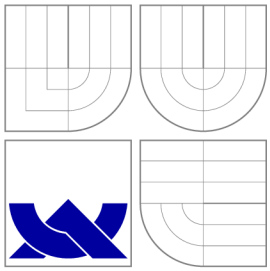
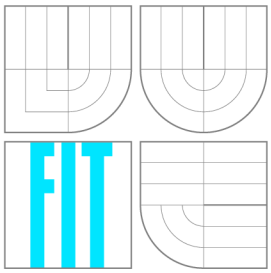


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

INTELIGENTNÍ VYZVÁNĚNÍ PRO PLATFORMU MAEMO NEBO MEEGO

INTELLIGENT RINGTONES FOR MAEMO OR MEEGO PLATFORM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

RADIM ČEJKA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JOZEF MLÍCH

BRNO 2012

Abstrakt

Tato bakalářská práce se zabývá platformou Maemo a MeeGo. Cílem práce bylo vytvoření aplikace inteligentní změny vyzváněcích profilů. Tato změna probíhá na základě času, senzorů v telefonu jako je akcelerometr, senzor přiblížení, orientace telefonu a okolního osvětlení či na základě lokace. Práce popisuje teorii vývoj a ladění pro jiné platformy za použití metody cross-kompilace. Podrobně rozebírá návrh a implementaci aplikace. Zhodnocuje dosažené výsledky a zmiňuje možná budoucí vylepšení.

Abstract

This bachelor's thesis deals with Maemo and MeeGo platform. The goal of the work was a creation of an application which handles the inteligent profile change. This change is contolled by time, phone sensors like accelerometer, proximity sensor, orientation senzor, ambient light sensor or by location. The work describes how to develop and debbug for different platforms by using cross-compiling. The Design description and implementation follows. Also there is evaluation of the result and discusion about further upgrades.

Klíčová slova

Maemo, N900, Nokia, vyzvánění, senzory, QT Mobility, DTW

Keywords

Maemo, N900, Nokia, ringtones, sensors, QT Mobility, DTW

Citace

Radim Čejka: Inteligentní vyzvánění pro platformu
Maemo nebo MeeGo, bakalářská práce, Brno, FIT VUT v Brně, 2012

Inteligentní vyzvánění pro platformu Maemo nebo MeeGo

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jozefa Mlícha

.....
Radim Čejka
16. května 2012

Poděkování

Rád bych poděkoval vedoucímu mé práce Ing. Jozefu Mlíchovi za odborné vedení, užitečné podměty a rady.

© Radim Čejka, 2012.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Kapitola 1

Úvod

Mobilní telefon vlastní téměř každý a často se stane, že někomu zazvoní na nevhodném místě. Nebo naopak někdo čeká důležitý hovor, ale zmešká ho, protože si ráno zapomněl zapnout vyzvánění. Dnešní mobilní telefony disponují spoustou užitečných senzorů, které dokážou zaznamenávat pohyb přístroje, okolní světlo či polohu. Je škoda jich nevyužít. Díky nim pak uživatel může definovat určité situace, kdy má telefon zvonit a kdy ne, a nemusí dále na nic myslet, telefon se o vše postará sám.

Bezdrátové mobilní telefony se poprvé objevily v Americe v padesátých letech minulého století. Byly pevnou součástí automobilů. První přenosné telefony následovaly v 70. letech. Jednalo se o velké a těžké přístroje velikosti kufříku, které uměly jen telefonovat. Jejich výdrž na baterii byla velmi slabá. U nás se mobilní telefony začaly prodávat až po revoluci od roku 1990. Šlo o analogové přístroje za desetitisíce korun. Větší rozmach nastal v roce 1996. Na český trh vstoupil druhý operátor a ceny klesly na přijatelnou úroveň. Mobily se tehdy vyznačovaly černobílým displejem, numerickou klávesnicí a anténou. Umožňovaly telefonovat, posílat krátké textové zprávy o délce 160 znaků a zvládly si zapamatovat maximálně desítky telefonních čísel či SMS.

Současné mobilní telefony se vyznačují velkými dotykovými displeji, většina již nemá klávesnici a anténa je integrovaná. Je s nimi možné kromě telefonování a posílání zpráv pořizovat a posílat fotografie, videa, prohlížet internetové stránky, poslouchat hudbu, hrát hry, využívat satelitní navigaci. Vrcholem nabídky jsou chytré telefony. Díky jejich operačním systémům lze chybějící funkce telefonu doplnit některou z mnoha dostupných aplikací.

Jednou z těchto funkcí je důkladnější správa vyzváněcích profilů na telefonu Nokia N900, založená na informacích z okolí přístroje. Ten umožňuje jen dvě možnosti nastavení definované výrobcem. Základní, kdy telefon zvoní, a Tichý, kdy telefon nezvoní. U každé možnosti pak už jen povolení nebo zákaz vibrací. Některé rozšiřující aplikace existují, ale neposkytují plně potřebnou funkcionalitu.

Cílem této práce je představení linuxové platformy pro mobilní telefony Maemo a navazující MeeGo, návrh a implementace aplikace spravující vyzváněcí profily telefonu. Vzhledem k aktuální dostupnosti hardware je aplikace primárně určena pro operační systém Maemo 5 a s ním pracující telefon Nokia N900. Na tomto telefonu byla aplikace implementována a testována.

Aplikace pracuje plně v interakci se svým okolím a využívá většiny dostupných senzorů telefonu. Na základě takto získaných dat je nastaven určitý profil, který byl předdefinován vývojářem nebo vytvořen uživatelem.

Ve druhé kapitole je představen operační systém Maemo a jeho nástupce MeeGo, historie a budoucnost tohoto ekosystému, jsou představena zařízení, na kterých se s ním můžeme se-

tkat. Seznámíme se se softwarovým a hardwarovým řešením telefonu se zaměřením na části relevantní pro tuto aplikaci.

Ve třetí kapitole je představena metoda cross-compiling, krátce shrnuty jednotlivé možnosti vývoje a ladění pro tuto platformu.

Čtvrtá kapitola seznamuje s API pro práci se senzory za použití Qt Mobility. Jsou zde vybrány vhodné senzory pro daný účel aplikace a popis získání a zpracování dat pro tvorbu profilů. Konkrétně metoda porovnání obrazů pohybů DTW a získání úrovně hluku pomocí knihovny PulseAudio.

V páté kapitole jsou uvedena v současnosti dostupná řešení, je popsán návrh programu, možnosti jeho strukturování, volba vhodného uživatelského rozhraní a tvorba vestavěných profilů.

Šestá kapitola se týká samotné implementace jednotlivých součástí aplikace, je popsáno rozdělení na aplikaci daemona a klientskou aplikaci, ukládání a načítání dat a testování samotné aplikace.

V poslední kapitole je shrnuta celá práce, její výsledky a směr možného budoucího vývoje.

Kapitola 2

Operační systém Maemo

V této kapitole je popsán operační systém Maemo a MeeGo z hlediska softwarové architektury [15]. Představí se telefon Nokia N900 po softwarové i hardwarové stránce, kde jsou podrobně popsány jeho senzory akcelerometru a senzoru přiblížení. Je zde rozebrána komunikace pomocí rozhraní D-Bus, s uvedením důležitých pojmů pro vytvoření D-Bus zprávy. Je ukázáno nastavení základního profilu[1] a položky tvořící vyzváněcí profil v telefonu.

2.1 Zařízení a hardware

První verze Maema se objevila v listopadu roku 2005 v internetovém tabletu Nokia N770. Původně se jednalo o multimediální operační systém pro přenosná zařízení s dotykovým displejem. K internetu se připojovala pomocí WiFi sítí. Měla sloužit k prohlížení internetu, práci s elektronickou poštou, elektronickými knihami,prohlížení fotografií a videa.

Posledním zástupcem se systémem Maemo a prvním s funkcí mobilního telefonu je Nokia N900. Telefon pohání procesor ARM Cortex A8 s taktem 600 MHz, je součástí čipu OMAP 3430 společnosti Texas Instruments. Schéma jednotlivých komponent je vidět na obrázku 2.1 převzato z[2]. Dále obsahuje 5 MPx fotoaparát s optikou Carl Zeiss, nabízí 32 GB flash paměti, FM vysílač, GPS čip, výsuvnou qwerty klávesnici. Ke spojení slouží USB, Bluetooth a WiFi IEEE 802.11 b/g. Bližší podobnosti o jednotlivých zařízeních je možné najít na stránkách společnosti Nokia[3].

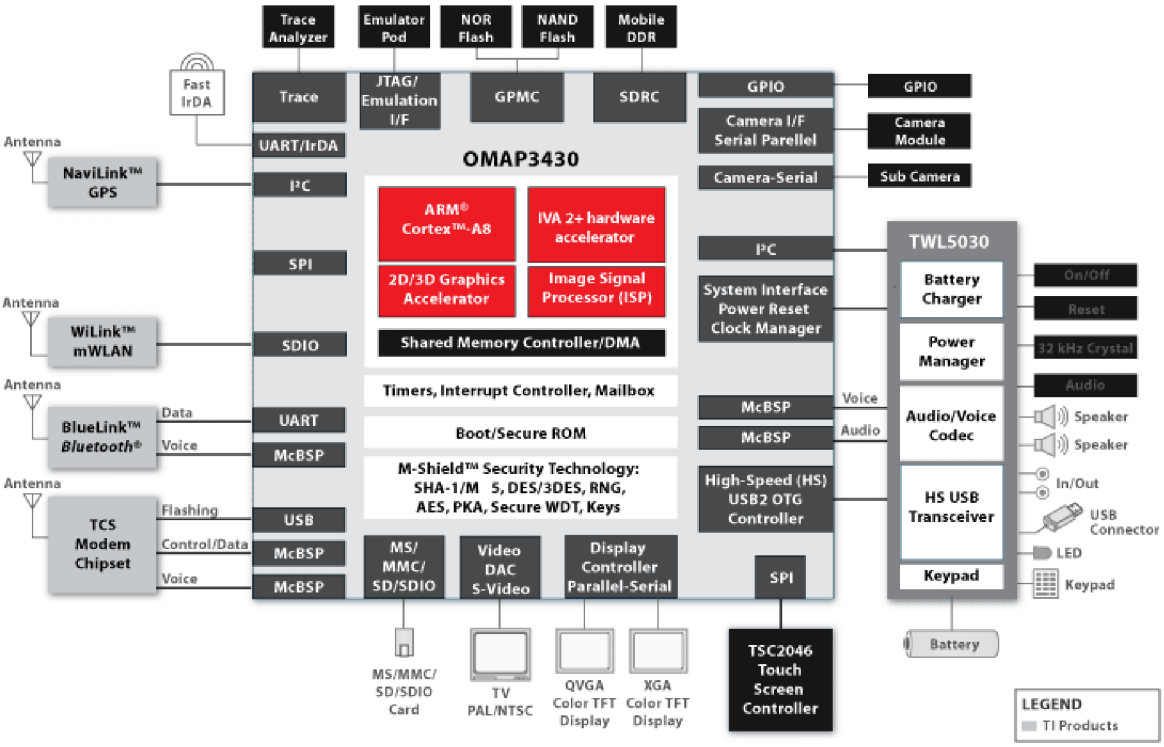
Senzor okolního světla v telefonu vyrábí společnost Taos, jedná se o čip TSL2563. Senzor v sobě kombinuje širokopásmovou fotodiodu (350nm až 1100nm) a infračervenou fotodiodu. Převádí získané analogové hodnoty na digitální, integruje z obou diod a mikroprocesor z nich vypočítá osvětlení v Luxech a výsledek posílá dál přes I2C rozhraní. Infračervená dioda slouží k eliminaci vlivu infračerveného světla na výstup a jeho přiblížení jasu, jak jej vnímá lidské oko. Perioda senzoru je 402ms. Převod na Luxy probíhá na základě empirických vzorců, které jsou dohledatelné ve specifikaci čipu [11].

Akcelerometr je reprezentován čipem LIS302DL od společnosti STMicroelectronics. Skládá se ze snímacího prvku a rozhraní IC komunikující přes sériové rozhraní I2C/SPI. Má dynamicky uživatelsky nastavitelný rozsah ± 2g ± 8g a výstupní frekvenci 100Hz až 400Hz. Snímací prvek je postaven na technologii trojrozměrného kapacitního akcelerometru. Seismická hmota je zavěšena na pružných křemíkových vláknech, která jsou na několika místech

[1] http://wiki.maemo.org/Phone_control#DBUS

[2] http://www.ti.com/general/docs/wtbu/wtbuproductcontent.tsp?templateId=6123&navigationId=12643&contentId=14649&DCMP=WTBU&HQS=Other+OT+omap3430

[3] http://www.nokia.com/



Obrázek 2.1: HW schéma čipu OMAP 3430 z telefonu Nokia N900.

přichycena k podkladu a mohou se pohybovat ve směru měřeného zrychlení. Při pohybu vznikne nerovnováha na kondenzátorech oproti klidovému stavu, která je měřena. Podrobnosti k nalezení na [12].

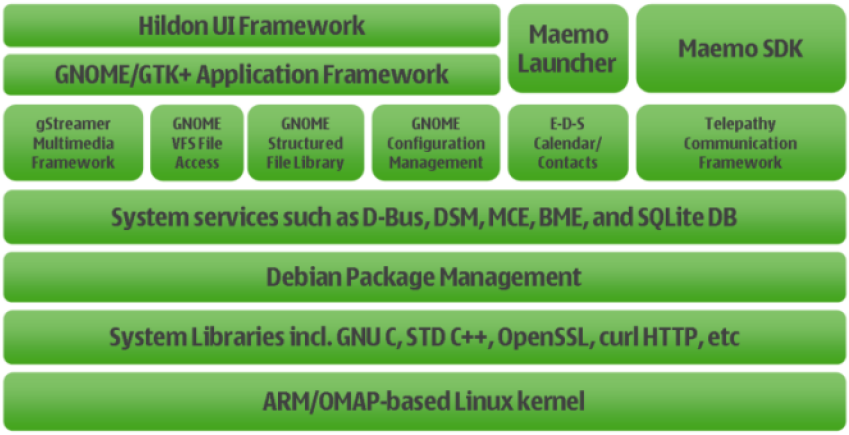
2.2 Systém

Maemo je softwarová platforma založená na open-source operačním systému Linux a dalších open-source projektech. Obsahuje linuxové jádro ve verzi 2.6 a upravené desktopové prostředí GNOME - knihovna Hildon. Ta přejímá grafickou knihovnu GTK+ a rozšiřuje ji o prvky použitelné na malých displejích přenosných zařízení. Jako engine uživatelského rozhraní slouží X window manager a Matchbox jako okenní správce. Vychází z distribuce Debian, využívá jeho balíčkovací systém deb. Vyvíjena byla společností Nokia.

Jak je vidět na obrázku 2.2 převzato[4] Maemo se svou architekturou příliš neliší od jiných linuxových distribucí. Nejníže se nachází jádro Linuxu, které se stará o správu paměti, procesů, síťových služeb, souborů a souborových systémů. Jedná se o verzi pro architekturu ARM s ovladači pro procesory OMAP. Nad jádrem pracují systémové knihovny postavené na standardních GNU C knihovnách OpenSSL, POSIX, BusyBox,.... V další vrstvě je balíčkovací systém Debianu, nad ním pak systémové služby. Většina komunikace mezi aplikacemi probíhá přes D-Bus. Systémové služby také poskytují služby aplikacím a koncovým uživatelům, jako je správa stavu zařízení DSM, mód kontroly MCE, správa baterie BME, SQL databázi SQL Lite 3 pro ukládání aplikačních dat či některé prvky grafického rozhraní pro správu služeb. Nejvýše pracují knihovny grafického rozhraní GNOME, Hildon a jed-

[4] http://maemo.org/intro/platform/

notlivá aplikační rozhraní Multimedia framework, Location framework, Communication framework[5].



Obrázek 2.2: Softwarové schéma systému Maemo.

Aktuální verze Maemo 5 se objevuje v prvním mobilním telefonu s Maemem Nokia N900. Přináší nový framework uživatelského rozhraní za použití knihovny Clutter OpenGL. Díky němu je systém obohacen o domácí obrazovku s možností více ploch, multitasking s živými náhledy, novou stavovou oblast a nové API pro obsluhu systémových notifikací. Nabízí integraci sociálních sítí přímo do systému. Krom telefonování není možné s telefonem pracovat na výšku.

Architektura systému MeeGo je podobná systému Maemo, základem je jádro, nad nímž pracují systémové knihovny, komunikace probíhá přes D-Bus, liší se až na vyšších vrstvách, kde knihovny uživatelského GNOME nahrazuje Qt. To je plně integrováno do systému a programátor může využít Qt API pro práci s kteroukoli částí zařízení. Systém MeeGo také nemá pevně dané uživatelské rozhraní a každý výrobce může nasadit vlastní prostředí [19].

2.3 D-Bus

Pro moji práci je podstatná služba D-Bus. Tato služba poskytuje meziprocesovou komunikaci (IPC), kdy programy mohou vystavit svoje vlastní programové rozhraní tak, aby je ostatní procesy mohly volat bez nutnosti definovat vlastní IPC protokol. V Maemu existuje knihovna libOSSO, která zjednodušuje použití RPC zpráv, dokáže zachytit a zpracovat zprávy hardwaru (např. uložení rozpracovaných dat při slabé baterii a vypnutí přístroje) [15].

Základem konceptu D-Bus je využití sběrnic. V tomto případě se sběrnicí myslí instance systémové aplikace dbusd, která obstarává komunikaci mezi procesy. Když aplikace volá potřebnou metodu, připojí se na sběrnici, pošlou signály a poslouchají příchozí signály. Existují dvě základní sběrnice session bus a systém bus. Session bus slouží ke komunikaci aplikací z jedné relace a standardně spuštěné jedním uživatelem. System bus slouží ke komunikaci aplikací či služeb z různých oddělených relací.

[5] http://maemo.org/intro/platform/

Na rozdíl od systémové sběrnice *systém bus* může existovat více relačních sběrnic *session bus*, teoreticky až pro každou relaci jedna. V Maemu však všechny aplikace běží pod stejným user ID, což znamená jen jeden session bus. O zpracování D-Bus zpráv se stará služba operačního systému tzv. daemon, který přeposílá jednotlivé zprávy. D-Bus totiž podporuje point-to-point komunikaci, ta probíhá vždy mezi aplikací a daemonem pomocí local domain socketů (AF_UNIX). Daemon zvládá i adresování, takže se aplikace nemusí starat, který konkrétní proces obdrží jejich volání. Pro adresování jsou důležité následující pojmy:

Pro pojmenování konkrétního připojení se používá **Bus name**, nikoli celé sběrnice. Každé připojení má svůj vlastní prostor jmen. Může to být unikátní identifikátor přidělený systémem D-Bus nebo "dobře známé jméno"(well-known name), které si aplikace přidělí sama.

Path je cesta k objektu, který vyslal signál nebo kterému je zpráva určena. Objekt je koncový bod každé komunikace, poskytuje jedno nebo více rozhraní službám. Využívá se lomítkové notace podobně jako u souborových systémů.

Konkrétního rozhraní objektu se nazývá **interface**, kde bude daná metoda provedena. Využívá se tečkové notace.

Konkrétní metodu nebo signál implementovaného daným objektem popisuje **member**. Jméno je unikátní v rámci jednoho rozhraní. V jedné zprávě se může vyskytovat více členů.

Zpráva pro změnu profilu[6] na profil základní pak bude vypadat následovně:

- bus name: com.nokia.profiled
- path:/com/nokia/profiled
- interface: com.nokia.profiled.set_profile
- member: string:"general"

Vyzváněcí profil telefonu, se kterým bude aplikace pracovat, se skládá z údajů o vibracích, alarmech, kalendáře a hodin, hodnota zapnuto nebo vypnuto, cestě k souboru s melodií a úrovní hlasitosti $0 - -100$ pro SMS a im zprávy, emaily a vyzvánění, tříúrovňových hodnot zvuků klávesnice, displeje a systémových zvuků a typu vyzvánění Ringing nebo Silent. Při nastavení Silent, jsou všechny zvuky bez ohledu na předchozí nastavení potlačeny.

[6] http://wiki.maemo.org/Phone_control#DBUS

Kapitola 3

Vývoj

Tato kapitola se věnuje vývoji na rozdílné platformy i konkrétně pro Maemo. V první části této kapitoly je popsán způsob překladu programu pro odlišné architektury procesoru [13]. Jsou zde rozebrány pojmy jako hostitel, cíl,*pkg-config* nebo *toolchain*. Dále je představeno prostředí Scratchbox, které umožnuje mít systém v systému [8]. Taktéž jsou zmíněny dostupné způsoby ladění aplikací pro Maemo, tedy přes gdb a valgrind. Jsou zmíněny možnosti knihoven Qt a Qt Mobility. Dále bylo čerpáno také z [4].

3.1 Cross-compiling

Každá aplikace musí být před spuštěním přeložena ze zdrojového kódu. Je přeložena překladačem pro určitý typ počítače (architektura, procesor). Při vývoji na stolní počítač se běžně program překládá pro stejnou architekturu, na které se program vyvíjí. Pokud se vyvíjí pro jinou architekturu, může se využít i pro vývoj a překlad, pokud to umožňuje, nebo se opět může použít stolní počítač s jinou architekturou a na něm využít metodu cross compilingu.

Cross compiling je překlad aplikace ze zdrojového kódu, na jiné architektuře než na jaké bude aplikace spuštěna. Zavádí se pojem Hostitel anglicky *Host* a Cíl anglicky *Target*. Hostitel je počítač, na kterém se vyvíjí a překládá. Cíl je počítač, na kterém má být aplikace spuštěna. Pokud Hostitel a Cíl běží na stejné architektuře, jedná se o nativní překlad.

Cross-compiling přináší spoustu výhod, ale je tu pár věci, na něž je důležité si dát pozor. Je potřeba řešit délku slova. Na 64 bitovém systému hrozí ztráta dat při kopírování pointeru datového typu `int`. Musí se hlídat alokace datového typu `long` a zarovnání paměti, kde se pořadí bytů se může lišit, např. x86 používá zarovnání nejvýznamnějšího bytu na nejvyšší adresu *Little Endian*, naopak RISC ukládá nejvýznamnější byte na nejnižší adresu *Big Endian*. Některé procesory např. ARM obsahují speciální příznak, který pak předepisuje, zda se bude chovat jako Little Endian či Big Endian. Některé platformy mají specifické požadavky pro čtení a zápis z paměti, kompilátor by pak měl umět převod na tvar vhodný pro cíl. Zda je datový typ `char` signed či unsigned, opět záleží na platformě, proto je možné překladači vnutit použití typické pro cíl.

3.2 Scratchbox

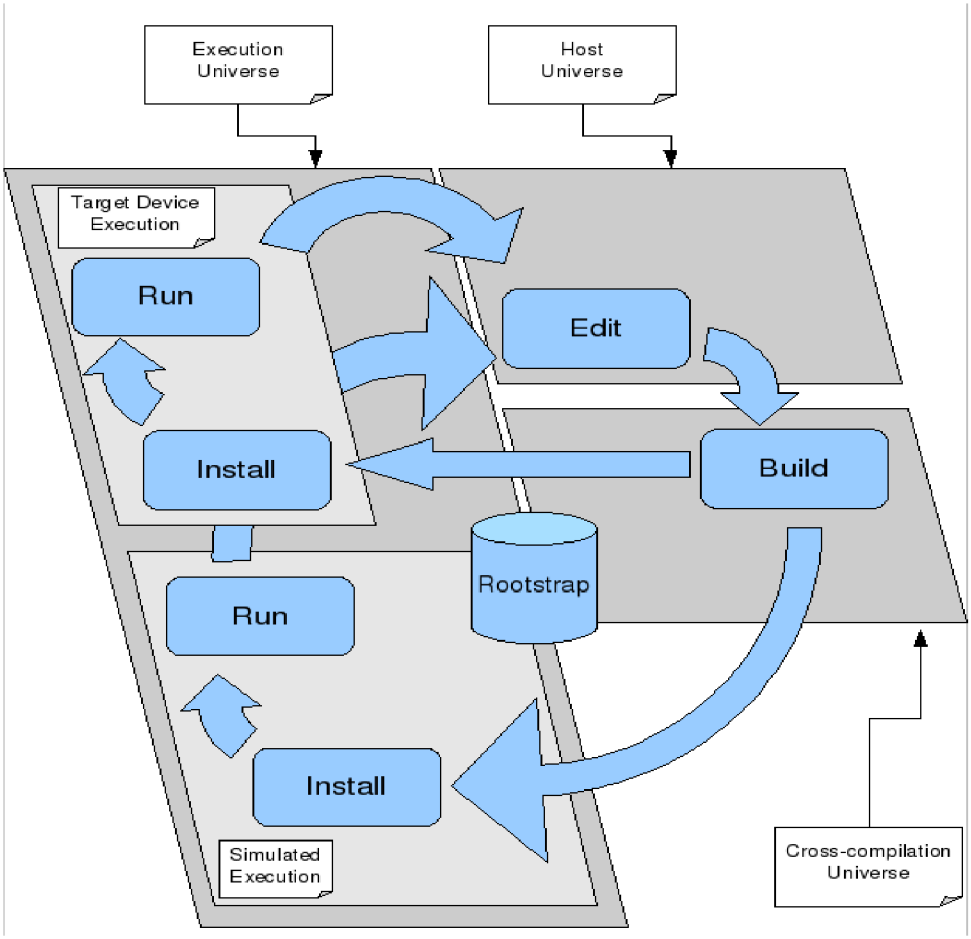
Při vývoji pro Maemo se nejčastěji používá balík nástrojů Scratchbox. Umožňuje překládat zdrojový kód pro x86 i pro ARM. Může mít gnu c a c++ překladač a umožňuje mezi nimi

přepínat. Je standardní součástí Maemo SDK od společnosti Nokia. Podporuje i pro jiné systémy. Přihlašuje se do něj jako do skutečného počítače. Obsahuje také balíčkovací systém Debianu, takže je pak možné ze zdrojových kódů sestavit více binárních deb balíčků pro více architektur.

Cyklus vývoje při cross-compiling je znázorněn na obrázku 3.1 převzat z [13]. Vývojové prostředí a editace zdrojového kódu spadá do prostoru Hostitele *Host Universe*. Prostor cross-kompilace *Cross-compilation Universe* se skládá z knihoven a hlavičkových souborů pro cílové zařízení. Prostor běhu *Execution Universe* poskytuje prostředky pro instalaci a běh aplikace. Je rozdělen na dvě části, jedna pro běh na skutečném zařízení *Target Device Execution*, druhá pak simuluje běh na vývojářské pracovní stanici.

Aby nebylo nutné u každého zdrojového souboru ručně přidávat potřebné hlavičkové soubory a knihovny pro danou platformu, obsahuje SDK nástroj pkg-config, který vypíše a doplní vše potřebné. Například pro přidání knihoven GTK+ stačí zadat: `pkg-config --cflags gtk+-2.0` a `pkg-config --libs gtk+-2.0`.

Rootstrap je virtuální kořenový systém, odpovídá kořenovému adresáři cíle. Funguje na principu změny adresářové struktury pomocí chroot. Skutečný kořenový adresář počítače se změní ve virtuální umístěný jinde. Aplikaci je však tato změna skryta. Díky tomu není potřeba mít práva superuživatele a každá aplikace může bezpečně běžet ve svém vyhrazeném prostoru.



Obrázek 3.1: Vývojový cyklus za použití metody cross compiling.

Scratchbox umožňuje uložit více cílových sestavení a každé z nich má unikátní nastavení. Ke změně a nastavení slouží nástroj sb-conf. Pokud je jako cíl překladu nastaven hostitel, přeložený kód se spustí přímo. Při překladu pro jinou platformu se musí nastavit i způsob spouštění aplikace nejčastěji emulátor QEMU. O tom, jaký cíl je nastaven, informuje Scratchbox konzole např: `[sbox-HOST: ]`, `[sbox-ARM: ]`,...

Pro překlad aplikací slouží toolchain, což je sada nástrojů vytvářející spustitelnou aplikaci pro cílovou platformu. Skládá se z konkrétních verzí překladače gcc, linkeru ld a dalších nástrojů strip, objdump, strings. Scratchbox standardně obsahuje toolchain pro x86 a ARM:
`scratchbox-toolchain-arm-gcc3.3-glibc2.3`
`scratchbox-toolchain-i686-gcc3.3-glibc2.3`
Všechny nainstalované toolchainy se nachází v adresáři `/scratchbox/compilers`. Každý projekt ve Scratchboxu musí mít vybrán jeden konkrétní toolchain.

Pro zobrazení cílové platformy slouží X server Xephyr, který se instaluje spolu se Scratchboxem. Pro ladění poskytuje Scratchbox typické linuxové nástroje jako je gdb, valgrind nebo strace.

3.3 Ladění

Pro ladění aplikací pro Maemo existují dva základní nástroje. Gdb – Gnu Project Debugger a Valgrind.

Pro ladění jsou potřeba ladící symboly. V případě samotné aplikace se přidají přepínačem `-g` překladače. Pro přidání ladících symbolů do knihoven je nutno knihovny také přeložit s tímto přepínačem. Do systému je možné takto přeložené knihovny doinstalovat. Balíčky obsahující ladící informace jsou v repositářích označeny "-dbg". Bez nich není možné provést správné trasování programu. Také funkce jazyka C ve tvaru `__attribute__((__noreturn__))` musí být překládány v gcc s volbou `-mapcs-frame`. Bez této volby by nebylo možné trasovat těmito funkcemi, což by při ladění mělo za následek, že se program dostane do nekonečné smyčky. Gdb také potřebuje mít přístup k ladícím symbolům (debug symbols), bez nich by nebylo schopno správně zobrazovat názvy funkcí.

Ladit pomocí gdb je možné buďto ve Scratchboxu, emulátoru QEMU nebo přímo na telefonu. U QEMU se v aktuální verzi Qt Creatoru není nutné o nic starat a hned po instalaci je všechno dostupné a funkční. Ve Scratchboxu je potřeba doinstalovat balíček `maemo-sdk-debug`, který do prostředí Scratchboxu nainstaluje standardní x86 gdb (příkaz `gdb`) a nativní verzi pro Maemo (příkaz `native-gdb`). Aplikace musí být přeložena s parametrem `-g` pro zachování ladících informací v přeloženém binárním kódu. Ten už se může použít v nativním gdb. Pak už se práce neliší od běžného ladění v gdb. Při ladění grafických aplikací je nutné pouze navíc na začátku spustit Xephyr server a ve Scratchboxu se na něj připojit.

Při ladění přímo na zařízení je nutné jej připravit. Pokud se již v systému nenachází, musí se doinstalovat SSH server a gdb. SSH server se nachází v repositářích `extras` a gdb pak v repositářích `fremantle/sdk` a `fremantle/tools`. Instalaci pak lze provést přes grafický správce balíčků nebo `apt-get`. Stejně jako při ladění ve Scratchboxu, se musí aplikace přeložit s parametrem `-g`. Pro přenos souboru i následnou obsluhu gdb je nutné se na zařízení přihlásit přes SSH. Jako uživatelské jméno slouží „*user*“. Při ladění přímo na telefonu je vývojář omezen jen velikostí dostupné paměti. Pokud dojde, aplikace se pouze ukončí.

Hojně užívaná forma ladění je využití některého vývojového prostředí. Nabízí se využít Qt Creator, dodávaný společností Nokia, zaměřený přímo na vývoj pro mobilní telefony. Jak již bylo zmíněno výše, emulátor v něm funguje okamžitě a poskytuje nástroje pro simulování systémových událostí, jako jsou údaje o připojených sítích, GPS pozici, baterii

nebo data ze senzorů. Hlavní devíza spočívá v nástroji MADDE. Umožňuje jednoduché vytváření `deb` balíčků, spojení přes ssh s telefonem přímo z prostředí Creatoru. Podporuje spojení přes USB, WiFi i GPRS. Pro správnou funkci je potřeba mít v zařízení nainstalovanou aplikaci Mad Developer z repositářů `extras-devel`. V něm se nastavuje IP adresa pro připojení přes USB a generuje heslo potřebné pro připojení. V Cretoru se pak připojuje s tímto heslem a uživatelským jménem „developer“. Pokud pokaždé nechceme generovat a zadávat nová hesla, stačí do zařízení přenést veřejný klíč a v Qt Cretoru, nastavit cestu k soukromému. Do zařízení je pak možné odesílat přeložené aplikace jako `deb` balíčky, či jako spustitelné soubory. Ladění pak probíhá přímo na telefonu s využitím vestavěného debbugeru využívajícího gdb.

Valgrind pracuje na principu simulace procesoru, sám vykonává veškeré příkazy a plně řídí provádění programu. Díky tomu může na zařízení aplikace běžet 10-300 krát pomaleji. Ladit pomocí Valgrindu je možné přímo na zařízení. Balíček se nachází v repositářích `extras-devel` a je závislý na knihovnách `libc6-dbg` z `fremantel/sdk-repository`. Nebo je možné využít prostředí Scratchboxu. Zde však není podporována verze pro architekturu ARM a je nutné překládat pro x86.

Mezi základní funkce patří Massif, Memcheck, Cachegrind a Callgrind. **Massif** kontroluje haldu a vytváří graf použití paměti. **Memcheck** detekuje chyby přístupu do paměti. **Cachegrind** kontroluje cache paměť a je zaměřen na výkon s možností krokování. **Callgrind** je podobný Cachegrindu, navíc oproti němu vytváří graf volání instrukcí.

3.4 Možnosti vývoje

Vývoj pro Maemo možný v programovacích jazycích C, C++ a Python. Pro grafické rozhraní se může využít pro Gnome typické GTK+ nebo knihovny QT.

Integrace knihoven QT souvisela se snahou Nokie sjednotit vývoj všech aplikací pro obě své platformy (Symbian, Maemo) pod jeden framework a to QT. Pro zachovaní kompatibility a celistvosti systému napodobuje vzhled Hildon programů, obsahuje podporu vkládaní z Hildon - možnost použití virtuální klávesnice i funkčnost pro Hildon specifických widgetů.

Qt Mobility je nástavbu pro Qt, která přináší přístup k funkcím telefonu. Tyto rozhraní však lze využít i na noteboocích či stolních počítačích. Skládá se z různých API. Pro tuto práci jsou podstatné: *Qt Service Framework*, které umožňuje klientům nacházet a přistupovat ke službám nezávisle na platformě, *Sensors*, které poskytuje přístup k hardwarovým senzorům zařízení jak vysokoúrovňovým, jako orientace displeje, tak nízkoúrovňovým, jako je akcelerometr. *System Information* poskytuje systémové informace klientovi, jako jsou informace o verzích programů, hardwarových komponentech, sítových připojeních nebo stavu paměťových úložišť.

Stěžejní pro tuto práci je Sensors API, které bude podrobně popsáno níže.

Kapitola 4

Práce se senzory telefonu

V této kapitole je popsána práce se senzory přístroje s pomocí Qt Mobility API [10], je uveden přesný postup čtení při čtení ze senzorů. Jsou zmíněny dostupné API a senzory, které jsou součástí telefonu Nokia N900. Je popsán výběr vhodných senzorů pro cílovou aplikaci, konkrétně akcelerometr, senzor přiblížení, orientace a okolního osvětlení. Je uveden způsob získání a zpracování úrovně hluku pomocí knihoven PulseAudio [7] a projektu Vumeter [9] a způsob porovnání záznamů pohybu pomocí metody DTW [3].

4.1 Dostupné API pro práci se senzory

Qt Mobily obsahuje speciální API pro práci se senzory. Pro měření polohy přístroje v prostoru se používají kladné a záporné hodnoty na osách x, y a z od původní pozice. Při měření rotace je systém souřadnic pravotočivý.

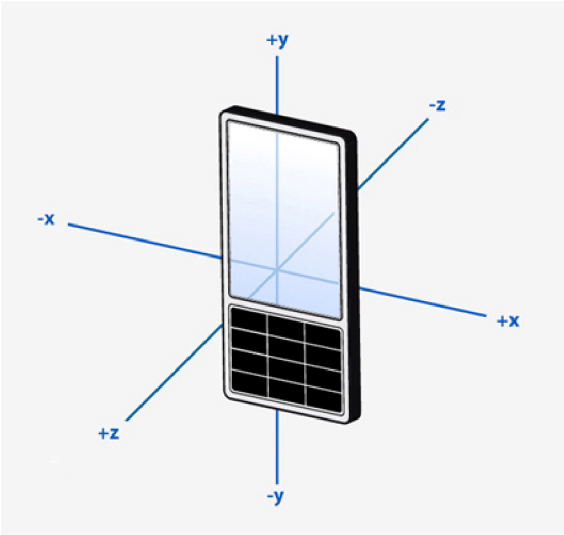
Použití senzorové třídy:

- vytvoření instance objektu QSenzor na haldě nebo zásobníku
- nastavení podle požadavků aplikace
- příjem hodnot ze senzoru
- použití senzorových dat aplikací
- ukončení přijmu hodnot ze senzoru

Preferovaný způsob čtení dat je přes předdefinované třídy na čtení (*Reading Classes*), ale ve chvíli, kdy například konkrétní třída chybí (nový senzor), je možné číst přímo přes nadřazenou třídu `QSensor`. Stačí vědět typ senzoru, název hodnoty nebo její index a typ hodnoty. Qt Mobility ve verzi 1.2 podporuje níže uvedené typy senzorů.

Akcelerometr měří vibrace a zrychlení. API poskytuje vektor zrychlení $\vec{a}$ v jednotlivých osách $\vec{a} = (x, y, z)$ podle fyzického natočení a směru pohybu přístroje (viz. obrázek 4.1 převzat z [10]). Jednotky jsou metry za sekundu na druhou, datový typ `real`. **AmbientLightSensor** neboli senzor okolního osvětlení udává v šesti úrovních hodnotu okolního osvětlení. Nejnižší 0 je určena pro neznámou úroveň, dále následuje tma, přítmí, světlo (interiérové osvětlení), jasno, slunečno (přímý sluneční svit). Výběr úrovně probíhá na základě hodnoty osvětlení v Luxech viz tabulka 4.1[1]. Pro přechod mezi úrovněmi pak platí,

[1] http://qt.gitorious.org/qt/qtsensors/blobs/b643afb7845d4fc15eb469d0d9a1ec18b0dbb254/src/plugins/sensors/generic/genericalssensor.cpp



Obrázek 4.1: Osy akcelerometru vůči poloze telefonu.

Jméno	min	max
Dark	0	5
Twilight	10	50
Light	100	200
Bright	500	2000
Sunny	5000	

Tabulka 4.1: Tabulka minimálních a maximálních hodnot pro jednotlivé úrovně QAmbientLightSensor.

že při přechodu na vyšší úroveň musí počet Luxů dosáhnout či překročit minimální hodnotu vyšší úrovně. Naopak při přechodu na nižší úroveň musí Luxy klesnout na nebo pod maximální hodnotu nižší úrovně. Datový typ je `enum`.

Compass vrací hodnotu azimutu ve stupních od magnetického severu ve směru hodinových ručiček. Datový typ je `real`. **Gyroscope** měří aktuální pohyb přístroje kolem os x, y a z. Na rozdíl od senzoru rotace (RotationSensor) se jedná o aktuální úhlové rychlosti. Jednotky jsou ve stupních za sekundu, datový typ `real`. **LightSensor** - světelný senzor měří aktuální hodnotu osvětlení v Luxech. **Magnetometer** měří složku lokálního magnetického pole v osách x, y a z. Jednotkou je Tesla, datový typ `real`. **OrientationSensor** nebo-li senzor orientace přístroje v sedmi hodnotách udává polohu přístroje, 0 opět reprezentuje neznámý stav. Následuje pořadí horní strana nahoru (TopUp), horní strana dolů (TopDown), levá strana nahoru (LeftUp), pravá strana nahoru (RightUp), čelem nahoru (FaceUp), čelem dolů (FaceDown). Pracuje v nižších vrstvách systému, nezná natočení uživatelského rozhraní ani zda je přístroj na výšku nebo šířku. Datový typ je `enum`. **ProximitySensor** -senzor přiblížení určuje, zda se k telefonu něco přiblížilo, vrací true nebo je okolo něj volný prostor, vrací false. **RotationSensor** - senzor rotace určuje natočení přístroje v prostoru. Nejdříve se měří rotace v ose z od osy y, pak rotace v ose x od aktuální osy y (jednou posunutá) k ose z a nakonec v ose y od aktuální osy z (dvakrát posunutá) k ose x. Nulový úhel na ose z je definován fixně výrobcem zařízení. Často se využívá magnetického

severu jako retenčního bodu. **TapSensor** neboli senzor dotyku registruje dotyky zařízení podél os x, y a z. Při základním nastavení zaznamenává jen dvojkliky. Vrací `enum hodnoty` dle směru a osy, viz dokumentace ke Qt Mobility.

Vzhledem k tomu, že Maemo má na rozdíl od Symbianu nebo MeeGo základní orientaci na šířku, jsou tak orientovány i senzory telefonu. To může být problém u přenositelných aplikací, které pracují s orientací displeje dle senzorů. Pak je potřeba při čtení dat na N900 brát osu x jako y a y jako x. Posloužit může podmíněný překlad `#ifdef Q_WS_MAEMO_5`, kde si vše může ošetřit programátor sám, nebo před vytvářením všech instancí senzorů použije funkci `qputenv("N900_PORTRAIT_SENSORS", "1")`, která se o to nastavením prostředí postará sama.

4.2 Použité API

V telefonu Nokia N900 se nachází akcelerometr, senzor orientace, přiblížení, okolního světla a rotace. K těmto senzorům je možné přistupovat přímo přes knihovny ovladačů nebo Qt Mobility API. Aplikace využije Qt Mobility, jejíž použití je jednodušší a elegantnější, není nutné využívat nízkoúrovňové komunikace. Sice pak není možné získat například přesné hodnoty například osvětlení v Luxech, ale rozlišovací úroveň plně dostačuje zaměření aplikace. Krom těchto senzorů je možné využít měření úrovně okolního hluku, určení lokace telefonu na základě údajů ze základové stanice mobilní sítě. K načtení údajů z mobilní sítě poslouží `QSystemNetworkInfo`.

Konkrétně tedy bude využito API pro akcelerometr, senzor orientace, okolního světla a přiblížení. Vzhledem k využití aplikace můžeme vynechat senzor rotace, není potřeba znát přesné natočení s přesností na stupně, stačí abstrakce v podobě šesti poloh telefonu ze senzoru orientace. V budoucnu by mohl sloužit k rozpoznávání různých pohybových gest.

4.3 Získání úrovně okolního hluku

K práci se zvukem je nutné použití knihoven PulseAudio. K měření hlasitosti se využívá hlavičkových a zdrojových souborů `pa_context_helper` a `pa_streams_helper` z opensource projektu Vumeter[9]. Tyto procedury a funkce jsou volány z objektu Volume, který se stará o získávání a vyhodnocení zvukových dat.

Získávání údajů probíhá přes API PulseAudia [7], nejdříve je nutné se připojit k PulseAudio serveru. K tomu slouží `pa_context_helper`, kde při vytváření kontextu vzniká hlavní smyčka s vlákny. Stavový callback čeká na úspěšné připojení. Po úspěšném vytvoření se smyčka spustí a přejde se k vytvoření a připojení se ke záznamovému streamu přes `pa_streams_helper`. Nejprve se definují vlastnosti zvukového vzorku, opět se čeká až stream přejde do aktivního stavu a nastaví se asynchronní čtení přes callback. Získaná data jsou posílána zpět objektu Volume. Tam se lineární faktor převede na hlasitost `pa_volume_t`, z ní se počítá úroveň v decibelech (bere se amplituda, ne výkon) dle vzorce

$$dB = \frac{v}{n} \cdot r \tag{4.1}$$

.

kde dB je úroveň hlasitosti v decibelech, v je softwarová hladina hlasitosti, n je normalizační koeficient a r je uživatelský rozsah. Po úspěšném čtení se aplikace odpojí od streamu a čeká na další aktualizaci senzorických dat. Odpojení od PulseAudio serveru se provádí až při ukončení aplikace.

4.4 Údaje z akcelerometru a metoda DTW

Data z akcelerometru jsou zaznamenávána pomocí asynchronního čtení vždy, když objekt `QAccelerometerSenzor` emituje signál `readingChanged`. Pro jednoduché zpracování jsou data ukládána do souboru způsobem každá hodnota na nový řádek. Čtení pak probíhá po trojicích.

S naměřenými údaji z se dá pracovat jako se signály. Díky tomu je možné využít metody používané například ve zpracování řeči i pro polohová data [3]. Několik získaných testovacích dat je pak možné porovnat pomocí metody dynamického borcení času s aktuálními daty a rozhodnout, zda odpovídají danému pohybu porovnáním, který vzor je nejblíže a zda je dostatečně blízko pro shodu.

Metoda dynamického borcení času anglicky *Dynamic Time Wrapping* dále jen DTW se nejčastěji používá při rozpoznávání mluvené řeči pro identifikaci jednotlivých slov. Většinou nebude mít vzorek stejné časování jako vzor. I jeden mluvčí nevysloví totéž slovo vždy stejně, analogicky k tomu i kroky nemá každý stejné a pohyb telefonu může trvat různou dobu. Proto se pracuje se vzdálenostmi jednotlivých vektorů. Po zaznamenání akcelerometrem bude daný pohyb reprezentován svým obrazem, tj. posloupností vektorů příznaků. Obraz referenčního pohybu se označí

$$R = \{r(1), r(2), ..., r(n) ..., r(I)\} \tag{4.2}$$

a obraz testovacího pohybu

$$T = \{t(1), t(2), ..., t(m) ..., t(J)\} \tag{4.3}$$

kde I je délka referenční sekvence R a J je délka testovací sekvence T. Definuje se obecná časová proměnná k a obě časové proměnné m a n se vyjádří jako funkce k

$$n = i\ (k) \quad k = 1, ..., K \tag{4.4}$$

$$m = j(k) \quad k = 1, ..., K \tag{4.5}$$

kde K je délka cesty, kterou se dá srovnání jednotlivých vektorů zakreslit. Její schéma je zakresleno na obrázku 4.2 převzato z [3]. Na horním grafu je průběh funkce n vzhledem k počtu kroků K. Na prostředním grafu je vidět funkce m v závislosti na počtu kroků K. Na spodním grafu je samotné porovnání obrazů v rovině (m, n). Reference je zobrazena na svislé ose, test na vodorovné.

Podmínky dané strukturou obrazů v časové ose, které se musí respektovat při hledání optimální cesty, jsou

Omezení na hraniční body Počáteční a koncové body referenčního i testovacího obrazu jsou přesně určeny a dají se vyjádřit podmínkami

$$i\ (1) = 1 \quad i\ (K) = I \tag{4.6}$$

$$j(k) = 1 \quad j(K) = J \tag{4.7}$$

Omezení na lokální souvislost a lokální strmost Pro vyloučení nadměrné komprese a expanze časového měřítka aplikujeme omezení na monotonnost a souvislost

$$0 \leq i\ (k) - i\ (k-1) \leq I^* \tag{4.8}$$

$$0 \leq j(k) - j(k-1) \leq J^* \tag{4.9}$$

V praxi se volí $I^*, J^* = 1, 2, 3$ Pro tuto aplikaci bude $I^*, J^* = 1$, takže každý vektor se musí použít alespoň jedenkrát.

Měření vzdálenosti Obecný tvar pro určení skutečné minimální vzdálenosti mezi obrazy R a T vyjadřuje vztah

$$D(R,T) = \min_{\{i\ (k),j(k),K\}} \frac{\sum_{k=1}^{K} d\left[i\left(k\right),j\left(k\right)\right] W\left(k\right)}{N\left(W\right)} \tag{4.10}$$

kde $d\left[i\ \left(k\right),j\left(k\right)\right]$ je vzdálenost mezi dvěma vektory, $W\left(k\right)$ je váhová funkce odpovídající k-tému úseku funkce DTW a $N\left(W\right)$ je normalizační faktor, který je funkcí váhové funkce.

Váhová funkce závisí pouze na lokální cestě a zde bude použita symetrická váhová funkce

$$W\left(k\right) = \left[i\left(k\right) - i\ \left(k-1\right)\right] + \left[j\left(k\right) - j\left(k-1\right)\right] \tag{4.11}$$

Normalizační faktor $N\left(W\right)$ kompenzuje délku nebo počet kroků funkce DTW

$$N\left(W\right) = \sum_{k=1}^{K} W\left(k\right) \tag{4.12}$$

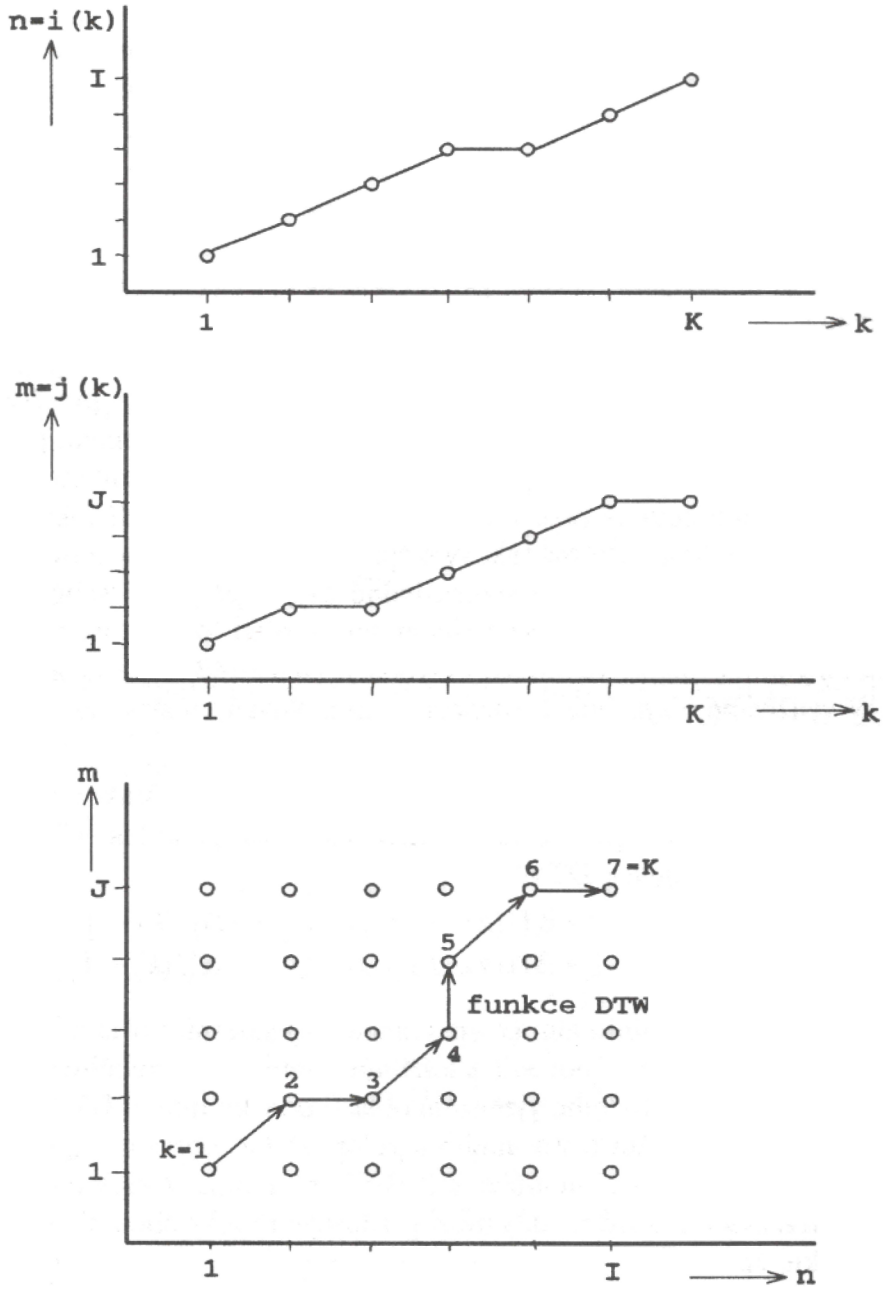
a pro symetrickou váhovou funkci bude normalizační faktor

$$N\left(W\right) = \sum_{k=1}^{K} \left[i\left(k\right) - i\ \left(k-1\right) + j\left(k\right) - j\left(k-1\right)\right] = i\ \left(K\right) - i\ \left(0\right) + j\left(K\right) - j\left(0\right) = I + J. \tag{4.13}$$

Pokud je normalizační faktor nezávislý na průběhu funkce nezávislý na průběhu funkce DTW, je možné vzorec (4.10) zjednodušit na

$$D\left(R,T\right) = \frac{1}{N\left(W\right)} \min_{\{i\ (k),j(k),K\}} \sum_{k=1}^{K} d\left[i\left(k\right),j\left(k\right)\right] W\left(k\right). \tag{4.14}$$

Pro výpočet vzdálenosti mezi jednotlivými vektory lze použít libovolnou metodu, vzhledem k zaměření této aplikace bude stačit použití Eukleidovy vzdálenosti.



Obrázek 4.2: Schéma cesty DTW.

Kapitola 5

Návrh

V této kapitole je popsán postup při návrhu aplikace, přehled v současnosti dostupných řešení. Konkrétně jsou zahrnuty aplikace N9Profile Profiles manager[18], Maemo Profiler [5], Profile changer Widget [16], ProfilesX Extended Profiles Manager [17], Timed Silencer [14] a Silencer [6]. Specifikují se požadavky této aplikace. Změna profilu se bude provádět na základě manuální aktivace, časového intervalu, rozpoznání pohybu, úrovně hluku a osvětlení, orientace telefonu a polohy. Dále je rozebrána struktura programu, rozdělení na aplikaci běžící na pozadí a aplikaci s uživatelským rozhraním včetně jejich tříd. Je zde popsán postup při návrhu uživatelského rozhraní a uvedeny příklady použití. Při návrhu se čerpalo z [1, 2].

5.1 Aktuálně dostupná řešení

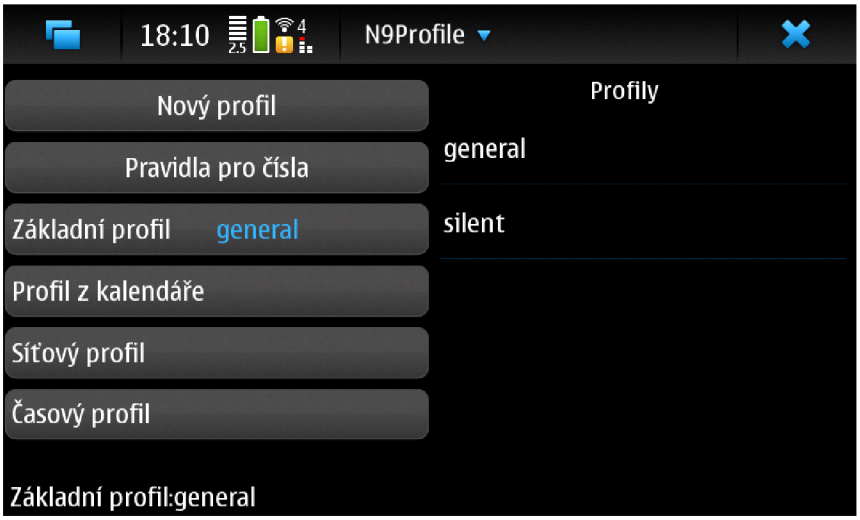
V této chvíli se správa profilů na telefonu Nokia N900 dá řešit následujícími způsoby.

Původní řešení od výrobce ve srovnání s jinými telefony i těmi bez operačního systému N900 zaostává. Má pouze dva základní profily Základní a Tichý.

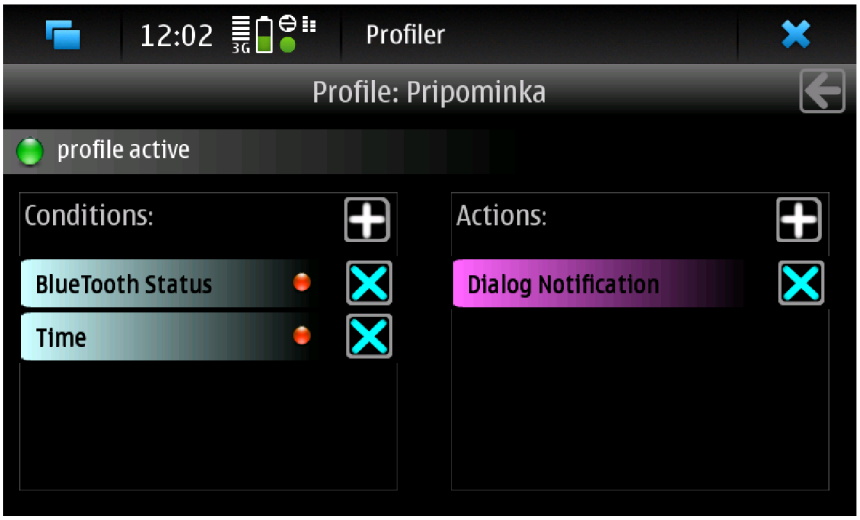
Základní poskytuje nastavení vibrací, vyzváněcích tónů a hlasitosti pro hovor, SMS, zprávu IM a email a úroveň zvuků systému, kláves a dotykového displeje. Pro Tichý existuje jediná volba a to zapnutí nebo vypnutí vibrací. Není možné vytvořit žádný další, systém nepodporuje ani načasované přepnutí z jednoho na druhý.

První aplikací pro plánování profilů je *N9Profile Profiles manager* [18]. Tato aplikace umožňuje tvorbu vlastních profilů. Jejich změna může nastat na základě času, událostí naplánovaných v kalendáři, polohy, čísla volajícího a manuálně. Při volbě podle času uživatel nastaví čas, po který má být profil aktivní. Polohu program zjišťuje podle základnové stanice BTS nebo WiFi AP. Po delší době běhu programu se objevují problémy s přepínáním profilů, neumožňuje nastavit časový interval platnosti profilu, pouze odpočet. Nevhodně zvolený název, který inklinuje spíše telefonu Nokia N9 s MeeGo než N900 s Maemem. V současné době autor na aplikaci již nepracuje a představil projekt Maemo Profiler. Náhled hlavní nabídky programu je na obrázku 5.1.

Dalším počinem téhož autora je *Maemo Profiler* [5]. Jedná se přepracovanou aplikaci N9Profile v QML/Qt. Funkce jsou rozděleny na akce a podmínky, které lze libovolně kombinovat. Ty jsou nyní řešeny pomocí Qt plugin systému a je možné přidávat svoje vlastní. Krom změny profilu tak aplikace nyní umí i zapnout displej (i trvale), poslat notifikační zprávu či vibrovat. Jako základní podmínky aplikace nabízí nabíjení baterie, stav baterie, stav Bluetooth, příchozí hovor, úroveň světla, čas a název WiFi. Projekt není dotažen



Obrázek 5.1: Hlavní nabídka programu N9Profile Profiles manager.



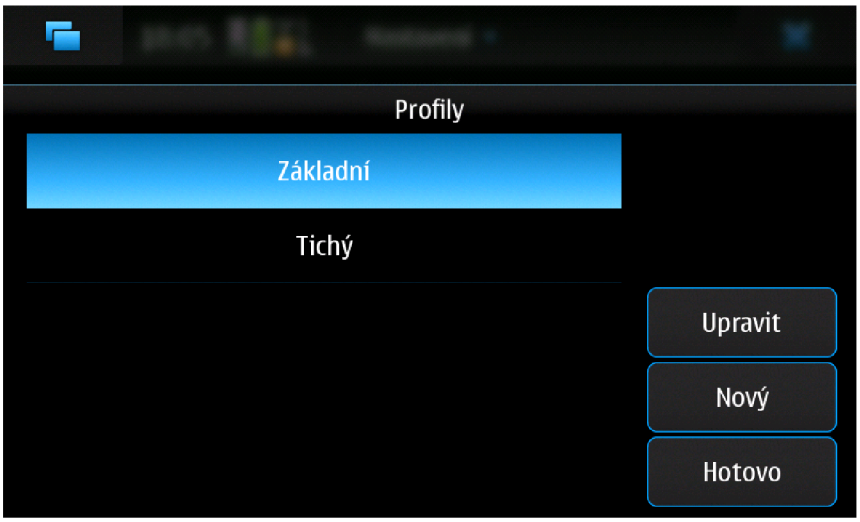
Obrázek 5.2: Maemo Profiler

do konce. V současné chvíli je vývoj zastaven a aplikace tak není plně funkční, občas padá. Nabízí přepracované a jednodušší uživatelské rozhraní. Nevyhnula se však módním ne však úplně přehledným rotujícím nabídkám a ikonám. Ukázkový příklad připomínky je na obrázku 5.2.

Aplikace *Profile changer Widget* [16] zobrazí na ploše jednoduchou ikonu. Po kliknutí na ni se profil změní. Podle aktuálního profilu se mění vzhled ikony. Uživatel může měnit i její velikost. Jak vypadá ikona pro změnu na hlasitý profil a pro změnu na tichý profil je vidět na obrázku 5.3.

ProfilesX Extended Profiles Manager [17] umožňuje k vestavěným profilům přidat vlastní. Jejich změna pak probíhá ručně, přes aplet aplikace ve stavovém oknu, který nahradí ten původní. Jednoduchá aplikace dobře plnící svůj úkol. Nápaditá možnost je přiřadit profilům různé ikonky, které se po aktivaci zobrazí ve stavové liště. Po instalaci se skryje do menu nastavení. Přehled profilů v programu ukazuje obrázek 5.4.

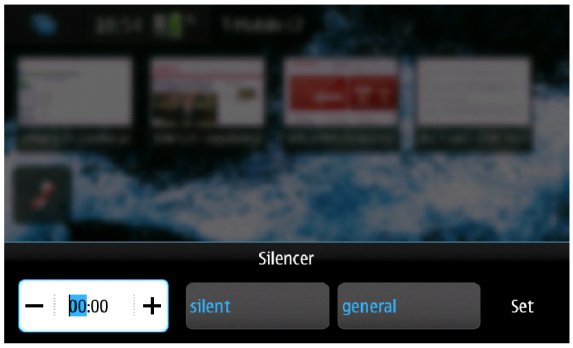
Obrázek 5.3: Widget aplikace Profile changer Widget.



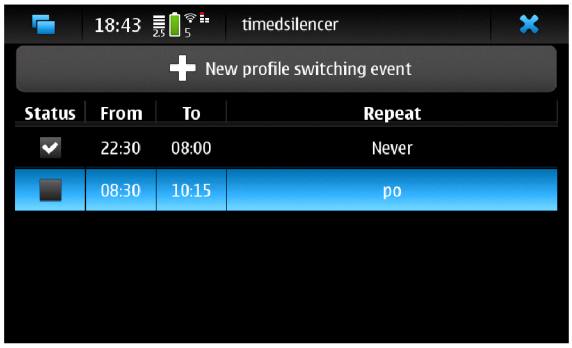
Obrázek 5.4: Aplikace ProfilesX Extended Profiles Manager.

Dvě aplikace s podobou funkcí přepínání profilů jsou *Timed Silencer* [14] a *Silencer* [6]. *Silencer* přepne aktuální profil na uživatelem zvolený po uplynutí předem daného časového úseku. Celé uživatelské rozhraní se skládá jen z dialogu pro nastavení času na obrázku 5.5.

Oproti tomu *Timed Silencer* dokáže pracovat i s časovými intervaly a dny v týdnu. Uživatel si může uložit více pravidel. Na obrázku 5.6 jsou vidět dva uložené profily. První aktivovaný a bez opakování, druhý neaktivní opakující se každé pondělí. Přepnout je možné vždy jen na tichý profil.



Obrázek 5.5: program Silencer



Obrázek 5.6: program Timed Silencer

Žádné z těchto řešení plně nepokrývá směr, kterým se moje aplikace bude ubírat. Aplikace bude integrovat některé funkce stávajících programů a navíc využije senzorických dat telefonu, konkrétně se bude jednat o změnu profilu na základě následujících možností.

Manuální aktivace uživatelem znamená, že uživatel nemusí využít automatické správy profilů a aplikaci využít jen jako rozšíření o další jím definované profily.

Při použití **časového intervalu vztahující se ke konkrétnímu dni v týdnu** si bude moci uživatel nastavit od kdy do kdy a které dny v týdnu bude profil aktivní.

Aplikace bude schopna **rozpoznání typu pohybu**. V telefonu budou zaznamenány určité způsoby pohybu. Uživatel si vybere konkrétní pohyb a aplikace porovná, kterému ze zaznamenaných pohybů je aktuální nejblíže. Pokud se bude jednat o uživatelem nastavený, provede se změna profilu.

Aplikace rozpozná **přiblížení předmětů k telefonu**. Nad displejem vedle sluchátka je umístěn senzor přiblížení, pokud se před ním bude nacházet nějaký objekt, bude profil aktivován. To může symbolizovat několik typických situací: telefon je v kapse, v kabelce či batohu.

Aplikace určí **úroveň okolního osvětlení a orientaci telefonu** Uživatel si může definovat, jaká úroveň osvětlení a poloha přístroje je pro něj vhodná a aktivuje profil. Může toho využít například při zasunování telefonu do kapsy, jestli jej nosí horní hranou nahoru či dolů.

Z mikrofonu aplikace dokáže získat **úroveň okolního hluku**. Uživatel si pak nastaví určitou hranici a odkud ji aktuální hodnota nesmí překročit – zda shora či zdola. Využití je zřejmé, čím vyšší hluk, tím vyšší hlasitost vyzvánění.

Stačí přibližná **poloha** na základě základové stanice mobilního operátora. Uživatel si uloží čísla buněk v místech, kde se často pohybuje a pak se třeba při příchodu do práce provede změna.

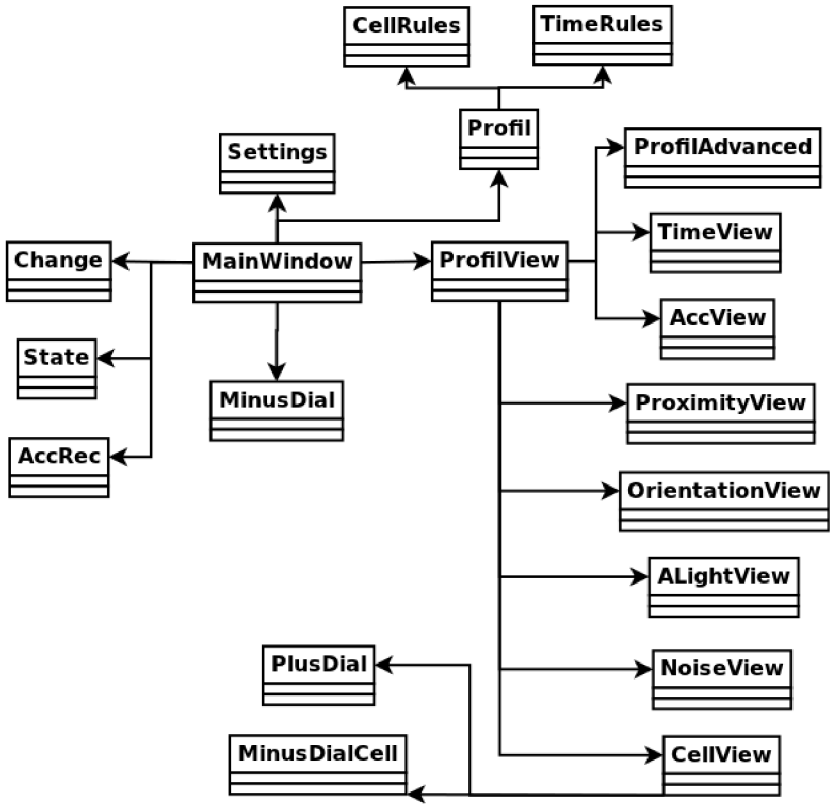
5.2 Struktura programu

Při vývoji aplikace, která pro svůj běh potřebuje vždy aktuální data, je nutné řešit, jak se bude chovat v případě, kdy ji uživatel nebude chtít mít neustále viditelně spuštěnou. Jednou z možností je reimpentovat standardní zavíraní aplikace a celou běžící aplikaci skrýt do pozadí s tím, že pro opětovné zobrazení ji stačí znovu spustit přes ikonu. K obsluze slouží D-Bus příkazy. U tohoto způsobu se obtížně řeší spouštění aplikace po startu telefonu. Druhou eventualitou je rozdělit program do dvou aplikací, jedné s grafickým uživatelským rozhraním komunikující s uživatelem a druhou konzolovou, která se stará o vyhodnocení dat a změny profilů. Obě si pak určitým způsobem vyměňují informace. Bylo rozhodnuto pro druhou možnost.

Základním objektem je profil. Každý profil má jméno, část nastavení, kde se nachází přesné nastavení zvonění, a části pravidel, kde jsou údaje o aktivovaných i pasivních pravidlech a jejich hodnotách. Profil má dva stavy, aktivovaný a neaktivovaný. Pokud je profil neaktivovaný, jednoduše při vyhodnocování přeskočí. Část nastavení se liší podle toho, jakého je profil typu. Pokud se jedná o typ ringing, telefon je ve stavu, kdy zvoní. V tomto stavu jsou v nastavení uloženy údaje o úrovních hlasitosti a melodiích jednotlivých vyzvánění (hovor, SMS, Email, IM), povolení vibrací a alarmů, kalendáře a budíku. U typu silent stačí uložit jen stav vibrací, jinak jsou všechny zvuky standardně utlumeny. Pravidla a nastavení senzorů se vždy vztahují k jednomu konkrétnímu profilu. Každý profil má určitou prioritu, která je dána pozicí v seznamu profilů. Čím nižší pořadí, tím vyšší priorita. Uživatel může prioritu měnit přesouváním konkrétního profilu. Profily pocházející přímo z telefonu mají neměnnou nejvyšší prioritu.

Základem zobrazení bude třída `MainWindow`, která zobrazí všechny uložené profily, nabídne jejich aktivaci, přidání a smazání. Na ni navazují řídící třídy`State` pro načtení a uložení do souboru,`Change` pro manuální změnu profilu,`AccRec` pro záznamy pohybu,

`Profil` pro jednotlivé profily a widgety Settings pro nastavení programu a ovládání daemona, `MinusDial` pro mazání profilů a `ProfilView` zobrazující podrobnosti profilu a jeho pravidel. Z něj potom vycházejí jednotlivé widgety pro nastavení pravidel a widget podrobným nastavením vyzvánění. Diagram tříd části s uživatelským rozhraním je na obrázku 5.7.



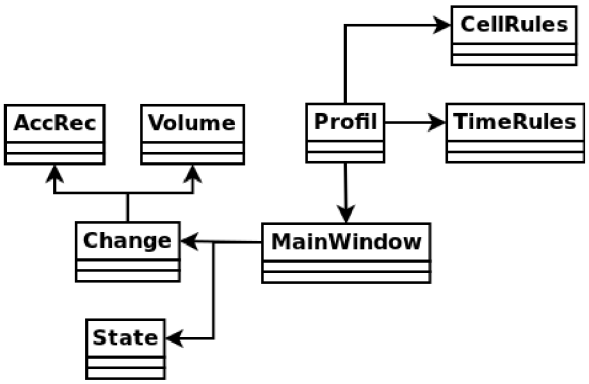
Obrázek 5.7: Diagram tříd části aplikace s uživatelským rozhraním.

Některé třídy v daemonovi jsou stejné nebo podobné třídám z klientské aplikace, `MainWindow` se stará už jen o propojení profilu s prováděcími třídami. Třída `Profil` je totožná. `Change` nyní navíc pracuje se senzory a vyhodnocuje správný profil. Novou třídou je `Volume`, která se stará o vyhodnocení hlasitosti. Diagram tříd je na obrázku 5.8.

5.3 Uživatelské rozhraní

Při návrhu uživatelského rozhraní [2] se většinou rozhodovalo mezi více variantami. U hlavní obrazovky se uvažovalo o variantě s úvodním hlavním menu, kde by byly možnosti správy profilů (přidání, mazání, editace). Později převládla myšlenka seznamu profilů, kde se volilo z více možností aktivace, zda bude dostupná již na úvodní obrazovce nebo až na detailu profilu a zda bude implementována pomocí RadioButton či CheckBox. Kvůli úspoře místa bylo rozhodnuto o zatrhávání.

U detailu profilu se opět nabízela možnost v menu rozdělení na zvonění a pravidla. Padl i nápad rozdělení na záložky. Dvě podobně jako u myšlenky menu nebo víc, kde první by byla se zvoněními a další pro každé pravidlo zvlášť. Jednu dobu převládala myšlenka zobrazit nastavení zvonění přímo, pravidla rozdělit na senzory, čas, hluk a lokaci. Vzhledem



Obrázek 5.8: Diagram tříd daemona.

k uživatelské přívětivosti bylo nastavení zjednodušeno na ticho, vibruje, zvoní, případně zvoní a vibruje. Další možnosti nastavení jsou pak skryty pod položkou rozšířené. Pravidla jsou vypsána všechna s možností aktivace.

Rozhraní nastavení akcelerometru se měnilo s upřesňováním práce se senzorem. Ze začátku převládaly nápady s nastavením pro jednotlivé osy, kde by se použily posuvník se dvěma jezdci pro určení intervalu pohybu, dva posuvníky pro nastavení minima a maxima, jeden posuvník pro nastavení změny pohybu. Když bylo rozhodnuto o sofistikovanější metodě vyhodnocování dat z akcelerometru, vybíral se nejlepší způsob výpisu záznamů a byl zvolen List Widget.

U nastavení času se také naskytla spousta různých řešení od výběru přes dvě rolety, záložky pro každý den, radio tlačítko a jeden interval. Pro implementaci bylo zvoleno řešení se všemi dny i intervaly na jedné obrazovce, které bylo vylepšeno o hromadné nastavování a zatrhávání.

U dalších senzorů se většinou rozhodovalo mezi výběrem z rolety, seznamu a radia. U senzoru přiblížení, okolního světla a orientace nakonec vyhrály radio tlačítka, kde navíc u orientace je nad každým tlačítkem obrázek dané polohy. Nastavení hluku je kompromisem a je možno jej nastavit dvěma způsoby. Pokud uživatel nechce hodnotu napsat přímo, může použít posuvník a případné drobné odchylky korigovat přes plus minus tlačítka Spin Boxu.

5.4 Případy použití

Zde jsou uvedeny ukázkové příklady profilů a přiblížení situace. Jsou součástí instalace.

Noc ticho je situace, kdy o sobě telefon nebude vůbec dávat vědět. Modelová situace: je noc, uživatel spí, nepřeje si být vůbec rušen, v místnosti je tma, ticho, telefon někde leží, je v klidu a displejem dolů.
Podmínky: záznam akcelerometru Klid, okolní osvětlení Dark, hluk pod 52 dB, telefon leží displejem dolů.

Noc vibrace umožňuje telefonu jen vibrovat. Modelová situace je podobná jako v předchozím příkladu, jen uživatel položí telefon displejem nahoru, jako znamení, že může být rušen.
Podmínky: záznam akcelerometru Klid, okolní osvětlení Dark, hluk pod 52 dB, telefon leží displejem nahoru.

Práce kancelář je případ, kdy telefon nevibruje a zvoní s poloviční intenzitou. Modelová situace: uživatel pracuje v kanceláři, moc se nehýbe, telefon většinu času leží na stole.

Na pracovišti není ticho, ale nic tam nehraje, jediný zdroj hluku je lidský hovor.
Podmínky: záznam akcelerometru Klid, senzor přiblížení Far, hluk pod 65 dB, telefon leží displejem nahoru.

Práce výroba telefonu se nastaví zvonění na plno i vibrace. Uživatel pracuje ve výrobním závodu, většinu času se pohybuje, telefon má někde zastrčený, v okolí je hluk od výrobních strojů.
Podmínky: záznam akcelerometru Chůze, senzor přiblížení Near, hluk nad 55 dB.

Při **Běh** telefon zvoní na devadesát procent, nevibruje. Uživatel běží, sportuje, dobíhá tramvaj,... Nemá cenu vibrovat, je v pohybu, necítil by to. Telefon má někde schovaný.
Podmínky: záznam akcelerometru Běh, senzor přiblížení Near.

Kapitola 6

Implementace

Tématem této kapitoly je samotná implementace programu, to je jaké byly zvoleny programátorské prostředky. Program byl rozdělen na daemona a klientskou aplikaci. Níže bude podrobněji popsán způsob automatického startu daemona, metody získání aktuálních dat ze senzorů a vyhodnocení pravidel profilů. U části aplikace s uživatelským rozhraním budou popsány úkony při startu aplikace. U jednotlivých obrazovek budou popsány jejich součásti a za nimi skryté funkce. Další prostor je věnován datové reprezentaci profilů, jejich uložení a načtení ze souboru ve formátu XML. Nakonec následuje vyhodnocení testování aplikace. U daemona bylo čerpáno z [4] u klientské aplikace pak z [1, 2].

6.1 Vývojové prostředky

Pro implementaci byl zvolen jazyk C++ a knihovny Qt a Qt Mobility. Při vývoji bylo použito vývojové prostředí Qt Creator s přímým překladem pro telefon Nokia N900. Aplikace byla testována a laděna přímo na telefonu přes ssh.

Qt Mobility je bohužel primárně navrženo pro operační systém Symbian a Maemo je spíše okrajovou platformou. To se projevuje i v API potřebných pro tuto práci. Rozhraní pro úpravy profilů telefonu se sice v Qt Mobility vyskytuje, ale počítá se se strukturou ze Symbianu, kde je profilů více než 2 a každý má více možností nastavení, než jen úroveň hlasitosti a zapnutí nebo vypnutí vibrací. Proto je nutné použít volání D-Bus.

6.2 Daemon

Daemon je konzolová aplikace běžící na pozadí, nepotřebuje přímo komunikovat s částí s grafickým rozhraním. Předávání veškerých informací probíhá přes XML soubor s uloženými pravidly. Ten je načten vždy při startu daemona. Spouští se sám při startu telefonu. K tomu slouží upstart skript `intringd`, který se při instalaci aplikace kopíruje do složky `/etc/event.d/`. Daemon se pak spouští pomocí init skriptu `intringd` ze složky `/etc/init.d/` a parametru `start`. Je nutné pomocí aplikace `su` spustit skript pod účtem běžného uživatele, aby bylo možné daemona zapínat a vypínat z uživatelského rozhraní.

Po startu se pustí časovače `timerSens`, který zajišťuje aktualizaci a vyhodnocení senzorických dat, a `timerToMidnight`, ten vyprší vždy o půlnoci. Pokud je aktivovaný některý profil jen se změnou podle času a má být aktivní v daném týdnu, spustí se dva časovače. Jeden obsluhuje aktivaci `STimer` a druhý deaktivaci `ETimer` profilu. Tyto časovače jsou

součástí každé instance třídy `Profil`. Pokud jsou aktivována i jiná pravidla, časovače se nespustí a čas je součástí standardního vyhodnocení.

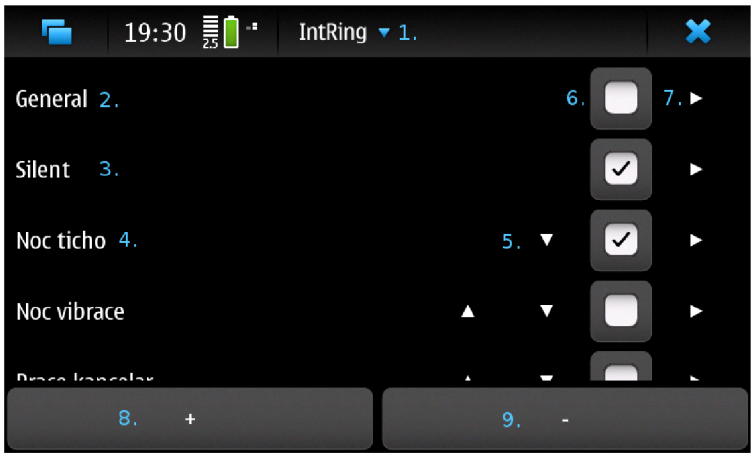
O celý proces rozhodování o změně profilu se stará metoda `handleSens`. Začíná aktivací senzorů, načtením aktuálních údajů ze senzoru přiblížení, orientace a okolního světla, zjistí se číslo buňky základové stanice, ke které je telefon připojen. Spustí se záznam údajů z akcelerometru pro rozpoznávání pohybu. Zaznamenává se 5 sekund pohybu. Uložené záznamy jsou následně porovnány pomocí metody DTW s aktuálním záznamem. Jméno záznamu, který nejvíce odpovídá aktuálnímu pohybu, se použije pro vyhodnocení. Spustí se proces zjištění úrovně okolního hluku. Vzorkovací frekvence je 25Hz a čte se po dobu 300ms. Bere se poslední vzorek, časová prodleva slouží k připojení na stream a ustálení hodnot. Po aktualizaci se senzory opět deaktivují.

Při vyhodnocování nejvhodnějšího profilu má uživatel možnost si vybrat mezi přesnou shodou a nejlepší shodou. Při přesné shodě musí u profilu pro každé aktivované pravidlo aktuální údaj odpovídat nastavenému. Nastaví se první odpovídající profil, to je profil s nejvyšší prioritou. Při nejlepší shodě se u každého profilu počítá podíl aktivovaných pravidel, kde aktuální údaj odpovídá nastavenému na všech aktivovaných pravidlech. Nastaví se profil s nejvyšším poměrem, který musí být větší než nula. Pokud se nenajde shoda, zůstane aktivovaný současný profil.

6.3 Aplikace komunikující s uživatelem

Klientská aplikace obsahuje uživatelské rozhraní, kde se nastavují jednotlivé profily a podmínky, za kterých bude jaký profil aktivován. Po spuštění načte a zobrazí uložené profily z XML souboru profiles.xml pomocí třídy `State`. Ten se nachází na velkokapacitní paměti telefonu ve skryté složce `IntRing`. Jako první dva profily načtou General a Silent z telefonu a pokud se v xml souboru nachází nějaká pravidla pro tyto základní profily, načte je. Současně načte do paměti aplikace záznamy pohybu z akcelerometru metodou `fileToMatrix`. Nachází se ve složce `acc`, která je podsložkou výše zmíněné složky `IntRing`. Každý záznam se načte do matice v samostatné instanci třídy `AccRec`. Ta poskytuje metodu pro výpočet Eukleidovy vzdálenosti `EuclidDistance` i metodu pro výpočet vzdálenosti metodou DTW `DTWDistance`. Jako jméno záznamu slouží jméno souboru s uloženými daty. Jednotlivá okna jsou do sebe zanořena pomocí `WA_Maemo5StackedWindow`. Při zobrazení jsou data vždy znovu načtena kvůli případným změnám. Stejně tak při skrytí jsou data uložena. Proto si každé okno hlídá signál `hideEvent`. Při změně nastavení profilů či podmínek senzorů se pak vše uloží pomocí metody`saveForDaemon` xml souboru a zavolá se metoda `restartDaemon`. Ta zkontroluje, jestli daemon zrovna běží, pokud ano, tak jej voláním skriptu restartuje. Tím dojde k načtení aktuálních dat. Aplikace dokáže pracovat i v režimu na výšku, ale není pro tento způsob zobrazení plně optimalizována.

Hlavní obrazovka je zobrazena na obrázku 6.1. Zobrazuje řádkově jednotlivé profily, na prvních dvou pozicích se nachází výše zmíněné profily z telefonu. Na obrázku jsou označeny čísly 2 a 3. Mají neměnnou pozici a nejvyšší prioritu, protože je uživatel používá, když není tato aplikace aktivní. Je zde však možnost je mít neaktivní (6). Každý profil je reprezentován svým jménem, jak je vidět pod číslem 4, dále zde najdeme šipky pro změnu priority uživatelských profilů skrývající se pod číslem 5, což jsou dvě instance `QToolButton` a zatržení aktivace pomocí `QCheckBox` označeného číslem 6. Na podrobnosti o profilu se dá dostat dvojím způsobem. Buďto dvojklikem na jméno profilu nebo přes kontextovou šipku na konci řádku (7). Všechny šipky jsou instancemi třídy `QToolButton`. Jméno profilu je zobrazeno pomocí třídy `ClickableLabel`, pro správnou funkci dvojkliku bylo nutné defi-



Obrázek 6.1: Hlavní obrazovka aplikace.

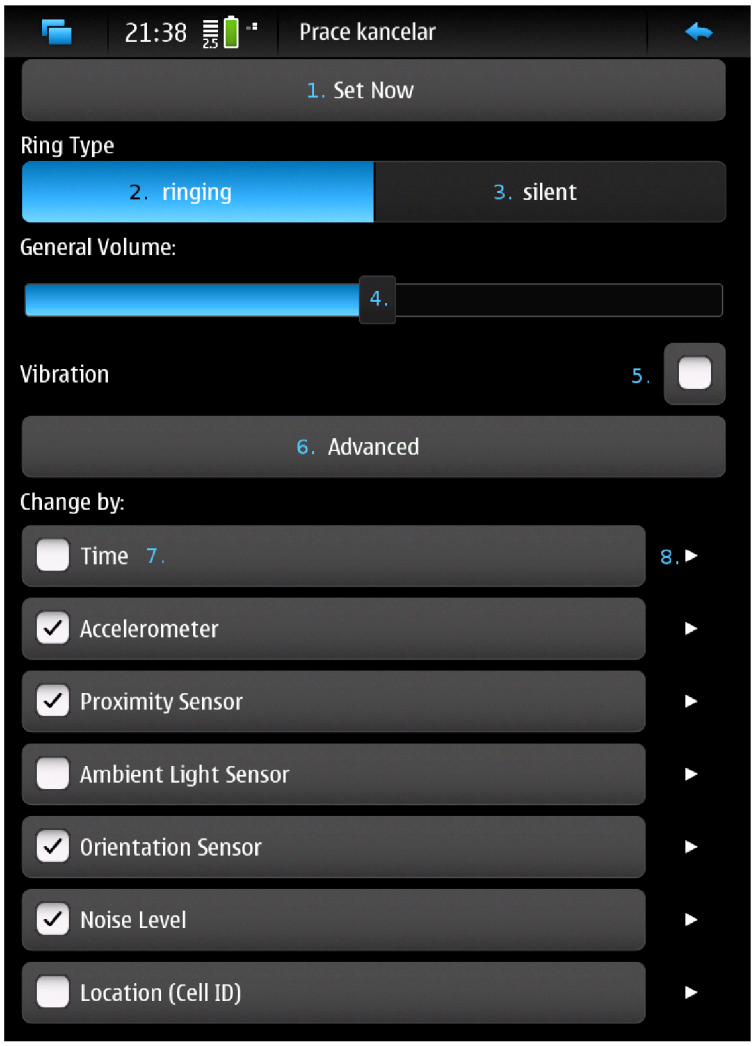
novat tuto vlastní třídu vycházející z třídy `QLabel` a podporující `mouseDoubleClickEvent`. Ve spodní části obrazovky se nacházejí tlačítka pro přidání nového profilu s číslem 8 a smazání stávajících (9). Při přidávání se zobrazí `QInputDialog` pro jméno nového profilu. Funkce mazání se zobrazuje v samostatném dialogu a podporuje smazání více profilů najednou. Přes kontextovou nabídku, která je označena číslem 1., se pak dá dostat do nastavení, kde je možné nastavit způsob rozhodování při výběru profilu, zobrazuje se zde stav daemona a je možné jej zde spustit nebo zastavit. Díky tomu, že je daemon spouštěn přes init skript, který ve složce `.IntRing` vytváří PID soubor, aplikace zjistí, zda běží. K samotnému vypnutí či zapnutí slouží `QProcess`, jenž volá skript s parametry `start` nebo `stop`.

Detailní pohled profilu nabízí rychlé nastavení chování. Jak je vidět na obrázku 6.2 pod číslem 1, uživatel může konkrétní profil okamžitě aktivovat, může se rozhodnout mezi tichem na obrázku volba 2 a zvoněním (3) s možností aktivace vibrací u obou, která má číslo 5, a nastavení celkové hlasitosti u volby zvonění označeno jako 4. Ta pak bude aplikována na všechny volby hlasitosti. Pokud by chtěl uživatel podrobnější nastavení, lze využít pokročilých nastavení tlačítkem číslo 6, kde jsou dostupné všechny volby nastavení profilu i možnost jeho přejmenování. Nedostupné možnosti pro danou volbu jsou skryty. Níže se pak již nachází jednotlivá pravidla, podle kterých se může profil řídit (7). Na detail pravidla se opět dostaneme pomocí kontextové šipky pod číslem 8.

Při časové změně je možné nastavit interval na obrázku 6.3 očíslován jako 4, kdy bude profil aktivovaný zvlášť pro každý den v týdnu (3). Nastavení probíhá přes `QtimeEdit`. Pokud konkrétní den není aktivován, nastavení intervalu je neaktivní. Pro jednoduché nastavení všech hodnot je možné využít nastavení základního intervalu na prvním řádku s číslem 2, dalším pomocníkem je tlačítko číslo 1 pro označení a odznačení všech dnů.

Jednotlivé záznamy pohybu z akcelerometru se načtou do widgetu `QListWidget` v levé části obrazovky, který vidíme pod číslem 1 na obrázku 4.1. Vpravo se nachází tlačítka pro záznam nového profilu na obrázku jako 2 a smazání aktuálně označeného profilu (3). Záznam aktuálního pohybu se spustí po potvrzení jeho jména, je řízen časovačem a o jeho průběhu a dokončení je uživatel informován patřičnými dialogy. Nový záznam se pak uloží jak do souboru, tak i do nové instance třídy `AccRec`. Program po instalaci obsahuje ukázkové záznamy klidového stavu, chůze a běhu.

Nastavení hodnoty pro senzor přiblížení, okolního osvětlení a orientace probíhá v samostatných oknech přes `QRadioButton`.



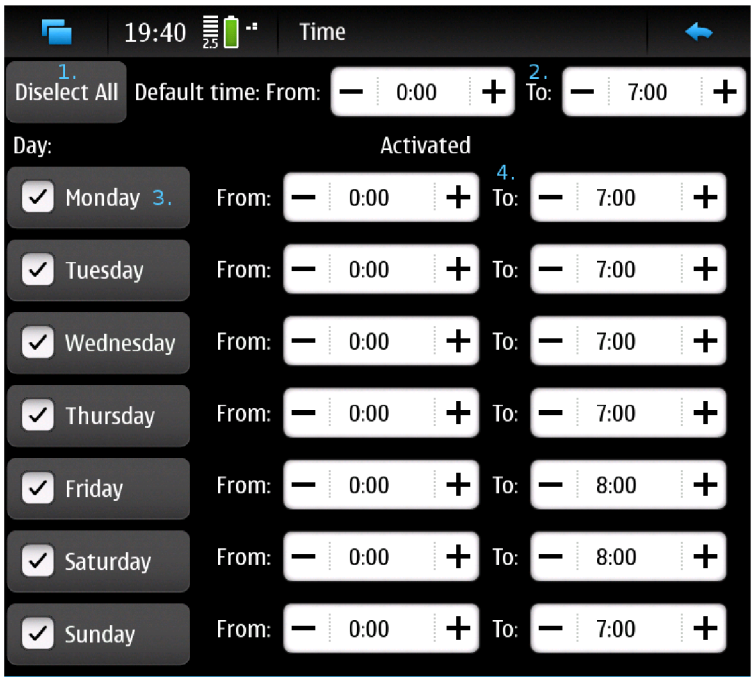
Obrázek 6.2: Detail profilu.

U hladiny okolního hluku si může uživatel vybrat, zda změřené hodnoty mají být menší, což je volba číslo 1 na obrázku 6.5, nebo větší, což je volba 2, než hodnota nastavená. K určení konkrétní hodnoty v dB pak může využít přímé zadání přes `QSpinBox` (3) nebo přes jezdce instanci `QSlider` s číslem 4.

6.4 Datová reprezentace profilu

Data aplikace se skládají z údajů o jednotlivých profilech. Mají textovou formu. Pro jejich permanentní uložení je možné použít ukládání do externího souboru nebo využít vestavění SQLite databáze telefonu. Bylo zvoleno uložení do externího souboru, kde jsou data strukturována pomocí značkovacího jazyka založeného na XML. Toto řešení umožňuje jednoduchou integraci předdefinovaných pravidel při instalaci a přenos mezi více zařízeními.

Struktura dat v uloženém souboru začíná definicí souboru typu xml, data jsou pak součástí elementu intRing s verzí formátu uložených dat, následuje nastavení typu rozhodování při výběru profilu, každý profil je jedním elementem, který je rozdělen na část nastavení a pravidel. Pokud se některá položka v pravidlech může objevit vícekrát, je každý prvek



Obrázek 6.3: Nastavení dnů a jejich intervalů pro změnu podle času.

opět samostatným elementem. Většina dat je reprezentována datovým typem int, popřípadě řetězcem, u typu vyzvánění, orientace telefonu a úrovně okolního osvětlení se symbolické konstanty z původního typu enum převedou na řetězec. V příloze B je ukázka uloženého souboru obsahující jeden profil General.

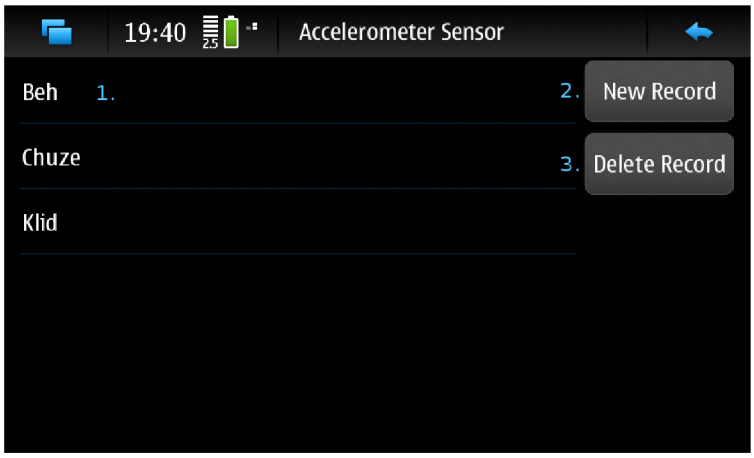
Pro načtení dat ve formátu xml slouží třída `QdomDocument`, která převádí data do stromové struktury a zjednodušuje práci s jednotlivými prvky. Vždy stačí načít kořenový element a pak je možné cyklicky projít všechny jeho potomky. Zároveň při zpracování kontroluje správnou strukturu xml souboru, v případě chyby, program zobrazí dialog s přesnou pozicí problému. Pro zápis je použito třídy `QXmlStreamWriter`, která poskytuje metody pro jednoduché vytvoření xml elementů a práci s nimi.

6.5 Testování

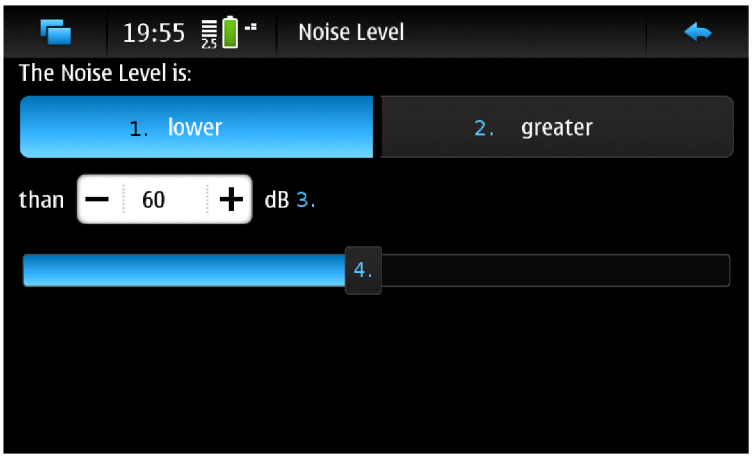
Testování aplikace lze rozdělit na dvě fáze, testování celé aplikace a testování rozpoznávání pohybů pomocí metody DTW.

Testování rozpoznávání pohybu probíhalo přímo v terénu, kdy v aplikaci bylo zapnuto ukládání porovnání uložených pohybů s aktuálním záznamem. Každý takový záznam byl opatřen časovou značkou včetně data. Mimo aplikaci pak byla zaznamenávána forma aktuálního pohybu s telefonem a interval tohoto pohybu s přesností na vteřiny. Velikost testovaného okna byla 5s, FPS záznamu 100.

Vždy po návratu bylo provedeno ruční porovnání údajů a vyhodnocení jejich správnosti. Během testování bylo získáno 500 vzorků klidového stavu, kde bylo dosaženo téměř stoprocentního rozpoznání (99.4% - 3 chybné vzorky). Z 300 vzorků chůze bylo úspěšně rozpoznáno cca 91.66% vzorků (25 chybných vzorků). Nejmenší přesnost vykázalo rozpoznávání běhu, kde opět u 300 vzorků byla úspěšnost cca 86.66% (40 chybných vzorků).



Obrázek 6.4: Nastavení konkrétního záznamu pohybu z akcelerometru.



Obrázek 6.5: Nastavení hranice hluku a vymezení se vůči ní.

Pro případné zlepšení rozpoznávacích schopností aplikace by bylo možné nahradit stávající metodu DTW přesnějšími metodami HMM nebo GMM, případně změnit velikost testovacího okna.

Při testování celé aplikace se všechny profily nejdříve zkoušely samostatně s postupným zapínáním jednotlivých pravidel, nastavením telefonu do různých podmínek, kde se testovala správná konjunkce pravidel. Následně se přešlo k aktivaci více pravidel a jejich správného vyhodnocení, testování opět probíhalo i za běžného provozu, kdy byly zaznamenávány předěly jednotlivých podmínek a činností. Chod aplikace se ukázal jako bezproblémový.

Byl také zkoušen vliv aplikace na celkovou výdrž telefonu. Aplikace běžena na telefonu, který nebyl využíván jinými aplikacemi a měl plně nabitou baterii. Výdrž v tomto stavu činila 26 hodin a 43 minut, pokud byla aplikace vypnuta, zvýšila se výdrž na 26 hodin a 56 minut. Rozdíl činí 13 minut, což vzhledem k celkově nízké výdrži telefonu a nutnosti nabíjet každý den běžné používaní telefonu nijak neomezuje.

Kapitola 7

Závěr

Cílem práce bylo navrhnout a implementovat aplikaci inteligentní správy vyzváněcích profilů pro operační systém Maemo nebo MeeGo. Vzhledem k aktuální dostupnosti hardware byla zvolna implementace pro systém Maemo. Vznikla aplikace nazvaná IntRing, která na základě údajů ze svého okolí dokáže vyhodnotit situaci a dle nastavení uživatele změnit vyzváněcí profil. Získává údaje ze senzoru přiblížení, orientace a okolního světla. Údaje z akcelerometru zpracovává na záznamy různých způsobů pohybu a porovnává aktuální pohyb s uloženými. Dále aplikace dokáže určit přibližnou úroveň okolního hluku pomocí mikrofonu telefonu nebo lokaci, kde se uživatel nachází na základě údajů o základové stanici mobilního operátora. Profil lze také měnit podle času, kdy si uživatel nadefinuje časové úseky, kdy má být profil aktivní pro jednotlivé dny v týdnu.

Při návrhu aplikace jsem se nejprve seznámil s operačním systémem Maemo 5 a mobilním telefon Nokia N900 po softwarové i hardwarové stránce se zaměřením na prvky podstatné pro tuto práci, jak je popsáno v kapitole 2. Dále jsem v kapitole 3 rozebral způsob vývoje a ladění pro rozdílnou architekturu, nástroje dostupné pro Maemo. Podrobně rozebrané získávání a zpracování aktuálních dat relevantních pro vyhodnocení změny profilu je popsáno v kapitole 4. Při návrhu jsem se seznámil s dostupnými programy podobného zaměření a specifikoval vlastnosti budoucí aplikace kapitola 5. Aplikace byla implementována za pomoci jazyka C++ a knihoven Qt. Je rozdělena na aplikaci běžící na pozadí starající se o vyhodnocení aktuálních údajů a změny profilu a aplikaci s grafickým uživatelským rozhraním, kde je možné profily a pravidla vytvářet, editovat a mazat. Samotná implementace je pak podrobně popsána v kapitole 6. Testování aplikace ukázalo její minimální vliv na výdrž telefonu a při zvolení změny podle akcelerometru, bylo v nejhorším případu dosaženo přesnosti rozpoznání cca 86.66%.

Aplikace obsahuje některé již používané prvky práce s profily, integruje ty hlavní možnosti, jako je přidávání nových profilů uživatelem, jejich manuální či časová aktivace. Snaží se uživateli zjednodušit práci s profily, kdy má ihned dostupné základní nastavení zvoní nebo nezvoní případně možnost vibrací. Do správy profilů také přináší nové prvky v podobě využití senzorů telefonu.

Cílem budoucího vývoje by mohla být opravdu inteligentní aplikace, která bude vyžadovat minimální interakci s uživatelem a bude schopna sama vyhodnotit aktuální situaci a vhodné řešení. Samotná funkčnost by se mohla rozšířit z oblasti profilů na další funkce telefonu a vytvářet pravidla, která se na jedné straně budou skládat z akcí, jako je změna profilu, statusu na IM nebo odeslaní zprávy a na druhé straně podmínek, kdy se má akce spustit např. nízký stav baterie, dosažení určité lokace, způsob pohybu. Kde modelovou situací by bylo třeba otevření nákupního seznamu po příchodu do obchodu.

Součástí práce je také demonstrační video, které ukazuje základní práci s aplikací. Je převedeno zastavení a spuštění daemona, aktivace profilu, jeho podrobné nastavení a volba pravidel. Je k dispozici na cd.

Literatura

[1] Blanchette, J.: *C++ GUI Programming with Qt 4*. Prentice Hall PTR, 2008.

[2] Molkentin, D.: *The Book of Qt 4: The Art of Building Qt Applications*. No Starch Press, 2007.

[3] Psutka, J.: *Komunikace s počítačem mluvenou řečí*. Academia Prahas, 1995.

[4] Rischpater, R.; Zucker, D.: *Beginning Nokia Apps Development*. Apress, 2010.

[5] WWW stránky: Maemo Profiler [online]. `https://gitorious.org/maemo-profiler`, 09.2.2012 [cit. 2012-05-01].

[6] WWW stránky: Silencer [online]. `http://maemo.org/packages/view/silencer/`, 12.3.2011 [cit. 2012-05-01].

[7] WWW stránky: PulseAudio 1.1 [online]. `hhttp://freedesktop.org/software/pulseaudio/doxygen/index.html`, 1.3.2012 [cit. 2012-04-30].

[8] WWW stránky: Scratchbox 1.0 [online]. `http://www.scratchbox.org/documentation/user/scratchbox-1.0/`, 14.12.2009 [cit. 2012-02-08].

[9] WWW stránky: Vumeter [online]. `https://garage.maemo.org/projects/vumeter`, 17.11.2010 [cit. 2012-04-28].

[10] WWW stránky: Qt Mobility 1.2 [online]. `http://doc.qt.nokia.com/qtmobility/`, 2011 [cit. 2012-02-03].

[11] WWW stránky: Light-to-Digital : TSL2563CL [online]. `http://www.taosinc.com/ProductDetails.aspx?id=97`, 2011 [cit. 2012-04-29].

[12] WWW stránky: LIS302DL - STMicroelectronics [online]. `http://www.st.com/internet/analog/product/152913.jsp`, 2011 [cit. 2012-04-29].

[13] WWW stránky: Maemo SDK+ User Guide [online]. `http://maemo-sdk.garage.maemo.org/user-guide.html`, 2.11.2009 [cit. 2012-03-06].

[14] WWW stránky: Timed Silencer [online]. `https://garage.maemo.org/projects/timedsilencer/`, 23.7.2010 [cit. 2012-05-01].

[15] WWW stránky: Documentation/Maemo 5 Developer Guide.
http://wiki.maemo.org/Documentation/Maemo_5_Developer_Guide/, 26.11.2011 [cit. 2012-03-03].

[16] WWW stránky: Profile Changer Widget [online].
http://talk.maemo.org/showthread.php?t=82611, 27.2.2012 [cit. 2012-05-01].

[17] WWW stránky: ProfilesX Extended Profiles Manager [online].
http://maemo.org/packages/view/profilesx/, 27.2.2012 [cit. 2012-05-01].

[18] WWW stránky: N9Profile Profiles manager [online].
http://maemo.org/packages/view/n9profil/, 29.7.2010 [cit. 2012-05-01].

[19] WWW stránky: MeeGo Architecture [online].
https://meego.com/developers/meego-architecture/, 29.9.2010 [cit. 2012-02-03].

Příloha A

Obsah CD

CD obsahuje:

- text této práce ve formátu pdf
- zdrojové texty a obrázky této práce v jazyce LaTeX
- zdrojové soubory aplikace s grafickým uživatelským rozhraním a aplikace běžící na pozadí
- instalační balíček deb pro telefon Nokia N900
- demonstracní video

Příloha B

Konfigrační soubor

Ukázka souboru XML s jedním uloženým profilem.

```
<?xml version="1.0"?>
<!DOCTYPE xml>
<intRing version="1.0">
    <bestMatch>0</bestMatch>
        <profil>
        <name>General</name>
        <active>1</active>
        <settings>
            <calendarAlarmEnabled>1</calendarAlarmEnabled>
            <clockAlarmEnabled>1</clockAlarmEnabled>
            <emailAlertTone>/usr/share/sounds/Emailalert1.aac</emailAlertTone>
            <emailAlertVolume>100</emailAlertVolume>
            <imAlertTone>/usr/share/sounds/Emailalert1.aac</imAlertTone>
            <imAlertVolume>100</imAlertVolume>
            <keyboardSoundLevel>0</keyboardSoundLevel>
            <ringingAlertTone>/media/mmc1/ANGRY BIRDS_FINAL MASTERED.mp3</ringingAlertTo
            <ringingAlertType>ringing</ringingAlertType>
            <ringingAlertVolume>100</ringingAlertVolume>
            <smsAlertTone>/usr/share/sounds/Message1.aac</smsAlertTone>
            <smsAlertVolume>100</smsAlertVolume>
            <systemSoundLevel>1</systemSoundLevel>
            <touchscreenSoundLevel>1</touchscreenSoundLevel>
            <vibratingAlertEnabled>1</vibratingAlertEnabled>
        </settings>
        <Rules>
            <timeActive>0</timeActive>
            <timeRule>
                <active>0</active>
                <startTime>00:00:00</startTime>
                <endTime>23:59:00</endTime>
            </timeRule>
            <timeRule>
                <active>0</active>
```

```
    <startTime>00:00:00</startTime>
    <endTime>23:59:00</endTime>
</timeRule>
<timeRule>
    <active>0</active>
    <startTime>00:00:00</startTime>
    <endTime>23:59:00</endTime>
</timeRule>
<timeRule>
    <active>0</active>
    <startTime>00:00:00</startTime>
    <endTime>23:59:00</endTime>
</timeRule>
<timeRule>
    <active>0</active>
    <startTime>00:00:00</startTime>
    <endTime>23:59:00</endTime>
</timeRule>
<timeRule>
    <active>0</active>
    <startTime>00:00:00</startTime>
    <endTime>23:59:00</endTime>
</timeRule>
<timeRule>
    <active>0</active>
    <startTime>00:00:00</startTime>
    <endTime>23:59:00</endTime>
</timeRule>
<proximityActive>1</proximityActive>
<proximityClose>1</proximityClose>
<ambientLightActive>0</ambientLightActive>
<ambientLight>Undefined</ambientLight>
<orientationActive>0</orientationActive>
<orientation>Top Up</orientation>
<cellActive>0</cellActive>
<cellRule>
    <active>1</active>
    <name>Doma</name>
    <cellID>3217</cellID>
</cellRule>
<cellRule>
    <active>0</active>
    <name>Prace</name>
    <cellID>246331</cellID>
</cellRule>
<noiseActive>1</noiseActive>
<noiseLow>0</noiseLow>
<noiseLevel>55</noiseLevel>
```

```
            <accelerometerActive>1</accelerometerActive>
            <accelerometerRecordName>Chuze</accelerometerRecordName>
        </Rules>
    </profil>
</intRing>
```