VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELIGENT SYSTEMS

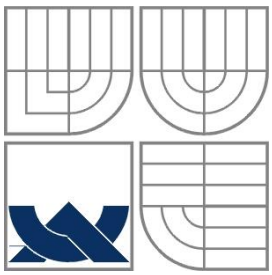
BEZDRÁTOVÁ SENZOROVÁ SÍŤ NA PLATFORMĚ ANDROID

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

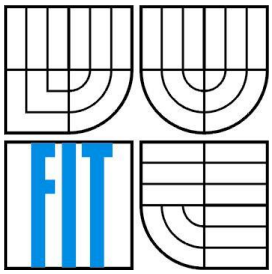
AUTOR PRÁCE
AUTHOR

TOMÁŠ BĚLOHOUBEK

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELIGENT SYSTEMS

BEZDRÁTOVÁ SENZOROVÁ SÍŤ NA PLATFORMĚ ANDROID

WIRELESS SENSOR NETWORK BASE ON ANDROID PLATFORM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — TOMÁŠ BĚLOHOUBEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. SAMEK JAN, Ph.D.
SUPERVISOR

BRNO 2013

Abstrakt

Současná mobilní zařízení s operačním systémem Android v sobě obsahují množství senzorů, bezdrátových modulů a prostředků pro zjišťování polohy a stávají se zajímavé pro využití v oblasti senzorových sítí. Práce se zabývá zkoumáním možnosti těchto zařízení a jejich praktickým využitím v oblasti bezdrátových senzorových sítí. Z těchto poznatků vychází návrh a implementace vlastní aplikace.

Abstract

Mobile devices on Android platform includes many sensors, wireless modules and methods for determining the position. This makes them interesting for use in area of sensor networks. This thesis examines the potential of these devices and their practical use in wireless sensor networks. Design and implementation of the custom application is based on these findings.

Klíčová slova

Senzorová síť, operační systém Android, emulátor, senzorový uzel, bezdrátová komunikace, určování polohy, server

Keywords

Wireless network, operating system Android, emulator, sensor node, wireless communication, positioning, server

Citace

Bělohoubek Tomáš: Bezdrátová senzorová síť na platformě Android, bakalářská práce, Brno, FIT VUT v Brně, 2013

Bezdrátová senzorová síť na platformě Android

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jana Samka, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Bělohoubek
9.5.2013

Poděkování

Na tomto místě děkuji vedoucímu mé bakalářské práce, panu Ing. Janu Samkovi, Ph.D, za ochotu a poskytnutí mnoha cenných rad v průběhu psaní této práce.

Obsah

Obsah 1
1 Úvod 3
1.1 Struktura práce 3
2 Bezdrátové senzorové sítě 4
2.1 Důležité vlastnosti sítě 4
2.1.1 Typy sensorů, jejich přesnost a rychlost snímání 4
2.1.2 Aktuálnost zasílaných informací 4
2.1.3 Schopnost vyrovnávání se s výpadky 4
2.1.4 Energetické nároky uzlů 5
2.1.5 Možné topologie sítě 5
2.2 Zařízení na platformě Android jako senzor 5
3 Operační systém Android 6
3.1 Vývoj aplikací 6
3.1.1 Aplikační rozhraní systému 6
3.1.2 Oprávnění 7
3.1.3 Android software development kit (SDK) 7
3.1.4 Android debug bridge (ADB) 8
3.1.5 Android emulátor 9
3.1.6 Vývojové prostředí 11
4 Způsoby bezdrátové komunikace 12
4.1 Bluetooth 12
4.2 Wi-Fi Direct 13
4.3 NFC 13
4.4 Síť Internet 14
4.4.1 Wi-Fi 14
4.4.2 Datové přenosy 15
4.5 SMS 15
5 Senzory mobilních zařízení 16
5.1 Poloha 16
5.1.1 GPS 16
5.1.2 Network location provider 17
5.2 Senzory ostatních veličin 17
5.2.1 Práce se senzory v OS Android 18
5.2.2 Dostupnost jednotlivých senzorů 18

5.2.3 Pozice 19
5.2.4 Pohyb 21
5.2.5 Prostředí 22
6 Návrh vlastní sítě 23
6.1 Požadované vlastnosti 23
6.2 Návrh řešení 23
6.3 Komunikační protokol 24
6.3.1 Klient - server 24
6.3.2 Zařízení – zařízení 26
6.4 Mobilní klient 26
6.4.1 Nastavení aplikace 26
6.4.2 Zaznamenávání senzorů a informací o poloze 26
6.4.3 Komunikace se serverem 27
6.4.4 Komunikace mezi zařízeními 28
6.4.5 Služba běžící na pozadí 28
6.4.6 Uživatelské rozhraní 29
6.5 Server 30
6.5.1 Aplikační rozhraní pro komunikaci s klientem 31
6.5.2 Uchování dat 31
6.5.3 Webové prostředí 32
7 Implementace sítě 35
7.1 Mobilní klient 35
7.1.1 Logická struktura aplikace 35
7.1.2 Použité nástroje a prostředky 36
7.2 Serverová část 37
7.2.1 Použité nástroje a frameworky 37
7.3 Zhodnocení výsledků 38
7.3.1 Problémy při implementaci 38
7.3.2 Možné rozšíření aplikace 40
8 Závěr 41
Literatura 42
Seznam příloh 43
Příloha 1 44

1 Úvod

Žijeme v digitální době. Naučili jsme se využívat počítače a internet nám poskytuje nezměrné množství informací. Přístup k internetu se stal samozřejmostí a my jsme začali pociťovat potřebu, mít tyto informace stále k dispozici. Tato potřeba stála za vznikem a rychlým vývojem mobilních telefonů tak jak je známe dnes. Představa, že budeme žít své životy spjaty se zařízeními, které nám umožní komunikovat s blízkými, navigovat nás po cestě do práce, najdou nám restauraci nebo nás v dlouhé chvíli zabaví online videem, či moderní hrou, byla před lety považována za hudbu vzdálené budoucnosti. Vývoj však ubíhá mílovými kroky a v současnosti nám dělají společnost komplexní zařízení. V honbě za uživatelským komfortem a inteligentním chováním jsou tato zařízení vybavena mnohými senzory, pomocí kterých dokáží odhadnout aktuální situaci a přizpůsobit se (náklon obrazovky, jas displeje atd.). Dokáží komunikovat s okolím pomocí bezdrátových modulů nebo zjistit vaši polohu s přesností na jednotky metrů. Masové rozšíření zapříčinilo, že ceny těchto zařízení klesly natolik, že za cenu mnohdy nepřevyšující 3000Kč máme k dispozici výkonný stroj vybavený čidly, a GPS přijímačem který je schopný komunikace s okolím. A právě díky své komplexnosti a velice nízké ceně jsou tato zařízení zajímavá pro oblast bezdrátových senzorových sítí.

1.1 Struktura práce

Druhá kapitola této práce seznamuje čtenáře s problematikou bezdrátových sítí a důležitými vlastnostmi takovéto sítě. Zabývá se přínosem zařízení s OS Android v této oblasti.

Kapitola třetí přibližuje pojem OS Android a vytváření aplikací pro tento systém. Seznamuje čtenáře s aplikačními rozhraními a bezpečnostními mechanizmy operačního systému. Dále popisuje nástroje, které jsou potřebné pro vývoj aplikace.

Kapitola čtvrtá prozkoumává možnosti mobilních zařízení na platformě Android v oblastech bezdrátové komunikace. Shrnuje vlastnosti jednotlivých způsobů komunikace a přibližuje možnosti příslušných aplikačních rozhraní.

Kapitola pátá shrnuje teoretické poznatky o práci se senzory a metodami určování polohy v OS Android. Kapitola popisuje funkce podporovaných senzorů a možnosti jejich aplikačního rozhraní. Dále kapitola popisuje dostupné možnosti zjišťování polohy zařízení a vlastnosti těchto metod.

Poznatky z předchozích kapitol jsou využity v kapitole 6 pro návrh vlastní senzorové sítě. Je zde popsána struktura sítě, komunikační protokol a vlastnosti jednotlivých senzorových uzlů. Kapitola dále popisuje návrh aplikací klienta a webového serveru.

V kapitole 7 je popsána implementace našeho návrhu. Jsou zde zmíněny použité nástroje a knihovny při vývoji. V této kapitole jsou také zmíněny poznatky a praktické zkušenosti s vývojem vlastní sítě. Na závěr je výsledný stav a vlastnosti vlastní sítě zhodnoceny spolu s návrhem na možné budoucí rozšíření aplikace.

2 Bezdrátové senzorové sítě

Bezdrátové senzorové sítě neboli WSN[1] slouží k zaznamenávání a uchování informací z různých senzorů (uzlů). Jsou využívány jak v domácnostech, tak i v mnoha odvětvích průmyslu. Bezdrátová senzorová síť je tvořena množstvím uzlů, která zaznamenávají potřebné informace a předávají je dál pomocí bezdrátové technologie. Zpravidla jsou napájeny baterií. Samotná zařízení nemusí obsahovat složité obvody pro zpracování informací, pouze prostředky pro jejich zaslání. Tímto lze docílit nízkých nákladů na pořízení jednotlivých senzorů. Síť obsahuje jedno, nebo více zařízení, která slouží ke sběru a zpracování informací z okolních senzorů. Tato zařízení také komunikují s okolním světem [1].

2.1 Důležité vlastnosti sítě

Požadované vlastnosti bezdrátových senzorových sítí se liší dle způsobu použití a určení sítě. Pro tuto práci však můžeme uvažovat následující základní vlastnosti.

2.1.1 Typy sensorů, jejich přesnost a rychlost snímání

Každý sensor se vyznačuje svojí přesností měření a rychlostí snímání dané veličiny. Je zapotřebí zvolit sensor, který bude vyhovovat našim potřebám.

2.1.2 Aktuálnost zasílaných informací

Některé senzorové uzly snímají hodnoty měřené veličiny a ukládají si je do vlastní paměti. Informace z této paměti periodicky zpracovávají nebo zasílají do dalšího zařízení v rámci senzorové sítě. Některé aplikace senzorových sítí kladou důraz na aktuálnost zasílaných informací. Proto je zapotřebí volit tuto periodu v souladu s požadavky sítě. Snížení periody má za následek větší zpoždění zasílaných informací, avšak projevuje se nižšími energetickými nároky.

2.1.3 Schopnost vyrovnávání se s výpadky

V bezdrátových sítích může docházet k výpadkům komunikace mezi jednotlivými uzly. Mohou být způsobeny rušením, překážkami mezi zařízeními nebo výpadkem funkčnosti okolního zařízení. V případě neúspěšného odeslání informace musí být daný uzel schopen si data uchovat do té doby, než ji bude moci opět odeslat. Požadavky na síť mohou tolerovat jistou míru výpadků. V případě, že tomu tak není, je zapotřebí navrhnout senzorový uzel s dostatečně velkou vnitřní pamětí, kde se budou moci data dočasně ukládat.

[1] Wireless sensor networks [1]

2.1.4 Energetické nároky uzlů

Další důležitou vlastností senzorové sítě je požadovaná provozní doba senzorového uzlu na jedno nabití baterie. Tomuto požadavku se přizpůsobuje jak kapacita baterie, tak i provoz samotného uzlu. Za cílem dosažení delší výdrže uzlu může být například omezena bezdrátová komunikace, která se na energetické náročnosti zařízení často projevuje nejvíce.

2.1.5 Možné topologie sítě

Při návrhu sítě zvažujeme možné způsoby propojení jednotlivých prvků mezi sebou. Senzor může zasílat informace přímo na server nebo do centrálního uzlu s konektivitou, který data předává dál. Data však nemusí být zasílána do centrálního uzlu přímo, ale přes jiný uzel nebo mohou být zaslána více možnými cestami pro zvýšení spolehlivosti sítě.

2.2 Zařízení na platformě Android jako senzor

Současné chytré telefony a obecně mobilní zařízení obsahují množství různých senzorů, modulů pro bezdrátové komunikace, metod pro zjišťování polohy a mnohých dalších možností. V kombinaci s dostatečnou kapacitou baterie a mnohdy velice příznivou cenou se stávají zajímavými pro oblast bezdrátových senzorových sítí.

Možnosti použití těchto zařízení jsou rozsáhlé. Dostupný operační systém Android nabízí prostředky pro rychlý vývoj potřebné aplikace. Pro vývojáře jsou k dispozici aplikační rozhraní pro práci s dostupnými senzory, bezdrátovými moduly a dalšími prostředky systému.

3 Operační systém Android

Operační systém Android je systém určený pro mobilní zařízení. Je vyvíjen společností Google a šířen jako open-source. Historicky první zařízení s tímto systémem bylo uvedeno v roce 2008 (T-mobile G1). V současnosti se jedná o nejrozšířenější operační systém mezi mobilními zařízeními.

Nejprve byl dostupný na chytrých mobilních telefonech, později však tento systém pronikl do mnoha různých přístrojů (čtečky knih, chytré televizory), kde je implementován někdy s četnými úpravami [2].

Obr. 3.1: Logo operačního systému Android [3].

3.1 Vývoj aplikací

OS Android je postaven na linuxovém jádře, které bylo optimalizováno pro mobilní zařízení. Nativní aplikace jsou psány v Jazyce Java a mívají k dispozici četné množství knihoven a aplikačních rozhraní, pomocí kterých komunikují se systémem a jeho prostředky. Aplikace jsou interpretovány pomocí nástroje Dalvik virtual machine. Díky zvolené technologii nejsou aplikace omezené na jednu architekturu. V současnosti je systém k vyvíjen pro architektury ARM, x86 a MIPS [3].

3.1.1 Aplikační rozhraní systému

Operační systém Android nabízí Aplikační rozhraní, pomocí kterého mohou aplikace komunikovat se samotným OS a jeho prostředky. S vývojem OS jde ruku v ruce vývoj jeho aplikačního rozhraní. Pro odlišení verzí aplikačního rozhraní slouží číselný identifikátor s názvem API Level. Při vývoji aplikace musíme v souboru `AndroidManifest.xml` specifikovat, jakou verzi aplikačního rozhraní naše aplikace využívá.

Verze OS	API Level	Název	Oficiální představení
4.2.x	17	Jelly Bean	Říjen 2012
4.1.x	16	Jelly Bean	Červen 2012
4.0.3, 4.0.4	15	Ice Cream Sandwich	Prosinec 2011
4.0, 4.0.1, 4.0.2	14	Ice Cream Sandwich	Říjen 2011
3.2	13	Honeycomb	Červen 2011
3.1.x	12	Honeycomb	Květen 2011
3.0.x	11	Honeycomb	Únor 2011
2.3.3, 2.3.4	10	Gingerbread	Únor 2011
2.3, 2.3.1, 2.3.2	9	Gingerbread	Listopad 2010
2.2.x	8	Froyo	Červen 2010
2.1.x	7	Éclair	Leden 2010
2.0.1	6	Éclair	Prosinec 2009
2.0	5	Éclair	Listopad 2009
1.6	4	Donut	Září 2009
1.5	3	Cupcake	Květen 2009
1.1	2	Beze jména	Únor 2009
1.0	1	Beze jména	Říjen 2008

Tabulka 3.1: Historie verzí aplikačního rozhraní [3].

3.1.2 Oprávnění

Součástí OS Android a jeho aplikačních rozhraní jsou bezpečnostní opatření pro běh aplikace. Jestliže vyžadujeme použití prostředků operačního systému nebo jiných aplikací, které podléhají bezpečnostním opatřením, musí naše aplikace zažádat o oprávnění k daným prostředkům přistupovat. Samotná oprávnění mohou omezovat přístupu k datům uživatele, komunikaci po síti internet či omezovat přístup k mnohým jiným prostředkům. Oprávnění, o které aplikace žádá, vývojář specifikuje v souboru `AndroidManifest.xml`.

3.1.3 Android software development kit (SDK)

Pro vývoj je k dispozici Android software development kit (SDK), který vývojáři poskytuje v kombinaci s vývojovým prostředím všechny potřebné prostředky pro vývoj mobilní aplikace. Tyto nástroje jsou dostupné pro OS Windows, Linux i Mac OS.

Android software development kit obsahuje následující [3]:

- knihovny aplikačních rozhraní pro přístup k prostředkům operačního systému,

- vývojářské nástroje pro překlad a ladění aplikace,
- Android emulátor (AVD),
- nástroj pro ladění na reálném zařízení (ADB),
- úplná dokumentace,
- ukázkové kódy,
- plugin pro vývojové prostředí Eclipse (ADT, Android developer toolkit).

Platforma se stále vyvíjí a s ní i aplikační rozhraní. V důsledku tohoto faktu jsou na trhu zařízení s různými verzemi systému, nástup nových verzí bývá velice pozvolný. Vývojář proto má dostupné nástroje pro vývoj na všech verzích OS Android. V grafickém prostředí má vývojář možnost stáhnutí potřebných součástí SDK. Těmito součástmi mohou být vzorové aplikace, zdrojové kódy, ovladače, virtuální obrazy jednotlivých verzí systému atd.

3.1.4 Android debug bridge (ADB)

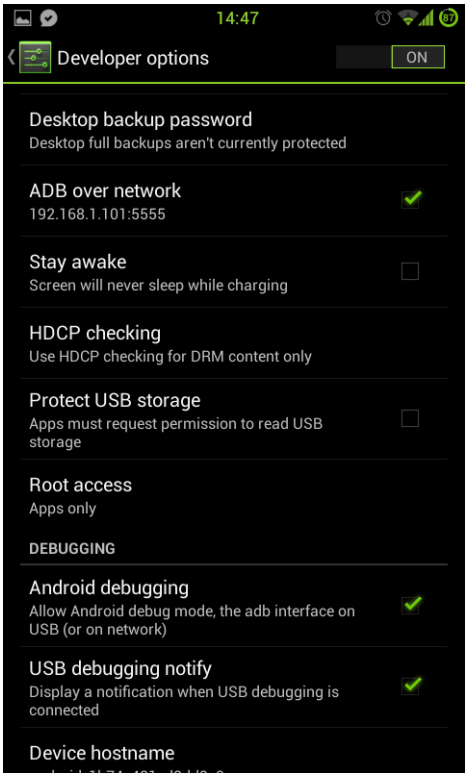
Jedná se o nástroj sloužící ke komunikaci s připojenými zařízeními s OS Android nebo s virtuálními zařízeními spuštěnými na hostitelském počítači. Je nezbytnou součástí pro vývoj a především ladění aplikací. Umožňuje manipulaci se soubory na připojeném zařízení, instalaci aplikací, poskytování ladících výstupů a mnoho dalšího. Nástroj je ovládán z příkazové řádky. Některé jeho příkazy, jako například nahrání aplikace nebo výpis ladících informací, však lze spouštět přímo z vývojového prostředí (Eclipse s ADT). Nástroj ADB se nachází ve složce platform-tools v SDK. Android debug bridge se skládá ze tří částí [3]:

- **Server** - služba na pozadí, běžící na vývojářském počítači. Zprostředkovává komunikaci mez klientem a Daemonem běžícím na zařízení nebo na emulátoru.
- **Klient** - konzolová aplikace na počítači. Slouží k zadávání příkazů a ovládání ADB (příkaz adb).
- **Daemon** – služba spuštěná na připojeném zařízení nebo na zařízení virtuálním.

Fyzická zařízení mohou být připojena k počítači pomocí USB kabelu, nebo bezdrátově pomocí Wi-Fi. Pro úspěšnou komunikaci přes USB musí být v zařízení aktivována možnost Ladění (Settings – Developer options – Android debugging). V počítači dále musejí být nainstalované patřičné ovladače. Zařízení řady Nexus (referenční zařízení vyvíjená ve spolupráci se společností Google) mají tyto ovladače k dispozici ke stažení v samotném SDK. Pro ostatní je nutné tyto ovladače vyhledat na webových stránkách výrobce. Po připojení kabelem je zařízení automaticky detekováno a registrováno v nástroji ADB.

Možnost spojení pomocí bezdrátové sítě Wi-Fi je oficiální cestou dostupné v zařízeních s OS Android ve verzi 4.2 a novější. Pro zařízení s nižší verzí systému je zapotřebí instalace dodatečné aplikace. Pro bezdrátové spojení je nutné, aby zařízení i počítač byli připojeni ke stejné síti. V zařízení musí být aktivní ladění, a dále aktivována možnost ladění přes síť (Settings – Developer options –

ADB over network). Následně se počítač připojí k zařízení příkazem `adb connect <ip>`. Pro kontrolu a zobrazení všech připojených zařízení (přes USB i síť) použijeme příkaz `adb devices`. V počítači není zapotřebí instalace dodatečných ovladačů jako v případě připojení pomocí USB [3].

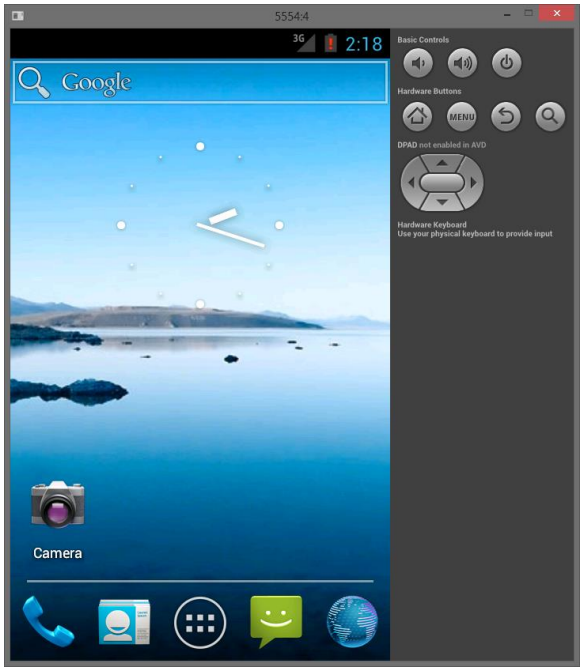


Obr. 3.2: Snímek obrazovky nastavení telefonu.

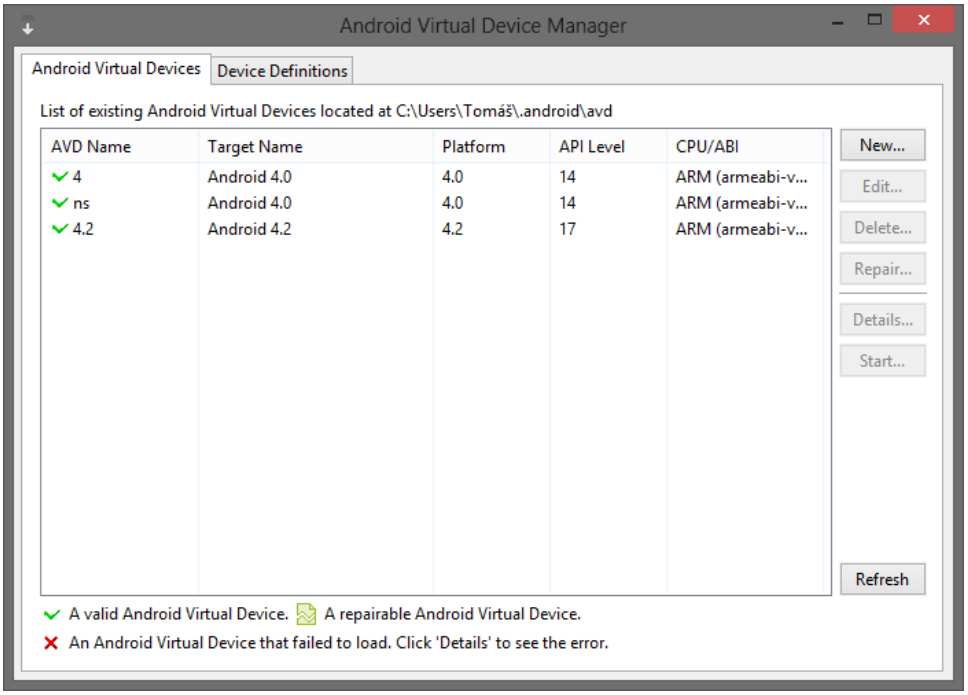
3.1.5 Android emulátor

Pro případ že nemáme k dispozici zařízení s požadovanými parametry (velikost displeje, rozlišení, verze OS a další) máme k dispozici Emulátor. Tento nástroj nám umožní běh virtuálního zařízení a ladění aplikací na tomto stroji. Takto spuštěné zařízení je automaticky registrováno v ADB. V Emulátoru je implementována podpora pro komunikaci po síti avšak ne pro senzory a bezdrátové moduly. Proto je zapotřebí provádět rozsáhlejší testování na reálných zařízeních.

Pro správu virtuálních zařízení slouží Android virtual device manager (AVD). V jeho prostředí lze vytvářet nová virtuální zařízení s požadovanými parametry. Dále umožňuje spravovat existující. Z tohoto prostředí je možné jednotlivé stroje také spouštět.



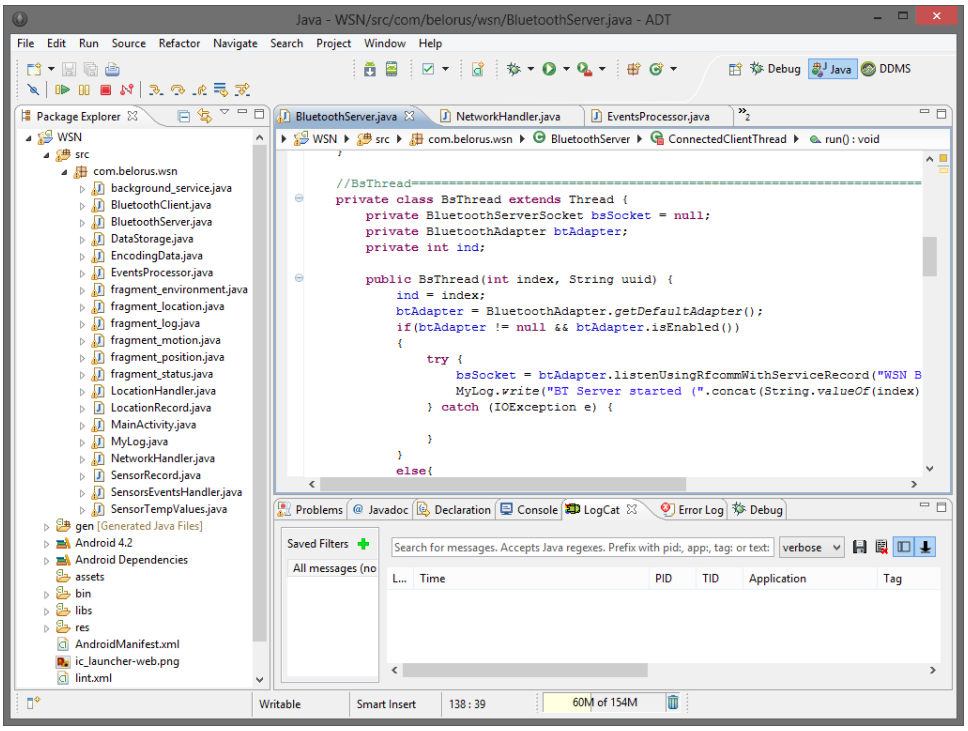
Obr. 3.3: Spuštěný emulátor s běžícím OS Android 4.0.



Obr. 3.4: AVD s výběrem virtuálních strojů.

3.1.6 Vývojové prostředí

Android software development kit nabízí doplněk pro open source vývojové prostředí Eclipse. Toto prostředí je doporučováno pro snadnější konfiguraci, dostupné návody a podporu ze strany Google. Vývojář má k dispozici nástroje pro vývoj v jazyce Java a vytváření vzhledu aplikace pomocí XML. Dále mu prostředí umožňuje překlad a ladění aplikace na zařízeních pomocí ADB.



Obr. 3.5: Prostředí Eclipse s otevřeným projektem.

4 Způsoby bezdrátové komunikace

Zařízení s OS Android jsou zpravidla zařízení vysoce mobilní. Proto disponují mnohými moduly pro bezdrátové komunikace, které umožňují komunikaci v síti internet nebo s jinými zařízeními nebo periferiemi v blízkosti. Tyto bezdrátové technologie se často výrazně liší svými vlastnostmi a jejich určením.

4.1 Bluetooth

Technologie Bluetooth je určena pro bezdrátovou komunikaci mezi zařízeními na krátké vzdálenosti. Využívá rádiové pásmo 2,4GHz. Tato technologie byla vytvořena společností Ericsson v roce 1994 jako bezdrátová alternativa k sériovému rozhraní RS-232. Popisuje ji standard IEEE 802.15.1 a v současnosti je dostupná ve verzi 4.0 [4].

Jednotlivé verze se liší rychlostí přenosu, energetickou náročností, teoretickým dosahem, možnostmi šifrování a dalšími. Teoretický dosah technologie se pohybuje v rozmezí 10-100m a maximální rychlost přenosu dat u Bluetooth verze 4.0 je stanovena na 24Mbit/s. Od Bluetooth 2.1 je šifrování komunikace povinné pro všechny protokoly s výjimkou SDP (Service discovery protocol).

Pro popis komunikačního rozhraní mezi zařízeními určují Bluetooth profily. Tyto profily slouží pro specifická zařízení a aplikace, jako například výměna souborů, síťová komunikace, sériový přenos dat, headset, stereo sluchátka, zdravotnická zaříení a mnoho dalších [4]. Pro používání Bluetooth v OS Android je zapotřebí oprávnění `BLUETOOTH` a `BLUETOOTH_ADMIN`.

Bluetooth aplikační rozhraní dostupné v OS umožňuje následující činnosti [3]:

- přistupovat k již připojeným zařízením,
- vyhledávat okolní zařízení,
- vytvářet nová spojení,
- spravovat více připojení,
- přenášet data sériovou komunikací,
- komunikace zařízení pomocí socketů,
- práce s profily Headset, A2DP, Health Device,
- možnost vlastní implementace podpory ostatních Bluetooth profilů.

Moduly Bluetooth v mobilních zařízeních jsou často velice omezené. Výkon těchto modulů umožňuje stabilní spojení s několika zařízeními, avšak s přibývajícími připojeními spolehlivost klesá.

Pro úspěšnou komunikaci pomocí Bluetooth musejí, až na výjimky, být obě komunikující zařízení spárovány. Párování je proces, kdy se zařízení vzájemně autentizují a předají si základní informace. Součástí párování je i předání klíčů pro šifrovanou komunikaci.

Jedno z komunikujících zařízení zastává roli serveru a druhé roli klienta. Server otevře nový kanál, na kterém očekává nové příchozí spojení. Následně se klient k danému kanálu připojí a započne samotná komunikace. Kanály jsou označovány jedinečným identifikátorem UUID, který musí být známý pro obě komunikující zařízení. Server na každém kanálu může obsluhovat současně pouze jedno zařízení. Jestliže naše aplikace vyžaduje současné spojení s více klienty, je zapotřebí obsluhovat více kanálů, každý s vlastním UUID. Klient poté zkouší který kanál je volný a na ten se připojí. Samotné připojování a odpojování zařízení jsou operace náročné na systémové zdroje, proto je vhodné udržovat jedno spojení po celou dobu komunikace. V případě ukončení spojení operace může trvat jednotky vteřin. Po tuto dobu je kanál blokovaný a pro případnou novou komunikace je nutné vyčkat, nebo použít kanál jiný.

Pro využívání služeb Bluetooth modulu v OS Android musí být aplikaci přiřazeno oprávnění službu využívat. Oprávnění týkající se Bluetooth jsou rozděleny do dvou úrovní. Oprávnění `BLUETOOTH` povoluje aplikaci vytvářet spojení a přenášet data. Oprávnění `BLUETOOTH_ADMIN` umožňuje vyhledávání okolních zařízení nebo manipulaci s nastavením samotného adaptéru [3].

4.2 Wi-Fi Direct

Wi-Fi Direct je standard, umožňující vzájemnou komunikaci zařízení prostřednictvím sítě Wi-Fi bez nutnosti dostupného přístupového bodu. Tato technologie umožňuje komunikaci na větší vzdálenosti a vyšší rychlostí nežli technologie Bluetooth, avšak za cenu vyšších energetických nároků. Rychlosti a dosah odpovídají technologii Wi-Fi. Wi-Fi Direct aplikační rozhraní je v OS Android dostupné od API level 14. Pro využití služeb Wi-Fi Direct jsou nutná oprávnění `ACCESS_WIFI_STATE`, `CHANGE_WIFI_STATE`, `CHANGE_NETWORK_STATE`, `INTERNET` a `ACCESS_NETWORK_STATE` [3].

Wi-Fi Direct aplikační rozhraní dostupné v OS umožňuje následující činnosti [3]:

- detekovat okolních zařízení podporujících Wi-Fi Direct,
- zaslat žádost na spojení a spojení přijmout,
- vytvářet skupiny připojených zařízení,
- přenášet data pomocí socketů.

4.3 NFC

NFC (Near field communication, ISO/IEC 18092) je technologie sloužící pro přenos malých objemů dat na krátké vzdálenosti. Teoretická maximální pracovní vzdálenost zařízení je 20cm, v praxi je však tato vzdálenost menší (typicky okolo 4cm). Ke komunikaci se využívá rádiového pásma 13,56MHz a rychlosti přenosu této technologie jsou od 106kbit/s po 424kbit/s [5].

Technologie je založena na standardech RFID. Na rozdíl od tohoto umožňuje i oboustrannou komunikaci zařízení. Komunikace může probíhat mezi dvěma aktivními, nebo jedním aktivním a jedním pasivním zařízením, kterým může být například bezkontaktní RFID karta. Zařízení mohou komunikovat pomocí NDEF (NFC Data Exchange Format), nebo RAW zpráv. NFC aplikační rozhraní je v OS Android dostupné od API level 9. Pro využívání NFC je nutno zažádat o oprávnění `NFC` [3].

NFC aplikační rozhraní dostupné v OS umožňuje následující činnosti [3]:

- čtení NDEF dat z NFC tagů,
- zasílání NDEF zpráv z jednoho zařízení druhému dotykem (Android Beam),
- čtení a zasílání RAW zpráv.

4.4 Síť Internet

K síti Internet se mobilní zařízení připojují pomocí datových přenosů v síti operátora, popřípadě pomocí technologie Wi-Fi v závislosti na dostupnosti dané technologie. V OS Android jsou tato připojení spravována třídami v balíku android.net. Pro umožnění aplikaci komunikovat po síti Internet, je zapotřebí zažádat o oprávnění `INTERNET` a `ACCESS_NETWORK_STATE` [3].

Balík `android.net` mimo jiné umožňuje [3]:

- zjišťovat způsob připojení k internetu a stav sítě,
- komunikovat pomocí socketů,
- pracovat se zabezpečením spojení,
- procházet statistiky připojení a přenesených dat,
- připojit se k síti VPN.

4.4.1 Wi-Fi

Tato technologie slouží pro bezdrátovou komunikaci zařízení pomocí počítačové sítě. Technologii Wi-Fi definují standardy 802.11. Pro komunikaci se využívá frekvenčního pásma 2,4GHz a 5GHz.

Teoretické rychlosti přenosu mohou dosahovat hodnot až 300Mbit/s v případě 802.11n (54Mbit/s pro 802.11g) v závislosti na konfiguraci sítě a možnostech zařízení. Dosah sítě závisí na výkonu a směrovosti antén, zástavbě prostředí, zarušení používaného pásma a dalších aspektech. Data přenášená po síti mohou být šifrována pomocí následujících protokolů: WEP, WPA, WPA2 [6].

Správu Wi-Fi připojení zajišťuje balík `android.net.wifi`. Mimo jiné, tento balík umožňuje [3]:

- spravovat stav rozhraní,
- detekce okolních sítí a připojení k nim,
- přistupovat k informacím o aktuálním připojení (zabezpečení, rychlost, kvalita připojení a další).

4.4.2 Datové přenosy

Mobilní zařízení mají k dispozici technologie pro přístup k síti internet prostřednictvím sítě mobilního operátora. Dostupnost těchto technologií a jejich rychlost je dána možnostmi samotného zařízení a infrastrukturou dostupné sítě.

Možné je připojení pomocí následujících technologií (v závorce maximální teoretické rychlosti) [7]:

- GPRS (171kbit/s),
- EDGE (384kbit/s),
- HSDPA (14Mbit/s),
- a další.

4.5 SMS

SMS (Short message service) je služba umožňující zasílání a přijímání krátkých textových zpráv prostřednictvím sítě operátora. Tyto zprávy mají standartní délu 160 znaků. Obsluhu těchto zpráv v OS Android zajišťuje třída `SmsManager` a umožňuje [3]:

- zaslat datovou zprávu na specifický aplikační port,
- zaslat textovou zprávu o standartní délce,
- zaslat delší textovou zprávu rozdělenou na zprávy standartní délky.

5 Senzory mobilních zařízení

Výčet a popis jednotlivých senzorů v této kapitole vychází z informací ze zdroje [3].
Zařízení s OS Android disponují mnoha senzory. Zpravidla jsou k dispozici senzory, které usnadňují práci se samotným zařízením, jako například senzor okolního osvětlení pomocí kterého je možno regulovat jas displeje nebo čidla orientace umožňující přetočení obrazovky zařízení v závislosti na způsobu držení zařízení. Některá zařízení však obsahují i další čidla, která nemají v samotném systému významnější uplatnění a jsou zde k dispozici pro vývojáře a aplikace třetích stran.

5.1 Poloha

Operační systém Android nabízí určení polohy daného zařízení třemi způsoby, a to systémem GPS, nebo zjišťováním polohy dle dostupných Wi-Fi sítí v okolí a dle pozice v síti operátora. Tyto metody se liší jak přesností určení polohy, tak rychlostí zjištění této informace nebo energetickou náročností.

Zjištění polohy v OS Android zajišťuje balíček `android.location`, který mimo jiné umožňuje [3]:

- detekce dostupných možností zjišťování polohy,
- zjišťování polohy vybraným způsobem,
- zadat podmínky pro zjištění polohy,
- údaje o přesnosti metody a o energetických nárocích,
- zjištění nadmořské výšky, rychlosti,
- přístup k poslední známé poloze bez nutnosti nového měření.

5.1.1 GPS

Global Positioning System. Jedná se o veřejně dostupný satelitní navigační systém. Tuto službu provozuje ministerstvo obrany Spojených Států Amerických. Systém je založen na družicích, které periodicky vysílají zprávy o přesné pozici a čase v daný moment. Přijímací zařízení tyto zprávy zpracovává a vyhodnocuje svoji polohu podle vypočtené vzdáleností od viditelných družic.

Pro určení polohy je zapotřebí viditelnost s alespoň čtyřmi družicemi. Přesnost navigačního systému ovlivňuje počet viditelných družic, atmosférické podmínky v dané lokalitě a citlivost přijímacího zařízení. Podle podmínek je přesnost udávána v jednotkách až desítkách metrů. Prvotní zjištění polohy však může trvat minuty. Toto je způsobeno detekcí pozic satelitů a dalších informací. Další určování polohy je však okamžité.

Tato metoda určování polohy je ze všech dostupných nejpřesnější. Je však mnohonásobně náročnější na spotřebu elektrické energie.

Asistovaná GPS

Tato technologie slouží pro urychlení prvotního zjištění polohy zařízení. Některá data, jako například polohy družic, nejsou zjišťována z družic samotných ale pomocí dostupného připojení k internetu. Tímto lze dosáhnout prvotního zjištění polohy v jednotkách vteřin. Dále Asistovaná GPS umožňuje zpřesnění výpočtu polohy díky dostupnosti informací o aktuálních atmosférických podmínkách a dalších informací, které nelze zjistit z družic.

5.1.2 Network location provider

Network location provider je souhrn metod zjišťování polohy založen na informacích z okolí jako například okolní sítě Wi-Fi a vysílače mobilních operátorů v okolí. Tyto metody slouží k rychlému a energeticky nenáročnému určení přibližné polohy.

Cell-ID

Tato metoda určuje pozici zařízení dle jeho polohy v síti mobilního operátora. Každý vysílač má určeno své jednoznačné číslo (Cell ID). Zařízení komunikující v této síti má informace o dostupných vysílačích v okolí a dle těchto informací lze určit přibližnou polohu. Ve městech je vyšší koncentrace vysílačů a proto zde lze dosáhnout vyšší přesnosti nežli v méně obydlených oblastech. V závislosti na místě lze dosáhnout určení polohy s přesností desítek metrů až jednotek kilometrů.

Tato metoda je ze všech uvedených nejméně přesná, na druhou stranu je nejméně energeticky náročná a dostupná vždy, když je k dispozici signál mobilního operátora.

Wi-Fi

Metoda určování pozice pomocí Wi-Fi je obdobná metodě zjišťování pomocí Cell ID. Zařízení vyhledává dostupné sítě Wi-Fi a podle nalezených sítí v okolí a intenzity jejich signálů určí svoji polohu. Tímto způsobem lze dosáhnout přesnosti na desítky metrů avšak pouze se zapnutou Wi-Fi a v oblastech kde jsou dostupné Wi-Fi sítě. Energetická náročnost samotného hledání je velmi nízká.

5.2 Senzory ostatních veličin

V této kapitole jsou zmíněny senzory zpravidla fyzikálních veličin. Některé nalezneme v převážném množství zařízení, některé jsou implementovány pouze zřídka. Jsou zde zmíněny veškeré senzory podporované platformou android. Programová obsluha následujících senzorů je do jisté míry totožná a proto jsou obsluhovány jednotným aplikačním rozhraním.

5.2.1 Práce se senzory v OS Android

Pro přístup k informacím ze všech dostupných senzorů slouží Android Sensor Framework. Tento Framework je součástí balíčku `android.hardware` a umožňuje následující činnosti [3]:

- zjištění přítomnosti určitého senzoru, seznam všech dostupných senzorů v zařízení,
- přístup k informacím ze senzorů,
- přístup k informacím o přesnosti senzoru, rozsahu, výrobci, energetických nárocích, času měření atd.
- ošetřování událostí, jakými mohou být například změna měřené hodnoty nebo změna přesnosti.

Podporované senzory mohou být dvojího typu, a to hardwarové, nebo softwarové. Hardwarové senzory interpretují přímo informace z fyzického senzoru. Softwarové svoji hodnotu odvozují, nebo vypočítávají na základě hodnot jednoho nebo více ostatních senzorů.

Po zaregistrování senzoru v aplikaci vyvolává daný senzor událost, pokaždé když dojde ke změně měřené veličiny. Vývojář má k dispozici kompletní informace o stavu senzoru v daný okamžik společně s informací o čase, kdy byla změna zaznamenána. Zde je nutné poznamenat, že zachytávání informací a množství předávaných informací se může lišit v závislosti na zařízení. Některé senzory mohou vyvolávat událost jen při změně. Některé v pravidelných intervalech, i když ke změně nedošlo. Také se může stát, že senzor nepodává kompletní informace, kdy například může chybět informace o čase.

5.2.2 Dostupnost jednotlivých senzorů

S vývojem OS Android a jeho aplikačního rozhraní se měnila podpora různých typů senzorů v systému. V jednotlivých zařízeních však tyto senzory nejsou vyžadovány a mohou být implementovány dle rozhodnutí výrobce. Proto je nutné i přes vhodnou verzi operačního systému kontrolovat dostupnost senzorů v zařízení. V současnosti jsou podporovány níže uvedené senzory.

V tabulce 5.1 jsou uvedeny jednotlivé senzory a jejich podpora v různých verzích aplikačního rozhraní.

Senzor	Verze systému Android (API level)			
	4.0 (14)	2.3 (9)	2.2 (8)	1.5 (3)
Pozice				
Magnetické pole	Ano	Ano	Ano	Ano
Orientace	Ano	Ano	Ano	Ano
Proximity sensor	Ano	Ano	Ano	Ano
Pohyb				
Akcelerometr	Ano	Ano	Ano	Ano
Gravitace	Ano	Ano	Ne	Ne
Gyroskop	Ano	Ano	Ne	Ne
Lineární akcelerometr	Ano	Ano	Ne	Ne
Rotace	Ano	Ano	Ne	Ne
Prostředí				
Okolní teplota	Ano	Ne	Ne	Ne
Intenzita osvětlení	Ano	Ano	Ano	Ano
Tlak	Ano	Ano	Ne	Ne
Relativní vlhkost	Ano	Ne	Ne	Ne
Teplota zařízení	Ano	Ano	Ano	Ano

Tabulka 5.1: Dostupnosti jednotlivých senzorů [3].

5.2.3 Pozice

Magnetické pole

Senzor magnetického pole měří intenzitu okolního magnetického pole ve třech osách (X, Y, Z). Udávané hodnoty veličiny jsou v jednotkách uT (mikro Tesla). Osy X, Y, a Z jsou lokálního charakteru, relativní k displeji zařízení.

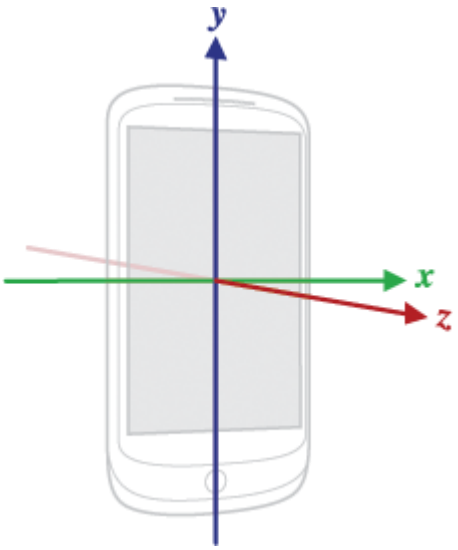
Tento senzor je hardwarový a lze ho nalézt ve většině zařízení. Společně s akcelerometrem poskytuje například informace pro senzor orientace zařízení.

Orientace

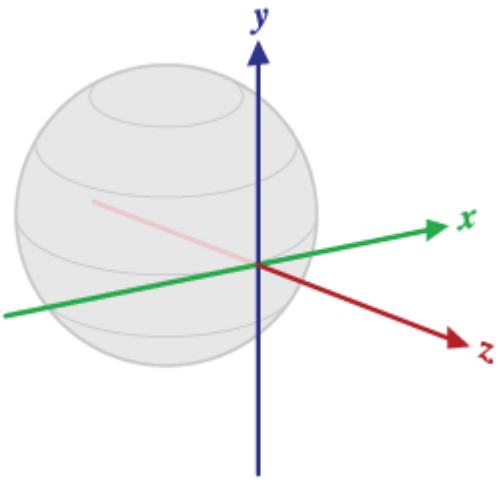
Tento senzor zjišťuje orientaci pomocí detekce rotace zařízení okolo os X, Y a Z. Osa Z směřuje do centra země. Ostatní osy jsou na z kolmé. Osa x směřuje na západ a osa y přímo k severnímu pólu. Předávány jsou údaje o rotaci okolo těchto tří os v jednotkách stupňů.

Rotace okolo osy Z je udávána v rozmezí 0°-359° a přímo určuje směr, kterým zařízení směřuje (sever = 0°, východ = 90°, jih = 180°, západ = 270°). Rotace okolo osy X se udává v rozmezí -180° až 180° a rotace okolo osy Y v hodnotách -90° až 90°.

Senzor je softwarový a pro vyhodnocování používá informací ze senzoru magnetického pole a akcelerometru. Tento senzor je k dispozici z historických důvodů a je zastaralý. Místo práce s ním se doporučuje použití funkcí `getRotationMatrix()`, `remapCoordinateSystem()` a `getOrientation()`.



Obr. 5.2: Lokální osy zařízení [3].



Obr. 5.1: Osy čidla orientace [3].

Proximity sensor

Jedná se o zařízení detekující blízkost okolních předmětů u zařízení. Samotné čidlo bývá zpravidla umístěno na přední straně a snímá tak prostor v okolí displeje. Výstupní hodnota udává vzdálenost předmětu v centimetrech. Obvyklé rozmezí zjistitelných vzdáleností bývá 0-5cm. Některá čidla jsou schopna zjistit pouhou přítomnost předmětu bez informace o vzdálenosti. V tomto případě senzor předává nejnižší hodnotu v případě detekovaného předmětu. Jestliže není předmět detekován, je vrácena hodnota maximální.

Tento senzor je hardwarový a slouží například k vypnutí displeje při přiložení telefonu ke tváři v průběhu hovoru.

5.2.4 Pohyb

Akcelerometr

Toto zařízení slouží k měření zrychlení ve třech osách. Osy jsou lokálního charakteru relativní k obrazovce zařízení.

Hodnoty udávané senzorem obsahují gravitační zrychlení, proto zařízení v klidu bude vykazovat zrychlení rovno gravitačnímu a zařízení v pádu zrychlení nulové. Hodnoty zrychlení jsou udávány v jednotkách m/s^2. Jedná se o hardwarový senzor a bývá přítomen ve většině zařízení.

Gravitace

Tento senzor měří hodnotu gravitačního zrychlení a jeho směr. Výsledné hodnoty jsou předávány stejným způsobem jako u akcelerometru, tedy zrychlením v jednotlivých osách relativních k displeji v jednotkách m/s^2. Může se jednat buďto o hardwarový senzor, nebo i softwarový.

Gyroskop

Gyroskop slouží k zaznamenávání úhlové rychlosti okolo os X, Y a Z. Tyto osy jsou lokálního charakteru a odpovídají osám akcelerometru. Předávané informace jsou udávány v rad/s.

Tento senzor je hardwarový a bývá přítomen v mnohých novějších zařízení. Funkce gyroskopu však může vykazovat chyby a nepřesnosti. Tyto nepřesnosti proto bývají kompenzovány pomocí dat z ostatních senzorů.

Lineární akcelerometr

Pro měření zrychlení bez vlivu gravitace slouží senzor lineární akcelerace. Měřené hodnoty jsou předávány totožným způsobem jako u akcelerometru.

Tento senzor může být hardwarový nebo softwarový, přičemž vždy platí vztah, že hodnoty akcelerometru jsou rovny součtu hodnot lineárního akcelerometru a senzoru gravitace.

Rotační vektor

Senzor rotačního vektoru udává rotaci zařízení okolo svých os. Osy jsou totožné s osami senzoru magnetického pole. Předávané informace jsou bezrozměrné. Senzor bývá softwarový, ke své funkci využívá informací z ostatních senzorů.

5.2.5 Prostředí

Tyto senzory umožňují zaznamenávat informace o okolním prostředí. Všechny jsou hardwarové, bývají však zřídka kdy implementovány. Výjimku tvoří senzor okolního osvětlení.

Okolní teplota

Senzor zaznamenává okolní teplotu a předává ji ve stupních celsia.

Intenzita osvětlení

Senzor zaznamená okolní osvětlení a předává informaci v jednotkách lux. Tento senzor bývá často implementován a umisťován do okolí displeje a využívá se pro regulaci jeho jasu.

Tlak

Senzor zaznamenává atmosférický tlak a hodnotu předává v jednotkách hPa.

Relativní vlhkost

Senzor zaznamenává relativní vlhkost a hodnotu předává v procentech.

Teplota zařízení

Senzor zaznamenává teplotu zařízení a předává ji ve stupních celsia. Jeho přítomnost a umístění se však liší zařízení od zařízení.

6 Návrh vlastní sítě

Tato kapitola se zaměřuje na návrh a implementaci vlastní senzorové sítě. Nejprve se zaměříme na požadované vlastnosti našeho řešení. Navrhneme způsob komunikace společně s komunikačním protokolem. Popíšeme implementaci aplikace mobilního klienta a serverové části. Závěr kapitoly se poté zabývá možnými dalšími rozšířeními.

6.1 Požadované vlastnosti

Senzorová síť bude monitorovat informace ze senzorů na mobilních zařízeních. A to v reálném čase. Síť se bude skládat z nejméně tří přístrojů. Snímané veličiny jednotlivých senzorových uzlů budou přehledně zobrazeny ve vizuální aplikaci.

6.2 Návrh řešení

Senzorová síť je tvořena zařízeními s klientskou aplikací a serverem dostupným ze sítě internet. Senzorové uzly mohou komunikovat přímo se serverem nebo informace zasílat prostřednictvím jiného uzlu v blízkosti. Tato vlastnost je potřebná pro případ, kdy pouze omezené množství zařízení v síti disponuje internetovou konektivitou. Zvolený způsob komunikace umožňuje také informace zasílat pomocí několika zřetězených uzlů, čehož může být využito v případě, kdy dané zařízení nedisponuje internetovou konektivitou a ve svém dosahu takovéto zařízení nemá. Data proto zašle jinému uzlu, který je přepošle dále. Jestliže klient nemůže informaci předat, uchovává si ji do té doby, nežli bude opět schopný komunikace.

Klientská aplikace umožňuje volbu způsobu komunikace, kdy si uživatel zvolí, zda bude uzel komunikovat přímo se serverem, nebo pomocí jiného zařízení. Dále aplikace zobrazuje aktuální hodnoty snímaných veličin a jednoduchý log. V logu je zaznamenán stav a činnosti daného uzlu společně s případným hlášením o chybách.

Obr. 6.1: Logo aplikace.

Server přijímá data ze senzorových uzlů a ukládá je do své databáze. Umožňuje také tyto informace zobrazovat ve webovém rozhraní. Toto prostředí je dostupné ze sítě internet. Pro přístup ke grafickému rozhraní je zapotřebí znalost uživatelského jména a hesla. Tímto je sníženo riziko zneužití informací nepovolanými osobami. Součástí rozhraní jsou grafy hodnot a informace o poloze vykreslené na mapě.

6.3 Komunikační protokol

Nedílnou součástí návrhu senzorové sítě je komunikační protokol, pomocí kterého se jednotlivé uzly dorozumívají a pomocí kterého komunikují se serverem. Následující část je, z důvodu mírných odlišností, rozdělena na popis komunikace klienta se serverem a popis komunikace uzlů mezi sebou.

6.3.1 Klient - server

Komunikace klientů se serverem prostřednictvím sítě internet je zajišťována protokolem HTTP[2]. Jedná se o nejrozšířenější aplikační protokol používaný pro internetovou komunikaci. Byl zvolen pro svoji rozšířenou podporu jak ze strany webových hostingů, tak i ze strany nástrojů pro vývoj klientské aplikace.

Vlastní komunikace probíhá pomocí dvou zpráv a to požadavkem od klienta a následnou odpovědí serveru. Klient zašle serveru zprávu obsahující dávku naměřených dat společně s informacemi o zařízení. Server tyto data zpracuje a informuje klienta o úspěšnosti operace.

Požadavek klienta

Data jsou zasílána klientem metodou HTTP POST ve formě souboru v jazyce XML[3]. Tento jazyk byl zvolen především pro jeho přehlednost a rozšířenou podporu. Nevýhodou může být větší objem zasílaných dat. Konkurenční JSON[4] se v tomto ohledu vyznačuje menší náročností, je však méně přehledný.

Kořenový element `device` obsahuje dva povinné atributy. Prvním je atribut `name` který obsahuje jedinečný identifikátor zařízení v celé síti. Následuje atribut `model` obsahující název zařízení. Uvnitř kořenového elementu se nachází údaje ze senzorů a informace o poloze daného zařízení. Tyto elementy se opakují pro každé měření.

Informaci o poloze představuje element `location`. Zde specifikujeme atributy `time` a provider, `time` představuje informaci o čase, ve kterém byla poloha určena. Provider určuje způsob určení polohy, zda se jedná o informaci ze satelitů GPS nebo odhad podle dostupných informací z okolí. Element `location` dále obsahuje dceřiné elementy `lon`, `lat`, `alt`, `spd` a `acc` které v sobě uchovávají informace o zeměpisné šířce a délce, nadmořské výšce, rychlosti pohybu a přesnosti měření.

Naměřené údaje ze senzorů představuje element `sensor`. Jeho atributy jsou `time` a `id`. Atribut `time` plní shodnou funkci jako u elementu `location`, `id` představuje číselnou identifikaci typu senzoru. Dceřiné elementy uchovávají naměřené údaje. Podporované senzory, jejich číselné

[2] Hypertext Transfer Protocol [8]
[3] Extensible Markup Language [9]
[4] JavaScript Object Notation

identifikátory a počet předávaných hodnot odpovídá specifikacím v Aplikačním rozhraní OS Android (kap. 5.2).

XML soubor však může obsahovat data z více různých zařízení, a to povolením elementu `device` v kořenovém elementu na úrovni `location` a `sensor`. Pro tento element platí shodná pravidla jako pro element kořenový. Takto lze vkládat kompletní informace libovolného zařízení do jednoho výstupu, nebo zasílat neúspěšně odeslaná předchozí data daného uzlu (kap. 6.4.3).

```
<?xml version='1.0' standalone='yes' ?>
<device name="bs" model="Samsung Nexus S">
  <location time="2013-04-27T18:21:14.500+0200" provider="network">
    <lon>15.8905098</lon>
    <lat>49.096723</lat>
    <alt>0.0</alt>
    <spd>0.0</spd>
    <acc>24.0</acc>
  </location>
  <sensor time="2013-04-27T18:21:17.489+0200" id="4">
    <val0>-0.6414085030555725</val0>
    <val1>-0.40194934606552124</val1>
    <val2>0.4740314185619354</val2>
  </sensor>
  <sensor time="2013-04-27T18:21:18.251+0200" id="4">
    <val0>-0.46670106053352356</val0>
    <val1>0.29077184200286865</val1>
    <val2>0.28588494658470154</val2>
  </sensor>
  <sensor time="2013-04-27T18:21:17.561+0200" id="5">
    <val0>13.12664794921875</val0>
  </sensor>
  <sensor time="2013-04-27T18:21:17.761+0200" id="5">
    <val0>11.20907974243164</val0>
  </sensor>
</device>
```

Záznam 6.1: Ukázka zprávy od klienta.

Odpověď serveru

Server zasílá nezformátovanou informační zprávu o úspěšném provedení operace, případně o nastalých chybách. Jedná se o čistý text, který je v nezměněné podobě zaznamenán v logu klienta.

Omezení protokolu

Klient po úspěšném odeslání zprávy předpokládá korektní zpracování serverem. Neanalyzuje příchozí zprávu, pouze ji zobrazuje. Chyby při zpracování na straně serveru proto mohou vést ke ztrátě

informací. Chybový stav však může nastat pouze při přijetí nepodporovaného souboru nebo při kritické chybě samotného serveru.

6.3.2 Zařízení – zařízení

Jednotlivé senzorové uzly si mohou předávat data přímo, bez nutnosti serveru. Tato komunikace je pouze jednosměrná. Uzel, který data odesílá jinému, informace nejprve uloží do souboru v jazyce XML, který odpovídá struktuře souboru specifikovaném v kapitole 6.3.1. Po úspěšném zaslání souboru uzel předpokládá následné úspěšné doručení na server.

6.4 Mobilní klient

Mobilní zařízení bude vykonávat funkci senzorového uzlu. Tuto činnost nám umožní klientská aplikace, která bude napsána v jazyce Java s využitím Android SDK. Aplikaci bude možno nainstalovat běžným způsobem formou APK balíčku na přístroje s operačním systémem Android ve verzi 4.0 a vyšší. Nižší verze systému nebudou podporovány. V Aplikačním rozhraní verze 14 (Android 4.0) došlo, mimo jiné, ke změnám při práci se senzory a se službami běžícími na pozadí. Společně s ubývajícím počtem zařízení s nižší verzí systému jsou změny v aplikačním rozhraní hlavním důvodem pro toto omezení.

6.4.1 Nastavení aplikace

Pro správnou činnost senzorového uzlu je zapotřebí nastavit několik základních údajů. Konfigurace senzorového uzlu bude probíhat pomocí grafického rozhraní aplikace. V senzorové síti je každé zařízení identifikováno podle unikátního jména. Bez vyplnění této informace senzorový uzel nemůže komunikovat s okolím. Následuje volba způsobu komunikace a případné vyplnění adresy HTTP serveru pro zasílání dat pomocí internetové sítě. Údaje budou uloženy v paměti zařízení, a proto je nebude nutné vyplňovat při každém zapnutí klientské aplikace. Adresa serveru je při prvním spuštění aplikace nastavena na adresu ukázkového serveru. Změny v nastavení se projeví při příštím spuštění služby.

6.4.2 Zaznamenávání senzorů a informací o poloze

Aplikace umožní zaznamenávat hodnoty veličin ze všech dostupných senzorů na mobilním zařízení. Společně s těmito hodnotami bude registrovat informace o poloze z GPS satelitů a pomocí informací z okolí zařízení (Network location provider). Pro každé měření je zaznamenán čas události.

Senzory

Naměřené hodnoty jsou senzorem zasílány formou událostí při změně. Každý senzor takto může zaznamenat desítky změn každou vteřinu. Uchování takovéhoto množství údajů, jejich zpracovávání a zasílání je náročné na systémové prostředky a mohlo by vést až k neschopnosti uzlu zpracovat informace v požadovaném čase.

Pro naši aplikaci není zapotřebí evidovat každou změnu. Postačí uložení několika vzorků pro každou vteřinu. Tohoto dosáhneme následujícím mechanizmem. Každá změna oznámená senzorem bude dočasně zaznamenána a jednou za daný časový interval bude z dostupných hodnot vypočtena jedna, která bude uložena trvale. Výpočet výsledné hodnoty může být realizován libovolnou funkcí, pro naše řešení si vystačíme s jednoduchým výběrem nejextrémnější (maximum) hodnoty ze vzorku. Tímto zajistíme relativní nenáročnost výpočtu a také, že budou zaznamenány krátce trvající výkyvy v měření, které by při použití například prostého průměru byly potlačeny. Vypočtené hodnoty budou uloženy v paměti až do jejich odeslání.

V aplikaci budou dostupné informace o dostupných senzorech jako např. výrobce, model, rozlišení, rozsah měřitelných hodnot nebo energetickou náročnost. Tyto údaje však nebudou zasílány v rámci senzorové sítě.

Poloha

Informace o poloze lze získat pomocí satelitů GPS, či s využitím služby Network location provider a každá metoda se vyznačuje svoji přesností a aktuálností. Výpočet jedné výsledné polohy proto není triviální a přesahuje zaměření této práce. Naše aplikace bude zaznamenávat údaje z obou metod odděleně. Frekvence zasílání změn polohy není zdaleka tak vysoká jako v případě informací ze senzorů. Proto budou údaje o poloze ukládány přímo a bez vzorkování. Zaznamenány budou informace o zeměpisné šířce a délce, nadmořské výšce, rychlosti a přesnosti měření.

Jelikož prvotní zjištění polohy nebývá okamžité, aplikace si při zapnutí zažádá o informaci o poslední známé poloze a uloží ji. Takto máme okamžitě k dispozici přibližný údaj.

6.4.3 Komunikace se serverem

Zařízení v pravidelných časových intervalech posílá zaznamenané údaje na vzdálený server, kde jsou poté uloženy. Před samotným zasláním informací je zapotřebí sestavit z dostupných údajů zprávu ve formátu XML, která odpovídá používanému komunikačnímu protokolu. V případě že se zprávu nepodaří zaslat, je uložena v jejím textovém formátu a odeslána při dalším pokusu o spojení. Není opětovně vytvářena, tím šetříme procesorový čas potřebný k jejímu vytvoření. Časový interval zasílání dat je zapotřebí zvolit s ohledem na požadovanou aktuálnost údajů na serveru. Také zohledníme čas potřebný k vytvoření zprávy a její zaslání. Příliš krátký interval může způsobit, že zařízení nebude schopné dostatečně rychle data zpracovat a odeslat. Aplikace vyčká na odpověď ze strany serveru. Tuto informaci však nezpracovává. Pouze ji vypisuje uživateli v logu.

Při komunikaci se používá dostupné mobilní datové připojení. Zda se jedná o Wi-Fi, či data mobilního operátora necháváme v režii operačního systému.

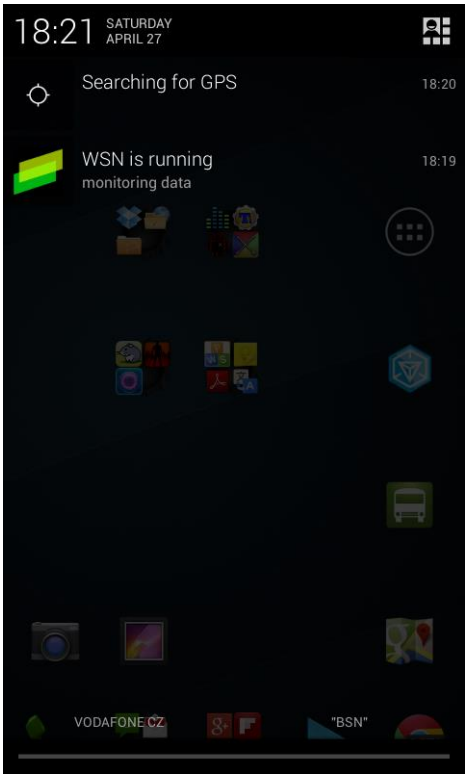
6.4.4 Komunikace mezi zařízeními

Komunikace s okolními uzly probíhá pomocí technologie Bluetooth formou schématu klient-server. Každý uzel s aktivním Bluetooth modulem zastává roli serveru a může přijímat data od okolních zařízení. Po navázání komunikace na daném kanálu je spojení s klientem udržováno do té doby, než se klient sám odpojí. Tato možnost je zvolena pro omezení režie na navazování a ukončování spojení. Jelikož může být jeden Bluetooth kanál obsluhován pouze jedním zařízením současně, server udržuje více otevřených kanálů pro komunikaci s více zařízeními. V případě přijetí dat, je server obratem zasílá dále dle vlastního nastavení komunikace.

Uzel dále může zasílat data, ať už svá, nebo uzlu v okolí, přímo na server, nebo bude komunikovat se zařízením v okolí. Tuto volbu určí uživatel v rozhraní aplikace. Pro zasílání dat okolnímu uzlu, musí být zařízení vzájemně spárovaná. Poté zastává uzel roli klienta a vyhledává na vzdáleném zařízení volný kanál, na kterém může zahájit komunikaci. Po zahájení komunikace jsou data zasílána na vzdálené zařízení v pravidelných intervalech a ve formátu jako v případě komunikace se vzdáleným serverem na síti (kap. 6.3.2.). Komunikace přes Bluetooth však v naší aplikaci probíhá pouze jednosměrně. V případě nežádoucího ukončení spojení se klient pokouší spojení opětovně navázat. Jestliže data nemohou být přenesena, jsou uchována, jako v případě komunikace přes síť, v textové podobě do příštího pokusu o odeslání.

6.4.5 Služba běžící na pozadí

Běžné vizuální aplikace běžící na operačním systému Android bývají pozastaveny při přepnutí do jiné aplikace, či při vypnutém displeji. Pro naši aplikaci klienta však vyžadujeme neustálou činnost bez závislosti na těchto vlivech. Služba běžící na pozadí jim však nepodléhá a proto jádro naší aplikace vytvoříme jako službu. Služba bude provádět veškerou činnost senzorového uzlu. Operační systém obvykle ukončuje déle nepoužívané běžící aplikace i služby z důvodu uvolňování operační paměti. Toto chování je v našem případě nežádoucí jev. OS Android však nabízí prostředky pro označení služby proti ukončování. V implementaci jádra naší aplikace proto těchto prostředků využijeme. Informace o běžící službě bude zobrazena v notifikační oblasti operačního systému, odkud bude možné dotykem přejít do grafického rozhraní aplikace.



Obr. 6.2: Informace o běžící službě.

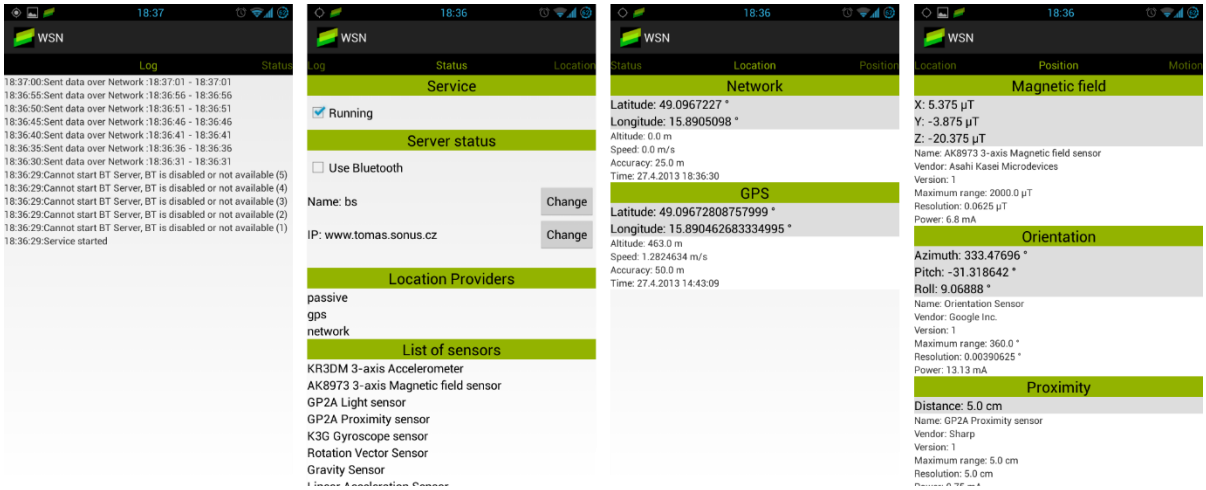
6.4.6 Uživatelské rozhraní

Naše aplikace bude disponovat i vizuální částí. Toto rozhraní poslouží pro konfiguraci služby, její spouštění a ukončování. Dále bude součástí prostředí také jednoduchý log, který umožní zobrazovat aktuální informace o stavu služby a přehled aktuálních údajů ze všech senzorů a metod určování polohy. Grafické rozhraní je rozděleno do šesti částí, které jsou vzájemně odděleny jak logicky, tak vizuálně. Jednotlivé části budou zobrazeny jako oddělené obrazovky formou karet, mezi kterými bude možné plynule přecházet dotykovými gesty (swipe).

Obr. 6.3:Ukázka dotykového gesta swipe [3].

Log

Tato karta slouží pro zobrazení informací o stavu a činnostech senzorového uzlu. Údaje jsou řazeny chronologicky od nejnovějšího po nejstarší. Součástí záznamu je informace o čase, kdy daná událost nastala. Celý log je načítán v pravidelných intervalech, čímž zajistíme aktuálnost zobrazovaných informací.



Obr. 6.4: Ukázka karet vizuální aplikace.

Status

Karta slouží pro přehled o nastavení uzlu, dostupných senzorech a podporovaných metodách zjišťování polohy. Je zde možnost zvolení identifikátoru zařízení, způsobu komunikace a adresy vzdáleného serveru. Dále karta nabízí spuštění a ukončení samotné služby.

Location, Position, Motion, Environment

Zde se nacházejí informace o aktuálně určené poloze a údaje naměřené jednotlivými senzory. Společně s naměřenými hodnotami zde jsou k dispozici další dostupné informace o čidlech. Zobrazované údaje jsou v pravidelných časových intervalech obnovovány.

6.5 Server

Data ze senzorové sítě je zapotřebí ukládat a přehledně zobrazovat. Tyto činnosti budou zajištěné serverovou částí naší aplikace. Server bude přijímat data od uzlů, zpracovávat je a uchovávat v databázi. Dále bude server umožňovat zobrazení webového rozhraní, ve kterém bude mít uživatel po přihlášení přístup k informacím z celé senzorové sítě. Informace budou zobrazeny jak v textové podobě, tak ve formě grafů a poloh vyznačených na mapě.

Serverová část bude implementována ve skriptovacím jazyce PHP[5]. Tento jazyk disponuje potřebnými nástroji pro práci s protokolem HTTP a databázemi. Dále nabízí prostředky pro analýzu XML, které je použito pro komunikační protokol naší aplikace. Data budou uložena v relační databázi MySQL.

Jazyk PHP a databáze typu MySQL byly zvoleny pro jejich rozšířenost a podporu u poskytovatelů webových hostingů.

6.5.1 Aplikační rozhraní pro komunikaci s klientem

Klient zašle data serveru na URL aplikačního rozhraní. Server nejprve celou zprávu uloží v jejím textovém formátu pro účely archivace a ladění. Poté zpracuje samotnou zprávu a uloží jednotlivé údaje do databáze k danému zařízení. Server také zaznamená čas, ve který byl požadavek obsloužen. Po dokončení operace je klientu předána informace o délce trvání požadavku. V případě nepodporovaného formátu vstupních dat je vráceno chybové hlášení.

Aplikační rozhraní bude dostupné na URL: `<domenove jmeno>/api.php`.

6.5.2 Uchování dat

Server bude ukládat data v relační databázi. Budou zde uloženy dostupné údaje o zařízeních, údaje z jednotlivých uzlů a základní informace o uživatelských účtech webového rozhraní. Databáze naší aplikace je složena z osmi tabulek.

Uživatelská jména a hesla jsou uložena v tabulce `users`. Hesla jsou pro zjednodušení ukládána v otevřené podobě. Společně se jménem a heslem je evidována informace, zda je účet aktivní. Tímto lze libovolný účet deaktivovat bez nutnosti jeho smazání.

Informace o uzlu jsou uložena v tabulkách `devices` a `devices_info`. První zmíněná eviduje jméno uzlu (identifikátor v rámci sítě) a název zařízení. V rámci databáze je však uzel reprezentován automaticky generovaným číselným identifikátorem. Tabulka `devices_info` uchovává podrobné informace o zařízení, jako například stav baterie, způsob komunikace a další. V současném návrhu je v této tabulce evidován pouze čas kdy došlo ke komunikaci s klientem. Návrh tabulky však počítá s dalším rozšířením aplikace.

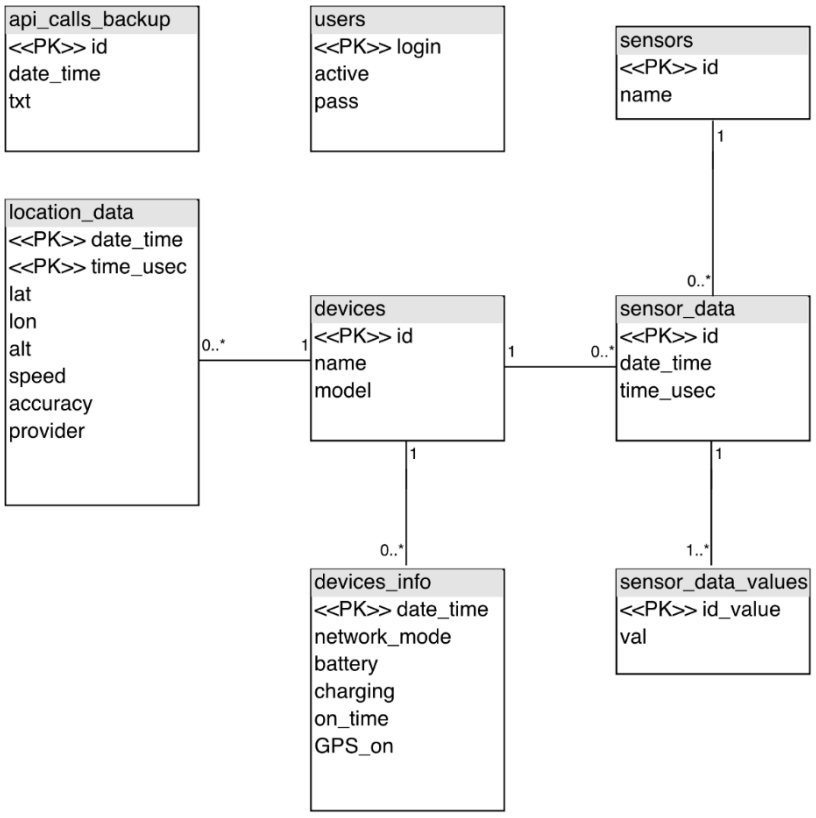
Tabulka `location_data` slouží k uchování informací o poloze jednotlivých uzlů.

Pro ukládání údajů ze senzorů slouží tabulky `sensor_data` a `sensor_data_values`. V první z nich jsou uloženy souhrnné informace o konkrétním měření a druhá již obsahuje samotné naměřené hodnoty. Dvou tabulek je využito ve snaze navrhnout databázi univerzálně pro všechny druhy čidel. Některé předávají hodnotu jednu, některé více.

[5] Hypertext Preprocessor [10]

Dále návrh obsahuje tabulku `sensors`, která nám poskytuje názvy jednotlivých senzorů v závislosti na jejich identifikátoru.

Poslední tabulka `api_calls_backup` zaznamenává všechny přijaté zprávy od uzlů v původní textové podobě.



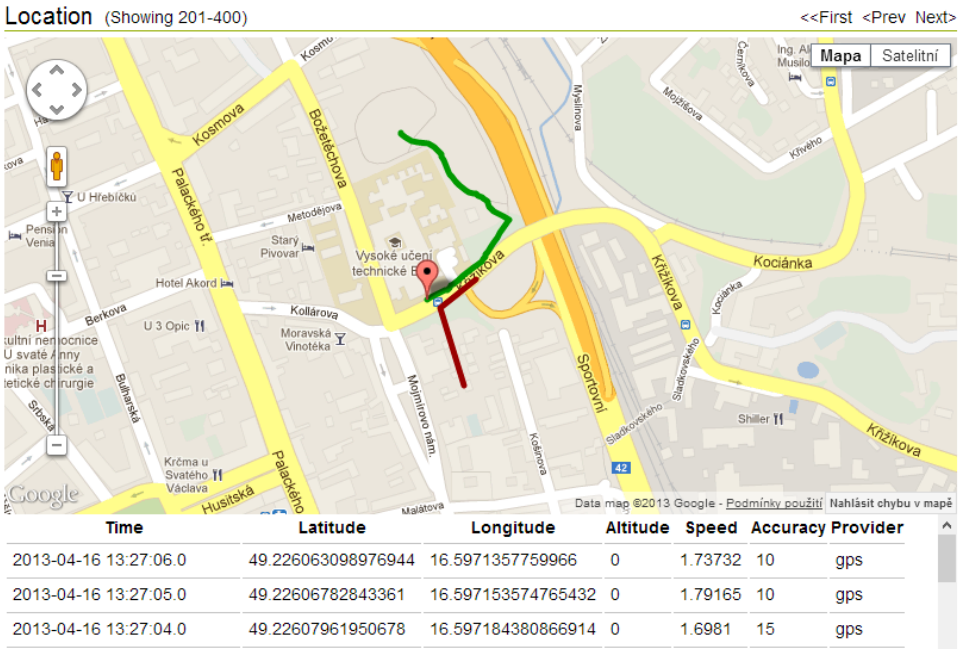
Obr. 6.5: ER diagram databáze.

6.5.3 Webové prostředí

Součástí serverové části je webové rozhraní z internetové adresy serveru. Pro přístup k informacím ze senzorové sítě je od uživatele vyžadováno přihlášení. Aktuální informace o přihlášení uživatele jsou ukládána pomocí PHP sessions.

Webové rozhraní se po přihlášení skládá ze tří hlavních částí. Levý postranní panel zobrazuje hlavní nabídku. V sekci Devices nalezneme seznam všech uzlů, o kterých jsou na serveru uložené informace. V seznamu jsou vypsány identifikátory uzlů, pomocí kterých jsou zařízení identifikována v rámci sítě. Po kliknutí na název zařízení se zobrazí jeho informace. V levém panelu následuje část Service menu která v současném návrhu obsahuje pouze položku Api calls backup, která slouží pro zobrazení archivovaných zpráv od uzlů.

Napravo od postranního panelu se nachází oblast pro zobrazování požadovaného obsahu a v hlavičce stránky je zobrazeno jméno aktuálně přihlášeného uživatele s možností odhlášení.

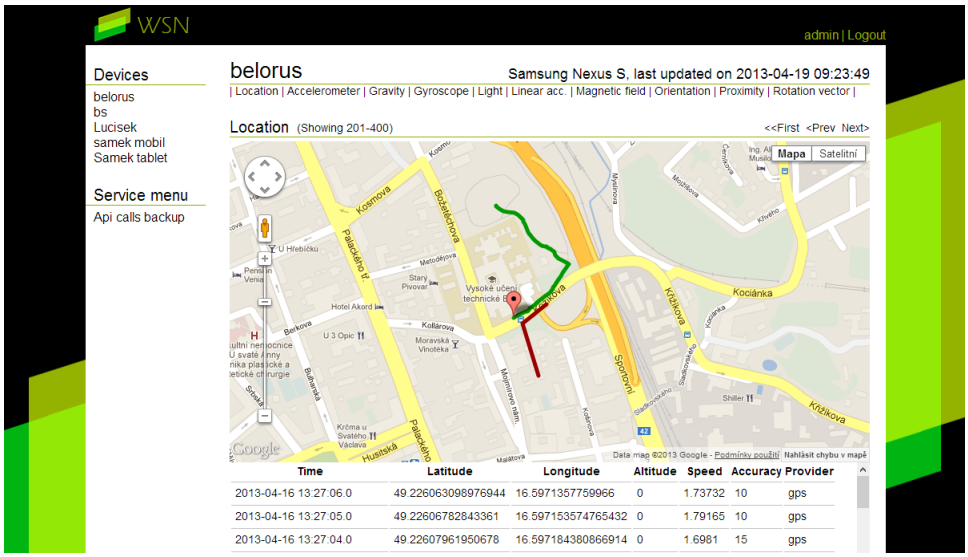


Obr. 6.6: Ukázka webového rozhraní.

Karta senzorového uzlu

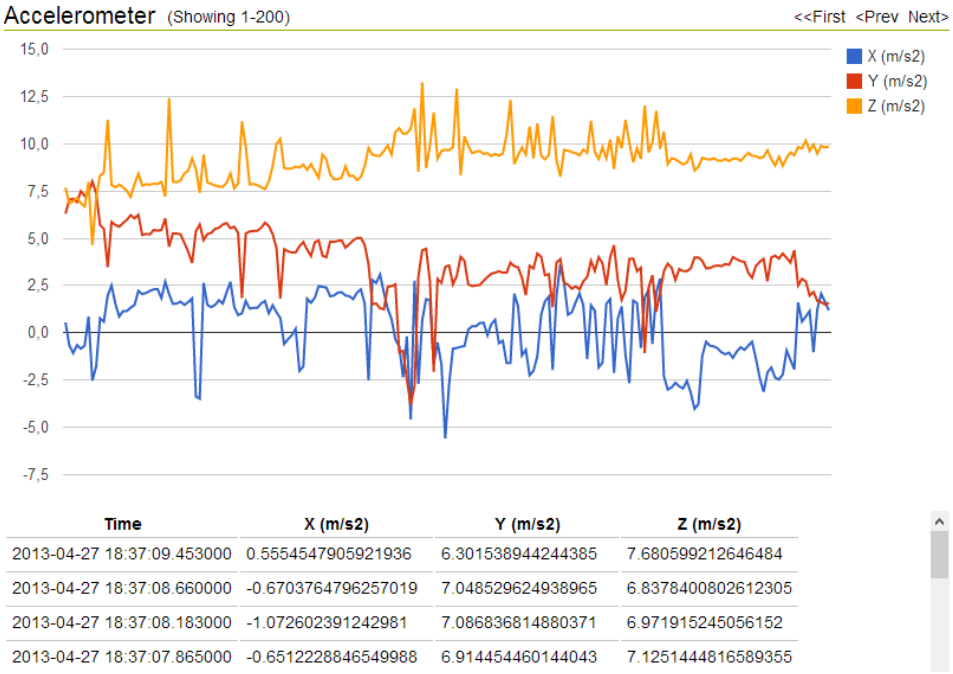
Karta každého zařízení je rozdělena do dvou oblastí. V horní části je zobrazeno jméno uzlu v rámci sítě, skutečný název a informace o času poslední aktualizace. Pod těmito informacemi se nachází výčet typů údajů, které uzel zaznamenává. Po kliknutí na jednu z položek jsou v těle karty zobrazeny příslušné naměřené hodnoty. Při vybrání zařízení v postranním panelu jsou jako první vykresleny údaje o poloze uzlu.

Naměřené údaje jsou stránkovány, přičemž současně je zobrazeno pouze určité množství naměřených hodnot. V rozhraní jsou umístěny navigační prvky pro posun na následující, předchozí, nebo první stránku.



Obr. 6.7: Zobrazení informací o poloze.

Zobrazovaná data jsou opětovně načítána v pravidelných časových intervalech. Načítána je pouze oblast stránky s daty, nikoli celé webové rozhraní. Tímto je omezena zátěž serveru a sníženy nároky na rychlost internetového připojení.



Time	X (m/s2)	Y (m/s2)	Z (m/s2)
2013-04-27 18:37:09.453000	0.5554547905921936	6.301538944244385	7.680599212646484
2013-04-27 18:37:08.660000	-0.6703764796257019	7.048529624938965	6.8378400802612305
2013-04-27 18:37:08.183000	-1.072602391242981	7.086836814880371	6.971915245056152
2013-04-27 18:37:07.865000	-0.6512228846549988	6.914454460144043	7.1251444816589355

Obr. 6.8: Zobrazení údajů ze senzoru.

Informace ze senzorů jsou zobrazeny v grafu nebo vykresleny na mapě. Pod tímto zobrazením jsou tytéž hodnoty vypsány v tabulce. Pro zvýšení přehlednosti tohoto řešení je tabulka umístěna jako rolovací objekt. Při zobrazování informací o poloze jsou barevně odlišeny údaje z GPS (zelená) a hodnoty ze sítě (červená). Při kliknutí na řádek v tabulce daný bod zvýrazněn i na mapě. V případě informací vykreslených v grafech toto neplatí a konkrétní hodnoty jsou zobrazeny přímo při najetí myší na průběh grafu.

Pro vykreslení polohy na mapě je využito služeb Google Maps API [11] a pro zobrazení grafů Google Chart Tools [12].

7 Implementace sítě

Tato kapitola popisuje strukturu mobilního klienta a serverové aplikace. Dále se zaměřuje se na způsoby implementace dílčích částí senzorové sítě.

7.1 Mobilní klient

Aplikace mobilního klienta je vyvíjena v jazyce Java ve vývojovém prostředí Eclipse s nainstalovaným rozšířením ADT. Při implementaci aplikace je využito standartních knihoven dostupných v SDK.

7.1.1 Logická struktura aplikace

Programovací jazyk Java je objektově orientovaný. Z tohoto důvodu je naše aplikace vytvořena jako množství komunikujících objektů. Na tomto místě se seznámíme s jednotlivými třídami a jejich rolemi. Diagram tříd aplikace se nachází v Příloze 2.

- **BackgroundService -** Rozšiřuje třídu `Service`. Umožňuje běh služby na pozadí.
- **BluetoothClient -** Implementuje komunikaci pomocí Bluetooth v roli klienta.
- **BluetoothServer** – Implementuje roli Bluetooth serveru.
- **DataStorage** – Umožňuje uložení naměřených informací o poloze a údajů ze senzorů.
- **EncodingData** – Tato třída nabízí prostředky pro vytvoření XML zprávy odesílané na server. Využívá přitom dat uchovaných v instanci třídy `DataStorage`.
- **EventsProcessor** – Přijímá informace o senzorech a poloze od `SensorEventsHandler` a `LocationHandler`. Ukládá je do instance třídy `DataStorage` a v pravidelných intervalech odesílá.
- **FragmentEnvironment, FragmentLocation, FragmentLog, FragmentMotion, FragmentPosition, FragmentStatus** – Jedná se o třídy, které umožňují zobrazování grafických prvků jednotlivých obrazovek aplikace.
- **LocationHandler** – Komunikuje s aplikačním rozhraním pro získávání polohy zařízení. Získané údaje předává instanci třídy `EventsProcessor`.
- **LocationRecord** – Třída sloužící pro uložení údajů z jednoho určení polohy. Je využita v třídách `EventsProcessor` a `EncodingData`.
- **MainActivity** – Hlavní třída vizuální části aplikace, umožňuje zobrazení grafického rozhraní a využívá tříd fragmentů.
- **MyLog** – Třída zastřešuje práci s logem. Umožňuje vytvářet nový záznam, záznamy uchovávat či zprostředkovávat jiným třídám.
- **NetworkHandler** – Komunikuje se serverem prostřednictvím sítě internet

- **SensorRecord** - Třída sloužící pro uložení údajů z jednoho měření senzoru. Je využita v třídách `EventsProcessor` a `EncodingData`
- **SensorEventsHandler** - Komunikuje s aplikačním rozhraním pro získávání dat z jednotlivých senzorů. Získané údaje uchovává v instanci třídy `SensorTempValues`, odkud je v časových intervalech zasílá instanci třídy `EventsProcessor`.
- **SensorTempValues** – Slouží k dočasnému ukládání množství naměřených údajů ze senzorů. V případě potřeby z těchto hodnot vypočte jednu výslednou a tu předá dále.

7.1.2 Použité nástroje a prostředky

Při implementaci je využito aplikačních rozhraní pro bezdrátovou komunikaci prostřednictvím Bluetooth, pro práci se senzory a pro komunikaci prostřednictvím sítě internet. Práce s těmito rozhraními je popsána v kapitolách 4 a 5. Mimo těchto prostředků je využito několik dalších, jejichž funkce bude v následující části práce přiblížena. Tyto balíčky a třídy jsou součástí Android SDK.

Procházení kartami grafického rozhraní

Pro tuto činnost je využito instance třídy `ViewPager`, která nám umožní přistupovat k jednotlivým obrazovkám aplikace prostým gestem prstu (swipe). Obrazovky jsou reprezentovány pomocí tzv. fragmentů. Těchto možností je v implementaci využito v třídě `MainActivity`.

Komunikace pomocí protokolu HTTP

Součástí standardních knihoven je balíček `org.apache.http`. Tento balíček obsahuje rozhraní a třídy pro komunikaci pomocí protokolu HTTP. V implementaci je využito tříd `HttpClient` a `HttpPost`, které poskytují potřebné rozhraní klienta a metody pro zaslání dat serveru a přijetí odpovědi. Balíček `org.apache.http` je využit v třídě `NetworkHandler`.

Vytváření zprávy ve formátu XML

Pro práci s XML soubory máme k dispozici třídu `XmlSerializer`, jejíž metody jsou využity v implementaci pro vytvoření zprávy odesílané na server. Hodnoty elementů a atributů vkládáme jednoduchým převedením na textový tvar. Při vkládání informací o čase je tento údaj nejprve zapotřebí převést do textového formátu, který odpovídá specifikaci XML. Převod do požadovaného formátu provedeme použitím třídy `SimpleDateFormat`.

Časovače událostí

Pro plánování jednorázových, či opakovaných událostí je v jazyce Java dostupná třída `Timer`. Vykonáním metody `shelude()`, kde specifikujeme požadovaný čas a úlohu, která se bude vykonávat v časových intervalech. Pro tyto úlohy je vyčleněno samostatné vlákno. Naše aplikace však

má příslušný kód vykonávat v hlavním vlákně, proto je využito tříd `Handler` a `Runnable`, které nám umožní poslat zprávu hlavnímu vláknu a vykonat příslušnou metodu zde.

7.2 Serverová část

Serverová aplikace je vytvořena ve skriptovacím jazyce PHP. Pro správnou činnost je na hostitelském vyžadována podpora PHP ve verzi 5. Webové rozhraní je klientu předkládáno jako HTML stránka, využívající CSS a JavaScriptových prvků.

7.2.1 Použité nástroje a frameworky

Při implementaci serveru bylo využito mnohých knihoven. Některé jsou nabízené samotným PHP a jsou pro funkci serveru klíčové, jiné jsou JavaScriptové a slouží pro vytvoření požadovaných vlastností webového rozhraní.

Analýza souborů XML

Prostředí PHP disponuje rozšířením SimpleXML, které nabízí jednoduchou sadu nástrojů pro převod dokumentu ve formátu XML na objekt, který je dále zpracován běžným způsobem.

Práce s databází

V jazyce PHP máme k dispozici funkce pro práci s různými typy databází. Naše aplikace využívá ty z nich, které umožňují komunikaci s databází MySQL. Použité jsou funkce `mysqli_connect()` a `mysqli_select_db()` pro navázání spojení s databázovým serverem a připojení k databázi. Pomocí funkce `mysqli_query()` specifikujeme SQL příkazy. Pro případ procházení dat v databázovém pohledu slouží funkce `mysqli_fetch_array()`, která převede informace na aktuálním řádku v pohledu do jednoduchého pole [10].

Periodické načítání části stránky

Pro průběžnou aktualizaci části stránky je využito možností JavaScriptové knihovny jQuery, která nám umožní opětovně načíst obsah oblasti stránky pomocí funkce `load()`[13]. Pro opětovné načítání stránky slouží funkce `setInterval()`, která vykoná určitý kód v pravidelných časových intervalech.

Zobrazení informací na mapě

Informace o poloze jsou ve webovém prostředí vykresleny do mapy pomocí Google Maps JavaScript Api v3. Toto aplikační rozhraní nám umožní vložit do stránky Google mapu, kde mimo jiné můžeme označovat místa a vykreslovat trasu. Pro jeho využití je zapotřebí vlastnictví unikátního klíče.

Bezplatné použití tohoto rozhraní je omezeno na 25 000 načtení mapy za den v rámci jednoho klíče [11].

Vynesení údajů do grafu

Ve webovém rozhraní jsou grafy vykresleny pomocí JavaScriptových nástrojů Google Chart Tools. Tyto nástroje umožňují vizualizaci dat prostřednictvím mnoha typů grafů. Pro naše účely je využito jednoduchého spojnicového grafu. Nástroje umožňují volbu rozsahu jednotlivých os, zadání popisků, či vynesení více grafů do jednoho. Samotný graf je poté vykreslen pomocí technologie HTML5/SVG [12].

7.3 Zhodnocení výsledků

Výsledná síť snímá údaje ze všech čidel senzorových uzlů s frekvencí dvou měření za vteřinu. Každých 5 vteřin uzel odesílá tato data společně s informacemi o poloze na server nebo jinému uzlu v okolí pomocí technologie Bluetooth, který data okamžitě předá dále. Výslednému stavu sítě však předcházelo množství výkonových optimalizací a překonávání omezení testovacích zařízení.

Při implementaci sítě bylo zapotřebí dodržet dva protichůdné požadavky. Aktuálnost a dostatečné množství naměřených informací a přijatelnou náročnost na hardware zařízení. Výsledná implementace se snaží dodržet obojího.

7.3.1 Problémy při implementaci

Následující podkapitola shrnuje poznatky z vytváření sítě a přibližuje čtenáři problémy, které bylo zapotřebí vyřešit při cestě k výsledné implementaci.

Výkon mobilních zařízení, přesnost měření a aktuálnost dat

Při implementaci zaznamenávání údajů ze senzorů jsem narazil na překážku, kdy senzor zasílá údaje příliš často (desítky údajů za vteřinu). Zpracování a ukládání takovéhoto množství údajů vedlo ke zvýšení náročnosti na paměť a příliš vysokého zatížení procesoru testovacího zařízení. Programově nebylo možné frekvenci zasílání omezit, a proto jsem navrhl mechanizmus, který hodnoty dočasně zaznamenává a ze vzorku dat za určitý čas vypočítá jednu hodnotu, která je předávána dále. Tento mechanizmus nám umožnil kontrolovat množství ukládaných informací.

Další problémy s výkonem mobilního zařízení nastaly při vytváření XML zprávy zasílané na server. Původní návrh počítal se snímáním deseti naměřených hodnot z každého senzoru za vteřinu a jejich odesílání každé dvě vteřiny. Pro testování jsem měl k dispozici telefony Goodle Nexus S, Sony Xperia U a tablet Asus Transformer. Ukázalo se však, že ani jedno z testovacích zařízení nebylo schopné data zpracovávat a odesílat v požadovaném čase. Proto bylo sníženo množství zaznamenávaných údajů z deseti na dvě měření za vteřinu. Také frekvence zasílání dat musela být

s nížena na konečnou hodnotu 5 vteřin. Toto opatření vedlo ke snížení přesnosti měření a zvýšení prodlevy mezi naměřením informace a jejím odesláním na server. Byl však ušetřen procesorový čas, paměť zařízení a samotné množství zasílaných dat.

Chování senzorů

Dalším závažnějším problémem bylo nestandartní chování některých čidel (proximity senzor na testovacím Asusu Transformeru). Čidlo může zasílat naměřené hodnoty, avšak nepřiřadí k nim časovou značku. Toto nestandartní chování je proto kontrolováno a v případě chybějící časové značky se při obsluze události vloží čas systémový.

Komunikace prostřednictním Bluetooth

Nejvíce času při vývoji trvala implementace spolehlivé komunikace prostřednictvím Bluetooth. Prvotní návrh počítal s jedním otevřeným kanálem, pomocí kterého budou zařízení komunikovat. Zařízení se připojí k druhému, zašle data a poté se odpojí. Z testování na všech testovacích zařízení však vyplynulo, že čas potřebný pro připojení a především odpojení zařízení dosahuje hodnot jednotek vteřin. Dále z měření vyplynulo, že každý kanál je schopen obsluhovat pouze jedno aktivní připojení. I při okamžitém zaslání informací je běžné, že je kanál blokován více než 5 vteřin, což vedlo ke stavu kdy i jediné zařízení nebylo schopné se dostatečně rychle odpojovat a připojovat. Ve výsledné implementaci se klient připojí a udržuje své spojení po celou dobu provozu. Jestliže je spojení z jakéhokoli důvodu ztraceno, klient se pokouší spojení znovu navázat. Dále byl navýšen počet otevřených kanálů na Bluetooth serveru na 5. Tím dosáhneme možnosti komunikovat s více zařízeními současně. Při vyšším počtu otevřených kanálů a komunikujících zařízení docházelo k výpadkům spojení (Google Nexus S) a lze tedy usuzovat, že Bluetooth modul obsažený v mobilních zařízení nemusí být vždy optimalizovaný pro práci s takovým množstvím připojení.

Aktuálnost dat při komunikaci pomocí Bluetooth

V případě že uzel přijme data od okolního uzlu, ihned je zasílá dále, aby nenarůstalo zpoždění při předávání dat. V prvotním návrhu byla tato data předána dále až při dalším zasílání vlastních dat. V případě zřetězení většího množství však docházelo k výraznému zpoždění (5 vteřin na každém uzlu) a proto byl tento mechanizmus přepracován. Při okamžitém přeposílání dat je zpoždění dané pouze množstvím času, které je potřebné pro přijetí zprávy a opětovné odeslání.

Výkon serveru

Během testování sítě byla serverová část spuštěna na běžném počítači s veřejnou IP adresou. Při testování sítě s více senzorovými uzly jsem však zaznamenal výkonnostní problémy na dostupném stroji. Při zpracování zprávy od uzlu dochází k ukládání stovek až tisíců záznamů do databáze. I přes optimalizaci práce s databází však nebyla rychlost serveru dostačující a proto byla celá serverová část přenesena na placený webový hosting u Active24 kde již k výkonnostním problémům nedocházelo.

7.3.2 Možné rozšíření aplikace

Jako budoucí rozšíření sítě by bylo vhodné zasílat podrobnější informace o samotném uzlu, jako například stav baterie či aktuální konfiguraci systému. Dále implementovat automatické vyhledávání okolních uzlů či zasílání serveru informace o topologii sítě. Bylo by vhodné pozměnit formát komunikačního protokolu z XML na jiný, méně datově náročný.

8 Závěr

Úlohou této práce bylo prostudovat a popsat možnosti současných mobilních zařízení s operačním systémem Android v oblastech bezdrátové komunikace a práce se senzory, a těchto poznatků poté využít pro návrh a implementaci vlastní senzorové sítě.

Prvním krokem pro vytvoření senzorové sítě bylo nastudování problematiky senzorových sítí a postupů pro vytváření mobilní aplikace pro daný operační systém. Poté jsem navrhl základní vlastnosti prvků sítě tak, aby splňovala zadání práce. Do návrhu byla zahrnuta možnost komunikace senzorových uzlů mezi sebou pomocí technologie Bluetooth a síť samotná byla navrhnuta tak, aby nebyla omezena počtem senzorových uzlů a prvky sítě mohly být zapojeny různými způsoby.

Výsledná senzorová síť musela být přizpůsobena možnostem mobilních zařízení. Při cestě k cíli jsem narazil na několik překážek. Při implementaci bylo nutné z důvodu nedostatečného výkonu testovacích zařízení přehodnotit požadavky na aktuálnost dat a přesnost měření a ve výsledné implementaci byla snížena jak frekvence zaznamenávání údajů, tak i frekvence jejich zasílání na server.

Testování senzorové sítě přineslo mnoho poznatků. I když jsou současná mobilní zařízení dostatečně výkonná pro mnoho oblastí, s vývojem aplikace jde ruku v ruce nutnost optimalizace na každém kroku. Vývojář mobilních zařízení musí počítat s výkonovými omezeními, která jsou o mnoho výraznější nežli v případě běžných osobních počítačů. Operační systém Android disponuje mnoha automatickými nástroji pro šetření energie a vývojář s nimi musí být obeznámen a vyvíjet aplikaci v souladu s nimi. Komplikace při vývoji jsou však vyvažovány mnohými klady těchto zařízení, jakými jsou například mobilita, množství specializovaných hardwarových modulů a dostatečný výkon.

Literatura

[1] LI, Yingshu, Weili WU a My T. THAI. *Wireless sensor networks and applications*. New York: Springer, 2007, 441 s. ISBN 978-038-7495-927

[2] MEIER, Reto. *Professional Android application development*. Indianapolis, IN: Wiley, 2009, 409 s. ISBN 978-047-0344-712

[3] GOOGLE. *Android Developers* [online]. [2013] [cit. 2013-05-06]. Dostupné z: http://developer.android.com/

[4] BLUETOOTH SIG. *Bluetooth: Developer Portal* [online]. © 2013 [cit. 2013-05-02]. Dostupné z: http://developer.bluetooth.org

[5] NFC FORUM. *NFC Forum* [online]. © 2013 [cit. 2013-05-02]. Dostupné z: http://www.nfc-forum.org/

[6] IEEE 802.11 standards tutorial. ADRIO COMMUNICATIONS LTD. *Radio-Electronics.com* [online]. [2013] [cit. 2013-05-06]. Dostupné z: http://www.radio-electronics.com/info/wireless/wi-fi/ieee-802-11-standards-tutorial.php

[7] ETSI. *ETSI: Mobile Communications* [online]. © 2012 [cit. 2013-05-02]. Dostupné z: http://www.etsi.org/technologies-clusters/technologies/mobile

[8] Hypertext Transfer Protocol -- HTTP/1.1. FIELDING, R., J. GETTYS, J. MOGUL, H. FRYSTYK, L. MASINTER, P. LEACH a T. BERNERS-LEE. *World Wide Web Consortium* [online]. World Wide Web Consortium, 1999 [cit. 2013-05-06]. Dostupné z: http://www.w3.org/Protocols/rfc2616/rfc2616.html

[9] Extensible Markup Language (XML) 1.0 (Fifth Edition). BRAY, Tim, Jean PAOLI, C. M. SPERBERG-MCQUEEN, Eve MALER a François YERGEAU. *World Wide Web Consortium* [online]. 5. vyd. World Wide Web Consortium, 2008, 7.2.2013 [cit. 2013-05-06]. Dostupné z: http://www.w3.org/TR/xml/

[10] *PHP: Hypertext Preprocessor* [online]. The PHP Group, © 2001-2013 [cit. 2013-05-06]. Dostupné z: http://php.net/

[11] *Google Maps API: Google Developers* [online]. Google, 2013, 9.4.2013 [cit. 2013-05-06]. Dostupné z: https://developers.google.com/maps/

[12] *Google Chart Tools: Google Developers* [online]. Google, 2013, 3.4.2013 [cit. 2013-05-06]. Dostupné z: https://developers.google.com/chart/

[13] .load(). *JQuery API Documentation* [online]. The jQuery Foundation, 2013 [cit. 2013-05-06]. Dostupné z: http://api.jquery.com/load/

Seznam příloh

Příloha 1. Struktura a soubory serverové aplikace
Příloha 2. Diagram tříd mobilní aplikace senzorového uzlu
Příloha 3. DVD

Příloha 1

Struktura a soubory serverové aplikace

Implementace serveru je pro zvýšení přehlednosti rozdělena do více souborů. Adresářová struktura serverové aplikace je následující:

- **index.php** – hlavní soubor webového rozhraní. Implementuje práci se sessions.
- **api.php** – soubor zastřešující obsluhu aplikačního rozhraní a zpracování požadavků senzorových uzlů.
- **data/api_calls_backup.php** – zobrazuje ovládací prvky stránky se zálohovanými požadavky klientů a pravidelně načítá oblast s daty.
- **data/api_calls_backup_data.php** – implementuje zobrazování zálohovaných požadavků.
- **data/connection.php** – Provádí připojení k databázovému serveru a výběr požadované databáze.
- **data/device.php** – Zastřešuje práci se zařízeními. Zobrazuje informace o uzlu a dále vkládá kartu s požadovanými naměřenými údaji.
- **data/device_location.php** - zobrazuje ovládací prvky stránky s informacemi o poloze uzlu, vykresluje mapu a pravidelně načítá oblast s daty.
- **data/device_location_data.php** - implementuje oblast se zobrazováním informací o poloze. Vykresluje jednotlivé body do mapy.
- **data/device_sensor.php** – zobrazuje ovládací prvky stránky s informacemi ze senzorů a pravidelně načítá oblast s daty.
- **data/device_sensor_data.php** – implementuje oblast se zobrazováním dat ze senzorů. Vykresluje jednotlivé body do grafu.
- **data/front.php** – soubor obsahující data titulní stránky.
- **data/interface.php** – vykresluje obsah levého panelu a vkládá požadovaný soubor do těla stránky.
- **data/login.php** – zobrazuje požadavek na přihlášení.

Dále jsou v kořenu serveru přítomny složky `images`, `js` a složka `styles`. V `images` jsou přítomny obrázky, které jsou součástí grafického rozhraní. Ve složce `styles` je uložen soubor `style.css`, který definuje vlastnosti jednotlivých prvků zobrazované stránky. Složka `js` obsahuje JavaScriptové soubory `jquery-1.9.1.min.js` a `tools.js`, kde první z nich je knihovna použitého frameworku jQuery. Soubor `tools.js` obsahuje naše vlastní funkce používané v grafickém rozhraní.