý kapacitní snímač, v novějších membrána podobná klávesnicím s mechanickými spínači s gumovou membránou.

### ***Magnetické spínače***

Magnetické spínače využívají Hallovy sondy (elektronický prvek reagující na změnu magnetického pole elektrickým napěťovým signálem). Klávesnice jsou velmi kvalitní, ale

také i velmi drahé a proto jsou velmi málo používané.

Vlastnosti jednotlivých typů spínačů jsou shrnuty v Tab. 1:

*Tab. 1 Přehled vlastností jednotlivých typů spínačů*

<i>Technologie</i>	<i>Zdvih</i>	<i>Odezva na stisk</i>	<i>Slyšitelnost</i>	<i>Odolnost</i>	<i>Cena</i>	<i>Využití u PC</i>
Čistě mechanické spínače	Vysoký	Vysoká	Vysoká	Vysoká	Vysoká	Nízké
Membránové spínače	Nízký	Nízká až střední	Nízká až střední	Střední až vysoká	Nízká	Nízké
Mechanické snímače s pěnovými prvky	Střední až vysoký	Nízká	Nízká	Nízká až střední	Nízká	Nízké až střední
Mechanické spínače s gumovou membránou	Střední až vysoký	Střední	Nízká až střední	Střední	Nízká až střední	Vysoké
Kapacitní spínače	Vysoký	Vysoká	Vysoká	Velmi vysoká	Velmi vysoká	Nízké

### 1.2.6 Další součásti klávesnice

Klávesnice se dále skládá z plastového obalu, který ji drží pohromadě udává její vzhled a barvu, volitelné opěrky rukou a výklopných nožiček umožňující nastavení sklonu klávesnice a tím i většího komfortu při psaní, stíněného kabelu zakončeného konektorem, stavových LED a řídicího procesoru, který má na starosti zjišťování stištené klávesy.

Řídicí procesor se stará o dvě důležité věci. Zjišťuje která klávesa byla stištená a následně ji odesílá přes sériové rozhraní do počítače. Zjišťování probíhá pomocí detekce matice, takže není potřeba tak složitý plošný spoj a tolik vstupů řídicího procesoru. V rámci zjišťování klávesy je potřeba vyfiltrovat zákmity, které mohou vzniknout při stisknutí klávesy a které by způsobily detekci vícenásobného stisknutí klávesy, přestože uživatel by chtěl stisknout klávesu jenom jednou. Následně procesor přeloží informaci o stisknuté klávese do takzvaného „scan kódu“, které má každá klávesa dva. Jeden se odesílá v případě stisknutí klávesy, druhý v případě puštění klávesy.

Kabel zajišťuje nejen přenos informací z klávesnice do počítače, ale taktéž napájení klávesnice.

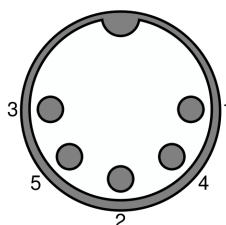
Většina klávesnic je vybavená třemi stavovými LED, které indikují stav klávesnice, ve kterém se nacházejí. První je NumLock, který zobrazuje zda je aktivní numerický blok klávesnice. Druhý je CapsLock, který zobrazuje zda je aktivní psaní verzálek. Třetí je ScrollLock, který zobrazuje zda je aktivní „Scroll Lock“, který v některých programech mění funkčnost kurzorových kláves.

Některé moderní klávesnice obsahují další prvky, které přímo nesouvisí s klávesnicí jako se vstupním zařízením. Jedná se o USB rozbočovač (kdy je možné do klávesnice připojit další zařízení a není nutné jej zapojovat do počítače a plýtvat jeho porty) nebo například kalkulačka nebo LCD displej (který ukazuje například hodiny nebo jiné informace na základě ovládacího software z počítače).

### 1.2.7 Druhy připojení a konektorů

#### *Konektor DIN*

Byl využíván v minulosti pro připojení klávesnice. Obsahuje pět vodičů, pro připojení klávesnice jsou využity pouze čtyři. V současné době se již nepoužívá.

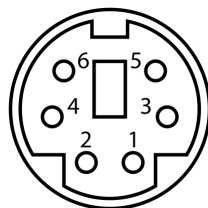


Obr. 23: Zapojení  
konektoru DIN

Význam jednotlivých pinů je následující: (1) CLOCK – hodiny, (2) DATA – slouží k přenosu dat, (3) RESET - slouží k inicializaci klávesnice, ale u novějších klávesnic nebývá zapojen, (4) GND – zem napájení, (5) +5V – napájení.

#### *Konektor PS/2*

Nahradil konektor DIN a i v současné době se využívá pro připojení počítačové klávesnice. Obsahuje šest vodičů, využívají se čtyři stejně jako u konektoru DIN.



Obr. 24: Zapojení  
konektoru PS/2

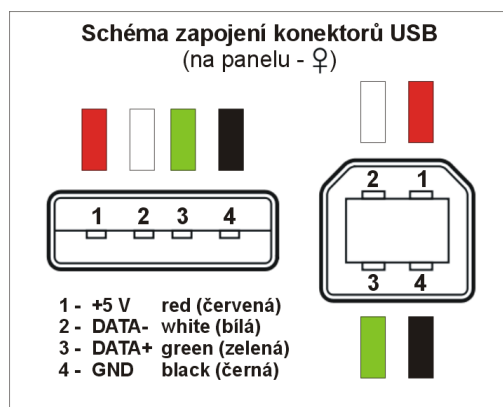
Význam jednotlivých pinů je následující: (1) DATA – slouží k přenosu dat, (2) nezapojeno – u některých typů notebooků může sloužit jako DATA pro druhé zařízení, (3) GND – zem

napájení, (4) +5V – napájení (5) CLOCK – hodiny (6) nezapojeno – u některých typů notebooků může sloužit jako CLOCK pro druhé zařízení.

Stejný konektor slouží i k připojení počítačové myši a proto by mohlo dojít k záměně. Konektory bývají proto barevně odděleny. Pro klávesnici je fialová barva, pro počítačovou myš zelená. V případě, že vlastníme klávesnici s konektorem DIN a v počítači máme konektor PS/2 nebo naopak, lze využít redukce a nemusíme kupovat novou klávesnici.

### ***Konektor USB***

V současné době nejčastěji používaný konektor pro připojení počítačové klávesnice. Umožňuje připojení i dalších periférií jako je počítačová myš, tiskárna, externí paměť a další. Využívá čtyři vodiče, kdy dva datové jsou krouceny, napájecí nikoliv. Celý kabel je potom stíněný hliníkovou fólií. Výhodou je, že je možné klávesnici připojit i odpojit za chodu počítače a není potřeba jej kvůli tomu vypínat. Stejně jako existují redukce mezi konektory DIN a PS/2, stejně tak existují i redukce mezi konektory USB a PS/2.



*Obr. 25: Zapojení konektoru USB*

### ***Bezdrátové klávesnice***

Se v dnešní době začínají masivněji prosazovat. Nepotřebují pro připojení k počítači kabel. Jejich napájení je realizováno z baterií, což je i jednou z nevýhod. Další nevýhodou je relativně jednoduchý odposlech přenášených informací. Přenos informací může být rádiovým přenosem nebo infračerveným přenosem. Infračervený přenos se již nevyužívá, je totiž nutné aby mezi klávesnicí a přijímačem zapojeným v počítači byla přímá viditelnost. V případě připojení pomocí rádiového přenosu, je možné i využít standardu Bluetooth a počítač tedy nemusí být vybavený speciálním přijímačem od výrobce klávesnice, ale postačí standardní Bluetooth přijímač.

Název Bluetooth je odvozen z anglického jména dánského krále Haralda Modrozuba (bluetooth je „modrozub“) vládnoucího v 10. století. Ten využil svých diplomatických schopností k tomu, aby válčící kmeny přistoupily k diskuzi a ukončily vzájemné rozepře. Právě této analogie bylo využito pro název technologie Bluetooth, která podobně jako kdysi král Harald slouží k usnadnění vzájemné komunikace. Technologie Bluetooth je definovaná standardem IEEE 802.15.1. Spadá do kategorie osobních počítačových sítí, tzv. PAN (Personal Area Network). Vyskytuje se v několika verzích, z nichž v současnosti nejvíce využívaná je verze 2.0 a je implementována ve většině aktuálně prodávaných zařízení jako jsou např. mobilní telefony, notebooky, ale i televize. V současné době je nově vyvinuto rozhraní Bluetooth 4.0, u kterého výrobci slibují větší dosah (až 100 metrů), menší spotřebu elektrické energie a také podporu šifrování AES-128. Specifikace Bluetooth 2.0 EDR (Enhanced Data-Rate) zavádí novou modulační techniku  $\pi/4$ -DQPSK a zvyšuje tak datovou propustnost na trojnásobnou hodnotu oproti Bluetooth 1.2 (2,1 Mbit/s). Tímto se dosahuje daleko větší výdrže baterií, protože samotné navázání spojení a i přenos samotný probíhá v daleko kratší době, než u starších verzí Bluetooth. Bluetooth pracuje v ISM pásmu 2,4 GHz (stejném jako u Wi-Fi). K přenosu využívá metody FHSS, kdy během jedné sekundy je provedeno 1600 skoků (přeladění) mezi 79 frekvencemi s rozestupem 1 MHz. Tento mechanismus má zvýšit odolnost spojení vůči rušení na stejné frekvenci. Je definováno několik výkonových úrovní (1 mW, 2,5 mW, 100 mW), s nimiž je umožněna komunikace do vzdálenosti cca 10–100 m. Udávané hodnoty ovšem platí jen ve volném prostoru. Pokud jsou mezi komunikujícími zařízeními překážky (typicky například zdi), dosah rychle klesá. Většinou ovšem nedochází ke skokové ztrátě spojení, ale postupně se zvyšuje počet chybně přenesených paketů. Přenosová rychlost se pohybuje okolo 720 kbit/s (90 KiB/s) a je možné vytvořit datový spoj symetrický případně asymetrický, kdy přenosová rychlost při příjmu (downlink) je vyšší než při odesílání (uplink). Jednotlivá zařízení jsou identifikována pomocí své adresy BD\_ADDR (BlueTooth Device Address) podobně, jako je MAC adresa u Ethernetu. Bluetooth podporuje jak dvoubodovou, tak mnohabodovou komunikaci. Pokud je více stanic propojeno do ad-hoc sítě (tzv. piconet), působí jedna rádiová stanice jako řídicí (master) a může simultánně obsloužit až 7 podřízených (slave) zařízení. Všechna zařízení v pikosíti se synchronizují s taktem řídicí stanice a se způsobem přeskokování mezi kmitočty. Specifikace dovoluje simultánně použít až 10 pikosítí na ploše o průměru 10 metrů a tyto pikosítě dále sdružovat do tzv. „scatternets“ neboli „rozprostřených“ sítí.

### 1.3 Popis funkce klávesnice

Počítačová klávesnice potřebuje ke svému provozu napájení. Klávesnice, která je připojena k počítači kabelem je napájena z počítače jakmile je počítač zapojený do elektrické sítě. Dříve nebyla klávesnice zapnutá, dokud nebyl zapnutý i počítač. Nicméně od doby napájecího standardu ATX je klávesnice pod napětím i při vypnutém počítači (nikoliv odpojenému od elektrické sítě), aby bylo možné počítač zapnout stiskem speciální klávesy, kombinací kláves nebo napsáním speciálního hesla. Bezdrátová klávesnice je napájena z baterií, které jsou do ní vloženy.

Za detekci stisknuté klávesy jsou odpovědné snímače popsané v předchozích kapitolách. Aby nebylo nutné mít pro každý snímač speciální pár vodičů, jsou snímače uspořádány v matici. Řídící mikroprocesor vyšle signál a zkontroluje, zda byla stisknutá nějaká klávesa. Pokud ano, zjistí, že se jednalo například o řádek jedna a sloupec čtyři a dokáže tedy vyslat dále informaci o konkrétní stisknuté klávese na základě mapy znaků, které má uložené ve své permanentní paměti. Mikroprocesor vysílá takzvaný „scan kód“, který má každá klávesa dva (tedy i pokud jsou na klávesnici dvě klávesy „Enter“, tak každá obsahuje svůj vlastní „scan kód“). Přehled „scan kódů“ je uvedený v příloze. Jeden pro stisknutí klávesy a jeden pro uvolnění. Je tedy možné zjistit, že byla klávesa stisknutá a uživatel ji stále drží. Následně mikroprocesor tuhle informaci vyšle přes sériové rozhraní do počítače, kde tuto informaci zpracuje ovladač klávesnice, který se liší podle použitého operačního systému. Nejstarší klávesnice uměly pouze odesílat data, novější umí i data přijímat a nastavovat například stavy modifikátorů a stavové LED indikátory. Jako řadič klávesnice se používal 8-bitový mikroprocesor 8042 od firmy Intel. V dnešní době již není používán, nicméně pro jednoduchost a dosažení kompatibility bývá emulován. K přenosu se využívá datového vodiče, který je synchronizován pomocí signálu v hodinovém vodiči. Takhle komunikují klávesnice s připojením pomocí konektoru DIN nebo PS/2. V případě klávesnic s rozhraním USB jsou tyto data převedeny do protokolu, který se používá pro přenos po této sběrnici. V případě bezdrátových klávesnic je v klávesnici obsažen ještě vysílač, který data vyšle vzduchem k přijímači. Znat způsobem jakým klávesnice odesílá svá data je důležitý při konstrukci hardwarových keyloggerů.

## 2 METODY ODPOSLOUCHÁVÁNÍ KLÁVESNICE

Rozeznáváme dva základní druhy odposlouchávání klávesnice (keyloggingu) a to softwarový keylogging, kdy je nutné, aby na odposlouchávané stanici běžel nějaký záškodnický software a hardwarový keylogging, kdy je potřeba nějakého dalšího zařízení, které je buďto nutné vložit k odposlouchávané klávesnici nebo stačí zařízení mít a samotný přístup k odposlouchávané klávesnici není potřeba.

### 2.1 Právní aspekty odposlouchávání klávesnice

Legálně využívané keyloggery představují pouhý zlomek jejich využití. Může se jednat o zabezpečení vlastního počítače, kdy nainstalovaný keylogger dokáže zjistit, zda počítač nevyužíval někdo jiný případně v zaměstnání, kdy může zaměstnavatel kontrolovat využití počítače k pracovním účelům. Většina keyloggerů nicméně slouží k záškodnické činnosti a k získávání údajů bez vědomí uživatele a k jejich následnému zneužití.

#### 2.1.1 Zásah do Ústavních i zákonných práv

Dle čl. 10 odst. 2 Listiny základních práv a svobod má každý právo na ochranu před neoprávněným zasahováním do soukromého a rodinného života. V každém případě je nutno jakékoli utajené sledování činnosti uživatele na počítači, s nímž tento uživatel neprojevil souhlas, nebo které nemá podklad v zákoně, považovat za odporující tomuto ustanovení Listiny. Dále Listina v čl. 13 stanoví nedotknutelnost listovního tajemství a tajemství jiných písemností nebo záznamů, uchovávaných v soukromí nebo zasílaných poštou nebo jiným způsobem (s výjimkou případů, stanovených zákonem). Listovní tajemství se dle tohoto článku vztahuje i na zprávy podávané telefonem nebo jiným podobným zařízením - v případě, že je pomocí keyloggeru sledována e-mailová nebo jakákoli jiná komunikace, jde o současné porušení práva na soukromí i listovního tajemství. Určitým promítnutím Ústavních zásad do zákonných norem je ustanovení § 11 občanského zákoníku o ochraně osobnosti - fyzická osoba má právo na ochranu své osobnosti, mimo jiné i soukromí. Ochranou soukromí se rozumí i respektování telekomunikačního tajemství, které je pochopitelně použitím keyloggeru bez vědomí uživatele počítače porušováno více než razantně. Jako obranu osobnostních práv, a tedy i práva na soukromí, nabízí občanský zákoník žalobu na ochranu osobnosti, v rámci níž se lze domáhat upuštění od neoprávněných zásahů do práva na ochranu osobnosti, odstranění následků těchto zásahů, a přiměřeného zadostiučinění (i v penězích). Vedle osobnostních

práv zasahuje neoprávněné použití keyloggeru i do práva od osobnostních práv odvozeného - práva na ochranu osobních údajů. Osobním údajem je podle zákona č. 101/2000 Sb. jakákoliv informace týkající se určitého subjektu údajů, podle níž lze tento subjekt spolehlivě identifikovat (plné znění definice osobního údaje viz. § 4 písm. a) zák. na ochranu osobních údajů). Osobním údajem tedy ještě není pouhé jméno a příjmení, je jím však jméno, příjmení a telefonní číslo, jméno, příjmení a datum narození, e-mailová adresa (nikoliv z domény veřejného poskytovatele, ale např. z domény zaměstnavatele), apod. Osobním údajem par excellence je rodné číslo - dvě různé osoby nemohou mít za žádných okolností stejné rodné číslo. Subjekt, který s osobními údaji nakládá, je považován za správce osobních údajů. Správce, který zpracovává osobní údaje bez souhlasu subjektu údajů mimo případy uvedené v zákoně, a neplní zákonnou oznamovací povinnost, se dopouští přestupku na poli ochrany osobních údajů, za který mu může být Úřadem pro ochranu osobních údajů udělena pokuta až 1.000.000,- Kč. Je však pravdou, že vzhledem k povaze ostatních možností nápravy je postup z titulu ochrany osobních údajů pro poškozeného značně neefektivní (Úřad na ochranu osobních údajů je správním orgánem, který může udělovat pokuty v rámci zákona o ochraně osobních údajů, nicméně kvůli stanovení dalších povinností nebo sankcí za porušení osobnostních práv je třeba zahájit řízení před soudem).

### 2.1.2 Trestněprávní odpovědnost

Neoprávněné užívání keyloggeru samozřejmě může být za určitých okolností kvalifikováno jako trestný čin. V trestním zákoně lze nalézt dvě skutkové podstaty, jichž se takové jednání může dotýkat - trestný čin „Poškození a zneužití záznamu na nosiči informací“ dle § 257a tr. zákona, a „Porušování tajemství dopravovaných zpráv“ dle § 239 tr. zákona. Prvně jmenovaná skutková podstata dopadá na situaci, kdy pachatel získá přístup k nosiči informací a v úmyslu způsobit jinému škodu nebo získat pro sebe nebo třetí osobu neoprávněným prospěch tyto informace neoprávněně zneužije (odst. 1 písm. a), nebo učiní zásah do technického nebo programového vybavení počítače nebo jiného telekomunikačního zařízení (odst. 1 písm. c). Základním předpokladem této skutkové podstaty je získání přístupu k nosiči informací (záleží na tom, jak široce soud tento pojem vyloží), a úmysl orientovaný ke způsobení škody jinému nebo k získání neoprávněného prospěchu. Ustanovení o porušování tajemství dopravovaných zpráv je možno aplikovat vždy, když je s pomocí keyloggeru monitorována elektronická komunikace uživatele kontrolovaného počítače s jinou osobou. Vzhledem k tomu, že keylogger může velice

snadno posloužit k získání přístupových hesel např. do internetového bankovníctví, této problematiky se snadno mohou dotýkat i trestné činy krádeže dle ustanovení § 247 tr. zákona, případně podvodu dle ustanovení § 250 tr. zákona.

### 2.1.3 Zákoník práce a kontrola zaměstnanců

Asi nejaktuálnějším problémem, řešeným v souvislosti s keyloggery (vedle keyloggerů jako spyware) je jejich použití na pracovišti ke kontrole zaměstnanců. Zaměstnavatelé se v zájmu kontroly řádného výkonu práce zaměstnanců na svěřených počítačích i v zájmu ochrany obchodního tajemství uchylují k nejrůznějším formám této kontroly, které mohou sahát od obyčejného procházení historie v prohlížeči přes kontrolu uložených dat až po instalaci speciálních kontrolních programů, včetně keyloggerů. Pracovní prostředí má z tohoto hlediska mimo jiné i tu výhodu, že na své počítače může zaměstnavatel snadno instalovat jak HW, tak i SW keylogger, a málokterý zaměstnanec instalaci odhalí. Zaměstnavatel má na určitou míru kontroly právo ze zákona. Toto právo vychází z ustanovení § 301 zákoníku práce, podle něhož zaměstnanci nesmějí bez souhlasu zaměstnavatele užívat pro svou osobní potřebu výrobní a pracovní prostředky zaměstnavatele včetně výpočetní techniky ani jeho telekomunikační zařízení. Dodržování tohoto zákazu je zaměstnavatel oprávněn přiměřeným způsobem kontrolovat. Kontrolní právo zaměstnavatele potvrdil svými rozhodnutími i Nejvyšší soud ČR (např. rozsudek č.j. 21 Cdo 2172/2002). Toto kontrolní právo je však v neustálém protikladu k opačnému zájmu zaměstnance na respektování jeho soukromí a ochranu osobních údajů. Podle § 316 odst. 2 zákoníku práce „nesmí zaměstnavatel bez závažného důvodu spočívajícího ve zvláštní povaze činnosti zaměstnavatele narušovat soukromí zaměstnance na pracovištích a ve společných prostorách zaměstnavatele tím, že podrobuje zaměstnance otevřenému nebo skrytému sledování, odposlechu a záznamu jeho telefonických hovorů, kontrole elektronické pošty nebo kontrole listovních zásilek adresovaných zaměstnanci“. Na základě četné judikatury (zejména zahraniční) bylo zaujato stanovisko, podle něhož má zaměstnavatel právo přiměřeným způsobem kontrolovat způsob, jakým zaměstnavatel využívá pracovní dobu, tedy dobu strávenou na internetu a navštívené stránky, obsah paměti počítače, četnost a odesílatele e-mailů, došlých do přidělené e-mailové schránky zaměstnance, nebo skutečnost, že zaměstnanec využívá instant messenger. Toto právo kontroly vyplývá ze zákona, a zaměstnavatel nemá povinnost o ní zaměstnance informovat. Pokud by tedy zaměstnavatel nainstaloval keylogger do počítače, využívaného zaměstnancem, a využíval tento keylogger skutečně pouze ke sledování výše uvedených

aspektů činnosti zaměstnance, byl by tento způsob využití keyloggeru možný, dokonce i bez souhlasu zaměstnance. Zakázanou formou kontroly je však sledování obsahu jakékoli komunikace soukromého charakteru, kterou zaměstnanec za pomoci svěřeného počítače provádí. Zákoník práce umožňuje takový zásah do základních práv zaměstnance pouze v případech, kdy jsou dány závažné důvody, spočívající ve zvláštní povaze činnosti zaměstnavatele. Takovými zaměstnavateli jsou např. banky, různá call centra nebo pracoviště, na nichž zaměstnanci přicházejí do styku s velkými finančními částkami. V těchto případech je vždy účelem kontroly ochrana majetku zaměstnavatele, a nikoliv samotné sledování zaměstnance. O těchto závažnějších způsobech kontroly je zaměstnavatel povinen zaměstnance informovat, stejně jako o rozsahu kontroly. Jiným zaměstnavatelům není sledování obsahu komunikace a podobné způsoby narušování soukromí zaměstnanců umožněno - právo na ochranu soukromí má v tomto případě přednost před právem zaměstnavatele na kontrolu řádného výkonu pracovních povinností. Použití keyloggeru je tedy teoreticky možné v případě, že by zaznamenával výhradně údaje, k jejichž sledování zaměstnavatel nepotřebuje svolení zaměstnance. Z povahy keyloggeru však plyne, že těžko budou jím zaznamenávaná data omezena pouze na tyto údaje, když mohou zaznamenávat každou stisknutou klávesu a přesný stav obrazovky (např. v situaci, kdy je na ní zobrazen soukromý e-mail zaměstnance). Proto je jeho použití v pracovněprávních vztazích krajně složité, a zaměstnavatel by se vystavoval poměrně značnému riziku. Navíc riziku zbytečnému, když v dnešní době existují inteligentnější programy na monitorování činnosti zaměstnanců na počítačích, při jejichž použití konflikt se zájmem zaměstnance na ochranu svého soukromí nehrozí.

## 2.2 Softwarový keylogging

Při keyloggingu pomocí software je vždy nutné nějakým způsobem nainstalovat na odposlouchávaný počítač záškodnický software. Tento software můžeme nainstalovat na počítač sami, pokud k němu máme přístup nebo je možné jej na počítač dostat pomocí ostatního škodlivým softwarem (pomocí virů, trojských koňů, spyware, malware ...). Výhodou softwarových keyloggerů jsou v tom, že můžeme zaznamenávat i další akce na odposlouchávaném počítači a to například pohyb myši, data ze schránky, data z webových formulářů. Další výhodou je, že nejsme závislí na konkrétním rozložení klávesnice (jak hardwarovém, tak i v operačním systému), jelikož si dokážeme skutečně zjistit, jakou klávesu uživatel zamýšlel stisknout. Nevýhodou je, že data musíme nějakým nepozorovaným způsobem dostat z počítače. To je možné provést buďto přístupem k

odposlouchávanému počítači a stažením dat případně pomocí vzdáleného přístupu. Data je možné odeslat pomocí na databázový, webový nebo ftp server. Data mohou být odesílána na emailovou adresu nebo je možné odesílat přes počítačovou síť vlastními protokoly. Tato nevýhoda může být někdy i vhodná, jelikož po nainstalování a skrytí keyloggeru před uživatelem už nepotřebujeme přístup k počítači. Druhou a závažnější nevýhodou je, že samotný keylogger je potřeba skrýt před uživatelem. Úspěšnost softwarového keyloggeru závisí na tom, jak se mu podaří potlačit tuto nevýhodu. Software si můžeme napsat vlastní případně využít volně dostupný software, který je zdarma nebo využít komerční software, který je v nabídce v cenách už od řádově stokorun. Softwarové keyloggery můžeme rozdělit do pěti následujících kategorií.

### 2.2.1 Na základě hypervizoru

Keylogger na základě hypervizoru využívá hypervizoru běžícího na nižší úrovni než je operační systém, stavá se virtuálním strojem. Typickým příkladem je například software Blue Pill.

#### *Hypervizor*

Hypervizorem (též Virtual Machine Monitor, VMM) se označuje jedna z technik virtualizace hardware počítače, která umožňuje na jednom počítači spustit zároveň více operačních systémů. Hypervizor je nejvyšším řídicím prvkem, který řídí přístup virtualizovaných počítačů k hardwaru počítače a jejich běh a zároveň je od sebe odděluje. Rozlišujeme dva typy hypervizorů. Hypervizor typu 1 běží přímo na hostitelském hardwaru. Zde monitoruje a řídí běh virtualizovaných strojů z hlediska hardwarových prostředků. Hostovaný operační systém tak funguje na jiné úrovni, pod hypervizorem. Tento model představuje klasické provedení architektury virtuálního stroje, původní hypervizor byl CP/CMS, vyvinutý v IBM v roce 1960. Dnes tento typ můžeme najít například u hypervizoru ESXi od firmy VMware. Hypervizor typu 2 je spuštěn v konvenčním prostředí operačního systému. Vrstva hypervizora se nachází nad vrstvou operačního systému. Podle implementace rozlišujeme hardwarovou virtualizaci umožňující virtualizaci nemodifikovaných operačních systémů, která však vyžaduje podporu virtualizace přímo v procesoru (to nicméně není v dnešní době závažný problém, jelikož téměř každý moderní procesor používaný v osobním počítači je podporou virtualizace vybavený), a softwarovou virtualizaci, která je obvykle méně výkonná, protože jednoduché operace vstupu a výstupu musí být obsluhovány komplexními obslužnými funkcemi.

### ***Blue Pill***

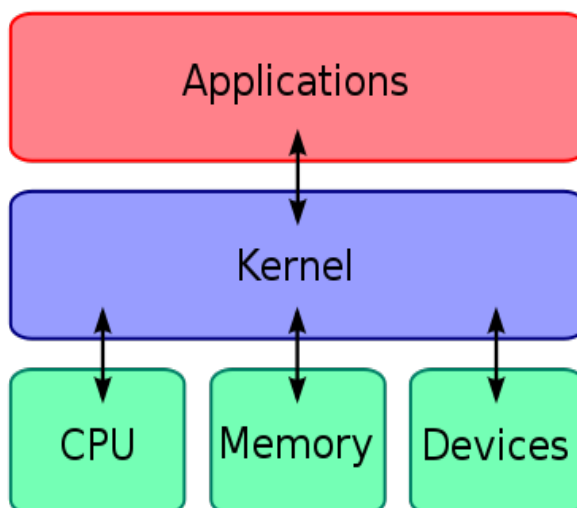
Je kódové označení rootkitu založeného na virtualizaci procesorů třídy Intel x86. Vytvořila jej Joanna Rutkowská a byl poprvé prezentován na „Black Hat Briefings“ v roce 2006. Referenční implementace byla pro operační systém Microsoft Windows Vista. Program původně vyžadoval virtualizaci od firmy AMD (AMD-V), nicméně následně byl naprogramován i pro virtualizaci VT-x od firmy Intel. Blue Pill vytvoří tenký hypervizor, pod kterým následně běží celý operační systém. Má tedy přístup k jakékoliv akci, kterou uživatel provede a měl by být dle tvůrců stoprocentně nezjistitelný. Později byly uvolněny zdrojové kódy pro program, nicméně podléhají relativně omezujícím licenčním podmínkám. Původ názvu je z filmu Matrix, kde hlavní hrdina dostane na výběr mezi modrou a červenou pilulkou. Modrá pilulka by znamenala, že dále nebude vědět, že žije ve virtuálním světě ovládaném stroji. Stejně tak se snaží tvářit i program.

#### **2.2.2 Na základě kernelu**

Tato metoda je těžká jak na napsání tak na boj proti ní. Keylogger se usadí na úrovni jádra (kernelu) operačního systému a následně monitoruje akce, které provádí uživatel. Jelikož je na úrovni jádra operačního systému, je těžké pro aplikace běžící nad jádrem detekovat jeho přítomnost. Často bývají implementovány jako rootkit, který poškodí jádro operačního systému a dokáže získat přístup k celému počítači nebo jako ovladač zařízení, který nahradí standardní ovladač klávesnice nainstalovaný v systému.

### ***Jádro (kernel)***

Jádro je zavedeno do operační paměti při startu (bootování) počítače a je mu následně předáno řízení. U pokročilých operačních systémů jádro nikdy neztrácí kontrolu nad počítačem a po celou dobu jeho běhu koordinuje činnost ostatních běžících procesů. Podle architektury operačního systému rozlišujeme mikrokernél (mikrojádru, jádro je velice jednoduché a minimalistické a obsahuje pouze zcela základní funkce, zbytek operačního systému je v aplikacích) a makrokernél (jádro je velmi rozsáhlé, obsahuje velké množství funkcí pro všechny možné činnosti operačního systému). Kompromisem je modulární jádro, které je fakticky makrojádrem, ovšem jeho podstatná část je tvořena moduly, které je možné přidávat a odebírat za běhu systému. Základním účelem jádra je ovládání prostředků počítače a vytvoření prostředí pro běh ostatních programů. Jádro také obvykle poskytuje metody pro synchronizaci a komunikaci mezi procesy.



Obr. 26: Schéma umístění kernelu (jádra)

### **Rootkit**

Rootkitem rozumíme sadu technologií a počítačových programů pomocí kterých lze maskovat přítomnost škodlivého software v počítači. Rootkit schovává adresáře, volání API, položky registru Windows, procesy případně síťové spojení tak, aby přítomnost zákeřného software nebyla běžně dostupnými systémovými prostředky odhalitelná. Dále bez vědomí uživatele odesílá tyto informace dále.

### **Ovladač zařízení**

Zprostředkovává komunikaci mezi hardwarem a jádrem operačního systému případně aplikacemi. Ten, kdo píše ovladač zařízení musí znát přesně jak hardware funguje. Na druhé straně je potřeba dodržet formát a způsob předávání dat dle příslušného operačního systému, pro který je ovladač zařízení napsaný.

#### **2.2.3 Na základě API**

Keylogger využívá API funkce příslušného operačního systému, pod kterým běží. Může buďto využít volání API, kdy se na ně pověsí a použije výsledek, případně může toto volání nahradit a získat uživatelské data. API označuje v informatice rozhraní pro programování aplikací. Jedná se o sbírku procedur, funkcí či tříd nějaké knihovny (ale třeba i jiného programu nebo jádra operačního systému), které může programátor využívat. API určuje, jakým způsobem se funkce knihovny volají ze zdrojového kódu programu. Příklad jednoduchého keyloggeru pro operační systémy Windows pro jazyk C#:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Utilities;
using System.IO;

namespace watcher
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        globalKeyboardHook gkh = new globalKeyboardHook();
        private void HookAll()
        {
            foreach (object key in Enum.GetValues(typeof(Keys)))
            {
                gkh.HookedKeys.Add((Keys)key);
            }
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            gkh.KeyDown += new KeyEventHandler(gkh_KeyDown);
            HookAll();
        }

        void gkh_KeyDown(object sender, KeyEventArgs e)
        {
            DateTime now = DateTime.Now;
            using (StreamWriter SW = new
                StreamWriter(@"c:\temp\Keylogger " + now.Month + "-" + now.Day +
                    ".txt", true))
            {
                switch (e.KeyCode)
                {
                    case Keys.Space:
                        SW.Write(" ");
                        break;

                    case Keys.Enter:
                        SW.Write(Environment.NewLine);
                        break;

                    case Keys.LShiftKey:
                        SW.Write("{SHIFT}");
                        break;

                    default:
                        if (Control.ModifierKeys != Keys.Shift)
                            SW.Write(e.KeyCode.ToString().ToLower());
                        else
                            SW.Write(e.KeyCode);
                        break;
                }
            }
        }
    }
}
```

```
        SW.Close();  
    }  
} }  
}
```

Metoda je jednoduchá na naprogramování, nicméně může zvyšovat zatížení procesoru počítače a díky tomu může být keylogger odhalený a také může docházet k náhodným výpadkům v sekvenci odposlouchávaných kláves.

#### 2.2.4 Sledování dat z formulářů

Keylogger čeká na odeslání dat webového formuláře internetovým prohlížečem pomocí události – onSubmit (skriptování). Data formuláře si keylogger uloží ještě než jsou předána přes internet, což mu umožní obejít i http šifrování.

#### 2.2.5 Analyzátoři paketů

Programy založené na tomto principu zachytávají veškerou síťovou komunikaci, kterou následně analyzují a umožňují přístup k citlivým datům uživatele.

### 2.3 Hardwarový keylogging

Nevyžaduje neoprávněný zásah do softwarového vybavení počítače nebo klávesnice a dá se rozlišit na dvě základní skupiny. V prvním případě je potřebné mít přístup k odposlouchávané klávesnici případně počítači a instalaci hardwaru, který nám zachytávání kláves zprostředkuje. V druhém případě nám stačí mít zařízení, které odposlech provede a není nutné mít kontakt s příslušným počítačem nebo klávesnicí.

#### 2.3.1 Na základě modifikace firmware

Jedná se o modifikaci BIOSu počítače tak, aby rovnou zaznamenával klávesy. Uživatel tedy nemá šanci cokoliv zjistit. Nicméně je potřeba fyzického přístupu k počítači a velkých technických znalostí. BIOS implementuje základní vstupně–výstupní funkce pro počítače PC a představuje vlastně firmware pro osobní počítače. V současné době se BIOS používá hlavně při startu počítače pro inicializaci a konfiguraci připojených hardwarových zařízení a následnému zavedení operačního systému, kterému je pak předáno další řízení počítače. Programový kód BIOSu je uložen na základní desce v stálé paměti typu ROM, EEPROM nebo modernější flash paměti s možností jednoduché aktualizace.

### 2.3.2 Vložené zařízení

Nejtypičtější a nejdostupnější z hardwarových keyloggerů. Jedná se o zařízení, které se vloží mezi konektor klávesnice a konektor v počítači, případně se schová přímo do počítačové klávesnice.



*Obr. 27: Příklad keyloggeru pro konektor PS/2*

Je tedy velmi obtížné pro uživatele jej objevit. Zařízení se skládá z mikroprocesoru a paměti, do které se ukládají zachycené klávesy. Cena zařízení se pohybuje již od několika stokorun, případně není problém pro středně pokročilého radioamatéra zařízení vyrobit za odpoledne.



*Obr. 28: Modul keyloggeru před vložením do počítače*

### 2.3.3 Bezdrátové odposlouchávání

Využívá odposlouchávání přenosu u bezdrátové klávesnice a následnou analýzou signálu. Využívá se toho, že komunikace například přes Bluetooth nebývá šifrovaná. Odposlech může fungovat na vzdálenost, na kterou funguje přenos mezi klávesnicí a počítačem, tedy řádově metry až desítky metrů. Pokud je komunikace šifrovaná, odposlech se stane obtížnějším o nutnost dešifrování signálu.

### 2.3.4 Využití elektromagnetického záření

Využívá kompromitujícího elektromagnetického záření. Slovní spojení kompromitující elektromagnetické vyzařování je doslovným překladem anglického „compromising

electromagnetic emanations“. Jedná se o záření, které je druhotným efektem přenosu dat kabelem, vodivým spojem na desce plošných spojů nebo bezdrátovým přenosem. Toto záření je vytvářeno vždy. Jeho vytváření není úmyslné a jeho zachycením a příslušným dekódováním je možné získat citlivá data, která by měla být dobře chráněna. V popisu budu vycházet z bakalářské práce David Swiateka.

### ***Metody zachytávání elektromagnetického záření***

První ze standardních metod využívá spektrálního analyzáru k detekci přenosu signálu. Ten lze využít pouze v případě dostatečně dlouhé doby přenosu a proto je s jeho pomocí těžko odhalitelný přenos kompromitujícího signálu složený pouze ze signálů, které mají tvar špiček.

Druhá standardní metoda je založena na širokopásmovém přijímači naladěném na konkrétní frekvenci. Proces detekce signálu spočívá v testování signálu v celém frekvenčním rozsahu přijímače a demodulaci signálu v závislosti na jeho frekvenční nebo amplitudové modulaci. Když je objevena frekvence, která nás zajímá, tak se pomocí úzkopásmové antény a příslušných filtrů oddělí signál o hodnotách kompromitujícího elektromagnetického signálu.

Některé přímé a nepřímé vyzařování může zůstat u obou standardních metod skryto, především signály složené z nepravidelných špiček nebo proměnných nosných frekvencí. Spektrální analyzáry potřebují především neměnnou nosnou frekvenci. Podobně ani příjem širokopásmových přijímačů není okamžitý a potřebuje hodně času na pokrytí celého frekvenčního rozsahu. Navíc demodulační proces může část elektromagnetického vyzařování skrýt. Proto byla navržena třetí metoda k zachycení záření z klávesnic. Nejprve je získán kompletní surový signál přímo z antény namísto filtrování a demodulace signálu s omezenou šířkou pásma. Potom je vypočítána krátkodobá Fourierova transformace, která dává trojrozměrný signál s časem, frekvencí a amplitudou. Výpočtem krátkodobé Fourierovy transformace dokážeme odhalit i jen krátké špičky a všechny nosné frekvence.

### ***Popis experimentu***

David Swiatek provedl i experiment k potvrzení faktu vyzařování elektromagnetického záření počítačovou klávesnicí. Pro pokus vybral čtyři různé prostředí. První byla profesionální polo-zvukotěsná komora o rozměrech 7 krát 7 metrů. Důvodem bylo odstínit rušivé elektromagnetické vyzařování z okolí. Anténa byla umístěna 5 metrů od klávesnice

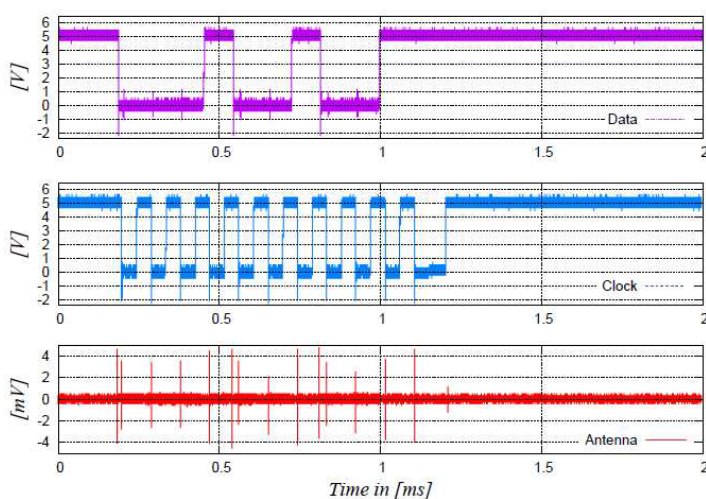
zapojené do počítače. Testovaná klávesnice byla na stole 1 metr vysokém a počítač byl položen na zemi. Druhou místností byla kancelář o rozměrech 3 krát 5 metrů se dvěma napájenými počítači a třemi LCD displeji. Elektromagnetický šum v pozadí tvořilo 40 počítačů vzdálených 10 metrů od kanceláře, více než 60 počítačů na patře a 802.11 wifi router vzdálený minimálně 3 metry od kanceláře. Anténa byla posouvána zpět skrz otevřené dveře až do desetimetrové vzdálenosti od klávesnice s cílem určit maximální vzdálenost. Kancelář byla vybrána z důvodu ověření schopnosti útoku i v prostředí s elektromagnetickým rušením na pozadí. Třetí místností byla přilehlá kancelář. Elektromagnetické vlnění bylo měřeno ze sousední kanceláře přes 15 cm širokou zeď ze dřeva a sádkartonu. Čtvrtou místností byl byt, který se nacházel v budově s pěti podlažními v centru středně velkého města. Klávesnice byla v pátém patře. První měření proběhlo s anténou umístěnou na stejném patře, poté byla přesunuta do nejvzdálenějšího přízemí minimálně 20 metrů od odposlouchávané klávesnice.

Jako anténa byla použita dvoukružlová anténa, aby se zlepšil poměr signálu šumu, jelikož většina vyzařování je v pásmu mezi 25 a 300 MHz. Také bylo ozkoušeno, zda je možné zařízení zachytit menší anténou jako například smyčkou z 1 metru dlouhého měděného drátu. Jako testované klávesnice posloužilo 12 různých modelů zakoupených mezi roky 2001 a 2008. Jednalo se o 7 kusů klávesnic s PS/2 připojením, 2 klávesnice s USB připojením, 2 notebookové klávesnice a 1 bezdrátovou klávesnici. Aby se vyloučily vodivé spojení přes napájecí kabel, byly klávesnici připojeny k notebooku napájeného z baterie.

### ***Metody***

Klávesnice byla umístěna do zvukové komory, signál bych zachytáván anténou a průběh sledován na osciloskopu. Jednalo se o surová data, jelikož nebylo uplatněno žádné filtrování nebo frekvenční omezení. Vzhledem k tomu, že paměť osciloskopu není neomezená, bylo třeba přesně spustit datový výběr. Za prvé byly použity nejbližší sestupné hrany signálů dat vyslaných při stisku klávesy. Sonda byla připojena k datovému vodiči mezi klávesnicí a počítačem. S pouhým jedním zachycením je možné reprezentovat celé spektrum po celou dobu zachytávání. Mimo spektra zachyceného signálu je k dispozici i časový průběh signálu. Zejména byly známy některé nosné (vertikální linky) a širokopásmové impulzy (horizontální linie). První tři metody jsou založeny na těchto kompromitujících vyzařováních. První je technika sestupné hrany signálu. Byla využita pro komunikaci po rozhraní PS/2. Jedná se o nízkorychlostní obousměrné rozhraní mezi

zařazením master a slave. Hodinový signál určuje okamžik vzorkování dat. Jeho kmitočet se pohybuje mezi 10 kHz a 16,7 kHz. Zařízení slave může zahájit přenos jednoho bytu, pokud jsou obě linky pro přenos dat (CLK i DATA) v úrovni high po dobu nejméně 50  $\mu$ s. Datový rámec se skládá z jedenácti bitů. Přenos je zahájen start bitem s úrovní low, následuje 8 bitů dat řazených od nejméně významného, dále je zde paritní bit (lichá parita - high pro sudý počet jedniček v datech) a konečně stop bit s úrovní high.



Obr. 29: Datový, hodinový a kompromitující signál při stisku klávesy 0x24 (E na anglické klávesnici)

Metoda je založena na rozpoznání sestupných hran signálu trvajících 200 ns na rozdíl od delších náběžných hran, které trvají 2  $\mu$ s. Proto by mělo být kompromitující vyzařování mnohem silnější s vyšší maximální frekvencí právě na sestupné hraně. Jelikož jsou signály hodin a dat generovány společně, je i kompromitující záření jejich kombinací. Porovnáním grafu datového signálu a kompromitujícího záření bylo však zjištěno, že vrcholy zachycené anténou nemusí nutně generovat sestupné hrany datového a hodinového signálu, ale jsou pravděpodobně vytvářeny sepnutím tranzistoru. Nicméně i tak mohou být zneužity, přestože se však může objevit i kolize znaků. Pokud vezmeme v úvahu jen sestupné hrany signálu, mají například znaky 0x24 (E na anglické klávesnici) a 0x34 (G na anglické klávesnici) stejnou stopu. Další technikou je technika zobecněného přenosu. Pokud je mezi dvěma stopami jedna vzestupná hrana datového signálu a je možné ji zachytit, pak je také možné plně obnovit stisky kláves. Pro zvýraznění kompromitujícího vyzařování na vzestupné hraně datového přenosu lze použít softwarovou pásmovou propust', která izoluje frekvence širokopásmových impulsů. Tímto filtračním procesem také výrazně zlepšujeme

odstup signálu od šumu (SNR), čímž se zvyšuje také efektivita algoritmu detekce vrcholů. Třetí technikou je modulační technika. Některé elektromagnetické emise pocházejí z neúmyslného vyzařování emitovaného hodinami, nelineárními prvky v klávesnici. Nosná frekvence je přibližně 4 MHz, což je velmi pravděpodobně hodnota frekvence vnitřních hodin mikrokontroléru klávesnice. Zajímavé je, že pokud byly tyto harmonické srovnány jak s hodinovými tak datovými signály, jasně byly vidět modulované signály (v amplitudě a tak i ve frekvenci), které plně popisují stav hodinového i datového signálů. To znamená, že scan kód klávesy může být kompletně získán z harmonických frekvencí. Stojí za povšimnutí, že i když se objeví některá silná elektromagnetická rušení, může útočník využít harmonických frekvencí, které nejsou ovlivněny interferencí, aby získal jasný signál. Ve srovnání s předchozími technikami, modulace založená na nosných frekvencích se stává mnohem zajímavější pro vzdálený příjem. Amplitudově a frekvenčně modulované přenosy jsou obecně méně rušeny šumem a překážkami. Tato nepřímá vyzařování, která nemají formální vysvětlení a jsou pravděpodobně založena na přeslechu se zemí, vnitřních hodinách mikroprocesoru, datových a hodinových signálech. Nicméně umožní útočníkovi obnovit psaný text na klávesnici. Poslední technikou je technika skenování matice. Výše popsané techniky jsou využitelné na klávesnici s připojením pomocí PS/2. Metoda skenování matice je využitelná jak pro klávesnice s připojením USB, tak i dokonce pro bezdrátové klávesnice. Téměř všechny klávesnice sdílejí stejný způsob detekce stisknutí klávesy. Hlavní technické omezení je v tom, že aby byla klávesa považována za stisknutou, musí být stlačena na dobu 10 ms. Během této doby by měl být zjištěn každý stisk. Výrobci se snaží omezovat náklady na vývoj přístroje. Proto se nepoužívá řešení, kdy se skenuje stisk každé klávesy odděleně, které je nákladné cenově, nákladné na řízení a vyžaduje velkou skenovací rutinu a také větší zpoždění. Proto se používá maticové uspořádání. Klávesnicový mikrokontrolér (často 8-bit procesor) analyzuje sloupce jeden po druhém a získává stav osmi tlačítek najednou. Proces skenování matice lze popsat jako 192 kláves uspořádaných v 24 sloupcích a 8 řádcích. Sloupce jsou spojeny s řídicím čipem, řádky jsou připojeny k detekčnímu čipu. Klávesy jsou umístěny v průsečíku sloupců a řádků. Ke zjištění, která klávesa v matici byla stlačena využívá klávesnice podprogram a ten vyžaduje určitý čas. Sloupce v matici vytvářejí dlouhé vedení, protože spojují zpravidla osm tlačítek. Díky tomu trvá přenos alespoň 3 $\mu$ s a vytváří kompromitující elektromagnetické záření. Útočník, který elektromagnetické záření zachytí, může snadno obnovit sloupce stisknutých kláves. Ve skutečnosti budou tyto impulzy částečně zpožděné a pro obnovení úhozů je třeba průběžně sledovat kompromitující elektromagnetické vyzařování způsobené rutinou

skenovací matice konkrétním spoušťovým modelem. Šest prvních špiček je vždy přítomno stejně jako poslední tři. Nikdy nechybí ani se nezpožďují. Proto se tato pevná struktura používá k definici spoušťového modelu. Kromě toho čtení matice nepřetržitě vyznačuje kompromitující záření od stisku klávesy. Když je zjištěna podmnožina úhozu, získává se více příkladů až do zjištění dalšího vzoru. Proto je vybírán nejčastěji zachycený vzor. Tato metoda je vzhledem k zaměnitelnosti hledem k zaměnitelnosti znaků o něco méně účinná než metoda první.

### *Výsledky experimentu*

Útok lze považovat za úspěšný, když je možné správně obnovit alespoň 95% z více než 500 úhozů. Analýza ukazuje, že vyznačování z klávesnice je v praktickém scénáři skutečně problematické a vyhodnocení rizik způsobených elektromagnetickým zářením není snadné. Nicméně v ideálním prostředí zvukotěsné komory je alespoň na jednu metodu odposlechu náchylná vždy alespoň jedna klávesnice, jak je vidět v příložené tabulce.

*Tab. 2: Přehled účinnosti metod*

Klávesnice	Typ	Metoda	Metoda	Metoda	Metoda
		1	2	3	4
A1	PS/2	x	x	x	x
A2	PS/2	x	x		x
A3	PS/2	x	x	x	x
A4	PS/2	x	x	x	
A5	PS/2	x	x	x	x
A6	PS/2	x	x		x
A7	PS/2	x			x
B1	USB				x
B2	USB				x
C1	Laptop	x	x		x
C2	Laptop				x
D1	Wireless				x

Při použití v dalších prostředích bylo dosaženo taktéž úspěchu. Zkracovala se pouze maximální použitelná vzdálenost.

### **2.3.5 Překrytí klávesnice**

Využívá se většinou u klávesnic bankomatů, kdy je původní klávesnice jakoby překrytá klávesnicí novou. Uživatel zadává data, která jsou jak zachytávána, tak i posílána dále,

takže není možné zjistit, že jsem odposloucháván.

## 2.4 Keylogging pomocí analýzy zvukového signálu

Využívá toho, že každá klávesa nevydává při stisku úplně stejný zvuk. Na toto téma bylo uveřejněno několik prací, které tady postupně rozeberu po krátkém teoretickém úvodu, kde rozeberu techniky a technologie použité v samotných pracích.

### 2.4.1 Fourierova transformace

Fourierova transformace je vyjádření časově závislého signálu pomocí harmonických signálů (goniometrických funkcí sinus a cosinus), obecně tedy funkce komplexní exponenciály. Slouží pro převod průběhu signálů z časové oblasti do oblasti frekvenční. Signál může být buď ve spojitém či diskrétním čase. Fourierova transformace  $S(\omega)$  funkce  $s(t)$  je definována integrálním vztahem:

$$S(\omega) = \int_{-\infty}^{\infty} s(t) e^{-j\omega t} dt \quad (1)$$

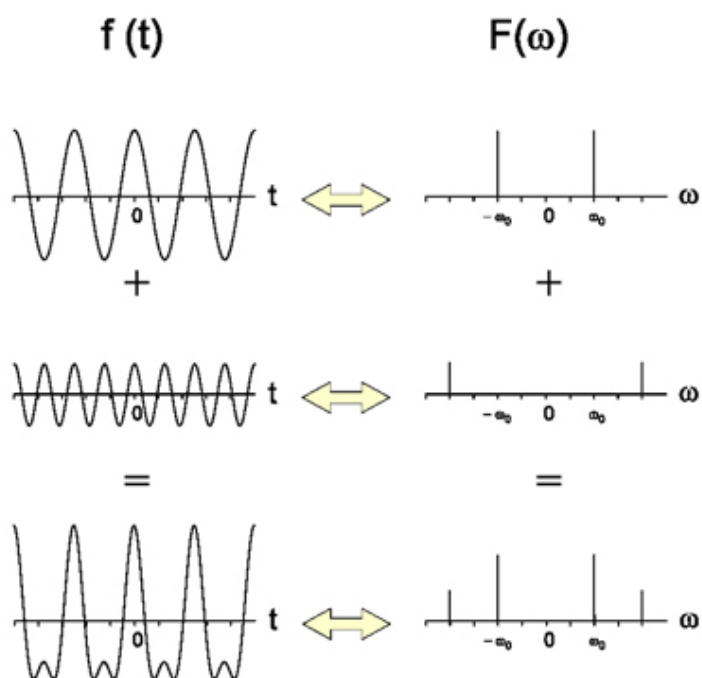
Dvojice ve Fourierově transformaci se nazývají originál (zde  $s(t)$ ) a obraz (zde  $S(\omega)$ ). Vztah mezi originálem a obrazem vyjadřujeme zápisem:

$$\begin{aligned} S(\omega) &= F[s(t)] \\ s(t) &= F^{-1}[S(\omega)] \end{aligned} \quad (2)$$

Spektrum  $S(\omega)$  je komplexní veličina a lze ji vyjádřit:

$$S(\omega) = |S(\omega)| e^{j \arg S(\omega)} \quad (3)$$

Fourierovu Transformaci můžeme rozdělit do čtyř základních typů podle signálu, se kterými se můžeme setkat. Signál může být buď spojitý nebo nespojitý a dále může být periodický nebo neperiodický. Kombinací těchto dvou vlastností dostáváme čtyři kategorie. Fourierova transformace neperiodického, spojitého signálu (například slábnoucí exponenciální signál nebo Gaussova křivka) je nazývána jako Fourierova Transformace. Fourierova transformace periodického, spojitého signálu (například sinusoidy, obdélníkové průběhy a jiné průběhy, které jsou periodické) je nazývána jako Fourierova Řada. Fourierova transformace neperiodického, nespojitého signálu (signály, které jsou definované pouze jako body a periodicky se neopakují) je Diskrétní Časová Fourierova Transformace. Fourierova transformace periodického, nespojitého signálu (signály, které se periodicky opakují, ale netvoří je spojitý signál) je Diskrétní Fourierova Transformace.



Obr. 30: Příklady fourierovy transformace

Pro Fourierovu transformaci v diskretním čase platí:

$$S(\Omega) = \sum_{k=-\infty}^{\infty} s(k) e^{-j\%omegak} \quad (4)$$

Někteří autoři označují tuto transformaci DtFT (discrete-time Fourier transformation), aby ji odlišili od Fourierovy transformace spojitého signálu. Spektrum diskretního signálu se od spektra spojitého signálu liší tím, že je periodické s periodou  $2\pi$ .

Pro spektrum reálného signálu platí, že amplitudové spektrum je sudou funkcí, fázové spektrum je lichou funkcí, spektrum sudého signálu je sudou reálnou funkcí a spektrum lichého signálu je lichou ryze imaginární funkcí.

### ***Diskrétní fourierova transformace***

Definiční vztahy Fourierovy transformace vyžadují znalost matematického vyjádření signálu či spektra. Pokud však zpracováváme naměřené hodnoty, tedy máme vzorky signálu či spektra z konečného intervalu, stojíme před problémem, jak určit spektrum z vzorků signálu či signál ze vzorků spektra. K tomu účelu používáme numerické metody, která je známa jako diskretní Fourierova transformace (DFT), která je popsána následujícím vzorcem:

$$D(n) = \sum_{k=0}^{N-1} d(k) e^{-jnk2\pi/N}, n=0, \dots, N-1 \quad (5)$$

Diskrétní Fourierova transformace našla velké uplatnění zejména s rozvojem výpočetní techniky. Součástí řady přístrojů jsou jednoúčelové procesory realizující tuto transformaci. Výpočet DFT podle definičního vztahu vyžaduje  $N^2$  komplexních součinů a  $N^2$  komplexních součtů. Toto množství operací výrazně snižovalo možnost aplikace DFT na výpočty v reálném čase. Situace se změnila po roce 1965, kdy J.W. Cooley a J.W. Tukey popsali velmi efektivní algoritmus výpočtu DFT, tzv. rychlou Fourierovu transformaci (FFT - Fast Fourier Transform), který vyžaduje jen  $N / 2 \log_2(N)$  komplexních součinů a  $N \log_2(N)$  komplexních součtů. Díky tomuto algoritmu se stala diskrétní Fourierova transformace nejrozšířenějším prostředkem pro numerický výpočet Fourierovy transformace.

### ***Rychlá fourierova transformace***

FFT je efektivní algoritmus pro spočtení diskrétní Fourierovy transformace (DFT) a její inverze. FFT je velmi důležitá v mnoha oblastech, od digitálního zpracování signálu a řešení parciálních diferenciálních rovnic až po rychlé násobení velkých celých čísel. Obecně jsou FFT algoritmy založeny na faktorizaci  $N$ , nicméně existují i FFT algoritmy se složitostí  $O(N \log N)$  pro všechna  $N$ , tedy i pro prvočísla. Nejpoužívanějším algoritmem je Cooley-Tukey algoritmus. Je příkladem rozděl a panuj algoritmu, který rekurzivně dělí DFT s velikostí složeného čísla do menších DFT o velikostech  $N_1$  a  $N_2$ . Tato metoda (a obecně idea FFT) byla popularizována v práci pánů J. W. Cooley a J. W. Tukey z roku 1965, nicméně později se přišlo na to, že tito autoři pouze znovuobjevili algoritmus známý již Gaussovi kolem roku 1805 (který byl poté v omezené podobě několikrát znovu objeven). Nejpoužívanější podobou Cooley-Tukey algoritmu je dělení transformace v každém kroku na dva kusy velikosti  $N / 2$  (čímž je omezena na velikosti mocniny dvojky), nicméně je možné použít kteroukoli faktorizaci (čehož si byli vědomi jak Gauss, tak i Cooley a Tukey). Přestože je idea rekurzivní, většina tradičních implementací algoritmus modifikují, aby se vyhnuli explicitní rekurzi. Vzhledem k tomu, že Cooley-Tukey algoritmus dělí DFT do několika menších DFT, je možné zkombinovat tento algoritmus s kterýmkoli jiným DFT. Jelikož je inverzní DFT stejná jako DFT jen s rozdílem opačného znaménka v exponentu a koeficientu  $1/N$ , kterýkoli algoritmus je možné snadno modifikovat i pro počítání inverzní DFT.

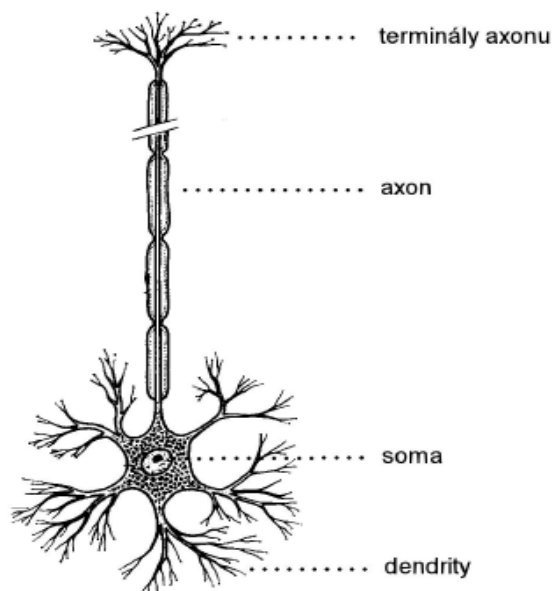
### *Inverzní fourierova transformace*

Slouží k převodu signálu z frekvenční oblasti zpátky do časové a lze ji popsat následujícím vzorcem:

$$f(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{f}(\omega) e^{j\omega t} d\omega \quad (6)$$

#### 2.4.2 Neuronové sítě

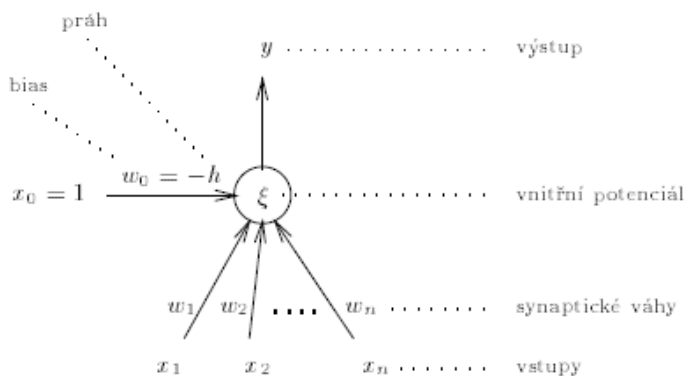
Vznikly inspirací fungování nervové soustavy u živých organismů. Využívá se jejich schopnosti řešit silně nelineární úlohy, schopnosti extrahovat a reprezentovat závislosti v datech, které nejsou na první pohled zřejmé, schopnosti učit se a schopnosti zevšeobecnovat stejně jako například lidský mozek. Za počátek vzniku oboru neuronových sítí lze považovat práci Warren McCullocha a Walter Pittse z roku 1943, kteří vytvořili velmi jednoduchý matematický model neuronu. Ukázali, že i nejjednodušší neuronové sítě mohou v principu počítat libovolnou logickou nebo aritmetickou funkci. Postupně probíhal další vývoj, kdy ale většina badatelů přistupovala k neuronovým sítím pouze z experimentální hlediska. Počátkem 80. let minulého století došlo k osmělení a podávání grantů na konstrukci neuropočítačů.



Obr. 31: Biologický neuron

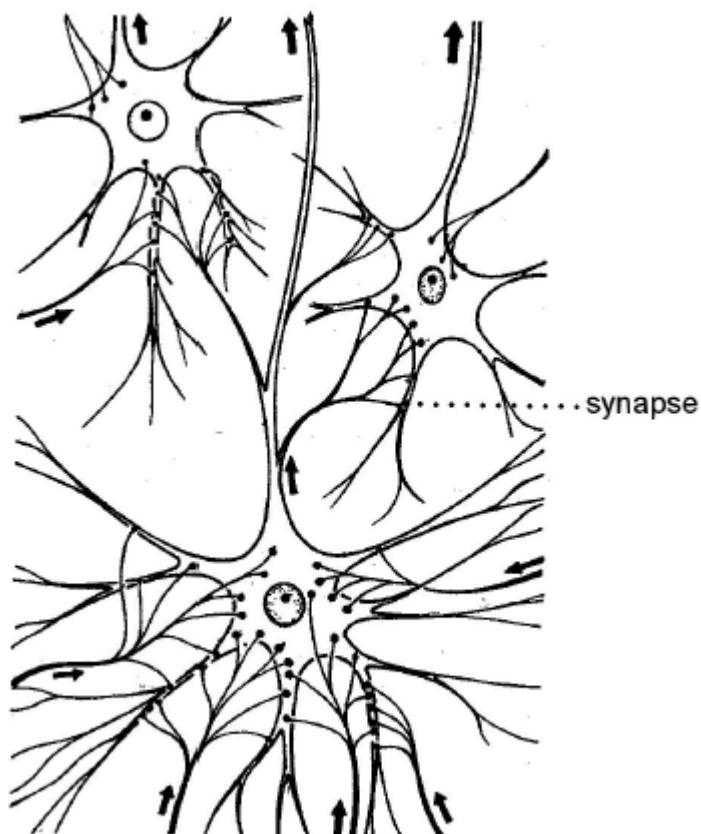
Neuron je základní stavební jednotkou nervové soustavy. Jen mozková kůra člověka je tvořena asi 13 až 15 miliardami mozkových buněk, neuronů, z nichž každý může být spojený s 5000 dalšími neurony. Neurony jsou samostatné buňky specializované k přenosu,

zpracování a uchovávání informací nutných pro realizaci životních funkcí organismu. Základem matematického modelu neuronové sítě je formální neuron, který získáme přeformulováním funkce biologického neuronu do matematické řeči. Jeho struktura je zobrazena na obrázku.



Obr. 32: Formální neuron

Neuronová síť se skládá z formálních neuronů, které jsou propojeny tak, že výstup z neuronu je vstupem několika dalších neuronů. Počet neuronů a jejich další propojení určuje topologii (architekturu) neuronové sítě.

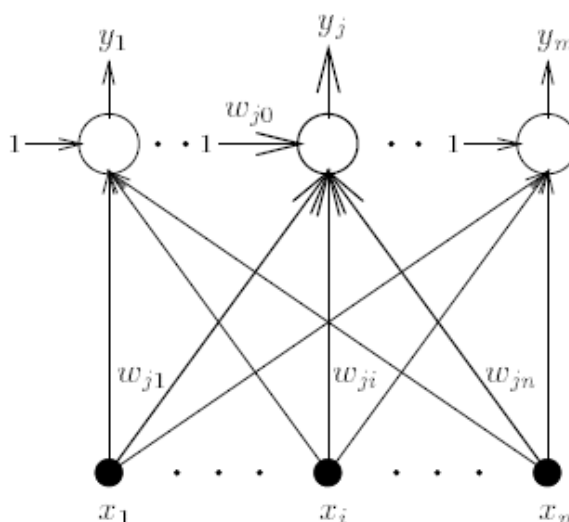


Obr. 33: Biologická neuronová síť

Neuronová síť se v čase vyvíjí, to znamená že se mění propojení a stav neuronů a adaptují se váhy. Dynamiku neuronové sítě lze rozdělit do třech částí (třech režimů práce): organizační (změna topologie), aktivní (změna stavu) a adaptivní (změna konfigurace). Odlišná architektura neuronových sítí vyžaduje speciální hardwarovou realizaci. Jedná se o tzv. neuropočítače. Nejčastější implementací ale stále ještě bývá netware, což je realizace neuronové sítě pomocí softwarových prvků na hardware klasického PC.

### *Klasické modely neuronových sítí*

Historicky první úspěšnou sítí byla síť perceptronů, jedná se o  $n$  vstupních neuronů z nichž je každý vstupem  $m$  výstupních neuronů. Schéma sítě je naznačeno na obrázku.



Obr. 34: Síť perceptronů

Nejznámější a nepoužívanější model sítě je vícevrstvá neuronová síť s učícím algoritmem zpětného šíření chyby (backpropagation), který je používám v až 80 % případů využití neuronové sítě.

Dalším historicky důležitým modelem je síť „MADALINE“ (Multiple ADALINE). Model byl navržen Widrowem a Hoffem. Základním prvkem je neuron „ADALINE“ (ADAPtive LINear Elemant) perceptronu. Proto je i model sítě podobný síti perceptronů.

Sítě s kaskádovou architekturou byly navrženy Fahlmanem a Labierem na přelomu 80. a 90. let minulého století a daly by se označit za zobecnění sítě perceptronů.

Mimo klasických modelů neuronových sítí existuje nepřeberné množství dalších modelů, o který se dá dočíst v odborných publikacích zabývající se neuronovými sítěmi.

### 2.4.3 Cepstrum

Cepstrum je výsledek Fourierovy transformace z logaritmu spektra signálu. Název byl odvozený přehozením prvních čtyř písmen slova spectrum (anglický výraz pro spektrum). Existuje reálné cepstrum, komplexní cepstrum, výkonové cepstrum a fázové cepstrum. Existuje spousta algoritmů pro jeho výpočet. Výkonové cepstrum bylo poprvé popsáno B. P. Bogertem, M. J. R. Healyem, and J. W. Tukeyem. Matematicky by šlo popsat následující rovnicí:

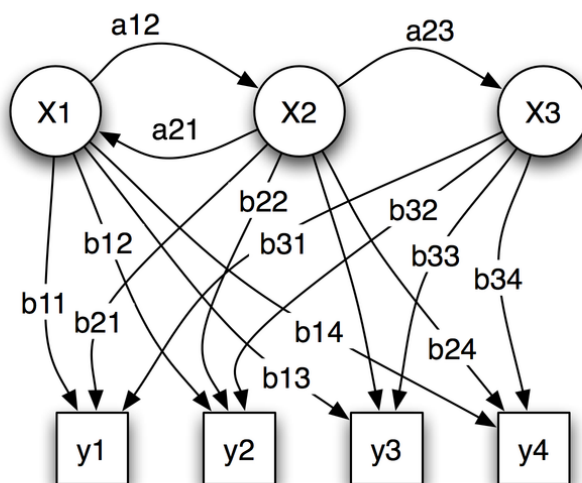
$$\text{power cepstrum of signal} = |F\{\log(|F\{f(t)\}|^2)\}|^2 \quad (7)$$

### 2.4.4 Technologie rozpoznávání řeči

Některé z technologií rozpoznávání řeči jsou použity v pracích zabývajících se odposlechem klávesnice analýzou zvukového signálu.

#### *Skryté Markovské modely*

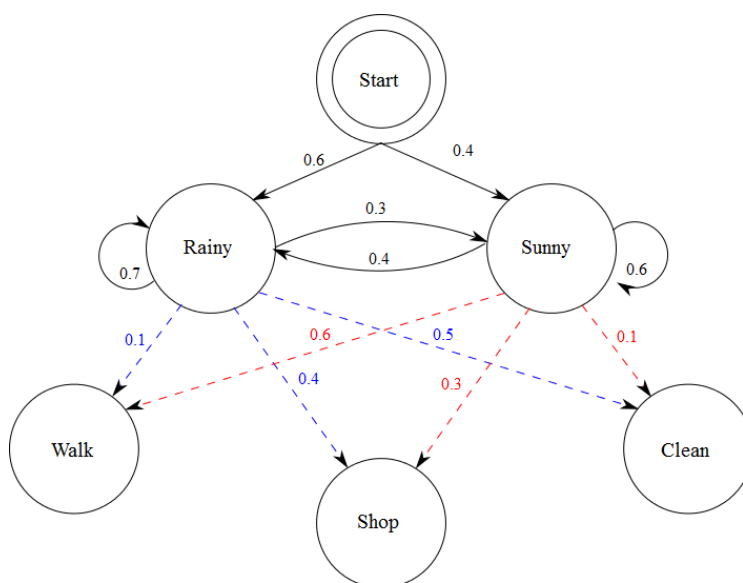
Skrytý Markovský model je statistický Markovský model, kterého lze využít při například při rozpoznávání řeči nebo psaného textu. Popsat by se dal následujícím obrázkem:



Obr. 35: Diagram HMM (x - stavy, y - možné výsledky, a - pravděpodobnosti přechodu mezi stavy, b - pravděpodobnosti výstupů)

### Viterbiho algoritmus

Viterbiho algoritmus je dynamický programový algoritmus pro nalezení tzv. Viterbiho cesty. Jedná se o nejpravděpodobnější sekvenci určitých událostí. Viterbiho algoritmus byl objevený v roce 1967 Andrewem Viterbim pro dekódování konvolučních kódů. Fungování algoritmu si můžeme ukázat na příkladu. Alice se potká s Bobem a zjistí, že první den šel na procházku, druhý den šel nakupovat a třetí den si uklidil byt. Jaká je nejpravděpodobnější posloupnost deštivých a slunečných dní?



Obr. 36: Schéma příkladu pro hledání Viterbiho cesty

Příklad realizace algoritmu pro programovací jazyk C:

```

# Helps visualize the steps of Viterbi.
def print_dptable(V):
    print "    ",
    for i in range(len(V)): print "%7s" % ("%d" % i),
    print
    for y in V[0].keys():
        print "%.5s: " % y,
        for t in range(len(V)):
            print "%7s" % ("%f" % V[t][y]),
        print
def viterbi(obs, states, start_p, trans_p, emit_p):
    V = [{}]
    path = {}
    # Initialize base cases (t == 0)

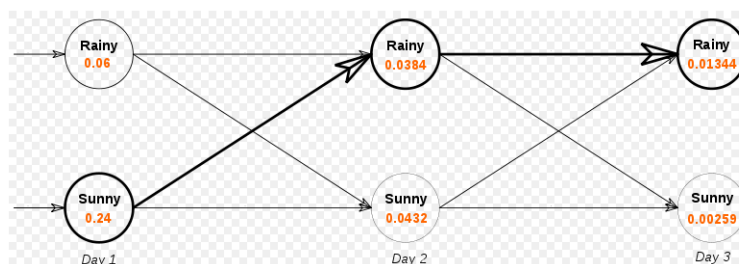
```

```

for y in states:
    V[0][y] = start_p[y] * emit_p[y][obs[0]]
    path[y] = [y]
# Run Viterbi for t > 0
for t in range(1, len(obs)):
    V.append({})
    newpath = {}
    for y in states:
        (prob, state) = max([(V[t-1][y0] * trans_p[y0][y] *
emit_p[y][obs[t]], y0) for y0 in states])
        V[t][y] = prob
        newpath[y] = path[state] + [y]
    # Don't need to remember the old paths
    path = newpath
print_dptable(V)
(prob, state) = max([(V[len(obs) - 1][y], y) for y in
states])
return (prob, path[state])
def example():
    return viterbi(observations,
                    states,
                    start_probability,
                    transition_probability,
                    emission_probability)
print example()

```

Výstup z algoritmu nám určí pořadí „slunečno“, „deštivo“, „deštivo“. Výsledek lze i zobrazit v mřížkovém diagramu.



Obr. 37: Mřížkový diagram příkladu pro hledání Viterbiho cesty

### ***Dynamické borcení času***

Jedná se o algoritmus, který zjištění podobnosti dvou různých sekvencí dat, které se liší rychlostí. Například při analyzování dvou videí, kde se jedna osoba pohybuje rychleji a druhá pomaleji. Příklad zdrojové kódu pro DWT pro jazyk C.

```

int DTWDistance(char s[1..n], char t[1..m]) {
    declare int DTW[0..n, 0..m]
    declare int i, j, cost

    for i := 1 to m
        DTW[0, i] := infinity
    for i := 1 to n
        DTW[i, 0] := infinity
    DTW[0, 0] := 0

    for i := 1 to n
        for j := 1 to m
            cost:= d(s[i], t[j])
            DTW[i, j] := cost + minimum(DTW[i-1, j ], //
insertion                                     DTW[i , j-1], //
deletion                                     DTW[i-1, j-1]) //
match

        return DTW[n, m]
    }

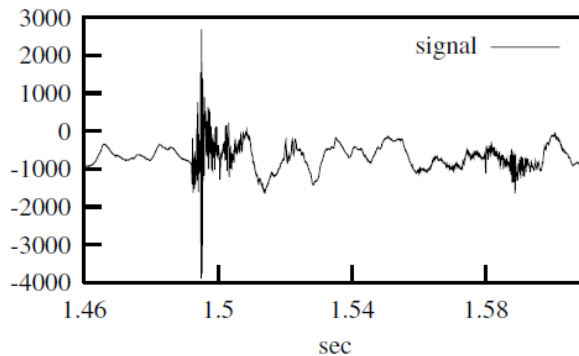
```

#### 2.4.5 Rozbor prací na téma keyloggingu analýzou zvukového signálu

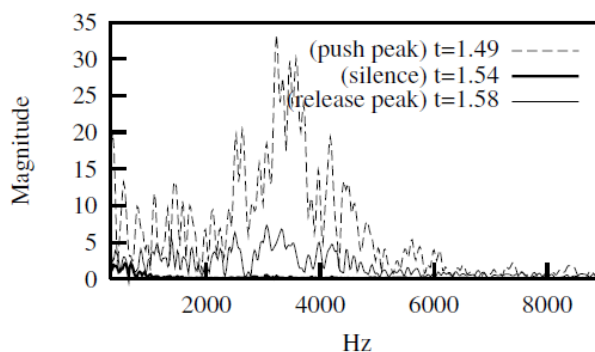
##### *Práce autorů Asonova a Agrawala*

Práce se jmenuje Keyboard Acoustic Emanations. V práci se zaměřili na problém odposlechu klávesnice (jak počítačové, tak i mobilního telefonu) pomocí analýzy zvukového signálu. Vycházeli z předpokladu, že přestože klávesy lidskému uchu nezní příliš odlišně, tak je možné je rozlišit analýzou zvukového signálu. Jako zařízení použili několik počítačových klávesnic IBM, tři počítačové klávesnice GE Power keyboards HO97798 a jednu klávesnici mobilního telefonu Siemens MR 240. Pro odposlech do 1 metru použili obyčejný mikrofon, pro vzdálenější odposlech použili parabolický mikrofon. Pro vstup byla použita standardní zvuková karta se vzorkovací frekvencí 44.1 kHz. Jako software pro záznam zvuku byl vybrán software SigView. V programu byla taktéž realizována FFT v 2 ms oknech. Pro rozpoznávání byl použit simulátor neuronové sítě JavaNNS. Jedná se o simulátor neuronové sítě s učícím algoritmem zpětného šíření chyby. Počet uzlů záležel na počtu vzorků. Například pro hodnotu FFT pro každých 20 Hz je potřeba 200 vstupních uzlů pro frekvenční pásmo 0 až 4 kHz. Bylo použito 6 až 10

skrytých uzlů podle počtu vzorků a rozpoznávaných kláves. Počet výstupních uzlů odpovídal počtu kláves v případě experimentu s více klávesami. V případě pokusu se dvěma klávesami stačil jeden uzel.

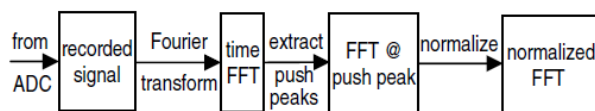


Obr. 38: Asonov, Agrawal, průběh signálu pro stisk klávesy



Obr. 39: Asonov, Agrawal, spektrum signálu pro stisk klávesy

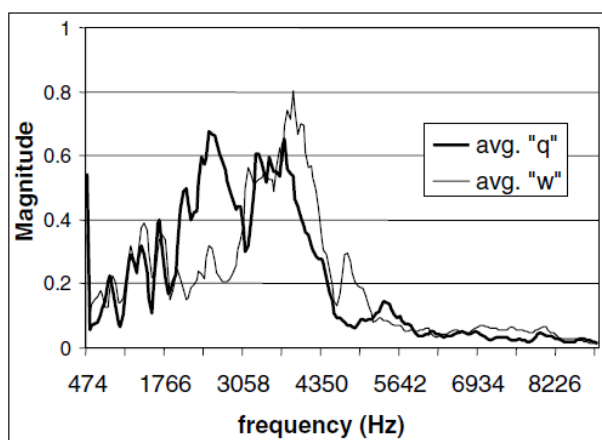
Jelikož surová zvuková data nejsou příliš vhodná pro analýzu pomocí neuronové sítě (doporučuje se maximální velikost do 1kB a hodnoty sestávající jenom z hodnot 0 a 1), provedli další operace se signálem.



Obr. 40: Asonov, Agrawal, postup zpracování signálu

Dále využili zjištění, že mezi stiskem a uvolnění klávesy proběhne asi 100 ms. Jako frekvenční pásmo použili frekvenční rozsah od 300 Hz do 3400 Hz. Jedná se o pásmo, ve kterém probíhají telefonní hovory a postačí pro porozumění mluvenému slovu. Výhodou je, že pokud dokážeme spolehlivě rozpoznávat stisknuté klávesy, lze využít i útoku přes

telefon, kdy bylo možné přenášet odposlouchávaný signál i na velmi velkou vzdálenost pomocí telefonní sítě a tímto by se stavál potenciální útok mnohem nebezpečnější.



Obr. 41: Asonov, Agrawal, příklad zpracovaných spekter dvou různých kláves (q a w)

Nejdříve otestovali s úspěchem rozpoznávání mezi dvěmi klávesami. Dosáhli přesnosti průměrně kolem 0,5 chybného určení na každých 20 stisknutí. Při testování použitelné vzdálenosti použili parabolický mikrofon a zjistil, že do nějakých 15 metrů kvalita rozpoznávání neutrpí. Následně zkusili rozpoznat 30 různých kláves. Neuronová síť měla proto 30 uzlů. Výsledky jsou shrnuty na následujícím obrázku.

Keyboard A, ADCS: 1.99						
key pressed	q	w	e	r	t	y
recognized	9,0,0	9,1,0	1,1,1	8,1,0	10,0,0	7,1,0
key pressed	u	i	o	p	a	s
recognized	7,0,2	8,1,0	4,4,1	9,1,0	6,0,0	9,0,0
key pressed	d	f	g	h	j	k
recognized	8,1,0	2,1,1	9,1,0	8,1,0	8,0,0	8,0,0
key pressed	l	;	z	x	c	v
recognized	9,1,0	10,0,0	9,1,0	10,0,0	10,0,0	9,0,1
key pressed	b	n	m	,	.	/
recognized	10,0,0	9,1,0	9,1,0	6,1,0	8,1,0	8,1,0

Obr. 42: Asonov, Agrawal, výsledky rozpoznávání 30 různých kláves na klávesnici A

Výsledky byly potěšující, 79 % stisků kláves bylo rozpoznáno správně. Následně zkusili dvě různé klávesnice, na které nebyla neuronová síť naučena. Výsledky nebyly už tak příznivé, jak je vidět z následujících obrázků.

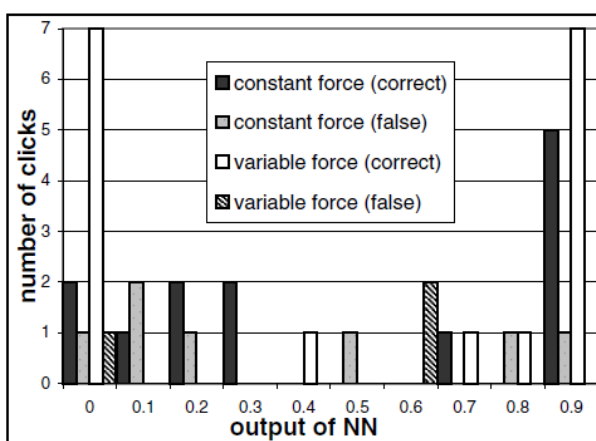
Keyboard B, ADCS: 9.24						
key pressed	q	w	e	r	t	y
recognized	6,1,1	4,1,1	0,1,0	0,2,1	5,1,1	1,0,0
key pressed	u	i	o	p	a	s
recognized	1,2,1	4,1,1	4,3,1	4,1,1	4,1,0	2,1,0
key pressed	d	f	g	h	j	k
recognized	1,4,0	0,0,0	1,0,1	5,1,1	9,0,0	1,0,2
key pressed	l	;	z	x	c	v
recognized	5,0,1	3,2,0	1,0,2	0,0,0	2,0,0	0,2,2
key pressed	b	n	m	,	.	/
recognized	3,3,1	3,1,1	5,1,1	0,2,1	2,1,0	7,2,1

Obr. 43: Asonov, Agrawal, výsledky rozpoznávání  
30 různých kláves na klávesnici B

Keyboard C, ADCS: 9.10						
key pressed	q	w	e	r	t	y
recognized	1,1,3	0,0,1	0,0,1	4,3,1	0,0,0	0,0,0
key pressed	u	i	o	p	a	s
recognized	2,3,0	1,3,0	3,3,3	1,1,1	0,1,0	1,2,0
key pressed	d	f	g	h	j	k
recognized	2,0,1	0,1,0	2,0,4	2,4,1	0,3,1	3,1,0
key pressed	l	;	z	x	c	v
recognized	1,0,0	1,1,0	2,2,0	0,1,1	10,0,0	1,0,2
key pressed	b	n	m	,	.	/
recognized	7,1,1	7,1,1	5,0,2	1,1,3	4,1,0	2,1,1

Obr. 44: Asonov, Agrawal, výsledky rozpoznávání  
30 různých kláves na klávesnici C

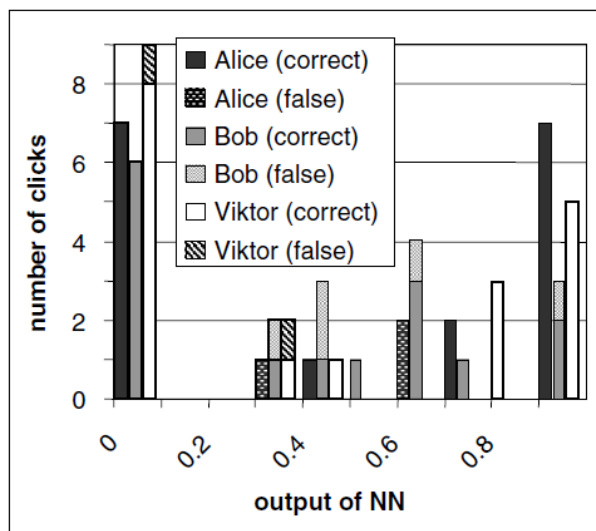
Dále provedli experimentu k ověření vlivu síly stisku s konstatováním, že síla stisku klávesy ovlivňuje kvalitu rozpoznávání značně, jak je i vidět z přiloženého obrázku.



Obr. 45: Asonov, Agrawal, vliv síly stisku na  
kvalitu rozpoznávání

Poslední pokusem bylo ověření, zda má vliv způsob psaní. Otestovali tedy rozpoznávání při používání stejné klávesnice třemi různými osobami. Narozdíl od vlivu síly stisku

konstatovali, že vliv různých stylů psaní není tak značný, což činí tuto metodu vhodnou i pro praktické využití, kdy jednou naučená neuronová síť se dá využít pro odposlech různých uživatelů. Výsledek je vidět i na obrázku.

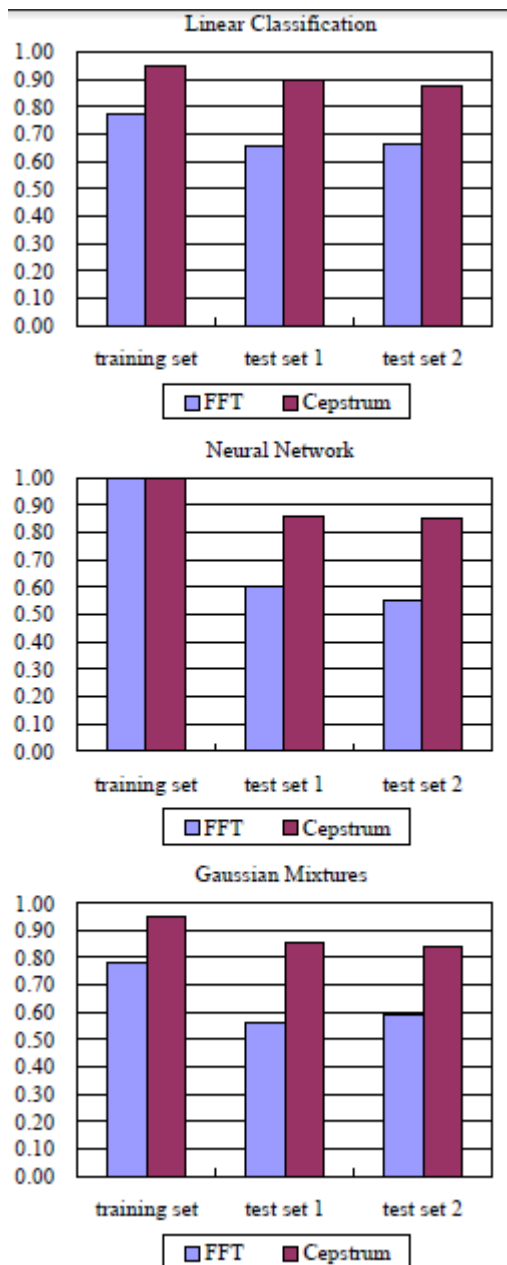


Obr. 46: Asonov, Agrawal, vliv stylu psaní (osoby)  
na kvalitu rozpoznávání

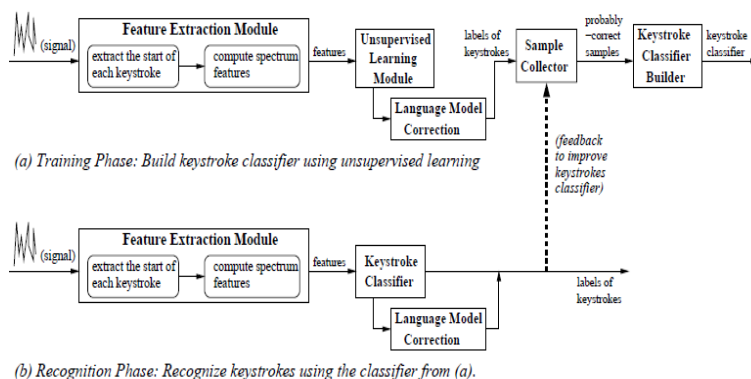
Dále v práci rozebrali důvody, proč různé klávesy vydávají různý zvuk a je možno je odposlouchávat. Jako nejpravděpodobnější se po sérii experimentů ukázala hypotéza, že různý zvuk vydávají klávesy v momentě úderů na desku kontaktů tím, že na ni dopadají v různém místě. Závěrem provedli testy zaměřené na klávesnice mobilních telefonů a bankomatů.

### ***Práce autorů Zhuaga, Zhoua a Tygara***

Vychází z práce Asonova a Agrawala. Jedná se o práci s názvem Keyboard Acoustic Emanations Revisited. Doplnili ale další metody analýzy signálu a přepracovanou detekci stisknuté klávesy. Dalším rozdílem bylo, že analyzovali 10 minut anglicky psané textu na rozdíl od pouze detekce jednotlivých kláves. Pro zlepšení rozpoznávání si ale pomohli anglickým slovníkem, který předpokládal, že uživatel píše srozumitelný text a některé znaky tedy opravovali slovníkem, aby text dával smysl. Použili také zpětnou vazbu pro další zlepšení úspěšnosti odposlouchávání. Pro analýzu použili lineární klasifikaci, neuronové sítě a Gaussovo mixování. Pro detekci stisknuté klávesy použili buďto FFT jako autoři Asonov a Agrawal nebo Cepstrum se kterým dosáhli lepších výsledků. Porovnání jejich výsledků jednotlivých metod je vidět z následujících obrázků.

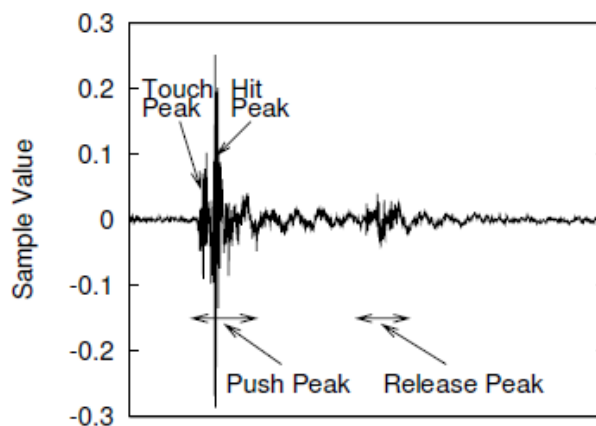


Obr. 47: Zhuang, Zhou, Tygar, výsledky jednotlivých metod



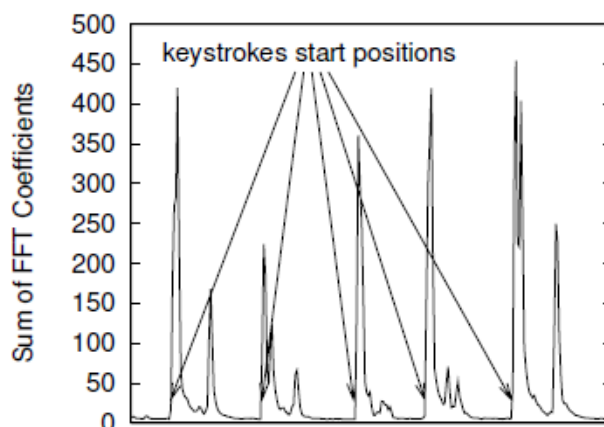
Obr. 48: Zhuang, Zhou, Tygar, schéma

Jelikož pokročilý uživatel může psát i kolem 300 úhozů za minutu předělali systém na detekci stisknuté klávesy.



Obr. 49: Zhuang, Zhou, Tygar, průběh signálu stisku klávesy

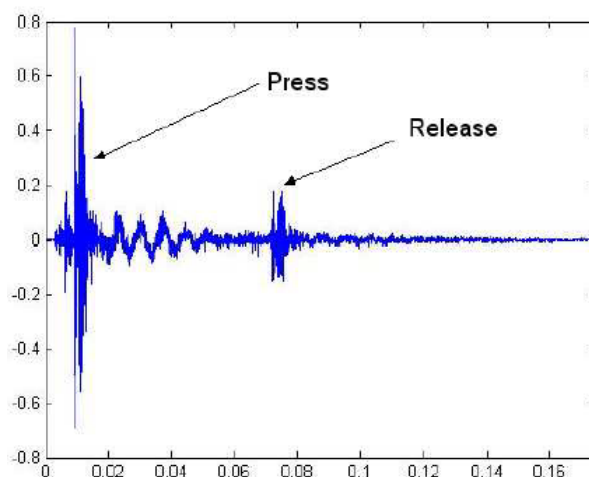
Hlavním rozdílem mezi délkou stisknuté klávesy a tichem je v úrovni energie přenášené určitými frekvencemi. Zjistili, že hlavní energie přenášená stiskem klávesy je ve frekvenčním pásmu 400 až 12000 Hz. Programem Matlab tedy provedli DFT signálu použitím plovoucího okna. Použili amplitudové spektrum. Nasčítali amplitudu spektra ve frekvenčním pásmu od 0,4 kHz do 12 kHz. Následně nastavili komparační úroveň, která detekovala stisk klávesy.



Obr. 50: Zhuang, Zhou, Tygar, výstupní energie prvních pěti stisků klávesy

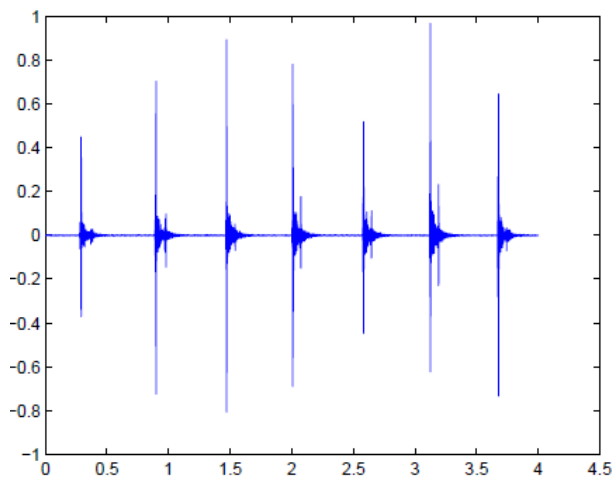
### ***Práce autorů Haleviho a Saxeny***

Práce nazvaná Eavesdropping over Random Passwords via Keyboard Acoustic Emanations vycházela z obou prací již zde zmíněných. Zabývala se taktéž rozpoznáváním psaném textu na rozdíl od rozpoznávání jednotlivých kláves.

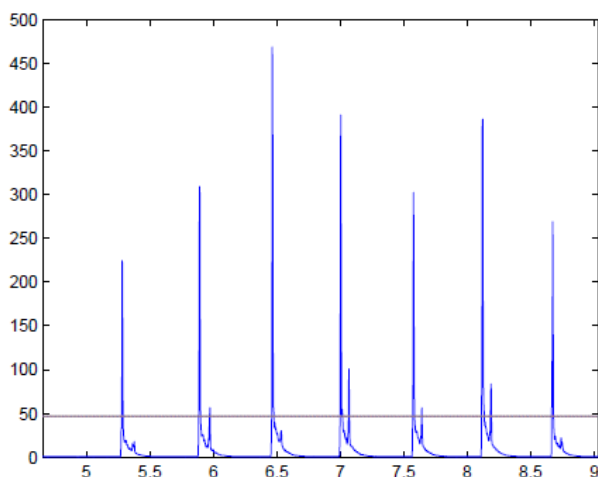


Obr. 51: Halevi, Saxena, průběh signálu

Pro nahrávání signálu použili vzorkovací frekvenci 44.1 kHz. Pro detekci začátku stisku klávesy použili FFT signálu s velikostí okna o 440 vzorcích. Dále sečetli hodnoty koeficientů FFT pro frekvenční pásmo od 0,4 kHz do 22 kHz. Jakmile tato hodnota dosáhla určité úrovně detekovali stisk klávesy. Pro detekci uvolnění klávesy použili analýzu signálu po 50 ms od detekce stisku. Pro určení okamžiku použili FFT z 88 vzorků a jejich následné sečtení.



Obr. 52: Halevi, Saxena, průběh signálu pro stisk několika kláves za sebou

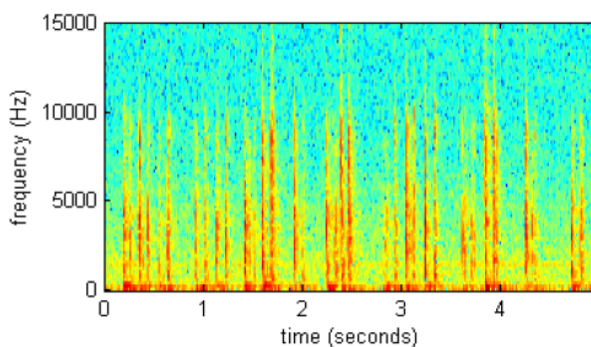


Obr. 53: Halevi, Saxena, součet koeficientů FFT pro stisk několika kláves za sebou

Pro analýzu použili některé další metody, například DTW.

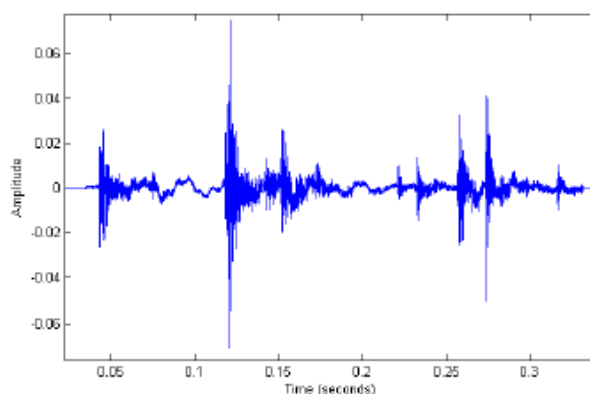
### ***Práce autora Kellyho***

Práce nazvaná Cracking Passwords using Keyboard Acoustics and Language Modeling se zabývala hlavně rozpoznáváním hesel a vycházela z prací Asonova a Agrawala a Zhouanga, Zhoua a Tygara. Autor uvádí, že dokáže asi 50 krát snížit dobu potřebnou ke zjištění uživatelského hesla. Pro zaznamenávání zvuku použil vysokou vzorkovací frekvenci 96 kHz a 32-bitovou přenos. Potvrdil, že většina energie je v pásmu od 0,4 do 12 kHz.



Obr. 54: Kelly, spectrogram signálu stisku několika kláves

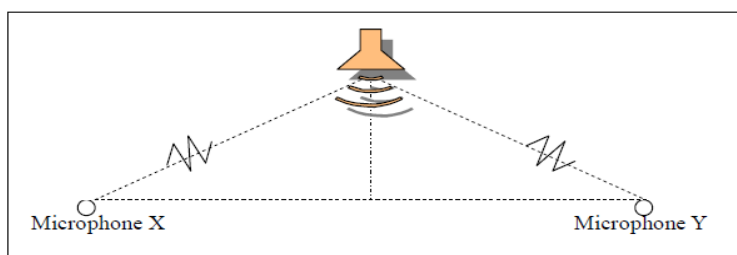
Zabýval se také vlivem šumu na pozadí a také vlivu toho, že stisknuté klávesy se mohou v průběhu signálu částečně překrývat. I tak ale dosáhl úspěšnosti přes 80 %.



Obr. 55: Kelly, průběh signálu pro překrývající se stisk kláves

### *Práce autora Paka*

Práce nazvaná Keyboard Acoustic Triangulation Attack se jako jediná ze zmíněných prací zabývá nahráváním signálu pomocí dvou mikrofonů na rozdíl od jednoho.



Obr. 56: Pak, schéma metody

Práce vychází z prací Asonova a Agrawala a Zhuanga, Zhoua a Tygara a měla by kombinovat jejich výhody. Nasazení odposlouchávání podle jejich metody by mělo být levné, jednoduché, přesné a mělo by jít aplikovat do praxe.

## 2.5 Další metody keyloggingu

Jedná se například o optické sledování. Při optickém sledování se jedná o vhodně umístěnou kameru, která snímá jak uživatel zadává data. Následně tyto data analyzuje osoba u záznamu, případně stroj. Nejčastěji se využívá pro získání PINu u bankomatů. Tento keylogging nicméně naráží na několik obtíží. Kameru je potřeba dobře schovat, zároveň musí ale dobře vidět na snímané místo, kameru bude často instalována ve veřejných prostorech, musí mít dostatečné rozlišení a musí být schopna pracovat i v horších světelných podmínkách.

## 2.6 Ochrana proti odposlechu

### 2.6.1 Ochrana proti softwarovým keyloggerům

Proti softwarovým keyloggerům je možné se bránit tím, že nebude mít žádná další osoba přístup k našemu počítači, tudíž nebude možné nainstalovat jakýkoliv software. V případě, že počítač používá více osob, právo instalovat aplikace bude mít pouze důvěryhodný správce, který instalaci škodlivého softwaru neprovede. Dále je určité vhodné mít nainstalovaný kvalitní antivirový software s aktuální virovou databází (tato podmínka je jedna z nejdůležitější, jelikož nové virové hrozby vznikají prakticky neustále) a případně další programy k odstraňování dalšího škodlivého softwaru (trojské koně, spyware, malware, ...). Taktéž je možné použít programy, které by měly detekovat softwarového keyloggery nebo jim alespoň zabránit v činnosti. Nicméně test provedený Martinem Dřímalem pro server LUPA.CZ nepotvrdil funkčnost těchto programů.

Název AntiKeyloggeru	GetAsyncKey State	GetKeyboard State	DirectX	PrintScreen
AntiKeylogger 8.2	Detekoval	Detekoval	Nedetekoval	Nedetekoval
Snoop Free	Detekoval	Nedetekoval	Nedetekoval	Detekoval
ID AntiKeylogger	Detekoval	Nedetekoval	Nedetekoval	Nedetekoval
1-ACT AntiKeylogger 2006	Detekoval	Nedetekoval	Nedetekoval	Nedetekoval
AntiKeylogger 1.1	Detekoval	Nedetekoval	Nedetekoval	Nedetekoval
I Hate KeyLoggers	Nedetekoval	Nedetekoval	Nedetekoval	Nedetekoval
AntiKeylogg 1.0	Nedetekoval	Nedetekoval	Nedetekoval	Nedetekoval

Obr. 57: Výsledky testů antikeyloggerů

### 2.6.2 Ochrana proti hardwarovým keyloggerům

Proti hardwarovým keyloggerům je možné se bránit zamezením přístupu jakékoliv záškodnické osoby k našemu počítači, čímž jí znemožníme instalaci jakéhokoliv zařízení. Dále je možné se bránit proti některým metodám změnou konstrukce klávesnice, případně jejím stíněním. Nicméně nejspolehlivější obranou proti hardwarovému keyloggeru je instalace a používání softwarové klávesnice pro zadávání velmi citlivých dat (hesla do systémů, přístupové údaje do internetového bankovníctví, ...).

### 2.6.3 Ochrana proti ostatním keyloggerům

Proti optickému snímání se dá bránit opět zamezením přístupu cizí osoby, který znemožní instalaci kamery a pak samozřejmě zakrytím klávesnice při zadávání údajů tak, aby měla osoba sledující záznam ztíženou případně úplně znemožněnou činnost rozpoznávání

zadávaných znaků.

### 3 PROGRAM PRO SBĚR A ANALÝZU DAT

Pro realizaci programu pro sběr a analýzu dat jsem si vybral programovací jazyk C#, jelikož se jedná o moderní programovací jazyk, je dostupné kvalitní vývojové prostředí od firmy Microsoft včetně dokumentace a není potřebné se starat složitě o grafický návrh aplikace. Jazyk vychází z jazyka C a C++, nicméně na rozdíl od nich je plně objektový. Byl navržen společností Microsoft a je nativním jazykem platformy .NET Common Language Runtime. Program bude využívat .NET Framework verze 3.5, který je tedy nutné mít nainstalovaný na počítači, kde chci software používat.

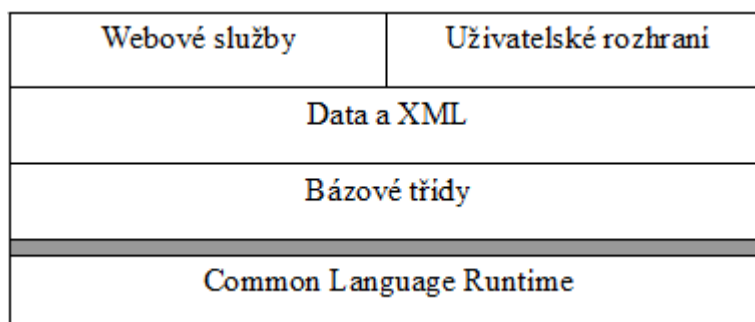
#### 3.1 Základní schéma programu

Mnou navržený a realizovaný program se skládá ze tří oddělených částí. Dvou samostatných programů a jedné podpůrné knihovny. Podpůrná knihovna se stará o vytvoření základní třídy pro záznam zvuku a zápis do souboru. Jelikož prostředí .NET Runtime neobsahuje žádné standardní třídy pro nahrávání zvukového signálu, bude použito rozhraní DirectSound od firmy Microsoft. Rozhraní DirectSound je součástí rozhraní DirectX a jedná se o interface mezi aplikací a zvukovou kartou v počítači. První program se bude starat o sběr dat z aktuální klávesnice a jejich ukládání do standardních „wav“ souborů. Tyto soubory je následně možné editovat či přehrávat jakýmkoliv klasickým zvukovým editorem. Uživatel bude mít možnost tyto data kdykoliv modifikovat, takže bude možné program naučit a použít na jinou klávesnici. Druhý program se bude starat o porovnávání načtených dat s aktuálně stisknutou klávesou z odposlouchávané klávesnice. Rozdělení do dvou programů jsem provedl, jelikož se jedná o dvě části, které je možné od sebe oddělit a není nutné je držet v jednom programu. Jelikož tento program se nebude spouštět na počítači, na který se bude útočit, ale na našem počítači, ničemu to nevádí. Navíc pokud bychom chtěli v budoucnu udělat kvalitnější rozpoznávání, není nutné měnit část, která se stará o sběr dat z klávesnice. V programu pro rozpoznávání je použita diskrétní fourierova transformace. Pro realizaci DFT bude použita volně dostupná knihovna FFTW. Pro porovnávání vzorků po Fourierově transformaci bude použita jednoduchá metoda sčítající odchylku jednotlivých vzorků.

#### 3.2 Prostředí .NET Runtime

Vytvářet moduly, které by bylo možné volat a používat v různých jazycích bylo dříve obtížné. Různé programovací jazyky využívají pro běh svých programů různé prostředí se

specifickým, tedy i odlišným chováním. Z tohoto důvodu byla vyvinuta technologie COM. Nicméně ani technologie COM nebyla úplně dobře přenositelná mezi různými programovacími jazyky. Za účelem nápravy této složité situace bylo navrženo prostředí .NET Runtime. Existuje zde jen jeden způsob popisu kódu (metadata) a jedno běhové prostředí a knihovna (Common Language Runtime and Frameworks). Způsob uspořádání .NET Runtime je znázorněn na následujícím diagramu:



Obr. 58: Uspořádání .NET Frameworks

Běhové prostředí poskytuje základní prováděcí služby. Co je ale běhové prostředí? Většina dnešních aplikací, jsou zkompileovány přímo pro danou platformu. To znamená, že zdrojový kód je kompilací převeden do strojového kódu počítače. To ve výsledku přináší velmi dobrou rychlost běhu výsledné aplikace. Avšak na druhou stranu z toho plynou i některé nevýhody – nepřenositelnost mezi jednotlivými platformami, popřípadě verzemi operačních systémů a nezdědky jsou k vidění chyby v přístupech do operační paměti. Princip řízených běhových prostředí, použitý právě u platformy .NET na to jde trochu jinak a přidává k převodu zdrojového kódu do kódu strojového ještě jednu vrstvu. Tuto vrstvu představuje mezikód, do kterého jsou zdrojové kódy zkompileovány, a tento mezikód je běhovým prostředím na cílové platformě převeden do strojového kódu. Tento převod je na cílové platformě realizován vždy při spuštění konkrétní aplikace. Mínusem tohoto překladu je vyšší náročnost na výkon uživatelského počítače, a z tohoto důvodu se tento způsob nepoužívá pro vývoj výpočetně náročných aplikací (např. počítačové hry). S jeho častým použitím se naopak můžete setkat spíše u obchodních aplikací, které přeci jen nejsou tak náročné na výpočetní výkon a rychlost běhu daných aplikací je naprosto vyhovující. Předchozí větu nicméně neznamená, že aplikace pod těmito platformami jsou nepoužitelně pomalé. U spousty úloh (přístup k databázi, souborům atd.) uživatel snížení rychlosti aplikace ani nepocítí. Navíc je dobré dodat, že u těchto běhových prostředí při spuštění aplikace nedochází k překladu celé aplikace najednou, ale používá se JIT (Just-in-Time) kompilace. JIT kompilace znamená, že do strojového kódu je převedena pouze

potřebná část mezikódu a při opětovném použití této (již přeložené) části se spouští její zkompileovaná forma, což se příznivě projeví na rychlosti, která si již nezadá s během neřízeného programu zkompileovaného přímo pro danou platformu. Nyní přejdeme ke klíčovým vlastnostem platformy Microsoft .NET, která tento způsob běhu aplikací přináší. Mezikód zmíněný o pár řádků výše se ve světě této platformy nazývá MSIL, tedy Microsoft Intermediate Language. Tento jazyk relativních adres je spouštěn klíčovou součástí .NET frameworku pojmenovanou CLR (Common Language Runtime neboli společné běhové prostředí) a firma Microsoft jej dala ke standardizaci organizaci ECMA. V prostředí CLR existuje věc, která programátorům velmi usnadňuje práci s operační pamětí – Garbage Collector. Jedná se o sadu složitých algoritmů pro uvolňování nepotřebných programových objektů z paměti. Díky Garbage Collectoru se již vývojáři nemusejí starat o přiřazování nebo uvolňování operační paměti a odpadá tak riziko již zmíněné nekorektní práce s ní, která ve většině situací končí pádem aplikace. Velmi důležitou vlastností, kterou dal Microsoft své platformě do vínku, je CLS – Common Language Specification (Společná jazyková specifikace). S ní souvisí CTS neboli Common Type System (společný typový systém). Výsledkem použití CLS a CTS je rovnocennost programovacích jazyků. Jinými slovy – pro vývoj .NET aplikací je možné použít jeden z několika programovacích jazyků vyšší úrovně. S tím souvisí i další výhoda této platformy – výrobcům třetích stran nic nebrání ve vývoji dalších jazyků. Jediné, co stačí, je, aby tento nový jazyk měl kompilátor se schopností kompilovat zdrojové kódy do mezijazyku MSIL; to znamená splnit specifikaci CLS danou Microsoftem. Dostupnost jednotlivých verzí .NET Frameworku u jednotlivých verzí Windows je shrnuta v následujícím obrázku:

.NET verze	1.0	1.1	2.0	3.0	3.5	4.0
Windows 95	nelze	nelze	nelze	nelze	nelze	nelze
Windows NT	lze doinstalovat	lze doinstalovat (SP6a)	nelze	nelze	nelze	nelze
Windows 98, Windows 98 SE	lze doinstalovat	lze doinstalovat	lze doinstalovat	nelze	nelze	nelze
Windows Me	lze doinstalovat	lze doinstalovat	lze doinstalovat	nelze	nelze	nelze
Windows 2000	lze doinstalovat	lze doinstalovat	lze doinstalovat (SP3)	nelze	nelze	nelze
Windows XP	lze doinstalovat	lze doinstalovat	lze doinstalovat (SP2)	lze doinstalovat (SP2)	lze doinstalovat	lze doinstalovat (SP3)
Windows Server 2003	??	součást systému	lze doinstalovat	lze doinstalovat (SP1)	lze doinstalovat	lze doinstalovat (SP2)
Windows Vista	částečná kompatibilita	částečná kompatibilita	součást systému	součást systému	lze doinstalovat	lze doinstalovat (SP1)
Windows Server 2008	??	??	??	součást systému	lze doinstalovat	lze doinstalovat
Windows Server 2008 R2	??	??	??	??	součást systému	lze doinstalovat
Windows 7	částečná kompatibilita	částečná kompatibilita	součást systému	součást systému	součást systému	lze doinstalovat

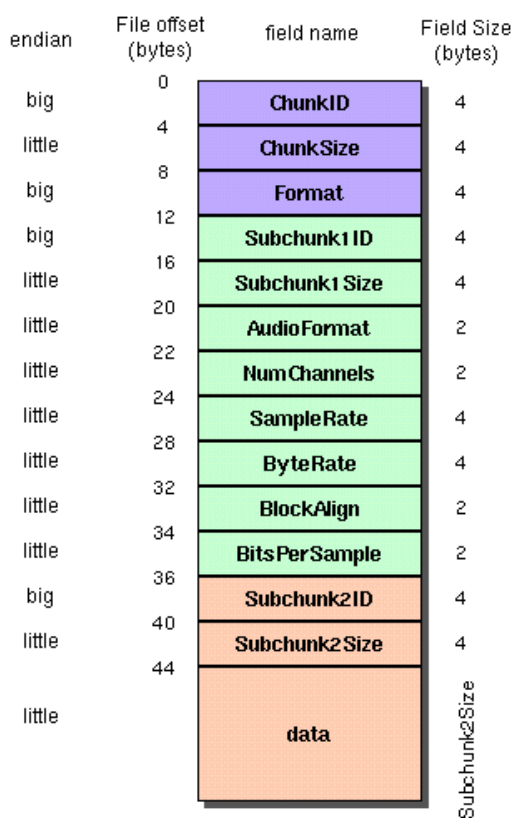
Obr. 59: Dostupnost verzí .NET Frameworku pro operační systémy Windows

### 3.3 Postup vytváření programu

Jako vývojové prostředí pro vytváření programu bylo použito vývojové prostředí Microsoft Visual C# 2010 Express, které je volně dostupné z webových stránek firmy Microsoft. Dále bylo potřeba stáhnout si a nainstalovat DirectX SDK, které je taktéž volně k dispozici na webových stránkách firmy Microsoft.

#### 3.3.1 Soubory WAV

Programy pro svoji činnost využívají soubor typu WAV. WAV je zkratka WAVE a vychází z anglického „Waveform Audio File Format“. Jedná se o soubor ve formátu RIFF. Jedná se o souborový formát firmy Microsoft určený pro ukládání multimediálních zvukových a obrazových předloh. RIFF se skládá z datových struktur zvaných shluky (anglicky chunk), každý shluk má svoji čtyřznakovou signaturu (ID) definovanou v hlavičce shluku. V případě souboru WAV se jedná o zvuková data a shluky dat jsou pouze dva, hlavička (označená „fmt“) a samotná data (označená „data“). Základní struktura formátu WAV je na následujícím obrázku.



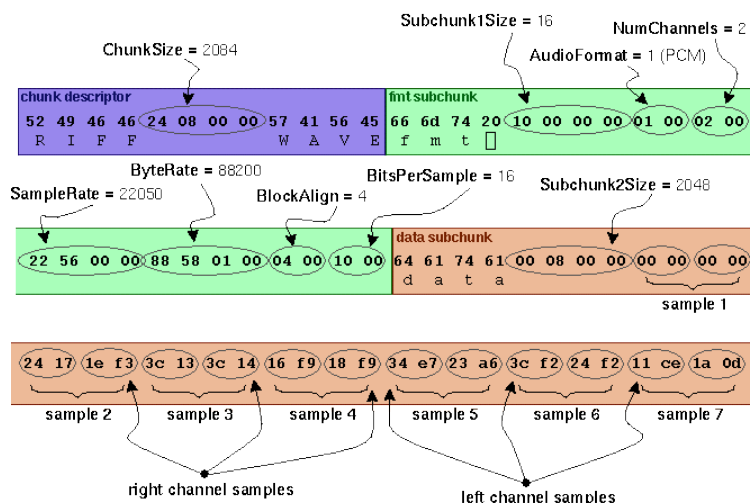
Obr. 60: Formát WAV souboru

Jednotlivá políčka mají význam shrnutý v následující tabulce:

Tab. 3 Formát WAV souboru

Pozice	Délka	Název	Popis
0	4	ChunkID	Obsahuje písmena „RIFF“ ve formátu ASCII, uložena jako big-endian
4	4	ChunkSize	36 + délka podshluku 2 (Subchunk2Size) nebo přesněji 4 + (8 + Subchunk1Size) + (8 + Subchunk2Size)
8	4	Format	Obsahuje písmena „WAVE“, uložena jako big-endian
12	4	Subchunk1ID	Obsahuje písmena „fmt“, uložena jako big-endian
16	4	Subchunk1Size	Velikost následujícího podshluku, pro PCM 16
20	2	AudioFormat	PCM = 1, lineární kvantizace, ostatní hodnoty znamenají určitou formu komprese
22	2	NumChannels	Počet kanálů, 1 = Mono, 2 = Stereo, ...
24	4	SampleRate	Vzorkovací frekvence
28	4	ByteRate	= SampleRate * NumChannels * BitsPerSample/8
32	2	BlockAlign	= NumChannels * BitsPerSample/8
34	2	BitsPerSample	Počet bitů na vzorek
	2	ExtraParamSize	Pokud se jedná o formát PCM, není použito
	X	ExtraParams	Extra parametry, pokud je použito
36	4	SubChunk2ID	Obsahuje písmena „data“, uložena jako big-endian
40	4	SubChunk2Size	Počet bytů, které následují a obsahují data
44	*	Data	Vlastní data

Díky velikosti políčka „SubChunk2Size“ je velikost souboru WAV omezena na 4GB. Vzorový WAV soubor je na následujícím obrázku:



Obr. 61: Vzorový WAV soubor

Implicitní způsob ukládání dat je little-endian. Soubory, které jsou uloženy v big-endianu, mají počáteční identifikátor „RIFX“ místo „RIFF“. 8-bitové vzorky jsou uloženy jako čísla bez znaménka v rozsahu hodnot 0 až 255. 16-bitové vzorky a jakékoliv vyšší jsou uloženy jako čísla se znaménkem v rozsahu hodnot -32768 až 32767, případně více pokud se jedná o více bitů na vzorek. Endianita je v informatice způsob uložení čísel v paměti počítače,

který definuje, v jakém pořadí se uloží jednotlivé bajty číselného datového typu. Označuje se také jako pořadí bajtů (anglicky byte order) a je jedním ze zdrojů nekompatibility při zpracování, ukládání dat a jejich následné výměně v digitální podobě. Díky masovému rozšíření architektury Intel x86 je nejpoužívanější little-endian. V případě little-endianu se na nejnižší paměťové místo uloží nejméně důležitý byte a za něj se dále ukládají více důležité byte až po nejvíce důležitý byte. Nejtypičtějším zástupcem je Intel x86.

	100	101	102	103	
...	1D	2C	3B	4A	...

Obr. 62: Uložení čísla 0x4A3B2C1D na adresu 100 jako little-endian

V případě big-endianu se na nejnižší paměťové místo uloží nejvíce významný byte a následují méně významné byte až po nejméně důležitý byte. Jedná se o opačné ukládání než v případě little-endianu. Typickými zástupci jsou mikroprocesory Motorola řady 68000, případně procesory SPARC.

	100	101	102	103	
...	4A	3B	2C	1D	...

Obr. 63: Uložení čísla 0x4A3B2C1D na adresu 100 jako big-endian

Některé architektury používají middle-endian (někdy též označovaný jako mixed-endian), kde se používá složitější způsob uložení jednotlivých bytů. Jedná se o kombinaci obou předchozích způsobů. Používá se například pro ukládání čísel s pohyblivou desetinnou čárkou u procesorů ARM.

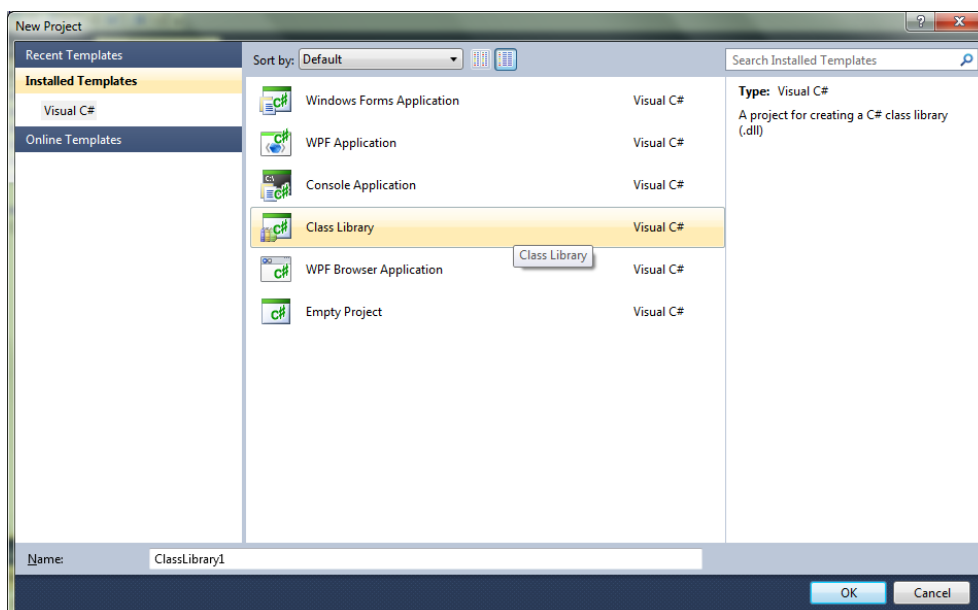
	100	101	102	103	
...	3B	4A	1D	2C	...

Obr. 64: Příklad uložení čísla 0x4A3B2C1D ve formátu middle-endian na adrese 100

### 3.3.2 Knihovna DXDSAR

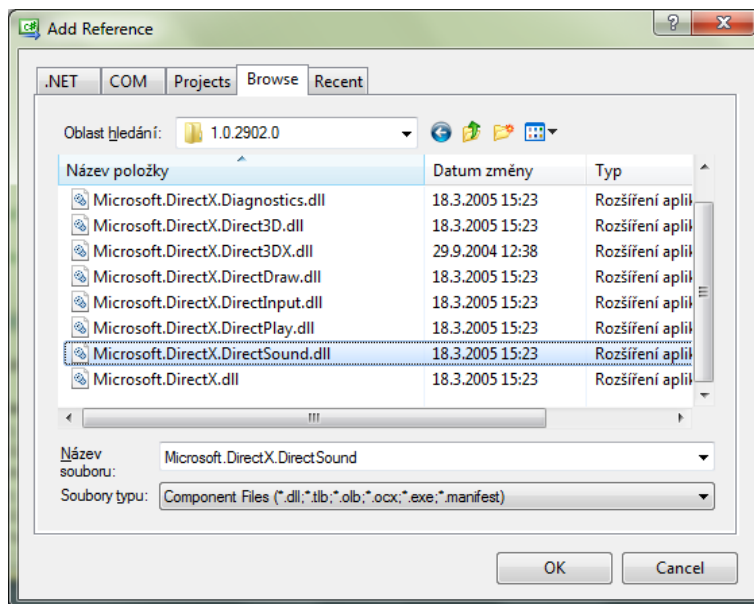
První součástí programu je knihovna DXDSAR, kdy je použitý částečně upravený kód ze stránek Aarona Ambermana. Knihovna se skládá ze dvou tříd. První je zodpovědná za vytvoření nahrávacího zařízení, nahrávacího bufferu a nahrání zvuku. Druhá o jeho uložení do souboru. Knihovna bude vytvořena volbou nová knihovna při zakládání nového

projektu v prostředí Microsoft Visual C# 2010 Express.



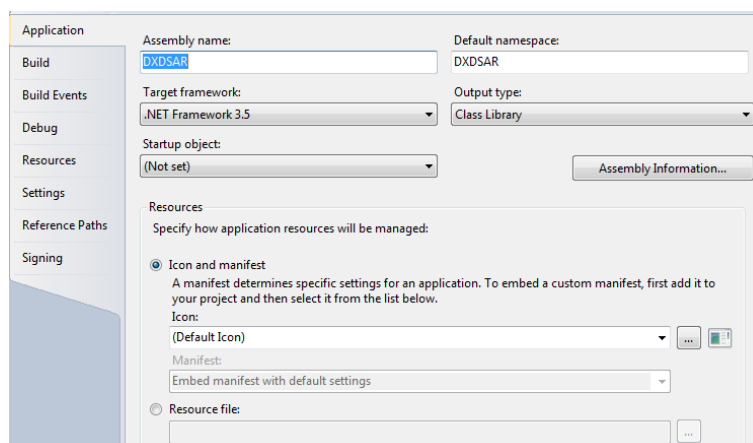
Obr. 65: Vytvoření knihovny v prostředí Microsoft Visual C# 2010 Express

Po vytvoření knihovny je potřeba přidat referenci pro Microsoft DirectSound. Reference se přidá v nabídce „Project“ - „Add Reference“ a následným nabrouzdáním knihovny.



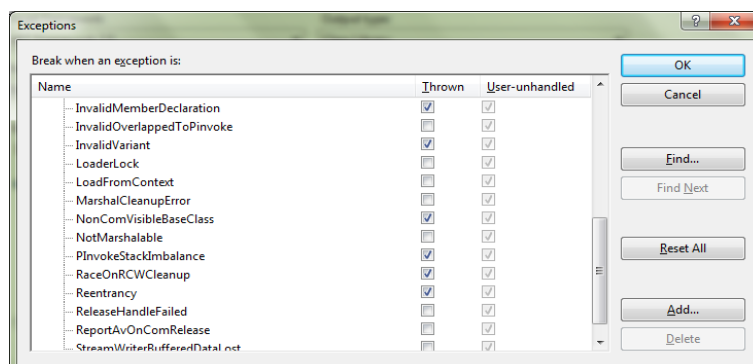
Obr. 66: Přidání knihovny DirectSound k projektu

Jelikož Microsoft Visual C# 2010 standardně vytváří soubory pro .NET Framework verze 4.0, je potřeba toho nastavení změnit. Nastavení je možné změnit po uložení celého projektu a provede se v menu „Project“ - „Project Properties“ volbou „Target framework“. Pro můj projekt jsem vybral verzi 3.5, která mně funguje s knihovnou DirectSound, kterou mám a zároveň se jedná o nejnovější verzi, kterou můžu použít.



Obr. 67: Výběr cílového .NET Frameworku

Jelikož se při následném ladění, objevila chyba způsobená mixováním řízeného a neřízeného kódu, je potřeba ještě vypnout tuhle kontrolu. Provedeme to v menu „Debug“ - „Exceptions“ vypnutím volby „LoaderLock“ ve stromečku „Managed Debugging Assistant“.



Obr. 68: Vypnutí "Loaderlocku"

Po těchto základních operacích můžeme začít psát jednotlivé třídy.

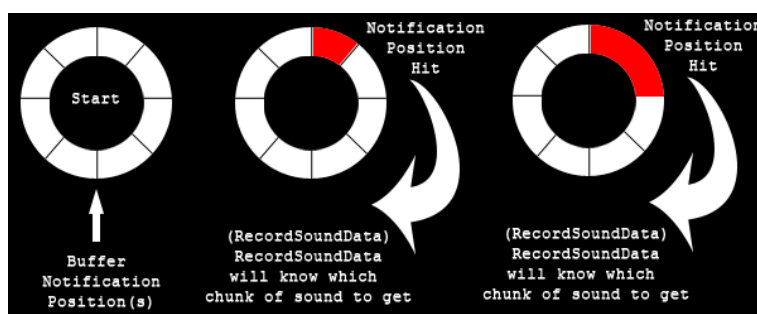
### ***Třída WaveFileWriter***

Je jednoduchá a stará se o uložení dat z paměti do souboru typu WAV. Obsahuje jeden konstruktor, jeden destruktork a metodu pro zápis dat do souboru, kdy se nejdříve zapíše hlavička a následně data. Kompletní zdrojový kód je uvedený v příloze.

### ***Třída SoundCapture***

Stará se o vytvoření nahrávacího zařízení, nahrávacího bufferu, záznam dat a využitím třídy WaveFileWriter i o zápis dat do souboru. Obsahuje konstruktor, destruktork a veřejné

metody pro začátek nahrávání, konec nahrávání a zápis dat do souboru. Třída využívá standardní třídy rozhraní Microsoft DirectSound. Jejich popis lze získat na webu firmy Microsoft v sekci MSDN. Jakmile začne nahrávání, třída vytvoří nový thread, který bude zodpovědný za nahrávání. Třída využívá kruhový buffer, jehož princip by se dal popsat následujícím obrázkem:

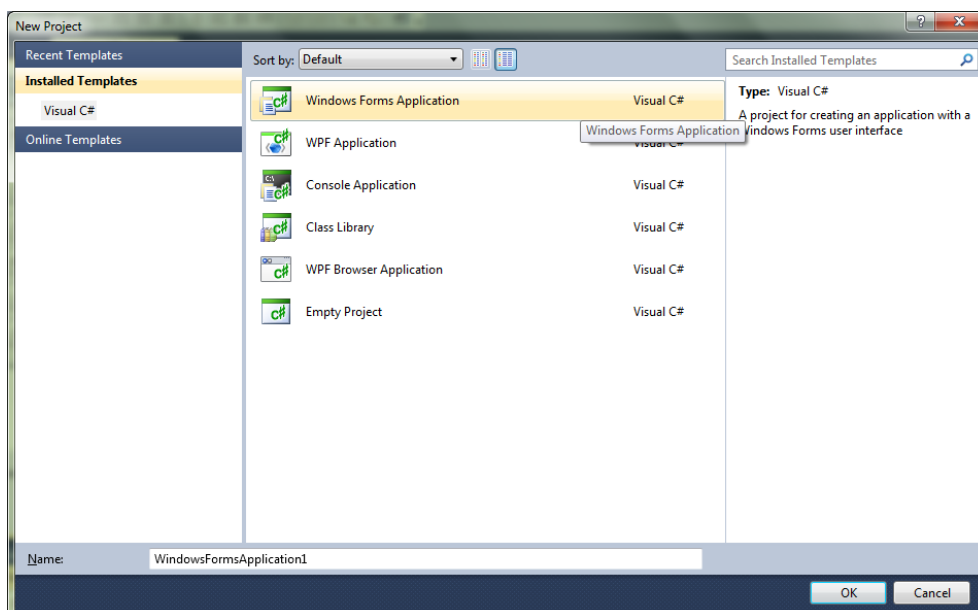


Obr. 69: Princip kruhového bufferu

Vytvoříme jednotlivé pozice a jakmile při nahrávání dosáhnou data v bufferu na tuto pozici, spustí se metoda, která nám data zkopíruje z bufferu do paměti, aby mohly být po ukončení nahrávání uložena do souboru. Kompletní zdrojový kód je uveden v příloze.

### 3.3.3 Program pro sběr dat

Program pro sběr dat vytvoříme jako novou aplikaci „Windows Forms“. Vývojové prostředí za nás vytvoří vše potřebné a nemusíme se starat o funkci grafického prostředí.



Obr. 70: Vytvoření projektu

Po vytvoření projektu je potřeba přidat referenci na námi vytvořenou knihovnu DXDSAR, na knihovnu Microsoft DirectSound, změnit cílový .NET Framework a vypnout „LoaderLock“. Tyto operace jsou shodné s totožnými operacemi v případě vytváření knihovny. Následně navrhne grafickou podobu našeho programu, rozhodl jsem se pro jednoduchost vytvořit repliku klávesnice jak je vidět z následujícího obrázku:



Obr. 71: Okno programu pro sběr dat

Na okno formuláře byla postupně přetažena všechna tlačítka, jedno textové políčko a časovač. Pro každé tlačítko byla vygenerována událost, která nastane po stisknutí, dvojitým poklepáním na tlačítko. Taktéž byla vygenerována událost pro otevření a zavření formuláře. Jelikož jsem chtěl, aby bylo možné měnit grafickou podobu tlačítek, bylo nutné zakomentovat následující řádek v hlavním programu Program.cs (kompletní zdrojový kód je uvedený v příloze).

```
//Application.EnableVisualStyles();
```

Změnu barvy tlačítek využijeme pro informování uživatele, zda už má uložený WAV soubor pro příslušnou klávesu. V případě, že ano, tlačítko dostane modrý popisek. O změnu barvy popisku tlačítka se stará metoda Buttons bez parametrů a bez návratové hodnoty. Pro každé tlačítko provede následující kontrolu a případně operaci.

```
private void Buttons()
{
    if (File.Exists("a.wav"))
        button1.ForeColor = Color.Blue;
```

V případě, že stisknu tlačítko, zavolá se metoda pro obsluhu stisku tlačítka, pro každé tlačítko je vlastní. Pokud nenahráváme (zjišťuje se flagem – proměnnou isRecording), metoda začne nahrávání, zneaktivní příslušné tlačítko, nastaví flag, aby se nedalo spustit nahrávání znovu, zapne časovač, který je nastavený na jednu vteřinu a nahrává zvuk. Zvuk je nahráván se vzorkovací frekvencí 44.1 kHz, která by měla být dostatečná, jelikož se jedná o kvalitu CD, počet kanálů mono, jelikož máme pouze jeden mikrofón a počet bitů

na vzorek 16, abychom měli možnost lepšího rozlišení než při použití bitů 8.

```
private void button1_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button1.Enabled = false;
        fileName = "a.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ,
SoundCapture._16_Bit, SoundCapture._MONO);
        sc.OnWavComplete += new
SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 1;
    }
}
```

Stejnou metodu obsahuje každé tlačítko, mění se pouze název souboru a číslo ukládané do proměnné button. Po spuštění nahrávání a časovače se každých 100 milisekund (tak je nastavený časovač), zavolá metoda pro obsluhu časovače. Metoda přičte do proměnné ticks jedničku a zkontroluje, zda nebylo již dosaženo jedné sekundy, což je limit pro nahrávání. Tento limit by měl bez problémů stačit, protože stisk jedné klávesy trvá v průměru 100ms, takže máme určitě rezervu. Pokud dosáhne časovač námi nastaveného limitu, ukončí nahrávání, začne zápis dat do souboru a vynuluje flag. V každém případě nastaví text v textovém poli vpravo nahoře na zbývající čas pro nahrávání.

```
private void timer1_Tick(object sender, EventArgs e)
{
    ticks++;
    if (ticks == 10)
    {
        timer1.Stop();
        sc.StopRecording();
        isRecording = false;
        sc.WriteWaveFile(fileName);
        sc.Dispose();
        ticks = 0;
    }
    label1.Text = Convert.ToString(1000 - (ticks * 100));
}
```

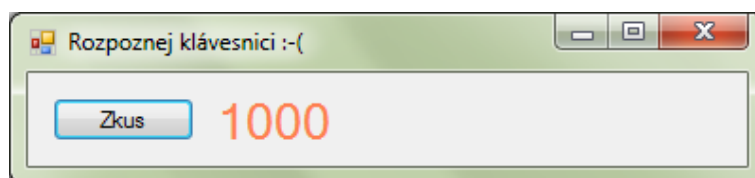
Po ukončení zápisu do souboru se pustí metoda `OnWavComplete`. Metoda zavolá metodu `Buttons`, která aktualizuje barvy na tlačítkách, následně opět zaktivní tlačítko, které bylo zneaktivněno v průběhu nahrávání.

```
private void sc_OnWavComplete(object sender, EventArgs e)
{
    Buttons();
    label1.Text = "1000";
    switch (button)
    {
        case 1: button1.Enabled = true;
            break;
    }
}
```

Kompletní zdrojový kód je uvedený v příloze.

### 3.3.4 Program pro analýzu dat

Založíme jako nový projekt „Windows Forms“ stejně jako předchozí program. Jako referenci přidáme knihovnu `DXDSAR`, `Microsoft DirectSound` a knihovnu `FFTWLib`, kterou použijeme k realizaci Fourierovy transformace. Stejně tak provedeme změnu cílového `.NET Frameworku` a vypnutí „`LoaderLocku`“. Grafická podoba aplikace je následující:



Obr. 72: Okno programu pro rozpoznání (analýzu) dat

Program využívá obsluhu grafického rozhraní a třídu pro načtení WAV souboru. Program využívá jednoho tlačítka, jednoho časovače a dvou labelů. Na začátku programu se načtou data ze souborů do paměti (pouze ty soubory, které jsou v adresáři obsaženy). Následně se pro všechna data provede Fourierova transformace. V případě stisku tlačítka se zavolá obslužná metoda, která provede nahrání zvuku do dočasného souboru. Nahrávání se provede se stejným nastavením jako v případě učení klávesnice (vzorkovací frekvence 44.1 kHz, mono a 16 bitů na vzorek).

```
private void button1_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
```

```
        timer1.Start();
        button1.Enabled = false;
        fileName = "temp.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ,
SoundCapture._16_Bit, SoundCapture._MONO);
        sc.OnWavComplete += new
SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        label1.Text = "";
    }
}
```

Po spuštění nahrávání a časovače se každých 100 milisekund (tak je nastavený časovač), zavolá metoda pro obsluhu časovače. Metoda přičte do proměnné ticks jedničku a zkontroluje, zda nebylo již dosaženo jedné sekundy, což je limit pro nahrávání. Pokud ano, uloží se soubor na disk, a vynuluje se počítadlo.

```
private void timer1_Tick(object sender, EventArgs e)
{
    ticks++;
    if (ticks == 10)
    {
        timer1.Stop();
        sc.StopRecording();
        isRecording = false;
        sc.WriteWaveFile(fileName);
        sc.Dispose();
        ticks = 0;
    }
    label2.Text = Convert.ToString(1000 - (ticks * 100));
}
```

Po skončení ukládání WAV souboru se zavolá metoda, která provede načtení souboru z disku do paměti, Fourierovu transformaci (stejným postupem jako v případě souborů s naučenými vzorky dat) a následné porovnání. Porovnání je prováděno postupným procházením všech dat a zjišťováním minimální odchylky mezi dvěma skupinami dat.

```
public int Porovnej()
{
    double dif = Double.MaxValue;
    int index = -1;
    for (int i = 0; i < 104; i++)
    {
```

```
        double adif = 0;
        if (datain[i].Length > 0)
        {
            for (int j = 0; j <
Math.Min(dataout[i].Length,tempout.Length); j++)
            {
                adif = adif + Math.Abs(tempout[j] -
dataout[i][j]);
            }
            if (adif < dif)
            {
                dif = adif;
                index = i;
            }
        }
    }
    return index;
}
```

Po skončení procházení je vrácen index z konstanty řetězců (názvů souboru) a zobrazen v textovém poli. Kompletní zdrojový kód je umístěný v příloze.

### ***Třída Class1***

Třída se stará o načtení dat z WAV souboru. Obsahuje pouze jedinou veřejnou metodu ReadWavFile. V případě, že se nepodaří soubor otevřít, buďto z důvodu, že neexistuje nebo není přístupný pro čtení, vrátí prázdné pole dat. V první části se provede čtení hlavičky a následně se provede čtení dat. Po načtení všech dat se provede větvení na základě bitů na vzorek a počtu kanálů, jelikož je nutné provést konverzi dat pro jednotný výstupní formát, kterým je vzorkovací frekvence 44.1 KHz, mono a 16 bitů na vzorek. Proto se v případě sterea provede průměrování z obou kanálů a v případě 8 bitů na vzorek rozšíření na bitů 16. Kompletní zdrojový kód je umístěný v příloze.

## 4 EXPERIMENTÁLNÍ OVĚŘENÍ PROGRAMU

### 4.1 Výběr experimentů k ověření funkčnosti programu

Po realizaci programu je nutné ověřit jeho funkčnost. Rozhodl jsem se pro soustavu testů, kterými by měl pravděpodobně projít každý keylogger založený na analýze zvukového signálu, pokud má být úspěšný. V rámci ověření můžeme vybírat mezi klávesnicí, na kterou byl program naučený, klávesnici na kterou program nebyl naučený (využijeme v případě, kdy nebudeme mít k odposlouchávané klávesnici přístup pro naučení). Znaky na klávesnici může zadávat osoba, která klávesnici naučila, případně osoba, která klávesnici nenaučila (tento případ bude mnohem častější). Záležet může na mikrofону, kterým bylo prováděno učení, případně rozpoznávání. Stejně tak může záležet na pozici snímacího mikrofону, který může být v rozdílné pozici a na okolním hluku (případně psaní na klávesnici od ostatních uživatelů v případě použití keyloggeru například v prostředí kanceláři typu „Open space“). Jelikož program testuje vždy pouze jednu klávesu, nedá se ověřit vliv rychlosti psaní (počtu úhozů za minutu). Taktéž nebudu uvažovat kombinaci těchto možností, protože složitost testu by neúměrně narůstala s každým novým kritériem.

### 4.2 Realizace experimentů

Realizace experimentů byla provedena v obyčejné tiché místnosti ve večerních hodinách. Místnost je bez speciálního odhlučnění. K zobrazení průběhu WAV souborů a jejich spektra mně posloužila testovací verze programu SigView od firmy SignalLab.

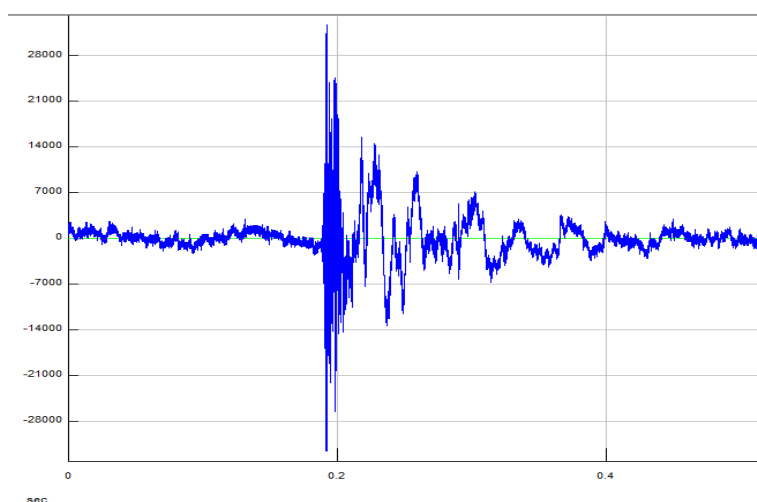
#### 4.2.1 Ověření funkčnosti na naučené klávesnici

K ověření funkčnosti programu jsem použil dvě různé klávesnice. Desktopovou a notebookovou klávesnici. K zaznamenání zvuku posloužil integrovaný mikrofon v notebooku Acer Aspire Timelin 5810TZ.

##### *Desktopová klávesnice*

Použitou klávesnicí byla obyčejná klávesnice A4Tech KB-5A v provedení USB. Klávesnice je vybavena mechanickými spínači s gumovou membránou. Jedinou odlišností od naprostého standardu je její vodotěsnost. Provedl jsem naučení klávesnice a poté následný test rozpoznání. Bohužel s touto klávesnicí byly výsledky rozpoznávání víceméně náhodné a rozhodně by se nedaly označit za uspokojivé. Zkoušel jsem i upravit částečně

program (vzít pouze omezenou část spektra případně použít jenom část, kdy předpokládám stisk klávesy), nicméně ani tak se mně nepodařilo dosáhnout žádné uspokojivé úspěšnosti.



Obr. 73: Průběh signálu klávesy "Space" na desktopové klávesnici

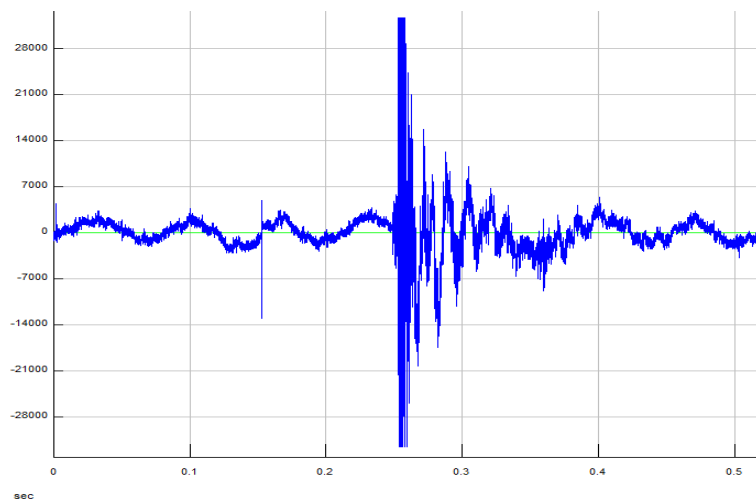
### ***Notebooková klávesnice***

Druhou použitou klávesnicí byla klávesnice z notebooku Acer Aspire Timeline 5810TZ. Použitou technologií spínačů jsi nejsem jistý, ale předpokládám „nůžkové“ spínače. S touto klávesnicí jsem se dostal k podobným výsledkům jako byla předchozí klávesnice.

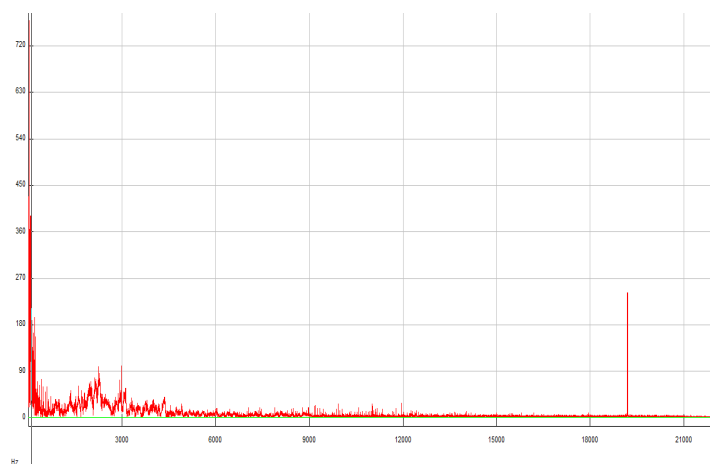
Za největší problém považuju kromě šumu mikrofonu, který se mně nepodařilo potlačit, tak i krátké špičky, které jsou vidět i na předchozím obrázku a kterých jsem se neuměl zbavit. Nejvhodnější nebyl zřejmě ani výběr použitých klávesnic, kdy většina kláves vydávala velmi podobný zvuk na poslech a rozdíly byly mezi různými stisky stejné klávesy větší než mezi rozdílem mezi různými klávesami. Jelikož byly zvuky velmi podobné, bylo následně podobné i jejich spektrum po transformaci z časové oblasti do frekvenční.

#### **4.2.2 Ověření funkčnosti na nenaučené klávesnici**

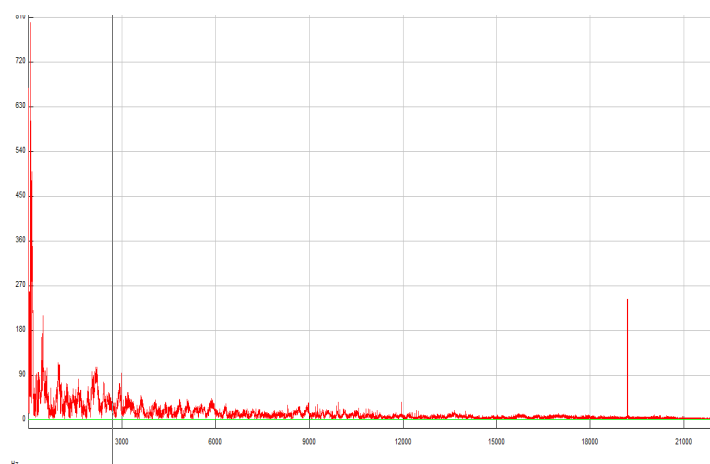
Pokus, který by mohl být použitý i v praxi je naučit program na jednu klávesnici a následně jej otestovat na klávesnici jiné, jelikož k odposlouchávané klávesnici nemusíte mít přístup a nemusíme mít možnost jejího naučení se. Jelikož měl program slabou úspěšnost i při rozpoznávání klávesnice, při použití nenaučené klávesnice byla úspěšnost rozpoznání nulová. Neočekávám ale větší úspěšnost ani u kvalitnějšího programu, jelikož se jednalo o klávesnice s rozdílnou konstrukcí a jak průběh signálu, tak i spektra signálu jsou velmi odlišná, jak je vidět z následujících obrázků.



Obr. 74: Průběh signálu klávesy "Space" na notebookové klávesnici



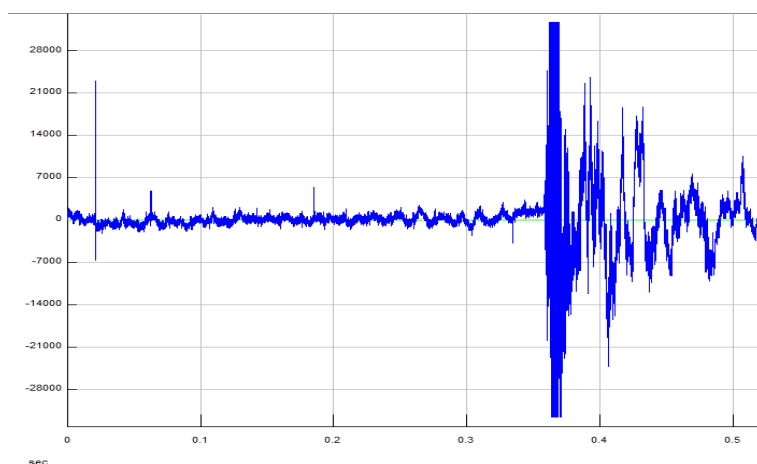
Obr. 75: Spektrum signálu klávesy "Space" desktopové klávesnice



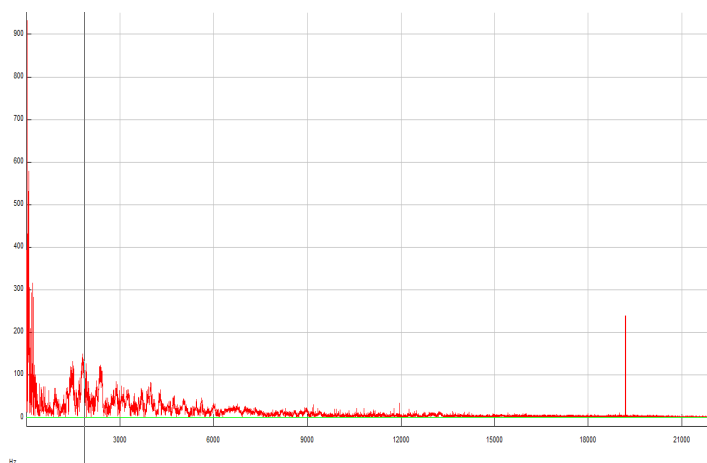
Obr. 76: Spektrum signálu klávesy "Space" notebookové klávesnice

### 4.2.3 Ověření funkčnosti při psaní různými osobami

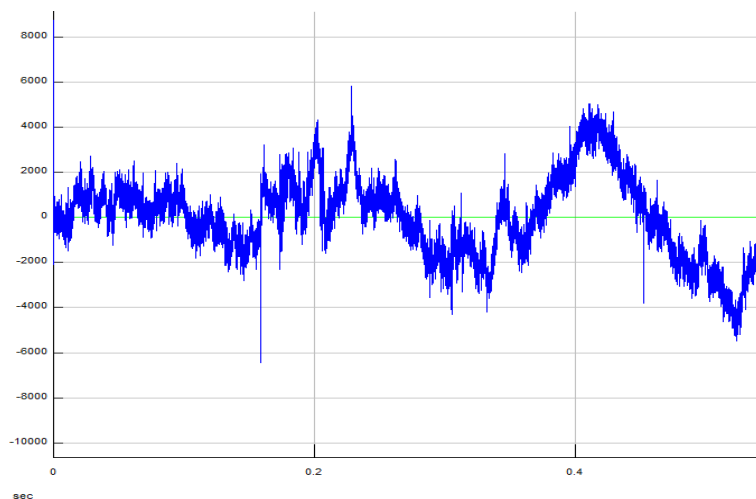
Každý člověk píše na klávesnici částečně jiným způsobem, proto je zajímavé zkusit ověřit, zda je možné použít program naučený jednou osobou pro odposlouchávání znaků psaných jinou osobou. Tento experiment je i velmi praktický, protože přístup ke klávesnici pro naučení můžeme získat, ale psát na ni bude většinou někdo jiný než my. K testu jsem použil tři různé osoby (jedna z nich klávesnici učila) a jelikož každá osoba psala jiným způsobem, nebylo dosaženo žádného pozitivního výsledku. Rozdíl ve stylu psaní je vidět jak z průběhu signálu, tak i z průběhu spekter signálu.



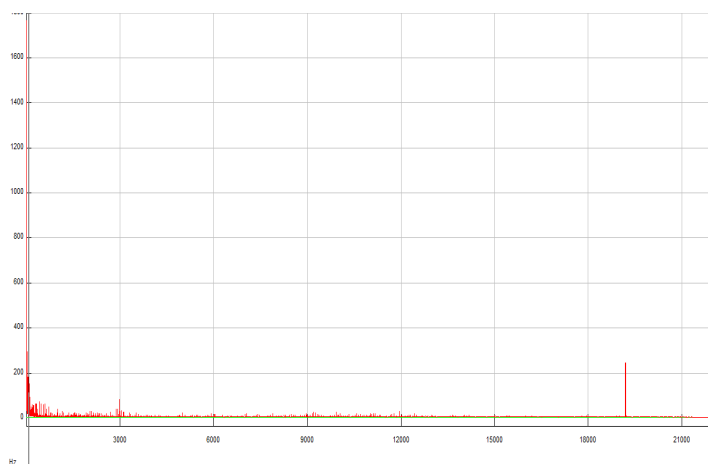
Obr. 77: Průběh signálu pro klávesu „Space“ a pro testovanou osobu č. 2



Obr. 78: Spektrum signálu pro klávesu "Space" a pro testovanou osobu č. 2



Obr. 79: Průběh signálu pro klávesu „Space“ a pro testovanou osobu č. 3



Obr. 80: Spektrum signálu pro klávesu "Space" a pro testovanou osobu č. 3

Průběhy signálu i spektra pro první testovanou osobu jsou v kapitolách nahoře.

#### 4.2.4 Ověření funkčnosti různými mikrofony

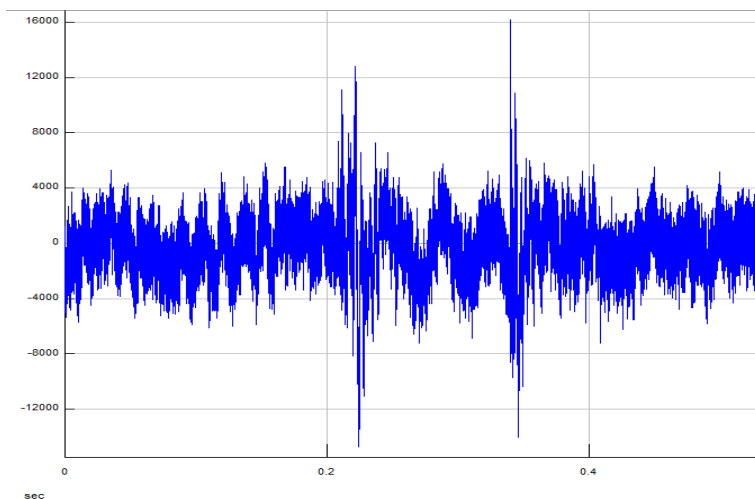
Další věcí, která se může změnit je mikrofon, kterým snímáme akustický signál. Tato varianta není příliš pravděpodobná, nicméně by se mohla vyskytnout. Pro pokus jsem zvolil interní mikrofon notebooku Acer Aspire Timeline 5810TZ a externí mikrofon (bez zjištění typu).

##### *Interní mikrofon*

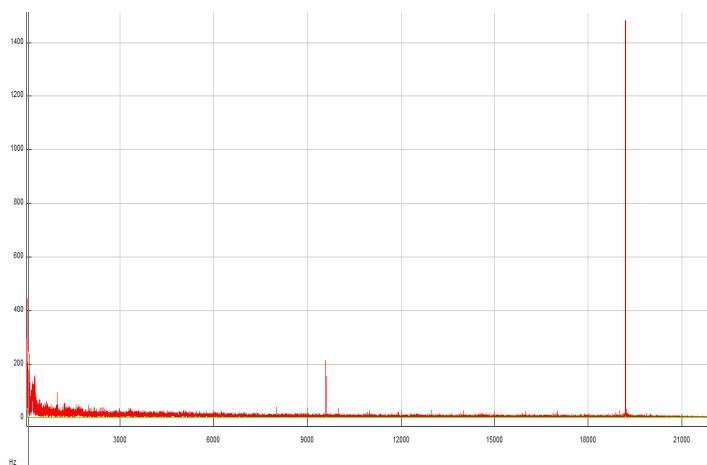
Všechny dosavadní pokusy probíhaly pro interní mikrofon, proto není potřeba jej dále speciálně testovat.

### *Externí mikrofon*

Jako další mikrofon pro experiment jsem zvolil externí mikrofon bez uvedení značky a typu. Při stejném nastavení byl mikrofon citlivější a proto i hladina šumu byla podstatně vyšší. Proto bych použití jiného mikrofonu než na který je program naučený nedoporučil.



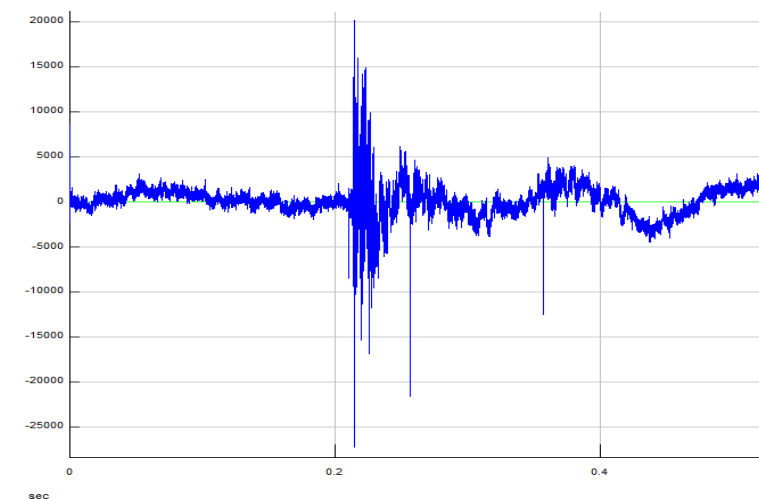
*Obr. 81: Průběh signálu pro klávesu „Space“ a pro externí mikrofon*



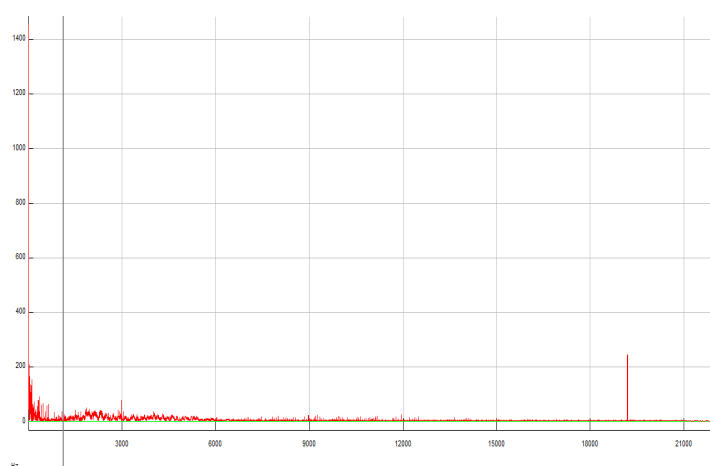
*Obr. 82: Spektrum signálu pro klávesu "Space" a pro externí mikrofon*

#### **4.2.5 Vliv změny polohy mikrofonu**

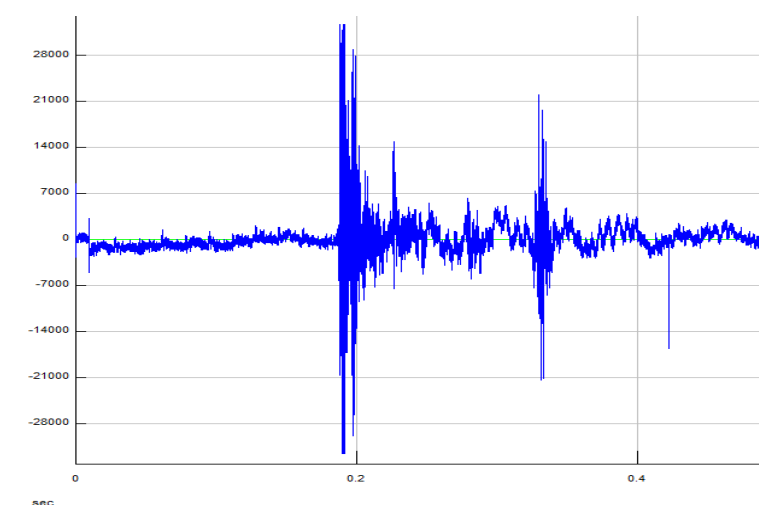
Při tomto experimentu jsem zkoumal zda má nějaký vliv poloha snímacího mikrofonu oproti odpovídané klávesnici. Mikrofon jsem postupně umístil vlevo, vpravo, dopředu a dozadu klávesnice. Jak je vidět z příložených průběhů signálů i spekter, poloha mikrofonu má na signál vliv, proto by bylo vhodnější s mikrofonem mezi fázemi učení a rozpoznávání nehýbat.



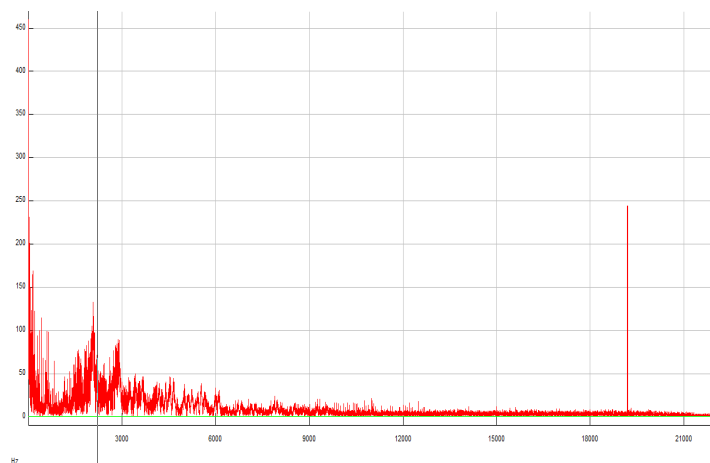
Obr. 83: Průběh signálu pro klávesu „Space“ a mikrofon vlevo



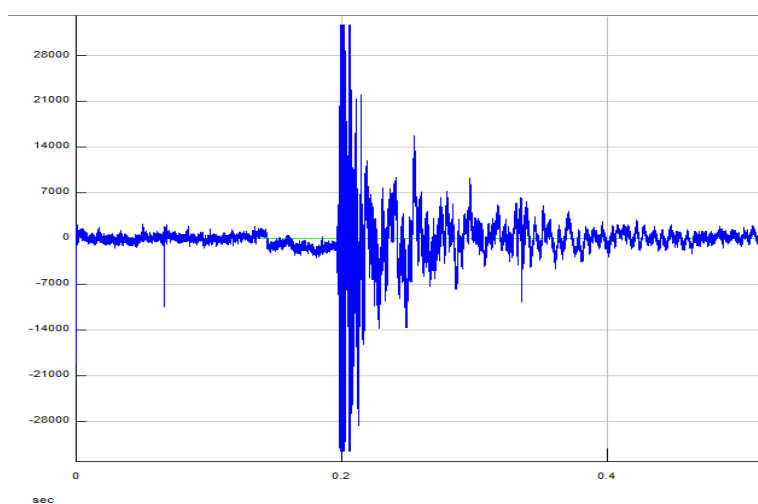
Obr. 84: Spektrum signálu pro klávesu "Space" a mikrofon vlevo



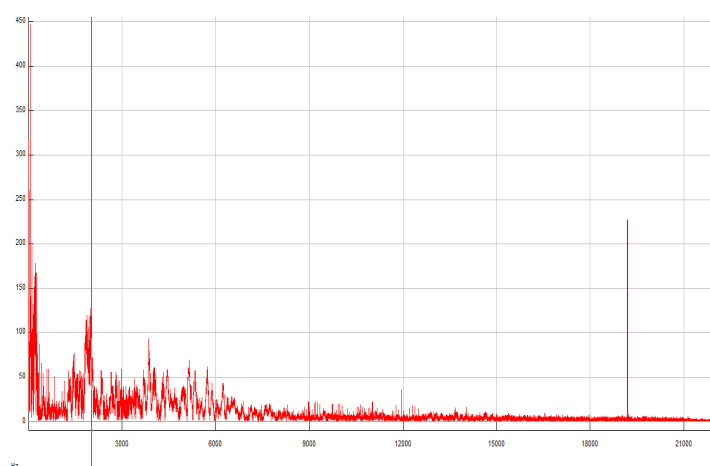
Obr. 85: Průběh signálu pro klávesu „Space“ a mikrofon vepředu



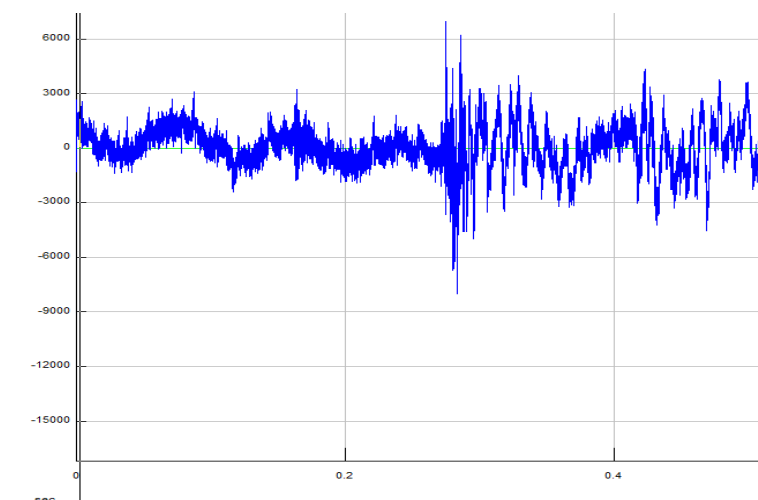
Obr. 86: Spektrum signálu pro klávesu "Space" a mikrofon vepředu



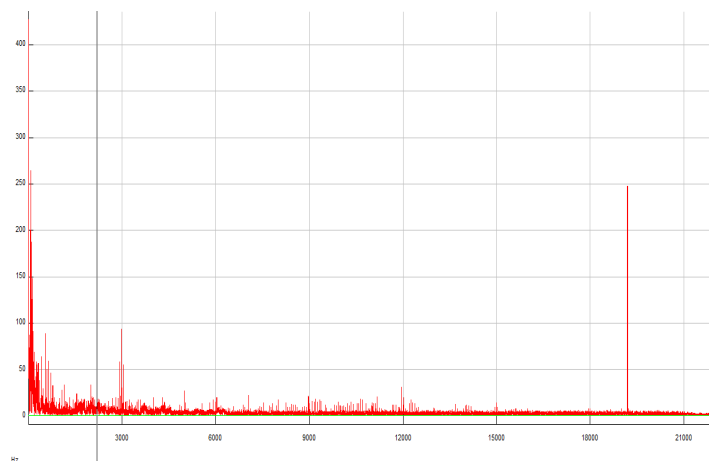
Obr. 87: Průběh signálu pro klávesu „Space“ a mikrofon vpravo



Obr. 88: Spektrum signálu pro klávesu "Space" a mikrofon vpravo



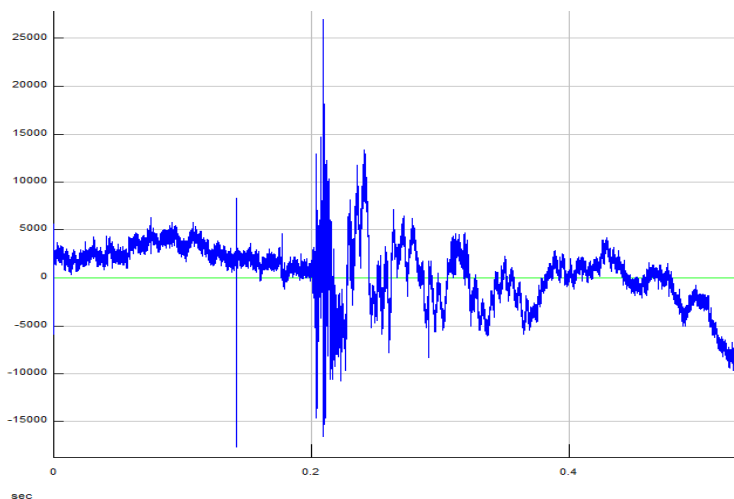
Obr. 89: Průběh signálu pro klávesu „Space“ a mikrofon za



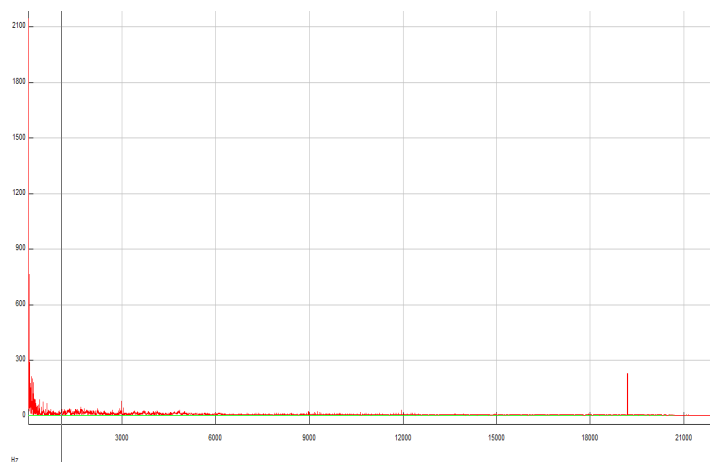
Obr. 90: Spektrum signálu pro klávesu "Space" a mikrofon za

#### 4.2.6 Vliv okolního hluku

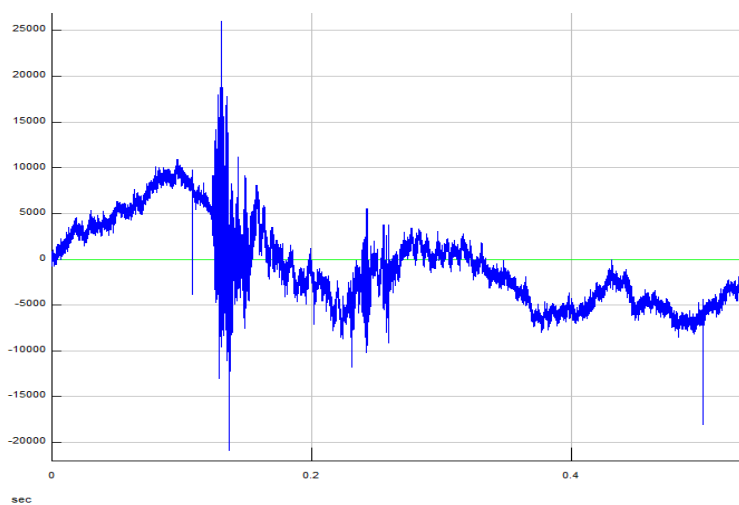
Ne vždy budeme mít při osposlouchávání k dispozici absolutní ticho. Keylogger v reálném provozu se naopak bude potýkat se všudypřítomným hlukem. Pokud je zvukové pozadí stále, existuje možnost nahrát jej a následně zkusit ze signálu, který je kombinací zvuku na pozadí a zvuku odposlouchávané klávesnice, extrahovat. Tuto možnost jsem nicméně do programu neimplementoval. Problematičtější variantou je použití keyloggeru v prostorách, kde se vyskytuje nepravidelný hluk (hovory, zvuky vydávané počítačovými perifériemi, ...). Troufnu si říci, že takovéto prostředí znemožňuje nasazení keyloggeru na bázi analýzy zvukového signálu. Jelikož jsem po předchozích experimentech neočekával, že by program dával relevantní výsledky, porovnal jsem pouze průběhy signálu a spektra signálu pro tři různé prostředí. První referenční prostředí bylo změřeno již v předchozích pokusech.



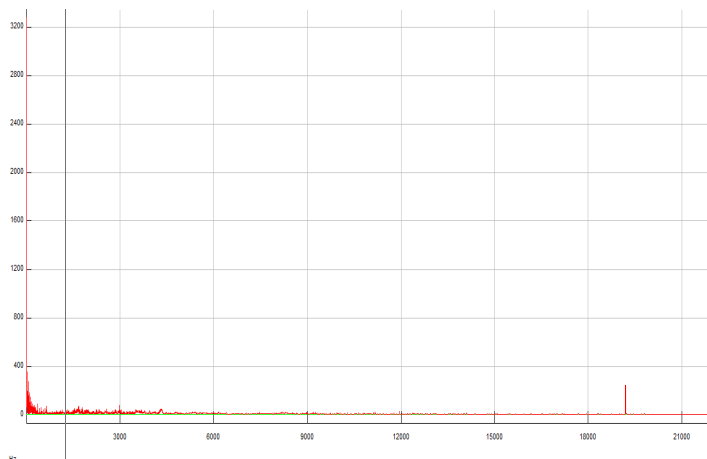
Obr. 91: Průběh signálu pro klávesu „Space“ a pro testované prostředí č. 2



Obr. 92: Spektrum signálu pro klávesu "Space" a pro testované prostředí č. 2



Obr. 93: Průběh signálu pro klávesu „Space“ a pro testované prostředí č. 3



Obr. 94: Spektrum signálu pro klávesu "Space" a pro testované prostředí č. 3

Z průběhu signálů a spekter signálů je nicméně vidět, že se vzrůstajícím hlukem se rozpoznávání stává těžším, jelikož signály jsou si podobnější (spektrum je plošší) a jsou menší rozdíly mezi různými klávesami. Potvrzuje to tedy mou domněnku o malé použitelnosti keyloggerů založených na analýze zvukového signálu v hlučných prostředích.

### 4.3 Návrhy na zlepšení programu

Jelikož mně program nefungoval podle očekávání, je vhodné se zamyslet, co by šlo udělat lépe a jak zlepšit jeho rozpoznávací schopnosti, případně i jak jeho funkcionalitu rozšířit o nějaké další možnosti pro zjednodušení obsluhy a používání.

#### 4.3.1 Zlepšení rozpoznávání

Ke zlepšení rozpoznávání by jistě přispěla hlučnější klávesnice s hlasitější odezvou na stisk klávesy. Nicméně odposlouchávanou klávesnici si nemůžeme vybrat. Použitím kvalitnějšího mikrofону bychom jistě dosáhli taktéž lepších výsledků, ale předchozí práce, ze kterých jsem se pokusil vycházet, také použily obyčejný mikrofon. Největší změny bych tedy udělal do programu pro rozpoznávání. Pro zlepšení analýzy programu by bylo určitě vhodné se pokusit odstranit nahodilé špičky, které následně hodně zkreslují vstupní signál. Jsou i důvodem, proč jsem neuspěl s modifikací, kdy jsem se snažil získat začátek stisknutí klávesy překročením nastavené hodnoty. Stejně tak je určitě možné laborovat s frekvenčním omezením, kdy většina signálu je mezi dvěma frekvencemi a ne v celém pásmu slyšitelného signálu. Nicméně i aktivací tohoto omezení jsem nedosáhl zvýšené přesnosti. Dále by bylo vhodné mít možnost uložení více vzorků pro každou klávesu pro

spřesnění analýzy, případně pro pokus o naučení neuronové sítě. Pro jakékoliv (i málo hlučné) prostředí by se hodila možnost redukce šumu na pozadí. Tyto vylepšení bych se pokusil implementovat do další verze programu pro sběr a analýzu dat.

#### **4.3.2 Rozšíření možností**

Program by bylo určitě dále vhodné rozšířit o databázi klávesnic, kdy bych měl možnost si nadefinovat, naučit a uložit vícero klávesnic a pak mít možnost mezi nimi přepínat. Dále by bylo vhodné mít možnost si nahraný vzorek přehrát, zobrazit jeho průběh, případně základní charakteristiky a například i spektrum. Pro aplikaci v hlučnějším prostředí by se hodila funkce pro nahrání zvukového pozadí (šumu na pozadí), aby jej bylo následně možné odstranit z užitečného signálu.

## ZÁVĚR

Diplomová práce se zabývala možností odposlouchávání klávesnice analýzou zvukového signálu. Odposlouchávání klávesnice je zajímavý problém a jedná se o hrozbu, která je v současné době celkem neprávem podceňována. Pomocí odposlouchávání klávesnice je možné získat přístupová hesla například do rozsáhlých informačních systémů nebo do internetové bankovnictví. Útočník, který tyto hesla zjistí může napáchat velké škody na majetku. V práci jsem postupně rozebral konstrukci počítačové klávesnice, různé metody softwarového keyloggingu, různé metody hardwarového keyloggingu a možnosti obrany proti nim. Speciální pozornost si zasloužily odborné práce, které se věnovaly odposlouchávání klávesnice pomocí analýzy zvukového signálu. Závěry prací byly pro moji diplomovou práci optimistické. Keylogging tímto způsobem je jistě možný. Dále jsem se pokusil realizovat jednoduchý program na zaznamenávání akustického signálu pro stisknuté klávesy. Program ukládal zvuky konkrétních kláves do souborů WAV pro pozdější použití. Pro rozpoznávání stisknuté klávesy jsem realizoval další jednoduchý program. V poslední části jsem tento program otestoval. I přes pozitivní očekávání, program nebyl schopný spolehlivě rozeznávat stisknuté klávesy. Dokázal rozlišit některé klávesy, u kterých je rozdíl slyšitelný i lidským uchem. Klávesy, které zní podobně nicméně rozlišit nedokázal. I přes špatné chování programu jsem pak provedl některé další testy a i tak objevil slabiny tohoto způsobu odposlechu. Jedná se o velmi hlučné prostředí, kdy může být i zvuk kláves srovnatelný s hlukem na pozadí. Případně se jedná o měnící se zvukové pozadí, které se nedá predikovat a tím pádem ani odstranit z analyzovaného signálu. Na základě neúspěchu jsem se pokusil navrhnout možné vylepšení programu a dále i možnosti jeho rozšíření. V každém případě bych se možnosti realizace keyloggeru na základě analýzy akustického signálu nevzdával.

## SEZNAM POUŽITÉ LITERATURY

### Monografie:

- [1] GUNNERSON, Eric. *Začínáme programovat v C#*. Vydání první. Hornocholupická 22, 143 00 Praha 4, <http://www.cpress.cz> : Vydavatelství a nakladatelství Computer Press, 2001. 316 s. ISBN 80-7226-525-3.
- [2] HANÁK, Ján. *C# praktické příklady*. Praha : Grada Publishing, a.s., 2006. 288 s. ISBN 80-247-0988-0.
- [3] ŠÍMA, Jiří; NERUDA, Roman. *Teoretické otázky neuronových sítí*. 1. vydání. Praha : Univerzita Karlova, 1996. 390 s. ISBN 80-85863-18-9.

### Internetové zdroje:

- [4] .NET Framework. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 18. 11. 2005, last modified on 18. 11. 2005 [cit. 2011-05-11]. Dostupné z WWW: <[http://cs.wikipedia.org/wiki/.NET\\_Framework](http://cs.wikipedia.org/wiki/.NET_Framework)>.
- [5] AMBERMAN, Aaron. *AaronAmberman.com* [online]. 2009 [cit. 2011-05-18]. C# Record Audio Tutorial. Dostupné z WWW: <<http://aaronamberman.com/tutorials/csaudio.php>>.
- [6] ASONOV, D.; AGRAWAL, R. Keyboard acoustic emanations. In Proceedings of 2004 IEEE Symposium on Security and Privacy. In [online]. [s.l.] : [s.n.], 2004 [cit. 2011-05-02]. Dostupné z WWW: <<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1301311>>.
- [7] BIOS. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 8. 4. 2005, last modified on 5. 3. 2011 [cit. 2011-05-20]. Dostupné z WWW: <<http://cs.wikipedia.org/wiki/BIOS>>.
- [8] Bluetooth. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 25. 6. 2005, last modified on 13. 5. 2011 [cit. 2011-05-18]. Dostupné z WWW: <<http://cs.wikipedia.org/wiki/Bluetooth>>.
- [9] Blue Pill %28malware%29. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 24 August 2006, last modified on 24 April 2011 [cit. 2011-05-20]. Dostupné z WWW: <[http://en.wikipedia.org/wiki/Blue\\_Pill\\_%28malware%29](http://en.wikipedia.org/wiki/Blue_Pill_%28malware%29)>.

- [10] *C# Corner* [online]. 2002 [cit. 2011-05-20]. Key Logger Application in C# . Dostupné z WWW: <<http://www.c-sharpcorner.com/UploadFile/grusso/KeyLoggerApplicationinCSharp11252005000941AM/KeyLoggerApplicationinCSharp.aspx>>.
- [11] Cepstrum. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 31 March 2002, last modified on 30 April 2011 [cit. 2011-05-22]. Dostupné z WWW: <<http://en.wikipedia.org/wiki/Cepstrum>>.
- [12] Chiclet keyboard. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 29 January 2004, last modified on 28 April 2011 [cit. 2011-05-02]. Dostupné z WWW: <[http://en.wikipedia.org/wiki/Chiclet\\_keyboard](http://en.wikipedia.org/wiki/Chiclet_keyboard)>.
- [13] Computer keyboard. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 23 August 2001, last modified on 26 April 2011 [cit. 2011-05-02]. Dostupné z WWW: <[http://en.wikipedia.org/wiki/Computer\\_keyboard](http://en.wikipedia.org/wiki/Computer_keyboard)>.
- [14] DŘÍMAL, Martin. *LUPA.CZ* [online]. 2008 [cit. 2011-05-20]. Ochranné programy proti keyloggerům selhávají. Dostupné z WWW: <<http://www.lupa.cz/clanky/ochranne-programy-proti-keyloggerum-selhavaji/>>.
- [15] Dynamic Time Warping. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2 December 2004, last modified on 30 November 2007 [cit. 2011-05-22]. Dostupné z WWW: <[http://en.wikipedia.org/wiki/Dynamic\\_Time\\_Warping](http://en.wikipedia.org/wiki/Dynamic_Time_Warping)>.
- [16] Eavesdropping over Random Passwords via Keyboard Acoustic Emanations. In HALEVI, Tzipora; SAXENA, Nitesh. *Eavesdropping over Random Passwords via Keyboard Acoustic Emanations* [online]. [s.l.] : Cryptology ePrint Archive, 24 Nov 2010, 8 May 2011 [cit. 2011-05-18]. Dostupné z WWW: <<http://eprint.iacr.org/2010/605.pdf>>.
- [17] Endianita. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 13. 4. 2006, last modified on 9. 5. 2011 [cit. 2011-05-18]. Dostupné z WWW: <<http://cs.wikipedia.org/wiki/Endianita>>.
- [18] *FFTW 3.2.2* [online]. 2009 [cit. 2011-05-11]. FFTW User Manual. Dostupné z WWW: <[http://www.fftw.org/fftw3\\_doc/](http://www.fftw.org/fftw3_doc/)>.

- [19] Fourierova transformace. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 14. 2. 2006, last modified on 5. 5. 2011 [cit. 2011-05-22]. Dostupné z WWW: <[http://cs.wikipedia.org/wiki/Fourierova\\_transformace](http://cs.wikipedia.org/wiki/Fourierova_transformace)>.
- [20] Hidden Markov Model. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 3 October 2002, last modified on 1 December 2007 [cit. 2011-05-22]. Dostupné z WWW: <[http://en.wikipedia.org/wiki/Hidden\\_Markov\\_Model](http://en.wikipedia.org/wiki/Hidden_Markov_Model)>.
- [21] Hypervizor. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 13. 11. 2010, last modified on 4. 5. 2011 [cit. 2011-05-20]. Dostupné z WWW: <<http://cs.wikipedia.org/wiki/Hypervizor>>.
- [22] KELLY, Andrew. Cracking Passwords using Keyboard Acoustics and Language Modeling. In *Cracking Passwords using Keyboard Acoustics and Language Modeling* [online]. University of Edinburgh : [s.n.], 2010 [cit. 2011-05-18]. Dostupné z WWW: <<http://www.inf.ed.ac.uk/publications/thesis/online/IM100855.pdf>>.
- [23] Kernel. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 21. 9. 2004, last modified on 18. 5. 2011 [cit. 2011-05-20]. Dostupné z WWW: <<http://cs.wikipedia.org/wiki/Kernel>>.
- [24] Keyboard technology. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 13 September 2001, last modified on 10 April 2011 [cit. 2011-05-02]. Dostupné z WWW: <[http://en.wikipedia.org/wiki/Keyboard\\_technology](http://en.wikipedia.org/wiki/Keyboard_technology)>.
- [25] Keystroke logging#Software-based keyloggers. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 28 February 2011, last modified on 9 May 2011 [cit. 2011-05-11]. Dostupné z WWW: <[http://en.wikipedia.org/wiki/Keystroke\\_logging#Software-based\\_keyloggers](http://en.wikipedia.org/wiki/Keystroke_logging#Software-based_keyloggers)>.
- [26] MALIŠ, Petr. *PrávoIT* [online]. 09.12.2008 [cit. 2011-05-09]. Právní aspekty používání keyloggerů. Dostupné z WWW: <<http://www.pravoit.cz/article/pravni-aspekty-pouzivani-keyloggeru>>.
- [27] OLZAK, Tom. *Adventures in Security* [online]. April 2008 [cit. 2011-05-18]. Keystroke Logging. Dostupné z WWW:

- <[http://adventuresinsecurity.com/images/Keystroke\\_Logging.pdf](http://adventuresinsecurity.com/images/Keystroke_Logging.pdf)>.
- [28] Po%C4%8D%C3%ADta%C4%8Dov%C3%A1kl%C3%A1vesnice. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 4. 7. 2004, last modified on 4. 4. 2011 [cit. 2011-05-02]. Dostupné z WWW: <<http://cs.wikipedia.org/wiki/Po%C4%8D%C3%ADta%C4%8Dov%C3%A1kl%C3%A1vesnice>>.
- [29] Rootkit. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 1. 3. 2006, last modified on 16. 4. 2011 [cit. 2011-05-20]. Dostupné z WWW: <<http://cs.wikipedia.org/wiki/Rootkit>>.
- [30] Rozlo%C5%BEn%C3%AD kl%C3%A1ves. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 3. 2. 2008, last modified on 14. 2. 2011 [cit. 2011-05-02]. Dostupné z WWW: <[http://cs.wikipedia.org/wiki/Rozlo%C5%BEn%C3%AD\\_kl%C3%A1ves](http://cs.wikipedia.org/wiki/Rozlo%C5%BEn%C3%AD_kl%C3%A1ves)>.
- [31] Scissor-switch. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 20 December 2005, last modified on 3 April 2011 [cit. 2011-05-02]. Dostupné z WWW: <<http://en.wikipedia.org/wiki/Scissor-switch>>.
- [32] Technologie kl%C3%A1vesnic. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 22. 5. 2007, last modified on 13. 7. 2010 [cit. 2011-05-02]. Dostupné z WWW: <[http://cs.wikipedia.org/wiki/Technologie\\_kl%C3%A1vesnic](http://cs.wikipedia.org/wiki/Technologie_kl%C3%A1vesnic)>.
- [33] *The PC Guide* [online]. April 17, 2001 [cit. 2011-05-02]. Keyboards. Dostupné z WWW: <<http://www.pcguides.com/ref/kb/index.htm>>.
- [34] Universal Serial Bus. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 27. 9. 2005, last modified on 12. 4. 2011 [cit. 2011-05-10]. Dostupné z WWW: <[http://cs.wikipedia.org/wiki/Universal\\_Serial\\_Bus](http://cs.wikipedia.org/wiki/Universal_Serial_Bus)>.
- [35] Viterbi algorithm. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 16 May 2003, last modified on 14 April 2011 [cit. 2011-05-22]. Dostupné z WWW: <[http://en.wikipedia.org/wiki/Viterbi\\_algorithm](http://en.wikipedia.org/wiki/Viterbi_algorithm)>.
- [36] WILSON, Scott. *Center for Computer Research in Music and Acoustics* [online]. Jan 20, 2003 [cit. 2011-05-18]. WAVE PCM soundfile format . Dostupné z WWW:

<<https://ccrma.stanford.edu/courses/422/projects/WaveFormat/>>.

- [37] ZHUANG, Li; ZHOU, Feng; TYGAR, J.D. Keyboard Acoustic Emanations Revisited. In *Keyboard Acoustic Emanations Revisited*. In Proceedings of the 12th ACM Conference on Computer and Communications Security : University of California, Berkeley, 2008. s. pp. 373-382.

Interní materiály:

- [38] PAK, Hui Shun. *Keyboard Acoustic Triangulation Attack* [online]. Hong Kong, 2006. 43 s. Bakalářská práce. The Chinese university of Hong Kong. Dostupné z WWW:  
<[http://personal.ie.cuhk.edu.hk/~kwwei/FYP/keyboard\\_acoustic\\_attack/Eric\\_Thesis\\_2\\_final.pdf](http://personal.ie.cuhk.edu.hk/~kwwei/FYP/keyboard_acoustic_attack/Eric_Thesis_2_final.pdf)>.
- [39] SKOTNICA, Martin. *Nástroje pro monitorování akcí uživatele : (keyloggers)*. Brno, 2007. 46 s. Diplomová práce. Masarykova Univerzita, Fakulta Informatiky.
- [40] SWIATEK, David. *Metody odsposlouchávání klávesnice*. Zlín, 2010. 51 s. Bakalářská práce. UTB ve Zlíně, Fakulta aplikované informatiky.

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

ASCII	American Standard Code for Information Interchange („americký standardní kód pro výměnu informací“)
USB	Universal Serial Bus („univerzální sériová sběrnice“)
LED	Light-Emitting Diode („dioda emitující světlo“)
DFT	Diskrétní Fourierova transformace
FFT	Rychlá Fourierova transformace
CLR	Common Language Runtime („společné běhové prostředí“)
CLS	Common Language Specification
CTS	Common Type Specification
JIT	Just In Time
COM	Component Object Model
SDK	Software Development Kit
RIFF	Resource Interchange File Format
MSDN	Microsoft Development Network
VMM	Virtual Machine Monitor
API	Application Programming Interface
BIOS	Basic Input Output System
HMM	Hidden Markov Model
DTW	Dynamic Time Warping

**SEZNAM OBRÁZKŮ**

Obr. 1: Stolní model dálnopisu T100 firmy Siemens.....	10
Obr. 2: Jeden z prvních psacích strojů.....	11
Obr. 3: Český psací stroj Consul.....	11
Obr. 4: Jedna z prvních počítačových klávesnic.....	12
Obr. 5: Klávesa z typické počítačové klávesnice.....	13
Obr. 6: Mezery mezi klávesami.....	13
Obr. 7: Druhy kláves.....	14
Obr. 8: Klávesnice PC XT 83-key.....	17
Obr. 9: Klávesnice PC AT 84-key.....	17
Obr. 10: Klávesnice „Enhanced“ 101-key.....	18
Obr. 11: Klávesnice "Windows" 104-key.....	18
Obr. 12: Ergonomická klávesnice Microsoft.....	19
Obr. 13: Jedna z dřívějších notebookových klávesnic.....	19
Obr. 14: Rozložení kláves "Dvorak".....	20
Obr. 15: Ruská klávesnice.....	20
Obr. 16: České programátorské rozložení.....	21
Obr. 17: Mechanický spínač.....	22
Obr. 18: Schéma membránového spínače.....	22
Obr. 19: Schéma spínače s pěnovými prvky.....	23
Obr. 20: Schéma klávesnice se mechanickými spínači s gumovou membránou.....	23
Obr. 21: "Nůžkový" snímač.....	24
Obr. 22: Schéma klávesnice s kapacitním snímačem.....	24
Obr. 23: Zapojení konektoru DIN.....	26
Obr. 24: Zapojení konektoru PS/2.....	26
Obr. 25: Zapojení konektoru USB.....	27
Obr. 26: Schéma umístění kernelu (jádra).....	36
Obr. 27: Příklad keyloggeru pro konektor PS/2.....	39
Obr. 28: Modul keyloggeru před vložením do počítače.....	39
Obr. 29: Datový, hodinový a kompromitující signál při stisku klávesy 0x24 (E na anglické klávesnici).....	42
Obr. 30: Příklady fourierovy transformace.....	46
Obr. 31: Biologický neuron.....	48

Obr. 32: Formální neuron.....	49
Obr. 33: Biologická neuronová síť.....	49
Obr. 34: Síť perceptronů.....	50
Obr. 35: Diagram HMM (x - stavy, y - možné výsledky, a - pravděpodobnosti přechodu mezi stavy, b - pravděpodobnosti výstupů).....	51
Obr. 36: Schéma příkladu pro hledání Viterbiho cesty.....	52
Obr. 37: Mřížkový diagram příkladu pro hledání Viterbiho cesty.....	53
Obr. 38: Asonov, Agrawal, průběh signálu pro stisk klávesy.....	55
Obr. 39: Asonov, Agrawal, spektrum signálu pro stisk klávesy.....	55
Obr. 40: Asonov, Agrawal, postup zpracování signálu.....	55
Obr. 41: Asonov, Agrawal, příklad zpracovaných spekter dvou různých kláves (q a w)....	56
Obr. 42: Asonov, Agrawal, výsledky rozpoznávání 30 různých kláves na klávesnici A.....	56
Obr. 43: Asonov, Agrawal, výsledky rozpoznávání 30 různých kláves na klávesnici B.....	57
Obr. 44: Asonov, Agrawal, výsledky rozpoznávání 30 různých kláves na klávesnici C.....	57
Obr. 45: Asonov, Agrawal, vliv síly stisku na kvalitu rozpoznávání.....	57
Obr. 46: Asonov, Agrawal, vliv stylu psaní (osoby) na kvalitu rozpoznávání.....	58
Obr. 47: Zhuang, Zhou, Tygar, výsledky jednotlivých metod.....	59
Obr. 48: Zhuang, Zhou, Tygar, schéma.....	59
Obr. 49: Zhuang, Zhou, Tygar, průběh signálu stisku klávesy.....	60
Obr. 50: Zhuang, Zhou, Tygar, výstupní energie prvních pěti stisků klávesy.....	60
Obr. 51: Halevi, Saxena, průběh signálu.....	61
Obr. 52: Halevi, Saxena, průběh signálu pro stisk několika kláves za sebou.....	61
Obr. 53: Halevi, Saxena, součet koeficientů FFT pro stisk několika kláves za sebou.....	62
Obr. 54: Kelly, spectrogram signálu stisku několika kláves.....	62
Obr. 55: Kelly, průběh signálu pro překrývající se stisk kláves.....	63
Obr. 56: Pak, schéma metody.....	63
Obr. 57: Výsledky testů antikeyloggerů.....	64
Obr. 58: Uspořádání .NET Frameworks.....	67
Obr. 59: Dostupnost verzí .NET Frameworku pro operační systémy Windows.....	68
Obr. 60: Formát WAV souboru.....	69
Obr. 61: Vzorový WAV soubor.....	70
Obr. 62: Uložení čísla 0x4A3B2C1D na adresu 100 jako little-endian.....	71
Obr. 63: Uložení čísla 0x4A3B2C1D na adresu 100 jako little-endian.....	71
Obr. 64: Příklad uložení čísla 0x4A3B2C1D ve formátu middle-endian na adrese 100.....	71

Obr. 65: Vytvoření knihovny v prostředí Microsoft Visual C# 2010 Express.....	72
Obr. 66: Přidání knihovny DirectSound k projektu.....	72
Obr. 67: Výběr cílového .NET Frameworku.....	73
Obr. 68: Vypnutí "Loaderlocku".....	73
Obr. 69: Princip kruhového bufferu.....	74
Obr. 70: Vytvoření projektu.....	74
Obr. 71: Okno programu pro sběr dat.....	75
Obr. 72: Okno programu pro rozpoznání (analýzu) dat.....	77
Obr. 73: Průběh signálu klávesy "Space" na desktopové klávesnici.....	81
Obr. 74: Průběh signálu klávesy "Space" na notebookové klávesnici.....	82
Obr. 75: Spektrum signálu klávesy "Space" desktopové klávesnice.....	82
Obr. 76: Spektrum signálu klávesy "Space" notebookové klávesnice.....	82
Obr. 77: Průběh signálu pro klávesu „Space“ a pro testovanou osobu č. 2.....	83
Obr. 78: Spektrum signálu pro klávesu "Space" a pro testovanou osobu č. 2.....	83
Obr. 79: Průběh signálu pro klávesu „Space“ a pro testovanou osobu č. 3.....	84
Obr. 80: Spektrum signálu pro klávesu "Space" a pro testovanou osobu č. 3.....	84
Obr. 81: Průběh signálu pro klávesu „Space“ a pro externí mikrofon.....	85
Obr. 82: Spektrum signálu pro klávesu "Space" a pro externí mikrofon.....	85
Obr. 83: Průběh signálu pro klávesu „Space“ a mikrofon vlevo.....	86
Obr. 84: Spektrum signálu pro klávesu "Space" a mikrofon vlevo.....	86
Obr. 85: Průběh signálu pro klávesu „Space“ a mikrofon vepředu.....	86
Obr. 86: Spektrum signálu pro klávesu "Space" a mikrofon vepředu.....	87
Obr. 87: Průběh signálu pro klávesu „Space“ a mikrofon vpravo.....	87
Obr. 88: Spektrum signálu pro klávesu "Space" a mikrofon vpravo.....	87
Obr. 89: Průběh signálu pro klávesu „Space“ a mikrofon za.....	88
Obr. 90: Spektrum signálu pro klávesu "Space" a mikrofon za.....	88
Obr. 91: Průběh signálu pro klávesu „Space“ a pro testované prostředí č. 2.....	89
Obr. 92: Spektrum signálu pro klávesu "Space" a pro testované prostředí č. 2.....	89
Obr. 93: Průběh signálu pro klávesu „Space“ a pro testované prostředí č. 3.....	89
Obr. 94: Spektrum signálu pro klávesu "Space" a pro testované prostředí č. 3.....	90

**SEZNAM TABULEK**

Tab. 1 Přehled vlastností jednotlivých typů spínačů.....	25
Tab. 2: Přehled účinnosti metod.....	44
Tab. 3 Formát WAV souboru.....	70

**SEZNAM PŘÍLOH**

Příloha P 1: KOMPLETNÍ VÝPIS ZDROJOVÉHO KÓDU.

Příloha P 2: PŘEHLED SCAN KÓDŮ KLÁVESNICE.

## PŘÍLOHA P 1: KOMPLETNÍ VÝPIS ZDROJÓVÉHO KÓDU.

### DXDSAR

#### WaveFileWriter.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Text;

namespace DXDSAR
{
    /// <summary>
    /// Direct-X Direct Sound Audio Recorder (DXDSAR).
    ///
    /// WaveFileWriter zapíše nahrané data z mikrofónu do wav souboru (.wav).
    /// </summary>

    internal class WaveFileWriter : IDisposable
    {
        #region Private Fields

        private string fileName;
        private short format;
        private short channels;
        private int sampleRate;
        private int byteRate;
        private short blockAlign;
        private short bitPerSample;
        private byte[] data;

        #endregion

        /// <summary>Vytvoří objekt, který zapíše data do wav souboru
        (.wav).</summary>
        /// <param name="fileName">Název souboru</param>
        /// <param name="format">Audio formát</param>
        /// <param name="channels">Počet kanálů</param>
        /// <param name="sampleRate">Vzorkovací frekvence</param>
        /// <param name="byteRate">byte rate</param>
        /// <param name="blockAlign">Zarovnání bloku</param>
        /// <param name="bitPerSample">Počet bitů na vzorek</param>
        /// <param name="data">Data, která ukládáme do souboru</param>
        public WaveFileWriter(string fileName, short format, short channels, int
sampleRate, int byteRate, short blockAlign, short bitPerSample, byte[] data)
        {
            this.fileName = fileName;
            this.format = format;
            this.channels = channels;
            this.sampleRate = sampleRate;
            this.byteRate = byteRate;
            this.blockAlign = blockAlign;
            this.bitPerSample = bitPerSample;
            this.data = data;
        } // end constructor

        /// <summary>Zruší všechny prostředky použité objektem</summary>
        public void Dispose()
        {
            data = null;
        }
    }
}
```

```

        fileName = String.Empty;
    } // end Dispose

    /// <summary>Zapiše zvuková data do wav souboru (.wav).</summary>
    /// <returns>Vrací true, pokud jsou zapsaná, false pokud zápis
selhal</returns>
    public bool WriteSoundToWaveFile()
    {
        int dataLength = data.Length;
        byte[] header = new byte[44];

        MemoryStream ms = new MemoryStream(header, 0, 44, true);
        UTF8Encoding utf8 = new UTF8Encoding();
        FileStream fs;

        //zapišeme hlavičku
        ms.Write(utf8.GetBytes("RIFF"), 0, 4);
        ms.Write(BitConverter.GetBytes(dataLength + 36), 0, 4);
        ms.Write(utf8.GetBytes("WAVE"), 0, 4);
        ms.Write(utf8.GetBytes("fmt "), 0, 4);
        ms.Write(BitConverter.GetBytes(16), 0, 4);
        ms.Write(BitConverter.GetBytes(format), 0, 2);
        ms.Write(BitConverter.GetBytes(channels), 0, 2);
        ms.Write(BitConverter.GetBytes(sampleRate), 0, 4);
        ms.Write(BitConverter.GetBytes(byteRate), 0, 4);
        ms.Write(BitConverter.GetBytes(blockAlign), 0, 2);
        ms.Write(BitConverter.GetBytes(bitPerSample), 0, 2);
        ms.Write(utf8.GetBytes("data"), 0, 4);
        ms.Write(BitConverter.GetBytes(dataLength), 0, 4);

        try
        {
            //zkusím vytvořit soubor
            fs = new FileStream(fileName, FileMode.Create);
            //zapišu hlavičku
            fs.Write(header, 0, header.Length);
            //zapišu data
            fs.Write(data, 0, dataLength);
            //zavřu soubor
            fs.Close();
            fs.Dispose();
            //zruším data v paměti
            ms.Close();
            ms.Dispose();

            return true;
        }
        catch (Exception)
        {
            return false;
        }
    } // end WriteSoundToWaveFile
} // end class
} // end namespace

```

### SoundCapture.cs

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading;
using Microsoft.DirectX.DirectSound;

```

```

namespace DXDSAR
{
    /// <summary>
    /// Direct-X Direct Sound Audio Recorder (DXDSAR).
    ///
    /// SoundCapture inicializuje nahrávací zařízení a nahraje zvuk z mikrofónu.
    /// </summary>
    public sealed class SoundCapture : IDisposable
    {
        //deklarace privátních proměnných
        #region Private Fields

        private Capture capture;
        private CaptureBuffer captureBuffer;
        private CaptureBufferDescription captureBufferDesc;
        private WaveFormat waveFormat;
        private BufferPositionNotify[] notifyPositions;
        private Notify notify;
        private AutoResetEvent notifyEvent;
        private Thread notifyThread;
        private int readPosition = 0;
        private bool isRecording = false;
        private List<byte[]> rawSoundData;
        private WaveFileWriter wfw;

        #endregion

        //deklarace veřejných vlastností
        #region Public Properties

        /// <summary>Vzorkovací frekvence v Hz.</summary>
        public int SamplingRate { get { return waveFormat.SamplesPerSecond; } }
        /// <summary>Počet bitů na vzorek. Typicky 8 bitů nebo 16 bitů.</summary>
        public short Bit { get { return waveFormat.BitsPerSample; } }
        /// <summary>Počet kanálů, které se nahrávají. Typicky MONO nebo
STEREO.</summary>
        public short Channels { get { return waveFormat.Channels; } }
        /// <summary>Bit rate (v kbps).</summary>
        public int BitRate { get { return waveFormat.AverageBytesPerSecond /
1024; } }

        /// <summary>Wav formát použitý pro nahrávání.</summary>
        public WaveFormat WavFormat { get { return waveFormat; } }
        //konstanty
        /// <summary>Vzorkovací frekvence 11025 Hz.</summary>
        public const int _11025_HZ = 11025;
        /// <summary>Vzorkovací frekvence 22050 Hz.</summary>
        public const int _22050_HZ = 22050;
        /// <summary>Vzorkovací frekvence 44100 Hz.</summary>
        public const int _44100_HZ = 44100;
        /// <summary>Vzorkovací frekvence 96000 Hz.</summary>
        public const int _96000_HZ = 96000;
        /// <summary>8 bit</summary>
        public const int _8_Bit = 8;
        /// <summary>16 bit</summary>
        public const int _16_Bit = 16;
        /// <summary>Mono</summary>
        public const int _MONO = 1;
        /// <summary>Stereo</summary>
        public const int _STEREO = 2;

        #endregion
    }
}

```

```

public delegate void Complete(object sender, EventArgs e);
/// <summary>Nastane pokud je soubor uložený na disk.</summary>
public event Complete OnWavComplete;

#region Constructors
/// <summary>Založí nahrávací zařízení s příslušným formátem.</summary>
/// <param name="samplingRate">Vzorkovací frekvence</param>
/// <param name="bitsPerSample">Počet bitů</param>
/// <param name="channels">Počet kanálů</param>

channels) public SoundCapture(int samplingRate, short bitsPerSample, short
{
    //vytvoříme wav formát
    waveFormat = new WaveFormat();
    waveFormat.SamplesPerSecond = samplingRate;
    waveFormat.BitsPerSample = bitsPerSample;
    waveFormat.Channels = channels;
    waveFormat.FormatTag = WaveFormatTag.Pcm;
    waveFormat.BlockAlign = (short)(waveFormat.Channels *
(waveFormat.BitsPerSample / 8));
    waveFormat.AverageBytesPerSecond = waveFormat.SamplesPerSecond *
waveFormat.BlockAlign;
    captureBufferDesc = new CaptureBufferDescription();
    captureBufferDesc.BufferBytes = waveFormat.AverageBytesPerSecond;
    captureBufferDesc.ControlEffects = false;
    captureBufferDesc.Format = waveFormat;
    captureBufferDesc.WaveMapped = true;
    //vytvoříme nahrávací zařízení
    capture = new Capture();
    //vytvoříme nahrávací buffer
    captureBuffer = new CaptureBuffer(captureBufferDesc, capture);
    //vytvoříme oblast dat v paměti, kam budeme ukládat nahrané data
    rawSoundData = new List<byte[]>();
    //nastavíme upozorňování, jakmile se nám zaplní buffer
    SetCaptureNotifications();
} // end constructor

#endregion
#region Private Methods

private void NotifyThreadWait()
{
    //dokud se má nahrávat, tak nahráváme
    do
    {
        notifyEvent.WaitOne(Timeout.Infinite, false);
        RecordSoundData();
    } while (isRecording);
} // end NotifyThreadWait

private void OnWavFileComplete(EventArgs e)
{
    if (OnWavComplete != null)
        OnWavComplete(this, e);
} // end OnWavFileComplete

private void RecordSoundData()
{
    int size = waveFormat.AverageBytesPerSecond /
waveFormat.BitsPerSample;

```

```

        byte[] soundData = new byte[size];

        captureBuffer.Read(readPosition, new MemoryStream(soundData), size,
LockFlag.None);
        readPosition += size;

        if (readPosition >= waveFormat.AverageBytesPerSecond - 1)
readPosition = 0;

        rawSoundData.Add(soundData);
        soundData = null;
    } // end RecordSoundData

    private void SetCaptureNotifications()
    {
        int blockSize = waveFormat.AverageBytesPerSecond /
waveFormat.BitsPerSample;

        notifyPositions = new BufferPositionNotify[waveFormat.BitsPerSample];
        notifyEvent = new AutoResetEvent(false);
        IntPtr notifyHandle =
notifyEvent.SafeWaitHandle.DangerousGetHandle();

        for (int a = 0; a < waveFormat.BitsPerSample; a++)
        {
            notifyPositions[a] = new BufferPositionNotify();
            notifyPositions[a].Offset = (blockSize * (a + 1)) - 1;
            notifyPositions[a].EventNotifyHandle = notifyHandle;
        }

        notifyThread = new Thread(new ThreadStart(NotifyThreadWait));
        notify = new Notify(captureBuffer);
        notify.SetNotificationPositions(notifyPositions,
waveFormat.BitsPerSample);
        notifyThread.Start();
    } // end SetCaptureNotifications

#endregion

#region Public Methods
    /// <summary>Uvolní všechny zdroje, které jsou používány
objektem</summary>
    public void Dispose()
    {
        if (capture != null)
        {
            capture.Dispose();
            capture = null;
        }

        if (captureBuffer != null)
        {
            captureBuffer.Dispose();
            captureBuffer = null;
        }

        if (notify != null)
        {
            notify.Dispose();
            notify = null;
        }
    }

```

```

        if (notifyEvent != null)
        {
            notifyEvent.Close();
            notifyEvent = null;
        }

        notifyPositions = null;

        if (notifyThread.ThreadState != ThreadState.Aborted)
        {
            notifyThread.Abort();
        }

        rawSoundData.Clear();
        rawSoundData = null;
        OnWavComplete = null;
    }

    /// <summary>Zkopíruje data po skončení nahrávání</summary>
    public byte[] GetSoundData()
    {
        List<byte> data = new List<byte>();

        for (int a = 0; a < rawSoundData.Count; a++)
        {
            for (int b = 0; b < rawSoundData[a].Length; b++)
            {
                data.Add(rawSoundData[a].ElementAt<byte>(b));
            }
        }

        return data.ToArray();
    } // end GetSoundData

    /// <summary>Začne nahrávání</summary>
    public void StartRecording()
    {
        isRecording = true;
        captureBuffer.Start(true);
    } // end StartRecording

    /// <summary>Zastaví nahrávání</summary>
    public void StopRecording()
    {
        isRecording = false;
        captureBuffer.Stop();
        RecordSoundData();
    } // end StopRecording

    /// <summary>Zapíše nahrané data do wav souboru</summary>

    /// <param name="fileName">Název souboru</param>

    public void WriteWaveFile(string fileName)
    {
        if (!fileName.Contains('\\') && !fileName.ToLower().EndsWith(".wav"))
            throw new ArgumentException("File name must be absolute path.
Example: C:\\Documents and Settings\\Users\\My Music\\Example.wav", "fileName");

        wfw = new WaveFileWriter(fileName, 1, waveFormat.Channels,
waveFormat.SamplesPerSecond, waveFormat.AverageBytesPerSecond, waveFormat.BlockAlign,
waveFormat.BitsPerSample, GetSoundData()); wfw.WriteSoundToWaveFile();

```

```

        OnWavComplete(this, EventArgs.Empty);
    } // end WriteWaveFile

    #endregion
} // end class
} // end namespace

```

## Diplomka

### Program.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace Diplomka
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            //Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}

```

### Form1.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using DXDSAR;
using Microsoft.DirectX.DirectSound;
using System.IO;

namespace Diplomka
{
    public partial class Form1 : Form
    {
        //nahraná data
        private List<byte[]> rawSoundData;
        string fileName;
        bool isRecording = false;
        byte button;
        SoundCapture sc;
        byte ticks;

        public Form1()
        {
            InitializeComponent();
        }
    }
}

```

```
private void Form1_Load(object sender, EventArgs e)
{
    ticks = 0;
    Buttons();
}

private void Buttons()
{
    if (File.Exists("a.wav"))
        button1.ForeColor = Color.Blue;
    if (File.Exists("b.wav"))
        button2.ForeColor = Color.Blue;
    if (File.Exists("c.wav"))
        button3.ForeColor = Color.Blue;
    if (File.Exists("d.wav"))
        button4.ForeColor = Color.Blue;
    if (File.Exists("e.wav"))
        button5.ForeColor = Color.Blue;
    if (File.Exists("f.wav"))
        button6.ForeColor = Color.Blue;
    if (File.Exists("g.wav"))
        button7.ForeColor = Color.Blue;
    if (File.Exists("h.wav"))
        button8.ForeColor = Color.Blue;
    if (File.Exists("i.wav"))
        button9.ForeColor = Color.Blue;
    if (File.Exists("j.wav"))
        button10.ForeColor = Color.Blue;
    if (File.Exists("k.wav"))
        button11.ForeColor = Color.Blue;
    if (File.Exists("l.wav"))
        button12.ForeColor = Color.Blue;
    if (File.Exists("m.wav"))
        button13.ForeColor = Color.Blue;
    if (File.Exists("n.wav"))
        button14.ForeColor = Color.Blue;
    if (File.Exists("o.wav"))
        button15.ForeColor = Color.Blue;
    if (File.Exists("p.wav"))
        button16.ForeColor = Color.Blue;
    if (File.Exists("q.wav"))
        button17.ForeColor = Color.Blue;
    if (File.Exists("r.wav"))
        button18.ForeColor = Color.Blue;
    if (File.Exists("s.wav"))
        button19.ForeColor = Color.Blue;
    if (File.Exists("t.wav"))
        button20.ForeColor = Color.Blue;
    if (File.Exists("u.wav"))
        button21.ForeColor = Color.Blue;
    if (File.Exists("v.wav"))
        button22.ForeColor = Color.Blue;
    if (File.Exists("w.wav"))
        button23.ForeColor = Color.Blue;
    if (File.Exists("x.wav"))
        button24.ForeColor = Color.Blue;
    if (File.Exists("y.wav"))
        button25.ForeColor = Color.Blue;
    if (File.Exists("z.wav"))
        button26.ForeColor = Color.Blue;
    if (File.Exists("esc.wav"))
```

```
        button27.ForeColor = Color.Blue;
    if (File.Exists("f1.wav"))
        button28.ForeColor = Color.Blue;
    if (File.Exists("f2.wav"))
        button29.ForeColor = Color.Blue;
    if (File.Exists("f3.wav"))
        button30.ForeColor = Color.Blue;
    if (File.Exists("f4.wav"))
        button31.ForeColor = Color.Blue;
    if (File.Exists("f5.wav"))
        button32.ForeColor = Color.Blue;
    if (File.Exists("f6.wav"))
        button33.ForeColor = Color.Blue;
    if (File.Exists("f7.wav"))
        button34.ForeColor = Color.Blue;
    if (File.Exists("f8.wav"))
        button35.ForeColor = Color.Blue;
    if (File.Exists("f9.wav"))
        button36.ForeColor = Color.Blue;
    if (File.Exists("f10.wav"))
        button37.ForeColor = Color.Blue;
    if (File.Exists("f11.wav"))
        button38.ForeColor = Color.Blue;
    if (File.Exists("f12.wav"))
        button39.ForeColor = Color.Blue;
    if (File.Exists("scrolllock.wav"))
        button40.ForeColor = Color.Blue;
    if (File.Exists("pause.wav"))
        button41.ForeColor = Color.Blue;
    if (File.Exists("insert.wav"))
        button42.ForeColor = Color.Blue;
    if (File.Exists("home.wav"))
        button43.ForeColor = Color.Blue;
    if (File.Exists("pageup.wav"))
        button44.ForeColor = Color.Blue;
    if (File.Exists("delete.wav"))
        button45.ForeColor = Color.Blue;
    if (File.Exists("end.wav"))
        button46.ForeColor = Color.Blue;
    if (File.Exists("pagedown.wav"))
        button47.ForeColor = Color.Blue;
    if (File.Exists("left.wav"))
        button48.ForeColor = Color.Blue;
    if (File.Exists("up.wav"))
        button49.ForeColor = Color.Blue;
    if (File.Exists("down.wav"))
        button50.ForeColor = Color.Blue;
    if (File.Exists("right.wav"))
        button51.ForeColor = Color.Blue;
    if (File.Exists("numlock.wav"))
        button52.ForeColor = Color.Blue;
    if (File.Exists("numlomitko.wav"))
        button53.ForeColor = Color.Blue;
    if (File.Exists("numhvezdicka.wav"))
        button54.ForeColor = Color.Blue;
    if (File.Exists("numminus.wav"))
        button55.ForeColor = Color.Blue;
    if (File.Exists("num7.wav"))
        button56.ForeColor = Color.Blue;
    if (File.Exists("num8.wav"))
        button57.ForeColor = Color.Blue;
    if (File.Exists("num9.wav"))
```

```
        button58.ForeColor = Color.Blue;
    if (File.Exists("numplus.wav"))
        button59.ForeColor = Color.Blue;
    if (File.Exists("num4.wav"))
        button60.ForeColor = Color.Blue;
    if (File.Exists("num5.wav"))
        button61.ForeColor = Color.Blue;
    if (File.Exists("num6.wav"))
        button62.ForeColor = Color.Blue;
    if (File.Exists("num1.wav"))
        button63.ForeColor = Color.Blue;
    if (File.Exists("num2.wav"))
        button64.ForeColor = Color.Blue;
    if (File.Exists("num3.wav"))
        button65.ForeColor = Color.Blue;
    if (File.Exists("numenter.wav"))
        button66.ForeColor = Color.Blue;
    if (File.Exists("num0.wav"))
        button67.ForeColor = Color.Blue;
    if (File.Exists("numtecka.wav"))
        button68.ForeColor = Color.Blue;
    if (File.Exists("tilda.wav"))
        button69.ForeColor = Color.Blue;
    if (File.Exists("1.wav"))
        button70.ForeColor = Color.Blue;
    if (File.Exists("2.wav"))
        button71.ForeColor = Color.Blue;
    if (File.Exists("3.wav"))
        button72.ForeColor = Color.Blue;
    if (File.Exists("4.wav"))
        button73.ForeColor = Color.Blue;
    if (File.Exists("5.wav"))
        button74.ForeColor = Color.Blue;
    if (File.Exists("6.wav"))
        button75.ForeColor = Color.Blue;
    if (File.Exists("7.wav"))
        button76.ForeColor = Color.Blue;
    if (File.Exists("8.wav"))
        button77.ForeColor = Color.Blue;
    if (File.Exists("9.wav"))
        button78.ForeColor = Color.Blue;
    if (File.Exists("0.wav"))
        button79.ForeColor = Color.Blue;
    if (File.Exists("minus.wav"))
        button80.ForeColor = Color.Blue;
    if (File.Exists("rovnitko.wav"))
        button81.ForeColor = Color.Blue;
    if (File.Exists("backspace.wav"))
        button82.ForeColor = Color.Blue;
    if (File.Exists("tab.wav"))
        button83.ForeColor = Color.Blue;
    if (File.Exists("printscreen.wav"))
        button84.ForeColor = Color.Blue;
    if (File.Exists("capslock.wav"))
        button85.ForeColor = Color.Blue;
    if (File.Exists("leftshift.wav"))
        button86.ForeColor = Color.Blue;
    if (File.Exists("leftcontrol.wav"))
        button87.ForeColor = Color.Blue;
    if (File.Exists("leftwin.wav"))
        button88.ForeColor = Color.Blue;
    if (File.Exists("leftalt.wav"))
```

```

        button89.ForeColor = Color.Blue;
    if (File.Exists("rightalt.wav"))
        button90.ForeColor = Color.Blue;
    if (File.Exists("rightcontrol.wav"))
        button91.ForeColor = Color.Blue;
    if (File.Exists("rightwin.wav"))
        button92.ForeColor = Color.Blue;
    if (File.Exists("menu.wav"))
        button93.ForeColor = Color.Blue;
    if (File.Exists("space.wav"))
        button94.ForeColor = Color.Blue;
    if (File.Exists("levahranata.wav"))
        button95.ForeColor = Color.Blue;
    if (File.Exists("pravahranata.wav"))
        button96.ForeColor = Color.Blue;
    if (File.Exists("backslash.wav"))
        button97.ForeColor = Color.Blue;
    if (File.Exists("strednik.wav"))
        button98.ForeColor = Color.Blue;
    if (File.Exists("apostrof.wav"))
        button99.ForeColor = Color.Blue;
    if (File.Exists("enter.wav"))
        button100.ForeColor = Color.Blue;
    if (File.Exists("carka.wav"))
        button101.ForeColor = Color.Blue;
    if (File.Exists("tecka.wav"))
        button102.ForeColor = Color.Blue;
    if (File.Exists("lomitko.wav"))
        button103.ForeColor = Color.Blue;
    if (File.Exists("rightshift.wav"))
        button104.ForeColor = Color.Blue;
}

private void button1_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button1.Enabled = false;
        fileName = "a.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 1;
    }
}

private void button2_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button2.Enabled = false;
        fileName = "b.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 2;
    }
}

```

```

    }
}

private void button3_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button3.Enabled = false;
        fileName = "c.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 3;
    }
}

private void button4_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button4.Enabled = false;
        fileName = "d.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 4;
    }
}

private void button5_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button5.Enabled = false;
        fileName = "e.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 5;
    }
}

private void button6_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button6.Enabled = false;
        fileName = "f.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
    }
}

```

```

        isRecording = true;
        button = 6;
    }
}

private void button7_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button7.Enabled = false;
        fileName = "g.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 7;
    }
}

private void button8_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button8.Enabled = false;
        fileName = "h.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 8;
    }
}

private void button9_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button9.Enabled = false;
        fileName = "i.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 9;
    }
}

private void button10_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button10.Enabled = false;
        fileName = "j.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);

```

```

        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 10;
    }
}

private void button11_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button11.Enabled = false;
        fileName = "k.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 11;
    }
}

private void button12_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button12.Enabled = false;
        fileName = "l.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 12;
    }
}

private void button13_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button13.Enabled = false;
        fileName = "m.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 13;
    }
}

private void button14_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button14.Enabled = false;
        fileName = "n.wav";
    }
}

```

```

        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 14;
    }
}

private void button15_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button15.Enabled = false;
        fileName = "o.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 15;
    }
}

private void button16_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button16.Enabled = false;
        fileName = "p.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 16;
    }
}

private void button17_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button17.Enabled = false;
        fileName = "q.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 17;
    }
}

private void button18_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();

```

```

        button18.Enabled = false;
        fileName = "r.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 18;
    }
}

private void button19_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button19.Enabled = false;
        fileName = "s.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 19;
    }
}

private void button20_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button20.Enabled = false;
        fileName = "t.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 20;
    }
}

private void button21_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button21.Enabled = false;
        fileName = "u.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 21;
    }
}

private void button22_Click(object sender, EventArgs e)
{
    if (!isRecording)

```

```

        {
            timer1.Start();
            button22.Enabled = false;
            fileName = "v.wav";
            sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
            sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
            sc.StartRecording();
            isRecording = true;
            button = 22;
        }
    }

    private void button23_Click(object sender, EventArgs e)
    {
        if (!isRecording)
        {
            timer1.Start();
            button23.Enabled = false;
            fileName = "w.wav";
            sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
            sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
            sc.StartRecording();
            isRecording = true;
            button = 23;
        }
    }

    private void button24_Click(object sender, EventArgs e)
    {
        if (!isRecording)
        {
            timer1.Start();
            button24.Enabled = false;
            fileName = "x.wav";
            sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
            sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
            sc.StartRecording();
            isRecording = true;
            button = 24;
        }
    }

    private void button25_Click(object sender, EventArgs e)
    {
        if (!isRecording)
        {
            timer1.Start();
            button25.Enabled = false;
            fileName = "y.wav";
            sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
            sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
            sc.StartRecording();
            isRecording = true;
            button = 25;
        }
    }

    private void button26_Click(object sender, EventArgs e)

```

```

{
    if (!isRecording)
    {
        timer1.Start();
        button26.Enabled = false;
        fileName = "z.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 26;
    }
}

private void button27_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button27.Enabled = false;
        fileName = "esc.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 27;
    }
}

private void button28_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button28.Enabled = false;
        fileName = "f1.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 28;
    }
}

private void button29_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button29.Enabled = false;
        fileName = "f2.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 29;
    }
}

```

```

private void button30_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button30.Enabled = false;
        fileName = "f3.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 30;
    }
}

private void button31_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button31.Enabled = false;
        fileName = "f4.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 31;
    }
}

private void button32_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button32.Enabled = false;
        fileName = "f5.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 32;
    }
}

private void button33_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button33.Enabled = false;
        fileName = "f6.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 33;
    }
}

```

```

    }
}

private void button34_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button34.Enabled = false;
        fileName = "f7.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 34;
    }
}

private void button35_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button35.Enabled = false;
        fileName = "f8.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 35;
    }
}

private void button36_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button36.Enabled = false;
        fileName = "f9.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 36;
    }
}

private void button37_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button37.Enabled = false;
        fileName = "f10.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
    }
}

```

```

        isRecording = true;
        button = 37;
    }
}

private void button38_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button38.Enabled = false;
        fileName = "f11.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 38;
    }
}

private void button39_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button39.Enabled = false;
        fileName = "f12.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 39;
    }
}

private void button40_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button40.Enabled = false;
        fileName = "scrolllock.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 40;
    }
}

private void button41_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button41.Enabled = false;
        fileName = "pause.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);

```

```

        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 41;
    }
}

private void button42_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button42.Enabled = false;
        fileName = "insert.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 42;
    }
}

private void button43_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button43.Enabled = false;
        fileName = "home.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 43;
    }
}

private void button44_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button44.Enabled = false;
        fileName = "pageup.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 44;
    }
}

private void button45_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button45.Enabled = false;
        fileName = "delete.wav";
    }
}

```

```

        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 45;
    }
}

private void button46_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button46.Enabled = false;
        fileName = "end.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 46;
    }
}

private void button47_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button47.Enabled = false;
        fileName = "pagedown.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 47;
    }
}

private void button48_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button48.Enabled = false;
        fileName = "left.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 48;
    }
}

private void button49_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();

```

```

        button49.Enabled = false;
        fileName = "up.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 49;
    }
}

private void button50_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button50.Enabled = false;
        fileName = "down.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 50;
    }
}

private void button51_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button51.Enabled = false;
        fileName = "right.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 51;
    }
}

private void button52_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button52.Enabled = false;
        fileName = "numlock.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 52;
    }
}

private void button53_Click(object sender, EventArgs e)
{
    if (!isRecording)

```

```

        {
            timer1.Start();
            button53.Enabled = false;
            fileName = "numlomitko.wav";
            sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
            sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
            sc.StartRecording();
            isRecording = true;
            button = 53;
        }
    }

    private void button54_Click(object sender, EventArgs e)
    {
        if (!isRecording)
        {
            timer1.Start();
            button54.Enabled = false;
            fileName = "numhvezdicka.wav";
            sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
            sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
            sc.StartRecording();
            isRecording = true;
            button = 54;
        }
    }

    private void button55_Click(object sender, EventArgs e)
    {
        if (!isRecording)
        {
            timer1.Start();
            button55.Enabled = false;
            fileName = "numminus.wav";
            sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
            sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
            sc.StartRecording();
            isRecording = true;
            button = 55;
        }
    }

    private void button56_Click(object sender, EventArgs e)
    {
        if (!isRecording)
        {
            timer1.Start();
            button56.Enabled = false;
            fileName = "num7.wav";
            sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
            sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
            sc.StartRecording();
            isRecording = true;
            button = 56;
        }
    }

    private void button57_Click(object sender, EventArgs e)

```

```

    {
        if (!isRecording)
        {
            timer1.Start();
            button57.Enabled = false;
            fileName = "num8.wav";
            sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
            sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
            sc.StartRecording();
            isRecording = true;
            button = 57;
        }
    }

private void button58_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button58.Enabled = false;
        fileName = "num9.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 58;
    }
}

private void button59_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button59.Enabled = false;
        fileName = "numplus.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 59;
    }
}

private void button60_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button60.Enabled = false;
        fileName = "num4.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 60;
    }
}

```

```

private void button61_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button61.Enabled = false;
        fileName = "num5.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 61;
    }
}

private void button62_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button62.Enabled = false;
        fileName = "num6.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 62;
    }
}

private void button63_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button63.Enabled = false;
        fileName = "num1.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 63;
    }
}

private void button64_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button64.Enabled = false;
        fileName = "num2.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 64;
    }
}

```

```

    }
}

private void button65_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button65.Enabled = false;
        fileName = "num3.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 65;
    }
}

private void button66_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button66.Enabled = false;
        fileName = "numenter.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 66;
    }
}

private void button67_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button67.Enabled = false;
        fileName = "num0.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 67;
    }
}

private void button68_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button68.Enabled = false;
        fileName = "numtecka.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
    }
}

```

```

        isRecording = true;
        button = 68;
    }
}

private void button69_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button69.Enabled = false;
        fileName = "tilda.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 69;
    }
}

private void button70_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button70.Enabled = false;
        fileName = "1.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 70;
    }
}

private void button71_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button71.Enabled = false;
        fileName = "2.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 71;
    }
}

private void button72_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button72.Enabled = false;
        fileName = "3.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,

```

```

SoundCapture._MONO);
    sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
    sc.StartRecording();
    isRecording = true;
    button = 72;
}
}

private void button73_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button73.Enabled = false;
        fileName = "4.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 73;
    }
}

private void button74_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button74.Enabled = false;
        fileName = "5.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 74;
    }
}

private void button75_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button75.Enabled = false;
        fileName = "6.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 75;
    }
}

private void button76_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button76.Enabled = false;

```

```

        fileName = "7.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 76;
    }
}

private void button77_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button77.Enabled = false;
        fileName = "8.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 77;
    }
}

private void button78_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button78.Enabled = false;
        fileName = "9.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 78;
    }
}

private void button79_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button79.Enabled = false;
        fileName = "0.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 79;
    }
}

private void button80_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {

```

```

        timer1.Start();
        button80.Enabled = false;
        fileName = "minus.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 80;
    }
}

private void button81_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button81.Enabled = false;
        fileName = "rovnitko.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 81;
    }
}

private void button82_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button82.Enabled = false;
        fileName = "backspace.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 82;
    }
}

private void button83_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button83.Enabled = false;
        fileName = "tab.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 83;
    }
}

private void button84_Click(object sender, EventArgs e)
{

```

```

        if (!isRecording)
        {
            timer1.Start();
            button84.Enabled = false;
            fileName = "printscreen.wav";
            sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
            sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
            sc.StartRecording();
            isRecording = true;
            button = 84;
        }
    }

    private void button85_Click(object sender, EventArgs e)
    {
        if (!isRecording)
        {
            timer1.Start();
            button85.Enabled = false;
            fileName = "capslock.wav";
            sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
            sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
            sc.StartRecording();
            isRecording = true;
            button = 85;
        }
    }

    private void button86_Click(object sender, EventArgs e)
    {
        if (!isRecording)
        {
            timer1.Start();
            button86.Enabled = false;
            fileName = "leftshift.wav";
            sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
            sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
            sc.StartRecording();
            isRecording = true;
            button = 86;
        }
    }

    private void button87_Click(object sender, EventArgs e)
    {
        if (!isRecording)
        {
            timer1.Start();
            button87.Enabled = false;
            fileName = "leftcontrol.wav";
            sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
            sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
            sc.StartRecording();
            isRecording = true;
            button = 87;
        }
    }
}

```

```

private void button88_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button88.Enabled = false;
        fileName = "leftwin.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 88;
    }
}

private void button89_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button89.Enabled = false;
        fileName = "leftalt.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 89;
    }
}

private void button90_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button90.Enabled = false;
        fileName = "rightalt.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 90;
    }
}

private void button91_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button91.Enabled = false;
        fileName = "rightcontrol.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 91;
    }
}

```

```

}

private void button92_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button92.Enabled = false;
        fileName = "rightwin.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 92;
    }
}

private void button93_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button93.Enabled = false;
        fileName = "menu.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 93;
    }
}

private void button94_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button94.Enabled = false;
        fileName = "space.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 94;
    }
}

private void button95_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button95.Enabled = false;
        fileName = "levahranata.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
    }
}

```

```

        button = 95;
    }
}

private void button96_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button96.Enabled = false;
        fileName = "pravahranata.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 96;
    }
}

private void button97_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button97.Enabled = false;
        fileName = "backslash.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 97;
    }
}

private void button98_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button98.Enabled = false;
        fileName = "strednik.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 98;
    }
}

private void button99_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button99.Enabled = false;
        fileName = "apostrof.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);

```

```

        sc.StartRecording();
        isRecording = true;
        button = 99;
    }
}

private void button100_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button100.Enabled = false;
        fileName = "enter.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 100;
    }
}

private void button101_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button101.Enabled = false;
        fileName = "carka.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 101;
    }
}

private void button102_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button102.Enabled = false;
        fileName = "tecka.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 102;
    }
}

private void button103_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button103.Enabled = false;
        fileName = "lomitko.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,

```

```

SoundCapture._MONO);
    sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
    sc.StartRecording();
    isRecording = true;
    button = 103;
}
}

private void button104_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button104.Enabled = false;
        fileName = "rightshift.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        button = 104;
    }
}

public void Dispose_All()
{
    rawSoundData.Clear();
    rawSoundData = null;
    if (sc != null)
        sc.Dispose();
}

private void sc_OnWavComplete(object sender, EventArgs e)
{
    Buttons();
    label1.Text = "1000";
    switch (button)
    {
        case 1: button1.Enabled = true;
            break;
        case 2: button2.Enabled = true;
            break;
        case 3: button3.Enabled = true;
            break;
        case 4: button4.Enabled = true;
            break;
        case 5: button5.Enabled = true;
            break;
        case 6: button6.Enabled = true;
            break;
        case 7: button7.Enabled = true;
            break;
        case 8: button8.Enabled = true;
            break;
        case 9: button9.Enabled = true;
            break;
        case 10: button11.Enabled = true;
            break;
        case 12: button12.Enabled = true;
            break;
        case 13: button13.Enabled = true;
            break;
    }
}

```

```
case 14: button14.Enabled = true;
        break;
case 15: button15.Enabled = true;
        break;
case 16: button16.Enabled = true;
        break;
case 17: button17.Enabled = true;
        break;
case 18: button18.Enabled = true;
        break;
case 19: button19.Enabled = true;
        break;
case 20: button20.Enabled = true;
        break;
case 21: button21.Enabled = true;
        break;
case 22: button22.Enabled = true;
        break;
case 23: button23.Enabled = true;
        break;
case 24: button24.Enabled = true;
        break;
case 25: button25.Enabled = true;
        break;
case 26: button26.Enabled = true;
        break;
case 27: button27.Enabled = true;
        break;
case 28: button28.Enabled = true;
        break;
case 29: button29.Enabled = true;
        break;
case 30: button30.Enabled = true;
        break;
case 31: button31.Enabled = true;
        break;
case 32: button32.Enabled = true;
        break;
case 33: button33.Enabled = true;
        break;
case 34: button34.Enabled = true;
        break;
case 35: button35.Enabled = true;
        break;
case 36: button36.Enabled = true;
        break;
case 37: button37.Enabled = true;
        break;
case 38: button38.Enabled = true;
        break;
case 39: button39.Enabled = true;
        break;
case 40: button40.Enabled = true;
        break;
case 41: button41.Enabled = true;
        break;
case 42: button42.Enabled = true;
        break;
case 43: button43.Enabled = true;
        break;
case 44: button44.Enabled = true;
        break;
```

```
case 45: button45.Enabled = true;
        break;
case 46: button46.Enabled = true;
        break;
case 47: button47.Enabled = true;
        break;
case 48: button48.Enabled = true;
        break;
case 49: button49.Enabled = true;
        break;
case 50: button50.Enabled = true;
        break;
case 51: button51.Enabled = true;
        break;
case 52: button52.Enabled = true;
        break;
case 53: button53.Enabled = true;
        break;
case 54: button54.Enabled = true;
        break;
case 55: button55.Enabled = true;
        break;
case 56: button56.Enabled = true;
        break;
case 57: button57.Enabled = true;
        break;
case 58: button58.Enabled = true;
        break;
case 59: button59.Enabled = true;
        break;
case 60: button60.Enabled = true;
        break;
case 61: button61.Enabled = true;
        break;
case 62: button62.Enabled = true;
        break;
case 63: button63.Enabled = true;
        break;
case 64: button64.Enabled = true;
        break;
case 65: button65.Enabled = true;
        break;
case 66: button66.Enabled = true;
        break;
case 67: button67.Enabled = true;
        break;
case 68: button68.Enabled = true;
        break;
case 69: button69.Enabled = true;
        break;
case 70: button70.Enabled = true;
        break;
case 71: button71.Enabled = true;
        break;
case 72: button72.Enabled = true;
        break;
case 73: button73.Enabled = true;
        break;
case 74: button74.Enabled = true;
        break;
case 75: button75.Enabled = true;
        break;
```

```
case 76: button76.Enabled = true;
    break;
case 77: button77.Enabled = true;
    break;
case 78: button78.Enabled = true;
    break;
case 79: button79.Enabled = true;
    break;
case 80: button80.Enabled = true;
    break;
case 81: button81.Enabled = true;
    break;
case 82: button82.Enabled = true;
    break;
case 83: button83.Enabled = true;
    break;
case 84: button84.Enabled = true;
    break;
case 85: button85.Enabled = true;
    break;
case 86: button86.Enabled = true;
    break;
case 87: button87.Enabled = true;
    break;
case 88: button88.Enabled = true;
    break;
case 89: button89.Enabled = true;
    break;
case 90: button90.Enabled = true;
    break;
case 91: button91.Enabled = true;
    break;
case 92: button92.Enabled = true;
    break;
case 93: button93.Enabled = true;
    break;
case 94: button94.Enabled = true;
    break;
case 95: button95.Enabled = true;
    break;
case 96: button96.Enabled = true;
    break;
case 97: button97.Enabled = true;
    break;
case 98: button98.Enabled = true;
    break;
case 99: button99.Enabled = true;
    break;
case 100: button100.Enabled = true;
    break;
case 101: button101.Enabled = true;
    break;
case 102: button102.Enabled = true;
    break;
case 103: button103.Enabled = true;
    break;
case 104: button104.Enabled = true;
    break;
default:
    break;
}
}
```

```

        private void timer1_Tick(object sender, EventArgs e)
        {
            ticks++;
            if (ticks == 10)
            {
                timer1.Stop();
                sc.StopRecording();
                isRecording = false;
                sc.WriteWaveFile(fileName);
                sc.Dispose();
                ticks = 0;
            }
            label1.Text = Convert.ToString(1000 - (ticks * 100));
        }
    }
}

```

## Diplomka2

### Program.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace Diplomka2
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}

```

### Class1.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace Diplomka2
{
    class Class1
    {
        private UInt16 format;
        public UInt16 channels;
        public UInt16 bitPerSample;
        public UInt32 sampleRate;
        private UInt32 byteRate;
        private UInt16 blockAlign;
        private UInt32 dataLength;
        private byte[] data;
        private byte[] header;

        public void ReadWavFile(string fileName, out Int16[] wavData)
    }
}

```

```

{
    FileStream fs;
    try
    {
        header = new byte[44];
        fs = new FileStream(fileName, FileMode.Open);
        fs.Read(header, 0, header.Length);
        format = BitConverter.ToUInt16(header, 20);
        channels = BitConverter.ToUInt16(header, 22);
        sampleRate = BitConverter.ToUInt32(header, 24);
        byteRate = BitConverter.ToUInt32(header, 28);
        blockAlign = BitConverter.ToUInt16(header, 32);
        bitPerSample = BitConverter.ToUInt16(header, 34);
        dataLength = BitConverter.ToUInt32(header, 40);
        data = new byte[dataLength];

        fs.Read(data, 0, (UInt16)dataLength);
        fs.Close();
        fs.Dispose();

        if (channels == 2)
        {
            if (bitPerSample == 8)
            {
                int[] rightchannel = new int[(int)(dataLength / 2)];
                int[] leftchannel = new int[(int)(dataLength / 2)];
                wavData = new Int16[(int)(dataLength / 2)];

                int i = 0;
                int j = 0;

                while (i < dataLength)
                {
                    rightchannel[j] = ((data[i] - 128) * 256);
                    i = i++;
                    leftchannel[j] = ((data[i] - 128) * 256);
                    i = i++;
                    wavData[j] = (Int16)Math.Round((decimal)(rightchannel[j]
+ leftchannel[j]) / 2);
                    j++;
                }
            }
            else
            {
                int[] rightchannel = new int[(int)(dataLength / 4)];
                int[] leftchannel = new int[(int)(dataLength / 4)];
                wavData = new Int16[(int)(dataLength / 4)];

                int i = 0;
                int j = 0;

                while (i < dataLength)
                {
                    rightchannel[j] = BitConverter.ToInt16(data, i);
                    i = i + 2;
                    leftchannel[j] = BitConverter.ToInt16(data, i);
                    i = i + 2;
                    wavData[j] = (Int16)Math.Round((decimal)(rightchannel[j]
+ leftchannel[j]) / 2);
                    j++;
                }
            }
        }
    }
}

```



```

string fileName;
bool isRecording = false;
SoundCapture sc;
byte ticks;

double[] tempin, tempout;

GCHandle thin, thout;

double[][] datain = new double[104][];
double[][] dataout = new double[104][];

GCHandle hin, hout;

IntPtr fplan;

string[] fileNames = new string[104]
{"a.wav", "b.wav", "c.wav", "d.wav", "e.wav", "f.wav", "g.wav", "h.wav", "i.wav", "j.wav",
, "k.wav", "l.wav", "m.wav", "n.wav", "o.wav"
, "p.wav", "q.wav", "r.wav", "s.wav", "t.wav",
, "u.wav", "v.wav", "w.wav", "x.wav", "y.wav"
, "z.wav", "esc.wav", "f1.wav", "f2.wav",
, "f3.wav", "f4.wav", "f5.wav", "f6.wav", "f7
.wav", "f8.wav", "f9.wav", "f10.wav", "f11.wav",
, "f12.wav", "scrolllock.wav", "pause.wav",
, "insert.wav", "home.wav", "pageup.wav",
, "delete.wav", "end.wav", "pagedown.wav", "
left.wav", "up.wav", "down.wav", "right.wav",
, "numlock.wav", "numlomitko.wav", "numhvez
dicka.wav", "numminus.wav", "num7.wav",
, "num8.wav", "num9.wav", "numplus.wav", "nu
m4.wav", "num5.wav", "num6.wav", "num1.wav",
, "num2.wav", "num3.wav", "numenter.wav", "n
um0.wav", "numtecka.wav", "tilda.wav", "1.wav",
, "2.wav", "3.wav", "4.wav", "5.wav", "6.wav"
, "7.wav", "8.wav", "9.wav", "0.wav", "minus.wav",
, "rovnitko.wav", "backspace.wav", "tab.wav"
, "printscreen.wav", "capslock.wav",
, "leftshift.wav", "leftcontrol.wav", "left
win.wav", "leftalt.wav", "rightalt.wav",
, "rightcontrol.wav", "rightwin.wav", "menu
.wav", "space.wav", "levahranata.wav",
, "pravahranata.wav", "backslash.wav", "str
ednik.wav", "apostrof.wav", "enter.wav",
, "carka.wav", "tecka.wav", "lomitko.wav", "
rightshift.wav"};

public Form1()
{
    InitializeComponent();
    //nacti vsehny data a udelej fft
    Class1 clas = new Class1();
    for (int i = 0; i < 104; i++)
    {
        short[] j;
        clas.ReadWavFile(fileNames[i], out j);
        datain[i] = new double[Math.Min(j.Length, samples)];
        dataout[i] = new double[Math.Min(j.Length, samples) * 2];
        int l = 0;
        bool start = false;
        for (int k = 0; k < j.Length; k++)
        {

```

```

        if (Math.Abs((int)j[k]) > threshold)
            start = true;
        if ((start) && (1 < samples))
        {
            datain[i][l] = Convert.ToDouble(j[k]);
            l++;
        }
    }
    //get handles and pin arrays so the GC doesn't move them
    hin = GCHandle.Alloc(datain[i], GCHandleType.Pinned);
    hout = GCHandle.Alloc(dataout[i], GCHandleType.Pinned);
    fplan = fftw.dft_r2c_1d(datain[i].Length, hin.AddrOfPinnedObject(),
hout.AddrOfPinnedObject(), fftw_flags.Estimate);
    if (datain[i].Length > 0)
    {
        fftw.execute(fplan);
    }
}
}

private void button1_Click(object sender, EventArgs e)
{
    if (!isRecording)
    {
        timer1.Start();
        button1.Enabled = false;
        fileName = "temp.wav";
        sc = new SoundCapture(SoundCapture._44100_HZ, SoundCapture._16_Bit,
SoundCapture._MONO);
        sc.OnWavComplete += new SoundCapture.Complete(sc_OnWavComplete);
        sc.StartRecording();
        isRecording = true;
        label1.Text = "";
    }
}

private void Form1_Load(object sender, EventArgs e)
{
    ticks = 0;
}

private void timer1_Tick(object sender, EventArgs e)
{
    ticks++;
    if (ticks == 10)
    {
        timer1.Stop();
        sc.StopRecording();
        isRecording = false;
        sc.WriteWaveFile(fileName);
        sc.Dispose();
        ticks = 0;
    }
    label2.Text = Convert.ToString(1000 - (ticks * 100));
}

private void sc_OnWavComplete(object sender, EventArgs e)
{
    label2.Text = "1000";
    //zavolej fft a porovnali
    button1.Enabled = true;
    Class1 clas = new Class1();
}

```

```

short[] j;
clas.ReadWavFile("temp.wav", out j);
tempin = new double[Math.Min(j.Length, samples)];
tempout = new double[Math.Min(j.Length, samples) * 2];
int l = 0;
bool start = false;
for (int k = 0; k < j.Length; k++)
{
    if (Math.Abs((int)j[k]) > threshold)
        start = true;
    if ((start) && (l < samples))
    {
        tempin[l] = Convert.ToDouble(j[k]);
        l++;
    }
}
//get handles and pin arrays so the GC doesn't move them
thin = GCHandle.Alloc(tempin, GCHandleType.Pinned);
thout = GCHandle.Alloc(tempout, GCHandleType.Pinned);
fplan = fftw.dft_r2c_1d(tempin.Length, thin.AddrOfPinnedObject(),
thout.AddrOfPinnedObject(), fftw_flags.Estimate);
fftw.execute(fplan);
int i = Porovnej();
if (i == -1)
{
    label1.Text = "Nerozpoznáno!";
}
else
{
    label1.Text = fileNames[i].Substring(0, fileNames[i].Length - 4);
}
}

public int Porovnej()
{
    double dif = Double.MaxValue;
    int index = -1;
    for (int i = 0; i < 104; i++)
    {
        double adif = 0;
        if (datain[i].Length > 0)
        {
            for (int j = 0; j < Math.Min(dataout[i].Length, tempout.Length);
j++)
            {
                adif = adif + Math.Abs(tempout[j] - dataout[i][j]);
            }
            if (adif < dif)
            {
                dif = adif;
                index = i;
            }
        }
    }
    return index;
}

public void Dispose_All()
{
    rawSoundData.Clear();
    rawSoundData = null;
    if (sc != null)

```

```
    }  
  }  
}  
    sc.Dispose();
```

## PŘÍLOHA P 2: PŘEHLED SCAN KÓDŮ KLÁVESNICE.

Key #	Regular Character	Shifted Character	Make Code	Break Code
1	` (backwards quote)	~	29	A9
2	1	!	2	82
3	2	@	3	83
4	3	#	4	84
5	4	\$	5	85
6	5	%	6	86
7	6	^	7	87
8	7	&	8	88
9	8	*	9	89
10	9	(	0A	8A
11	0	)	0B	8B
12	- (dash)	_ (underscore)	0C	8C
13	=	+	0D	8D
16	<Tab>	<Backwards Tab>	0F	8F
17	q	Q	10	90
18	w	W	11	91
19	e	E	12	92
20	r	R	13	93
21	t	T	14	94
22	y	Y	15	95
23	u	U	16	96
24	i	I	17	97
25	o	O	18	98
26	p	P	19	99
27	[	{	1A	9A
28	]	}	1B	9B
29	\		2B	AB
31	a	A	1E	9E
32	s	S	1F	9F
33	d	D	20	A0
34	f	F	21	A1
35	g	G	22	A2
36	h	H	23	A3
37	j	J	24	A4
38	k	K	25	A5
39	l	L	26	A6
40	;	:	27	A7
41	' (regular quote)	" (double quote)	28	A8
42	#	~	2B	AB
43	<Enter>		1C	9C
45	\		56	D6
46	z	Z	2C	AC
47	x	X	2D	AD
48	c	C	2E	AE
49	v	V	2F	AF
50	b	B	30	B0
51	n	N	31	B1
52	m	M	32	B2
53	, (comma)	<	33	B3
54	. (period)	>	34	B4
55	/	?	35	B5
61	<Space Bar>		39	B9

Key #	Character	Make Code	Break Code
30	<Caps Lock>	3A	BA
44	<Left Shift>	2A	AA
57	<Right Shift>	36	B6
58	<Left Ctrl>	1D	9D
59	<Left Alt>	38	B8
62	<Right Alt>	E0 38	E0 B8
63	<Right Ctrl>	E0 1D	E0 9D
90	<Num Lock>	45	C5
125	<Scroll Lock>	46	C6

Key #	Character	Make Code	Break Code
15	<Backspace>	0E	8E
75	<Insert>	E0 52	E0 D2
76	<Delete>	E0 53	E0 D3
79	<Left Arrow>	E0 4B	E0 CB
80	<Home>	E0 47	E0 C7
81	<End>	E0 4F	E0 CF
83	<Up Arrow>	E0 48	E0 C8
84	<Down Arrow>	E0 50	E0 D0
85	<Page Up>	E0 49	E0 C9
86	<Page Down>	E0 51	E0 D1
89	<Right Arrow>	E0 4D	E0 8D

Key #	Regular Character	NumLocked" Charact	Make Code	Break Code
91	<Home>	7	47	C7
92	<Left Arrow>	4	4B	CB
93	<End>	1	4F	8F
95	/		E0 35	E0 B5
96	<Up Arrow>	8	48	C8
97	(none)	5	4C	CC
98	<Down Arrow>	2	50	D0
99	<Ins>	0	52	D2
100	*		37	B7
101	<PgUp>	9	49	C9
102	<Right Arrow>	6	4D	CD
103	<PgDn>	3	51	D1
104	<Del>	. (period)	53	D3
105	-		4A	CA
106	+		4E	CE
108	<Enter>		E0 1C	E0 9C

Key #	Regular Command	Alternate Command	Make Code	Break Code
110	<Esc>		1	81
124	<Print Screen>	<System Request>	E0 2A E0 37	E0 B7 E0 AA
126	<Pause>	<Break>	E1 1D 45 E1 9D C5	(none)

Key #	Character	Make Code	Break Code	Break Code
112	<F1>	3B	BB	C7
113	<F2>	3C	BC	CB
114	<F3>	3D	BD	8F
115	<F4>	3E	BE	E0 B5
116	<F5>	3F	BF	C8
117	<F6>	40	C0	CC
118	<F7>	41	C1	D0
119	<F8>	42	C2	D2
120	<F9>	43	C3	B7
121	<F10>	44	C4	C9
122	<F11>	57	D7	CD
123	<F12>	58	D8	D1
104	<Del>	. (period)	53	D3
105	-		4A	CA
106	+		4E	CE
108	<Enter>		E0 1C	E0 9C

Key #	Command	Make Code	Break Code	Break Code
-	<Left Windows>	E0 5B	E0 DB	81
-	<Right Windows>	E0 5C	E0 DC	E0 B7 E0 AA
-	<ContextMenu>	E0 5D	E0 DD	(none)