VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

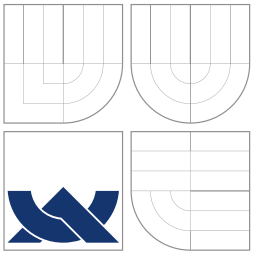
KOMUNIKACE A OVLÁDÁNÍ PRVKŮ ARDUINO S VYUŽITÍM MOBILNÍHO ZAŘÍZENÍ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

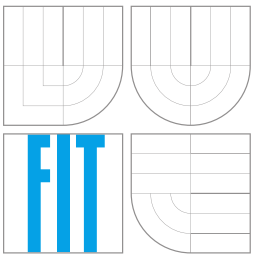
AUTOR PRÁCE
AUTHOR

ZBYNĚK MORAVEC

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

KOMUNIKACE A OVLÁDÁNÍ PRVKŮ ARDUINO S VYUŽITÍM MOBILNÍHO ZAŘÍZENÍ

COMMUNICATION AND CONTROLLING ARDUINO COMPONENTS WITH USING MOBILE DEVICE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ZBYNĚK MORAVEC
AUTHOR

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

BRNO 2015

Abstrakt

Cílem této práce je popsat základy použití bezdrátových technologií na platformách Android a Arduino. Dále se práce zabývá návrhem a implementací knihovny pro usnadnění vzájemné komunikace těchto platforem. Tato knihovna umožňuje komunikaci přes několik bezdrátových technologií a poskytuje přístup k funkcím platformy Arduino.

Abstract

The aim of this thesis is to describe basics of using wireless technologies on Android and Arduino platforms. The text also deals with design and implementation of library for simplifying communication between these platforms. The library ensure communication via several wireless technologies and provide access to functions of Arduino.

Klíčová slova

Android, Arduino, AVR, Bluetooth, WiFi, knihovna, dálkové ovládání

Keywords

Android, Arduino, AVR, Bluetooth, WiFi, library, remote control

Citace

Zbyněk Moravec: Komunikace a ovládání prvků Arduino s využitím mobilního zařízení, bakalářská práce, Brno, FIT VUT v Brně, 2015

Komunikace a ovládání prvků Arduino s využitím mobilního zařízení

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana pana Ing. Jana Samka Ph. D. Všechny zdroje, ze kterých jsem čerpal, jsem uvedl v seznamu použité literatury.

.....
Zbyněk Moravec
18. května 2015

Poděkování

Tímto bych chtěl poděkovat vedoucímu své bakalářské práce, Ing. Janu Samkovi, Ph.D, za odborné vedení, praktické připomínky a poskytnutí hardwarových komponent.

Obsah

Kapitola 1

Úvod

Mobilní telefony již dlouho neslouží pouze pro textovou či hlasovou komunikaci. Postupem času začaly podporovat stále více funkcí. Dnes je běžně používáme pro organizaci kalendáře, elektronické pošty, procházení webů či hraní her. Interakce s mobilním zařízením neprobíhá pouze pomocí tlačítek, ale pomocí dotykového plochy displeje, která kromě základních doteků je schopna rozpoznat také více-dotyková gesta. Jako vedlejší či hlavní způsob ovládání aplikace mohou sloužit také jednotlivé senzory, ať už jde o akcelerometr, senzor přiblížení, GPS a další. Za standard chytrých telefonů lze považovat také konektivitu pomocí Bluetooth či WiFi. Právě díky těmto vlastnostem jsou chytré mobilní telefony vhodné jako dostupný nástroj pro ovládání zařízení. Hlavními operačními systémy na tomto poli jsou iOS, Windows Phone a Android. Tato práce se bude zabývat pouze třetím zmiňovaným systémem, neboť zaujímá většinu těchto zařízení.

Chceme-li vytvořit elektronické zařízení, které bude softwarově programovatelné a schopné interagovat s elektrické obvody, pravděpodobně zvolíme jako základ aplikace jeden z velkého množství mikrokontrolérů. Přestože lze dnes tyto mikrokontroléry programovat z vyšších programovacích jazyků, nebývá to jednoduché. Především z potřeby znalosti problematiky mikrokontrolérů, nutnosti procházení datových listů a hledání vhodných registrů a jejich nastavení pro provedení potřebné činnosti. Konkrétní znalost těchto registrů navíc nemusí být přímo přenosná na jiný mikrokontrolér. Dnes lze jako nejpopulárnější platformou pro zjednodušení práce s mikrokontroléry označit Arduino. Kromě zjednodušení práce s registry (V/V piny, PWM, A/D převodník, ...) jsou k dispozici také knihovny pro externí moduly.

Hlavním cílem práce je poskytnout rozšiřitelný základ pro komunikaci mezi těmito platformami a snadnou tvorbu vzájemně komunikujících aplikací.

1.1 Obsah

Cílem této práce je základní seznámení s možnostmi bezdrátové komunikace mezi mobilní platformou Android a hardwarové platformy Arduino. Popsány jsou především běžné dostupné technologie pro komunikaci v reálném čase, tedy Bluetooth (Low Energy) a WiFi. Dále bude popsána knihovna pro usnadnění tvorby takto komunikujících zařízení. Hlavním činitelem je vždy mobilní zařízení, které interaguje se vzdáleným Arduinem. Především běžnými příkazy, jako je například nastavení napětí výstupního pinu, či přečtení stavu analogového vstupu. Komunikace může proběhnout přes několik podporovaných rozhraní, které abstrahují konkrétní (bezdrátovou) technologii. Pro jednotnou konfiguraci obou zařízení je možné využití konfiguračního souboru, pomocí kterého je vygenerována dvojice zdrojových

kódů. Ty pak poskytují konfigurační parametry pro obě platformy. Tyto vlastnosti jsou prezentovány v podobě bezdrátového vozítka řízeném pomocí Bluetooth. Poslední kapitolu tvoří výsledky testování bezdrátových technologií.

Kapitola 2

Arduino

V této kapitole se dozvíte základní informace o otevřené platformě Arduino, dále také obecné informace o bezdrátových komunikačních prostředcích spolu s použitím na této platformě.

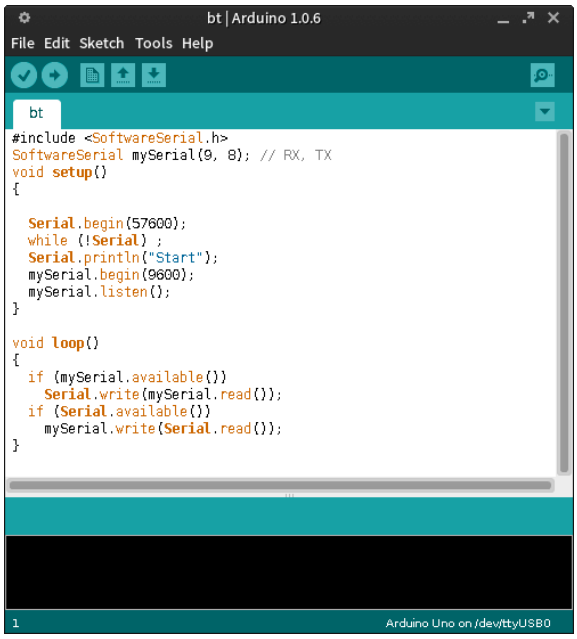
2.1 Popis platformy

Arduino je otevřená hardwarová a softwarová platforma určená především k prototypování. Nabízí se zde několik druhů desek lišících se funkcemi i zaměřením. Jednotlivé softwarové Arduino projekty se nazývají sketche. Jsou spravovány v oficiálním multiplatformním IDE, které kromě psaní kódu a programování hardwaru poskytuje také sériový terminál. Schopnější alternativou k tomuto prostředí je použití například Microsoft Visual Studio spolu s doplňkem Visual Micro. Programování probíhá v jazyce odvozeném z jazyka Wiring - podobný C++. Pro překlad je využívám překladač `avr-gcc` distribuovaný spolu s vývojovým prostředím.

Srdcem desek Arduino bývá většinou mikrokontrolér firmy Atmel. Produkty obvykle poskytují USB rozhraní, pomocí kterého lze Arduino ladit či programovat s využitím bootloaderu. Jednoduchou Arduino aplikaci lze provozovat třeba i na obvodu na nepájivém poli skládajícím se z pouze z několika součástek. Řešení vlastního plošného může být výhodné v situaci, kdy v cílové aplikaci nemáme dostatečný prostor pro hotové řešení, či pokud chceme minimalizovat cenu výrobku. Schémata desek Arduino jsou volně k dispozici, při tvorbě plošného spoje se lze tedy inspirovat. Program do mikrokontroléru lze nahrát pomocí rozhraní SPI. Lze využít jednoduchý programátor STK200 pro paralelní (LPT) port, který je možné složit výhradně z rezistorů. Často používaný je také USBasp[1] či USBtinyISP[2] pro USB sběrnici.

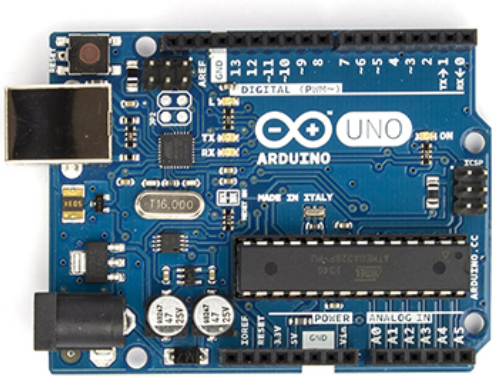
Samotné Arduino desky obvykle poskytují pouze základní periferie - vývody mikrokontroléru a USB rozhraní. Toto je dostatečné pouze pokud si v aplikaci vystačíme s LED diodami, spínači a jinými jednoduchými součástkami. Pokud však má naše aplikace komunikovat přes bezdrátové rozhraní, spínat energeticky náročná zařízení, zobrazovat na displeji či používat jinou netriviální technologii, je nejsnadnější cestou využit modul Arduina, nazývaný shield[3]. Tyto shieldy po připojení zakrývají celou Arduino desku, včetně konektorů. Tyto konektory jsou vyvedeny na shield, ve stejném rozložení jako na původním

[1] http://www.fischl.de/usbasp/
[2] https://learn.adafruit.com/usbtinyisp
[3] Seznam oficiálních i neoficiálních shieldů http://shieldlist.org/.



Obrázek 2.1: Ukázka IDE

Arduinu. Je tedy možné připojit další shield, či jiné periferie. Výhodou tohoto řešení je, že není třeba řešit propojení desek pomocí drátů - kde je možné udělat, ať už z neznalosti či nepozornosti, chybu. Ke shieldům jsou vždy k dispozici softwarové knihovny, které zaštiťují práci se shieldem.



Obrázek 2.2: ArduinoUno - jedna z oficiálních Arduino desek. Převzato z [10]

Arduino však není uzavřený ekosystém, kde by bylo možné používat pouze tyto shieldy. Na trhu jsou různé univerzální moduly, které často komunikují například přes univerzální sériové rozhraní UART. Nevýhodou těchto modulů je problém s fyzickým uchycením a zabrání potřebných pinů Arduina - např. napájecích pinů. Výhodou často může být cena. Některé moduly prakticky stačí připojit a používat - jako je například HC-05 (kapitola 2.2.2), u jiných není obsluha takto triviální. U těch je třeba obsluhu doprogramovat - např. řízení pomocí AT příkazů. Arduino je však natolik rozšířenou platformou, že neoficiální

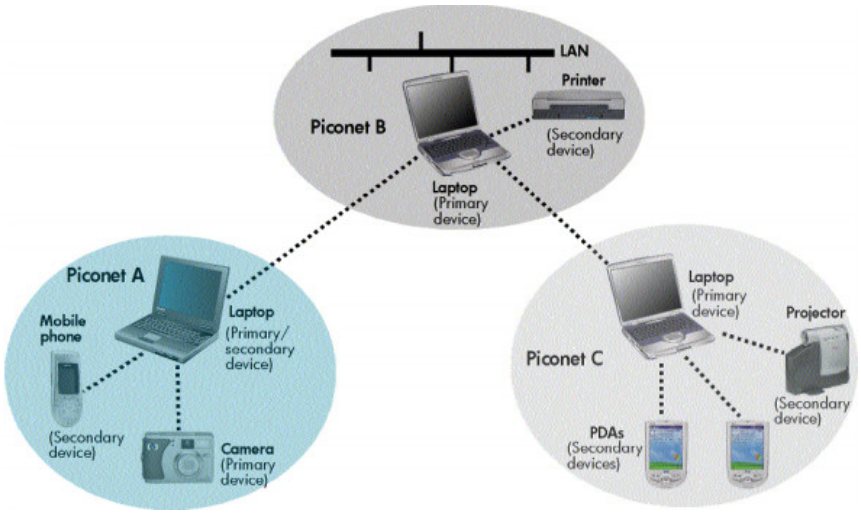
knihovnu lze téměř vždy získat.

2.2 Bluetooth

Bluetooth je bezdrátová technologie pro propojení zařízení na krátkou vzdálenost. Její prioritou je nízká cena spolu s nízkými energetickými požadavky. Původní účel byl nahradit proprietární kabelové spojení mezi zařízeními. Dnes se lze s Bluetooth setkat v mobilních telefonech, noteboocích, počítačových myších, tiskárnách, Hi-Fi systémech a dalších.

Bluetooth pracuje ve volném ISM[4] 2.4GHz pásmu, ve kterém se vyskytuje poměrně velké množství rušení. Bluetooth se s tímto zarušením vyrovnává pomocí techniky zvané `frequency hopping`. Data jsou rozdělena do jednotlivých paketů, které si vysílač a přijímač určité frekvenci, ale následující paket je však odeslán na jiné pseudonáhodné frekvenci. V případě, že se během přenosu vyskytne rušení, zasáhne přenos pouze tedy na velmi krátký okamžik. [2, str. 8]

Síťová topologie bluetooth se nazývá `piconet`. Skládá se z jednoho hlavního zařízení - `master` a až 7 vedlejších - `slave`. Master určuje vzor pro frequency hopping, kterým se řídí zbytek piconetu. Topologie ve které několik síti piconet sdílí společná zařízení se nazývá `scatternet` viz 2.3. [24]



Obrázek 2.3: Ukázka topologie scatternet. Obrázek převzat z [24]

Podle vysílacího výkonu (dosahu) zařazujeme zařízení do několika tříd: [3, str. 81]

Třída	**Vysílací výkon**	**Dosah**
Class 1	100mW	100m
Class 2	2.5mW	10m
Class 3	1mW	1m

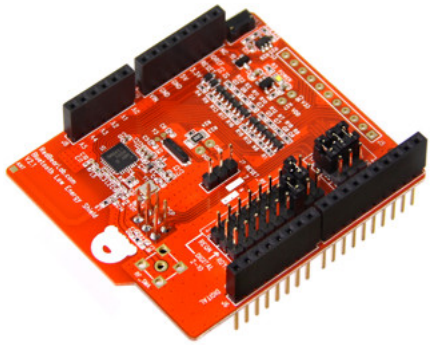
[4]Industrial, Scientific and Medical

2.2.1 Bluetooth Low Energy

Tato technologie je označována také jako Bluetooth Smart, či zkráceně Bluetooth LE nebo BLE. Bluetooth Smart není obecně kompatibilní s předchozí verzí Bluetooth (Classic). Lze se však setkat se zařízeními označenými jako `Bluetooth Smart ready`, které podporují obě zmiňované varianty. BLE nelze považovat za úspornější náhradu předchozí verze Bluetooth, neboť obě varianty jsou vhodné pro jiný typ aplikace. Ideálním určením jsou aplikace vyžadující zřídkavé odesílání dat, jako například hodnoty tepu, teploty apod. Teoretická rychlost se pohybuje okolo 260kb/s[5]. Data jsou vysílána v krátkých intervalech mezi 7.5ms až 4s, prokládanými vypínáním rádia. Zkrácením těchto intervalů sice lze zvýšit propustnost, efektivita však bude horší než u Bluetooth Classic [5, str.20]. Bluetooth LE v současné době není tak rozšířené jako Bluetooth Classic[6].

Knihovna Android2Arduino využívá pro svou potřebu knihovnu třetí strany[7], která poskytuje zjednodušené rozhraní pro práci s modulem.

Obrázek BLE shieldu převzat z `http://redbearlab.com/bleshield`.



Obrázek 2.4: BLE shield od RedBearLab.

2.2.2 Serial port Bluetooth module

Jako nejjednodušší řešení Bluetooth konektivity se jeví použití modulu s SPP[8]. Zde bude popsáno použití modulu HC-05, alternativní moduly však budou mít použiti velmi podobné. Bohužel i popisovaná varianta je na trhu k dispozici v různých variantách. Samotný modul bývá umístěn na plošném spoji s podpůrnou elektronikou a množina výstupních pinů, včetně jejich uspořádání se mohou lišit. Proto je třeba se během zapojení orientovat podle popisu pinů na konkrétním modulu.

Zapojení HC-05

Komunikace probíhá přes UART[9] rozhraní. Modul je konfigurovatelný pomocí AT příkazů. V případě, potřeby zadávání AT příkazů, je třeba během startu komponenty mít zapojený pin `KEY` na vysoké úrovni. S tímto pinem se můžeme setkat buď v hlavním konektoru

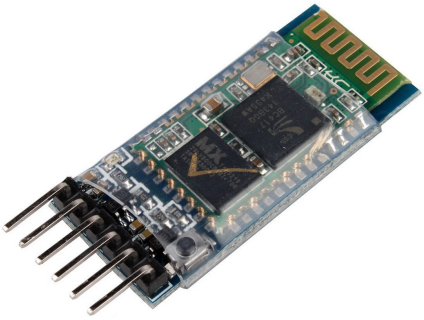
[5] Podle zdroje [5]. Jiné zdroje uvádějí mírně lišící se hodnoty.

[6] Podle údajů z [26] má verzi Android 4.3 a vyšší 55% zařízení. Což je softwarová podmínka pro podporu této technologie.

[7] `https://github.com/RedBearLab/nRF8001`

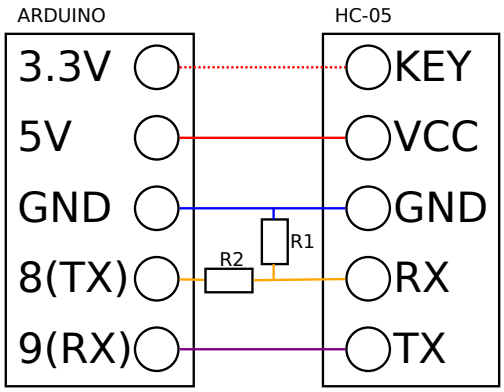
[8] Serial port profile

[9] Universal asynchronous receiver/transmitter



Obrázek 2.5: Jedna z variant bluetooth modulu HC-05

modulu, spínači na plošném spoji, v nejméně pohodlném případě jej lze získat přímo z menšího plošného spoje modulu. Nastavení modulu je často jednorázová záležitost, je tedy možné potřebný vstup připojit pouze přiložením kontaktu. Modul HC-05 používá logickou 1 na napětí 3.3V, avšak Arduino obvykle používá úrovně 5V[10]. Je vhodné výstupní 5V napětí snížit před zapojením do modulu. Výstupních 3.3V z modulu správně spadá do logické 1 na Arduinu, není tak třeba další elektroniky. Na obrázku 2.5 lze vidět jednu z variant tohoto modulu[11].



Obrázek 2.6: Příklad připojení HC-05 k Arduino UNO

Zobrazené zapojení využívá děliče napětí, kde ideální hodnoty rezistorů R1:R2 jsou v poměru 3.3:1.7. Pin `State`[12] lze využít například pro indikací připojení pomocí LED diody. Často ovšem bývá již samotný modul vybaven indikační diodou, je tedy užitečnější připojit tento vývod na některý ze vstupů Arduina, který umožňuje čtení digitálního vstupu, popř. podporuje vyvolání přerušení při změně této hodnoty. Vysoká úroveň na tomto pinu signalizuje připojené zařízení.

[10] Závisí na verzi Arduina.

[11] Převzato z http://www.amazon.com/JBtek-Wireless-Bluetooth-Transceiver-Arduino/dp/B00L083QAC.

[12] Možná jej naleznete pod označením `Led`.

AT příkazy

AT příkazy byly původně určeny ke konfiguraci telefonních modemů. Slouží také pro konfiguraci modulu HC-05. Jedná se o textové příkazy ve formátu `AT+<příkaz>\r\n`. [28]
Ověření reakce na AT příkazy zjistíme nejsnadněji pomocí odeslání příkazu `AT\r\n`.

```
AT
OK
```

Příklad další konfigurace je k nalezení v příloze G.

Programová část

Jelikož standardní sériové rozhraní mikrokontroléru bývá často již využito pro komunikace přes USB, je výhodné použít knihovnu `SoftwareSerial` která poskytne sériové rozhraní i na ostatních datových pinech. Nutno upozornit, že aktuální implementace nepodporuje tzv. loopback[13].

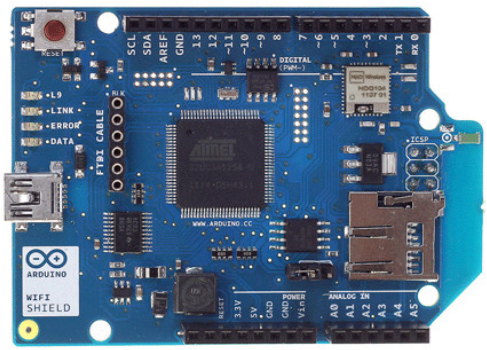
Konstruktor instance SoftwareSerial obsahuje označení RX, TX pinu, následná metody instance jsou stejné jako metody `Serial`.

Kód 2.1: Použití SoftwareSerial.

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(9, 8); // RX, TX
mySerial.write(c);
```

Ukázka kódu dostupná v příloze E

2.3 Wi-Fi



Obrázek 2.7: Oficiální Wifi shield. Obrázek převzat z [16]

Arduino pro svou síťovou konektivitu poskytuje Wifi shield, který spolu s knihovními funkcemi poskytuje snadnou obsluhu TCP či UDP protokolu. Ihned po připojení do sítě již modul automaticky může získat IP adresu z DHCP serveru, pro běžné použití tedy není třeba řešit nic víc než implementaci protokolu na aplikační vrstvě.
Při prvotním použití shieldu je však možné se potkat s jistou nekompatibilitou. Z hlediska hardware se jedná o použití starších Arduino desek[14], kde je nutné připojit pin IOREF

[13] Přímé přípojení RT <->TX. Jednoduchá cesta k otestování spojení.
[14] Podrobnější informace o konkrétních deskách na http://arduino.cc/en/Guide/ArduinoWiFiShield.

na referenční napětí. Nejvhodnějším řešením je použití drátové propojky k propojení pinu IOREF a 3.3V na Wifi shieldu.
Druhým, softwarovým, problémem může být neaktuální verze firmwaru Wifi shieldu. Firmware lze jednoduše aktualizovat s využitím USB konektivity na shieldu. Pro programování je nutné propojit propojku J3[15]. Tento konektor by měl zůstat nepřipojený pro běžné použití. Shield je doporučeno odpojit z hlavní desky během programování. Po této přípravě hardwaru nastane fáze aktualizace firmwaru. Tento postup zde nebude popsán, neboť je jeho provedení na různých operačních systémech různé. Postup lze nalézt na http://arduino.cc/en/Hacking/WiFiShieldFirmwareUpgrading. Ukázky připojení k Wifi sítím je k dispozici v příloze E

Kód 2.2: Wifi TCP.

```
WiFiServer server(12345); // vytvoreni instance serveru s~danym
    portem
  server.begin();

  WiFiClient client = server.available(); // ziskani instance klienta
     "accept"
  if (client) {
    while (client.connected()) {

      // precist data klienta pokud jsou dostupna
      if (client.available()) {
        char c = client.read();
      }
      client.write('A'); // odeslani dat klientovi

    client.stop(); // ukonceni spojeni s~klientem

  }
```

Knihovna Android2Arduino umožňuje využití TCP spojení tohoto shieldu pro své účely. Jelikož je díky používání DHCP[16] serverů na síti problémové určit adresu cílové zařízení, je implementována možnost využít UDP broadcastu[17] pro informování o adrese cíle. Rozesílaný packet je ve formátu <IPv4 adresa><port>. Je-li jako udaná IP adresa použita adresa 0.0.0.0, připojí se zařízení na adresu odesílatele, jinak se připojí na explicitně udanou adresu. Ve verzi 1.2.0 oficiální knihovny shieldu se však nachází chyba, která způsobuje nesprávnou funkci TCP socketu v souvislosti s použitím UDP[18]. Uskutečnění spojení mezi Androidem a Arduinem je možné buď pomocí společného přístupového bodu, či pomocí vytvoření přenosného Wifi hotspotu na mobilním zařízení.

2.4 GSM

Dnes zkratkou GSM myslíme **G**lobal **S**ystem for **M**obile Communications [18]. Do tohoto systému se v dnešní době zapojuje prakticky každý pomocí svého mobilního telefonu. Původně se tento systém používal pro analogový přenos hlasu. Pozdější generace již byly

[15] Na desce není označená jménem. Jedná se o konektor pod micro-SD card slotem.
[16] Dynamic Host Configuration Protocol
[17] všesměrové vysílání
[18] https://github.com/arduino/Arduino/issues/1732

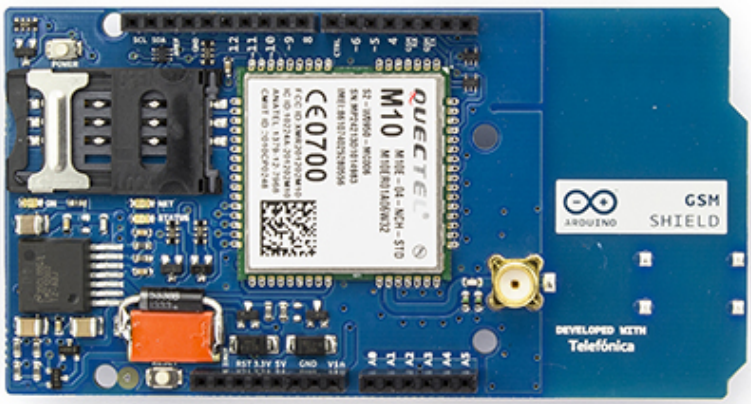
digitální a začaly podporovat datové přenosy [19]. Dnes není výjimkou používání sítě GSM pro mobilní připojení k internetu.

GSM shield

Tento shield poskytuje prakticky všechno co lze považovat za základ klasického mobilního telefonu. S použitím externího mikrofonu a reproduktoru lze uskutečňovat hovory. Bez externích komponent je shield možné využít pro přijímání/odesílání SMS či připojení k sítí internet. Po přihlášení shieldu do GSM sítě je práce s TCP[19] podobná jako při používání Ethernet/Wifi shieldu. UDP[20] je shieldem dle datového listu [20] hardwarově podporováno, bohužel knihovna Arduina pro něj neposkytuje podporu. Komunikace knihovny s modulem probíhá pomocí AT příkazů.

Shield se připojuje k Arduinu běžným způsobem. GSM modul má však příliš vysoký odběr na to aby byl napájený pouze z USB. Je třeba připojit externí zdroj k Arduinu, který dokáže poskytnou proud 0.7A až 1A [17]. Podle oficiálních stránek může shield ve špičce odebírat až 2A, ale kondenzátor shieldu by měl tento výkyv pokrýt.

Přibalená SIM karta slouží pro tvorbu aplikace v které komunikují jednotlivá zařízení mezi sebou. Po aktivaci a dobití kreditu je možné mezi těmito kartami přijímat a zasílat SMS zprávy, bez přístupu k „běžným" SIM kartám. Oproti běžné SIM kartě podporující plnou funkčnost modulu je výhoda aktivovaného roamingu v podporovaných zemích[21].



Obrázek 2.8: Oficiální GSM shield. Obrázek převzat z [17]

[19] Transmission Control Protocol protokol zajišťující spolehlivý přenos dat
[20] User Datagram Protocol protokol pro nespolehlivý přenos dat
[21] http://arduinosim.movilforum.com/service.php

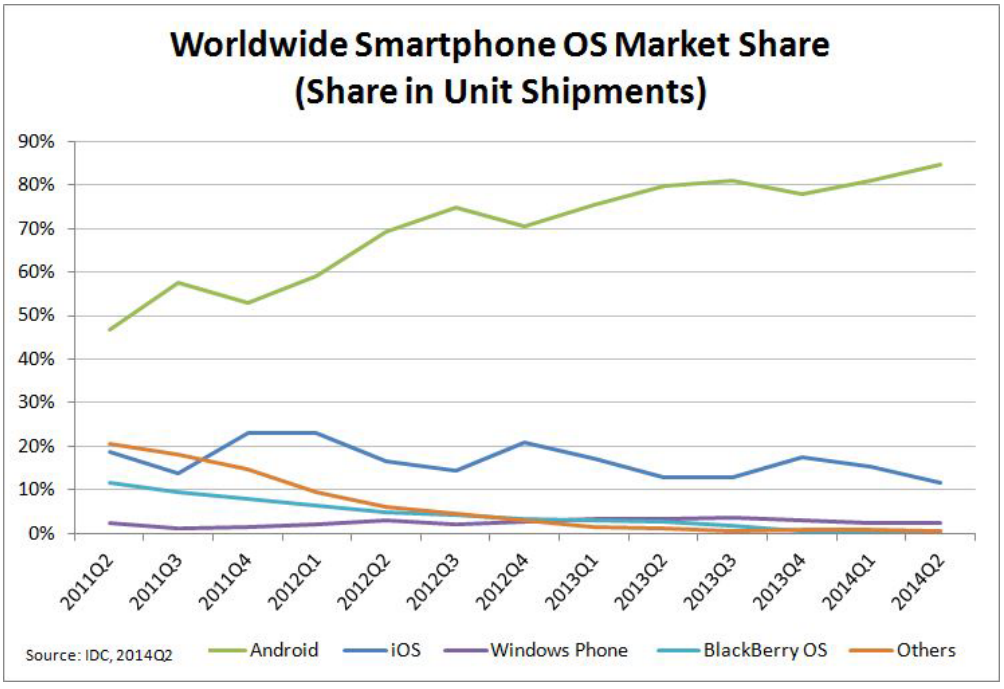
Kapitola 3

Android

Tato kapitola pojednává o mobilní platformě Android, základech problematiky vývoje pro tuto platformu a o použití bezdrátových technologií.

3.1 Popis platformy

Android je operační systém založený na Linuxu. Jeho určení je primárně pro mobilní zařízení - mobilní telefony, tablety či třeba hodinky. Díky verzi pro x86 procesory je možné jej provozovat i na klasickém osobním počítači. V současné době se jedná o nejrozšířenější platformu v segmentu svého určení. Graf převzat z [8].



Obrázek 3.1: Tržní podíl OS na poli smartphonů

3.2 Vývoj aplikací

Pro Android je možné vyvíjet nativní aplikace s pomocí Android NDK[1] v C/C++[2]. Tato možnost však není preferována, neboť výkonová výhoda často nepřevyšuje nevýhody komplikovaného vývoje.

Preferovaným způsobem je psaní aplikací v jazyce Java. K dispozici jsou v současné době dvě oficiální vývojové platformy - první je Eclipse s ADT[3] pluginem, druhou platformou je Android Studio založenou na IntelliJ IDEA. Přestože první stabilní verze Android Studia byla vydaná poměrně nedávno, dlouhou dobu byla k dispozici beta verze, takže jej lze považovat za preferovanou a spolehlivou platformu.

Android poskytuje snadný způsob vývoje pro různá zařízení/jazykové mutace. Grafické prvky jsou umístěny v adresářích podle příslušného typu displeje. Je tak možné snadno řešit využití grafického obrázku vyššího rozlišení pro větší displeje bez programového řešení. Stejně tak jsou jednotlivé řetězce uloženy mimo programový kód, takže je snadné udělat jazykový překlad aplikace bez nutnosti zasahovat do kódu programu.

Grafický design aplikace lze tvořit pomocí XML kódu či WYSIWYG[4] editoru. V praxi bývá ideální oba tyto přístupy kombinovat.

Pro ladění programů na zařízení je k dispozici ADB (Android Debug Bridge), který kromě možnosti krokování programu, zasílání ladících zpráv umožňuje přesměrování portů, kopírování souborů, přístup k textovému rozhraní či instalaci/spouštění aplikací (využívané především pro vývoj).

Ladění probíhá standardně pomocí USB kabelu. K zařízením na kterých máte root[5] oprávnění, můžete připojit ladící most také přes síť [6].

Jako výtku programování na platformě Android lze zmínit nevhodně časté úpravy API.

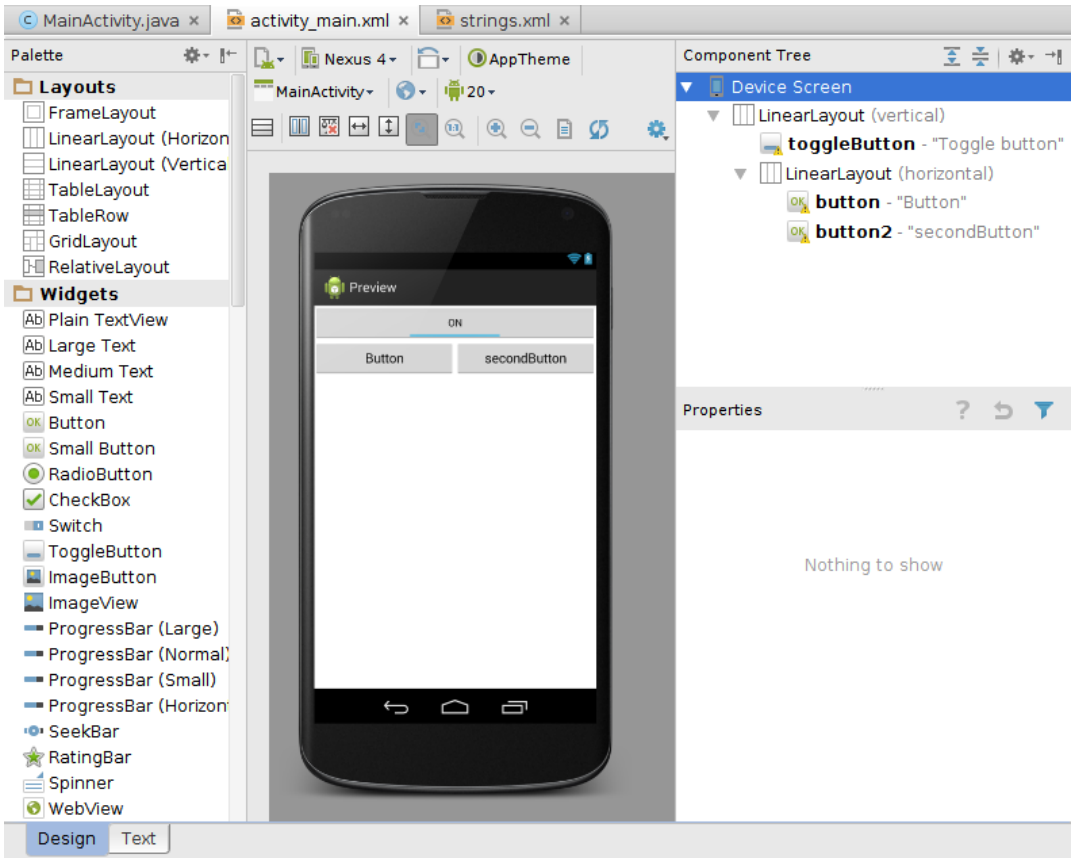
3.2.1 Fyzická a virtualizovaná zařízení

Aplikace je možno vyvíjet i bez fyzického zařízení. K tomuto účelu je k dispozici Android Emulator, který umožňuje jak plnou emulaci ARM[7] procesoru, tak virtualizaci x86 procesoru, která je znatelně rychlejší. K dispozici jsou i další alternativy, jako je použití Android-x86[8] na běžném/virtualizovaném počítači nebo například rychlý Android Emulátor GenyMotion[9], poskytuje například pohodlné „vnucení" hodnot do periferních zařízení. Na obrázku lze vidět emulátor Genymotion, v pravém panelu se nachází ikony pro nastavení stavu baterie, GPS, zdroje kamery.

3.2.2 Oprávnění

Ochrana systému je, mimo jiné, realizována pomocí požadavků oprávnění jednotlivých aplikací - např. přístup k externímu úložišti, využívání schránek apod[10]. Uživatel systému tak může usoudit, zda aplikace vyžaduje nerelevantní přístupy a mohla by být nebezpečná.

[1] https://developer.android.com/tools/sdk/ndk/index.html
[2] Například Qt Creator umožňuje vývoj aplikací následně přeložitelných jak pro PC tak Android.
[3] Android developer tools
[4] What you see is what you get
[5] Správce zařízení
[6] adb connect <ip adresa>
[7] Advanced RISC Machine
[8] http://www.android-x86.org/
[9] http://www.genymotion.com/
[10] Kompletní seznam na [6].



Obrázek 3.2: Navrhování grafického rozhraní

Kód 3.1: Seznam požadovaných oprávnění je uveden v manifestu aplikace.

```
<uses-permission android:name="{oprávnění}" />
```

3.3 Bezdrátové komunikační prostředky

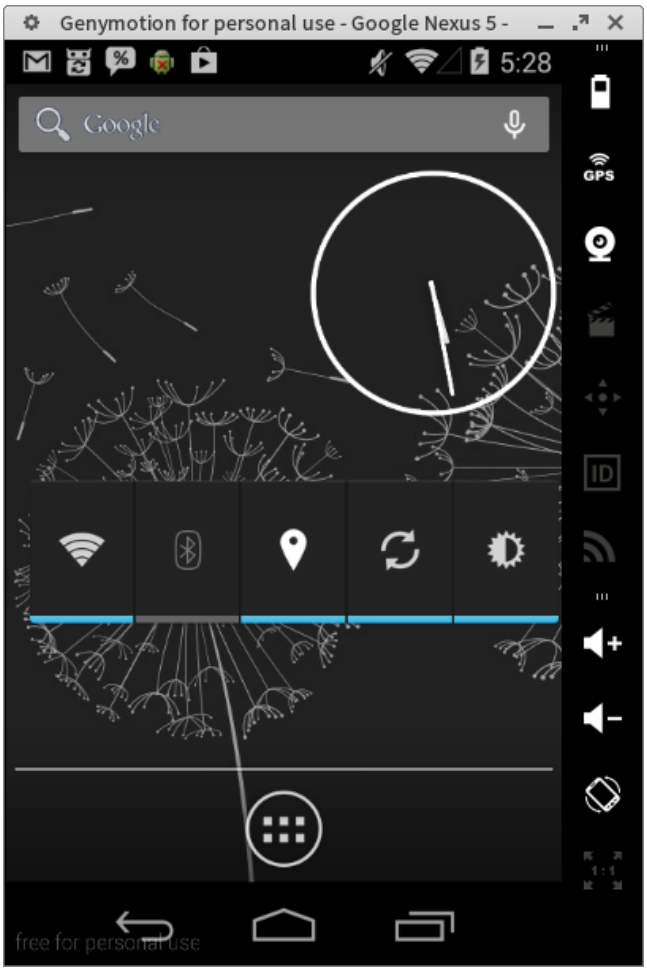
Přístup k některým funkcím systému není vždy příliš intuitivní/snadno zapamatovatelný, proto byly jednotlivé akce zapouzdřeny do konkrétních metod za účelem lepší názornosti. Tyto metody naleznete v příslušných přílohách.

3.3.1 Bluetooth

Konkrétní adaptér reprezentuje instanci třídy `BluetoothAdapter`[11], která lze získat statickou metodou z této třídy. Od verze API 18 lze instanci získat ze systémové služby BLUETOOTH_SERVICE. Instance rozumí očekávaným zprávám, jako je povolení/zakázaní, získání stavu či seznamu spárovaných zařízení.

Získání výchozího adaptéru

[11] V práci nebudou uváděny kompletní názvy tříd (včetně balíčků), jejich název lze získat z oficiálního webu http://developer.android.com/ či automatickým doplněním IDE.



Obrázek 3.3: Pohled na Genymotion

Kód 3.2: Získání výchozího adaptéru.

```
// ziskani apdateru API <= 17
BluetoothAdapter adapter = BluetoothAdapter.getDefaultAdapter();

// ziskani adapteru API > 18
Context ctx = ...;
BluetoothManager manager = (BluetoothManager) ctx.getSystemService(
    Context.BLUETOOTH_SERVICE);
adapter = manager.getAdapter();
```

K bluetooth adaptéru lze zaregistrovat jednotlivé služby, které se z hlediska programování chovají podobně jako TCP server/klient.

Kód 3.3: Zaregistrování RFCOMM služby.

```
public BluetoothServerSocket startRFCOMMService() throws
    IOException {
    return adapter.listenUsingRfcommWithServiceRecord(this.
        serviceName,
```

```
            UUID.fromString("00001101-0000-1000-8000-00805F9B34FB")
            );
    }
```

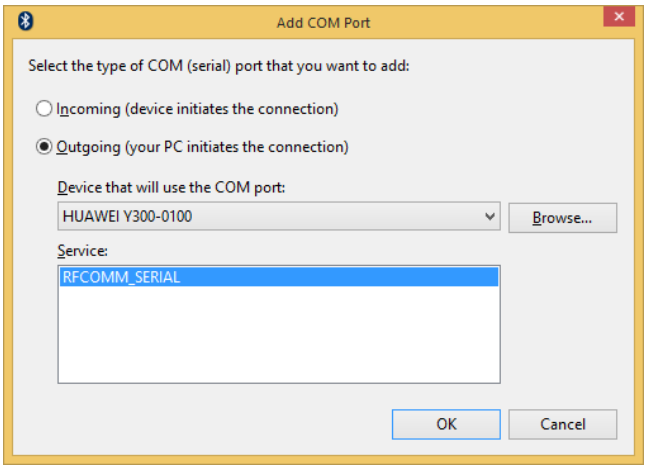
Za zmínku stojí stojí snad jen metoda BluetoothServerSocket.accept(), která vrací BluetoothSocket, tedy socket reprezentující konkrétního klienta. Metoda `accept()` je blokující a není tedy umožněno ji volat v hlavním vlákně programu, neboť to Android nepovoluje. Další metody budou popsány v příloze B.2.

Service Discovery Protocol - SDP

Tento protokol je určen k výměně informací o běžících službách mezi jednotlivými zařízeními. Každá služba je identifikována UUID viz F. [3]

SPP/RFCOMM

Jedná se o emulovaný seriový port. Komunikace probíhá transparentně na vrstvě L2CAP[12], která zajišťuje spolehlivý přenos dat. [3]
Pokud se k takovému zařízení chceme, například z ladících účelů, připojit z operačního systému Windows, provedeme to pomocí menu Nastavení Bluetooth > COM porty > Přidat Jelikož se jedná pouze o emulaci seriového portu, při komunikaci není třeba nastavovat konkrétní parametry (Rychlost přenosu - Baud rate, ...). Data budou poslána maximální možnou rychlostí v danou chvíli. Mezi klientem a serverem je možné navázat až 60 jednosměrných spojení (30 obousměrných).



Obrázek 3.4: Přidání seriového portu v systému Windows 8

3.3.2 Wifi

Wifi je v dnešní době populární technologie k bezdrátovému připojení na větší vzdálenosti s požadavkem na vyšší datový tok. Pojem Wifi zašťiťuje několik standardů [3], mezi běžně používané patří následující:

[12]Logical link control and adaptation protocol

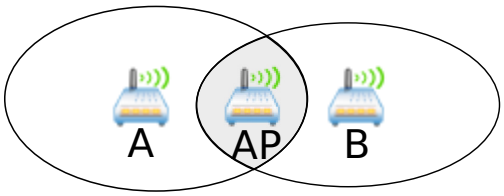
Název standardu	Maximální teoretická rychlost	Frekvenční pásmo
IEEE 802.11a	54Mbps	5GHz
IEEE 802.11b	11Mbps	2.4GHz
IEEE 802.11g	54Mbps	2.4Ghz
IEEE 802.11n	600Mbps	2.4Ghz a 5Ghz

Wifi zařízení mohou spolupracovat ve dvou operačních módech. Prvním módem je **Infrastruktura**. Jedná se o nejběžnější druh připojení. Topologii tvoří přístupový bod (AP) a jednotlivé stanice. Druhým módem je **Ad hoc**, kdy jednotlivé stanice mezi sebou komunikují bez třetího prvku.

Kolize na bezdrátové síti Wifi využívá princip CSMA/CA. Pokud by některá stanice začala vysílat v případě že v jejím okolí jiná stanice nevysílá, mohlo by se snadno stát, že v okolí cílové stanice již jiná vysílací stanice je. Tuto situaci může řešit mechanismus RTS/CTS:

1. Zdroj vyšle RTS packet obsahující mj. adresu zdroje, cíle, délku trvání přenosu.
2. Cíl odešle odpověď - CTS obsahující mj. délku trvání a adresu odesílatele RTS.
3. Zdroj odešle data.
4. Cíl odešle ACK.

Všechny stanice které obdržely RTS a CTS použijí tuto informaci k detekci používání sdíleného media. V případě že MAC rámec není značně větší než RTS/CTS, odešle se přímo bez RTS/CTS.



Obrázek 3.5: Vizualizace vysílání/kolizí

Implementace

O reprezentaci Wifi adaptéru se stará instance třídy `WifiManager`. Je možné ji získat jako systémovou službu.

Kód 3.4: Získání instance WifiManager.

```
// Ctx je instance třídy Context
WifiManager manager=(WifiManager)ctx.getSystemService(Context.
    WIFI_SERVICE);
```

Wifi adaptér umožňuje funkci hotspot, Android však neposkytuje oficiální rozhraní k ovládání této funkce. Existují implementace potřebného rozhraní využívající Java reflexi[13], toto řešení však nelze považovat za spolehlivé. Popis potřebných akcí je dostupný v příloze B.1

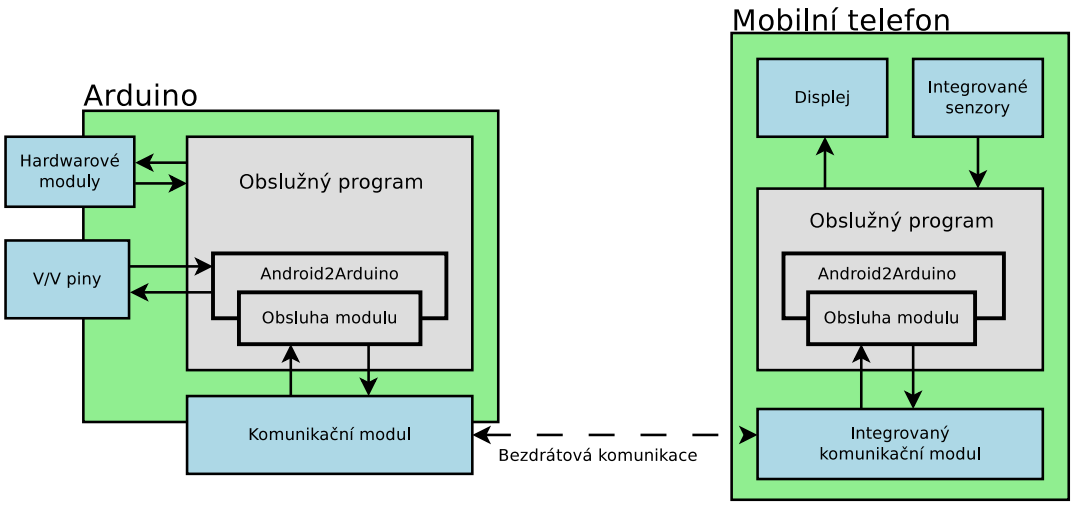
[13]Například: http://www.whitebyte.info/android/android-wifi-hotspot-manager-class

Kapitola 4

Knihovna Android2Arduino

V rámci této práce byla napsána knihovna Android2Arduino. Knihovna má za cíl poskytnout přístup z platformy Android k univerzálním funkcím Arduino jednotným způsobem, s možností specifického rozšíření pro danou aplikaci.

V první sekci 4.1 se dozvíte přehled funkcí knihovny z pohledu programátora. V další části je popsán formát serializace příkazů po síti - univerzální označení pinů, proměnných, funkcí 4.1.1. Následující dvě sekce se věnují použití knihovny na obou platformách, včetně instalace a popisu problémů s kterými se lze setkat. Kapitola 4.4 se věnuje generátoru konfiguračních souborů pro obě platformy, aby se zamezilo nekonzistenci v používání knihovny. Kapitola 4.5 se věnuje testování knihovny na Androidu a testování Arduino knihovny na počítači.



Obrázek 4.1: Struktura aplikace knihovny

4.1 Obecná implementace knihovny

Knihovna se skládá z části pro Android a části pro Arduino. Jelikož mobilní telefony mají nesrovnatelně více zdrojů než mikrokontroléry, byla snaha o uchování co největší části funkcionality právě na mobilních zařízeních. Funkci knihovny lze částečně ztotožnit s implemen-

tací RPC[1]. Jednotlivé příkazy lze řadit za sebe a následně celou dávku odeslat do Arduino zařízení. Dávkové odesílání může snížit režii u některých typů spojení. Zprávy jsou na Androidu přijímány ve formě callbacků, což umožňuje číst nejen odpovědi na příkazy, ale také zprávy iniciované Arduinem.

Knihovna poskytuje podporu (de)serializace datových typů spolu s podporou základních standardních funkcí platformy Arduino. Serializovaná data odesílána/přijímána skrze rozhraní abstrahující konkrétní přenosovou technologii. Obecnou část Java knihovny lze provozovat na stolním počítači, jelikož vazby na platformu Android jsou až skrze konkrétní přenosová rozhraní. Je možné například s využitím rozhraní pro sériový port komunikovat mezi desktopovou Java aplikací a Arduinem podobně jako z mobilního zařízení.

Podporovaná funkcionalita

- `analogWrite()`,
- `analogRead()`,
- `digitalWrite()`,
- `digitalRead()`,
- `pinMode()`,
- zápis proměnné,
- čtení proměnné,
- volání funkce,
- vlastní příkazy.

Vzdálená volání vestavěných funkcí Arduino jsou vhodná spíše pro jednodušší aplikace. Pro větší posloupnost akcí či specifičtější činnosti je výhodnější použít vzdálené volání uživatelských funkcí, což nám také umožní přistupovat většímu množství, například knihovních, metod.

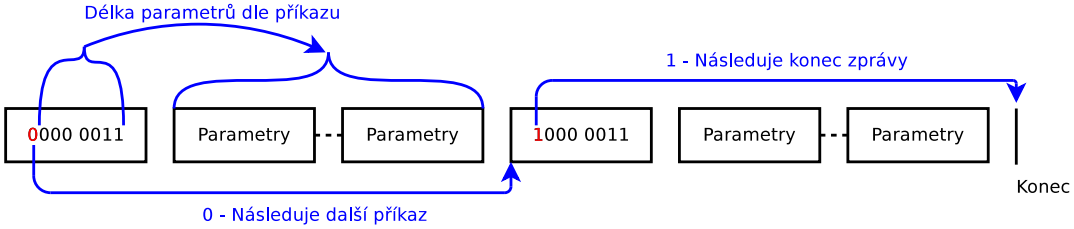
Během čtení a zápisu proměnné je nutné správně uvést index a typ proměnné. Přestože je obvyklé přenášet hodnoty přes síť ve formátu big-endian, knihovna využívá formátu little-endian. Je tomu tak proto, že naprostá většina relevantních procesorů využívá little-endian a neustálé překódovávání by způsobovalo pouze režii navíc. V případě, že by byla knihovna provozována na zařízení s opačnou endianitou, je zajištěno překódovávání - na straně Javy pomocí vlastností vestavěných tříd a na straně Arduina pomocí výběru vhodného algoritmu při překladu.

4.1.1 Formát serializace

Jednotlivé příkazy je identifikován 7bity. Osmi bitová hlavička příkazu se skládá z bitu označovaným `poslední příkaz` a zmiňovaného 7 bitového identifikátoru příkazu. Každý příkaz je třeba zpracovat za účelem přečtení dostatečného množství znaků, aby mohlo následně proběhnout rozpoznání začátku dalšího příkazu. Každý zpracovávaný příkaz musí sám rozhodnout o počtu znaků které přečte. To lze určit už podle typu příkazu, nebo až podle dat konkrétního příkazu. Počet znaků je nutné zpracovat správně, jinak je následné

[1] Remote procedure call

chování nedeterministické. Problém může vzniknout při implementaci vlastních funkcí. Ačkoliv tento způsob vkládá jistou zodpovědnost do rukou programátora, umožňuje dynamické odesílání trvalého proudu dat a také ušetřit režii příkazu v podobě explicitního kódování délky zprávy.



Obrázek 4.2: Znázornění formátu serializace

4.1.2 Serializace označení pinů

Na platformě Arduino se jednotlivé piny označují číselnou hodnotou. Piny připojené k A/D převodníku jsou označeny konstantami A0, A1, .. An, kde hodnota `n` závisí na zvoleném mikrokontroléru. Jelikož tyto konstanty často reprezentují jiné číslo, nelze je přímo adresovat z jiné platformy. Proto jsou jednotlivá označení pinů zakódována do jednoho bajtu, kde nejvyšší bit signalizuje, zda se jedná o analogový, či digitální pin. Na Arduinu po přijetí takto zakódovaného označení proběhne rozpoznání, zda se jedná o označení analogového pinu. Pokud ano, výsledná číselná hodnota pinu Arduino vypočítána pomocí přijaté hodnoty a konstanty A0. Jelikož piny podporující analogový vstup jsou často v jednom portu, lze předpokládat že bude platit $A_n + 1 = A_{n+1}$. Po nahlédnutí do zdrojových souborů Arduina[2] lze tento předpoklad pro všechny oficiální varianty s AVR i potvrdit.

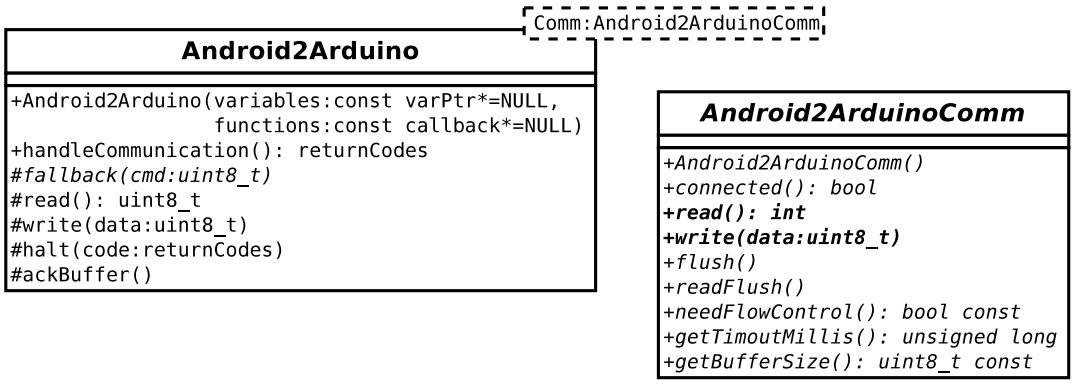
4.2 Implementace na Arduino

Knihovna se skládá z třídy `Android2Arduino` a abstraktní třídy `Android2ArduinoComm` sloužící jako rozhraní zaštiťující komunikační technologii. Experimentálně bylo zjištěno, že překladač nezvládá klasický objektový způsob volání metod skrze rozhraní ideálně optimalizovat. Za účelem možnosti sestavení knihovny spolu s volitelným komunikačním rozhraním bylo využito šablon jazyka C++. Překladač je tak schopen například optimalizovat zřetězené provolávání funkcí, či úplně vypustit překlad některých částí, které jsou pro dané rozhraní zbytečné. V knihovně je použito několik šablon, za účelem zpřehlednění použití pro programátora bylo vytvořeno několik odvozených tříd zakrývající parametry šablon. Zdrojové kódy knihovny tak musí být z velké části umístěny v hlavičkových souborech, což zapříčiňuje déle trvající překlad. Jelikož zdrojové kódy určené pro mikrokontroléry často nejsou příliš rozsáhlé, lze čas kompilace zanedbat.

Konstruktoru instance knihovny je předáván seznam ukazatelů na proměnné a seznam ukazatelů na uživatelské funkce, které mohou být volány.

Každá třída abstrahující datový přenos po různé technologii by měla být potomkem třídy `Android2ArduinoComm`. Využití dědičnosti není povinné, avšak zajišťuje kompatibilitu s knihovnou.

[2] /hardware/arduino/avr/variants/*/pins_arduino.h



Obrázek 4.3: Zjednodušený diagram tříd

Popis rozhraní:

- `read()` čtení znaku z rozhraní (-1 = nepřečteno),
- `write(data)` zapsání znaku na rozhraní,
- `flush()` vyprázdnění odchozí vyrovnávací paměti,
- `getTimeoutMillis()` v případě, že doba kontinuálního čtení s výsledkem `-1` z rozhraní dosáhne této hodnoty je metoda zpracovávající komunikaci ukončena s chybou hodnota -1 zde znamená, že timeout nemá být použit vůbec,
- `connected()` v případě chybného čtení je kontrolována tato metoda pokud již rozhraní není připojené, je zpracování komunikace ukončeno s odpovídající chybou v případě, že nelze zjistit reálný stav spojení, je doporučeno vracet trvale hodnotu `true` a spoléhat se pouze na timeout,
- `getBufferSize()` velikost vstupního bufferu.

Nejasnost může tvořit metoda `getBufferSize()`. Na mikrokontroléru je poměrně malá velikost přijímacího bufferu[3]. Některé komunikační prostředky neumožňují implicitní řízení toku. Tato kombinace vlastností by způsobovala časté zahlcení bufferu a následní zahazování další komunikace. Potvrzování volného místa ve vyrovnávací paměti je tedy řešeno programově, na základě velikosti bufferu a počtu zpracovaných bytů.
Zpracování zprávy proběhne zavoláním metody `handleCommunication()` vytvořené instance. Tato metoda zpracuje celou zprávu a následně se ukončí. Při zpracování určitého počtu znaků je na mobilní zařízení odeslána zpráva o volném místě ve vstupním bufferu. Nová data tak mohou přicházet během zpracování starých dat, což v některých případech může komunikaci urychlit.
Pro implementaci příkazu slouží přepsání metody `fallback()`, která je volána v případě, že přijímaný příkaz nebyl rozpoznán. Pokud není přepsána, vrací automaticky chybu pomocí metody `halt()`. Při každém čtení skrze `read()` může být kvůli chybě zpracování přerušeno. V případě, že toto chování je pro uživatelskou metodu nepřípustné, je možné si nejdříve načíst požadovaná data a až poté je zpracovat, či číst přímo z komunikačního rozhraní pomocí vlastnosti `comm`. Pokud programátor zvolí tento postup, tak v případě

[3]Výchozí hodnota pro UART je 64B.

že to rozhraní vyžaduje, musí během čtení zasílat informaci o zpracování bufferu pomocí metody `ackBuffer()`.

Ve výše zmíněném odstavci bylo psáno o ukončení zpracování při chybě čtení. Bohužel je čtení poměrně častou záležitostí a řešení zpracování chyb pomocí návratové hodnoty není příliš vhodné a to hlavně z důvodu zatemnění kódu. Běžným moderním způsobem zpracování podobných chyb jsou výjimky, ty však na této platformě nejsou podporovány. V našem případě, je možné výjimky plně nahradit pomocí dvojice funkcí `setjmp` `longjmp`. Tyto funkce umožňují návrat k uloženému stavu registrů - včetně instrukčního a ukazatele zásobníku. Nevýhodou je potřeba uložení poměrně velké množství registrů, které následně zabírá 23-24 bytů, v závislosti na velikosti instrukčního registru.[4]

Ukázka použití knihovny v příloze C.

4.2.1 Práce s kritickou sekcí

Přestože běžné programy pro mikrokontrolér neobsahují paralelně prováděné úseky kódu, jsou zde situace, které mohou způsobovat podobné „náhodné" chyby. Těmito situacemi lze označit obsluhu přerušení. Je-li přerušení povoleno, může být vyvoláno prakticky v kterémkoliv úseku kódu programu. Hlavním problémem je fakt, že pravděpodobnost vyvolání přerušení v nevhodném okamžiku je poměrně malá, takže se prakticky během testování nemusí vůbec projevit. Vedlejším problémem mohou být optimalizace překladače, který místo opakovaného čtení/zápisu uchovává proměnnou v registru. Z přerušení tak může být použita neaktuální hodnota proměnné. Vypnutí této optimalizace lze pomocí klíčového slova `volatile` v deklaraci proměnné. Bohužel i zde platí, že opomenutí klíčového slova nemusí vést k jasnému projevu chyby. Ukázka použití klíčového slova volatile v příloze D.

Řešení kritické sekce s Android2Arduino

V případě, že chceme číst či zapisovat proměnnou, která je mimo tělo programu využívána také v přerušení, využijeme k tomu třídu[5] **Android2ArduinoAtomic** či **Android2ArduinoAtomicFlag**. V případě použití první z těchto dvou tříd je kritická sekce řešena pomocí zakázání přerušení. Problémem je, že po patřičnou dobu budou přerušení ignorována. Druhá verze využívá příznaku provádění kritické sekce, ten je nutno v odpovídající přerušovací rutině přečíst, případně rutinu ukončit.

4.2.2 Instalace do Arduino IDE

Cela knihovna se skládá s několika podknihoven. Obvykle je využita základní knihovna a jedno specializované rozhraní pro práci s určitou komunikační technologií. Knihovna se nachází v jednotlivých souborech ve formátu zip. Pomocí menu `Sketch` ⇒ `Include Library` ⇒ `Add .ZIP library...` přidáme potřebné knihovny do vývojového prostředí. Poté je již možné knihovny používat standardní cestou.

Kód 4.1: Ukázková hlavička správně naimportovaných knihoven.

```
#include <Android2Arduino.h> // Obecná knihovna
#include <Android2ArduinoSerial.h> // Konkrétní rozhraní
```

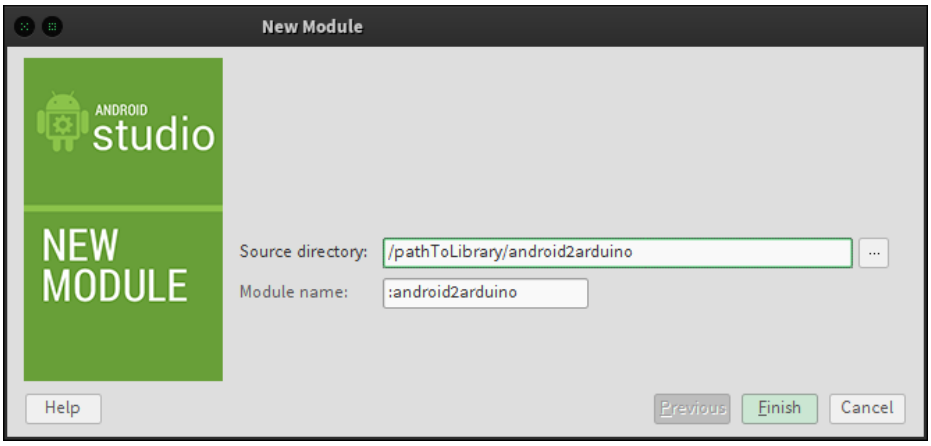
[4] avr/include/setjmp.h
[5] Jedná se o šablonu.

4.3 Implementace na Android

O reprezentaci Arduino zařízení se stará třída `GenericArduino` poskytující funkce Arduino 4.1 jako své metody. Dávku příkazů lze odeslat metodou `commit()` či zrušit metodou `rollback()`. Konstruktor očekává instanci třídy implementující rozhraní `IConnection`, která se stará o komunikaci přes konkrétní rozhraní.
Čtení a zápis probíhá v oddělených vláknech. Čtecí vlákno je blokováno čtením z odpovídajícího socketu, v případě rozpoznání příkazu zavolá dynamicky přidělenou funkci, která lze k příkazu připojit pomocí metody třídy `GenericArduino`
`attachToReader(byte key, ArduinoAction action)`.
Zápis v odděleném vlákně umožňuje postupné odesílání dat s čekáním na volný buffer bez blokace hlavního vlákna aplikace. V případě že data nelze okamžitě odeslat, jsou připojena do fronty. Délka aktuální fronty lze zjistit pomocí metody `int pending()` instance třídy `GenericArduino`. Délku fronty lze využít například v situaci, kdy chceme odesílat aktuální data senzorů, ale jsou nám poskytována rychleji než je Arduino zpracovává. Podle délky fronty můžeme některé snímky senzorů vynechat.

4.3.1 Instalace

Knihovna je distribuována ve formě modulu. Modul lze přidat do hotového projektu pomocí nabídky `File -> Import Module...`. Následně zvolíme cestu k modulu knihovny. V případě zvolení špatné cesty nás dialog upozorní chybovou hláškou. Poté co je modul při-

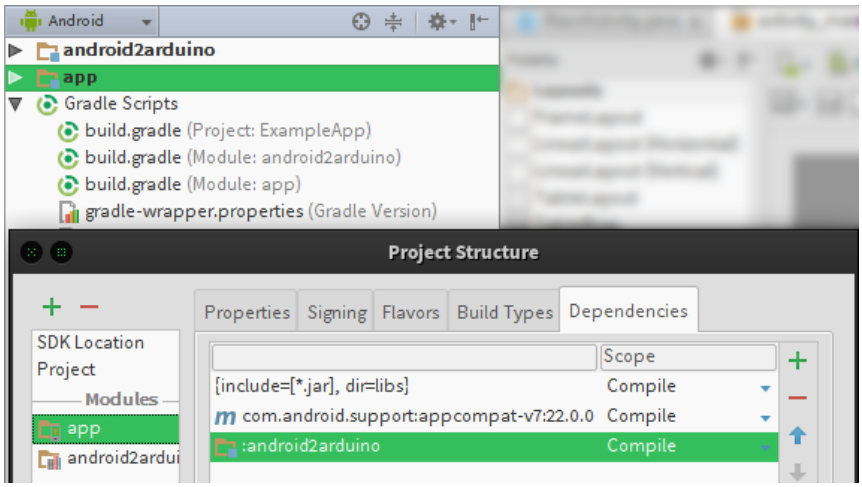


Obrázek 4.4: Přidávání modulu do projektu

dán do projektu, je třeba jej přidat do seznamu závislostí naší aplikace. To lze buď pomocí přidání příslušného řádku do `gradle` souboru aplikace, či skrze grafický dialog.

```
dependencies {
...
compile project(':android2arduino')
}
```

Grafickou cestou se tohoto dá docílit pravým klikem na modul aplikace(zde je pojmenována `app`) `Open Module Settings`. V levém seznamu zvolíme naší aplikaci a otevřeme záložku `Dependencies`. Nyní pomocí tlačítka plus přidáme závislost na modulu `android2arduino`.



Obrázek 4.5: Přidávání modulu do seznamu závislostí

Kód 4.2: Ukázka použití třídy.

```
// Vytvoření Bluetooth spojení
BluetoothConnection conn = new BluetoothConnection();
conn.connect(macAddress);

GenericArduino arduino=new GenericArduino(conn);
arduino.pinMode(Pin.analog(0), PinMode.OUTPUT);
arduino.analogWrite(Pin.analog(0),128);

//odeslání příkazů
arduino.commit();
```

4.4 Společná konfigurace

Vzdálené volání funkcí Arduino či nastavování jeho proměnných není nijak kontrolováno. Kontrola by způsobovala nejenom zvýšenou režii na mikrokontroléru, ale také režii pro programátora - např. přidávání zarážek na konce polí, extra pole pro každý typ proměnné apod.

Za účelem omezení možnosti chyby a zvýšení pohodlnosti práce, je k dispozici generátor zdrojových souborů. Vstupem tohoto generátoru je konfigurační INI soubor, jenž obsahuje seznam volatelných funkcí arduino, seznam proměnný a definici jednotlivých pinů. Výsledkem jsou 2 soubory. Prvním je třída jazyka Java v podobě souboru s koncovkou `.java` a druhým je hlavičkový soubor jazyka C++.

Třída v jazyce Java poskytuje konkrétní metody a konstanty pro práci s cílovým Arduinem, není tak třeba zadávat indexy ručně, což by mohlo být potenciálním zdrojem chyb. Je doporučeno takto vygenerovaný soubor dále neopravovat, ale využít dědičnosti či vhodného návrhového vzoru, což umožní dodatečnou úpravu vygenerované třídy bez ztrát konkrétních modifikací programátora.

Hlavičkový soubor obsahuje deklaraci proměnných a funkcí a následné umístnění jejich ukazatelů do příslušných polí. Tato pole jsou pak využita v konstruktoru třídy Android2Arduino.

Kód 4.3: Ukázka konfigurace.

```
[functions]
  blink
[analog]
  accelX = in A0
[digital]
  led = out 8
[variables]
  servo = atomic uint8_t
```

4.4.1 Popis sekcí konfiguračního souboru

Konfigurační soubor se skládá z těchto nepovinných sekcí:

- **functions** - Seznam funkcí.
- **analog/digital** U pinů lze definovat zda jsou vstupní - `in`, výstupní `out`, vstupně-výstupní `inout`. U vstupních pinů lze definovat pullup - `in pullup`. Pro všechny piny, které jsou definovány jako pouze vstupní, nebo pouze je vygenerována společná metoda pro inicializaci. Jednotlivé piny jsou pojmenovány podle konvencí Arduino - číselnou hodnotou, popř. číselnou hodnotou s prefixem `A`.
  - **analog** - Seznam pinů ke kterým je přistupováno analogově - např. čtení hodnoty A/D převodníku, nastavení PWM signálu.
  - **digital** - Seznam pinů ke kterým je přistupováno digitálně - např. čtení stavu tlačítka.
- **constants** - Seznam 1 Bytových konstant.
- **commands** - Seznam příkazů - pro vygenerování jednotného a nekolizního identifikátoru.
- **variables** - Seznam proměnných, s kterými lze pracovat. Datový typ je z množiny bool, uint8_t, int8_t, uint16_t, int16_t, int32_t, uint32_t, int64_t, uint64_t, float. V případě, že je potřeba pracovat s proměnnou, která je využívaná v přerušení 4.2.1, je třeba použít modifikátor `atomic`.

4.5 Testování knihovny

Knihovna pro Android je testována pomocí `Android testing framework`[6] založeném na JUnit. Jelikož je desktopové prostředí Javy velmi podobné tomu na Androidu, bylo by možné knihovnu otestovat například pomocí klasického JUnit, avšak Android testing framework poskytuje větší pohodlí pro obsluhu testů. Pro testování Arduino aplikací ovšem neexistuje standardizovaný framework k testování. Pro jednotkové testování přímo na čipu existuje například ArduinoUnit[7]. Tento způsob nabízí otestování reálného chování aplikace, ale neposkytuje takové možnosti ladění[8], jako ladění desktopových aplikací. V případě provádění testů na desktopu máme k dispozici libovolné nástroje, například pro krokování, čí generování statistik pokrytí kódu a podobně.

[6] Dostupný z https://code.google.com/p/googletest/.
[7] https://code.google.com/p/arduinounit/
[8] Placená varianta VisualMicro poskytuje ladění skrze sériové rozhraní.

Primárně pro účel knihovny Android2Arduino bylo vyvinuto mock[9] [9] prostředí Arduino, poskytují základní funkce, jako například `digitalWrite()` či `pinMode()`. Správné volání těchto metod lze následně otestovat pomocí odpovídajících funkcí. Program je překládán pro jinou platformu, než pro kterou je reálně určen, proto je nutné používat datové typy s explicitně uvedenou velikostí - například `int16_t` místo `int`. Mock prostředí slouží primárně k otestování funkčnosti algoritmu, pro hardwarově specifické úlohy je vhodnější využít jiné metody testování.

Software pro testování knihovny lze rozdělit na tyto části:

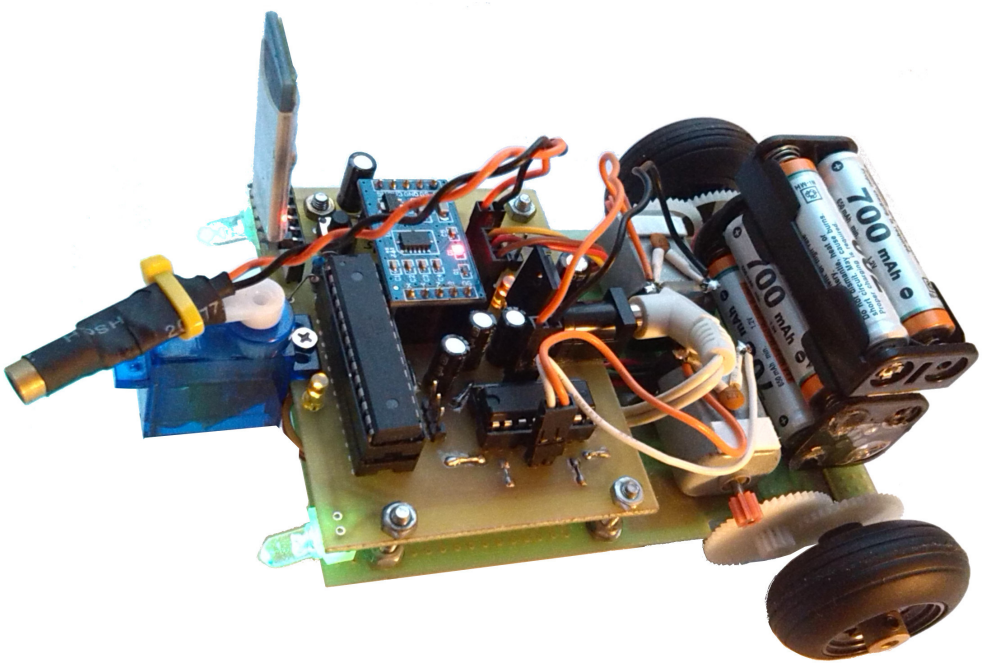
- Mock prostředí Arduino.
- Android2Arduino knihovna pro Arduino.
- Google C++ Testing Framework - poskytující jednotkové testování pro C++.
- Testy spouštějící Java rutiny, následně testují stav mock Arduina.
- JNI + Java adaptér pro volání Android knihovny. [15]
- Android2Arduino knihovna pro Android/Javu.

[9] Kontrolovaný objekt napodobující reálné prostředí.

Kapitola 5

Ukázková aplikace knihovny

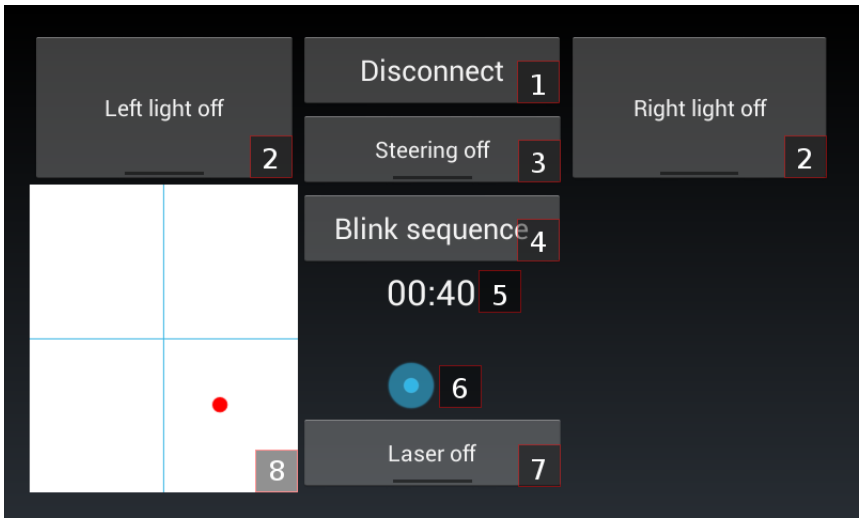
Jako jednoduchá ukázka praktického použití knihovny bylo zvoleno dálkově řízené auta. Komunikace v této aplikaci je zprostředkovávána přes Bluetooth rozhraní, které je díky své jednoduchosti a cenou ideálním kandidátem. Cílem bylo využít co nejvíce způsobu použití knihovny, i přesto, že by zvolené řešení nebylo pro reálnou aplikaci příliš vhodné.



Obrázek 5.1: Fotografie vozítka

5.1 Mobilní aplikace

Tato umožňuje komunikovat se spárovaným zařízením pomocí klasického Bluetooth. Párování zařízení tedy není řešeno přímo v aplikaci, neboť v praxi je tato činnost prováděná pouze jednou a oproti konfigurace přímo v systémovém nastavení prakticky nepřináší výhody.



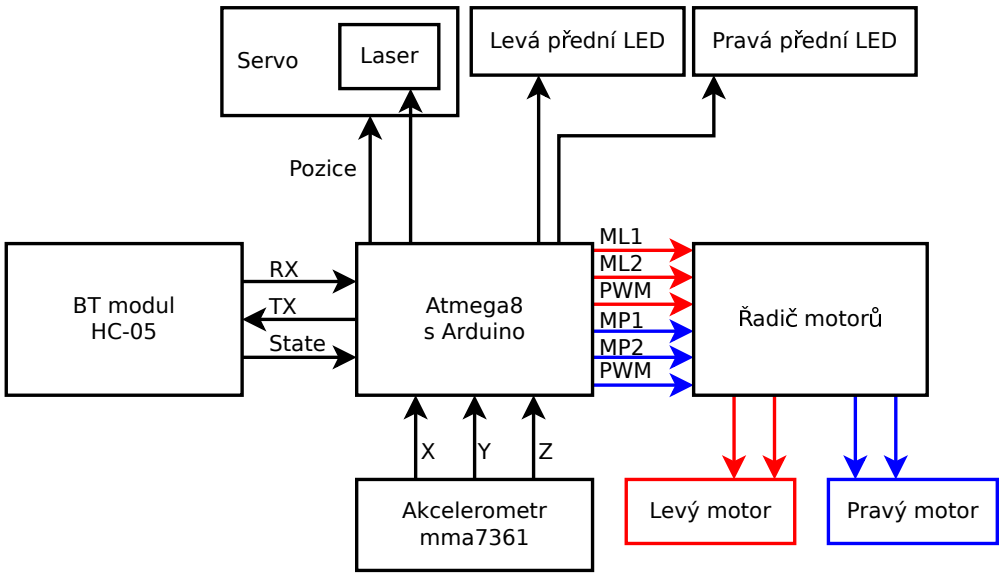
Obrázek 5.2: Ukázka řídící aplikace

Popis obrázku:

1. Tlačítko pro **připojení/odpojení** od vzdáleného zařízení. Pro připojení je poskytnut dialog se vstupem pro MAC adresu cílového zařízení.
2. Přepínací tlačítka pro ovládání **světlometů**.
3. Přepínací tlačítko pro (de)aktivaci řízení vozítka pomocí **akcelerometru** mobilního zařízení.
4. Odeslání uživatelského příkazu pro **blikací sekvenci**.
5. **Čas od spuštění**. Jeho odesílání je iniciováno Arduinem.
6. Posuvník pozice **serva**. Údaj je odeslán po uvolnění.
7. Přepínací tlačítko pro (de)aktivaci **laser**u.
8. Sledování **hodnot akcelerometru** Arduina. Sledování je iniciováno z mobilního zařízení.

5.2 Blokové schéma vozítka

Centrálním prvkem schématu schématu se nachází ATmega8, taktován interním oscilátorem na 8Mhz. Arduino software lze provozovat na poměrně velkém rozsahu mikrokontrolérů, zde vybraný mikrokontrolér byl zvolen díky nízké ceně, dostatečné flash paměti a aktuální dostupnosti. Arduino IDE má přednastavené určité konfigurace desek. Pro tuto desku bylo třeba vytvořit konfiguraci novou, neboť dostupná konfigurace pro ATmega8 je pouze pro variantu s 16Mhz, což s interním oscilátorem není možné dosáhnout. Nově zakoupený mikrokontrolér je taktován rychlostí 1Mhz, toto lze změnit pomocí nastavený tzv. pojistek



Obrázek 5.3: Blokové schéma

s využitím programu `avrdude`[1]. Jelikož volba parametrů podle datového listu není triviální, jsou k dispozici pomocné nástroje[2].

Jediným vstupním senzorem je akcelerometr MMA7361. Jedná se o integrovaný obvod na plošném spoji s podpůrnou elektronikou [29], lze jej tedy přímo zapojit do obvodu. Výstupem tohoto senzoru jsou tři analogové a jedna digitální hodnota. Digitální hodnota slouží k detekci 0g - volného pádu, zbylé analogové reprezentují jednotlivé osy. Senzor je v této aplikaci určen pouze pro měření snímání natočení vozidla - pouze gravitační zrychlení. Výsledné intervaly hodnot byly zjištěny experimentálně.

Pro řízení motorů by zvolen obvod `L293D`. Tento obvod lze využít pro řízení dvou stejnosměrných motorů. Vstupem pro každý motor jsou 2 řídící signály a jeden PWM signál.

Pro natočení laseru je využit servo motor. Jelikož s ATmega8 nelze současně řídit servo pomocí knihovních funkcí a zároveň generovat 2 PWM signály [23], je servo řízeno čistě softwarově [1]. Signál pro řízení serva se skládá z pulzů, jejichž délka koresponduje s požadovaným úhlem natočení. Perioda těchto pulzů je typicky 20ms. Delší perioda způsobuje neplynulou rotaci. Nevýhoda programového řízení serva je taková, že po dobu natáčení je program touto činnosti blokován. V případě, kdy na servo působí vnější síla, nemá tendenci udržovat požadovaný úhel natočení, jako je to u periodického řízení pomocí časovače. Pro tuto aplikaci lze tyto problémy tolerovat.

5.3 Konfigurace

Pro konfiguraci byl využit konfigurační ini soubor (viz kapitola 4.4). V sekci analogových pinů lze vidět 3 vstupní piny pro data akcelerometru a 2 výstupní pro PWM ovládání rychlosti motorů. Mezi digitálními piny lze vidět piny pro řízení motorů a spínání LED

[1] Dostupný v `arduino/hardware/tools/avr/bin/avrdude`.
[2] `http://www.engbedded.com/fusecalc/`

diod včetně laseru.

Kód 5.1: Konfigurace pinů.

```
[analog]
   accelX = in A0
   accelY = in A1
   accelZ = in A2

   motorLeftPwm = out 9
   motorRightPwm = out 10

[digital]
   led = out 8
   motorLeft1 = out 5
   motorLeft2 = out 6

   motorRight1 = out 7
   motorRight2 = out 8

   laser = out A4

   ledLeft = out A5
   ledRight = out 3
```

Řízení natočení serva je prováděno přes nastavovaní proměnné. V případě že se tato hodnota změní, mikrokontrolér zavolá rutinu pro natočení serva.

Kód 5.2: Konfigurace proměnných.

```
[variables]
   servo = uint8_t
```

Aplikace podporuje dva uživatelské příkazy. Jeden je používán pro nahrávání hodnoty z Arduino do Androidu - `uptime`. Druhým příkazem je zpracování sekvence blikání LED diod. Jelikož takový příkaz je plně v režii programátora, bylo by možné z mobilního zařízení vysílat trvalý proud dat a data postupně na mikrokontroléru zpracovávat. V této aplikaci je sekvence blikání LED řízena řetězcem znaků z množiny {'L', 'R', 'B', 'N'}. Kvůli neopakování konfigurace jsou tyto znaky zastoupeny konstantami z množiny left, right, both, none. Jednotlivé označení příkazů je zastoupeno konstantami také. Tento způsob je preferovaný, neboť zabraňuje kolizi s vestavěnými funkcemi.

Kód 5.3: Konfigurace příkazů.

```
[constants]
   left = 'L'
   right = 'R'
   both = 'B'
   none = 'N'
[commands]
 sequence
 uptime
```

5.3.1 Arduino

O hlavní činnost se stará třída `Car`, dědící z třídy `Android2ArduinoSerial` - jedná se o použití šablony `Android2Arduino` spolu s třídou pro standardní sériovou komunikaci jako parametrem. Třída `Car` přepisuje metodu `fallback(uint8_t command)` pro zpracování blikací sekvence. Dále obsahuje metodu uptime, která serializuje vstupní hodnotu a odešle mobilnímu zařízení. V praxi by tato metoda mohla odesílat řetězce, či jakákoliv jiná data, které lze zpětně deserializovat.

5.3.2 Android

Z konfiguračního souboru byla vygenerována třída `ArduinoCar` která obsahuje potřebné konstanty a metody. Metody této třídy sice manipulují s potřebnými piny, avšak jednotlivě. To je například pro řízení motorů nevhodné. Proto jsou zde naimplementovány metody pro potřebné rutiny. Vozítko je ovládáno ve dvou režimech:

- `režim stání` - jednotlivé příkazy odešlou ihned po stisku tlačítka,
- `režim jízdy` - příkazy se řadí do fronty a jsou odeslány spolu s příkazy pro řízení motorů při příchodu události od akcelerometru.

Kód 5.4: Ukázka metody třídy Car.

```
public synchronized void setLaser(boolean laser) {
   writeLaser(laser); // metoda vygenerovane tridy
   conditionCommit(); // odeslani prikazu v rezimu stani
}
```

Jak již bylo zmíněno dříve, z mobilního zařízení je možné číst analogové hodnoty z akcelerometru a také zpracovávat vlastní příkaz `uptime`. Po přečtení všech tří os je vyvolána příslušná akce pro vyobrazení na displeji.

Kód 5.5: Registrace posluchačů.

```
// Nastaveni posluchacu pro cteni analogovych hodnot akcelerometru
// pomoci vygenerovanych metod
  setAnalogAccelXListener(accelAnalogListener);
  setAnalogAccelYListener(accelAnalogListener);
  setAnalogAccelZListener(accelAnalogListener);
// Zaregistrovani posluchače pro zpracování příkazu
  attachToReader(ArduinoCar.CMD_UPTIME, uptimeAction);
```

Kapitola 6

Testování bezdrátových technologií

Jednotlivé technologie byly testovány v praxi. Cílem byla primárně zjistit dosah spotřebu těchto technologií.

6.1 Proudový odběr

Energetické náročnosti bezdrátových technologií byly měřeny na Arduino Uno, bez dalším modulů, které by mohly způsobovat nekonstantní odběr proudu. Toto Arduino je možné napájet z USB portu, či externího zdroje. Stabilizátor napětí z externího zdroje však pracuje z jistými ztrátami, které mohou být různé v závislosti na vstupním napětí či protékajícím proudu. Proto byl měřen proudový odběr přímo na napájení přes USB port. Proudové hodnoty v této kapitole tedy budou vždy souviset s napětím 5V - napájecím napětím USB.

Bylo naměřeno, že Arduino Uno má při provádění prázdné programové smyčky odběr 38mA. Proudové odběry samotných modulů tedy lze získat odečtením této hodnoty.

Proudové odběry bezdrátových modulů kolísaly, nebyla však pozorována závislost mezi proudovým odběrem a vzdáleností bezdrátových bodů.

Během testování byla zjištěna nestabilita Android2Arduino knihovny s BLE modulem, proto byl modul otestován aplikacemi třetích stran[1]. Jednalo se o aplikaci pro odesílání fiktivní hodnoty tepu srdce, takže její činnost byla naprosto různá od funkce knihovny. Z měření však lze vidět, že spotřeba modulu pro odesílání velmi malého množství dat je opravdu nízká.

Byla měřena také maximální rychlost komunikace jedním a druhým směrem ve vzdálenosti cca 50cm.

	Proudový odběr (mA)		**Přenosová rychlost (B/s)**	
	Zapojení modulu	**Komunikace**	**Z Androidu**	**Z Arduina**
HC-05	77	65 - 75	960	11 700
WiFi shield	80	150	5 800	130
BLE shield	42	42	-	-

Z tabulky je viditelné, že konkrétní implementace WiFi není vzhledem ke své rychlosti výhodná. Velmi nízká odchozí rychlost napovídá nevhodné implementaci či špatnému způsobu použití modulu. U Bluetooth modulu HC-05[2] je také vidět rozdíl rychlostí, ten však

[1] http://tinyurl.com/nrftoolbox a http://tinyurl.com/ble-heart-rate-example

[2] Pro zvýšení rychlosti byla zvýšená přenosová rychlost mezi modulem a mikrokontrolérem na 115 200 baudů z výchozích 9 600.

lze alespoň z části přisoudit nutnosti potvrzování příchozích dat knihovnou.

6.2 Měření dosahu

Měření dosahu probíhalo na místech s co nejmenším počtem překážek. Sledovaná komunikace probíhala mezi Arduino zařízením a mobilním telefonem - Android 4.1.1 (bez podpory BT LE), případně Arduino zařízením a notebookem - virtualizovaný Android 4.4.2 pro x86.

První měření probíhalo v areálu fakulty FIT VUT v Brně. Všemi výše zmiňovanými technologiemi bylo možné uskutečnit spojení z jednoho konce chodby na druhý, stejně tak pro nádvoří. Tyto vzdálenosti odpovídaly přibližně 75 metrům.

Jelikož prostory areály fakulty nebyly dostatečné, proběhlo druhé měření s mobilním telefonem na ulici Božetěchova v Brně, při kterém s použití WiFi byl ztracen signál po 120 metrech v případě Bluetooth po 135.

Další měření proběhlo na ulici Purkyňova v Brně, kde se nachází velké množství aktivních WiFi spojení.

	Notebook	**Mobilní telefon**
WiFi shield	160m	107m
HC-05	94m	85m
BLE shield	60m	-

Dosah Bluetooth byl poměrně překvapivý, neboť pro `class 2` je udávaný dosah kolem 10 metrů.

Během měření byla komunikace často pozastavena výpadkem signálu. S rostoucí vzdáleností byly tyto výpadky stále častější a bylo třeba „vyhledávat" vhodnou pozici pro komunikaci. Obecně lze tedy říct, že měřené technologie mají poměrně velký dosah, ale pro praktické použití je třeba zvolit vzdálenost mnohem kratší.

6.3 Možnosti optimalizace

Všechny měřené Arduino moduly používaly integrovanou anténu na plošném spoji, v případě potřeby komunikace na dlouhé vzdálenosti je vhodné použít směrovou anténu. Bohužel ne všechny varianty modulů poskytují nedestruktivní možnost připojení externí antény.

Proudovou náročnost aplikace lze omezit například snížením spotřeby mikrokontroléru. Podle datového listu [22, str. 589] by bylo možné jen snížením výchozí taktovací frekvence[3] hlavního mikrokontroléru Arduino Uno dosáhnout snížení odběru až o 9mA. Dále například modul HC-05 poskytuje na svém rozhraní pin `State`, který by mohl probouzet uspaný mikrokontrolér pomocí externího přerušení při připojení zařízení.

Další možností jak snížit spotřebu aplikace je vypínání bezdrátového rozhraní. To lze u HC-05 snadno pomocí pinu `enable`, u kompletních shieldů však standardní cesta nemusí existovat. Vypínání modulů však nelze provádět automaticky, neboť pro efektivní využití je třeba znalost chování konkrétní aplikace.

[3] Na Arduino Uno je výchozí frekvence 16Mhz.

Kapitola 7

Závěr

V rámci této práce byla vytvořena knihovna v Javě pro Android a v C++ pro Arduino. Pro tuto knihovnu byly napsány moduly pro komunikaci přes Wifi (TCP), klasický Bluetooth a Bluetooth Low Energy.

Pro praktické použití Wifi si lze představit zařízení Arduino připojené k trvalému přístupovému bodu a příležitostné připojování pomocí mobilního zařízení. Bluetooth Classic si lze představit v aplikacích, ve kterých a není třeba další interakce například s domácí sítí. Mezi tyto aplikace lze zařadit například řízení modelů aut. Funkce knihovny přes Bluetooth Low Energy je vhodná spíše pro velmi specifické aplikace. Například použití tohoto řešení pro řízení vozítka (kapitola 5) by bylo možné, avšak by se vůbec nevyužil potenciál technologie. Kvůli potřeby přenášet velké množství a nižšího rozšíření by bylo vhodné použít klasické Bluetooth. Další otázkou je míra snížení energetické náročnosti oproti běžnému Arduinu - 16Mhz takt mikrokontroléru, spotřeba LED diod a dalších komponent. Během testování byla zjištěna nestabilita knihovny s touto technologií. Problém lze přisuzovat softwarovému modulu knihovny pro obsluhu této technologie.

Pro potřeby knihovny bylo napsáno testovací prostředí, které zjednodušilo testování softwarové části Arduino, především v interakci s knihovnou.

Jako praktické použití bylo vytvořena elektronika jednoduchého vozítka řízeného pomocí klasického Bluetooth ze specializované aplikace na mobilním zařízení. Tato aplikace se snaží využít co největší část knihovny. Jako ukázka použití ostatních zmíněných technologií slouží univerzální aplikace, sloužící k ovládání pinů Arduina.

7.1 Možné použití

Jádro knihovny v Javě lze bez modifikace provozovat na stolním počítači. Bylo by tak možné vytvořit aplikaci vhodnou jako pomůcku pro rychlé ověřování činnosti pomocných hardwarových obvodů - spínání motorů, čtení stavů, generátor PWM a podobně. Vzniklo by tak univerzální zařízení bez nutnosti přeprogramování pro uzpůsobení činnosti prototypu. Jako příklad by mohl sloužit akcelerometr použitý na vozítku. V případě potřeby otestování přesnosti, rozsahu hodnot či celkového by jej stačilo připojit do univerzálního obvodu. Předpřipravená aplikace by nastavila potřebné hodnoty pinů - například pin pro volbu přesnosti. Vzorkované hodnoty akcelerometru by byly zobrazeny například v grafu.

Většina aplikací využívající knihovnu slouží spíše ke vzdálenému ovládání. Lze se však na použití podívat z opačného pohledu. Spojením mobilního telefonu s Arduinem můžeme získat univerzální zařízení poskytující jak interakci s elektronikou, tak pohodlné programo-

vání na výkonném procesoru mobilního telefonu. Jelikož jsou mobilní telefony, na rozdíl od specializovaných modulů, vyráběny velkých sériích, je možné je získat za mnohem nižší cenu. Mobilní telefon také poskytne jednoduchý přístup k periferiím, jako je například kamera, GPS a podobně.

Literatura

[1] MARGOLIS, Michael. *Arduino cookbook*. 1st ed. Sebastopol: O'Reilly, 2011, xxi, 631 s. ISBN 978-059-6802-479.

[2] GANGULI, Madhushree. *Getting started with Bluetooth*. Cincinnati, Ohio: Premier Press, c2002, xviii, 389 s. ISBN 19-318-4183-7.

[3] LABIOD, H, H AFIFI a C DE SANTIS. *Wi-Fi, Bluetooth, ZigBee and WiMAX*. Dordrecht: Springer, c2007, xvi, 316 s. ISBN 978-1-4020-5396-2.

[4] TOWNSEND, Kevin, Robert DAVIDSON, AKIBA. a Carles CUFI. *Getting started with Bluetooth low energy: tools and techniques for low-power networking*. Revised First Edition. xii, 164 pages. ISBN 978-149-1949-511.

[5] *Bluetooth low energy* [online]. CSR. Dostupné z: https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=227336

[6] Manifest.permission. *Android developers* [online]. [cit. 2014-10-16]. Dostupné z: http://developer.android.com/reference/android/Manifest.permission.html

[7] Sensors Overview. *Android developers* [online]. [cit. 2015-05-04]. Dostupné z: http://developer.android.com/guide/topics/sensors/sensors_overview.html

[8] Smartphone OS Market Share, Q4 2014. *IDC* [online]. [cit. 2015-05-04]. Dostupné z: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[9] Unit testing. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-05-04]. Dostupné z: http://cs.wikipedia.org/w/index.php?title=Unit_testing

[10] ArduinoBoardUno. *Arduino* [online]. 2015 [cit. 2015-05-04]. Dostupné z: http://www.arduino.cc/en/Main/ArduinoBoardUno

[11] BluetoothManager. In: *Android Developers* [online]. [cit. 2014-10-04]. Dostupné z: http://developer.android.com/reference/android/bluetooth/BluetoothManager.html

[12] Android Bluetooth Tutorial. *Tutorials point* [online]. [cit. 2014-10-16]. Dostupné z: http://www.tutorialspoint.com/android/android_bluetooth.htm

[13] Bluetooth: DiscoveringDevices. *Android developers* [online]. [cit. 2014-10-16]. Dostupné z: http://developer.android.com/guide/topics/connectivity/bluetooth.html#DiscoveringDevices

[14] WifiManager. *Android developers* [online]. [cit. 2014-10-20]. Dostupné z: http://developer.android.com/reference/android/net/wifi/WifiManager.html

[15] Creating a JVM from a C Program. *InOnIt.com* [online]. [cit. 2015-02-23]. Dostupné z: http://www.inonit.com/cygwin/jni/invocationApi/c.html

[16] ArduinoWiFiShield. *Arduino* [online]. [cit. 2014-10-22]. Dostupné z: http://arduino.cc/en/Guide/ArduinoWiFiShield

[17] Arduino GSM Shield. *Arduino* [online]. [cit. 2015-05-04]. Dostupné z: http://www.arduino.cc/en/Main/ArduinoGSMShield

[18] GSM. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-05-04]. Dostupné z: http://en.wikipedia.org/wiki/GSM

[19] 2G. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-05-04]. Dostupné z: http://en.wikipedia.org/wiki/2G

[20] *Quectel M10 datasheet*. [cit. 2015-05-04] Dostupné z: http://www.arduino.cc/en/uploads/Main/Quectel_M10_datasheet.pdf

[21] *ATmega8 datasheet*. [cit. 2015-05-04] Dostupné z: http://www.atmel.com/Images/Atmel-2486-8-bit-AVR-microcontroller-ATmega8_L_datasheet.pdf

[22] ATMEL CORPORATION. *Atmel 8-bit microcontroller with 4/8/16/32kbytes in-system programmable flash*. 2014. Dostupné z: http://tinyurl.com/atmega-datasheet

[23] Servo library. *Arduino* [online]. [cit. 2015-05-04]. Dostupné z: http://www.arduino.cc/en/Reference/Servo

[24] *Bluetooth wireless technology basics* [online]. Hewlett-Packard Development Company, 2004. Dostupné z: http://www.hp.com/ctg/Manual/c00186949.pdf

[25] Google I/O 2013 - Best Practices for Bluetooth Development. In: Youtube [online]. 16. 5. 2013 [vid. 2014-11-09]. Kanál uživatele Google Developers. Dostupné z: https://www.youtube.com/watch?v=EC5-cEbr520

[26] Dashboards. *Android Developers* [online]. [cit. 2015-05-06]. Dostupné z: http://developer.android.com/about/dashboards/index.html

[27] Designing for Backwards Compatibility. In: *ANDROID DESIGN PATTERNS* [online]. [cit. 2015-04-08]. Dostupné z: http://www.androiddesignpatterns.com/2012/06/designing-for-backwards-compatibility.html

[28] Hayes command set. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-02-13]. Dostupné z: http://en.wikipedia.org/wiki/Hayes_command_set

[29] APEX ELECTRIX, LLC. 2013. *MMA7361 3-Axis Accelerometer Module* [online]. [cit. 2015-04-20]. Dostupné také z: http://www.apexelectrix.com/PDFs/MMA7361/MMA7361_module_datasheet.pdf

Příloha A

Obsah CD

Soubory na CD jsou ve stromové struktuře, každý podstrom obsahuje `README` soubor s popisem.

- **Android** - Projekt pro Android Studio, obsahující zdrojové soubory pro Android.
  - **android2arduino** - Knihovna pro komunikaci.
  - **car** - Aplikace pro řízení vozítka.
  - **preview** - Aplikace pro ukázku použití všech technologií.
- **Arduino**
  - **aplikace** - Ukázkové aplikace.
  - **knihovny** - Hlavní knihovna android2arduino spolu rozhraními pro potřebná komunikační rozhraní.
  - **ukazky_optimalizace** - Ukázky zdrojových kódů spolu s popisem a přeloženou variantou v jazyce symbolických instrukcí.
- **ClassCreator** - Generátor společných konfiguračních souborů.
- **SpolecneTesty** - Testy pro otestování kooperace Java a C++ kódu knihoven.
- **Schema** - Schéma vozítka.
- **Text** - Text práce.

Příloha B

Popis Android api

B.1 Wifi

Kód B.1: Seznam potřebných oprávnění.

```
<!-- Povolení přístupu k informacím o Wi-Fi sítích -->
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE
    " />

<!-- Povolení změn připojení Wi-Fi sítí -->
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE
    " />
```

Kód B.2: Zapnutí Wi-Fi adaptéru.

```
public void enable(){
  manager.setWifiEnabled(true);
}
```

Kód B.3: Vypnutí Wi-Fi adaptéru.

```
public void disable(){
   manager.setWifiEnabled(false);
}
```

Kód B.4: Zjištění síly signálu v procentech.

```
public int getSignal() {
  WifiInfo info = manager.getConnectionInfo();
  if (info == null)
  return -1;
  return WifiManager.calculateSignalLevel(info.getRssi(), 101);
}
```

Kód B.5: Zjištění rychlosti spojení.

```
public String getLinkSpeed(){
  WifiInfo info = manager.getConnectionInfo();
  if (info == null)
  return null;
```

```
    return String.format("%d%s",info.getLinkSpeed(),WifiInfo.
        LINK_SPEED_UNITS);
  }
```

Kód B.6: Připojení k nakonfigurované síti.

```
  public boolean connectToConfigured(String SSID) {
    for (WifiConfiguration res : manager.getConfiguredNetworks()) {
      if (SSID == res.SSID) {
        manager.enableNetwork(res.networkId,true);
        return true;
      }
    }
    return false;
  }
```

Kód B.7: Vyhledání Wi-Fi sítí.

```
  List<ScanResult> results = null; // zde bude uložen výsledek

  // Vytvoření broadcast receiveru
  final BroadcastReceiver wifiScanReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
      results = manager.getScanResults();
      ctx.unregisterReceiver(this);
    }
  };

  // zaregistrování receiveru a spuštěno scanování
  public void scan(){
    ctx.registerReceiver(wifiScanReceiver,new IntentFilter(WifiManager
        .SCAN_RESULTS_AVAILABLE_ACTION));
    manager.startScan();
  }
```

B.2 Bluetooth

Kód B.8: Seznam potřebných oprávnění.

```
  <!-- Povolení přístupu ke spárovaným zařízením -->
  <uses-permission android:name="android.permission.BLUETOOTH" />

  <!-- Povolení přístupu k vyhledávání a párování zařízení -->
  <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"
      />
```

Kód B.9: Žadost uživatele o zapnutí modulu.

```
  public void requestEnable() {
    Intent turnOn = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE)
        ;
    activity.startActivityForResult(turnOn, BT_ENABLE_REQUEST);
  }
```

Kód B.10: Zapnutí adaptéru.

```
public void enable() {
  adapter.enable();
}
```

Kód B.11: Vypnutí adaptéru.

```
public void disable() {
  switch (adapter.getState()){
    case BluetoothAdapter.STATE_TURNING_OFF:
    case BluetoothAdapter.STATE_OFF:
    return;
    default:
    adapter.disable();
  }
}
```

Kód B.12: Vyžádat umožnění zařízení objevitelnost.

```
public void makeDiscoverable() {
  Intent intent = new Intent(
  BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE
  );
  activity.startActivityForResult(intent, BT_DISCOVERABLE_REQUEST);
}
```

Kód B.13: Připojení klienta k serveru.

```
public BluetoothSocket connectUUID(String MAC, String UUIDString)
    throws IOException {
  BluetoothDevice device = adapter.getRemoteDevice(MAC);
  UUID uuid = UUID.fromString(UUIDString);
  BluetoothSocket socket = device.createRfcommSocketToServiceRecord(
     uuid);
  socket.connect();
  return socket;
}
```

Kód B.14: Spuštění objevování zařízení.

```
// Vytvoření Broadcast receiveru
private final BroadcastReceiver mReceiver = new BroadcastReceiver()
    {
  public void onReceive(Context context, Intent intent) {
    String action = intent.getAction();
    if (BluetoothDevice.ACTION_FOUND.equals(action)) {
      BluetoothDevice device = intent.getParcelableExtra(
          BluetoothDevice.EXTRA_DEVICE);
      devices.add(device); // Přidání nalezeného zařízení do seznamu
    }
  }
};

public void startDiscovering() {
```

```
    IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND
        );
    // Registrace broadcast receiveru
    activity.registerReceiver(mReceiver, filter);
    adapter.startDiscovery();
    devices.clear();
  }
```

Některé akce je možné provést jak programově, tak s pomocí vyžádání akce uživatele - toto je realizováno pomocí intentu[1].

[1] "Abstraktní popis operace k vykonání. http://developer.android.com/reference/android/content/Intent.html

Příloha C

Ukázka použití knihovny

Na níže zobrazeném příkladu je ukázka použití knihovny. Z mobilního telefonu bude možné kromě základních funkcí měnit hodnotu proměnné `ledVal` či volat funkci `switchLed();`

Kód C.1: Ukázka použití knihovny.

```
#include <Android2Arduino.h>
#include <Android2ArduinoSerial.h>
using namespace a2alib;

#define LED (13)

void switchLed() {
  static bool state = false;
  digitalWrite(LED, state);
  state = !state;
}

bool ledVal = true;    // vzdalene modifikovatelna promenna

varPtr vars[] = {{.BOOL = &ledVal}}; // pole ukazatelu

callback cbs[] = {switchLed}; // senzam funkci

// Android2ArduinoSerial = Android2Arduino<A2ASerial>
Android2ArduinoSerial a2a(vars, cbs);

A2ASerial* serial;

void setup() {
  pinMode(LED, OUTPUT);
  serial = a2a.getComm();
  serial->begin(9600);
}

void loop() {
  if (serial->available()) {
    a2a.handleCommunication();
  }
  // kod programu
}
```

Příloha D

Ukázka použití klíčového slova volatile

Následující příklad demonstruje použití/ opomenutí klíčového slova `volatile`. Funkce `waitForButton` čeká, dokud nebude stisknuto tlačítko. Předpokládejme, že proměnná `pressed` bude nastavena na `true` z obsluhy externího přerušení. Pro překlad bude využit překladač `avr-gcc`[1]. Použití: avr-gcc -O2 ./test.c -S Po úspěšném překladu lze v souboru ./test.s vidět vygenerovaný assembler. Instrukční sadu lze najít v odpovídajícím datasheetu.

Kód D.1: Ukázka použití volatile.

```
bool pressed = false; // nespravna varianta
volatile bool pressed = false; // spravna varianta

void waitForButton(){
  while(!pressed) {
    ; //wait
  }
}
```

Kód D.2: Úsek vygenerovaných instrukce bez použití volatile.

```
  lds r24,pressed   // načtení proměnné do registru r24
  cpse r24,__zero_reg__ // v případě, že je registr roven 0 - přeskoč
     následující instrukci
  rjmp .L5    // skoč na L5 - ukonči funkci
.L4:
  rjmp .L4    // skoč na L4 - nekonečný cyklus.
.L5:
  ret
```

Proměnná `pressed` se tedy vyhodnotila pouze jednou.

Kód D.3: Úsek vygenerovaných instrukcí s použitím volatile.

```
.L2:
  lds r24,pressed   // načení proměnné do registru
  tst r24     // otestuje, zda je promenna 0 nebo záporná
  breq .L2    // pokud je 0 - skoč na L2 - cyklus
  ret       // jinak ukonči funkci
```

[1] Verze 4.9.2 20141224 (prerelease)

Zde již lze vidět, že proměnná je každým průchodem opakovaně čtená, což bylo našim záměrem.

V případě použití proměnných delší, než jedna slabika[2] pouze použití klíčového slova`volatile` nestačí.

Kód D.4: Použití víceslabikové proměnné.

```
volatile uint16_t a = 0;
a = 513;
```

Kód D.5: Přírazení dvouslabikové hodnoty.

```
// Inicializace
1. std Y+2,__zero_reg__
2. std Y+1,__zero_reg__
// načtení hodnoty do registrů
3.. ldi r24,lo8(1) // 1
4. ldi r25,lo8(2) // 256 * 2 = 512
// uložení registrů do proměnné
4. std Y+2,r25
5. std Y+1,r24
```

V případě, že by přerušení přišlo například mezi 4. a 5. řádkem, místo konzistentních hodnot 0 či 513 by byla přečtena hodnota 1.

[2]Na jiných platformách se může jednat i jinou délku

Příloha E

Popis Arduino API

Kód E.1: Přeposílání dat mezi BT a PC.

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(9, 8); // RX, TX
void setup()
{

  Serial.begin(57600);
  while (!Serial) ;
  Serial.println("Start");
  mySerial.begin(9600);
  mySerial.listen();
}

void loop()
{
  if (mySerial.available())
  Serial.write(mySerial.read());
  if (Serial.available())
  mySerial.write(Serial.read());
}
```

Kód E.2: Získání seznamu Wi-Fi sítí.

```
#include <SPI.h>
#include <WiFi.h>

void setup() {
  Serial.begin(9600); // initialize serial and wait for the port to
      open:
  while(!Serial) ;
}

void loop() {
  listNetworks();
}

void listNetworks() {
  byte count = WiFi.scanNetworks();
```

```
    for (int index = 0; index<numSsid; index++) {
      Serial.println(WiFi.SSID(thisNet));
    }
  }
```

Kód E.3: Připojení k Wi-Fi síti.

```
  #include <WiFi.h>
  char* ssid = "MOJE_SSID"; // SSID požadované sítě
  int status = WL_IDLE_STATUS; // stav Wi-Fi

  void setup() {
    Serial.begin(9600);

    // Pokus o připojení
    status = WiFi.begin(ssid);


    if ( status != WL_CONNECTED) {
      // Obsluha chyby
      while(true);
    } else {
    // Obsluha připojení
    }
  }
```

Kód E.4: Připojení k síti s WPA/WEP zabezpečením.

```
  #include <WiFi.h>

  char* ssid = "MOJE_SSID";
  char* pass = "ABBADEAF01";
  int status = WL_IDLE_STATUS;

  void setup() {
    Serial.begin(9600);
    // Pokus o připojení WPA
    status = WiFi.begin(ssid, pass);

    /*
    // Pokus o připojení WEP
    status = WiFi.begin(ssid, 0, pass);
    */

    if ( status != WL_CONNECTED) {
      // Obsluha chyby
      while(true);
    }

    else {
      // Obsluha připojení
    }
  }
```

Příloha F

Ukázka části výpisu sdptool

```
Service Name: Voice Gateway
Service RecHandle: 0x10007
Service Class ID List:
"Handsfree Audio Gateway" (0x111f)
"Generic Audio" (0x1203)
Protocol Descriptor List:
"L2CAP" (0x0100)
"RFCOMM" (0x0003)
Channel: 10
Profile Descriptor List:
"Handsfree" (0x111e)
Version: 0x0105

Service Name: OBEX Phonebook Access Server
Service RecHandle: 0x10008
Service Class ID List:
"Phonebook Access - PSE" (0x112f)
Protocol Descriptor List:
"L2CAP" (0x0100)
"RFCOMM" (0x0003)
Channel: 19
"OBEX" (0x0008)
Profile Descriptor List:
"Phonebook Access" (0x1130)
Version: 0x0100

Service Name: RFCOMM_SERIAL
Service RecHandle: 0x10009
Service Class ID List:
UUID 128: 00001101-0000-1000-8000-00805f9b34fb
Protocol Descriptor List:
"L2CAP" (0x0100)
"RFCOMM" (0x0003)
Channel: 26
```

Část výpisu programu sdptool[1]

[1] http://linux.die.net/man/1/sdptool

Příloha G

Ukázka konfigurace pomocí AT příkazů

AT+NAME
+NAME:BC_BT_ZBYNEK
OK

AT+NAME=BC_BT_MODULE
OK

AT+NAME?
+NAME:BC_BT_MODULE
OK

AT+PSWD?
+PSWD:1234
OK

AT+version
+VERSION:2.0-20100601
OK

Seznam podporovaných příkazů lze najít například na
http://www.linotux.ch/arduino/HC-0305_serial_module_AT_commamd_set_201104_revised.pdf