Ekonomická fakulta
Faculty of Economics

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Fakulta ekonomická
Katedra aplikované matematiky a informatiky

Bakalářská práce

Vývoj aplikace pro jednodeskový počítač

Vypracoval: Ondřej Duda
Vedoucí práce: Mgr. Radim Remeš

České Budějovice 2016/2017

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta ekonomická
Akademický rok: **2015/2016**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Ondřej DUDA**
Osobní číslo:	**E14368**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Ekonomická informatika**
Název tématu:	**Vývoj aplikace pro jednodeskový počítač**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování:

Cílem práce je sestavení a naprogramování modelu s použitím jednodeskového počítače.

Metodický postup:

1. Studium odborné literatury.
2. Publikace výsledků rešerše.
3. Návrh, popis vývoje a implementace aplikace.
4. Zhodnocení, vypracování doporučení a závěrů.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **40 - 50 stran**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. **McRoberts, M. (2013).** *Beginning Arduino.* **2. vydání. New York, USA: Apress.**
2. **Monk, S. (2011).** *Programming Arduino: Getting Started with Sketches.* **Columbus, OH, USA: McGraw-Hill.**
3. **Monk, S. (2016).** *Make: Action: Movement, Light, and Sound with Arduino and Raspberry Pi.* **San Francisco, CA, USA: Maker.**
4. **Troelsen, A., & Japikse, P. (2015).** *C# 6.0 and the .NET 4.6 Framework.* **7. vydání. New York, USA: Apress.**
5. **Vobecký, J., & Záhlava, V. (2005).** *Elektronika: Součástky a obvody, principy a příklady.* **3., rozš. vyd. Praha: Grada.**

Vedoucí bakalářské práce: **Mgr. Radim Remeš**
Katedra aplikované matematiky a informatiky

Datum zadání bakalářské práce: **15. ledna 2016**

Termín odevzdání bakalářské práce: **14. dubna 2017**

doc. Ing. Ladislav Rolínek, Ph.D.
děkan

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
Studentské 13 (26)
370 05 České Budějovice

L.S.

prof. RNDr. Pavel Tlustý, CSc.
vedoucí katedry

V Českých Budějovicích dne 31. března 2016

Prohlášení

Prohlašuji, že svoji diplomovou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47 zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to - v nezkrácené podobě vzniklé vypuštěním vyznačených částí archivovaných Ekonomickou fakultou - elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

Datum

Podpis studenta

(v písemné verzi vlastnoruční podpis ve všech kopiích!)

Poděkování

Rád bych poděkoval vedoucímu své bakalářské práce, panu Mgr. Radimu Remešovi, za všechny rady a připomínky při tvorbě bakalářské práce. Dále bych pak chtěl poděkovat rodině za podporu, které se mi dostává při studiu na vysoké škole.

Obsah

1 Úvod

Jako téma mé bakalářské práce jsem si vybral Vývoj aplikace pro jednodeskový počítač. Konkrétně jde o vývojovou platformu Arduino Uno. Jedná se o velmi rozšířenou desku, která je oblíbená pro svou jednoduchost.

Díky velkému množství přídavných součástek, kterými mohou být různé senzory nebo výstupní zařízení, jako jsou LCD displeje, motory a mnoho dalších, má Arduino nepředstavitelné množství využití.

Rozhodl jsem se vytvořit vlastní vozidlo na dálkové ovládání. Srdcem celého vozidla bude samozřejmě Arduino Uno, avšak další důležitou součástí bude můj mobilní telefon, pro který taktéž navrhnu vlastní aplikaci, pomocí které budu ovládat Arduino.

V první části bakalářské práce se budu věnovat Arduinu, technologii Bluetooth a operačnímu systému Windows Phone, na kterém poběží řídící aplikace. Ve druhé části práce se zaměřím na konstrukci robota a součásti využité při jeho výstavbě. Následně se pak budu věnovat samotnému programu pro Arduino a řídící aplikaci.

1.1 Cíl práce

Cílem práce je vývoj aplikace pro jednodeskový počítač, konkrétně se jedná o jednodeskový počítač Arduino Uno. Hodlám sestavit pásové vozidlo, které bude řízeno právě Arduinem. Avšak aby Arduino vědělo, jak má vozidlo řídit, budu ho navigovat pomocí vlastní aplikace pro Windows Phone 8.1, běžící na mém mobilním telefonu Nokia Lumia 520. Komunikace mezi Arduinem a telefonním zařízením bude zprostředkována pomocí technologie Bluetooth. Lumia 520 má Bluetooth modul zabudovaný ve své základové desce, avšak o Arduinu se to již říci nedá. Proto budu muset k Arduinu připojit Bluetooth modul HC-06, který tuto komunikaci zajistí.

Aplikaci pro Arudino budu vyvíjet v integrovaném vývojovém prostředí Arduino IDE, které dodává společnost Arduino. Aplikaci pro Windows Phone pak budu vyvíjet ve Visual Studio od firmy Microsoft, konkrétně se bude jednat o verzi Microsoft Visual Studio 2015 Comunity. Jelikož vývojové prostředí Visual Studio neumožňuje vývoj

aplikací pro Windows Phone už od základní verze, ještě doinstaluji doplněk Windows Phone SDK, který slouží právě k vývoji mobilních aplikací. Bude se jednat o takzvanou blank aplikaci. To je jednostránková aplikace, kde není nic přednastaveno. Uživatelské rozhraní se bude vyvíjet ve značkovacím jazyce XAML a aplikační rozhraní pak navrhnu v objektovém jazyce Visual C#. Oba tyto jazyky byly vyvinuty firmou Microsoft.

2 Vývojová platforma Arduino

Arduino je open-source elektronická platforma, kdy jediné, co je chráněno, je označení Arduino. Ačkoli si většina lidí představí Arduino jen jako malý modrý počítač, tak pravdu mají jen z poloviny. Arduino desky tvoří jednu polovinu celku, druhou významnou polovinou je software, pomocí kterého můžeme tyto desky ovládat.

Arduino původně vzniklo jako pomoc studentům technických oborů. Autoři mikrokontroléru ho pojmenovali po baru „Bar Di Re Arduino“ v severoitalském městě Ivera, kde Arduino vznikalo. Později se stalo komerčním produktem, za jehož úspěchem stojí právě open-source technologie, snadné použití a vysoká odolnost. (Monk, 2012)

2.1 Arduino desky

Arduino není jen jedna jediná deska, ale několik různých desek, které se liší v různých parametrech, jako je výkon, velikost, způsob propojení s počítačem a další. Základem všech desek Arduino je Atmel 8, 16 nebo 32 bitový AVR mikrokontrolér. Další společnou vlastností pro desky Arduino je vybavenost dalšími podpůrnými obvody. Například obvody, které slouží pro řízení sériové komunikace, současné desky pro tento účel využívají mikročip ATmega16U2. Tyto obvody pak zajišťují komunikaci s počítačem pomocí emulace virtuální COM portu přes USB. Desky dále obsahují jednořadé piny nebo female headers, které usnadňují připojení dalších součástí elektrického okruhu nebo dokonce dalších vnitřních okruhů k desce, jako jsou například diody, motůrky, LCD displeje nebo takzvané „shieldy“, což jsou malé obvodové desky, které obsahují zařízení jako GPS lokátor nebo Ethernet modul. Většina desek dále obsahuje pětivoltový lineární regulátor, který zajišťuje stabilní napětí, a šestnácti megahertzový krystalický oscilátor nebo keramický rezonátor. (Monk, 2012)

2.1.1 Oficiální Arduino desky

Oficiálních desek vyráběných pod označením Arduino je mnoho. V tabulce 1 jsou vybrány základní modely těchto desek.

Tabulka 1 Technická specifikace Arduino desek

Název	Mikrokontrolér	Počet digitálních pinů	Počet analogový pinů	Operační napětí	Velikost délka/šířka
Arduino Uno	ATmega328P	14	6	5 V	68.6/53,4 mm
Arduino Mega	ATmega2560	54	16	5 V	101,52/53,3 mm
Arduino Nano	ATmega328	22	8	5 V	45/18 mm
Arduino Lilypad	ATmega168 nebo ATmega328V	14	6	2,7-5.5 V	50/50 mm
Arduino Leonardo	ATmega32U4	20	12	5 V	68.6/53,4 mm
Arduino Zero	ATSAMD21G18	20	7	3,3 V	68/30 mm

Zdroj: Arduino - ArduinoBoardUno, © 2017, Arduino - ArduinoBoardMega, © 2017, Arduino - ArduinoBoardNano, © 2017, Arduino - ArduinoBoardLilypad, © 2017, Arduino - ArduinoBoardLeonardo, © 2017, Arduino - ArduinoBoardZero, © 2017

2.1.2 Klony

Vzhledem k tomu, že Arduino je založeno na open-source, vzniklo mnoho neoficiálních klonů Arduina. Tyto klony bychom mohli rozdělit do dvou skupin. První skupinou jsou ty klony, které vezmou dokumentaci Arduina a podle ní vytvoří levnější verzi (např. Freeduino nebo Roboduino). Další skupinou klonů jsou pak ty, které původní desku upraví, ať už například zmenší (Teensy, Femtoduino) nebo nahradí procesor za výkonnější (Chipkit). (Monk, 2012)

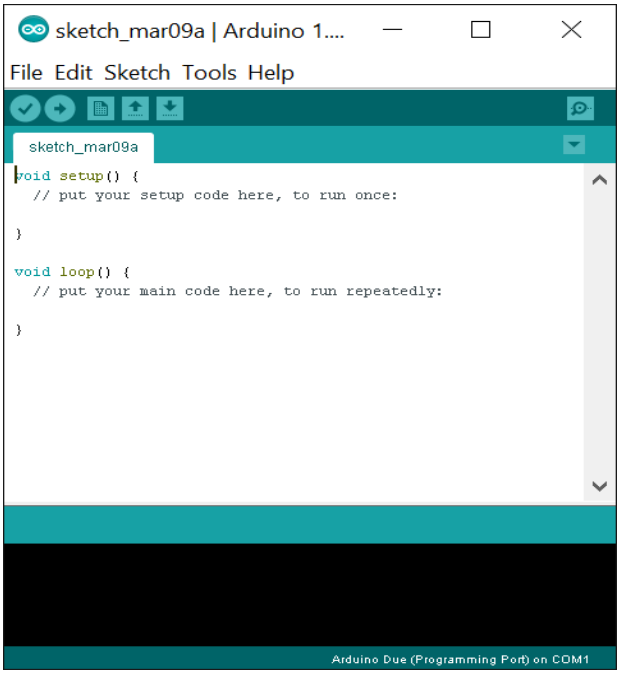
2.2 Programové vybavení

Aby vývojové desky mohly správně fungovat, potřebují k tomu určitý kód. Jelikož Arduino obsahuje procesor, a jako každý procesor i ten v Arduinu rozumí pouze binárnímu kódu, je možné napsat tyto kódy v jakémkoli programovacím jazyce, který může být přeložen právě do binárního kódu. Arduino k tomuto účelu poskytuje integrované vývojové prostředí Arduino IDE. (Monk, 2012)

2.2.1 Vývojové prostředí Arduino IDE

Arduino IDE je multiplatformní vývojové prostředí, které bylo napsáno v programovacím jazyce Java. Je odvozeno z vývojového prostředí pro jazyky Wiring a Processing. Kódům pro Arduino napsaným v tomto vývojovém prostředí se říká sketche a několik základních sketchů obsahuje Arduino IDE ihned po nainstalování. Každý sketch pak vždy musí obsahovat metodu setup a metodu loop. Metoda setup slouží k inicializaci proměnných, nastavení pinů či načtení knihoven. Tato metoda se provede vždy jen jednou, a to po zapnutí Arduina. Naopak metoda loop běží pořád dokola, dokud nebude Arduino restartováno nebo odpojeno od zdroje elektřiny. (Monk, 2012)

Obrázek 1 Ukázka vývojového prostředí Arduino IDE



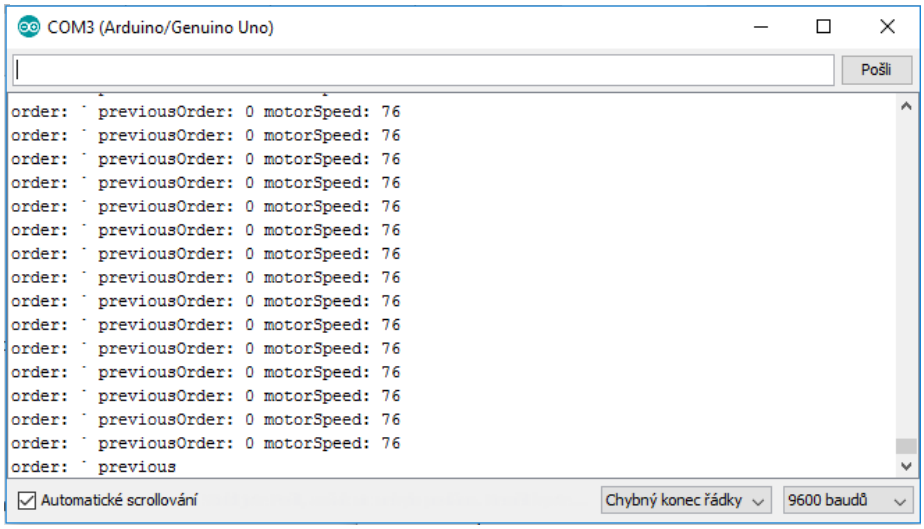
Zdroj: Vlastní

Na obrázku 1 můžeme vidět, že vývojové prostředí pro Arduino je rozděleno na čtyři základní a to: lišta menu, pod ní se nachází panel nástrojů, dále textový editor a dole můžeme vidět okno zpráv.

Lišta menu obsahuje několik záložek, první z nich je záložka Soubor. Ta obsahuje základní funkce jako uložit rozpracovaný projekt nebo naopak otevřít projekt už hotový. Další záložkou jsou Úpravy, které taktéž obsahují podobné funkce jako další editory, například kopírovat nebo vložit. Třetí záložkou je Projekt, jenž nabízí funkce jako kompilace projektu, nahrání kódu do Arduina nebo přidání knihovny do kódu. Čtvrtou

záložkou jsou Nástroje, které umožňují zapnutí sériového monitoru nebo vybrání portu, jímž je k počítači připojena deska Arduino. Poslední záložkou je Nápověda, která, jak říká název, obsahuje nápovědu k Arduino IDE.

Obrázek 2 Sériový monitor



Zdroj: Vlastní

Na panelu nástrojů můžeme vidět šest ikon. První ikona provede kontrolu kódu a najde syntaktické chyby, které pak zobrazí v okně zpráv, další ikona je pro nahrání. Při nahrávání se nejdříve provede kontrola. Proběhne-li v pořádku, následně se nahraje zdrojový kód v podobě binárního kódu do Arduino. Třetí ikona je pro otevření nového projektu, čtvrtá pro otevření již rozepsaného projektu, pátá pro uložení stávajícího projektu a šestá otevře sériový monitor, který můžeme vidět na obrázku 2. Největší část sériového monitoru zabírá obrazovka, na kterou můžeme pomocí metody Serial.print() zobrazit proměnné, například různé výstupy ze senzorů. Nad touto obrazovkou můžeme vidět textové pole, kde naopak můžeme posílat zprávy Arduinu my. Vpravo dole se nachází možnost zvolit přenosovou rychlost. Ta nám určuje, kolik bitů za sekundu se přenese z Arduina a zpět do něj, kdy základní přenosová rychlost je 9600 bitů za sekundu.

Další částí vývojového prostředí je textový editor. V tomto editoru se vytváří kód pro Arduino. V horní části editoru je možné přepínat mezi jednotlivými záložkami. Poslední částí je okno zpráv. Při kontrole a nahrávání zobrazuje případné syntaktické

chyby a v případě úspěšné kompilace zobrazuje, kolik zabírá paměti a jaká je maximální paměť.

3 Bezdrátová komunikace

Bezdrátovou komunikaci lze charakterizovat jako přenos dat či elektrické energie mezi dvěma body, aniž by tyto dva body byly propojeny elektrickým vodičem. Vzdálenost mezi těmito body může být jen pár metrů (dálkový ovladač od televize) nebo milióny kilometrů (vesmírné družice).

Bezdrátovou komunikaci můžeme rozdělit na satelitní, komunikaci pomocí infračerveného záření, rádiové vysílání, mikrovlnou komunikaci, Wi-Fi a Bluetooth. (Types of Wireless Communication and Its Applications, n. d.)

3.1 Rádiová komunikace

Představuje nejstarší bezdrátovou komunikaci. Každý systém musí obsahovat rádiový vysílač, ten obsahuje zdroj elektrické energie produkující střídavý proud o požadované frekvenci kmitání. Vysílač dále obsahuje systém k modulaci signálu, může jej představovat jednoduché vypínání a zapínání energie nebo detailnější změny jako amplituda, frekvence, fáze nebo kombinace těchto vlastností. Dále je takto upravený signál zesílen a pomocí antény vyslán do volného prostoru. Následně je signál zachycen anténou a přijímačem demodulován do střídavého proudu. (Types of Wireless Communication and Its Applications, n. d.)

3.2 Satelitní komunikace

Satelitní komunikace je rozšířená po celém světě a umožňuje nám být ve spojení. Je tvořena soustavou družic rozmístěných okolo planety. Její princip je jednoduchý, zachytí signál vysílaný ze země, následně ho zesílí a pošle zpět na zem, kde ho zachytí přijímač. (Types of Wireless Communication and Its Applications, n. d.)

3.3 Komunikace pomocí infračerveného záření

K přenosu dat zde dochází pomocí infračerveného záření, které má vlnovou délku delší než červené světlo. Princip je založen na tom, že speciální dioda vyšle infračervený signál, který následně zachytí fotoreceptory. (Types of Wireless Communication and Its Applications, n. d.)

3.4 Mikrovlnná komunikace

Tento typ komunikace je velmi podobný rádiové komunikaci s tím rozdílem, že využívá vlny do velikosti několika centimetrů. K přenosu dat slouží dvě metody, a to metoda pomocí satelitů nebo takzvaná pozemní metoda. Satelitní metoda spočívá v tom, že data jsou přenášena při frekvenci 11 GHz - 14 GHz a přenosovou rychlostí 1 Mb/s až 10 Mb/s. Zatímco principem pozemní komunikace je, že signál proudí mezi dvěma věžemi, mezi kterými nesmí být žádná překážka. Tato metoda využívá frekvenci 4 GHz – 6GHz a přenosovou rychlost mezi 1 Mb/s a 10 Mb/s. (Types of Wireless Communication and Its Applications, n. d.)

3.5 Bluetooth komunikace

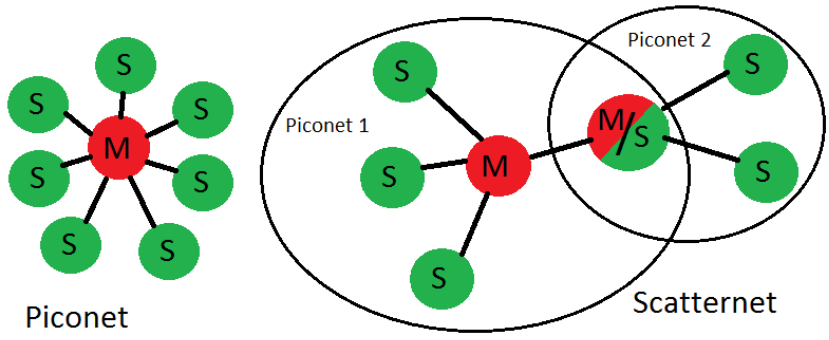
Bluetooth je standard pro bezdrátovou komunikaci na krátkou vzdálenost. Začalo vznikat v roce 1994, kdy firma Ericsson začala hledat způsob, jak propojit dva telefony bez využití kabelu. Následně roku 1998 založila společně se čtyřmi dalšími firmami (IBM, Toshiba, Intel a Nokia) skupinu Bluetooth Special Interest Group (BSIG), jejichž cílem bylo vyvinout standard pro tzv. Wireless Personal Area Network. V roce 1999 uveřejnili první verzi 1.0a. V rámci propagace se pak skupina rozšiřovala, až na stávajících 10 000 členů.

Bluetooth je definováno standardem IEEE 802.14.1 a spadá to kategorie tzv. osobních počítačových sítí. Bluetooth využívá ISM pásmo o frekvenci 2,4 GHz. Jelikož se jedná o hojně využívané pásmo, je zde vysoká pravděpodobnost obsazení kanálů. Musí tedy využívat metodu, která umožní převod signálu s menší šířkou pásma na signál s větší šířkou pásma. O to se stará metoda kmitočtových skoků rozprostřeného spektra, kdy se za jednu vteřinu provede 1600 skoků. (Základy technologie Bluetooth: původ a rozsah funkcí | PC World.cz, 2009)

3.5.1 Topologie

Bluetooth je postaveno na principu Master-Slave, kdy Master navazuje spojení a dále řídí tok dat. Piconet, někdy překládaný jako pikosíť, představuje základní typologickou jednotku. V jedné buňce Piconetu (Obrázek 3) se nachází vždy jeden Master a jeden až sedm Slave zařízení. Spojením několika piconetů vzniká Scatternet (Obrázek 3). (Dočekal, 2014)

Obrázek 3 Ukázka prvků Piconet a Scatternet



Zdroj: Vlastní

Bluetooth podporuje dva způsoby komunikace mezi dvěma zařízeními, a to synchronní spojovanou komunikaci a asynchronní spojovanou komunikaci. (Jäppinen, 2001)

Synchronní spojovaná komunikace vytváří symetrickou komunikaci bod-bod. Tedy mezi jedním zařízením Master a jedním zařízením Slave, kdy zařízení Master posílá data v určitých časových prodlevách a zařízení Slave má pak právo v následujících prodlevách zaslat svá data. Tento typ spojení se využívá pro typ připojení, kde integrita dat má menší prioritu než včasné doručení dat, jedná se například o hlasové přenosy. Zařízení Master může provozovat maximálně tři SCO linky s jedním nebo více zařízeními Slave, zatímco zařízení Slave může mít až tři SCO linky k jednomu zařízení Master nebo jednu linku ke dvěma zařízením Master. (Jäppinen, 2001)

Oproti tomu asynchronní spojovaná komunikace představuje spojení typu bod-multibod, tedy spojení jednoho zařízení Master s více zařízeními Slave. Mezi zařízením Master a zařízením Slave jde zřídit pouze jedno spojení typu ACL. Avšak zařízení

Master může vysílat signál ke všem zařízením Slave ve svém pikonetu, v případě že nezadá adresu příjemce. V tomto případě se jedná o takzvané broadcast vysílání. (Jäppinen, 2001)

4 Windows Phone

Windows Phone je operační systém pro mobilní zařízení od firmy Microsoft. Byl vytvořen jako nástupce operačního systému Windows Mobile, avšak na rozdíl od Windows Mobile, který byl zaměřen především na podniky, Windows Phone je zaměřen na spotřebitele. V roce 2015 byl nahrazen operačním systémem Windows 10 Mobile.

Pro hladký běh operačního systému Windows Phone Microsoft stanovil tyto minimální hardwarové požadavky:

- Dvoujádrový procesor – vzhledem k tomu, že Windows Phone umožňuje multitasking, je vyžadován minimálně dvoujádrový procesor postavený na architektuře ARM.
- Kapacitní dotykový displej – jelikož je Windows Phone ovládán stejně jako většina chytrých telefonů pomocí dotyku, je vyžadován kapacitní dotykový displej, který je schopen rozpoznat až 4 dotyky naráz. Další podmínkou ohledně displeje je minimální rozlišení WVGA (800x480).
- 512 MB RAM – pro hladký běh systému s WVGA rozlišením je vyžadováno minimálně 512 MB RAM paměti. Pro zařízení s větším rozlišením displeje je pak vyžadována dvojnásobná paměť.
- 4 GB flash paměti
- Fotoaparát
- Senzory – telefon musí obsahovat senzor přiblížení, senzor světla, GPS přijímač a akcelometr.
- Hardwarová tlačítka – telefon musí obsahovat tato tlačítka: odemykací, nastavení hlasitosti, fotoaparát, zpět, úvodní obrazovka a vyhledávání.
- Grafický procesor s podporou DirectX (Holuša, 2014)

4.1 Historie

V roce 2004 začal Microsoft vyvíjet aktualizaci pro Windows Mobile s číslem 7 označovanou jako „Photon“, s plánovaným datem vydání v roce 2017. Tato aktualizace měla stejně jako předchozí využívat Windows CE framework, určený pro malé počítače. Avšak vývoj byl příliš pomalý, a tak byl nakonec zrušen. (From Photon to the future: A not-so-brief history of Windows Phone, 2016)

V roce 2008 pak začal Microsoft vyvíjet nový operační systém, jenž měl být představen v roce 2009, ale nakonec, kvůli průtahům, byl představen až v roce 2010 jako Windows Phone 7. Ten se na první pohled od ostatních operačních systémů odlišoval tím, že úvodní obrazovky tvořily dynamické dlaždice. To jsou jednotlivé barevné bloky, které poskytují základní informace, aniž by bylo nutno danou aplikaci zapínat. Další zajímavou novinkou byly takzvané huby, ty spojovaly několik aplikací do jedné a eliminovaly tak nutnost přepínat mezi jednotlivými aplikacemi. Například hub Lidé čerpá kontakty z několika zdrojů, například Facebook či Gmail. Avšak trpěl mnoha nedostatky, jako absence možnosti kopírovat/vložit, nastavení vyzvánění nebo podpora multitaskingu. (From Photon to the future: A not-so-brief history of Windows Phone, 2016)

Obrázek 4 Ukázka úvodní obrazovky Windows Phone 8.1

Zdroj: Windows Phone 8.1: Die neuen Funktionen in Bildern - areamobile.de, 2014

Další významný update vyšel v roce 2011 pod označením Windows Phone 7.5 Mango. Tento update přinesl Microsoft Explorer 9 se stejnými standardy a grafickým vzhledem jako u počítačové verze. Dále obsahoval vylepšení multitaskingu, možnosti přepínání mezi aplikacemi bez nutnosti jejich vypnutí. Další změnou byla možnost vyhledávání aplikací podle názvu, kdy aplikace byly seřazeny podle abecedy. (From Photon to the future: A not-so-brief history of Windows Phone, 2016)

Další update Windows Phone 8 Apollo přinesl významnou změnu v tom, že už neběžel na Windows CE, ale na Windows NT, tedy na stejném základu jako Windows určené pro počítače. Od toho si Microsoft sliboval snadnější sdílení aplikací pro Windows a Windows Phone. Tento update byl vydán v roce 2012. Mezi nové funkce patřila nová, více flexibilní rozvržení úvodní obrazovky, podpora NFC komunikace, Internet Explorer 10, podpora microSD karet nebo možnost dělat snímky obrazovky. Velkou nevýhodou byla absence možnosti update 7.x verzí Windows Phone na verzi 8. Malou útěchou pro majitele telefonů s verzí 7.x mohlo být vydání updatu 7.8, který obsahoval většinu funkcí Windows Phone 8. (From Photon to the future: A not-so-brief history of Windows Phone, 2016)

Další verze Windows Phone byla vypuštěna v roce 2014 pod označením Windows Phone 8.1 Blue (Obrázek 4). Mezi významné prvky updatu patří přidání notifikačního centra, rozdělení nastavení hlasitosti pro vyzvánění a aplikace, nastavení pozadí dlaždic a možnosti přidání třetího sloupce a Internet Explorer 11. Dalším zajímavým přidaným prvkem je hlasový asistent Cortana a World Flow klávesnice, kde se nepíše klasickým mačkáním znaků ale pomocí tahu prstu. Bohužel obě tyto aplikace nejsou dostupné na území České republiky. (From Photon to the future: A not-so-brief history of Windows Phone, 2016)

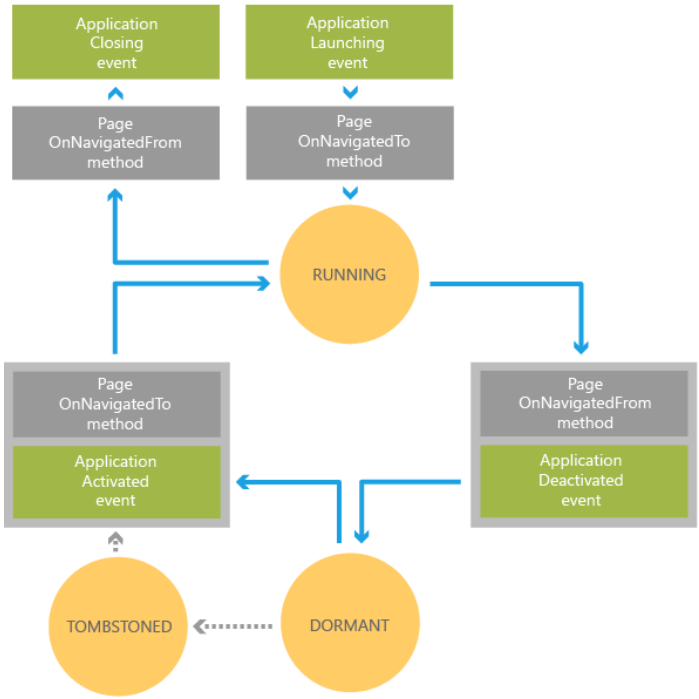
Zatím poslední update proběhl v roce 2015, pod názvem Windows 10 Mobile. Microsoft se snaží tímto názvem ukázat stávající snahu sjednotit operační systém pro různé platformy.

4.2 Životní cyklus aplikace

Výchozím stavem aplikace je stav not running, tedy neběžící. Při spuštění aplikace se zavolá událost *Application_Launching*, ve které se načtou aplikační data, a

aplikace se dostane do operační paměti. Následně aplikace přejde do stavu running, tedy běžící. Vypne-li se aplikace, zavolá se ještě událost *Application_Closing*, která nám umožní uložit data. Poté dojde k odstranění z aplikační paměti a aplikace se vrátí do stavu not running. Po zavolání události *Application_Closing* má aplikace na uložení dat 10 vteřin, proto je dobré data ukládat během celého životního cyklu aplikace (Obrázek 5). (App activation and deactivation for Windows Phone 8, ©2017)

Obrázek 5 Grafické znázornění životního cyklu aplikace



Zdroj: App activation and deactivation for Windows Phone 8, ©2017

Ačkoli Windows Phone podporuje multitasking, to znamená, že může běžet více aplikací naráz, v popředí může běžet vždy jen jedna. V případě, že přepneme na jinou aplikaci, zavolá se událost *Application_Deactivated*, tím se aplikace dostane do stavu Dormant, neboli spící. Do stavu Dormant se může dostat také bez zásahu uživatele, např. při příchozím telefonním hovoru nebo při automatickém uzamknutí obrazovky. Ve stavu Dormant aplikace přestane využívat procesor, ale zůstává uložena v operační paměti. (App activation and deactivation for Windows Phone 8, ©2017)

V případě, že uživatel spustí další aplikaci a operační paměť je nedostačující, systém převede nejstarší aplikaci ve stavu Dormant do stavu Tombstoned, neboli pohřbena. V tomto stavu je aplikace odstraněna z operační paměti, avšak je zachován

obraz, ve kterém byla aplikace potlačena. V případě, že je aplikace pak opětovně spuštěna, je zavolána událost *Application_Activated*, která otevře pohřbenou aplikaci. Aplikace je pak otevřena na stránce, na jaké byla pohřbena, avšak na rozdíl od stavu Dormant, musí svá data opět načíst. (App activation and deactivation for Windows Phone 8, ©2017)

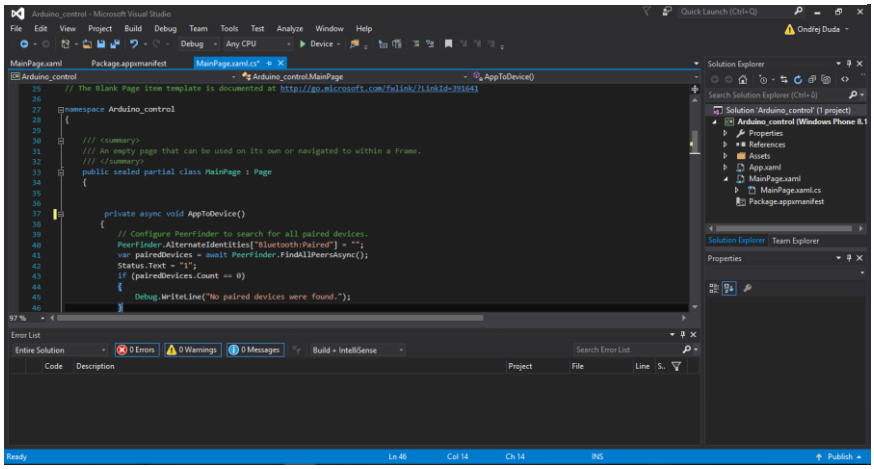
4.3 Vývoj aplikace

Pro samotný vývoj aplikace je potřeba splnit několik požadavků. Základním je znalost programovacího jazyka, ve kterém budeme aplikaci psát. Dalším důležitým požadavkem je vývojové prostředí schopné pracovat s daným jazykem.

4.3.1 Vývojové prostředí Visual Studio

Visual Studio (obrázek 6) je vývojové prostředí, které je vyvíjeno firmou

Obrázek 6 Vývojové prostředí Microsoft Visual Studio Comunity 2015



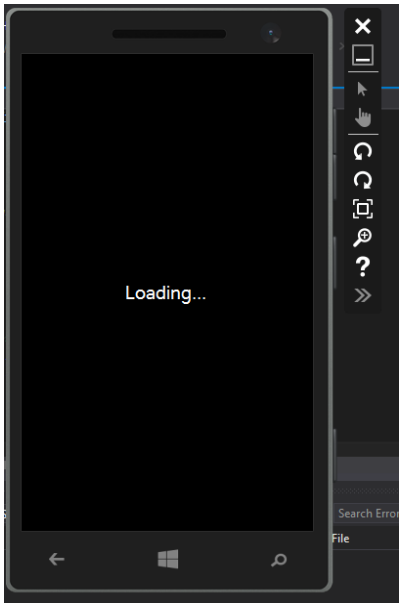
Zdroj: Vlastní

Microsoft. Umožňuje vývoj aplikaci pro operační systémy Windows, Android a iOS. Dále umožňuje tvorbu aplikací jak pro stolní počítače nebo mobilní telefony, tak i tvorbu webových stránek nebo databází. Visual studio podporuje programovací jazyky Visual Basic, Visual C#, Visual C++, Visual F# a Java Script. Jednou z výhod Visual Studia je funkce IntelliSense, ta v průběhu psaní kódu zobrazuje možné proměnné či metody, které jsou k dispozici, a tím významně ulehčuje práci a čas vývojáře. Visual Studio je dostupné ve verzích Community, Professional a Enterprise. (Welcome to Visual Studio 2015, ©2017)

4.3.2 Windows Phone SDK

Základní verze Visual Studia neobsahuje potřebné nástroje pro vývoj aplikací na Windows Phone, proto je nutné tuto sadu nástrojů nainstalovat. To jde provést dvěma způsoby, buď jako doplněk pro Visual Studio nebo jako samostatné vývojové prostředí, kde jde vyvíjet pouze aplikace pro Windows Phone. (Windows SDK pro Windows 8.1 – vývoj aplikací pro Windows, 2015)

Obrázek 7 Windows Phone Emulátor



Zdroj: Vlastní

Výhodou této sady nástrojů je, že obsahuje i takzvaný emulátor (obrázek 7). To je virtuální zařízení, na kterém daná aplikace poběží, odpadá nám tedy nutnost dané zařízení vlastnit. Ovšem chceme-li testovat aplikaci přímo na fyzickém zařízení, je nutné nejdříve toto zařízení registrovat jako zařízení vývojářské. To lze pomocí nástroje Windows Phone Developer Registration, který se nainstaluje společně s Windows Phone SDK. Pak už stačí pouze propojit zařízení s počítačem a přihlásit se pomocí vývojářského účtu. (Windows SDK pro Windows 8.1 – vývoj aplikací pro Windows, 2015)

4.3.3 Programovací jazyk C#

C# je programovací jazyk vyvinutý firmou Microsoft. Syntaxe vychází z jazyků Java a C++ a je tedy členem rodiny C jazyků. C# patří mezi takzvané řízené jazyky s tzv. garbage collectorem. Tyto jazyky mají automatickou správu paměti a není nutno

se tedy starat o takzvané pointery, což jsou ukazatele, které vývojář dostal od operačního systému. Tyto pointery ukazovaly na část paměti, kterou nám operační systém přidělil. Často se ale stávalo, že program využil více místa, než mu bylo přiděleno, a tím mohl přepsat jiná data. Právě o toto se automatická správa paměti, která za cenu snížení výkonu pomáhá vývojáři ušetřit čas, stará. Další zajímavou schopností jazyka C# je možnost asynchronního programování. To se využívá například v situaci, kdy aplikace pracuje s webem. V synchronním programování se celý program zastaví a přestane komunikovat s uživatelským rozhraním, dokud se webová stránka nenačte, zatímco v asynchronním programování program nepřestává komunikovat. Je zde pak využíván operátor await, který slouží k tomu, že se program zastaví, dokud se nedokončí asynchronní operace. (Troelsen, 2015)

4.3.4 Značkovací jazyk XAML

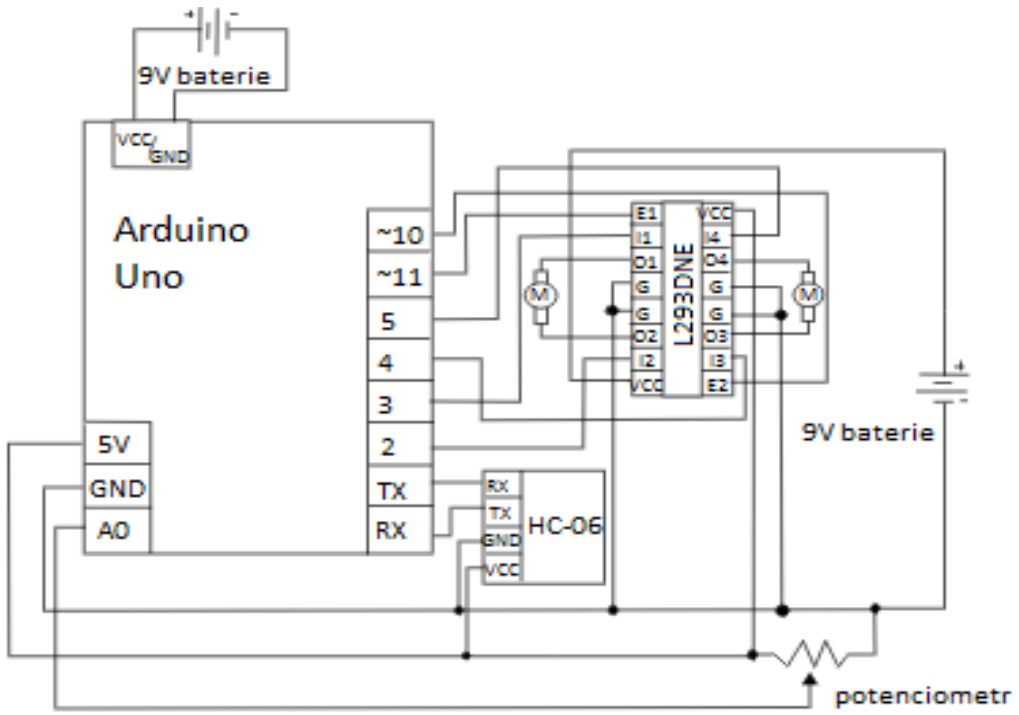
XAML je značkovací jazyk vyvinutý firmou Microsoft, který slouží k tvorbě uživatelského rozhraní pro aplikace využívající .NET Framework. Důvodem jeho vzniku byla snaha oddělit vývoj uživatelského rozhraní a aplikační logiky. Výsledkem této snahy je právě XAML jazyk, kdy je uživatelské rozhraní definováno v XAML souboru, zatímco aplikační logika je umístěna v souboru jiném. Jazyk XAML vychází ze struktury elementů XML, kde jsou jednotlivé elementy do sebe postupně zanořovány. (XAML Overview (WPF), ©2017)

5 Základní popis konstrukce robota a využitých součástí

Robot se skládá ze dvou základních částí. První je zařízení s bluetooth modulem a operačním systémem Windows Phone 8.1, v mém případě se jedná o chytrý telefon Nokia Lumia 520. Pro toto zařízení jsem naprogramoval vlastní aplikaci, jejíž jedinou funkcí je prozatím propojit se pomocí Bluetooth s Arduinem a následně mu pomocí čtyř šipek udávat směr jízdy.

Druhou částí je samotný robot, na obrázku 8 můžeme vidět schéma zapojení. Jeho podvozek je vyroben ze stavebnice Merkur. Jako zařízení umožňující pohyb jsem si vybral pásy. Pásům před koly jsem dal přednost ze dvou důvodů. Prvním důvodem byla manévrovatelnost, kdy se robot díky pásům otočí prakticky na místě. Druhým pak lepší schopnost překonávat menší překážky. Kola pohánějící pásy se mohou pohybovat díky dvojici motůrků od firmy Tamiya, které jsem získal i s převodovou skříní. Směr otáčení a počet otáček za minutu jsou řízeny jednodeskovým počítačem Arduino Uno. To dostává povely od telefonního zařízení pomocí technologie Bluetooth, kdy toto spojení na straně robota zajišťuje Bluetooth modul HC-06. Dalšími důležitými součástkami jsou H-můstek, potenciometr a baterie. H-můstek umožňuje změnu směru otáčení motorů, zatímco potenciometr využívám jako možnost nastavení rychlosti. Za zdroj napájení

Obrázek 8 Schéma zapojení



Zdroj: vlastní

jsem zvolil 9 V baterii. V robotu využívám dvě, jednu pro napájení Arduina, druhou pak pro motory. Poslední částí je kontaktní nepájivé pole, které umožňuje sestavení elektrického obvodu bez nutnosti pájení.

5.1 Telefonní zařízení Nokia Lumia 520

Nokia Lumia 520 (Obrázek 9) je mobilní telefon vyráběný finskou firmou Nokia. Tento telefon byl poprvé představen v únoru 2003 na veletrhu mobilních telefonů MWC v Barceloně. Jedná se o nejlevnější verzi mobilního telefonu z řady Lumia. Stejně jako ostatní telefony z této řady využívá operační systém Windows Phone, avšak na rozdíl od výkonnějších telefonů poslední aktualizace proběhla na verzi 8.1 a ani se neplánuje update na Windows 10 Mobile. Lumia 520 využívá kapacitní IPS LCD display o velikosti úhlopříčky 4 palce, rozlišení 480 x 800 pixelů a jemnosti 233 pixelů na jeden

Obrázek 9 Nokia Lumia 520

Zdroj: Nokia Lumia 520 - Full phone specifications, © 2000-2017

palec. Velkou nevýhodou je absence technologie ClearBlack, jejímž největším důsledkem je vysoká míra odlesku na slunci. Avšak velkou výhodou je velice citlivý display, díky kterému je možné využívat tento mobilní telefon i v rukavicích, což se velmi hodí v zimních měsících. Pod displejem se pak nachází ikonické trio tlačítek, typické pro celou řadu Lumií. Zleva je to tlačítko zpět, uprostřed je tlačítko, které nás vrátí na úvodní obrazovku s tím, že spuštěná aplikace bude přesunuta na pozadí, a poslední tlačítko slouží pro online vyhledávání. Nad displejem se nachází dva senzory, a to senzor okolního světla pro regulaci jasu displeje a senzor přiblížení. Další důležitou

částí prostoru nad displejem je reproduktor. Na zadní straně telefonu je fotoaparát o rozlišení 5MPx, bohužel zde však chybí dioda, která slouží jako blesk nebo může být využita jako baterka. V pravém dolním rohu se nachází další reproduktor. Na spodní straně je umístěn konektor pro microUSB, který slouží k nabíjení telefonu nebo pro propojení telefonu s počítačem. Na horní hraně se nachází 3,5mm vstup pro sluchátka. Na pravé straně jsou tlačítka pro nastavení hlasitosti, uzamknutí/odemčení obrazovky a tlačítko pro fotoaparát. Telefon využívá čipset Qualcomm Snapdragon MSM8227, s dvoujádrovým procesorem o výkonu 1 GHz, a grafický akcelerátor Adreno 305. Dále má 512 MB operační paměti a 8 GB stálé paměti, kterou je možné rozšířit až o dalších 64 GB pomocí paměťové karty. Baterie telefonu má kapacitu 1430 mAh (Nokia Lumia 520: okna s nejnižší cenovkou [recenze] - Mobil, 2013)

5.2 Bluetooth modul HC-06

HC-06 je Bluetooth modul pracující při napětí 3.3 V (obrázek 10), kdy na trhu je

Obrázek 10 Bluetooth modul HC-06

Zdroj: http://www.fecegypt.com/uploads/dataSheet/1480849570_hc06.pdf

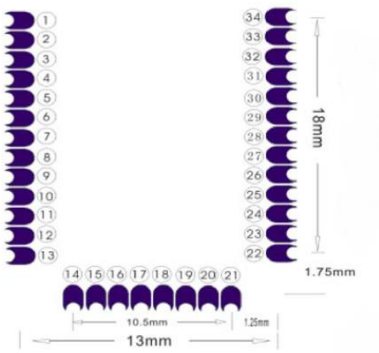
běžně k dostání připájený k desce plochých spojů, jíž můžeme vidět na obrázku 11, kde zelená část je samotný Bluetooth modul a modrá část je přídavná deska plochých spojů.

Obrázek 11 Bluetooth modul s přídavnou PCB

Zdroj: http://robotstore.cz/obchod/arduino/hc-06-bluetooth-rs232-ttl-adapter-arduino-modul/

Ta obsahuje navíc stabilizátor napětí, signalizační LED, několik diod a rezistorů, čímž je umožněno používat tento modul i s přístroji, které pracují s vyšším napětím. Uváděné maximální napětí je 6 voltů, při 7 voltech hrozí vážné poškození modulu. Z této rozšiřující destičky vedou 4 piny a to: VCC, RXD, TXD, a GND. Pin VCC slouží pro napájení, udávaná hodnota rozpětí pro napětí je udávána jako rozmezí 3.6–6 V, kdežto pin GND slouží jako uzemnění. Piny RXD a TXD slouží pro sériovou komunikaci při napětí 3.3 V, kde TXD je sériový výstup a RXD je sériový vstup. Signalizační dioda ukazuje stav propojení modulu se zařízením. Bliká-li dioda v rychlých sekvencích, znamená to, že modul není právě propojen se žádným zařízením. Svítí-li dioda konstantně, znamená to, že modul je právě s nějakým zařízením propojen. Co se týče samotného modulu, ten je vždy v režimu SLAVE, tudíž nikdy nemůže navázat spojení, a jen čeká, až nějaké zařízení spojení naváže. Pomocí AT příkazů je možné měnit konfiguraci modulu. Základní konfigurace je: přenosová rychlost 9600 bitů za sekundu, jméno a pin, případně si ještě můžeme nechat vypsat aktuální verzi firmware. (Sakul World - Bluetooth HC-06, © 2016)

Obrázek 12 Číslování pinů Bluetooth modulu HC-06



http://www.fecegypt.com/uploads/dataSheet/1480849570_hc06.pdf

Na obrázku 12 můžeme vidět číslování jednotlivých pinů, kdy přídavná PCB využívá jen některé piny. Seznam pinů, jichž využívá přídavná PCB, a jejich funkcí můžeme vidět v tabulce 2.

Tabulka 2 Tabulka využívaných pinů

PIN	Název pinu	Popis pinu
1	TX	UART výstup
2	RX	UART vstup
12	3.3 V	Zdroj energie, kdy standardní napětí je 3.3 V, můžeme však pracovat v rozpětí 3–4.2 V
13	GND	uzemnění
24	PI01	Slouží k indikaci spojení s jiným Bluetooth zařízením. Není-li spojení, každých 102 ms vyšle signál, naopak jeli spojení, vysílá signál nepřetržitě.
26	PI03	Určuje, zdali je možné zadávat AT příkazy, nebo ne.

Zdroj: http://www.fecegypt.com/uploads/dataSheet/1480849570_hc06.pdf

V tabulce 3 máme uvedeny jednotlivé technické specifikace.

Tabulka 3 Technická specifikace Bluetooth modulu HC-06

Napájecí napětí	3,6-6 V
Odběr proudu	30-40 mA
Komunikace	UART
Podporované rychlosti	1200-1382400 b/s
Dosah	cca 10 m
Konfigurace	Pomocí AT příkazů
Režim provozu	SLAVE
Bluetooth specifikace	v2.0 + EDR

Zdroj: Sakul World - Bluetooth HC-06, © 2016

5.3 Vývojová deska Arduino UNO

Arduino Uno (obrázek 13) představuje základní model z řady vývojových desek Arduino, a to díky jeho ceně, snadnosti použití a variabilitě využitelnosti.

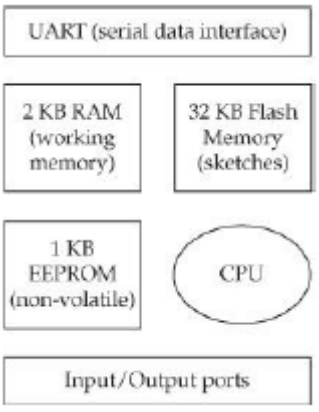
Obrázek 13 Arduino Uno

Zdroj: Arduino – Counterfeit, © 2017)

5.3.1 Mikrokontrolér

Srdcem celého Una je mikrokontrolér od firmy Atmel. Konkrétně se jedná o mikrokontrolér ATmega328 s 28 piny. Ten je zasazen ve dvouřadé patici určené právě pro mikrokontroléry s 28 piny.

Obrázek 14 Diagram součástí mikrokontroléru Atmega328



Zdroj: Monk, 2012

Na blokovém diagramu (obrázek 14) můžeme vidět jeho hlavní součásti. Mozkem celého mikrokontroléru je centrální procesová jednotka, která kontroluje veškeré výstupy z mikrokontroléru. Nejdříve vyhledá příkazy ve Flash paměti, v které jsou

uloženy sketche, a ty pak následně provede. To může zahrnovat práci s pracovní pamětí, takzvanou RAM pamětí, kde může data měnit a následně vracet nebo měnit digitální výstupy z 0 na 5 V. EEPROM paměť je stejně jako Flash paměť nevolativní, tedy po zapnutí a vypnutí zařízení nebude to, co bylo v paměti uloženo, vymazáno. Zatímco Flash paměť obsahuje programové instrukce ze sketche, EEPROM paměť slouží k uchovávání informací, které nechce ztratit ani po restartování mikrokontroléru. (Monk, 2012)

5.3.2 Zdroj napájení

Uno lze napájet pomocí USB konektoru nebo zásuvky. Ta může být napojena na zdroj elektrické energie o napětí 7 až 12 voltů. Toto napětí je pomocí regulátoru napětí regulováno na 5 voltů, při kterém Arduino pracuje. (Monk, 2012)

5.3.3 Napájecí piny

Tyto piny slouží k napájení elektrického okruhu, který bude ovládán Arduinem. Je zde pin pro 5 nebo 3,3 voltové napětí. Dále jsou zde dva piny pro uzemnění, vin pin, který má stejnou funkci jako zásuvka. Posledním pinem je reset. Ten slouží ke stejnému účelu jako tlačítko reset, tedy restartování programu. (Monk, 2012)

5.3.4 Analogové piny

Arduino Uno obsahuje šest analogových pinů, označené A0 až A5. Tyto piny slouží jako piny vstupní, měří vstupní napětí, které vysílají různé senzory. (Monk, 2012)

5.3.5 Digitální piny

Arduino Uno má celkem 14 digitálních pinů, ty mohou sloužit buď jako vstupní nebo výstupní. Pokud jsou využity jako vstupní, fungují stejně jako piny analogové, naopak, jsou-li využity jako piny výstupní, fungují podobně jako 5 voltový napájecí pin. Ovšem na rozdíl od napájecího pinu mohou být vypnuty a tím poskytovat napětí 0 voltů. Tento stav se určí pomocí příkazu ve sketchi. První dva piny 0 a 1 jsou též označovány jako RX a TX. Tyto dva piny jsou vyhrazeny pro sériovou komunikaci, kdy RX (Recived) slouží pro přijímání dat a TX (Transmited) pro posílání dat. (Monk, 2012)

5.4 Pohonná jednotka

Pohyb robota je zajišťován pomocí dvojice pásů. Tyto pásy pohání převodová skříň od výrobce Tamiya, konkrétně se jedná o Tamiya 89918 Double Gearbox Kit s převodovým poměrem 114.7:1. V této skříni jsou usazeny dva kartáčové jednosměrné motory Mabuchi FA-130RA-1810, jejichž základní technickou specifikaci můžeme vidět v tabulce 4.

Tabulka 4 Technická specifikace motorů

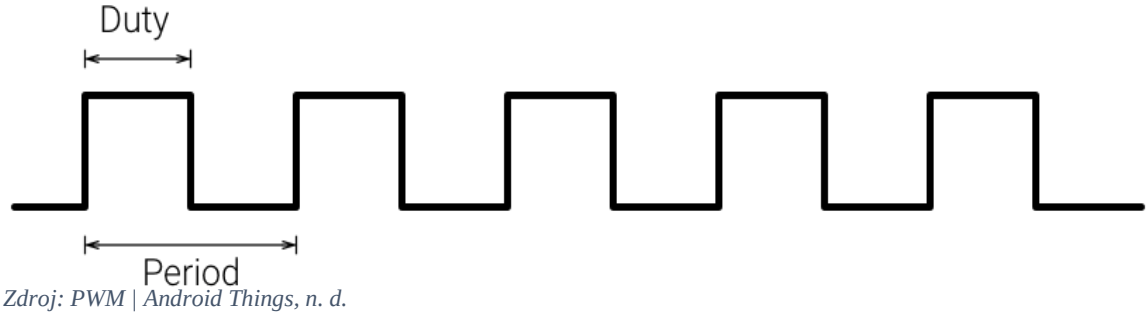
Model	FA-130RA-1810
Operační napětí	1,5~3,0 V
Nominální napětí	3 V
Otáčky bez zátěže	12 300 otáček za minutu
Proud bez zátěže	150 mA

Zdroj: https://www.pololu.com/file/0J11/fa_130ra.pdf

5.4.1 Ovládání motorů

Rychlost otáčení motoru je přímo úměrná k napětí poskytovanému motoru. Toho využívá jeden ze základních způsobů ovládání rychlosti motoru. Jedná se o takzvané odporové řízení. Jeho princip spočívá v tom, že se mezi motor a zdroj napájení umístí potenciometr. Pomocí potenciometru pak měníme odpor a tím i napětí, které proudí do motoru. Zvýšíme-li odpor a tím snížíme napětí proudící do motoru, snížíme tím rychlost

Obrázek 15 Ilustrace PWM cyklu



Zdroj: PWM | Android Things, n. d.

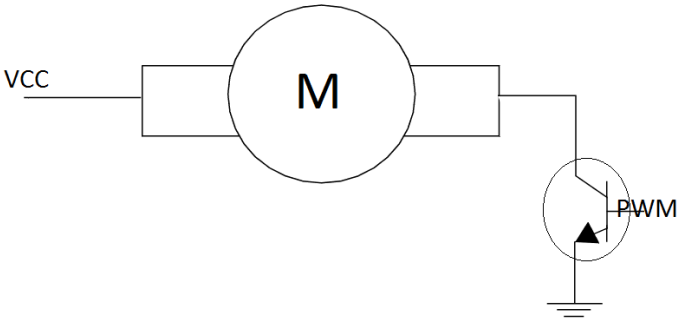
otáčení motoru a naopak, snížíme-li odpor, zvýší se napětí a následně i rychlost otáčení motoru.

Tento způsob řízení má však několik velkých nevýhod. Zásadní nevýhodou je nízká efektivita. Ta je způsobena tím, že proud prochází přes vysoký odpor a díky tomu se velká část elektrické energie přeměňuje na teplo. Další nevýhodou je snížení momentu síly v důsledku snížení napětí. Díky tomu se může při nižším napětí stát, díky jevu známému jako hystereze, že motor ztratí dostatek momentu síly a nebude se otáčet.

Kvůli těmto důvodům jsem musel tuto metodu ovládání motorů zavrhnout. Ovšem měl jsem k tomu ještě jeden důvod. Jelikož se nastavuje potenciometr fyzicky a ne digitálně, je toto řízení motorů pro dálkové ovládání nevhodné.

Tyto nedostatky však eliminuje pulzní metoda. Základem této metody řízení je takzvaná šířková modulace. Jedná se o typ modulace signálu, kdy signál může dosahovat dvou hodnot, a to jedna a nula. Signál je periodický, opakující se. Jeden cyklus se nazývá perioda, druhou veličinou je šířka pulzu, která vyjadřuje tu část periody, kdy byl signál jedna. Poměr mezi šířkou pulzu a zbytkem periody se označuje střída. Střída může nabývat hodnot nula až jedna, případně ji lze vyjádřit v procentech. Protože vstupním napětím Arduina je 5 V, při logické úrovni signálu 1 bude napětí rovno 5 voltům, v případě, že bychom měli zdroj s větším napětím, signál by byl roven právě tomuto napětí. Zatímco logická úroveň signálu 0 se vždy rovná nulovému napětí. Výhodou tohoto typu řízení je, že na rozdíl od obyčejné spojité regulace dochází pouze ke snižování napětí, a ne i proudu. Díky tomu má motor větší sílu při nižších otáčkách. Tento způsob regulace je navíc prakticky bezztrátový. To je způsobeno tím, že přechází do plně otevřeného stavu velmi rychle a tím se vyhýbá ztrátovým oblastem.

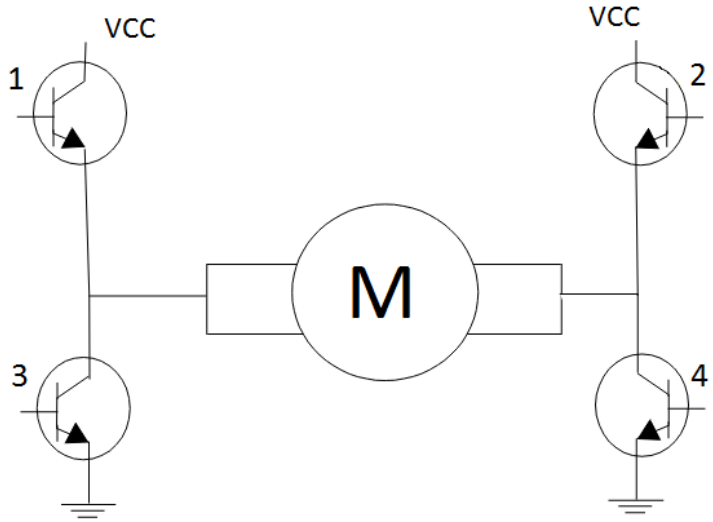
Obrázek 16 Řízení motoru pomocí tranzistoru a PWM signálu



Zdroj: Vlastní

Samotné řízení motoru se provádí pomocí tranzistoru zapojeného sériově s motorem. Motor je zapojen ke kolektoru tranzistoru a signál je přiváděn na bázi tranzistoru, jak je možno vidět na obrázku 16. Kvůli jevu zvanému napěťová špička, způsobená vinutím motoru, hrozí poškození tranzistoru. Aby se této situaci zabránilo, využívá se k tomu dioda, která se zapojí paralelně s motorem.

Obrázek 17 Schéma řízení motoru pomocí H-můstku



Zdroj: Vlastní

Nevýhodou tohoto zapojení je, že nejde určovat směr otáčení. Problém můžeme vyřešit použitím 4 tranzistorů. Takovému zapojení se říká můstkové. Pro zapojení jsou potřeba dva nosné signály. První signál slouží k určování rychlosti v jednom směru, druhý signál slouží k určování rychlosti v směru opačném. Pokud se budeme řídit obrázkem 17, a chceme roztočit motor jedním směrem, musíme otevřít tranzistory 1 a 4. Chceme-li roztočit motor na druhou stranu, musíme otevřít tranzistory 2 a 3. Otevřeme-li tranzistory 1 a 3 nebo 2 a 4, motor se nebude otáčet a vzniká zkrat, což je elektrický okruh bez spotřebiče. V tabulce 5 můžeme vidět, zdali jsou motory aktivní, podle toho, jaké tranzistory jsou otevřené, kdy 1 znamená otevřený a 0 zavřený.

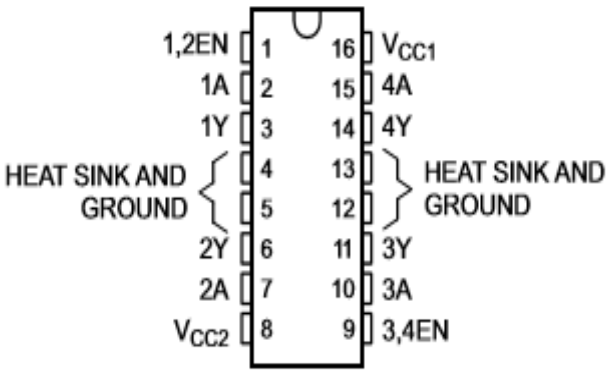
Tabulka 5 Směr otáčení motoru podle otevřených tranzistorů

Tranzistor 1	Tranzistor 2	Tranzistor 3	Tranzistor 4	Směr otáčení motoru
1	0	0	1	Po směru hodinových ručiček
0	1	1	0	Proti směru hodinových ručiček
1	1	0	0	Netočí se
0	0	1	1	Netočí se

Zdroj: vlastní

Pro řízení motorů jsem si vybral čtyřnásobný silnoproudý half-H budič L293DNE od firmy Texas Instruments. Tento typ obsahuje dva H-můstky s integrovanými diodami proti zpětnému proudu. Na obrázku 18 jsou vidět funkce jednotlivých pinů.

Obrázek 18 Rozložení pinů a jejich funkce budiče L293DNE



Zdroj: http://www.ti.com/lit/ds/symlink/l293.pdf

V tabulce 6 můžeme vidět bližší specifikaci těchto pinů.

Tabulka 6 Tabulka popisu pinů řadiče L293DNE

Pin		Typ	Popis
Název	Číslo		
1,2EN	1	Vstup	Umožní řízení kanálu jedna a dva
<1:4>A	2, 7, 10, 15	Výstup	Vstupní piny řadiče
<1:4>Y	3, 6, 11, 14	Vstup	Výstupní piny řadiče
3,4EN	9	Vstup	Umožní řízení kanálu tři a čtyři
GROUND	4, 5, 12, 13	-	Uzemnění
Vcc1	16	-	Pěti voltový zdroj pro řízení kanálů
Vcc2	8	-	Zdroj pro výstupní piny, umožňující rozsah 4,5-36V

Zdroj: http://www.ti.com/lit/ds/symlink/l293.pdf

6 Aplikace pro Arduino Uno

Program jsem psal ve vývojovém prostředí Arduino Uno. V následující kapitole popíšu jednotlivé části kódu pro Arduino.

6.1 Knihovny

Arduino umožňuje využívání knihoven, které poskytují, jako ve většině vývojových platform, výrazné usnadnění práce. Jedná se o funkční celek, který nabízí služby pro programy. Ve většině případů se jedná o sérii funkcí, datových typů a procedur. Knihovna nabízí aplikační programátorské rozhraní, které umožňuje využívat funkce poskytované touto knihovnou. Arduino IDE ihned po nainstalování poskytuje několik základních knihoven, které jde v kódu aktivovat. Já jsem v kódu využil knihovnu *SoftwareSerial*, jejíž implementaci můžeme vidět v ukázce kódu 1 na řádku 2.

Tato knihovna umožňuje využívat i jiné digitální piny, jako piny sloužící pro sériovou komunikaci, než ty původně nastavené, tedy piny nula a jedna. Tuto knihovnu jsem musel využít z důvodu, že veškerá sériová komunikace jde právě přes tyto dva piny. Tudíž, mám-li zapojený Bluetooth modul, ten zabírá piny pro sériovou komunikaci a já následně nemůžu nahrávat kód z počítače do Arduino či využívat sériový monitor.

Po implementaci knihovny je ještě nutné přiřadit piny pro sériovou komunikaci, jak můžeme vidět v ukázce kódu 1 na řádku 3, nastavil jsem piny 6 a 7 právě pro tento účel.

Ukázka kódu 1 Bluetooth komunikace

```
2  #include <SoftwareSerial.h>
3  SoftwareSerial BTserial(6, 7); // RX , TX

30  BTserial.begin(9600);

35  order = BTserial.read();
```

Zdroj: Vlastní

6.2 Nastavení vstupů a výstupů

Dalším důležitým krokem bylo nadefinování vstupů a výstupů. To můžeme vidět v ukázce kódu 2 na řádcích 5 až 11, kdy všechny proměnné jsou typu integer a jsou konstantní, slouží tedy pouze ke čtení a nejdou dále měnit. V případě, že bychom se je pokusili následně změnit, program nám nahlásí chybu, což se nám velice hodí. Jelikož se jedná o určení pinů, které budou vstupy a výstupy, je jen dobře, že si je nemůžeme v kódu omylem změnit. První čtyři proměnné nám budou určovat směr otáčení motorů, další dvě stav motorů, tedy jsou-li zapnuty nebo vypnuty. Poslední proměnná nám bude získávat data z potenciometru určujícího rychlost otáčení motorů. Zároveň jsem musel proměnným *LevyEnablePin* a *PravyEnablePin* nastavit piny 10 a 11, protože se jedná o piny umožňující pulzní šířkovou modulaci signálu, díky které se určuje rychlost motorů.

Následně jsem nadefinoval, jaké piny budou vstupní a jaké výstupní. To se provede ve funkci *setup()*; Tuto funkci musí obsahovat každý sketch a slouží například k inicializaci proměnných či k určení pinům, zdali budou vstupní nebo výstupní. To se dělá pomocí funkce *pinMode()*. Do této funkce se zadávají dvě proměnné. První je určení pinu, který budeme definovat, druhou proměnnou pak mód, jaký bude daný pin mít. Pin může mít tři módy, první je OUTPUT tedy výstup, druhý INPUT, česky vstup, a poslední INPUT_PULLUP. Tento mód je také vstupní, s rozdílem, že využívá zabudovaný rezistor v desce Arduino. Hodnota rezistoru se liší podle využívané desky.

Jak můžeme vidět v ukázce kódu 2 na řádcích 18 až 23, všechny piny jsou nastaveny jako výstupní, až na pin pojmenovaný jako *potPin*, který zde vůbec není. To je z důvodu, že pin A0 je přímo nastaven jako analogový vstup. Následně je potřeba nastavit u výstupů mód po zapnutí. To se provádí funkcí *digitalWrite()*. Tato funkce má opět dvě hodnoty. První je určení pinu, který budeme nastavovat. Do druhé hodnoty se pak zadávají hodnoty LOW a HIGH, které nám reprezentují logický výstup 0 a 1, v případě Arduina Uno se jedná a výstup 0 V a 5 V. A jak vidíme v ukázce kódu 2 na řádcích 24 až 27, všechny výstupní piny byly nastaveny na 0 V.

To se ale netýká pinů *LevyEnablePin* a *PravyEnablePin*, jelikož se jedná o analogové výstupy. Tam jsem využil funkci *analogWrite()*, která je stejná jako *digitalWrite()*, jak můžeme vidět v ukázce kódu 2 na řádcích 28 a 29. Tyto metody se liší pouze typem výstupu. Jak již bylo řečeno, digitální výstup má pouze logické úrovně

0 a 1. V případě analogového výstupu se ovšem jedná o pulzně šířkový modulační signál. Ten může dosahovat hodnot 0 až 255, kdy 255 se rovná logické 1.

Ukázka kódu 2 Nadefinování a nastavení pinů

```
5  const int Levycontrolpin1 = 2;
6  const int Levycontrolpin2 = 3;
7  const int Pravycontrolpin1 = 4;
8  const int Pravycontrolpin2 = 5;
9  const int LevyEnablePin = 10;
10  const int PravyEnablePin = 11;
11  const int potPin = A0;

18     pinMode(Levycontrolpin1, OUTPUT);
19     pinMode(Levycontrolpin2, OUTPUT);
20     pinMode(Pravycontrolpin1, OUTPUT);
21     pinMode(Pravycontrolpin2, OUTPUT);
22     pinMode(LevyEnablePin, OUTPUT);
23     pinMode(PravyEnablePin, OUTPUT);
24     digitalWrite(Levycontrolpin1, LOW);
25     digitalWrite(Levycontrolpin2, LOW);
26     digitalWrite(Pravycontrolpin1, LOW);
27     digitalWrite(Pravycontrolpin2, LOW);
28     analogWrite(LevyEnablePin, LOW);
29     analogWrite(PravyEnablePin, LOW);
```

6.3 Proměnné

V programu využívám 3 proměnné. Jedná se o proměnné *motorSpeed*, *order* a p*reviousOrder*. Jejich deklaraci můžeme vidět v ukázce kódu 3 na řádcích 12 až 14.

Proměnná *motorSpeed* je typu integer a představuje rychlost otáčení motoru, jež je určována pomocí potenciometru. Hodnoty získává pomocí funkce *analogRead()* z pinu A0, jak můžeme vidět v ukázce kódu 3 na řádce 36. Tato hodnota se musí následně vydělit čtyřmi, jelikož funkce *analogRead()* poskytuje hodnoty od 0 do 1023, avšak funkce *analogWrite()*, díky které určuji rychlost motoru, má rozmezí 0 až 255.

Proměnné *order* a *previousOrder* jsou typu char a slouží pro určování směru otáčení motorů. Proměnná *order* získává hodnoty pomocí funkce *read()*. Tyto hodnoty jsou získávány pomocí Bluetooth z mobilní aplikace. Jelikož jsem potřebovával porovnávat změnu v proměnné *order*, deklaroval jsem ještě proměnou *previousOrder*, která slouží právě k tomuto účelu.

Ukázka kódu 3 Práce s proměnnými

```
12  int motorSpeed = 0;
13  char order = '0';
14  char previousOrder = '0';

26  motorSpeed = analogRead(potPin) / 4;
```

Zdroj: Vlastní

6.4 Bluetooth komunikace

Jelikož Arduino komunikuje s telefonním zařízením pomocí technologie Bluetooth, bylo nejdříve nutné tuto komunikaci zahájit. O to se postará funkce *begin()*, jejíž využití můžeme vidět v ukázce kódu 1 na řádce 30.

Této funkci se zadává jediná hodnota. Hodnotou je baud rate, což je jednotka přenosové rychlosti, kdy jeden baud se rovná jednomu bitu za sekundu. Jak můžeme vidět, v mé aplikaci probíhá výměna 9600 bitů za sekundu. Následně pak ve funkci *loop()*, čteme pomocí funkce *read()* sériová data přicházející právě přes Bluetooth, to vidíme v ukázce kódu 1 na řádce 35.

6.5 Řízení motorů

Řízení motorů se provádí ve funkci *loop()*. Poté, co se uloží hodnota získaná přes Bluetooth z mobilního zařízení do proměnné *order*, se pomocí funkce *switch* vyhodnotí, zdali se jedná o jeden z deklarovaných rozkazů. Pokud ano, následuje funkce *if()*. Ta zjišťuje, zdali došlo ke změně v proměnné *order*. Porovná hodnoty v proměnných *order* a *previousOrder* a v případě, že se tyto dvě hodnoty rovnají, následně se pomocí funkce *digitalWrite()* uzavřou všechny tranzistory v H-můstcích a tím se zastaví motory. V opačném případě, kdy si nejsou tyto dvě hodnoty rovny, otevřou se příslušné tranzistory a nastaví se tak směr otáčení motorů. Následně se pomocí funkce *analogWrite()*, kde je jedním z parametrů proměnná *motorSpeed*, nastaví rychlost otáčení motoru podle této proměnné. Tento proces, pro případ že bude hodnota proměnné 1, můžeme vidět v ukázce kódu 4.

Ukázka kódu 4 Ukázka řízení motorů

```
37      switch (order)
38      {
39      case '1':
40            if (order != previousOrder)
41            {
42                  digitalWrite(Levycontrolpin1, HIGH);
43                  digitalWrite(Levycontrolpin2, LOW);
44                  digitalWrite(Pravycontrolpin1, HIGH);
45                  digitalWrite(Pravycontrolpin2, LOW);
46                  analogWrite(LevyEnablePin, motorSpeed);
47                  analogWrite(PravyEnablePin, motorSpeed);
48                  previousOrder = order;
49            }
50            else
51            {
52                  digitalWrite(Levycontrolpin1, LOW);
53                  digitalWrite(Levycontrolpin2, LOW);
54                  digitalWrite(Pravycontrolpin1, LOW);
55                  digitalWrite(Pravycontrolpin2, LOW);
56                  analogWrite(LevyEnablePin, LOW);
57                  analogWrite(PravyEnablePin, LOW);
58                  previousOrder = 0;
59            }
60            break;
```

Zdroj: Vlastní

7 Aplikace pro Windows Phone

Aplikace pro Windows Phone je napsána ve vývojovém prostředí Visual Studio. Má dvě vrstvy. První vrstvu, kterou vidí uživatel, představuje Uživatelské rozhraní. Druhá je Aplikační vrstva, kde se zpracovávají příkazy, které zadal uživatel pomocí Uživatelského rozhraní.

7.1 Schopnosti

Jelikož aplikace využívá technologii Bluetooth, bylo nutno nejdříve povolit RFCOMM komunikaci. To se dělá v souboru Pacakge.appmanifest. Jedná se o XML soubor, který je společný pro celou aplikaci a definuje například, jaké hardwarové požadavky aplikace bude mít, v jakém je jazyce, zdali je podporováno otáčení obrazovky, a spoustu dalších informací.

Mne ovšem zajímaly Capabilites, překládané jako schopnosti. Jedná se o vlastnosti, které musí dané telefonní zařízení splňovat. V mém případě se jednalo o schopnosti Proximity a Networking. Schopnost Proximity odkazuje na skupinu tříd, obsažených ve Windows Runtime, které umožňují Near field communication. Zatímco schopnost Networking se využívá, když potřebujeme zajistit datové spojení. Schopnost Proximity můžeme přidat do aplikace dvěma způsoby. Buď v designéru appmanifestu v záložce Capabilites zaškrtneme schopnost Proximity, nebo ji vypíšeme přímo do kódu appmanifestu. Na druhou stranu schopnost Networking od Windows Phone 8.1 jde přidat pouze přímým vepsáním do kódu. Jak můžeme vidět v ukázce kódu 5, u schopnosti Proximity stačilo přidat název schopnosti, co chceme přidat. Avšak u schopnosti Networking jsme museli nastavit alespoň tři parametry. Prvním parametrem je název. Zde máme na výběr dvě možnosti, buď bluetooth.rfcomm pro komunikaci se standardními Bluetooth zařízeními, nebo bluetooth.genericAttributeProfile pro komunikaci se zařízeními, která využívají Bluetooth LE(low energy), jenž se oproti standardním Bluetooth zařízením vyznačují nižší spotřebou elektrické energie. Dalším parametrem je Id. To nám specifikuje výrobce nebo výrobek využívající Bluetooth, případně jde výrobek specifikovat obecně pomocí slova *any*. Posledním parametrem je funkce, která nám udává službu, jakou bude dané zařízení využívat.

Ukázka kódu 5 Vložení schopností

```
<Capabilities>
  <DeviceCapability Name="proximity" />
  <m2:DeviceCapability Name="bluetooth.rfcomm">
    <m2:Device Id="any">
      <m2:Function Type="name:serialPort" />
    </m2:Device>
  </m2:DeviceCapability>
</Capabilities>
```

Zdroj: Vlastni

7.2 Uživatelské rozhraní

Uživatelské rozhraní slouží ke komunikaci s uživatelem. Je to ta část aplikace, kterou uživatel vidí a pracuje s ní. Je psána ve značkovacím jazyce XAML.

Hlavní část uživatelského rozhraní, jak můžeme vidět v příloze 1, tvoří čtveřice šipek. Pod těmito šipkami se nachází text, který nám říká, jaký je stav připojení Bluetooth. Mohou nastat následující stavy Connected, Connecting a Disconnected. Na spodní straně obrazovky se nachází Command bar, který po rozkliknutí obsahuje funkce Connect a Disconnect.

Tyto čtyři šipky představují tlačítka, kterými bude uživatel určovat směr jízdy robota. Po zmáčknutí šipky aplikace vyšle zprávu Arduinu, jaké tranzistory otevřít, a tím pádem jakým směrem má robot jet. Poté se robot rozjede daným směrem až do chvíle, kdy uživatel opět zmáčkne stejnou šipku, čímž robota zastaví. Nebo máčkne na jinou šipku a tím změní směr, kterým robot jede. V ukázce kódu 6 můžeme vidět deklarování jednoho z tlačítek. Jedná se konkrétně o horní šipku, která zadá Arduinu, aby zapnulo motory směrem vpřed. Jak můžeme vidět, element button má několik parametrů. Prvním parametrem je *Name,* neboli název elementu. Dalším parametrem je *Margin*, ten nám určuje vzdálenost od okraje obrazovky. Vzdálenost je udávána v pixelech, tedy jako absolutní vzdálenost. Parametry *Height* a *Width* nám nastavují šířku a výšku tlačítka. Dalším parametrem je *BorderThickness*, který nám nastavuje šířku ohraničení tlačítka. Posledním parametrem je *Click*. Tento parametr zaručuje, že po kliknutí na dané tlačítko se vyvolá událost, v mém kódu pojmenovaná *Forward_Click*. Toto není kompletní výčet všech parametrů, které může element button

mít. Je jich mnohem více, avšak já zde popsal jen ty, co jsem využíval ve své aplikaci. Následně tlačítko obsahuje element *Canvas*. Jedná se o panel, který umožňuje umístit tvary, jako například čtverec, trojúhelník nebo kruh. Tyto objekty se v panelu umisťují pomocí souřadnic, které určují polohu v panelu *Canvas*. Kód pro tyto obrazce jsem nevytvářel já, ale využil jsem k tomu aplikaci Microsoft Expression Design. Ta umožňuje namalovat různé obrazce, ke kterým vygeneruje XAML kód, a tím usnadní práci designéra.

Ukázka kódu 6 Nastavení tlačítka Forward

```
12      <Button x:Name="Forward" Margin="75,0,0,379" Height="162" Width="250"
13              BorderThickness="0" Click="Forward_Click">
14          <Canvas xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
15                  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
16                  x:Name="Layer_1"  Canvas.Left="0" Canvas.Top="0">
17              <Rectangle Width="125" Height="100" Stretch="Fill"
18                         StrokeThickness="13.3333" StrokeLineJoin="Round"
19                         Stroke="White" Fill="White" Margin="126,185,127,373"
20                         Canvas.Left="-190" Canvas.Top="-210"/>
21              <Path Width="250" Height="100" Stretch="Fill"
22                    StrokeThickness="13.3333" StrokeLineJoin="Round"
23                    Stroke="White" Fill="white"
24                    Data="F1 M 512,212.5L 396.024,11.5411L 279.976,212.459L
25                    512,212.5 Z " Canvas.Left="-127.574" Canvas.Top="-70.645"
26                    UseLayoutRounding="False"/>
27          </Canvas>
28      </Button>
```

Zdroj: Vlastní

Dalším elementem je TextBlock. Jeho funkcí je zobrazovat uživateli aktuální stav připojení k Arduinu, jak můžeme vidět v ukázce kódu 7. Obsahuje taktéž elementy *Name*, *Margin*, *Width* a *Height*. Avšak na rozdíl od tlačítka element TextBlock obsahuje navíc *TextWrapping*, *Text* a *FontSize*. Element *TextWrapping* nám určuje, zda v případě, že text už se nevejde do bloku, se blok rozšíří o další řádku, či nikoli. V mém případě je nastaveno *Wrap*, což znamená, že se blok v případě potřeby rozšíří. Dalším parametrem je *Text*, který nám udává, jaký bude obsah bloku po zapnutí aplikace. V mém případě se bude po zapnutí aplikace zobrazovat text Disconnected. Posledním parametrem je pak *FontSize*, jenž nám udává velikost písma v pixelech.

Ukázka kódu 7 Deklarace elementu TextBlock

```
47      <TextBlock x:Name="Status" HorizontalAlignment="Left" Margin="100,550,0,0"
48                 TextWrapping="Wrap" Text="Disconnected" VerticalAlignment="Top"
49                 Height="30" Width="200" FontSize="25" TextAlignment="Center"/>
```

Zdroj: vlastní

Dalším elementem je CommandBar, jehož deklaraci můžeme vidět v ukázce kódu 8. CommandBar má dva základní prvky, a to *PrimaryCommand* a *SecondaryCommands*. První prvek *PrimaryCommand* je určen k deklaraci tlačítek v podobě ikon. Tato tlačítka jsou vidět na rozhraní i bez nutnosti rozvinutí *CommandBar*. Jsou vidět pouze ikony a ne jejich název, ten se zobrazí až po rozvinutí menu. Tento typ tlačítek jsem se rozhodl nevyužít, jelikož mi přišly jako příliš rušivý element. Druhým typem je *SecondaryCommands*. V tomto prvku se deklarují tlačítka, která budou vidět až po rozvinutí menu. Avšak pouze jako text, ikona zde chybí. Tato tlačítka jsem využil dvě. První slouží pro připojení k Arduinu pomocí Bluetooth, druhé slouží pro odpojení. Obě tato tlačítka mají stejné parametry. Prvním parametrem je *x:Uid*. Tento parametr specifikuje, že jde o tlačítka obsažená v prvku *SecondaryCommands*. Druhým parametrem je *x:Name,* jenž udává název tlačítka. Třetím parametrem je *Label*, udávající obsah tlačítka, tedy co bude napsáno v rolovacím menu. Posledním parametrem je *Click*, ten zaručuje, že po kliknutí na tlačítko se spustí nějaká akce, která je definována v aplikační vrstvě.

Ukázka kódu 8 Deklarace elementu CommandBar

```
53    <Page.BottomAppBar>
54        <CommandBar>
55            <CommandBar.SecondaryCommands>
56                <AppBarButton x:Uid="SecondaryButton1" x:Name="Con"
57                              Label="Connect" Click="Connect_Click_1" />
58                <AppBarButton x:Uid="SecondaryButton2" x:Name="Dis"
59                              Label="Disconnect" Click="Disconnect_Click_1" />
60            </CommandBar.SecondaryCommands>
61        </CommandBar>
62    </Page.BottomAppBar>
```

Zdroj: Vlastní

7.3 Zdrojový kód

Pro celou aplikaci jsem deklaroval dvě proměnné, a to *socket* a *writer*. Proměnná *socket* je datového typu *StreamSocket* a představuje kanál, po kterém aplikace komunikuje pomocí RFCOMM a TCP protokolů. Proměnná *writer* je datového typu *DataWriter* a slouží k posílání dat přes *socket*.

7.3.1 Propojení

Propojení s Arduinem pomocí Bluetooth obstarává metoda *Connect()*, jejíž deklaraci můžeme vidět v ukázce kódu 9. Nejdříve pomocí metody PeerFinder.AlternateIdentities, která vyžaduje schopnost Proximity. Této metodě zadáme, aby vyhledala všechna spárovaná Bluetooth zařízení. Avšak problémem je, že tato metoda nám neposkytne vlastnost *ServiceName*, což je port, na kterém je povolena Bluetooth komunikace.

Následně pomocí metody *FindAllPeersAsync()* uložím všechna spárovaná zařízení do proměnné *pairedDevices*. Poté zkontroluji, zdali metoda našla nějaké zařízení. V případě že ano, vyberu první spárované zařízení. Potom inicializuji proměnnou *socket*, která slouží jako komunikační kanál mezi telefonním zařízením a Arduinem.

Dalším krokem je, že změním v uživatelském rozhraní obsah *TextBlocku* na Connecting, aby uživatel věděl, že se aplikace pokouší připojit k Arduinu. Následně se pak pomocí metody *ConnectAsync()* pokusím připojit k Arduinu. Metoda *ConnectAsync()* má dva parametry. Prvním parametrem je *HostName*, který obsahuje název nebo IP adresu zařízení, ke kterému se chci připojit. Tento parametr získáme jako vlastnost z vybraného zařízení. Parametr *ServiceName*, který představuje port, na kterém je povolena Bluetooth komunikace, ovšem získat nejde, ten musíme zadat sami. Porty určené pro Bluetooth komunikaci jsou 1 až 30, já si vybral port 1. Následně jsem inicializoval proměnnou *writer*, která bude sloužit pro posílání dat Arduinu. Ta má jeden parametr, a to kanál, po jakém bude data posílat, v mém případě se jedná o proměnnou *socket* s vlastností *OutputStream*. Tato vlastnost nastavuje, že půjde o proměnnou, která bude data posílat a ne přijímat. Poslední činností, kterou provedu v metodě *Connect()*, bude nastavení obsahu *TextBlocku* v uživatelském rozhraní na

hodnotu Connected, aby uživatel věděl, že se podařilo propojit telefonní zařízení s Arduinem.

Ukázka kódu 9 Metoda Connect

```
40  private async void Connect()
41          {
42            // Configure PeerFinder to search for all paired devices.
43            PeerFinder.AlternateIdentities["Bluetooth:Paired"] = "";
44            var pairedDevices = await PeerFinder.FindAllPeersAsync();
45            if (pairedDevices.Count == 0)
46            {
47                Debug.WriteLine("No paired devices were found.");
48            }
49            else
50            {
51                PeerInformation selectedDevice = pairedDevices[0];
52                socket = new StreamSocket();
53                Status.Text = "Connecting";
54                await socket.ConnectAsync(selectedDevice.HostName, "1");
55                writer = new DataWriter(socket.OutputStream);
56                Status.Text = "Connected";
57            }
58        }
```

Zdroj: Vlastní

7.4 Odpojení

Odpojení od Arduina se provádí pomocí metody *Disconnected()*, jejíž deklarace je vidět v ukázce kódu 10. Nejdříve pomocí metody *if()* zjistím, zdali je inicializována proměnná *writer*. Pokud ano, tak pomocí metody *DetachStream()* vypnu posílání dat přes proměnnou *socket* a následně zruším inicializaci proměnné *writer*. Poté u proměnné *socket* zjistím, zdali byla inicializována, a pokud ano, uzavřu kanál pomocí metody *Dispose()* a tak zruším komunikaci mezi telefonním zařízením a Arduinem. Následně zrušíme inicializaci proměnné *socket*. Posledním krokem je, že nastavím hodnotu v elementu *TextBlock*, nacházející se v uživatelském rozhraní, na hodnotu Disconnected, aby informoval uživatele, že odpojení proběhlo úspěšně.

Ukázka kódu 10 metoda Disconnect

```
73          private void Disconnect()
74          {
75              if (writer != null)
76              {
77                  writer.DetachStream();
78                  writer = null;
79              }
80              if (socket != null)
81              {
82                  socket.Dispose();
83                  socket = null;
84              }
85              Status.Text = "Disconnected";
86          }
```

Zdroj: Vlastní

7.5 Posílání dat

Posílání dat přes technologii Bluetooth zajišťuje metoda *SendMessegeAsync()*, jejíž deklaraci můžeme vidět v ukázce kódu 17. Jak můžeme vidět, má jeden parametr, jímž je proměnná *message*, která obsahuje zprávu, jež bude poslána přes Bluetooth Arduinu. Prvním krokem je deklarování proměnné *sentMessageSize*. Následně zjistím, zdali byla inicializována proměnná *writer*. Pokud ano, do proměnné *messageSize* uložím počet znaků v odesílané zprávě. Následně pomocí metody *WriteByte()* definuji, kolik bajtů zabere daná zpráva. Poté pomocí metody *WriteString()* uložím do proměnné *sentMessageSize* zprávu v podobě binárního kódu. Posledním krokem je odeslání zprávy Arduinu.

Ukázka kódu 11 metoda SendMessageAsync

```
60          private async Task<uint> SendMessageAsync(string message)
61          {
62              uint sentMessageSize = 0;
63              if (writer != null)
64              {
65                  uint messageSize = writer.MeasureString(message);
66                  writer.WriteByte((byte)messageSize);
67                  sentMessageSize = writer.WriteString(message);
68                  await writer.StoreAsync();
69              }
70              return sentMessageSize;
71          }
```

Zdroj: Vlastní

8 Závěr

Ovládání robota pomocí mobilního zařízení je velmi jednoduché a intuitivní. Toho bylo dosaženo minimálním počtem ovládacích prvků. V případě řízení směru jízdy je pomocí velkých šipek ihned vidět, pro jaký směr je dané tlačítko určené.

Nejen v aplikaci pro mobilní zařízení, ale i na samotném robotu vidím několik možností, které by mohly vést ke zlepšení. U mobilního zařízení se jedná především o možnost vybrat, k jakému spárovanému zařízení se chceme připojit, upravit vzhled aplikace, aby ji bylo možné spustit i na jiném zařízení, nebo například přidat možnost řídit robota pomocí zabudovaného gyroskopu.

Na straně robota vidím možnosti ke zlepšení například ve využití silnějších motorů, sestavení kompaktnějšího podvozku nebo přidaní sady senzorů, jako je ultrazvukový měřič vzdálenosti či kamera. Rozhodně by nebylo na škodu i přidání výstupních zařízení, jako je například LCD displej. Další možností je komunikace mezi Arduinem a mobilním zařízením, jelikož technologie Bluetooth ve verzi, kterou využívám, umožňuje jen komunikaci na malé vzdálenosti a menší přenos dat. Proto by jí bylo vhodné vyměnit buď za novější verzi Bluetooth, nebo za úplně jinou technologii, jako je například Wifi.

Přes veškerá možná vylepšení stávajícího systému se mi podařilo zadání splnit.

I. Summary and keywords

The topic of this thesis is programming of a single boarded computer called Arduino UNO. The goal is to program controlling software for this board with the usage of digital and analog accessories and to assemble an electrical circle. The theoretical part of the thesis deals with the theory of single boarded computers, wirless comunnication especially Bluetooth technology, programming language C# and integrated development environment Visual Studio. The main goal of the practical part is to create an electrical circle which contains the single boarded computer, two step motors, and a wireless Bluetooth. Another goal is to create an application for a smartphone (Windows Phone 8.1) which could wirelessly control step motors via Bluetooth. Windows Phone aplication is written in C# language. Arduino program is written in Arduino language bases on C language.

Key words: Arduino Uno, step motor, Visual C#, Windows Phone 8.1, electrical circuit

II. Seznam použitých zdrojů

App activation and deactivation for Windows Phone 8 (© 2017). *Msdn.microsoft.com* [online]. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/apps/ff817008(v=vs.105).aspx

Arduino - ArduinoBoardLeonardo (© 2017). *Arduino* [online]. Dostupné z: https://www.arduino.cc/en/Main/ArduinoBoardLeonardo

Arduino - ArduinoBoardLilyPad (© 2017). *Arduino* [online]. Dostupné z: https://www.arduino.cc/en/Main/ArduinoBoardLilyPad

Arduino - ArduinoBoardMega2560 (© 2017). *Arduino.* [online]. Dostupné z: https://www.arduino.cc/en/Main/ArduinoBoardMega2560

Arduino - ArduinoBoardNano (© 2017). *Arduino* [online]. Dostupné z: https://www.arduino.cc/en/Main/ArduinoBoardNano

Arduino - ArduinoBoardUno (© 2017). *Arduino.* [online]. Dostupné z: https://www.arduino.cc/en/Main/ArduinoBoardUno

Arduino - ArduinoBoardZero (© 2017). *Arduino* [online]. Dostupné z: https://www.arduino.cc/en/Main/ArduinoBoardzero

Arduino - Counterfeit (© 2017). *Arduino* [online]. Dostupné z: https://www.arduino.cc/en/products/counterfeit

DOČEKAL, David (2014). *BLUETOOTH VIBRAČNÍ VYZVÁNĚNÍ*. Brno. BAKALÁŘSKÁ PRÁCE. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ. Vedoucí práce Ing. ALEŠ POVALAČ Ph.D.

From Photon to the future: A not-so-brief history of Windows Phone (2016). *Neowin* [online]. Dostupné z: https://www.neowin.net/news/from-photon-to-the-future-a-not-so-brief-history-of-windows-phone

HOLUŠA, Václav (2014). *Vývoj aplikací pro Windows Phone 8 a portace na Windows 8*. Brno. DIPLOMOVÁ PRÁCE. MASARYKOVA UNIVERZITA. Vedoucí práce Ing. RNDr. Barbora Bühnová, Ph.D.

JÄPPINEN, Pekka (2001). *Bluetooth wireless technology based guidance system*. Lappeenranta. DIPLOMOVÁ PRÁCE. LAPPEENRANTA UNIVERSITY OF TECHNOLOGY. Vedoucí práce Professor Jari Porras.

MONK, Simon (2012). Programming Arduino getting started with sketches. New York: McGraw.

Nokia Lumia 520 - Full phone specifications (© 2000-2017). *GSMARENA* [online]. Dostupné z: http://www.gsmarena.com/nokia_lumia_520-5322.php

Nokia Lumia 520: okna s nejnižší cenovkou [recenze] - Mobil (2013). *Mobilmania* [online]. Dostupné z: http://www.mobilmania.cz/clanky/nokia-lumia-520-okna-s-nejnizsi-cenovkou-recenze/sc-3-a-1324226

PWM | Android Things (n. d.). Android Things [online]. Dostupné z: https://developer.android.com/things/hardware/index.html

TROELSEN, Andrew and Philip JAPIKSE (2015). *C# 6.0 and the .NET 4.6 Framework*. Seventh Edition. New York, USA: Apress.

Types of Wireless Communication and Its Applications (n.d.). *Elprocus: Electronic|Projects|Focus* [online]. Dostupné z: https://www.elprocus.com/types-of-wireless-communication-applications/

XAML Overview (WPF) (© 2017). *Msdn.microsoft* [online]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms752059%28v=vs.110%29.aspx

Windows Phone 8.1: Die neuen Funktionen in Bildern - areamobile.de (2014). areamobile.de [online]. Dostupné z: http://www.areamobile.de/b/2432-windows-phone-8-1-die-neuen-funktionen-in-bildern

Windows SDK pro Windows 8.1 – vývoj aplikací pro Windows (2015). *Developer.microsoft.com* [online]. Dostupné z: https://developer.microsoft.com/cs-cz/windows/downloads/windows-8-1-sdk

Welcome to Visual Studio 2015 (© 2017). *Msdn.microsoft* [online]. Dostupné z: https://msdn.microsoft.com/library/dd831853(v=vs.140).aspx

Základy technologie Bluetooth: původ a rozsah funkcí | PC World.cz (2009). *PC World* [online]. Dostupné z: http://pcworld.cz/hardware/Zaklady-technologie-Bluetooth-puvod-a-rozsah-funkci-6635

III. Seznam použitých tabulek, obrázků a ukázek kódů

Seznam obrázků

Seznam tabulek

Seznam ukázek kódů

IV. Seznam příloh

Příloha č. 1: Uživatelské rozhraní

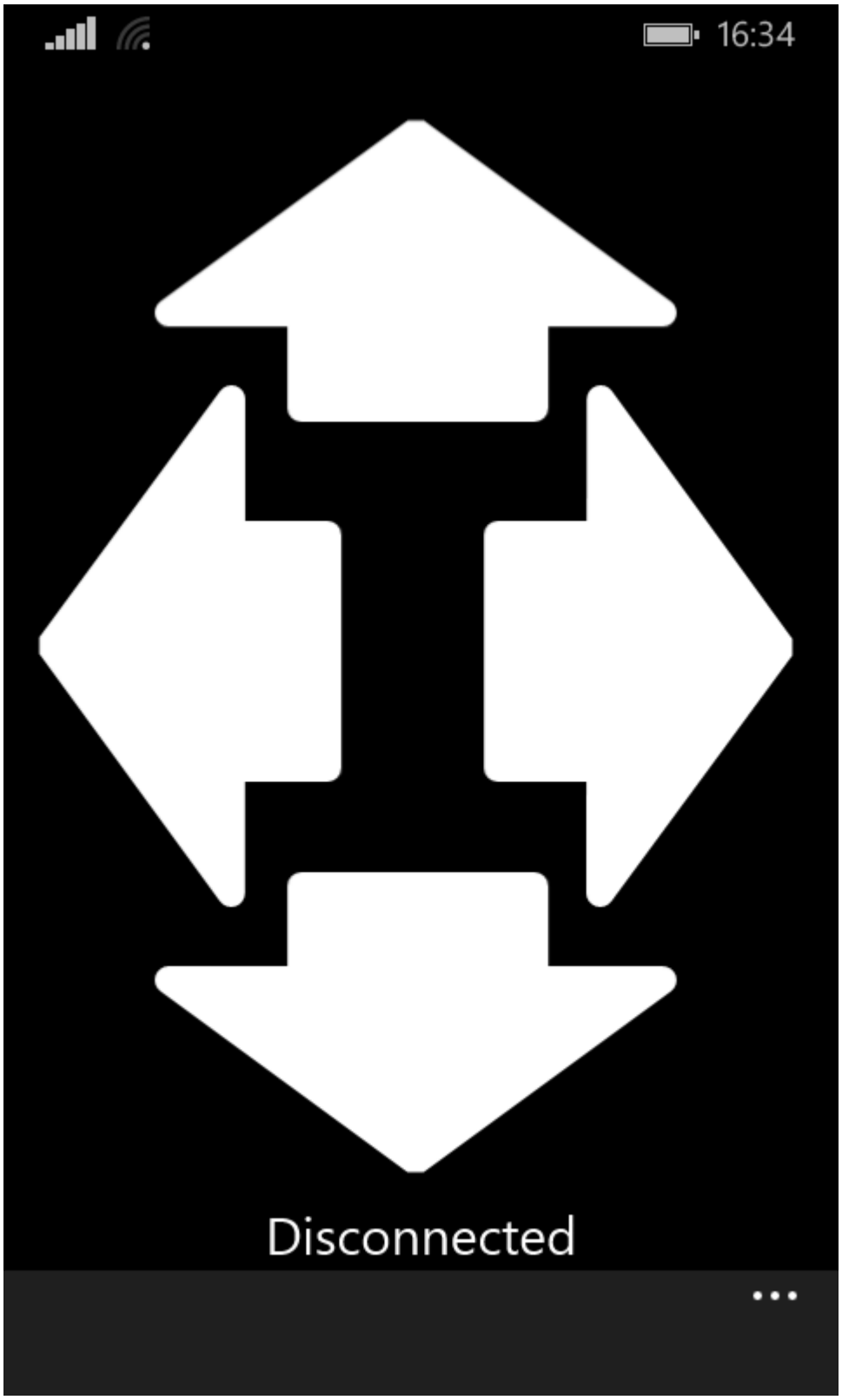
Příloha č. 2: Převodová skříň

Příloha č. 3: Celkový vzhled robota

Příloha č. 4: Obsah přiloženého CD

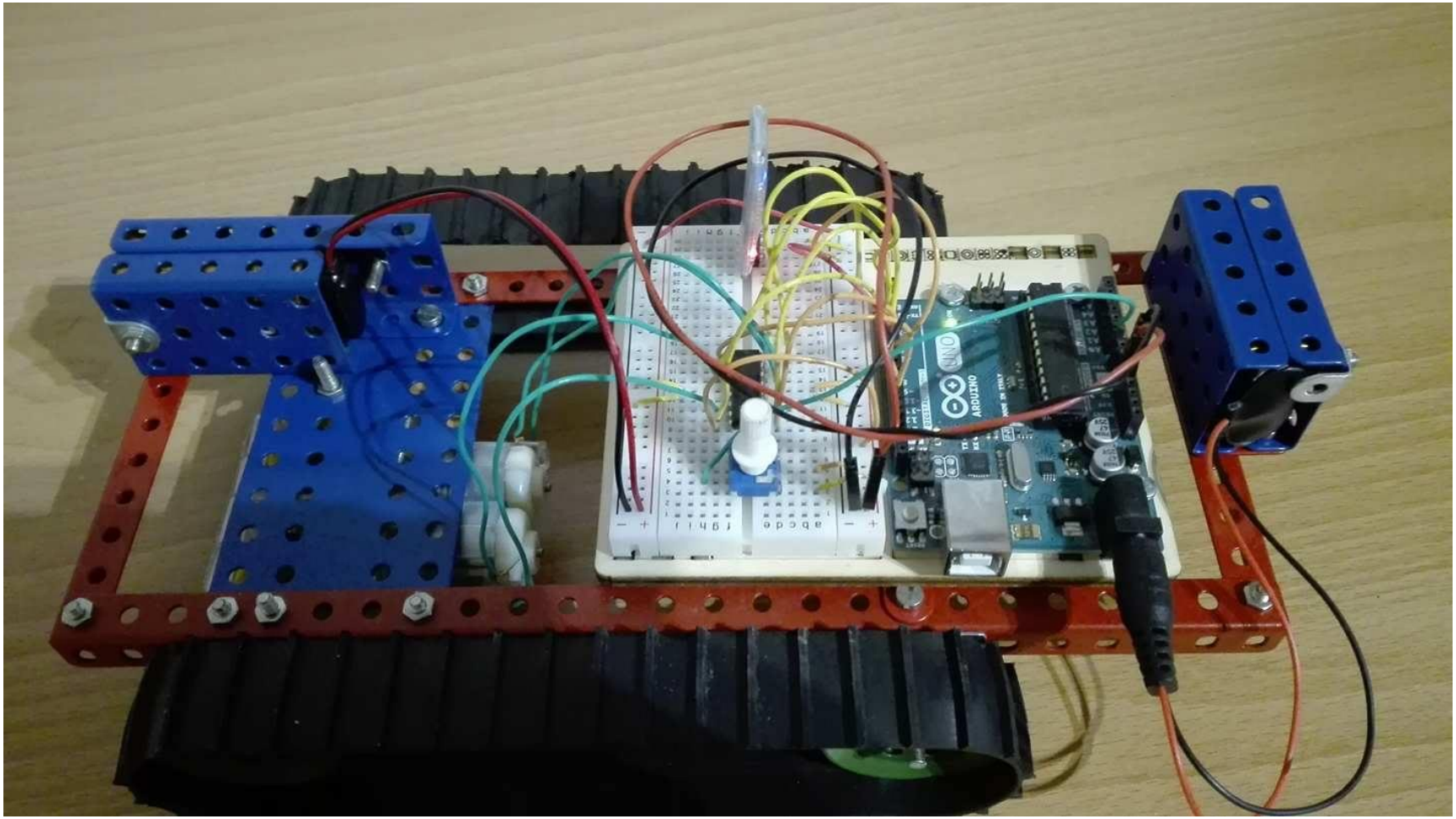
V. Přílohy

Příloha č. 1: Uživatelské rozhraní



Příloha č. 2: Převodová skříň

Příloha č. 3: Celkový vzhled robota



Příloha č. 4: Obsah přiloženého CD

Arduino_control.ino – Zdrojový kód pro Arduino

Arduino_control.zip - Aplikace pro Windows Phone

celkový_vzhled.jpg – Obrázek celkového vzhledu robota

detail_zapojení.jpg – Obrázek zapojení Arduina

napájení.jpg – Obrázek napájení robota

profil.jpg – Obrázek robota ze strany

předek.jpg - Obrázek robota zepředu

převodová_skříň.jpg – Obrázek převodové skříně

zadek.jpg – Obrázek robota zezadu

uzivatelske_rozhrani.png – Screenshot aplikace pro Windows Phone

Programování-jednodeskového-počítače.pdf – Tato bakalářská práce ve formátu pdf

Obsah.pdf – Obsah tohoto CD