Výukový model robota s ovládáním pomocí mobilní aplikace

Bc. Veronika Cichrová

Bakalářská práce
2019

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2018/2019

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Veronika Cichrová**
Osobní číslo:	**A17816**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**kombinovaná**
Téma práce:	**Výukový model robota s ovládáním pomocí mobilní aplikace**
Téma anglicky:	**An Educational Model of a Mobile Robot Controlled by a Mobile Application**

Zásady pro vypracování:

1. **Proveďte návrh modelu mobilního robota s ohledem na použití jako výukové pomůcky programování mikropočítačů.**
2. **Návrh hardwarově realizujte s využitím platformy Arduino**
3. **Zvolte a osaďte vhodné snímače k získávání informací z okolí.**
4. **Implementujte software pro řídicí mikropočítač pro získávání dat ze snímačů a pro řízení robota.**
5. **Vytvořte aplikaci pro mobilní telefon, která umožní zobrazovat údaje ze snímačů a ovládat robota.**

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **CATSOULIS, John. Designing embedded hardware. 2nd ed. Sebastopol, CA: O'Reilly, 2005, xvi, 377 p. ISBN 0596007558.**
2. **GRIFFITHS, Dawn a David GRIFFITHS. Head first Android development. Sebastopol: O'Reilly, [2015]. Head first series. ISBN 1449362184.**
3. **MARGOLIS, Michael. Arduino cookbook. 2nd ed. Sebastopol, Calif.: O'Reilly, 2012, xx, 699 p. ISBN 1449313876.**
4. **NOVÁK, Petr. Mobilní roboty: pohony, senzory, řízení. Praha: BEN – technická literatura, 2005. ISBN 80-7300-141-1.**
5. **PINKER, Jiří. Mikroprocesory a mikropočítače. 1. vyd. Praha: BEN – technická literatura, 2004, 159 s. ISBN 80-7300-110-1.**

Vedoucí diplomové práce: **Ing. Jan Dolinay, Ph.D.**
Ústav automatizace a řídicí techniky

Datum zadání diplomové práce: **3. prosince 2018**

Termín odevzdání diplomové práce: **15. května 2019**

Ve Zlíně dne 7. prosince 2018

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Mgr. Roman Jašek, Ph.D.
garant oboru

Prohlašuji, že

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

Prohlašuji,

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

Ve Zlíně, dne

Veronika Cichrová, v. r.
podpis diplomanta

ABSTRAKT

Cílem této práce je návrh a realizace modelu mobilního robota, ovládaného pomocí mobilního zařízení s operačním systémem Android prostřednictvím rozhraní Bluetooth. Robot je cílen jako výukový model s ohledem na motivaci studentů ke studiu vývoje embedded zařízení.

Model robota je postaven na základě elektronické vývojové desky Arduino s moduly LED diod, bzučáku a ultrazvukovým senzorem. Programové vybavení je vytvořeno ve vývojovém prostředí Arduino IDE. Robot bude ovládán mobilním zařízením s operačním systémem Andorid aplikací vytvořenou v Android Studiu. Aplikace bude umožňuje zobrazovat data naměřená ultrazvukovým senzorem a ovládat robota softwarovým joystickem.

Výsledkem práce je mobilní robot včetně programového vybavení pro řídicí mikropočítač a mobilní aplikace pro dálkové ovládání tohoto robota určená pro operační systém Android verze 4.0 a vyšší.

Klíčová slova: Android, Arduino, Mikropočitač, Embedded

ABSTRACT

The aim of this diploma thesis is to design and implement a mobile robot model controlled by a mobile device with Android operating system via Bluetooth. This mobile robot is meant to be used as an educational model. This model should motivate students to study in the field of embeded systems developement.

The robot model is based on the arduino electronic development board with a few secondary devices like a LED modules, buzzer and an ultrasonic sensor. The software is created in the Arduiono IDE development environment. The robot is controlled by a native mobile aplication running on Androind device created in the Android Studio. Application displays data measured with an ultrasonic sensor and a joystick that is used to control movement of the robot.

The outcome of this diploma thesis is a mobil robot including software for control microcomputer and mobile application used for a remote control of the robot. The mobile application is designed for operating system Android version 4.0 and hiegher.

Keywords: Android, Arduino, Microcomputer, Embedded

Chtěla bych poděkovat panu Ing. Janu Dolinayovi, Ph.D. za veškerý věnovaný čas a za mnoho cenných rad. Dále bych chtěla poděkovat rodině, za podporu po celou dobu studií.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Výuka programování patří dnes již k běžnému učivu na základních a středních školách. Při výuce je možné použití nejrůznějších metod a programovacích jazyků. K tomu, aby bylo učení co nejefektivnější a studenty bavilo je možno použít různé interaktivní stavebnice, kde výsledkem není jen program na obrazovce počítače, ale objekt pohybující se v reálném světě, který např. pomocí snímačů zjišťuje informace o svém okolí. K takové výuce je možno použít různé komerční robotické stavebnice, které jsou ale často poměrně drahé anebo mají různým způsobem omezené možnosti programování.

Je také možno použít vlastní konstrukci výukového modelu, kde ale můžeme narazit na mnohé problémy s volbou vhodných komponent, jejich kompatibilitou apod.

Tato práce má za cíl navrhnout a vytvořit model mobilního robota, který by byl použitelný pro výuku programování a pro zvýšení motivace studentů k učení se programování. Model bude možno ovládat vzdáleně přes rozhraní bluetooth pomocí mobilní aplikace, takže kromě výuky programování embedded systémů (řídicího mikropočítače) může posloužit i pro výuku programování mobilních aplikací a při základech elektroniky.

Výsledným produktem praktické části tedy bude model, který prezentuje práci s Arduinem (mikropočítačem) a mobilní aplikace, určená pro operační systém Android. Robot bude mobilní na pásovém podvozku poháněném motorem a bude naslouchat příkazům přicházejících z mobilního zařízení. Do mobilního zařízení budou také odesílána data z ultrazvukového senzoru. LED diody budou znázorňovat směr jízdy a bzučák upozorní na zpětný chod motorů.

V teoretické části jsou popsány základní části mikropočítače, platforma Arduino, rozhraní Bluetooth a také tvorba aplikací pro mobilní zařízení. V praktické části je pak popsán hardware modelu a vytvořené programové vybavení.

I. TEORETICKÁ ČÁST

1 MIKROPOČÍTAČ

Jednočipový mikropočítač je mikroprocesor zapojený do integrovaného obvodu včetně paměti a vstupně výstupního rozhraní. Tento obvod zahrnuje vše potřebné pro vykonávání programu. Mikropočítače se mohou lišit typem užití. Mohou být univerzální, a tedy obsahovat všechny obvykle potřebné prvky, nebo mohou sloužit konkrétní věci a být osazeny jen nezbytnými prvky. V každém případě je možné použít stejný mikroprocesor, protože zastává funkci výpočetní jednotky a další osazení mikropočítačové desky na něj nemá vliv. Dále lze rozšiřovat mikropočítač o přídavné senzory, moduly a periferie.

Vzhledem k různým druhům dostupné literatury je možné se setkat s pojmem mikrokontroler, nejedná se však o nic jiného než mikropočítač. Někdy bývá označován mikropočítač za mikroprocesor, což je chybné. Mikropočítač je soběstačná jednotka obsahující mikroprocesor, zatím co mikroprocesor bez nutných komponent nezmůže fungovat. [5]

1.1 Architektury

Mikropočítače lze dělit podle architektury. Existují dva druhy architektur. První je Von Neumannova. Tato architektura je specifická tím, že programová a datová paměť využívají stejnou adresovou a datovou sběrnici. Což může působit problémy v přístupu, kdy v jednom okamžiku je možné přistupovat pouze k programu nebo k datům. Tím je celý proces dost zpomalen. Druhá architektura je Harvardská a funguje tak, že je operační paměť rozdělena na dvě části. Oproti Von Neumannově je tedy možné najednou přistupovat k programu i k datům. V oblasti mikropočítačů se časti používá kombinace obou. Přičemž je mikroprocesor sestaven podle Harvardské architektury, ale pro přístup externích zařízení se využívá architektura Von Neumannova. [5]

1.2 Instrukční sady

Mikropočítače se liší i obsahem instrukčních sad. Jedna ze sad se nazývá RISC – Reduced Instruction Set Computer. Jedná se o redukovanou, velmi zjednodušenou sadu. Sada obsahuje pouze základní operace, z niž jdou složitější operace sestavit. Druhá je CISC – Complex Instruction Set Computing. Tato sada je velmi komplexní, obsahuje složitější instrukce, které vyžadují více operací v procesoru. Instrukce CISC mají různou délku vykonávání oproti instrukcím ze sady RISC, které jsou obvykle vykonávány v jednom taktu procesoru. [5]

1.3 Základní prvky mikropočítače

1.3.1 Mikroprocesor

Je sekvenční číslicový obvod, tvořící hlavní část počítače. Slouží k vykonávání logických a aritmetických operací. Sekvenčně zpracovává jednotlivé instrukce kódu. [5]

1.3.2 Paměť

Paměť mikropočítače je programové uložiště dat. Obvykle se pro uložení programu používá paměť typu FLASH a pro uložení dat, paměť typu RAM.[5]

1.3.3 Sběrnice

Sběrnice je skupina vodičů. Mikropočítače využívají sběrnice dělící se na řídící, datové a adresové. Datové vodiče přenáší data. Adresové vodiče přenáší adresu zařízení a řídící vodiče slouží k synchronizaci přenášeného signálu. Sběrnice zajišťuje přenos dat mezi jedním nebo dvěma elektronickými zařízeními. [5]

1.3.4 Registry

Slouží k uložení operandů instrukcí a výsledkům operací.

1.4 Kde se mikropočítače používají

Mikropočítače jsou využívány velkou škálou zařízení. Od prostého ovládání televizoru či domácího kina, po ovládání celé chytré domácnosti. Je možné vytvořit dokonce i ovládání okenních rolet, nebo spuštění kávovaru pomocí mobilního telefonu. Zautomatizování domácnosti je velmi praktické, díky možnosti ovládání na dálku. V chytrých domácnostech je výhoda regulace termostatu na topení a klimatizaci na dálku. Pořízení těchto komerčních zařízení může být velmi nákladné, avšak za pomoci Arduina, či jiného mikropočítače, proveditelné i doma. [5]

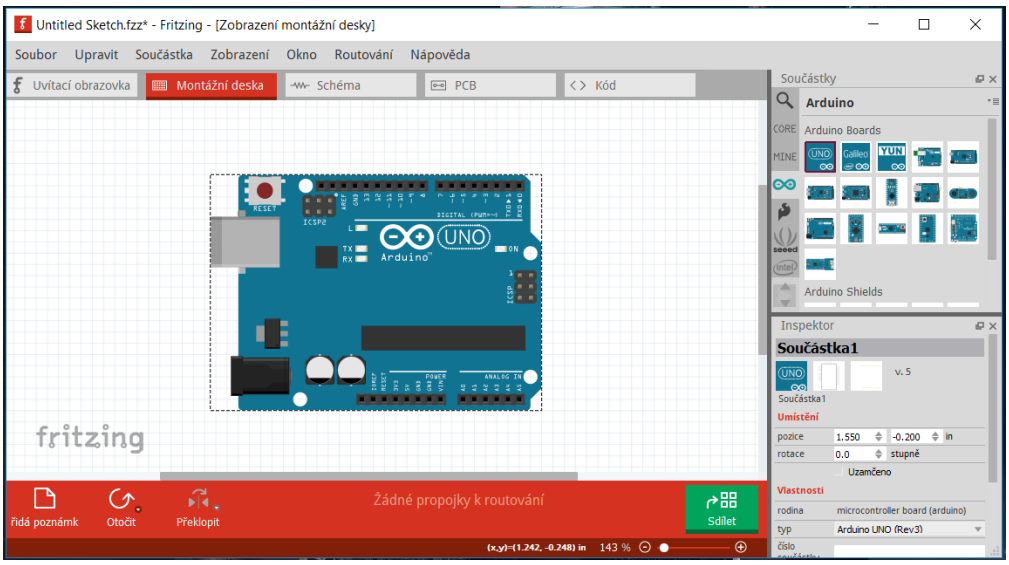
1.5 Návrh vlastního zařízení s mikropočítačem

Své vlastní zařízení s mikropočítačem si může každý navrhnout i sám. Je možné vycházet z Open source platforem, které své podrobné návrhy zveřejňují, nebo si navrhnout vše podle svých potřeb. Snadnější varianta je ta první, kde je možné použít již propracovaný návrh

a vyřadit, či vyměnit komponenty, které nejsou třeba. Výhoda vlastního návrhu oproti koupenému originálu může být třeba to, že na desce mohou být pouze komponenty, které jsou nezbytně nutné. Nebo může být cílem směstnat více funkcionalit na jednu destičku, na místo velkého množství senzorů a modulů propojeného drátky. [1]

1.5.1 Fritzing

Jedná se o open source software pro kreslení schémat zapojení. Je dobrým pomocníkem při návrhu. Tento software nabízí několik pohledů, v kterých lze tvořit. První z nich se nazývá Montážní deska. Jedná se o návrh za pomoci nepájivého kontaktní pole. Všechny součástky mají reálné poměry velikostí. Proto je možné vidět, jak konečný produkt bude vypadat. Druhou částí je schéma. Součástky z montážní desky jsou v tomto módu značeny oficiálními normalizovanými značkami. Třetí částí je deska plošného spoje. Každá součástka je vedena ve všech režimech. Software nabízí velkou spoustu součástek typu Arduino včetně senzorů a shieldů, ale i součástky jiného typu. V případě, že v základním setu knihoven, není požadovaná součástka, je možné si jí naimportovat stáhnutou z internetu. Jelikož s velkou pravděpodobností jí již někdo potřeboval. V případě nedostupnosti součástky, je možné jí vytvořit. Na tuto možnost je Fritzing připraven a disponuje speciální částí programu, sloužící právě pro návrh součástky.



Obrázek 1 Program Fritzing

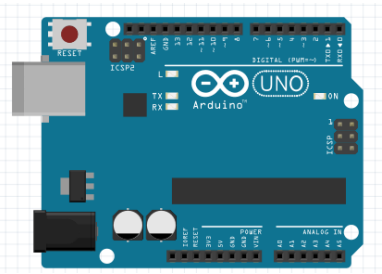
2 ARDUINO

Jedná se o open source elektronickou platformu, založenou na počítačové destičce osazené komponentami a speciálním vývojovém prostředím Arduino IDE. Arduino může být využito pro řízení mnoha zařízení. Existuje velké množství přídavných senzorů, motorů, modulů a shieldů.

2.1 Typy vývojových desek Arduino

2.1.1 Uno

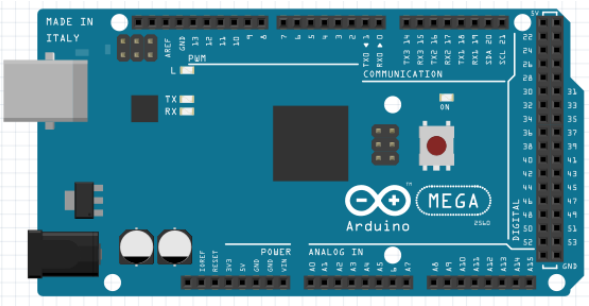
Jedná se asi o nejpoužívanější Arduino desku. Jedná se o základ, který stačí k většině projektů. Co není na samostatné desce, lze snadno dohnat shieldy, které jsou právě, pro tento druh desky vytvořeny. Více o Arduino Uno v kapitole 2.2. [10]



Obrázek 2 Arduino Uno

2.1.2 Mega

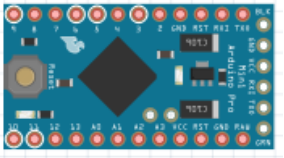
Arduino mega je vhodný pro tvorbu rozsáhlých projektů. Na desce se nachází velké množství analogových i digitálních pinů. V podstatě se jedná o rozšířené Uno. A proto, jsou Uno shieldy kompatibilní i s touto deskou. Základní rozložení je zachováno, jen se deska protáhla o desítky pinů navíc. Tato deska se vyrábí i ve výkonnější verzi. Osazení pinů je stejné, jen se hlavní čip změní z Mega Atmega 2560 na Atmel SAM3X8E. [12]



Obrázek 3 Arduino Mega

2.1.3 Mini

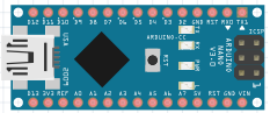
Arduino mini je absolutně nejmenší destička oficiální distribuce. Pro její naprogramování je nutné použít externí převodník, jelikož nedisponuje USB portem. Destička je vhodná do dálkových ovladačů a do projektů, kde je potřeba šetřit s místem. [12]



Obrázek 4 Arduino Mini

2.1.4 Nano

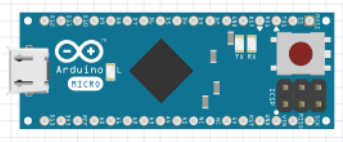
Arduino nano vypadá jako Mini na nožičkách. Nejhlavnější rozdíl však ale je, že má integrovaný převodník s USB portem a tedy, nepotřebuje externí. Výhodou mohou být i napájené kolíky, kterými jsou nahrazeny dírky z verze Mini. [12]



Obrázek 5 Arduino Nano

2.1.5 Micro

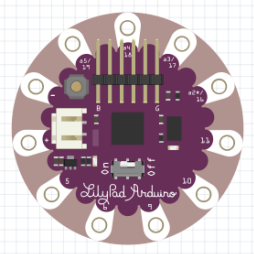
Arduino Micro je vhodné k tvorbě vlastních periferií, jako jsou myši nebo klávesnice. Má integrovaný převodník s USB portem, který se po připojení k počítači automaticky tváří jako vstupní zařízení. [12]



Obrázek 6 Arduino Micro

2.1.6 LilyPad

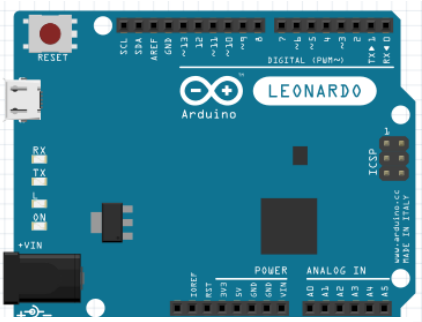
Arduino LilyPad je již od pohledu jiné. To kvůli tomu, že je určeno k nošení na textilu. Je speciálně vytvořena, aby rozblikala třeba mikinu. Existuje jich několik verzí. [12]



Obrázek 7 Arduino LilyPad

2.1.7 Leonardo

Arduino Leonardo na první pohled vypadá jako Arduino Uno, ale společného nemají absolutně nic. Funkcionalitou se podobá Arduinu micro. Čili může posloužit, jako základ pro novu periferii. Problémem může způsobovat nutnost, použití externího převodníku s USB. [12]

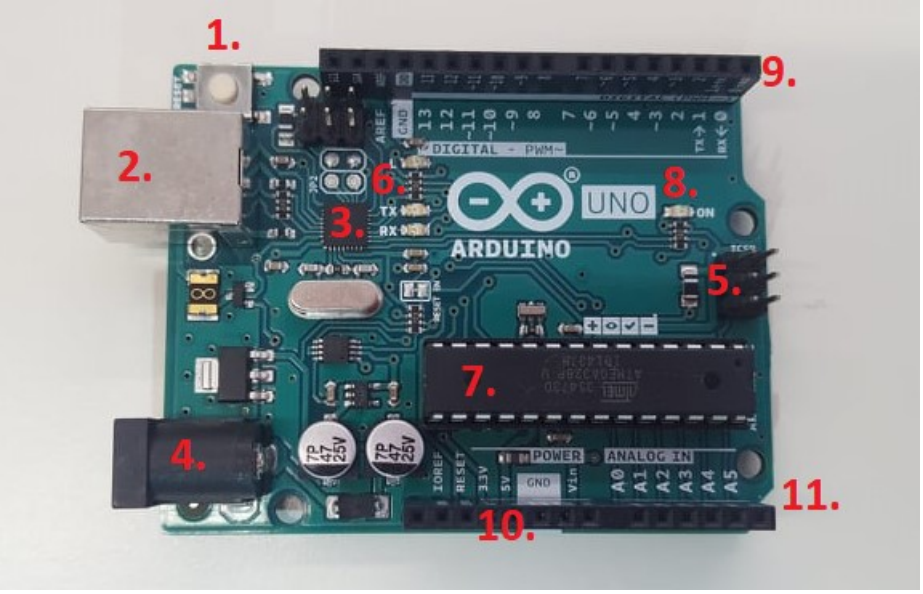


Obrázek 8 Arduino Leonardo

2.2 Arduino Uno

Arduino Uno, je deska osazená procesorem ATmega328P od firmy Atmel. Disponuje 14 vstupně výstupními piny, z kterých je až 6 možné použít, jako PWM výstupy. Dále také 6 analogovými vstupy, 16 MHz krystal, USB připojení s převodníkem a souosý napájecí konektor. [10][12]

Podrobnější popis jednotlivých komponent na desce lze vidět na následujícím obrázku s textem. [11]



Obrázek 9 Očíslované Arduino Uno k popisu

1. **Tlačítko reset** – Slouží k tomu, aby bylo možné, kdykoliv program spustit od začátku

2. **USB konektor typu B** – Hlavní funkce této součástky na desce slouží k nahrávání programu. Jedná se o snadný a rychlý způsob připojení. Není to však jediná funkce, z tohoto konektoru lze desku i napájet. Ne všechny Ardunina ho ale mají, na některých je nahrazen síťovým konektorem, bluetooth, micro USB nebo taky žádný a je potřeba externí propojení.

3. **USB převodník** – Jedná se o nezbytnou součást, která zajišťuje komunikaci mezi hlavní čipem a počítačem. Tato nezbytná věc v komunikaci je obvykle jako samostatná část, jako vidíme zde. Ale někdy jí na desce nenajdeme. To může znamenat 2 věci. První je ta, že převodník je obsažen v hlavním čipu, nebo se na desce nenachází vůbec. V tom případě je nutné použít převodník externí.

4. **Souosý napájecí konektor** – slouží výhradně k vstupnímu napájení desky. Více v kapitole Napájení.

5. **ICSP** – ICSP slouží pro externí programování hlavního mikropočítače ATmega328. Lze jej využít například v případě, že se poškodil bootloader, který normálně slouží k jeho programování.

6. **LED diody L, Rx, Tx** – Dioda se značením L je běžně využívaná, nachází se na pinu 13. Převážně se používá k vyzkoušení funkčnosti desky. Nebo je vhodné jí

rozblikat, jako první program napsaný pro začátečníky. Některé desky jí ale neobsahují. Zatím co diody značené Rx a Tx problikávají v okamžiku, kdy probíhá komunikace na sériové lince.

7. **Hlavní čip** – jedná se o mozek celého Arduina. Vše se totiž odehrává v něm. Zde můžeme vidět jednočipový mikropočítač Atmel.
8. **Indikační LED dioda napájení** – Tato dioda svítí v okamžiku, kdy je napájená
9. **Digitální piny** – Mezi klasickými digitálními piny můžeme vidět i piny značené vlnovkou. Tato vlnovka znamená, že na konkrétním pinu je možné využít PWN modulaci. Na Arduinu Uno se jedná o piny 3, 5, 6, 9, 10, 11. Dále mezi digitálními piny můžeme vidět značení Tx a Rx. Jedná se o piny, na které je možné připojit zařízení sloužící k sériové komunikaci. Např. Bluetooth.
10. **Napájecí výstupy** – Tyto výstupy je nutné využívat k napájení modulů a shieldů.
11. **Analogové piny** – Na těchto vstupech je možné měřit analogové hodnoty. V nouzi a nedostatku digitálních pinů, je možné i tyto piny využít jako digitální vstupy a výstupy.

2.3 Klony

Na trhu je k dispozici oficiální distribuce Arduino originál, nebo velké množství klonů. Klony jsou převážně přesné kopie originální řady. Výhodou originálu je 100% spolehlivost, za kterou se ale platí. Klony, převážně z Číny, mohou být méně kvalitní, nebo nemusí být 100 % kompatibilitu toho, že budou fungovat ovladače. Na druhou stanu se vyplatí riskovat, protože cena je i desetkrát menší. [13]

2.4 Napájení

Arduino lze napájet mnoha způsoby, přičemž vždy se jedná o napájení ze zdroje stejnosměrného napětí. Tyto způsoby se používají dle potřeby a možností, jaké deska nabízí. Ne každá deska disponuje USB portem nebo souosým konektorem, v tom případě je nutné vymyslet jiný způsob napájení. Ku příkladu použít napájení pomocí pinů. Také se může stát, že zvolený způsob nebude napájet dostatečně. Třeba není možné napájet dva DC motory pouze z USB portu, tedy je nutné využít jiný způsob. [13] [18]

2.4.1 Napájení pomocí souosého korektoru

Souosý konektor disponuje rozsahem napájecího napětí 6 až 15 V s maximálním proudem 1000 mA.

K tomuto typu napájení může být použit síťový adaptér, připojovací kabel na 9 V baterii či svorkový konektor, který má volné možnosti. U svorkové konektoru však vzniká problém při možnosti přepólování. Jako ochrana před přepólováním se využívá dioda. [18]

2.4.2 Napájení pomocí USB portu

USB port disponuje rozsahem napájecího napětí 4,75 až 5,25 V s maximálním proudem 500 mA.

Součástí větších Arduino desek, jako třeba UNO je USB TTL převodník. Touto variantou je deska napájena v průběhu programování a je jí možné použít i jako stálý zdroj napájení třeba z power banky. Jedná se však o velmi slabý zdroj napájení. Výhodou může být, že je možné jej kombinovat třeba s napájením ze souosého konektoru. Maximální proud je u USB portu omezen pouze na 500 mA z důvodu použití resetovatelné pojistky, která chrání počítač před chybným zapojením a zkraty. [18]

2.4.3 Napájení pomocí pinů VIN (+) a GND (-)

Kombinace konektorů VIN a GND disponují rozsahem napájecího napětí 5,8 až 14,8 V s maximálním proudem 1000 mA.

Napájení přes pin VIN je dosti podobné souosému konektoru, avšak je zde velká nevýhoda. Není zde ochrana proti přepólování, a tedy je velmi jednoduché desku zničit. Na druhou stranu výhoda použití pinů je v tom, že ochranná dioda u souosého konektoru zapříčiňuje úbytek napětí přibližně o 0,8 V, což může působit potíže s kapacitou při napájení z baterie. [18]

2.4.4 Napájení pomocí pinů 5 V (3V3) a GND

Kombinace konektorů 5 V a GND disponují rozsahem napájecího napětí 4,75 až 5,25 V s maximálním proudem 1000 mA. Tento typ napájení slouží především k výstupnímu napájení modulů a komponent. [18]

2.5 Mikropočítač ATMega328

V následující kapitole je popsán 8itový mikropočítač Atmega328 od firmy Atmel, který je využit v desce Arduino Uno, která je základem této práce. Funguje na architektuře AVR RISC, tedy se jedná o optimalizovaný procesor pouze s omezenou instrukční sadou. AVR je interní označení typů mikroprocesorů s Harvardskou architekturou. Disponuje pamětí flash, která zvládá zapisovat a číst data v jeden okamžik. Dále obsahuje 2 kB SRAM, polovodičové statické operační paměti realizované klopným obvodem. 1 kB EEPROM, elektronicky mazatelné paměti. Data v ní jsou uchována i při odpojení napájení. 32 registrů, časovače, programovatelný obvod, A/D převodník, 32 kB paměti flash. Tento čip existuje ve více variacích, dle typu určení. Je použit ve většině verzí Arduina, protože zvládne téměř každý projekt. [19]

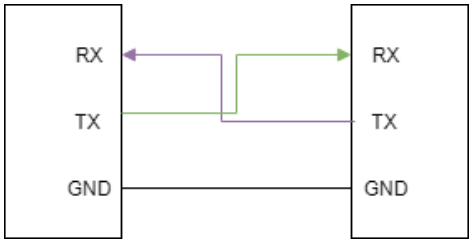
2.6 Vstupy a výstupy Arduina

K propojení Arduina se vstupy a výstupy slouží tzv. piny. Desky Arduino mají různý počet pinů, např. Arduino Uno použité pro tuto práci má 20 pinů, z toho 14 digitálních a 6 analogových. [9]

2.6.1 Digitální piny

Prakticky je možné téměř všechny piny na Arduinu použít jako digitální vstup či výstup. Základní rozsah, který je opravdu určený jako digitální vstupy a výstupy se nachází u Arduino Uno na pinech 0 až 13. Piny přijímají a odesílají pouze logickou 0 a 1. V tom případě hodnota 1 odpovídá přivedeným 5 V a naopak logická 0 odpovídá 0 V.

Některé piny nejsou pouze digitální, ale mají přídavné vlastnosti. Piny určené ke komunikaci na sériové lince se nachází pod čísly 0 a 1 a jsou značeny Tx a Rx. Význam těchto zkratek je jednoduchý. Tx – Transmited čili odeslané a Rx – Received, přijaté. [9]



Obrázek 10 Propojení pro sériovou linku

Jak již bylo psáno, jedná se o vstupní a výstupní piny. Tedy je nutné, tuto vlastnost definovat. Obvykle tuto vlastnost definujeme v části kódu Setup (). Tento kód proběhne pouze jednou, a to po spuštění programu, proto se jedná o vhodné místo. Mohou být však případy, kdy je nutné jednou použít pin jako vstup a poté ho v průběhu změnit. V tom případě je možné použití i v části Loop().

```
void setup() {
pinMode(13, OUTPUT);
}
```

Ukázka kódu 1 void setup()

Zápis nám říká, že digitální pin č. 13 bude sloužit jako výstupní.

```
void loop() {
digitalWrite(13, HIGH);
}
```

Ukázka kódu 2 void loop()

Tento zápis nastaví pin na hodnotu HIGH – 1. To znamená, že na diodu bude přivedeno maximální napětí. Vzhledem k tomu, že na Arduino Uno je na pinu č. 13 dioda, důsledkem tohoto zápisu bude její rozsvícení.

Následující kód demonstruje výše zmíněnou možnost změny vlastnosti pinu v průběhu vykonávání kódu. Je možné vidět, že v jednu chvíli je pin č. 13 výstupní a v druhou chvíli vstupní.

```
int vystup;                 //Deklarace proměnné datového typu Integer

void setup() {              //Definiční část, která proběhne pouze
                            //jednou při startu
Serial.begin(9600);         //Spuštění sériové linky, kvůli možnosti
                            //zobrazení hodnoty na pinu
pinMode(13, OUTPUT);        //Nastavení pinu č. 13 na výstupní pin
}                           //Uzavření setup()
void loop() {               //Část kódu, která běží neustále ve smyč-ce
digitalWrite(13, HIGH);     //Zapsání hodnoty HIGH - 1 na pin č. 13
pinMode(13, INPUT);         //Nastavení pinu č. 13 na vstupní pin
vystup=digitalRead(13);     //Uložení hodnoty zapsané na pinu č. 13
Serial.println(vystup);     //Tisk hodnoty a zobrazení na séri-ové lince
delay(2000);                //Vyčkání 2000milisekun, aby byly změny
                            //pozorovatelné
pinMode(13, OUTPUT);        //Nastavení pinu č. 13 na výstupní pin
digitalWrite(13, LOW);      //Zapsání hodnoty LOW
pinMode(13, INPUT);         //Nastavení pinu č. 13 na vstupní pin
                            //na pin č. 13
vystup=digitalRead(13);     //Uložení hodnoty zapsané na pinu č. 13
Serial.println(vystup);     //Tisk hodnoty a zobrazení na séri-ové lince
delay(2000);                //Vyčkání 2000milisekun
}                           //Uzavřeni smyčky loop()
```

Ukázka kódu 3 Zápis a čtení digitálních pinů

Výstup na sériové lince odpovídá řetězci jedniček a nul po dobu přivedeného napětí.

2.6.2 Analogové piny

Pro zpracování analogových signálů v mikropočítači je třeba analogově digitální převodník. Ten se většinou nachází v hlavním mikrokontroleru. Tak to je i u Arduina Uno, kde je v mikrokontroleru použit 10bitový převodník. To znamená že jsou přijaté analogové hodnoty převedeny na číslo o délce 10 bitů, tedy hodnotu v rozsahu 0 až 1023.Tedy na hodnoty od 0 do 1023. Maximální hodnota, která může být přivedena na analogový vstup je 5 V, to znamená, že jednotlivé hodnoty odpovídají necelým 5 mV. Analogové piny se značí písmenkem A. Konkrétně na desce Uno jsou to piny A0 až A5. [9]

```
int jas;                //Deklarace proměnné dat. typu Integer
void setup() {          //Definiční část
pinMode(13, OUTPUT);    //Nastavení pinu č. 13 na výstupní pin
}                       //Uzavření setup()
void loop() {           //Část kódu, která běží ve smyčce
analogWrite(13, jas);   //Zapsání analogové hodnoty uložené v pro měnné
                        //jas na pin č. 13 na kterém se nachází dioda
delay(2000);            //Vyčkání 2000milisekun
if (jas < 255){         //Dokud je hodnota jas menší než hraniční hodnota
                        //pro zápis analogové hodnoty (255), tak je
                        //podmínka splněna
jas=jas+5;              // Hodnota proměnné jas se zvětší o 5
  }                     //uzavření splněné části if
else{                   //V případě, že není podmínka splněna
jas = 0;                //V případě že je hodnota jas stejná nebo větší
                        //než 255, proměnná jas, bude 0
    }                   //uzavření nesplněné podmínky else
}                       //Uzavřeni smyčky loop()
```

Ukázka kódu 4 Použití analogových pinů

2.7 Shieldy

Arduino shieldy (štíty) jsou rozšiřující desky, které se dají nasadit na základní desku Arduino. Je možné na shield nasadit další shield a tím způsobem je stohovat na sebe. Každý štít je kompletním řešením rozšíření o zvolenou funkčnost. Tyto štíty jsou vytvořeny na rozložení desek Arduina Uno a Mega. [10]

Mezi nejběžněji používanými štíty najedeme:

- Wifi shield
- Bluetooth shield
- Ethernet shield
- LCD shield
- Relé shield
- GMS shield
- Motor shield

2.7.1 Motor shield

Motor shield využívá H-můstek k řízení DC motoru. Shield může obsahovat jeden či dva H-můstky, podle počtu stejnosměrných motorků, které chceme ovládat. V případě ovládání krokového motoru je nutné mít shield s dvěma můstky. Rychlost motorů je regulovatelná napájecím napětím nebo pomocí PWM. Regulace napájením je ztrátová, zatímco PWM tuhle nevýhodu nemá. [27]

Vstupní napětí je od 2,5 do 12 VDC. Většinou je vhodné motor doplnit a externí napájení.

2.8 Senzory a moduly

Různé senzory a moduly jsou něco, co základnímu mikropočítači dodávají potřebné množství informací a rozvíjí jeho možnosti. Dodávají kontakt se samotným okolím. Tyto součástky jsou elektronické nebo mechanické, dle typu určení. Měřené veličiny generované v senzoru pak mohou být kromě mechanických a elektrických i optické, nebo chemické. Když senzory a moduly přirovnáme k lidskému tělu, tak jsou senzory oči, uši nos, které sbírají informace k dalšímu zpracování. Výstupní moduly by naopak odpovídali ústům a prstům, kterým jsou naopak z hlavní soustavy (mikropočítače) odesílány příkazy. [4][14][25]

2.8.1 Hallův senzor

Jedná se o elektronickou součástku detekující magnetické pole. Senzor využívá Hallova jevu, který funguje na principu vychylování směru elektrického proudu v závislosti na velikosti indukce magnetického pole, které je kolmé na polovodičovou tenkou destičku. Výstupem je odlišné napětí na bočních stranách.

Pokud se objeví v jeho okolí magnetické pole, působí na elementem procházející proud elektronů tzv. Lorentzova síla, která elektrony vychyluje z přímého směru vždy k jedné boční straně destičky silou podle vzorce $F = Q (v \times B)$, kde Q je náboj, v je jejich rychlost a B je indukce působícího mag. pole. Změní se tak rozložení náboje, kdy na jedné straně je větší koncentrace nosičů náboje (elektronů) než na druhé a tedy obě boční stěny destičky mají rozdílný potenciál. Vzniká tak elektrické pole E a na svorkách Hallova elementu se generuje tzv. Hallovo napětí VH. [23] [14]

Senzor se používá podle typu určení na měření magnetického pole, elektrického proudu nebo napětí. Dále lze pomocí Hallova senzoru měřit a detekovat pohyb, přiblížení a umístění. [22][24]

Zapojení senzoru

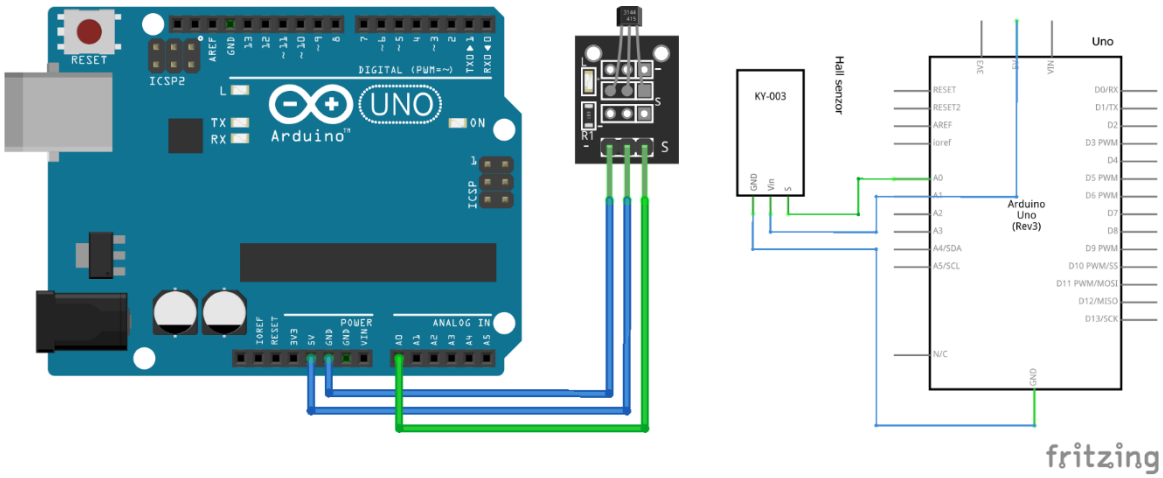


Schéma 1 Zapojení Hallova senzoru

Ukázka Arduino kódu

```
const int hall = A0;

void setup() {
  pinMode(hall, INPUT);
  Serial.begin(9600);
}
void loop() {
  int value = analogRead(hall);
  Serial.print("hall = ");
  Serial.print(value);
  Serial.print("\n");
  delay(100);
}
```

Ukázka kódu 5 Použití Hallova senzoru

2.8.2 Analogový teploměr

Modul je založen na termistoru, kde se odpor s teplotními změnami zvyšuje. Tento termistorový snímač využívá závislosti změny elektrického odporu na teplotě. Hodnotu získanou senzorem je nutné přepočítat na určitou stupnici. [14][26]

Zapojení senzoru

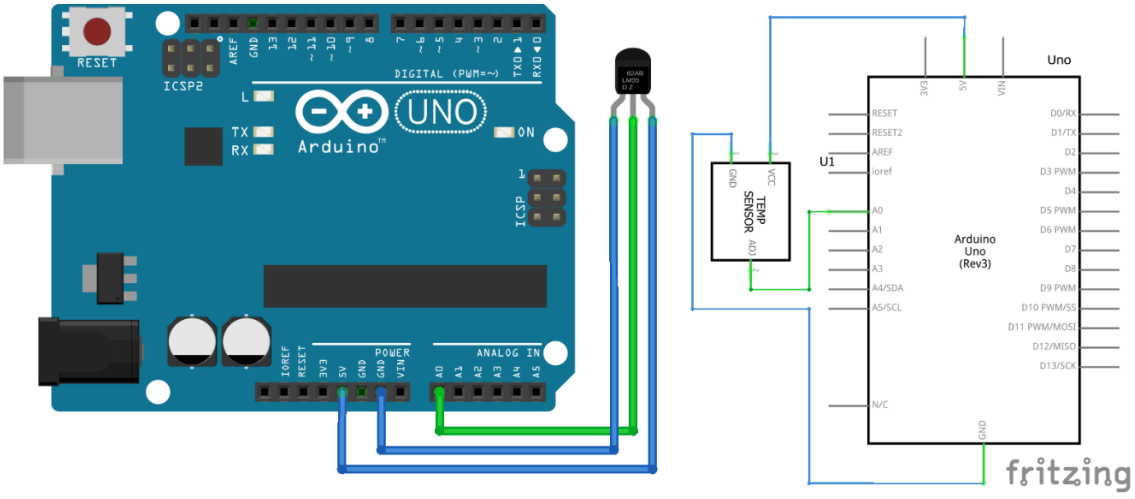


Schéma 2 Zapojení analogového teploměru

Ukázka Arduino kódu

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  float teplota = analogRead(A0);
  float celsius = teplota / 1024.0 * 500;
  Serial.println(celsius/10);
  delay(2000);
}
```

Ukázka kódu 6 Použití analogového teploměru

2.8.3 Detektor nárazu

Detektor reaguje na nárazy, otřesy a vibrace. Funguje na principu toho, že uvnitř hlavního válečku je umístěn středový sloupek, kolem kterého je namotaná pružina. Při otřesech dochází k přiblížení pružiny a středového sloupku. [14]

Zapojení senzoru

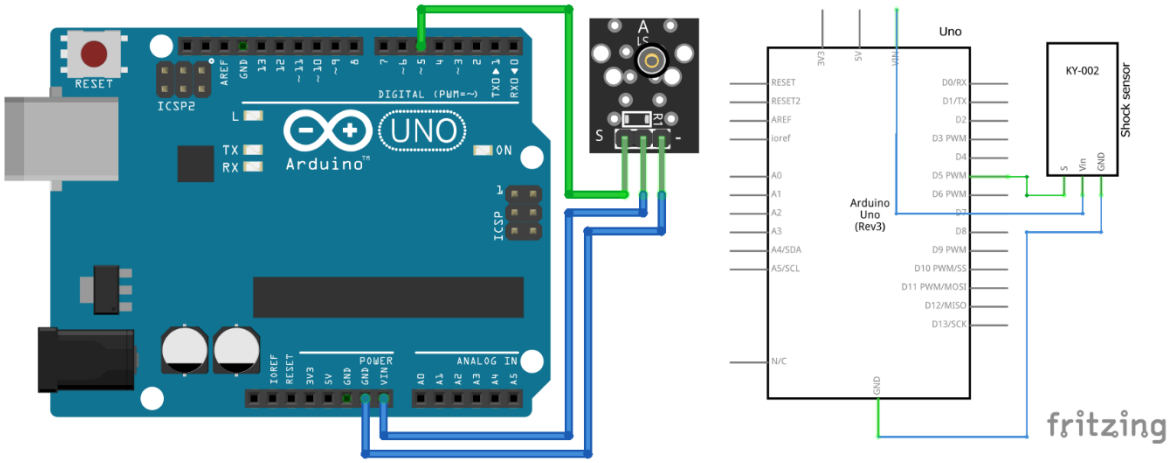


Schéma 3 Zapojení detektoru nárazu

Ukázka Arduino kódu

```
void setup() {
 pinMode(13, OUTPUT);
 digitalWrite(13, LOW); // turn off onboard LED
 pinMode(5, INPUT);
}
void loop() {
  if (digitalRead(5) == LOW){
    digitalWrite(13, HIGH);
    delay(500);
  }
  else{
    digitalWrite(13, LOW);
  }
}
```

Ukázka kódu 7 Použití detektoru nárazu

2.8.4 Náklonový spínač

Spínač obsahující kuličku rtuti slouží jako spínač při naklonění. V oválném skleněném válečku je navedený elektrický obvod, který se v případě náklonu uzavře rtuťovou kuličkou. [14]

Zapojení senzoru

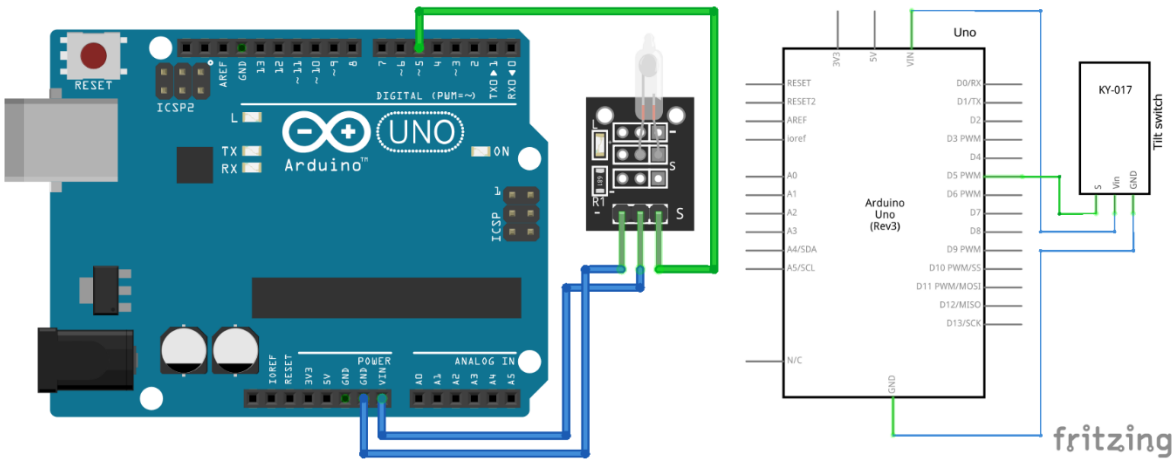


Schéma 4 Zapojení náklonového senzoru

Ukázka Arduino kódu

```
void setup() {
 pinMode(13, OUTPUT);
 digitalWrite(13, LOW); // turn off onboard LED
 pinMode(5, INPUT);
}
void loop() {
  if (digitalRead(5) == LOW){
    digitalWrite(13, HIGH);
    delay(500);
  }
  else{
    digitalWrite(13, LOW);
  }
}
```

Ukázka kódu 8 Použití náklonového senzoru

2.8.5 Laser modul

Laserové moduly lze využít k laserovým závorám, nebo třeba jako ukazovátko. Laser funguje jako optický zdroj světla. Tento zdroj světla je monochromatický, koherentní jednosvazkové záření. Lasery jsou tvořeny rezonátorem, aktivním prostředím a zdrojem energie. [14]

Zapojení senzoru

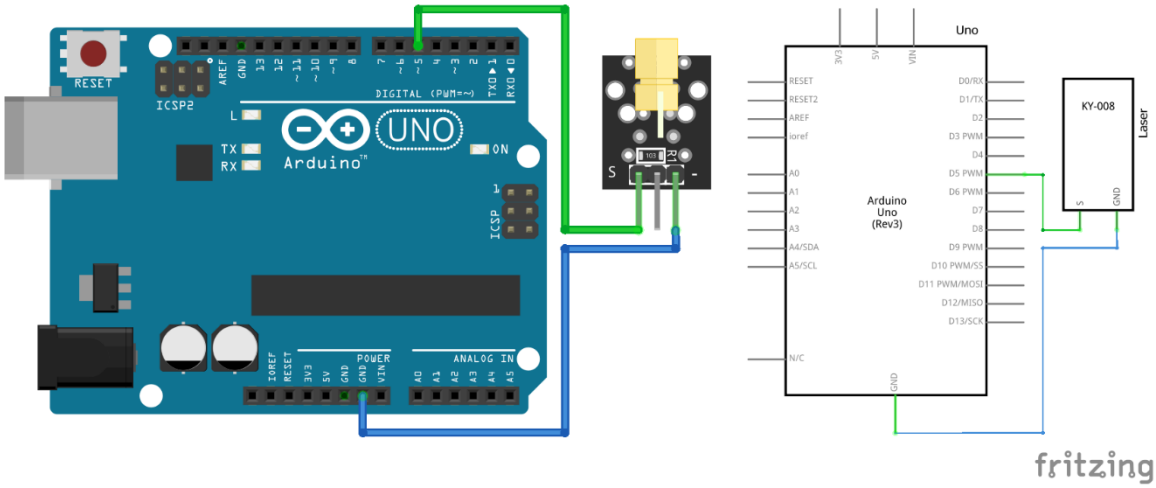


Schéma 5 Zapojení laser modulu

Ukázka Arduino kódu

```
void setup ()
{
pinMode (5, OUTPUT);
}
void loop () {
digitalWrite (5, HIGH);
delay (1000);
digitalWrite (5, LOW);
delay (1000);
}
```

Ukázka kódu 9 Použití laser modulu

2.8.6 Senzor pro měření vlhkosti vzduchu

Senzor slouží k měření teploty a vlhkosti v místnosti. Tyto senzory mají vše potřebné implementované v sobě, a tedy stačí pouze senzor zapojit a použít v hodný kód. Překážkou je pouze nutnost knihovny DHT, která je volně dostupná na internetu. Elektronické snímače měří vlhkost měřením kapacity nebo odporu vzorků vzduchu. Kapacitním vlhkoměrem proudí vzduch mezi dvěma kovovými deskami. Změna vlhkosti vzduchu je přímo úměrná změně kapacity mezi deskami. V odporovém vlhkoměru absorbuje keramický nebo vodivý polymer vlhkost, která pak ovlivňuje jeho odpor. Je připojen k obvodu, kde vlhkost ovlivňuje odpor materiálu. Relativní vlhkost je pak určena na základě změny proudu. [14][27]

Zapojení senzoru

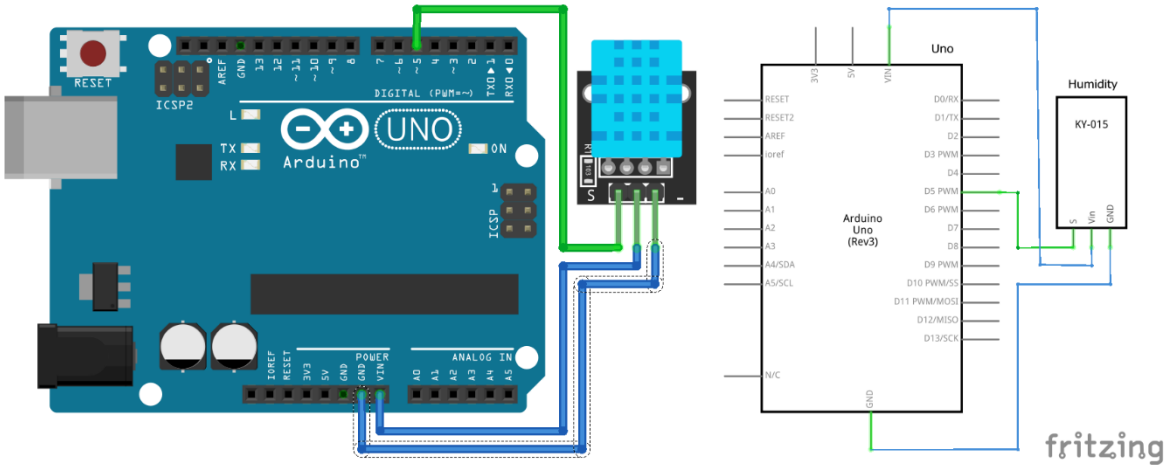


Schéma 6 Zapojení senzoru vlhkosti

Ukázka Arduino kódu

```
#include <DHT.h>
#include "DHT.h"
#define pinDHT 5
#define typDHT11 DHT11
DHT mojeDHT(pinDHT, typDHT11);

void setup() {
  Serial.begin(9600);
  mojeDHT.begin();
}
void loop() {
  float tep = mojeDHT.readTemperature();
  float vlh = mojeDHT.readHumidity();
    Serial.print("Teplota: ");
    Serial.print(tep);
    Serial.print(" stupnu Celsia, ");
    Serial.print("vlhkost: ");
    Serial.print(vlh);
    Serial.println("  %");
  delay(100);
}
```

Ukázka kódu 10 Použití senzoru pro měření vlhkosti

2.8.7 RGB dioda

Výstupní světelný modul ve verzi tří samostatných diod na jedné destičce. Dioda je polovodičová součástka obsahující P-N přechod. Díky němu vyzařuje světlo. O vyzařované barvě rozhoduje chemické složení. LED diody neobsahují žádné pohyblivé součástky a díky tomu vydrží téměř vše. [14]

Zapojení senzoru

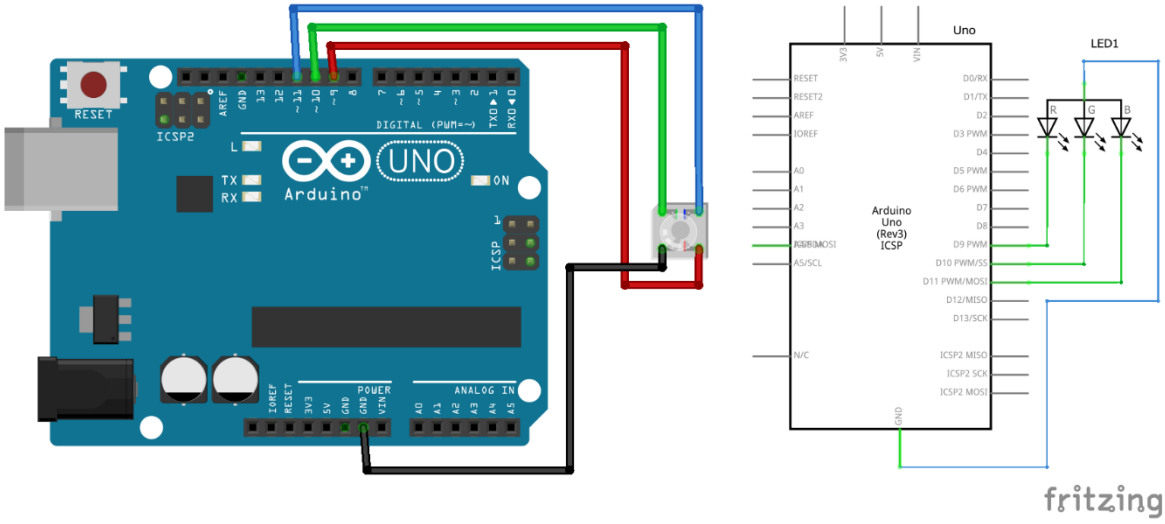


Schéma 7 Zapojení RGB diody

Ukázka Arduino kódu

```
#define pinR 9
#define pinG 10
#define pinB 11
void setup() {
  pinMode(pinR, OUTPUT);
  pinMode(pinG, OUTPUT);
  pinMode(pinB, OUTPUT);
}
void loop() {
  Rozpětí svítivosti 0..100
  analogWrite(pinR, 10);
  analogWrite(pinG, 10);
  analogWrite(pinB, 10);
}
```

Ukázka kódu 11 Použití RGB diody

2.8.8 Ultrazvukový senzor

Ultrazvukový senzory se v nejčastěji používají v případech, kdy je nutné přesnější mapovaní prostoru před senzorem. Senzory jsou schopné získávat spojité i nespojité veličiny. To znamená, že můžeme zjistit, zda v rozptylu senzoru je nějaký předmět, či spojité hodnoty vzdáleností senzoru od předmětu. Senzory fungují tak, že je vyslán kuželovitý (60°) ultrazvukový signál. V případě, že se v rámci tohoto zvukového dosahu objeví předmět, signál se od tohoto předmětu odrazí a putuje zpět k snímači. Vzdálenost je poté přepočítána z času, po jakém signál putoval. Je možné se setkat s ultrazvukovými senzory, které disponují odděleným vysílačem a snímačem, nebo tyto části mohou být kombinovány do jednoho prvku. [14]

Zapojení senzoru

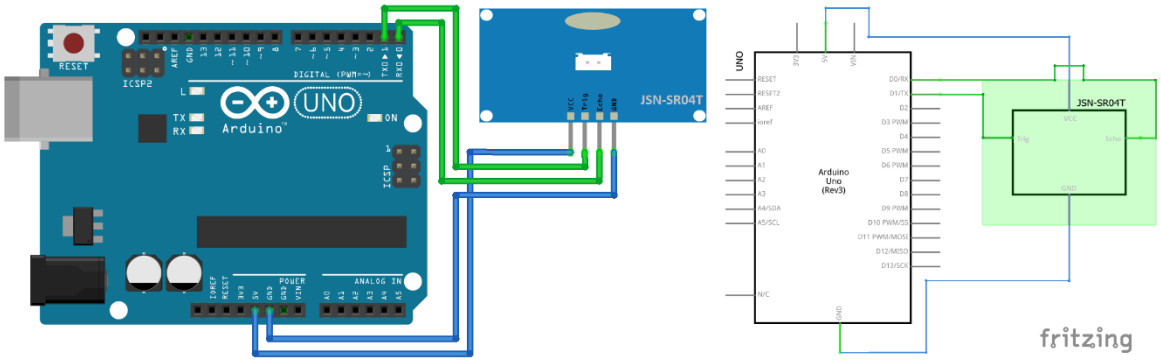


Schéma 8 Zapojení ultrazvukového senzoru

Ukázka Arduino kódu

```
void setup() {
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
}
void loop(){
digitalWrite(trigPin, HIGH);
delayMicroseconds(2);
digitalWrite(trigPin, LOW);
delayMicroseconds(5);
odezva = pulseIn(echoPin, HIGH);
vzdalenost = (odezva * (340/2))/10000
}
```

Ukázka kódu 12 Použití ultrazvukového senzoru

2.8.9 Relé

Relé slouží k nespojitému ovládání elektrické energie. Relé mohou být paměťová, polarizovatelná, jazýčková a elektromagnetická. Sepnutí relé je prováděno elektromagneticky. Je možné použití stejnosměrného i střídavého napětí. Mezní frekvence napětí je hodnota, při které relé spolehlivě spíná. [14][27]

Zapojení modulu

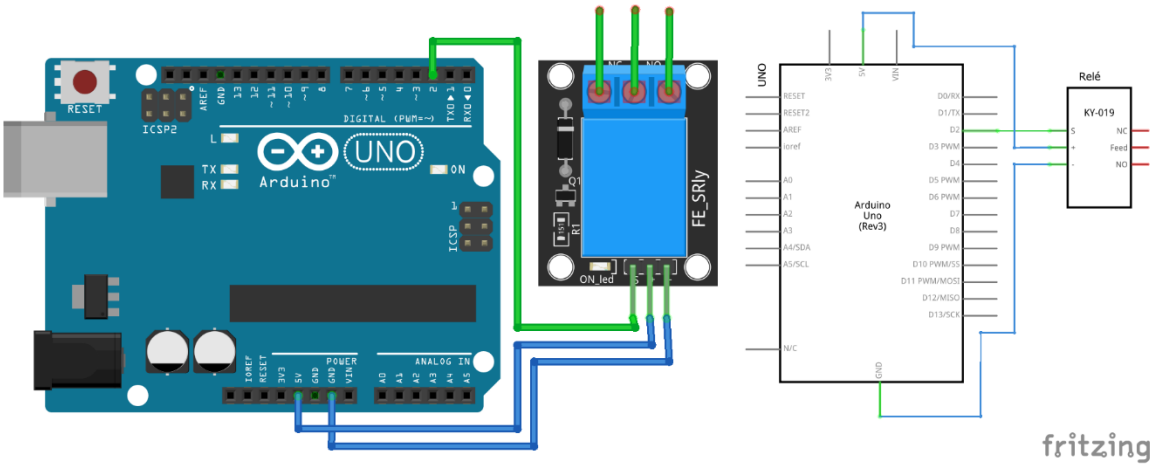


Schéma 9 Zapojení Relé

Ukázka Arduino kódu

```
int rele = 2;

void setup(){
pinMode(rele,OUTPUT);
}
void loop() {
  digitalWrite(rele,HIGH);
  delay(10000);
  digitalWrite(rele,LOW);
  delay(1000);
}
```

Ukázka kódu 13 Použití modulu s relé

3 BEZDRÁTOVÁ KOMUNIKACE

Bezdrátová komunikace se využívá tam, kde není možné, nebo nechceme využít k propojení zařízení kabeláž. Informace lze bezdrátově přenášet opticky, rádiově, elektromagneticky či ultrazvukově. Mezi nejznámější druhy bezdrátového přenosu patří WiFI (Bezdrátová komunikace v počítačových sítích), Bluetooth (propojení více elektronických zařízení), 3G (propojení v rámci mobilní sítě), IrDa (Přenos dat na krátké vzdálenosti), GSM a mnoho dalších (propojení v síti za pomoci SIM karty). [21]

3.1 Bluetooth

Bluetooth bylo představeno v roce 1989, ale ve skutečnosti se začalo používat až v roce 2000. V podstatě bluetooth není ani samostatný protokol, ale jedná se o soubor různých protokolů (LMP, L2CAP, SDP, ...) seskupený do specifikace. Bylo vyvinuto za účelem nahrazení kabelové komunikace. Jedná se o pohodlnější náhradu, dnes již velmi často používanou. Tento soubor protokolů není tak širokospektrální jako Wi-Fi. Je vhodný k využívání např. aplikacemi, komunikací na krátkou vzdálenost nebo synchronizaci dat. K synchronizaci dat je využíván ku příkladu Androidem. Kdy při koupi nového telefonu stačí spustit synchronizaci a zdrojový telefon mít v těsné blízkosti. Data jsou 1:1 překopírovány do nového zařízení. Zařízením Bluetooth disponuje 100 % telefonů, tabletů i notebooků vydaných v roce 2018. Jednotlivých Bluetooth komponent se vyrobí za rok zhruba 3,4 miliardy kusů. Aktuální nejnovější verze je Bluetooth 5 z roku 2016. Disponuje nižší náročností na napájení, lepším zabezpečením a dosahem až 240 m na volném prostranství a 40 m v interiéru. Čím dál modernější je využívat různé bezdrátové periferie. Nejčastěji jsou využívána bezdrátová sluchátka, klávesnice, myši, tiskárny nebo také lze využít v bezdrátovém ovládání. [20][34]

3.1.1 Párování

Párování slouží k tomu, aby se zařízení mohlo rychle a snadno připojit. Poté je vytvořeno pevné spojení mezi zařízeními, které již funguje automaticky. Příkladem může být: Je spárovaný telefon s bezdrátovými sluchátky. Po zapnutí sluchátek v dosahu telefonu se zapnutým bluetooth se automaticky zařízení připojí

Párování probíhá ve třech fázích. V první fázi se zařízení vzájemně informují. Využívají k tomu protokol L2CAP

, který většinou není šifrovaný. Také se určuje, jaký druh párování bude využit. Ve druhé fázi se vygeneruje dlouhodobý klíč v případě zabezpečeného módu připojení, jinak se generuje pouze krátkodobý klíč. Ve třetí fázi se pomocí klíče z druhé fáze odesílají ostatní klíče potřebné pro komunikaci. V této fázi se generují data jako klíč pro rozpoznání připojení, podepisování dat a rozpoznávání identity v podobě MAC adres. [20] [21]

3.1.2 Metody párování:

Číselné párování

Funguje na principu toho, že se na display obou zařízeních ukáže stejný, 6místný kód, který oba držitelé zařízení porovnají a pokud se shoduje, tak jej potvrdí. Slouží pouze jako kontrola, nikoli jako ochrana. Nepředstavuje téměř žádnou ochranu proti útoku MITM (Man in the middle). [21]

Přímé párování

Vzniklo, protože ne všechna zařízení disponují displejem. Jako třeba sluchátka, nebo reproduktory. Jedná se v podstatě o to stejné jako číselné párování, ale 6místná hodnota je nastavena na samé nuly. Tím pádem nepředstavuje vůbec žádnou ochranu proti útoku MITM. [21]

Přístupový klíč

Funguje tak, že na jednom zařízení se zobrazí 6místný kód a druhý účastník jej musí zapsat do svého zařízení. [21]

Mimo pásmová (OOB)

Metoda komunikace mimo komunikační kanál Bluetooth se nepoužívá, ale informace jsou stále zabezpečeny. Ku příkladu je využívá Apple watch. Dále tuto metodu využívá NFC.

3.1.3 Slabiny v bezpečnosti

Jednotlivé slabiny k celku přispívají k bezpečnostní hrozbě. Přispívá k tomu například krátký pin u párování, který má pouze 4 znaky. Problémem je také šifrovací klíč, jehož délka lze zvolit. Nejčastěji se však používá minimální délka. Sdílený hlavní klíč. Využití slabé proudové šifry – E0. Zabezpečení pouze navázání spojení komunikace, nikoli komunikace samotné. A nakonec Autentizace, která funguje na principu výzvy a odpovědi. Tímto způsobem jsou sice získány informace o zařízené, ale ne o uživateli samotném. [20][34]

4 TVORBA MOBILNÍ APLIKACE

Při tvorbě mobilních aplikací je důležité rozhodnout se, jaký cíl je kladen na tvorbu aplikace. Zda má aplikace cílit na konkrétní platformu a dodržení vzhledu, jako od tvůrců operačního systému, nebo „rozhodit sítě“ napříč platformami. Různé druhy vývoje mají své výhody a nevýhody a co mohou pro konkrétní problematiku nabídnout. [31]

4.1 Nativní vývoj

Nativní aplikace je tvořena na konkrétní platformu s dodržením prvků vzhledu. To znamená speciální aplikace pouze pro Android, IOS atd… Hlavními výhodami jsou grafický vzhled aplikace, optimalizace pro konkrétní platformu, efektivní přístup k funkcím zařízení bez potřeby pluginů a snadná aktualizace na vyšší verzi. Nativní aplikace jsou méně chybové než hybridní. Nevýhodou je při potřebě obstarání více platforem vyvíjet v podstatě dvě odlišné aplikace a tím narůstají i nároky na údržbu více „stejných“ aplikací.[31]

4.2 Hybridní vývoj

Hybridní aplikace je kombinace nativní i webové aplikace. V praxi to znamená, že v telefonu se aplikace zobrazuje nativně a na počítači jako prostý web. Tyto aplikace fungují za pomoci webových technologií HTML 5, Javascript, CSS atd… Nevýhodou je neoptimalizovanost pro konkrétní platformu. To znamená že aplikace mohou běžet výrazně pomaleji. Také jsou vyžadovány pluginy k ovládání konkrétních nativních funkcí. Bohužel, nejsou pluginy vyvíjeny výrobci operačních systémů, ale slouží pouze jako mezičlánek k jejich zpřístupnění. Pro tvorbu hybridních aplikací jsou používány různé knihovny a frameworky. Příkladem můžou být frameworky jako Cordova a Ionic. [31]

4.3 Webový vývoj

Webové aplikace jsou tvořeny bez omezení platformy. A tvoří se jak běžné WWW stránky. V tomto druhu aplikací se v základu využívá HTML 5 a Javascript. Jsou zobrazovány ve webovém prohlížeči, tedy je možné je spouštět na mobilních zařízeních i počítačích. Jako příklad může být webová verze Microsoft Office. Výhodou těchto aplikací může být cloudové uložiště, které data bezpečně ukládá u poskytovatelů hostingu. Snadná správa uživatelských účtů a jejich souběh v síti. Není nutná aktualizace, jelikož ta probíhá automaticky na vzdáleném serveru. [20][34]

4.4 Výuka programovacích jazyků

K výuce programovacích jazyků existuje spousta internetových kurzů, kde je možné získat nové znalosti, nebo si utvrdit znalosti již získané, třeba studiem ve škole. Kromě těchto kurzů existují i mobilní aplikace, kde je možné své znalosti procvičovat. Jedny takové aplikace vyvíjí SoloLearn. Nabízí aplikace na výuku programovacích jazyků vhodných k programování mobilních aplikací. Mezi nabízené jazyky patří Python, C++, C#, PHP, HTML, JavaScript, Java, SQL, Ruby, CSS, jQuery, Swift. Vše je možné zdarma stáhnout na Google play nebo Apple store. Dále může být podporou při výuce těchto jazyků webová stránka W3Schools.com, která se zaměřuje spíše na výuku webových aplikací. [17]

4.5 Ovládání Arduina pomocí mobilní aplikace

Existuje více cest, jakými se vydat při vývoji aplikace pro Android. Nejčastější a nejpodporovanější cestou je nativní vývoj v Android studiu za pomoci jazyka Kotlin, či Java. Nebo s nižšími nároky, je možné vytvořit rychlou mobilní aplikaci za pomoci bloků, stejně jako při programování samotného Arduina. [17]

4.5.1 OS Android

Jedná se o operační systém, navržený pro smartphony, tablety, chytré hodinky a televize. Základ tohoto systému je na Linuxovém jádře, které je distribuováno jako Open Source. Od roku 2016 tento systém využívá téměř 90 % chytrých telefonů. Pojmenování jednotlivých verzí má také svoji politiku. Všechny verze jsou pojmenovány po sladkostech. Od jablečného koláče přes banánový chléb, donut, perníček, oreo, až po koláč. V každé verzi jsou vždy přidány nové funkce, gesta, nebo je verze zoptimalizovaná a tedy svižnější. Nevýhoda je, že telefon, který je koupený teprve před dvěma lety, nemusí být již ze strany Androidu podporovaný.[2]

Mezi důležité vlastnosti systému patří multitasking. Tato vlastnost zajišťuje, že jednotlivé aplikace běží ve svých vlastní Virtuál machine. V případě odložení aplikace na pozadí, stará se o její obhospodařování systém sám.

4.5.2 Android studio

Jedná se o vývojové prostředí uzpůsobeno přímo pro potřeby vývoje pro systém Android. Mezi základní prvky prostředí rozhodně patří návrhová část, kde si na své čisté, či šablo-

nou navrhnuté okénko (Activity), skládáme a vytváříme jednotlivé prvky vzhledu. Tato část nemusí být vytvářena pouze přetahováním prvků do okénka, ale lze jí psát i jako text. Tato možnost je plynule přepnutelná v dolní části studia. Ke každému okénku náleží obvykle minimálně jeden soubor, který se stará o funkcionalitu okénka. Zde se dostáváme tedy ke kódové části. Tato část obsahuje všechny funkce, metody a procedury. Prostředí je uzpůsobeno k vývoji v jazyce Java nebo nově je plně podporovaný i jazyk Kotlin. Vytvořené aplikace lze testovat na vlastním zařízení, nebo je možné využít emulátor. V emulátoru je předvolených několik mobilních zařízení, na kterých je možné aplikaci simulovat. Nebo lze navrhnout a otestovat aplikaci na vlastnoručně nakonfigurovaném zařízení. [2] [17]

Nejdůležitější části aplikace vyvíjené pro android jsou uvedeny v Manifestu. Jedná se o soubor informací a povolení pro aplikaci. Jejími částmi jsou třeba Activity, které reprezentují obrazovku. Potom, servis, který provádění akcí na pozadí. Content providers, zajišťující přístup k datům a v neposlední řadě broadcast receiver, starající se o příchozí oznámení.

4.5.3 MIT App Invertor

Tento nástroj na tvorbu mobilních aplikací pro Arduino je dostupný ve webovém rozhraní. K jeho zprovoznění stačí pouze přihlášení přes Google účet. Samotný projekt je pak spravován ve dvou oknech. První je Designer, kde je předpřipravován vzhled aplikace. Na zobrazenou obrazovku, jsou postupně přetahovány komponenty, kterým se vdechuje život až v druhém okně, které se nazývá Blocks. Ekvivalentem by při vývoji klasické aplikace byl Kód. Blok editor je hodně podobný klasickému kódu, ale s textem ukrytým, do jednotlivých bloků. Mezi bloky najdeme i komunikaci za pomoci Bluetooth, která je zde podporovaná. K otestování aplikace je stejně jako v Android studiu možné použít emulátor, pro simulaci aplikace nahrané do mobilního zařízení. Pro spuštění vytvořené aplikace přímo v mobilním zařízení je nutné mít nainstalovanou aplikaci MITapp a za pomoci wifi či USB jí nahrát.

5 VÝUKA PROGRAMOVÁNÍ EMBEDDED SYSTÉMŮ

Spousta českých základních i středních škol přidala do svých osnov programování. Je to způsob, jak děti zaujmout a nenuceně a zábavně jim rozšířit znalosti. Především jde o rozvíjení logiky a chápání. Úkoly na školách zahrnují věci, z kterých je možné získat ponaučení nebo matematické myšlení. Programovací úlohy tak mohou znít například: Jak dostat objekt (auto, kouli, želvu…), na druhou stranu místnosti, když povolený pohyb je pouze ve směru vpřed, vzad, doleva, doprava. Nikoliv do úhlopříčky. Nejde však o programování v pravém slova smyslu. Na základních školách, se využívají především aplikace na programování s bloky. Cílem tedy není naučit programovací jazyk, ale pouze algoritmizaci a myšlení. Aktuálně je programování na většině škol pouze jako volitelná aktivita, do které se děti velice rády hlásí.

Hodina kódu je celosvětová akce probíhající v Týdnu informatiky. Cílem je děti seznámit s tím, že programovat, může umět každý a že je to vlastně i zábavné. Taky je součástí otevřít dětem dveře do světa informatiky a budoucího zaměření profese ve 21. století. Hodina kódu je pořádaná neziskovou organizací Code.org, která se věnuje zvýšení zájmu o informatiku ve společnosti. K tomu projektu se připojili i velikáni jako Microsoft, Apple či Amazon. Každý, kdo tuto hodinu absolvoval tak také dostane certifikát.

5.1 Robotické stavebnice

Robotické stavebnice jsou dobrým úvodem do elektroniky a světa programování. Děti i dospělí, tak mohou rozvíjet své znalosti, či získat úplně nové. Některé stavebnice jsou velmi intuitivní a vhodné pro ty nejmenší, jako start a úvod, některé jsou zase v hodné pro lidi se základními znalostmi. Rozvíjení logického myšlení, matematiky a fyziky je dobré už v nízkém věku a vzhledem k postupující době a zautomatizovaní všech procesů se postupně dostávají mikropočítače do chytrých domácností a dalších odvětví. Proto je vhodné seznamovat děti s touto situací už od nízkého věku. Na trhu je celá řada stavebnic a výukových kitů. Každá nabízí jiné vlastnosti, nároky na znalosti, věk a finance. [38]

5.1.1 LEGO boost

Stavebnice rozvíjí dětské logické myšlení již od nízkého věku. Obecně je doporučený věk od 7 do 12 let. První částí je seskládání lego kostek do požadovaného objektu a druhou částí je rozpohybování. Programování je jednoduché, a to za pomoci interaktivní aplikace

v mobilním telefonu či tabletu. Jedná se skládání bloků s instrukcemi. Tuto sérii úkonů pak robot jednorázově provede. Děti se učí principy základní výstavby kódu, kterou mohou využít v pokročilejším Scratchy. Postupně využívají jednoduchých příkazů, podmínek a interakcí se senzory. Cena stavebnice Lego boost se pohybuje kolem 3000 Kč. [38]

5.1.2 LEGO Mindstorms

Lego Mindstorms je nejvyšší řada programovatelného Lega. Výhodou je nespočetné množství součástek, jako jsou senzory nebo motory a s tím i velké množství různých robotů a pohyblivých objektů. Tato stavebnice je pro děti, či hravé dospělé, které již malé znalosti mají. Sada neobsahuje základní tutoriály, jako řada Boost. Opět se jedná o programování pomocí interaktivních bloků či využití klasického programovacího jazyku C nebo Java.

Základním dílkem každého robota je programovatelná kostka, která musí být obsažena v každém projektu. Díky ní je stavebnice pohyblivá a je možné jí ovládat. Tato kostka disponuje vstupy pro připojení senzorů a prostor pro napájení za pomoci tužkových baterií. Základním mozkem celé kostky je procesor ARM9 s 16 MB flash pamětí. Integrované Bluetooth a Wi-Fi, které je prostředkem komunikace pro ovládání a programování robota. Stejnou službu, ale drátově může vykonat i integrované mini-USB. V případě nedostatku paměti pro program je možné využít i SD kartu s kapacitou až 32 GB. Sada obsahuje několik základních senzorů pro tvorbu robota, jako jsou dotykové senzory, infračervený a barevný senzor. Dalšími prvky jsou pak třeba servo motory. Lego lze ovládat mobilním telefonem nebo dálkovým ovládáním. Cena stavebnice Lego Mindstorms se pohybuje od 7000 do 10 000 Kč, dle zvolené sady. [38]

5.1.3 Ozobot

Opět se jedná o interaktivní blokové programování, které bez znalosti programovacího jazyka učí algoritmizaci a logickému myšlení. Ozobot má vlastní webový programovací editor, tedy není nutné ho instalovat a je použitelný na jakékoliv platformě. Robot využívá barevný jazyk. To znamená že každá variace barev znamená jiný příkaz. Je možné měnit směr, rychlost, podsvícení. Nebo je možné nakreslit čáru, kterou bude robot sledovat. Krom naprogramovaného kódu se umí i sám rozhodovat. V případě že přijede na křižovatku a nemá určený směr, vybere cestu sám. Naprogramovaný kód v blocích, lze kdykoliv vidět v Javaskriptu. Cena robota Ozbot se pohybuje kolem 3500 Kč. [38]

5.1.4 Micro:bit

Micro:bit je mikropočítač, který je krokem k pokročilejšímu programování. Lze říct, že se jedná o Arduino pro začátečníky. Sada již na první pohled vypadá dost pokročile. Obsahuje různé senzory, nepájivá pole a propojovací drátky. Dokonce je možné použití i neoficiálních součástek pro rozvíjení modelu. Opět je možné pro začátky použít blokové aplikace a časem přestoupit na kód v Pythonu, Javascript nebo na jiný jazyk, který je prostředním podporován. Micro:bit je malá destička založená na procesoru Arm. Obsahuje také matici červených LED, dvě programovatelná tlačítka, akcelerometr a kompas. Jedná se o Open Source platformu. Cena samostatné destičky Micro:bit se pohybuje kolem 400kc, ale v případě koupi nějakého kitu, třeba startovacího, je cena o 1000 Kč dražší. [38]

5.1.5 Makeblock mBot

V tomto setu se již otvírají dveře do světa Arduina. Všechny komponenty fungují na principu Merkuru. Součástky lze k sobě smontovat a jednotlivými kabely spojit senzory s hlavním mozkem mBota. Z hlediska softwaru je možné použití interaktivní aplikace s bloky, nebo využít klasického prostředí Arduino IDE a programovat v jazyce C/C++. Stavebnice Makeblock mBot lze koupit ve více verzích kitů, jejíž ceny se pohybují od 3000 Kč do 5000 Kč. [38]

5.1.6 Arduino

Arduino je stavebnice, kde se základní znalostí elektroniky a fyziky je možné zvládnout snad vše. Těší se velké oblibě mezi domácími „bastlíři“. Vzhledem k oblíbenosti a platformě s licencemi open-source a open-hardware je velmi často kopírován čínskými výrobci. Díky tomu je však rozšířený a může si ho dovolit každý. Oproti předchozím stavebnicím za tisíce, až desetitisíce korun, se cena Arduino pohybuje od 50 Kč za klon, po částku k 1000 Kč za originál. Zde je samozřejmě nutné ovládat základy programovacích principů. K programování je využíván software Arduino IDE a jazyk C/C++. [3][9]

5.1.7 Raspberry Pi

Jedná se v podstatě o malý linuxový počítač. Výhodou je možnost připojení k televizoru, či vytvoření chytré domácnosti. Stejně jako k Arduinu, je možné připojení širokého množství komponent a prvků. Co se týče programování, je možné využití Scratche, C/C++, Pythonu,

Javascriptu a dalších jazyků. Cena Raspberry Pi se pohybuje v základu od 1000 Kč. Dle nabízených funkcionalit pak cena roste. [7][38]

5.2 Programovací jazyky

K programování mikropočítačů je možné použít téměř jakýkoliv jazyk. V dnešní době je na světě už téměř 2000 programovacích jazyků. Každý jazyk má své výhody i nevýhody. Nejlepší volbou je volba jazyku podle typu hardwaru a cíle programu.[31]

5.2.1 Vlastnosti programovacích jazyků

Paradigma

Jedná se o styl zápisu konkrétního programovacího jazyku. Liší se v jednotlivých prvcích, z kterých se program skládá. Jako objekty, funkce, proměnné.[31]

Jednotlivá paradigma:

- Imperativní programování – Základem je procedurální programování a definuje přesný postup řešení problému
- Aspektově orientované programování – Zvyšuje modularitu programu, rozděluje program na jasné jednotlivé části, u kterých, není překrývána funkcionalita.
- Deklarativní programování – Je specifikován cíl, ale o algoritmizace se stará jiný program (interpret)
- Function-level programování – Na program je nahlíženo jako na matematický objekt, program je bez proměnných
- Generické programování – Jde o obecné programování, bez ohledu datových typů,
- Meta programování – Jsou vytvářeny další podprogramy, nebo jsou za běhu programu potřeba činnosti, které se vytváří při kompilaci
- Paralelní programování – Jsou umožněny aplikace, které jsou schopny paralelního běhu úloh
- Strukturované programování – Snaha dosáhnout lepší srozumitelnosti, kvality a rychlosti vývoje
- Multiparadigmatický – Je využíváno více paradigmat s cílem lepšího výsledku

Typy programovacích jazyků

Je více způsobů, podle kterých lze programovací jazyky dělit.

Dělení jazyků podle abstrakce:

- Vysoko-úrovňové programovací jazyky – Jsou to jazyky s vyšší abstrakcí, to znamená, že je jazyk přiblížen myšlením člověka
- Nízko-úrovňové programovací jazyky – Jazyky s nižší abstrakcí jsou pak přiblíženy technickému zápisu z hlediska funkčnosti procesoru

Dále můžeme dělit podle způsobu překladu:

- Kompilované programovací jazyky – Napsaný kód je přeložen překladačem do strojového kódu a teprve z něj je program spuštěn
- Interpretované programovací jazyky – Interpret je program, který zajišťuje přímé vykonání jiného programu v jeho zdrojovém kódu

5.3 Grafické programování – jazyky pro výuku

Tento typ programování odprošťuje od klasického strukturovaného zápisu a znalosti syntaxe konkrétního strukturovaného jazyka. Algoritmy jsou vytvářeny vizuálně za pomocí bloků a ikonek znázorňující části kódu, jako jsou jednotlivé příkazy nebo řídící struktury. Tímto jsou pojištěny časté chyby překlepů v příkazech, či chybějících středníků. Programátor se tedy stará především o část návrhu funkčnosti. [37]

5.3.1 ARDUBLOCK

Jedná se o grafickou nadstavbu klasického Arduino IDE. Cílem je především zaujmout i nezkušenější programátory. Tato nadstavba je jako vše od Arduina Open source, tedy stažení je zdarma. Vše, co k zprovoznění ArduBlock je potřeba, je mít již stažené Arduino IDE. Po instalaci se ArduBlock spustí přes Arduino IDE. Nevýhoda pro mladší děti je tak, že se jedná o nástroj v angličtině, kterou děti nemusí úplně ovládat. Tento nástroj je v podstatě ulehčení hlavně v tom, že si člověk nemusí pamatovat zápis kódu. Ale pro naprosté začátečníky a nezkušené, se i tyto bloky zdají být složitější. Především proto, že příkaz kódu v jazyce C, odpovídá jednomu bloku. Stále se jedná o programovaní, i když s grafickou dopomocí. [9][36]

5.3.2 Scratch

Jednoduchý blokový programovací jazyk, určený především pro děti. Využívá se jako platforma k Arduino robotům nebo třeba mBotům. Jedná se o jednoduchý a srozumitelný nástroj vhodný pro děti začátečníky. Má velký okruh tvůrců sdílejících své projekty a tutoriály na internetu. Podporuje češtinu a tím je vhodný i pro malé programátory, kteří krom programovacího jazyku, neovládají ani jiný světový jazyk. Od tohoto jednoduchého grafického programovací jazyky jsou odvíjeny další jazyky pracující na stejném principu. [37]

5.3.3 SNAP!

Snap! je jazyk, vycházející přímo z jazyku Scratch. Jen zde odpadá nutnost instalace velkého prostředí. Vše je přeneseno do webového rozhraní. Snap oproti Scratchy nabízí možnost tvorby vlastních bloků a má již více datových typů.

5.4 Arduino IDE

Arduino IDE je software navrhnutý přímo k programování mikropočítačů Arduino. Obsahuje knihovny a ovladače ke všem originálním produktům Arduino. V případě použití klonu, při kterém neodpovídá originální knihovna, nebo je dokonce vytvořena deska trochu jiné koncepce, je možné nahrát do zařízení své ovladače nebo naimportovat knihovny.

Koncepce celého vývojového prostředí je vytvořena v jazyce Java. A práce v prostředí je podporována v jazycích C a C++ za podpory strukturovaného kódu.
Ke správné funkčnosti programu stačí definovat dvě hlavní funkce. První, která proběhne pouze jednou je funkce Setup. Druhá je funkce Loop, která běží celou dobu a to cyklicky. [3][9][33]

5.4.1 Barevné značení

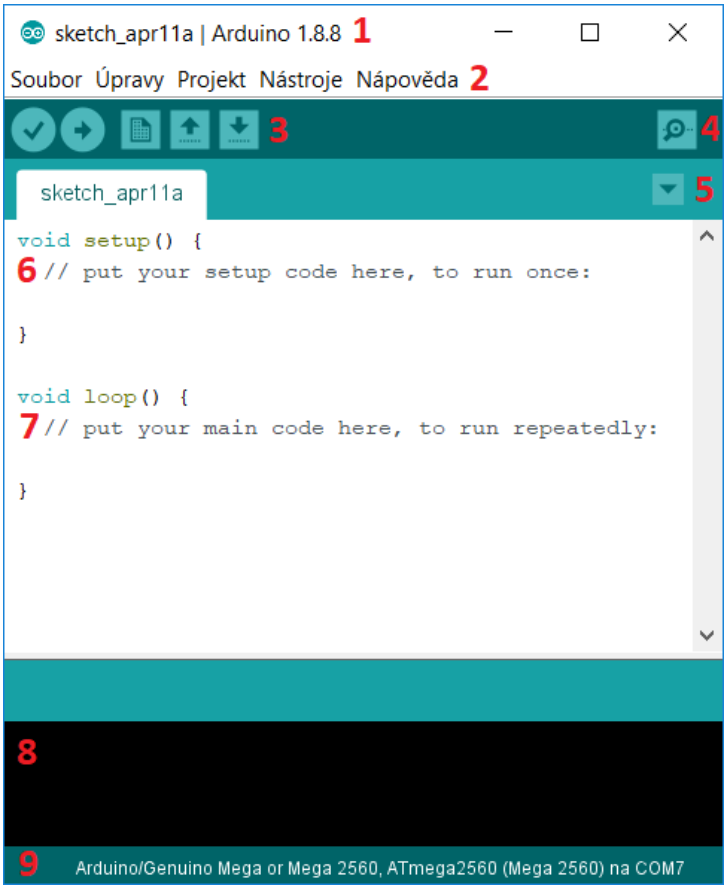
Barevné značení v softwaru Arduino IDE je děleno do 3 kategorií, přičemž tyrkysově jsou znázorněny proměnné, zeleně jsou znázorněna klíčová slova a oranžově jsou všechny ostatní funkce. [9]

5.4.2 Knihovny

Arduino IDE má spoustu vestavěných knihoven určených k práci s hardwarem či daty. Tyto knihovny lze spravovat v Manažéru knihoven, který je k nalezení v Nástrojích >

Spravovat knihovny... Knihovny, které nejsou součástí lze naimportovat nebo si vytvořit vlastní. Mezi standardní knihovny patří knihovna pro práci s Wi-Fi moduly, Servo moduly nebo Ethernet. [9]

5.4.3 Popis prostředí



Obrázek 11 Prostředí Arduino IDE

1. V záhlaví programu lze vidět název aktuální složky projektů. Dále lze vyčíst jaká verze Arduino IDE je aktuálně používaná.
2. Menu pro správu Arduino IDE i samotného projektu.
3. Ikonka fajfky slouží k ověření napsaného kódu, tedy zda v něm nejsou syntaktické chyby. Šipka ukazující směr vpravo zajišťuje nahrání projektu do zařízení (Arduino). List s ohrnutým rožkem slouží k vytvoření nového projektu. Šipka ukazující nahoru slouží k otevření, už existujícího projektu. Naopak šipka dolů projekty ukládá.

4. Po kliknutí na ikonku lupy se spustí sériový monitor. Na kterém lze ověřit data putující třeba mezi dvěma zařízení bluetooth.
5. Šipečka ukazující dolů otvírá další nabídku, v které je možné přidat záložku, přejmenovat záložku, smazat, nebo se mezi nimi pohybovat.
6. Funkce setup proběhne jednou, při spuštění. Proto je v hodná k nastavení jednotlivých prvků programu.
7. Zde se píše kód programu, co má Arduino vykonávat a tento kód běží v cyklu.
8. Příkazový řádek, vypisující chybová hlášení
9. Ze zápatí lze vyčíst jakou verzi Arduina právě používáme. Tahle informace se nastavuje ručně v nástrojích. A dále lze vidět, k jakém portu je zařízení připojeno.

II. PRAKTICKÁ ČÁST

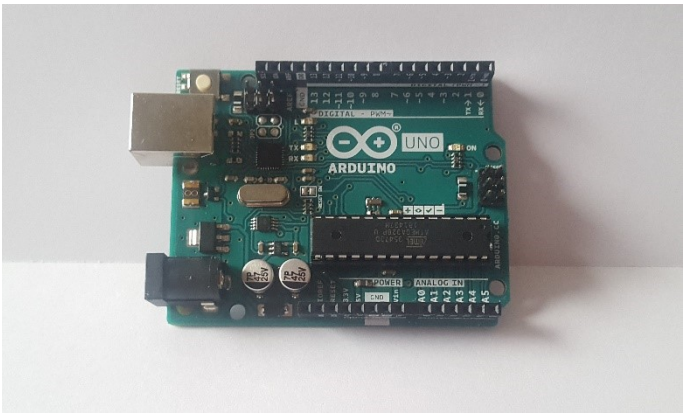
6 NÁVRH ŘEŠENÍ HARDWARU

Návrh celého řešení praktické části je nutné si rozdělit na více částí. Tyto části se skládají z hardwarových elektronických součástek, softwarového vybavení Arduino a softwarového vybavení mobilního zařízení s operačním systémem Android. V této první části ze tří, se budeme zabývat návrhem hardwarového řešení. Použitými komponenty a jejich propojení.

Vzhledem k široké škále možností senzorů a modulů, lze zvolit jakýkoliv senzor sehnatelný na trhu, nebo si i vyrobit svůj vlastní. Senzory zahrnuty v teoretické části s úryvky kódů lze přímo aplikovat na vytvořený model. Stačí si pouze vybrat, zapojit dle nákresů a přidat kód. Pro ukázku byly do praktické části zvoleny dvě LED diody, ultrazvukový senzor a bzučák.

6.1 Arduino Uno

Arduino Uno bylo zvoleno z důvodu snadného použití a predispozic k základním projektům. Disponuje všemi druhy vstupů a výstupů, jako jsou sériová komunikace, PWM piny. Analogové i digitální vstupy. Dále je k dispozici souosý konektor pro napájení a USB převodník pro snadné nahrání programu.

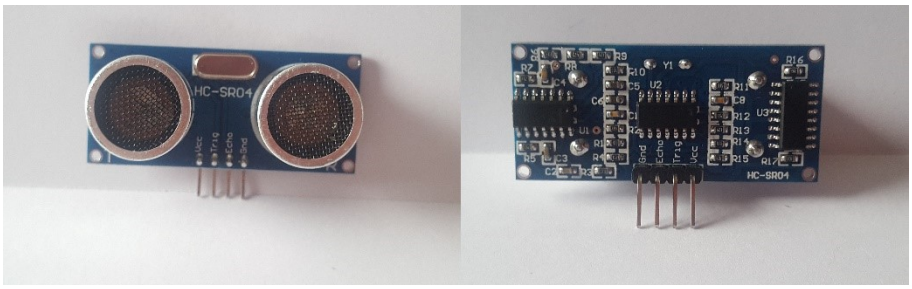


Obrázek 12 Použité Arduino Uno

6.2 Ultrazvukový senzor

Ultrazvukový senzor v modelu poslouží jako ochrana před nárazem proti zdi. Použitý senzor nese označení HC-SR04. Jedná se o senzor z Čínské distribuce určený pro jednočipové počítače k měření vzdálenosti objektů. V této verzi senzoru je oddělený vysílač od přijímače, proto vypadá, že jsou na modulu senzory dva. Rozsah měření je velmi přesný, psaný

rozsah je dokonce 2-400 cm s přesností, až na 3 mm. Zorný úhel je 12°. Jedná se o úhel, v rámci kterého, lze objekt zastihnout. Frekvence ultrazvukového signálu činí 40 kHz.



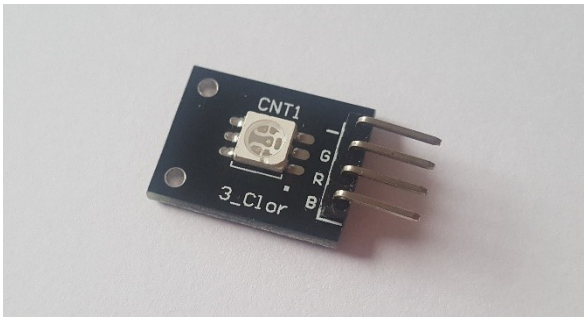
Obrázek 13 Použitý ultrazvukový senzor

6.3 LED diody

Modul použitý v práci obsahuje SMD LED diodu typu 5050. Jedná se nejčastěji používaný model diod. Dioda se skládá ze tří jednotlivých menších čipů. Tyto čipy jsou v jednom společném pouzdře. Díky této konstrukci jednotlivých čipů, mají diody vyšší světelný výkon. Na úkor vysokého výkonu dochází k velkému přehřívání. Proto je potřeba, při delším čase svícení, tyto diody chladit. Tento fakt snižuje i životnost diod.

Jednotlivé čipy zastupují barvy. Červená, modrá, bílá. Pro každou barvu je vhodné i jiné napájecí napětí. Optimální hodnota pro barvu červenou je 1,78 V, pro barvu zelenou 2,38 V a pro barvu modrou 2,59 V. Přičemž odběr proudu jednotlivých čipů se pohybuje kolem 0,2A.

Značení SMD označuje součástky, nejen LED diody, které mají povrchovou montáž, to znamená, že jsou součástky osazeny přímo svrchní část plošného spoje.

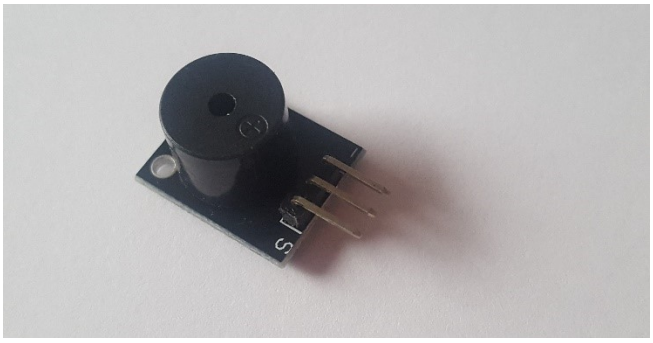


Obrázek 14 Použitá LED dioda

6.4 Aktivní bzučák

Bzučák slouží k reprodukci zvuku. Je založen na schopnosti deformaci krystalu, v případě, že je přiveden zdroj elektrického napětí. Aktivní bzučák je nutné vybudit vysokým proudem. Při přivedení tohoto proudu a nastavení hodnoty signálu na 1, je bzučák ihned rozezněn. Oproti tomu pasivní bzučák je nutné rozkmitávat. Proto je nutné ho vybuzovat na krátké časové intervaly ale v cyklech. Nevýhodou oproti aktivnímu bzučáku je nutnost využití cyklu, ale výhodu je, že k vybuzení stačí nižší proud.

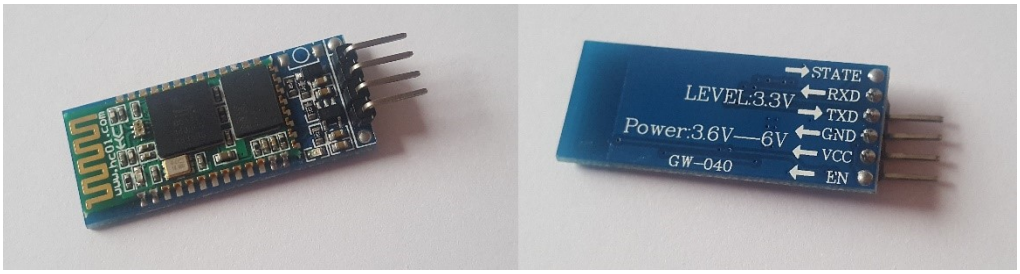
Z důvodu, že při tvorbě robota není prostor pro rozkmitávání bzučáku v cyklech, byl zvolen aktivní bzučák.



Obrázek 15 Použitý aktivní bzučák

6.5 Bluetooth modul

Jako bluetooth modul k bezdrátovému spojení byl zvolen HC-06. Jedná se o komunikační modul, který odesílá data na sériovou linku s rychlostí 9600 baudů. Baud je jednotka modulační rychlosti, to znamená, že udává počet změn stavu přenosového média za 1 sekundu.[24] Vzhledem k malé velikosti zařízení, je nutné počítat s tím, že i jeho anténa je dost malá. Proto je dosažitelná vzdálenost k 10 metrům. Ve členitém a rušivém prostředí mnohem méně. K zařízení je třeba přivést napětí od 3,3 do 6 V. Následný proudový odběr je kolem 2 mA v klidovém stavu a 40 mA v přenosovém, komunikačním stavu.



Obrázek 16 Použitý bluetooth modul

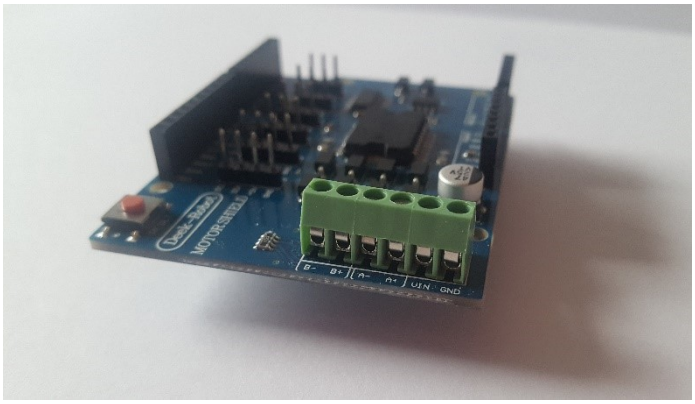
Bluetooth modul typu HC-06 pracuje v režimu Slave, oproti svému blízkému příbuznému HC-05, který může pracovat jak v režimu Slave, tak v režimu Master. Nic méně jde pouze o navázání spojení, které není pro tento projekt podstatné. V režimu Slave je možné pouze přijímat žádosti o navázání spojení. Režim Master umí krom toho i žádat jiné zařízení o navázání spojení.

6.6 Podvozek s motorem a převodovkou

Pro pojezdy robota byl zvolen podvozek s pásy a motorem s převodovkou od firmy Tamiya. Firma Tamiya vyrábí plastové stavebnice na baterie a dálkové ovládání. Je možné si koupit model, který je nutné si složit. Nebo lze koupit pouze komponenty a vytvořit si svůj model podle představ. Převodovka včetně celého podvozku při objednání přijdou v demontu. Proto je nutné vše složit a připojit motory, které budou převodovku pohánět. K napájení motorků je třeba 3-6 V. Motory pohání převodovky s poměrem převodu 114,7:1. A v konečné fázi, převodovky roztáčí přes dvě velká a dvě malá hnaná kola zbylých osm pojezdových kol a dvě napínací kola. [4][8]

6.7 Motor shield

Pro rozšíření základního Arduino Uno o modul ovládání motoru byl zvolen Motor shield L298 z Čínské distribuce. Tento shield není určený pouze pro ovládání krokového nebo stejnosměrného motoru, ale lze s ním také ovládat relé a solenoidy. Shield je založen na dvojitém můstkovém regulátoru navrženém pro řízení indukčních zátěží. Napájení shieldu je specifikováno na rozsah 5–12 V. Velikost napájecího napětí napájení se přímo odráží na výkonu motorů. Motory lze zapojit dva stejnosměrné nebo jeden krokový.

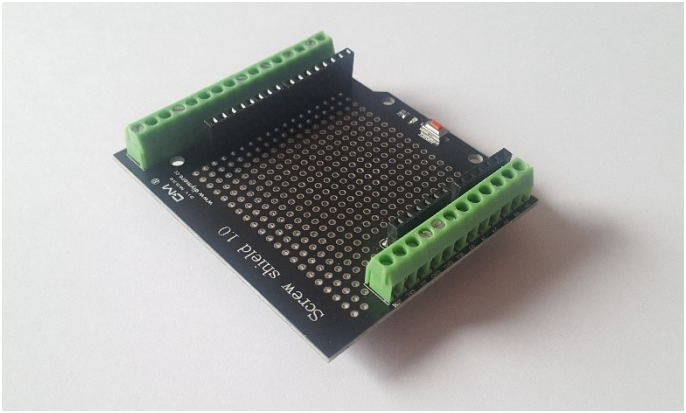


Obrázek 17 Použitý motor shield

6.8 Terminál shield

Jedná se o shield s vyvedenými svorkami pro konečné zapojení. Výhodou je to, že není nutnost součástky pájet, ale stačí pouze přišroubovat a v případě nutnosti odšroubovat. Je vhodný k použití v konečné fázi tvoření. Zajišťuje pevné, snadno měnitelné propojení.

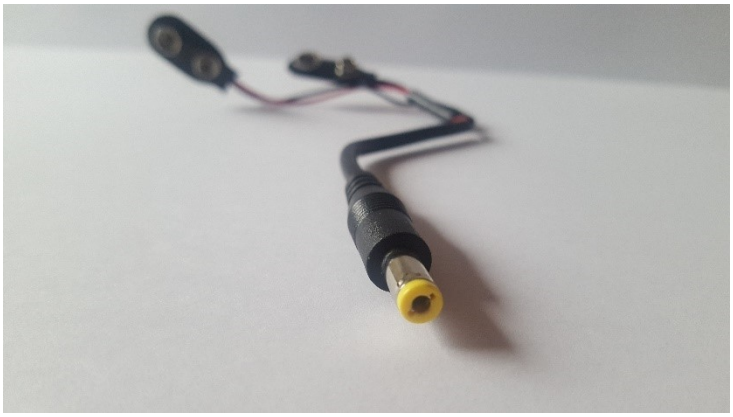
Krom samotných svorek jsou z pinů vyvedeny i původní dutinková lišta. Prostor uprostřed je určený k pájení. Je možné připájet jakoukoli komponentu, klidně i nepájivé pole.



Obrázek 18 Použitý terminál shield

6.9 Zdroj napájení přes souosý konektor

Adapter souosého konektoru typu T slouží k napájení celého projektu. Na druhém konci adaptéru se nachází zacvakávací „truky“ na 9 V baterii.



Obrázek 19 Použité napájení přes souosý konektor

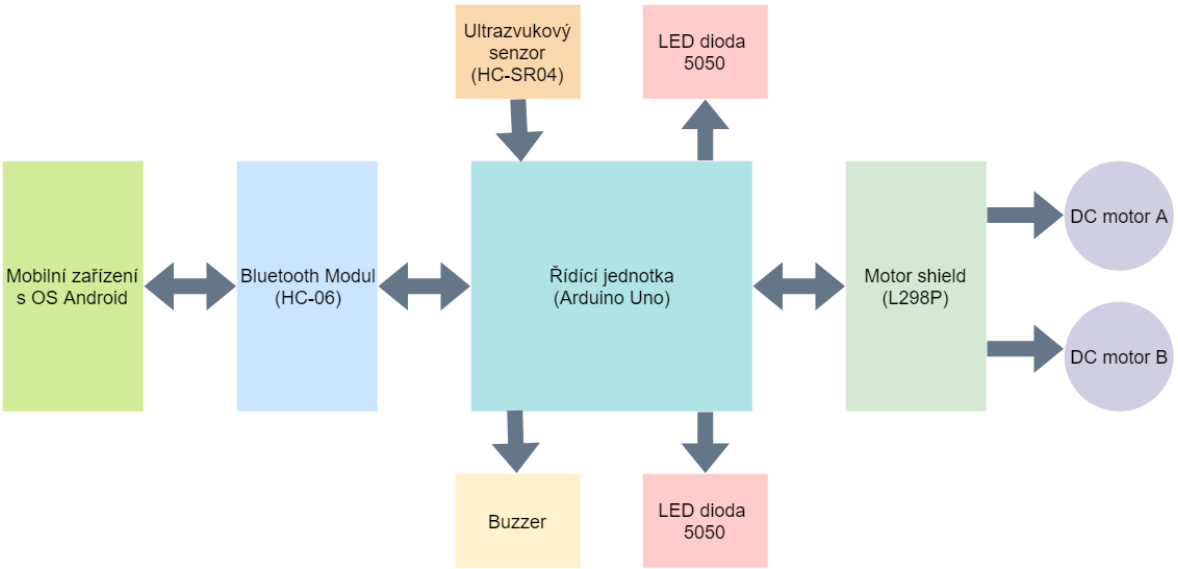
6.10 Návrh zapojení

V této části práce je uvedeno konečné propojení všech součástek.

6.10.1 Blokové schéma zapojení

Blokové schéma znázorňuje funkční prvky celého systému. Každý, z jednotlivých bloků odpovídá části a funkcionalitě systému. Blokové schéma je jednoduchým zobrazením celé problematiky.

Na obrázku č.12 je možné vidět největší blok s názvem řídicí jednotka. Jedná se o hlavní mozek systému. Tento blok zastupuje zvolený mikropočítač Arduino Uno. Z bloku s názvem Bluetooth Modul je oboustranná šipka z důvodu toho, že mikropočítač nejenom data odesílá, ale také dostává. Obdobný vztah je také mezi Mobilním zařízením a bluetooth. Opět funguje obousměrná komunikace o přijímání i odesílání dat. Na druhou stranu od řídí jednotky je možné vidět blok zastupující Motor shield. Tento blok je v praxi přidělán na řídící jednotku. Vpravo od bloku Motor shield jsou motory. Při návratu k řídící jednotce lze vidět další přídavné moduly. Systém obsahuje moduly výstupní. Jako jsou LED diody, které v systému slouží jako světelná výstraha pohybu. Nebo bzučák, zastupující zvukovou výstrahu. U posledního modulu se jedná o ultrazvukový senzor. Senzor získává data, která jsou k dalšímu zpracování odeslána do řídící jednoty.



Obrázek 20 Blokové schéma robota

Shrnutí základní funkce:

Z mobilního zařízení je odeslán příkaz o pohybu vlevo. Příkaz putuje přes Bluetooth komunikaci do řídící jednotky. Řídící jednotka předá informaci motor shieldu, který patřičně rozpohybuje motory. Ale také pošle příkaz do levé LED diody s informací, že má blikat. V průběhu celého procesu přijímá řídící jednotka informace o vzdálenostech od Ultrazvu-

kového senzoru. V případě možného nárazu je chod celého zařízení zastaven a je možné situaci opustit pouze vysouváním od objektu, který zastavení způsobil.

6.10.2 Elektrické schéma zapojení

Jde o znázornění celého elektrického odvodu s pomocí značek součástek a určitých pravidel. Schéma je dost podobné předchozímu znázornění zapojení. Avšak zde jde spíše o správné elektrické zapojení a zvýšení přehlednosti systému. Součástky se znázorňují smluvenými normalizovanými schématickými značkami. Spoje, které jsou vodivé se značí plnými čarami.

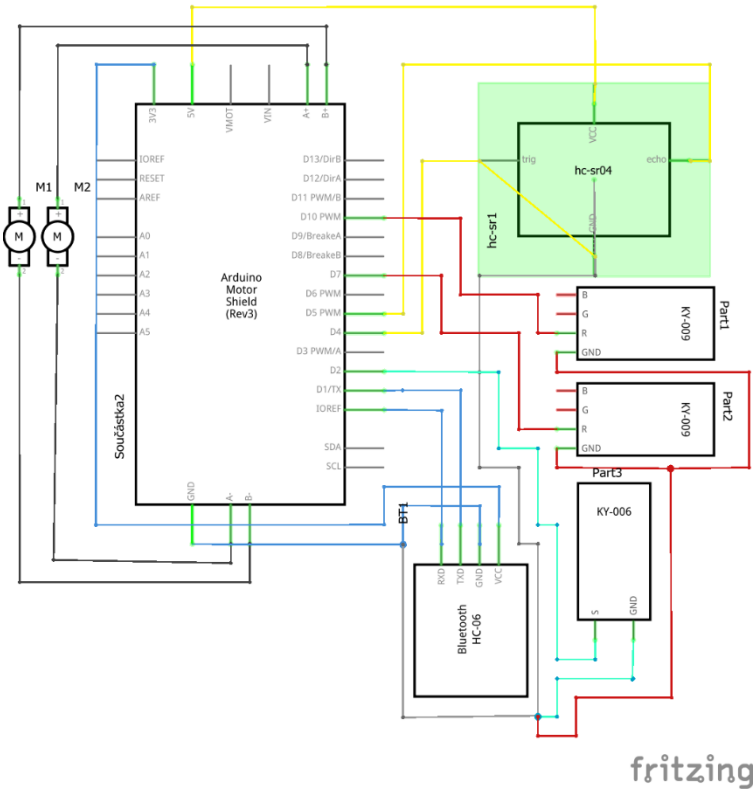


Schéma 10 Elektrické schéma zapojení

6.10.3 Nákres zapojení – montážní deska

Tento typ nákresu zapojení slouží především k lepší orientaci, s ohledem na barevnost propojovacích drátků a zachování velikostních poměrů. Podle tohoto nákresu lze snadno vytvořit stejné zapojení. Použité součástky odpovídají přesně dle nákresu.

Na obrázku je možné vidět hlavní řídící jednotku a tou je Arduino Uno. Přímo na něm jsou umístěny dva shieldy. První z nich je Motor shield a druhý, který není znázorněný

v nákresu je Terminal shield. Přímo z Motor shieldu, můžeme vidět vyvedené motory. Jako napájení je zvoleno připojení 9V baterie přes souosý konektor. Vstup pro tento konektor je umístěn přímo na základní desce Arduino Uno. Komponenty jako LED diody a bzučák jsou připojeny na digitální piny. Bluetooth je připojeno na piny pro sériovou komunikaci. Nepájivé pole je při návrhu použito z nedostatku zemnících pinů.

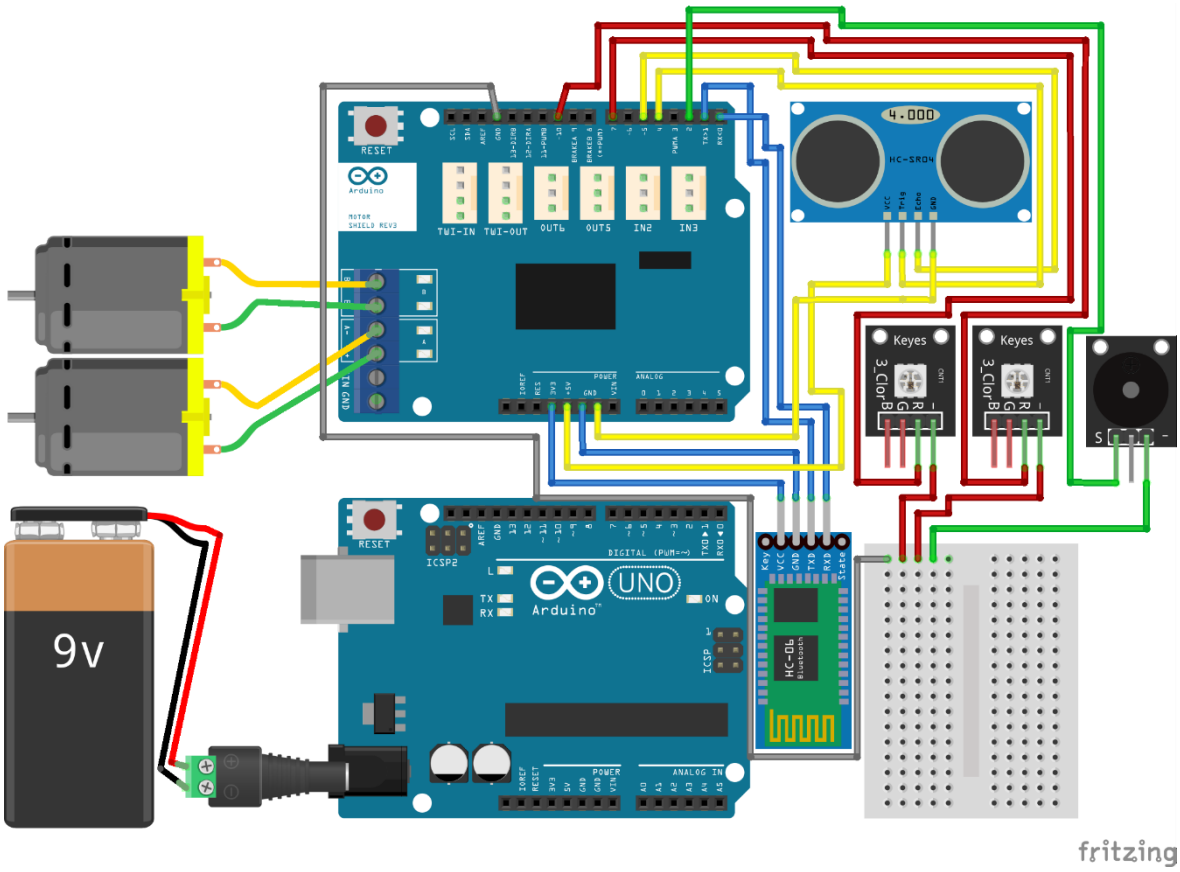


Schéma 11 Nákres zapojení na montážní desce

Pro větší přehlednost je schéma doplněno přehlednou tabulkou zapojení všech senzorů. Tabulka se skládá z typu modulu, zapojených pinů na modulu a jejich propojení do samotného Arduina.

Tabulka 1 Zapojení modulů

Modul	Piny modulu	Piny mikropočítače
Ultrazvukový senzor	VCC	+ 5 V
	Trig	4
	Echo	~5
	GND	GND
Bluetooth modul	Key	-
	VCC	+ 3,3 V
	GND	GND
	TXD	1
	RXD	0
	State	-
Levá LED	R	7
	GND	GND
Pravá LED	R	~10
	GND	GND
Bzučák	S	2
	GND	GND

7 SOFTWARE PRO ARDUINO

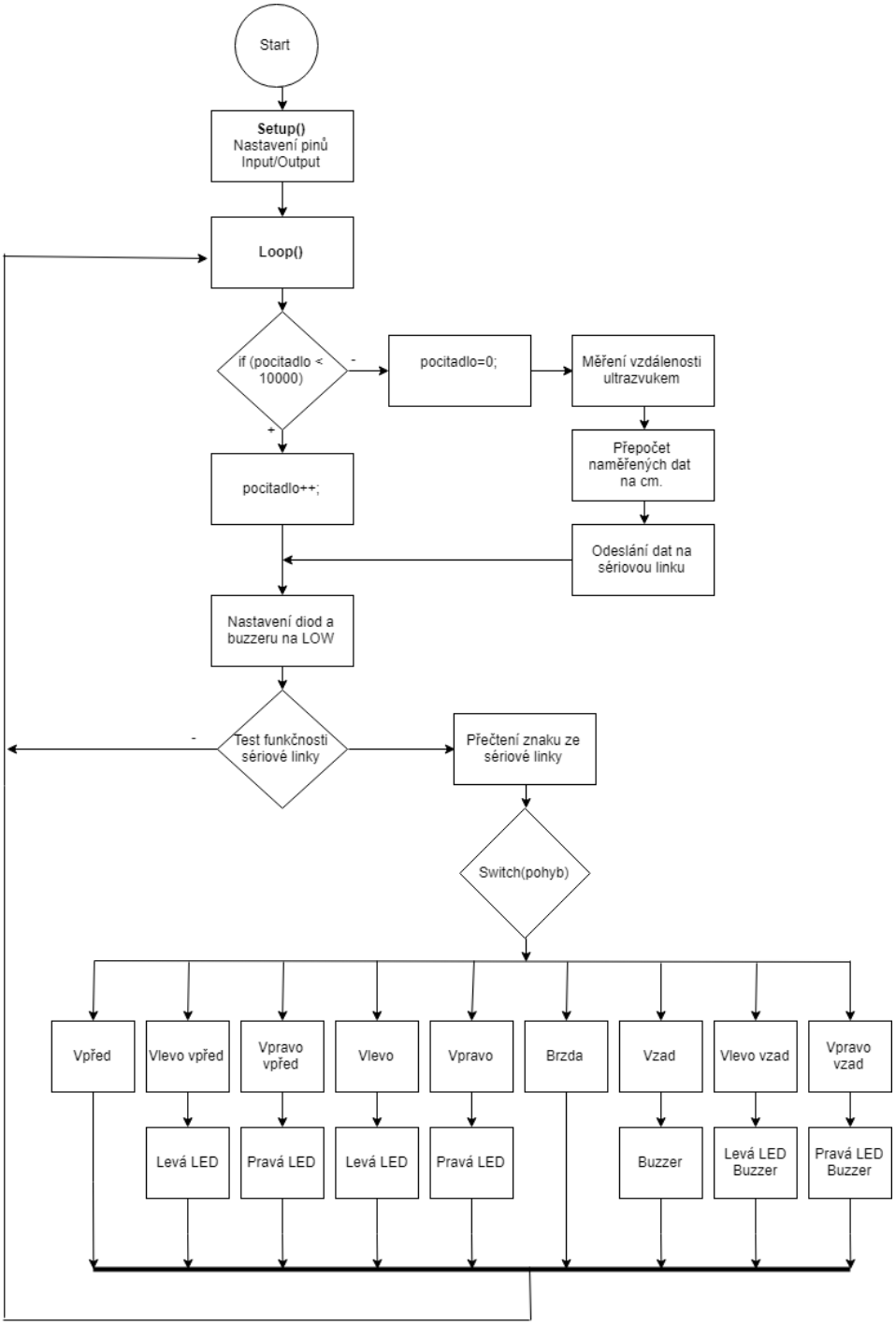
V této kapitole je popsáno programové vybavení pro vývojovou desku Arduino použitou pro řízení modelu robota.

7.1 Vývojový diagram

Vývojové diagramy slouží k popisu algoritmů a navazujících událostí na sebe. Včetně rozhodovacích akcí. Všechny události jsou znázorňovány geometrickými tvary jako jsou obdélníky, kosočtverce, kosodélník a další. Tyto objekty jsou propojovány šipkami, dle směrů úkonů a navazujících událostí.

Ve vývojovém diagram na následujícím obrázku č. 14 je znázorněn běh programu obsluhující mikropočítač Arduino.

Po spuštění Arduina dojde ve funkce *setup()* k nastavení všech pinů a počátečních stavů. Po doběhnutí této funkce se automaticky spustí funkce *loop()*, která běží neustále ve smyčce. V ní je implementováno počítadlo vnořené do podmínky. Když je hodnota počítadla menší, než 10 000, přičte se k hodnotě počítadla jeden bod. V případě že je hodnota vyšší, dochází k vynulování počítadla, změření vzdálenosti robota od objektu, přepočítání na měrnou hodnotu a odeslání dat na sériovou linku, která dopraví data do mobilního zařízení k prezentaci. V dalším kroku probíhá vypnutí všech diod a bzučáků. Dále je vyzkoušeno, zda na sériovou linku proudí nějaká data. Pokud ano, je vyhodnocen switch. Podle hodnoty přijaté ze sériové linky. Hodnoty můžou obsahovat číslice od 1–9. Pod každou číslicí jsou ukryty jiné úkony rozpohybují motor, či rozsvícení LED a pípání bzučáku. Tento algoritmus se neustále vyhodnocuje.



Obrázek 21 Vývojový diagram Arduino kódu

7.2 Popis kódu

Na začátek kódu je dobré definovat všechny potřebné knihovny a hned za nimi globální proměnné. V tomto projektu nejsou použity žádné knihovny. Celý kód je zcela samostatný. K testování, zda data proudí po sériové lince je možné použít softwarovo sériovou linku.

Tu je třeba definovat a také, použít třídu, která se nazývá *#include <SoftwareSerial.h>* a její definice je třeba taková *SoftwareSerial BT = SoftwareSerial(14,15)*. Využití softwarové sériové linky je mnohem rychlejší než pro první testování použít hardware.

```
int       pocitadlo   = 0;
char      pohyb       = 0;
float     odezva      = 0;
float     vzdalenost  = 0;
```

Ukázka kódu 14 Inicializace proměnných

V tomto kódu jsou definovány pouze globální proměnné. První je *pocitadlo*, tato proměnná slouží k tomu, aby nedocházelo k nadměrnému odesílání dat, které by mohlo způsobovat problémy se zahlcením. Proměnná typu *char*, s názvem *pohyb*, přijímá data ze sériové linky. *Odezva* je datového typu float, což je číslo včetně desetinných míst. Do této proměnné budou ukládána surová data z ultrazvukového senzoru. Za to proměnná *vzdalenost* obsahuje data, která jsou přepočítána na cm z proměnné *odezva.*

```
int const Proud_A  = 0; //SeriovaLinka
int const Proud_B  = 1; //SeriovaLinka
int const Dioda_A  = 2;
int const PWM_A    = 3;
int const TrigPin  = 5;
int const EchoPin  = 6;
int const Dioda_B  = 7;
int const Brzda_B  = 8;
int const Brzda_A  = 9;
int const Buzzer   = 10;
int const PWM_B    = 11;
int const Smer_A   = 12;
int const Smer_B   = 13;
```

Ukázka kódu 15 Definice konstant

V této části kódu jsou umístěné konstanty. Každá jedna konstanta odpovídá jednomu pinu umístěném na mikropočítači. Piny označené písmenem A se starají o jednu stranu motorů, piny označené písmenem B se starají o druhou stranu motorů. Konstanta *Buzzer* je pak pin, na který je připojen bzučák. Konstanty *TrigPin* a *EchoPin* se starají o odesílání a přijímání dat z ultrazvukového senzoru. Kdy Echo data odesílá a Trig data přijímá.

```
void setup() {
Serial.begin(9600);
pinMode(Smer_A,    OUTPUT);
pinMode(PWM_A,     OUTPUT);
pinMode(Brzda_A,   OUTPUT);
pinMode(Proud_A,   OUTPUT);
pinMode(Smer_B,    OUTPUT);
pinMode(PWM_B,     OUTPUT);
pinMode(Brzda_B,   OUTPUT);
pinMode(Proud_B,   OUTPUT);
pinMode(TrigPin,   OUTPUT);
pinMode(EchoPin,   INPUT);
pinMode(Dioda_A,   OUTPUT);
pinMode(Dioda_B,   OUTPUT);
pinMode(Buzzer,    OUTPUT);
}
```

Ukázka kódu 16 Setup()

Zde, ve funkci *void setup()* dochází k prvotnímu nastavování všech pinů a dalších prvků. Prvním řádkem funkce, je spuštění sériové komunikace s hardwarovým zařízením blueto-oth. Rychlost je nastavena na výchozí hodnotu, lze jí však zpomalit. V případě, že chceme použít softwarovou sériovou linku, je nutné implementovat i tento příkaz *BT.begin(9600)*.

Nastavování pinů probíhá definicí pomocí funkce *pinMode*, která je implementovanou součástí prostředí. Na každý pin tedy nastavíme, zda se jedná o výstupní, či vstupní pin. Namísto čísel pinů jsou zde použity konstanty, které byly na začátku kódu definovány.

Tento kód proběhne pouze jednou, a to při startu. Slouží tedy k prvotnímu nastavení. K běhu samotného programu se používá *void loop()*. Do kterého se vkládají funkcionality celého programu. Je možné také využití funkcí. Funkce se definují mimo *loop()* i *setup()*. Jsou samostatným prvkem, který se pouze zavolá. Funkce je možné použít i v *setup(),* jen se jejímu zavolání dočkáme pouze jednou.

```
void loop() {
    if (pocitadlo < 10000){
    pocitadlo++;
  }
  else{
    pocitadlo=0;
    digitalWrite(TrigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(TrigPin, HIGH);
    delayMicroseconds(5);
    digitalWrite(TrigPin, LOW);
    delayMicroseconds(5);
    odezva = pulseIn(EchoPin, HIGH);
    vzdalenost = (odezva * (340/2))/10000;
    Serial.println((String)vzdalenost+"~");
    }
```

Ukázka kódu 17 Začátek funkce loop()

Nyní se dostáváme do hlavní části celého programu a tou je *loop()*. Hned na prvním řádku narážíme, na již výše zmíněnou podmínku. Jedná se o ošetření proti nadbytečnému toku dat. Pokud je na počítadle hodnota menší, než 10 000, nic zásadního se nestane, pouze je hodnota počítadla zvýšená o 1. Ale pokud je hodnota větší, podmínka neprojde a jsme odkázáni na část kódu za klíčovým slovem *else*.

V první řadě je počítadlo opět přepsáno na hodnotu 0. Další na řadu přichází vyslání ultrazvukového signálu, kdy je na 2 milisekundy senzor pozastaven, pak znovu odeslán na 5 milisekund a potom opět vypnut. Po dalších 5 milisekundách je měřena délka odezvy, za kterou se signál od vysílače přes předmět v obzoru odrazí a vrací se zpět. Následně se změřená délka odezvy pomocí výrobcem definovaného vzorce přepočítá na vzdálenost v cm.

Kvůli rozlišení dat, odesílaných na sériovou linku, je nutné za každou odeslanou hodnotu vložit poznávací znamení. Něco, jakože nyní byla odeslána hodnota a za znakem již začíná hodnota nová. Pro tento případ je zvolena tilda. V sériové lince je proti číslům snadno rozeznatelná.

```
    digitalWrite(Dioda_B, LOW);
    digitalWrite(Dioda_A, LOW);
    digitalWrite(Buzzer, LOW);
```

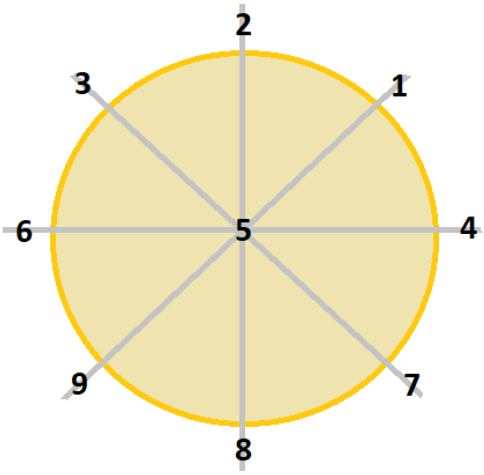
Ukázka kódu 18 Vypnutí diod a bzučáku

V každé otáčce smyčky, je nutné znamení nulovat, tedy vypnout. Jinak by docházelo k situaci, že hned po prvním couvání se spustí bzučák, který se již nikdy nevypne, a to ani při jízdě vpřed. Proto jsou hodnoty na pinech přepsány do stavu LOW-0.

```
if(Serial.available() > 0)
  {  pohyb = Serial.read();
     switch (pohyb)  {
     case '1': { motory(HIGH, HIGH, LOW, LOW, 155, 255);
                 digitalWrite(Dioda_B, HIGH);
                 break; }
     case '2': { motory(HIGH, HIGH, LOW, LOW, 150, 150);
                break; }
     case '3': {  motory(HIGH, HIGH, LOW, LOW, 255, 155);
                 digitalWrite(Dioda_A, HIGH);
                 break; }
     case '4': { motory(LOW, HIGH, HIGH, LOW, 255, 255);
                 digitalWrite(Dioda_B, HIGH);
                 break; }
     case '5': { motory(LOW, LOW, HIGH, HIGH, 155, 155);
                 break; }
     case '6': { motory(HIGH, LOW, LOW, HIGH, 255, 255);
                 digitalWrite(Dioda_A, HIGH);
                 break; }
     case '7': { motory(LOW, LOW, LOW, LOW, 255, 155);
                 digitalWrite(Buzzer, HIGH); digitalWrite(Dioda_B, HIGH);
                 break; }
     case '8': { motory(LOW, LOW, LOW, LOW, 255, 255);
                 digitalWrite(Buzzer, HIGH);
                 break; }
     case '9': { motory(LOW, LOW, LOW, LOW, 155, 255);
                 digitalWrite(Buzzer, HIGH); digitalWrite(Dioda_A, HIGH);
                 break; }
  } } }
```

Ukázka kódu 19 Nastavení chlívků pohybu

Zde se nachází nejpodstatnější část ovládání motorů. V rámci úspory místa je i více řádků vměstnaných do jednoho. První částí je vůbec zjištění, zda na sériovou linku přišel nějaký znak. Pokud ano, je ze sériové linky čtený jeden znak, který se uloží do proměnné *pohyb*. Ve switch se poté znak vyhodnotí. Je-li znak ‚2', spustí se oba motory stejnou rychlostí a robot jede rovně. Je-li znak ‚4', spustí se pouze jedem motor a robot se otáčí téměř na místě. Obrázek č. 15 znázorňuje jednotlivé pohyby a jejich značení, přicházející ze sériové linky. Krom ovládání motorů je možné ve switch najít také rozsvěcování diod, či spouštění bzučáku.



Obrázek 22 Rozdělení pohybu

V následující a poslední části Arduino kódu je funkce ovládající právě motory. Funkce je typu void. Což znamená, že se funkce provede, ale nevrací žádnou návratovou hodnotu. Vstupními parametry funkce jsou zapnutí směru jízdy na LOW nebo HIGH u motoru A i u motoru B. Odjištění a zajištění brzdy, opět stejnými klíčovými slovy pro oba motory. Nakonec analogovým zápisem nastavíme PWM modulaci na hodnotu od 0 do 255. Modulace zde reguluje rychlost jízdy. Čím je čísl větší, tím rychleji robot pojíždí.

```
void motory(char S_A, char S_B, char B_A, char B_B, int R_A, int R_B){
  digitalWrite(Smer_A, S_A);
  digitalWrite(Smer_B, S_B);
  digitalWrite(Brzda_A, B_A);
  digitalWrite(Brzda_B, B_B);
  analogWrite(PWM_A, R_A);
  analogWrite(PWM_B, R_B);
  }
```

Ukázka kódu 20 Funkce rozpohybování motorů

8 SOFTWARE PRO ANDROID

Tato část práce je zaměřena na programovou část pro mobilní zařízení s operačním systémem Android.

8.1 Android studio

Android studio je vývojové prostředí pro programování mobilních aplikací. V této kapitole je přehled základních prvků prostředí a ukázce vytvoření nového projektu.

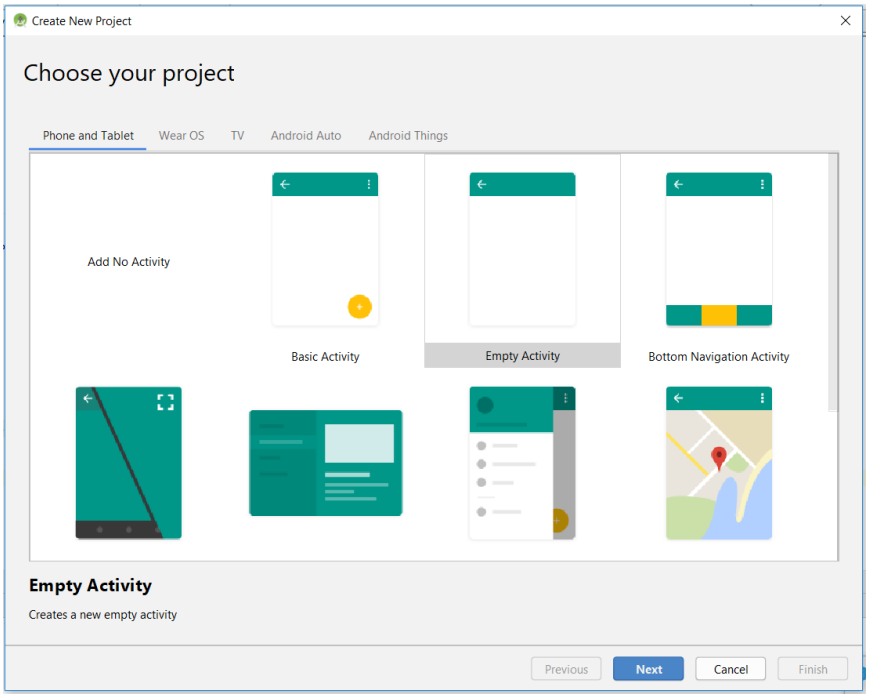
8.1.1 Emulátor

Testování naprogramovaných aplikací lze provádět dvěma způsoby. Buď je možné připojit vlastní mobilní zařízení pomocí USB nebo je možné použít emulátor. Emulátor simuluje zobrazení aplikace v mobilním zařízení. Dokonce je možné si vlastní mobilní zařízení nakonfigurovat. Simulátor poskytuje stejné možnosti testování jako samotné mobilní zařízení. Je možné dokonce simulovat telefonní hovory, SMS zprávy, určování polohy nebo navštívit obchod Google play. Emulátor je pro počítač větší zátěží než samotné Android studio, proto v případě starších, či méně výkonnějších počítačů je lepší využít mobilní zařízení. Naopak při výkonem počítači může být emulátor rychlejší a pohodlnější pro testování. Také je možné otestovat více druhů zařízení, které fyzicky není možné otestovat.

8.1.2 Vytvoření nového projektu

Při vytváření nového projektu v android studiu je hned prvním krokem, pro jaké zařízení budeme programovat a jak má vypadala obrazovka.

Na obrázku č. 16 jsou vidět záložky s typy zařízení pro které je možné programovat. Telefony a tablety, chytré hodinky, televize, aplikace do aut a další vychytávky. Po volbě záložky zbývá volba aktivity. Můžeme si vybrat od žádné, přes prázdnou po mapy atd.

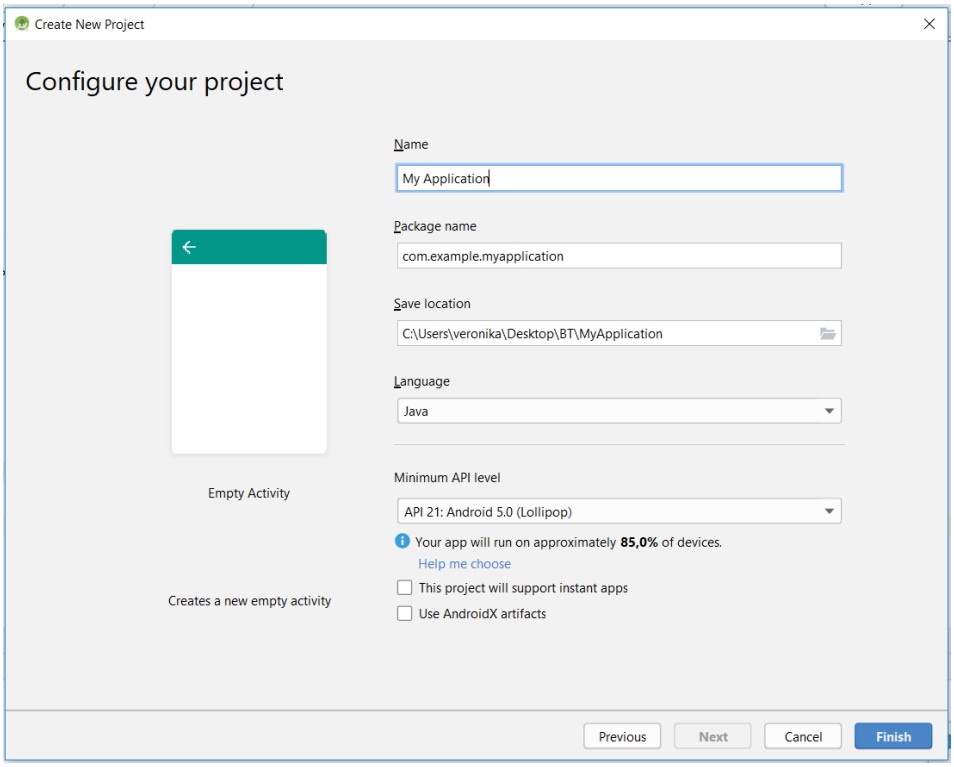


Obrázek 23 Volba aktivity

Klepnutím na tlačítko *next* se dostaneme do konfigurace projektu. Zvolíme si název projektu. Při změně názvu se samy přepíší i další řádky jako název balíčku a místo uložení projektu na disku.

Dalším bodem nastavení je jazyk. Lze si vybrat mezi tradiční Javou nebo využít novější Kotlin. Kotlin má svou syntaxí optimalizovat kód. V případě volby Kotlinu se Javě stejně vyhnout nelze. Minimálně využívané knihovny jsou psány v Javě. Někdy se může stát, že i samotný námi psaný kód bude nutné implementovat v jazyce Java.

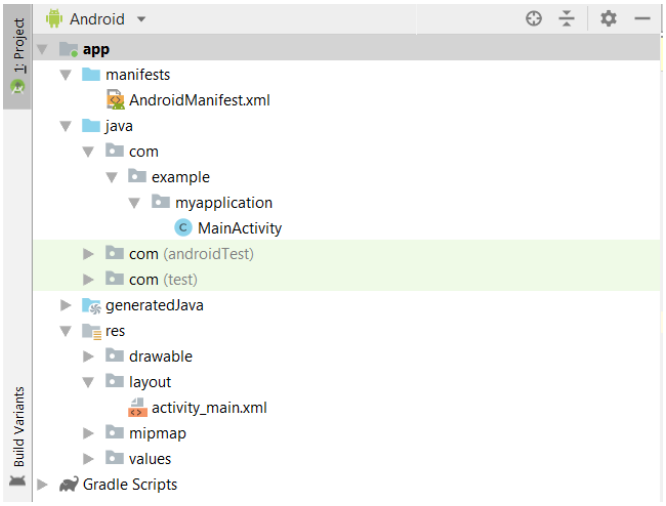
Minimální verze OS Android se volí z důvodu toho, že co naprogramujeme ve verzi pro OS 9.0 nemusí kvůli použitým funkcím a prvkům fungovat ve verzi 4.0. Proto je dobré si rozmyslet, na jaký trh cílíme. Zda na nová zařízení nebo na velké množství uživatelů. Je dobré zvolit volbu třeba 5 verzí zpět. Při volbě verze se zobrazí statistická informace o tom, na kolik procent lidí z celého trhu využívající OS Androind bude aplikace mířena. Poté stačí klepnout na tlačítko *finish* a nový projekt je vytvořen.



Obrázek 24 Vytváření projektu

Vytvořením nového projektu je vygenerována celá struktura projektu včetně všech nezbytných částí. Složka *app* je kořenovou složkou celého projektu. Zde se nachází další rozřazení složek jako je *manifest*, *java*, *generatedJava* a *res*. Ve složce *manifest* najdeme soubor informací a oprávnění *AndroidManifest.xml*. Ve složce *java* se nachází další dělení složek, kde v jejich konečných složkách najdeme soubory jako *MainActivity*. Kdyby projekt obsahoval více tříd, tak je najdeme právě zde, pospolu s touto třídou.

Další dvě složky řazené pod složkou *java* patří k testování. Najdeme zde jednotkové testy. Když se vrátíme o úroveň víš, tak najdeme složku *generatedJava*. V této složce najdeme soubor *BuildConfig*. V němž jsou všechna nastavení pro správný build. Tento soubor je generován automaticky, ale lze i ručně upravit. Složka *res*, tedy resource skrývá všechny zdroje. Jako podsložky najdeme *drawable*, v kterých jsou ukryty případné použité obrázky. Složku *layout,* v níž je skrytý celý návrh designu okna. V případě více oknové aplikace budou i další soubory tohoto typu uloženy zde. Složka *mipmap,* skrývá soubory s texturami a values skrývá soubory jako colors.xml, strings.xml, styles.xml. Jedná se o takové slovníky, v niž je možné si definovat vzhled, barvy, texty a v kódu je pouze používat. Případná změna v těchto prvcích je pak prováděna v jednom místě zde a není třeba procházet kód obzvláště při velkých projektech.



Obrázek 25 Stromová struktura projektu

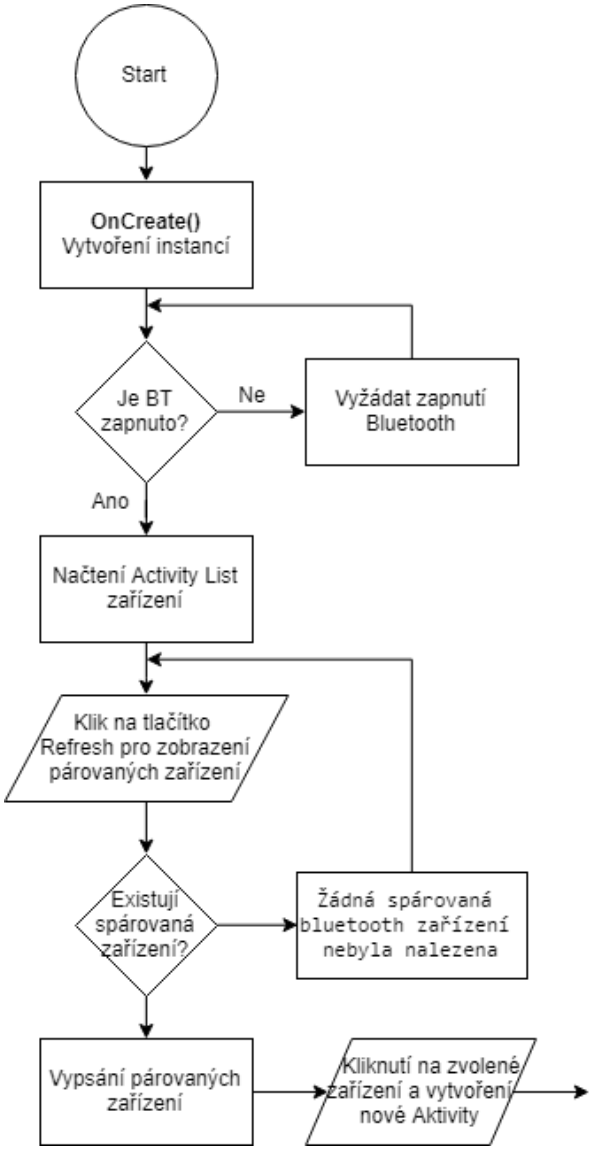
8.2 Programové vybavení

V této části budou popsány prvky samotné aplikace.

8.2.1 Vývojový diagram

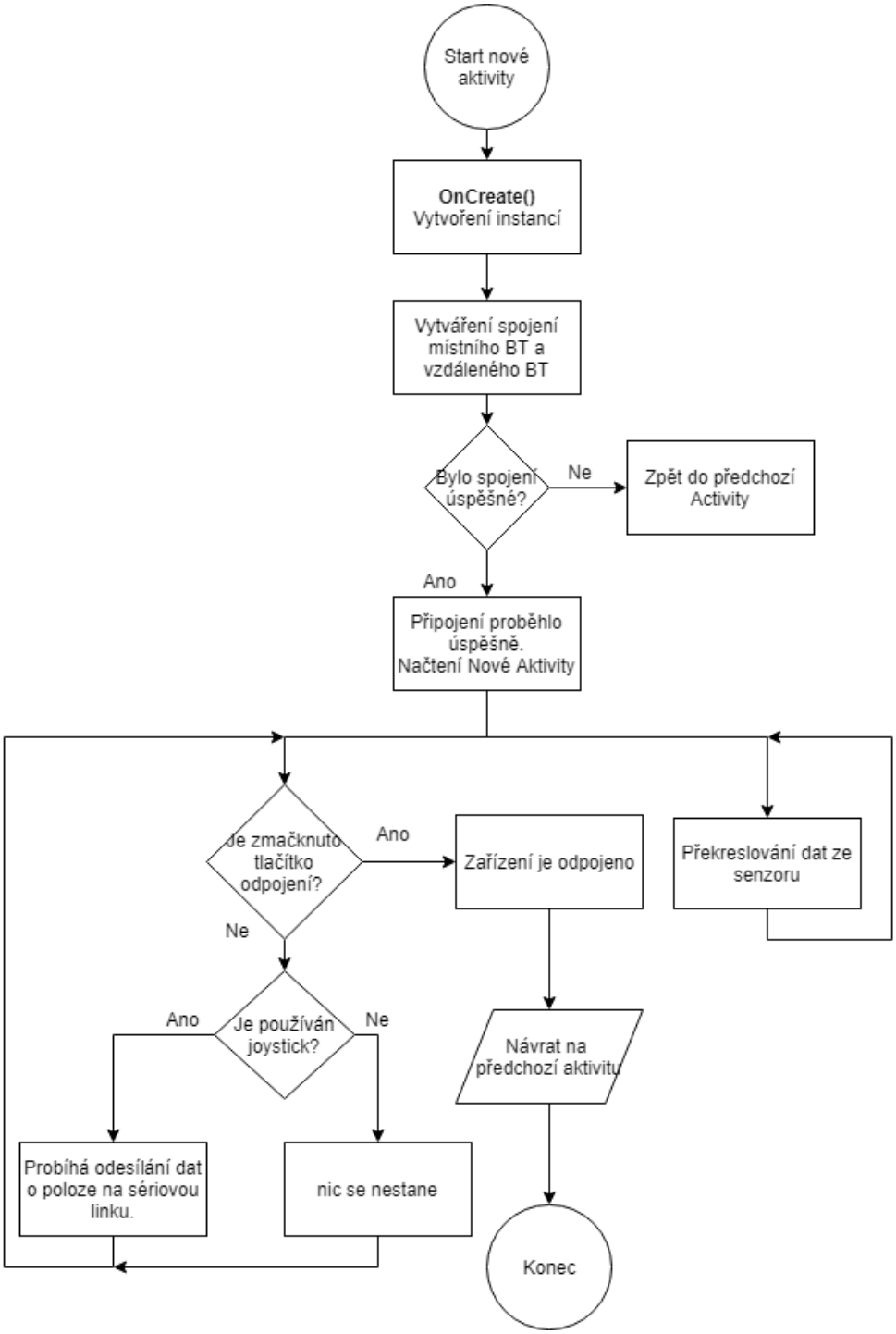
Vývojový diagram byl kvůli svým rozměrům a členitosti rozdělen na dvě části dle právě běžící aktivity. Diagramy poukazují pouze na hlavní členění programu. Podrobnější popis funkčnosti kódu je možné nalézt u jednotlivých úryvků kódu, které vysvětlují a naznačují jednotlivé problematiky a jejich posloupnosti. Pro lehký přehled projdeme hlavní funkcionality.

Po zapnutí aplikace v mobilním zařízení proběhne inicializace základních prvků běhu v metodě OnCreate(). Poté probíhá test na aktivitu vestavěného bluetooth zařízení. Pokud není, vynutí se zapnutí. Pokud je, načte se úvodní strana aplikace (aktivita). Na této aktivitě lze zvolit konkrétní zařízení, ke kterému se chceme připojit. Seznam se naplní spárovanými zařízeními po kliknutí na tlačítko *refresh*. Po kliknutí na zvolené zařízení, je vytvářena nová aktivita.



Obrázek 26 Vývojový diagram první Aktivity Android

V druhé části vývojového diagramu probíhá obdobný proces, kdy proběhne vytvoření inicializací ve funkci *OnCreate()*. Dále probíhá připojovaní místního a vzdáleného bluetooth zařízení. V případě nepovedeného připojení se program vrátí na předchozí aktivitu. Pokud se připojení povedlo, načte se nová aktivita, v které probíhá neustálé překreslování dat získaných ze senzorů. A na druhé straně probíhají akce ovládání joysticku, který ovládá robota. Všechno probíhá takřka současně ve více vláknech.



Obrázek 27 Vývojový diagram druhé aktivity Android

Vývojový diagram byl kvůli svým rozměrům a členitosti rozdělen na dvě části dle právě běžící aktivity. Diagramy poukazují pouze na hlavní členění programu. Podrobnější popis funkčnosti kódu je možné nalézt u jednotlivých úryvků kódu, které vysvětlují a naznačují jednotlivé problematiky a jejich posloupnosti.

8.2.2 Manifest

V souboru AndroidManifest.xml je možné nalézt různá oprávnění, přístupy k zařízením, aktivity a služby. Tento soubor je možné nalézt ve stromové struktuře projektu pod složkami app > manifest > AndroidManifest.xml. Soubor je odpovědný za celou ochranu vytvořené aplikace. Jde především o ochranu kvůli oprávněným nebo příliš častým poskytování oprávnění.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="e.Tank.bluetooth">

    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="bluetooth"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name="e.Tank.bluetooth.ListZarizeni">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="e.Tank.bluetooth.OvladaniTanku" />
    </application>

</manifest>
```

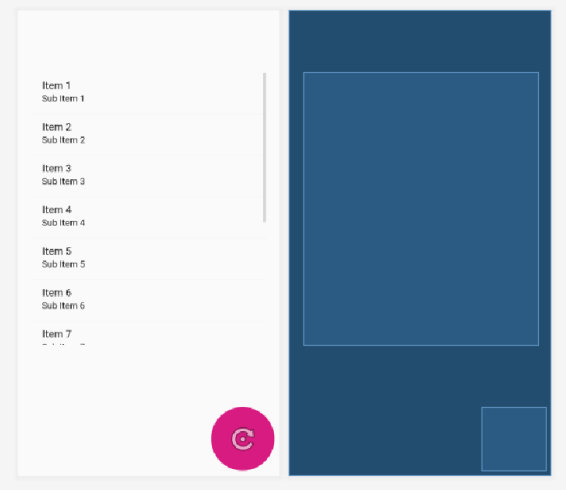
Ukázka kódu 21 Manifest

K obsahu ukázky kódu č. 31. Zde je možné vidět Manifest k vytvořené mobilní aplikaci. První řádek uvádí práci celého xml souboru je pouhým úvodem dokumentu. Druhý řádek, začínající klíčovým <manifest> je kořenovým prvkem celého manifestu. Obsahuje prvek package, který popisuje název celého balíčku. V tomto případě se jedná o `e.Tank.bluetooth`. Dalším prvkem struktury je uses-permission. Jedná se o přidělení oprávnění pro snazší práci s určitými prvky. Jako třeba zde použitý Bluetooth a Bluetooth_Admin. Tyto oprávnění zaručí nastavení bezproblémové využívání bluetooth modulu v mobilním zařízení. Přesněji `android.permission.BLUETOOTH` umožní aplikaci připojit se k již spárovaným zařízením a `android.permission.BLUETOOTH_ADMIN` dovoluje párovaní a vyhledávání zařízení v okolí s nimiž není navázáno spojení. V další části struktury se nachází application. Zde jsou deklarovány ikony, popisky, témata, spouštěcí režim a další. Ku příkladu `android:icon` deklaruje ikonu celého programu. `Android:theme` zase představuje téma stylu celého programu. Další nedílnou části tohoto manifestu je sepsání

všech potřebných Aktivit nebo je možno nazvat i jako obrazovky. Ze zápisu kódu lze vyčíst, že tento program disponuje dvěmi obrazovkami. První obrazovka se nazývá `e.Tank.bluetooth.ListZarizeni` a je používána jako úvodní obrazovka aplikace. Druhou obrazovkou je `e.Tank.bluetooth.OvladaniTanku`. Na této obrazovce se již odehrává veškeré dění aplikace.

8.2.3 Návrh designu seznamu zařízení

Jedná se o první obrazovku, kterou je možné vidět bezprostředně po spuštění programu. Jedná se o list spárovaných zařízení. Přesný název souboru je zarizeni_list.xml. Skládá se z designové části, ve které je možné komponenty přetahovat na obrazovku. A z části kódové, kde je možné komponenty přidávat, či upravovat kódem. Soubor je možné nalézt ve stromové struktuře složek app > layout > zarizeni_list.xml.



Obrázek 28 Design listu zařízení

Použité komponenty:

ConstraintLayout – Jedná se o komponentu, která má na starost zarovnání celého okna. Komponenta layoutu je důležitou součástí responsivních aplikací. Zaručuje flexibilitu celého okna. Při přepnutí do kódové části návrhu lze vidět, že se jedná o nadřazený prvek a ostatní komponenty jsou vnořeny uvnitř něho. Nejpodstatnějším nastavením této komponenty je, že její šířka a výška se určuje dle rozlišení zařízení.

FloatingActionButton – Tlačítko kulatého tvaru, působící, že neleží přímo na formuláři, ale trošku nad ním. Tak to je *FloatingActionButton*. Lze mu přiřadit velkou škálu vlastností. Mezi ty základní patří třeba barva, ale je možné přiřadit mu ikonku. Jak můžeme vidět na obrázku č. 21. Je použita fuchsiová barva s obrázkem naznačujícím *refresh*. Krom těch-

to vlastností má také nastavenou pevnou velikost 100 dp. Vzdálenost od dolního okraje a pravého okraje je také pevná a odpovídá 8 dp. Dále má tlačítko nastavenou vlastnost *clickable* na pravdu, aby bylo možné, na tlačítko kliknout.

ListView – Posledním prvkem okýnka je *ListView*, pro zobrazení spárovaných zařízení. Jedná se o prostý posouvací seznam prvků. Z nastavených vlastností stojí za zmínění opět vzdálenost komponenty od okraje, která je nastavena na 8 dp od horní a bočních částí.

8.2.4 Třída list zařízení

O této třídě lze říct, že obsluhuje aktivitu zarizeni_list.xml. Krom toho, připravuje zařízení na nadcházející pokus o spojení.

Použité knihovny

Ve třídě list zařízení jsou použity následující knihovny.

```
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Intent;
import android.support.design.widget.FloatingActionButton;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import java.util.ArrayList;
import java.util.Set;
```

Ukázka kódu 22 Importy

S každou importovanou knihovnou do programu, jsou přineseny nové funkcionality, které by bylo nutné v případě absence knihoven doplnit. Ku příkladu hned první knihovna ze seznamu je *BluetoothAdapter*. Tento adaptér je základním prvek při Bluetooth komunikaci, jelikož se stará o základní úkony. Stará se o vyhledávání zařízení, ověřování spárovanosti a vytvoření bluetooth socketu, pro naslouchání požadavků připojených zařízení. Naopak knihovna *BluetoothDevice* se stará o vzdálené zařízení. Slouží k vytvoření spojení se zvoleným zařízením, nebo je schopno informovat o názvu, adrese, stavu vazby či třídě. Knihovna Intent slouží jako lepící prvek při zahájení činnosti. Její obsah je abstraktním popisem akce, která bude provedena. Velká část importů z obrázku jsou podpůrnou knihovnou k jednotlivým použitým komponentám a zde slouží k vytvoření instance. Toast také patří do skupiny grafických komponent. Jedná se totiž o zběžnou informativní zprávu v dolní

části obrazovky. Může být použita třeba v případě, že se zdaří připojení. Na obrazovce se tedy na pár sekund zobrazí zpráva „připojeno“.

Definice třídy

Hned na začátku třídy ListZarizeni proběhla definice komponent a prvků obsluhující bluetooth zařízení.

```
FloatingActionButton btnAktualizovat;
ListView listzarizeni;

private BluetoothAdapter mojeBT = null;
private Set<BluetoothDevice> parovanaZarizeni;
public static String EXTRA_ADDRESS = "adresa_zarizeni";
```

Ukázka kódu 23 Definice prvků třídy

První dva řádky značí vytvoření proměnný typu *FloatingActionButton* a *ListView*, u kterých bude v následujících řádcích vytvořena instance na konkrétní komponenty z Aktivity *zarizeni_list*. Další dva řádky se připravují pro práci s Bluetooth zařízeními. Přičemž **mojeBT** má na starost bluetooth mobilního zařízení a **parovanaZarizeni** se bude starat o ty vzdálené. Poslední proměnná typu string je veřejná, slouží pouze k zobrazování a nebude se měnit.

Vytvoření instancí

Hlavním úkolem této části je uvedení potřebných komponent a vytvoření instancí mezi nimi, aby byly dostupné v rámci celé třídy.

```
protected void onCreate(Bundle ulozeniStavuInstance) {
    super.onCreate(ulozeniStavuInstance);
    setContentView(R.layout.zarizeni_list);
    btnAktualizovat = findViewById(R.id.btn_refresh);
    listzarizeni = findViewById(R.id.listview_paired_dev);
    mojeBT = BluetoothAdapter.getDefaultAdapter();
    if(mojeBT ==null) {
        Toast.makeText(getApplicationContext(), text: "Bluetooth neni podporovano",
                       Toast.LENGTH_LONG).show();
        finish();
    }
    else if(!mojeBT.isEnabled()) {
        Intent zapnutiBT = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(zapnutiBT, requestCode: 1);
    }
    btnAktualizovat.setOnClickListener((v) → { listParovanychZarizeni(); });
}
```

Ukázka kódu 24 Vytvoření instancí a počátečních inicializací

Tato funkce probíhá v případě vytváření nové instance Activity a tedy je odpovědná za první volání. Funkce je volaná i v případech, že nastala změna orientace zařízení z vertikální na horizontální a opačně.

V dalších řádcích jsou vytvářeny instance konkrétních komponent z activity zarizeni_list. Komponenty jsou přiřazovány dle unikátního ID. ID odpovídá zvolenému názvu. Do proměnné **btnAktualizovat** typu `FloatingActionButton` je plněna instance pomocí funkce, která najde komponentu právě podle ID. Parametrem je tedy název. Proměnná **listzarizeni** je plněna stejným způsobem, jen se svojí příslušnou komponentou odpovídající typu `ListView`.

Naplnění proměnné **mojeBT**, probíhá získáním výchozího bluetooth adaptéru, pokud nějaký je. Pokud ne, bude splněna následující podmínka. Upozornění na chybu bude zobrazeno pomocí Toastu, který oznámí, že zařízení není podporování. Pokud podmínka splněna nebude, znamená že je vše v pořádku a je možné prozatím pokračovat dál k další přímo navazující podmínce. Podmínka testuje, zda je zařízení dostupné. Pokud dostupné není, je spuštěn dotaz se systémovým dotazem na spuštění bluetooth.

Poslední částí vloženého kódu je funkce na kliknutí v listu. V případě kliknutí na nějakou volbu párovaného zařízeni bude zařízení přesměrováno do privátní funkce, ke které se dostaneme později.

Obsluha listu zařízení

Zde bude probíhat veškeré obhospodařování listu. Jedná se o privátní funkci typu void, tedy funkce dostupná pouze v této třídě a bez návratové hodnoty.

```
private void listParovanychZarizeni(){
    parovanaZarizeni = mojeBT.getBondedDevices();
    ArrayList list = new ArrayList();
    if(parovanaZarizeni.size()>0) {
        for(BluetoothDevice bt : parovanaZarizeni) {
            list.add(bt.getName() + "\n" + bt.getAddress());
        }
    }
    else {
        Toast.makeText(getApplicationContext(),
                text: "Zadna sparovana bluetooth zarizeni nebyla nalezena",
                Toast.LENGTH_LONG).show();
    }
    final ArrayAdapter adapter = new ArrayAdapter( context: this,
                                 android.R.layout.simple_list_item_1, list);
    listzarizeni.setAdapter(adapter);
    listzarizeni.setOnItemClickListener(vyberZListu);
}
```

Ukázka kódu 25 Obsluha listu párovaných zařízení

V druhém řádku jsou získány všechny spárovaná vzdálená zařízeni se zařízením místním (mobilním telefonem). Tyto zařízení nemusí být vůbec aktivní. Aktivita se neověřuje, pouze je „opsán“ systémový seznam párovaných zařízení.

Na dalším řádku probíhá prostá deklarace listu.

V nadcházející podmínce probíhá dotaz na velikost seznamu párovaných zařízení, tedy zda je počet `BluetoothDevice` větší než nula. Pokud je podmínka splněna, následuje for cyklus, který projde všechna párovaná zařízení a vloží jej do listu ve tvaru názvu a adresy. Pokud není podmínka splněna a počet párovaných zařízení je nula, vypíše se hláška o nedostupnosti spárovaných zařízení.

Dalšími řádky bude vytvořen adaptér pro zobrazení pole s určitým rozvržením. Nastavení obsahu list na podrobnosti adaptéru. A v posledním řádku je nastavena funkce pro kliknutí.

Volba zařízení

Poslední částí kódu této třídy je právě obsluha kliknutí a volba toho správného zařízení.

```
private AdapterView.OnItemClickListener vyberZListu = (av, v, arg2, arg3) → {
        String info = ((TextView) v).getText().toString();
        String adresa = info.substring(info.length() - 17);
        Intent i = new Intent( packageContext: ListZarizeni.this, OvladaniTanku.class);
        i.putExtra(EXTRA_ADDRESS, adresa);
        startActivity(i);
};
```

Ukázka kódu 26 Volba zařízení

První řádek `private AdapterView.OnItemClickListener vyberZListu = new AdapterView.OnItemClickListener()` označuje definici rozhraní zpětného volání, při akci kliknutí.

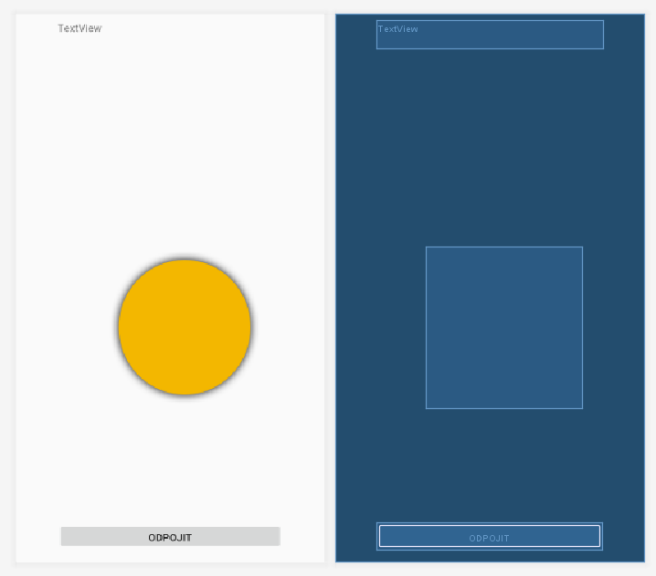
Další řádek definice `public void onItemClick(AdapterView<?> av, View v, int arg2, long arg3)` bude vyvolána při kliknutí na položku v adapteru.

Textové řetězce `info` a `adresa` jsou naplněny informacemi ze zvoleného adaptéru.

Na dalších řádcích již probíhá příprava na spuštění nové aktivity. Tedy je nutné vyvořit spojení za pomoci objektu typu Intent. Poté je aktivita změněna, předána adresa zvoleného zařízení a v poslední řadě je nová aktivita i spuštěna.

8.2.5 Návrh designu ovládání robota

Tato aktivita se spustí po volbě spárovaného zařízení. V okýnku je možné vidět TextView komponentu na níž se bude ukazovat vzdálenost objektů. Tato informace poputuje ze sériové bluetooth komunikace. Dále se v okně nachází joystick, který slouží jako vstup k ovládání celého hardwarového robůtka. Ve spodní části okna se nachází tlačítko pro ukončení spojení mezi zařízeními. Přesný název tohoto souboru je tank_ovladani.xml.



Obrázek 29 Design ovládání tanku

Použité komponenty:

ConstraintLayout – Constraint layou disponuje obdobným nastavením jako v předchozí aktivitě. I v tomto případu layout pohlcuje ostatní komponenty a tím tvoří responsibilitu aplikace.

TextView – TextView slouží k zobrazování dat přicházejících z bluetooth zařízení. Zobrazuje se jako prostý text o velikosti 24 sp. Mezi nastavené vlastnosti patří ukotvení komponenty k levé, horní a pravé straně ConstraintLayoutu ve vzdálenosti 8 dp. V případě potřeby použití dalších designových nastavení nabízí komponenta velkou škálu vlastností.

RelativeLayout – RelativeLayout je využit k hraničnímu poli joysticku. Má pevnou velikost a nastavené pozadí v podobě obrázků žlutého kruhovitého prostoru. Tento obrázek je čerpán ze složky app > res > drawable a v kódovém nastavení layoutu je pouze přiřazen: `android:background="@drawable/image_button_bg"`. Jako zdrojový obrázek je použit formát souboru PNG s průhledným pozadím.

Button – Button zde bude plnit funkci odpojení od připojeného bluetooth. Jedná se o klasický button bez zbytečných příkras. Opět ukotven, kvůli responsibilitě. Další vlastností je nastavený text, aby bylo jasně rozpoznatelné, jakou funkci tlačítko vykoná.

Třída ovládání tanku

Importy jsou opět obdobné. Krom těch, co jsou použité v předchozí třídě list zařízení, jsou použity ještě tyto nové knihovny. Hned první knihovnou je SuppressLint. SuppressLint má na starost soubor výstrah, které mají být ignorovány. Tyto výstrahy jsou rozeznávány podle ID.

```
import android.annotation.SuppressLint;
import android.app.ProgressDialog;
import android.bluetooth.BluetoothSocket;
```

Ukázka kódu 27 Importy

ProgressDialog dodává do aplikace nové možnosti při načítaní dat. Ve chvíli, kdy má být aplikace nečinná, je právě tohle řešením. Například se na pozadí vytváří spojení a cílem je, aby uživatel svými úkony tento stav nenarušil. Toto dialogové okno zabrání uživateli na cokoliv kliknout. Zobrazí se volitelná zpráva nebo nějaký jiný způsob zobrazení a okno se v průběhu přenosu zneaktivní. V případě, že chceme přenos dat přece jen i přes hlášku ukončit. Stačí kliknout na tlačítko telefonu s funkcí zpět.

BluetoothSocket použitý na straně klienta slouží k inicializaci odchozího spoje a ke správě připojení. Jedná se o obdobné rozhraní jako je TCP. Pomocí této knihovny tedy probíhá připojení ke vzdálenému zařízení.

První tři řádky následující ukázky kódu vytváří proměnné typu jednotlivých komponent. Jsou připravovány k propojení s konkrétními komponentami a jejich následovného ovládání. Proměnná *adress* je obyčejný řetězec znaků, který je v tento okamžiky vynullován. Dalším řádkem si vytváříme proměnnou zastupující práci s dialogovým oknem. Vytvoření prázdných proměnných *BluetoothSocket* a *BluetoothAdapter*. Privátní proměnné typu boolean pro kontrolu, zda je zařízení připojeno (isBtConnected) a zda je v dostatečné vzdálenosti od objektu (canGo). Privátní proměnná *bluetoothIn*, bude složit k překreslování okýnka ve vlastním vlákně. Proměnná *handlerState* slouží k indikaci typu zprávy. *StringBuilder* bude součástí dělení *stringu* přicházejícího ze sériové linky. Součástí běhu vlákna na pozadí je *vlaknoPripojeni*. Pomocí kterého bude vlákno vytvořeno a spuštěno. Proměnná *js*, je pouhou instancí třídy *JoyStickClass*.

```
Button Discnt;
RelativeLayout layout_joystick;
TextView TV;
String address = null;
private ProgressDialog progress;
BluetoothAdapter myBluetooth = null;
BluetoothSocket btSocket = null;
private boolean isBtConnected = false;
private boolean canGo = true;
private Handler bluetoothIn;
final int handlerState = 0;
private StringBuilder recDataString = new StringBuilder();
private ConnectedThread vlaknoPripojeni;
JoyStickClass js;
```

Ukázka kódu 28 Nastavení třídy

Na následujícím řádku kódu můžeme vidět doplnění neměnného univerzálního jedinečného identifikátoru třídy *UUID*. Tento identifikátor disponuje 128bitovou hodnotou. Tedy i když modul vyměníme, tato hodnota bude stále stejná.

```
static final UUID myUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
```

Ukázka kódu 29 UUID

Na začátku funkce *onCreate* dochází, jak již v předešlé třídě k přechodu mezi starým a novým okýnkem a je vytvářena instance nové třídy. Krom tohoto přepnutí mezi třídami je nutné předat i adresu zvoleného zařízení. Dále jsou tradičně vytvářeny instance použitých komponent tlačítka *Odpojit* a textového pole zobrazující data jdoucí z ultrazvukového senzoru přes sériovou linku, až do tohoto *TextView*.

```
protected void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    setContentView(R.layout.tank_ovladani);
    Intent newint = getIntent();
    address = newint.getStringExtra(ListZarizeni.EXTRA_ADDRESS);
    Discnt = findViewById(R.id.btn_disconnect);
    TV = findViewById(R.id.tv_prijate);
```

Ukázka kódu 30 Prvotní definice

K této aktivitě je nutné připojit ještě třídu obsluhující joystick. Na úryvku kódu je znázorněné vytvoření instance komponenty *RelativeLayout* a třídy *JoyStickClass* včetně vytvoření ovládacího tlačítka. Dále jsou nastavovány jednotlivé velikosti, vzdálenosti a hraniční body.

```
layout_joystick = (RelativeLayout)findViewById(R.id.layout_joystick);
System.out.print(layout_joystick);
js = new JoyStickClass(getApplicationContext(), layout_joystick, R.drawable.image_button);
js.setStickSize( width: 150,  height: 150);
js.setLayoutSize( width: 800,  height: 800);
js.setLayoutAlpha(125);
js.setStickAlpha(125);
js.setOffset(90);
js.setMinimumDistance(50);
```

Ukázka kódu 31 Připojení a definice třídy joystick

Jak již v předchozím případě, i zde se vyvolává kliknutím metoda zpětného volání *setOnTouchListener.* Kde podle polohy tlačítka je rozhodnuto o funkci, která bude použita. Jednotlivé funkce slouží k rozhodnutí o pohybu. Rozhodnutí o tom, v jaké části pole se tlačítko nachází je skryto ve funkci *chlivkovani.* Tato funkce je dostupná ve třídě *JoystickClass*. Pomocí podmínek *else if* je rozhodnuto, kterým směrem bylo tlačítko vysláno.

```
layout_joystick.setOnTouchListener((arg0, arg1) → {
        js.drawStick(arg1); int direction = js.chlivkovani();
        if(direction == JoyStickClass.Akce_V) {
            goV();
        } else if(direction == JoyStickClass.Akce_VP) {
            goVP();
        } else if(direction == JoyStickClass.Akce_P) {
            goP();
        } else if(direction == JoyStickClass.AkceZP) {
            goZP();
        } else if(direction == JoyStickClass.Akce_Z) {
            goZ();
        } else if(direction == JoyStickClass.Akce_ZL) {
            goZL();
        } else if(direction == JoyStickClass.Akce_L) {
            goL();
        } else if(direction == JoyStickClass.Akce_VL) {
            goVL();
        } else if(direction == JoyStickClass.Akce_B) {
            goB();
        }
        return true;
});
```

Ukázka kódu 32 Akce na klik

Zde probíhá nové volání třídy *ConnectBT* pro vytvoření nového připojení. Naopak dalším řádkem je definováno ukončení spojení na akci kliknutí. Toto spojení je ukončeno odpojením od vzdáleného zařízení bluetooth.

```
new ConnectBT().execute();
Discnt.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Disconnect();
    }
});
```

Ukázka kódu 33 Připojení, odpojení

V další části kódu je možné vidět *handler* obsluhující příchozí data a překreslování GUI. *Handler* dokáže odesílat a zpracovávat objekty runable. Každou instancí *handleru* je vytvořeno nové vlákno programu. Jakmile handleru přijde vstupní informace, začne jí okamžitě zpracovávat. V ukázce kódu je dotaz na to, co handler obsahuje. Když je vše v pořádku a podmínka projde, je vytvořena proměnná, která je následně plněna řetězcem přicházejícím ze sériové linky. Tato surová data jsou testována, zda obsahují znak *tilda (~)*, jelikož data z Arduina přichází ve formě „24,3~26,1~*27,6~*“. V první řadě je testována nultá pozice, zda není nutné *string* o tuto pozici posunout. Pokud ano, bude podmínka splněna a bude vytvořen *substring*. Dále je v novém řetězci vyhledáváno, na jaké pozici se další rozdělovací znak nachází. Index nalezeného znaku specifikuje konec čísla. Proto jsou data k vytisknutí v *TextView* vytvořeny z původního řetězce na indexech od pozice nula, po pozici ukončovacího znaku. Vypreparované číslo je pote prezentováno v kontextu ve větě.

```
bluetoothIn = new Handler() {
    public void handleMessage(android.os.Message msg) {
        if (msg.what == handlerState) {
            String readMessage = (String) msg.obj;
            recDataString.append(readMessage);
            if (recDataString.charAt(0) == '~') {
                recDataString = new StringBuilder(recDataString.substring( start: 1));
            }
            int endOfLineIndex = recDataString.indexOf("~");
            if (endOfLineIndex > 0) {
                String dataInPrint = recDataString.substring(0, endOfLineIndex);
                TV.setText("Vzdálenost od objektu je " + dataInPrint + " cm");
                if (dataInPrint!="")
                if (Float.parseFloat(dataInPrint) > 10){
                    canGo=true;
                }else{
                    canGo=false;
                }
                recDataString.delete(0, recDataString.length());
            }
        }
    }
```

Ukázka kódu 34 Handler

Další podmínka se stará o trochu něco jiného, než je prezentace dat. Zde je zjišťováno, zda se robot nachází dále než 10 cm od objektu do kterého by mohl narazit. K této funkcionalitě je vytvořena globální proměnná, která říká pouze *ano* nebo *ne*. Podle této proměnné je dále rozhodováno, zda budou odcházet data do sériové komunikace. Dále jsou data smazány.

```
private void Disconnect()
{
    if(btSocket!=null)
    {
        try
        {
            btSocket.close();
        }
        catch (IOException e)
        {
            msg( e: "Error" + e.getMessage());
        }
    }
    finish();
}
```

Ukázka kódu 35 Odpojení od připojeného zařízení

Metoda *Disconnect()* slouží k vyrušení propojení místního a vzdáleného bluetooth. Zařízení se odpojí pomocí ukončení propojení *btSocket*. V případě že se zařízení nepodaří odpojit, vypíše se chybová hláška. Nakonec se pomocí ukončení aktuální aktivity navrátíme zpět na list zařízení.

```
public void goV()
{
    if(btSocket!=null)
    {
        try
        {   if (canGo == true) {
            btSocket.getOutputStream().write("2".toString().getBytes());}
        }
        catch (IOException e)
        {
            msg( e: "Error" + e.getMessage());
        }
    }
}
```

Ukázka kódu 36 Funkce příkazu jízdy vpřed

Funkce *goV()* je jednou z 9 funkcí obstarávající směr jízdy. Každá jednotlivá funkce směru má téměř stejný obsah. První je test na vytvořené spojení a vnitřním obsahem je odeslání dat na sériovou linku. Vždy je odesláno patřičné číslo. Pro jízdu vpřed je odesláno číslo 2. Vzhledem k tomu, že odchozí stream není možné odeslat jako číslo, je nutné odeslání v podobě textového řetězce *string*.

Konkrétně tato funkce, odpovídající jízdě vpřed a ještě další dvě funkce odpovídající úhlopříčné jízdě jsou obohaceny podmínkou. Zde je právě prováděna kontrola, na data přijaté z ultrazvukových senzorů. Pokud jsou data ultrazvukových senzorů menší než 10 cm, je proměnná *canGo* naplněna hodnotou false. Tedy stůj. Pokud je hodnota větší než 10 cm, je proměnná naplněná hodnotou true a je tedy možné, data vyslat na sériovou linku.

```
private class ConnectBT extends AsyncTask<Void, Void, Void>
{
    private boolean ConnectSuccess = true;
    @Override
    protected void onPreExecute(){
        progress = ProgressDialog.show( context: OvladaniTanku.this,
                title: "Připojování...",  message: "Čekejte..");
    }
```

Ukázka kódu 37 Připojování Bluetooth

Připojování bluetooth probíhá na pozadí. Mezi tím je na obrazovce viditelná *progressMessage* informující o stavu, tedy připojování. Připojování začíná testováním, zda již není vytvořený bluetooth socket nebo zda není bluetooth připojeno. Do proměnné *myBluetooth* je uloženo místní bluetooth zařízení, tedy adaptér telefonu. Poté pomocí adresy je zjišťováno vzdálené zařízení. Do btSocketu je vytvořeno RFCOMM spojení se vzdáleným zařízení, Arduinem. Následuje ukončení vyhledávání a zahájení připojení. Poté je vytvořeno a spuštěno hlavní vlákno běhu. Pokud propojování selže, spadne program do výjimky a bude vypsána chybová hláška.

```
private class ConnectBT extends AsyncTask<Void, Void, Void>
{  private boolean ConnectSuccess = true;
    @Override
    protected void onPreExecute(){
        progress = ProgressDialog.show( context: OvladaniTanku.this,
                title: "Připojování...",  message: "Čekejte.."); }
    @Override
    protected Void doInBackground(Void... devices)
    { try
        { if (btSocket == null || !isBtConnected)
            {   myBluetooth = BluetoothAdapter.getDefaultAdapter();
                BluetoothDevice dispositivo = myBluetooth.getRemoteDevice(address);
                btSocket = dispositivo.createRfcommSocketToServiceRecord(myUUID);
                BluetoothAdapter.getDefaultAdapter().cancelDiscovery();
                btSocket.connect();
                vlaknoPripojeni = new ConnectedThread(btSocket);
                vlaknoPripojeni.start();
            }
        }
        catch (IOException e) {
            ConnectSuccess = false;
            msg( e: "Připojení selhalo:" + e.getMessage());
        } return null;
    }
```

Ukázka kódu 38 Připojovací vlákno

8.2.6 Třída manipulace s joystickem

Nové knihovny oproti předchozím třídám jsou i zde. Jedná se o grafické knihovny, které umožňují práci s bitmapami. Knihovna BitmapFactory umí vytvářet nové bitmapové objekty z různých zdrojů. Je možné využívat externí soubory, datové proudy a pole. Knihovna *Canvas* obsahuje funkce vykreslování do plátna. Třída *paint* obsahuje informace a možnosti vykreslování textů, geometrických tvarů nebo bitmap.

```
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Paint;
```

Ukázka kódu 39 Importy

Proměnná stick je naplněna bitmapou. Z této bitmapy je zjištěna šířka a výška. Pomocí DrawCanvas je vykreslována ovládací „páčka“ v aplikaci puntík.

```
public JoyStickClass (Context context, ViewGroup layout, int stick_res_id) {
    mContext = context;
    stick = BitmapFactory.decodeResource(mContext.getResources(), stick_res_id);
    stick_sirka = stick.getWidth();
    stick_vyska = stick.getHeight();
    draw = new vykresleni(mContext);
    paint = new Paint();
    mLayout = layout;
    params = mLayout.getLayoutParams();
}
```

Ukázka kódu 40 Počáteční nastavení třídy

V následující funkci jsou vypočítávány souřadnice polohy ovládacího puntíku. Zvlášť je vypočítána pozice X a pozice Y. Poté je z obou poloh dopočítána vzdálenost a nakonec úhel. Na výpočet úhlu je napsána funkce, která nám zodpoví výsledek.

```
public void drawStick(MotionEvent arg1) {
    pozice_x = (int) (arg1.getX() - (params.width / 2));
    pozice_y = (int) (arg1.getY() - (params.height / 2));
    distance = (float) Math.sqrt(Math.pow(pozice_x, 2) + Math.pow(pozice_y, 2));
    angle = (float) uhel(pozice_x, pozice_y);
```

Ukázka kódu 41 Výpočet pozice stick

Kód pro výpočet pohybu není z důvodu přehlednosti vložen. Je však možné jej najít v přiložené příloze zdrojových kódů. Pro úplnost bude předveden alespoň výpočet. Výpočet probíhá pro každou souřadnici zvlášť. Pro příklad ukázka výpočtu *x*: **float** x = (**float**) (Math.*cos*(Math.*toRadians*(uhel(**position_x**,**position_y**)))*((**params.width**/2)-**OFFSET**)); Nakonec je k tomuto výpočtu přičtena ještě polovina šířky puntíku. U souřadnice *y*, by se

jednalo o polovinu výšky. Výsledná data *x* a *y* jsou dosazena pro funkce na vykreslení puntíku.

Výše popsané vykreslení proběhne v případě, běží-li událost *MotionEvent.ACTION_DOWN*. V případě povolení kontaktu s obrazovkou proběhne událost *MotionEvent.ACTION_UP*. Na tuto událost bude puntík simulující ovládání joysticku vymazán z obrazovky.

```
public int chlivkovani() {
    if(distance > min_vzdalenost && dotyk) {
        if(angle >= 247.5 && angle < 292.5 ) {
            return Akce_V;
        } else if(angle >= 292.5 && angle < 337.5 ) {
            return Akce_VP;
        } else if(angle >= 337.5 || angle < 22.5 ) {
            return Akce_P;
        } else if(angle >= 22.5 && angle < 67.5 ) {
            return AkceZP;
        } else if(angle >= 67.5 && angle < 112.5 ) {
            return Akce_Z;
        } else if(angle >= 112.5 && angle < 157.5 ) {
            return Akce_ZL;
        } else if(angle >= 157.5 && angle < 202.5 ) {
            return Akce_L;
        } else if(angle >= 202.5 && angle < 247.5 ) {
            return Akce_VL;
        }
    } else if(distance <= min_vzdalenost && dotyk) {
        return Akce_B;
    }
    return 0;
}
```

Ukázka kódu 42 Rozřazovací funkce

Funkce s poetickým názvem *chlívkování* má na starost rozřazení toho, jaká část pole ještě patří do určitého směru a jaká už ne. Podle toho, který chlívek byl zvolen, bude přiřazena i patřičná hodnota. Pokud hodnota nesplňuje podmínku, je odeslána hodnota 0.

```
public void setLayoutSize(int width, int height) {
    params.width = width;
    params.height = height;
}
```

Ukázka kódu 43 Nastavení rozměru joysticku

Funkce nastavující rozměry pozadí. Tedy ohrádky, v které se pohybuje *stick*. Stejným způsobem jsou nastavovány i rozměry *sticku*.

```
private double uhel(float x, float y) {
    if(x >= 0 && y >= 0)
        return Math.toDegrees(Math.atan(y / x));
    else if(x < 0 && y >= 0)
        return Math.toDegrees(Math.atan(y / x)) + 180;
    else if(x < 0 && y < 0)
        return Math.toDegrees(Math.atan(y / x)) + 180;
    else if(x >= 0 && y < 0)
        return Math.toDegrees(Math.atan(y / x)) + 360;
    return 0;
}
```

Ukázka kódu 44 Výpočet úhlu

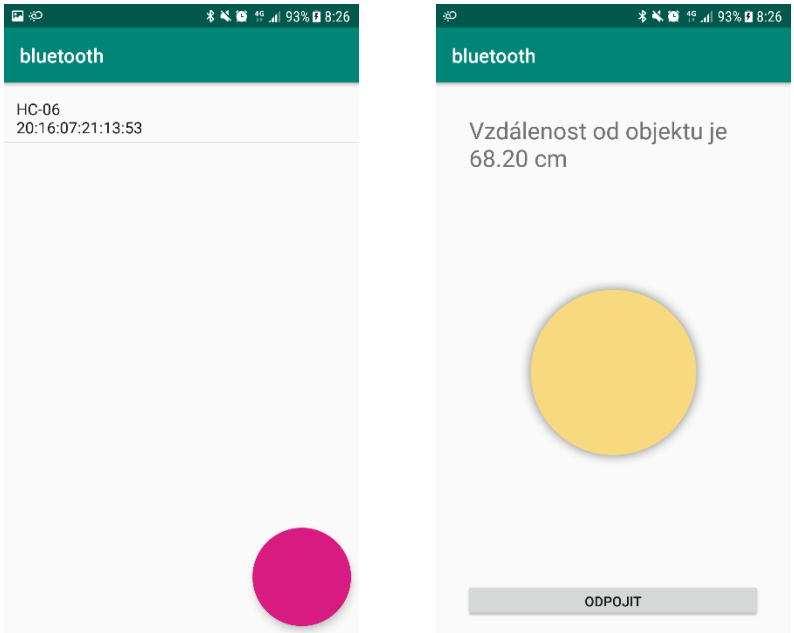
Výpočet úhlů, z kterého se vychází v případě *chlívkování*. K výpočtu jsou použity goniometrické funkce z matematické knihovny. Tento výsledek je poté převeden do stupňů a dopočítán do kruhovitého modelu.

9 OVĚŘENÍ PRAKTICKÉ ČÁSTI

9.1 Zprovoznění celého systému

K zprovoznění robota je nutné vlastnit mobilní zařízení s operačním systémem Android. Tedy tablet nebo mobilní telefon. Dále je nezbytné připojit robota ke zdroji. Jako zdroj jsou použity dva kusy 9 voltové baterie. Před samotným spuštěním je vhodné ověřit, zda je bluetooth na mobilním zařízení spuštěné, nebo dokonce spárované. Tento postup usnadní problémy, které by vznikali v průběhu spouštění. Párování zařízení je popsáno v další kapitole s číslem 9.1.1.

Když jsou všechny tyto kroky splněny, je možné začít. Po spuštění aplikace, se jako první zobrazí obrazovka se spárovanými zařízeními, vždy ve formátu název a adresa. V pravém dolním rohu je umístěné tlačítko *refresh*, kterým je možné seznám zaktualizovat. Výběrem zařízení jsme odkázáni na novou obrazovku (aktivitu). Na této obrazovce již běží ultrazvukový senzor, který v celém průběhu aktualizuje svá data. Uprostřed obrazovky je umístěný joystick, pomocí kterého je zařízení možné ovládat. Ve spodní části je pak uloženo tlačítko pro odpojení od aktuálně zvoleného zařízení. Při odpojení jsme opět vráceni na původní obrazovku s listem zařízení.



Obrázek 30 Obrázky běžících aplikací

9.1.1 Propojení pomocí Bluetooth

Jak už bylo psáno výše, pro připojení pomocí bluetooth je nutné mít zařízení spárovaná. Párování probíhá tak, že na mobilním zařízení zvolíme možnost hledání dostupných zařízení v okolí. Zvolíme příslušné bluetooth náležící robotu. Které musí být v okamžik vyhledávání také aktivní. Naše mobilní zařízení si vyvolá požadavek na zadání ověřovacího kódu. Jelikož se jedná o zařízení bez displaye, tak bývá kód ve výchozím nastavení čísla: 1234 nebo čtyři 1. Tento kód lze přeprogramováním i změnit. Po vepsání a potvrzení správného kódu jsou již zařízení spárovaná.

9.2 Ochrana proti nárazu

Jako ochrana proti nárazu je použit ultrazvukový senzor. Který by měl měřit vzdálenost od několika set metrů po 2 cm. Realita je však jiná a senzor funguje lépe na krátké vzdálenosti. Při jednotkách až desítkách centimetrů dosahuje velice uspokojivých výsledků. Tyto výsledky jsou pro použití jako ochrany proti nárazu dostačující, jelikož je kontrolována pouze hodnota 10 cm. Pokud je hodnota senzoru od objektu menší než 10 cm, je robot zastaven, a tedy není možný pohybu vpřed. Z této situace, se může dostat pouze vycouváním. Náraz při couvání však možný je, jelikož se ochranný senzor vyskytuje pouze v čele robota.

9.3 Výstražné diody

Výstražné diody značí směr jízdy. Zda robot zatáčí vpravo nebo vlevo. Jsou použity dvě RGB diody, které mají každou barvu diody zvlášť. Připojeny jsou však pouze k červené barvě. V každé otočce smyčky mikropočítače probíhá zapnutí a vypnutí diody, což způsobuje blikání červené barvy.

9.4 Výstražný bzučák

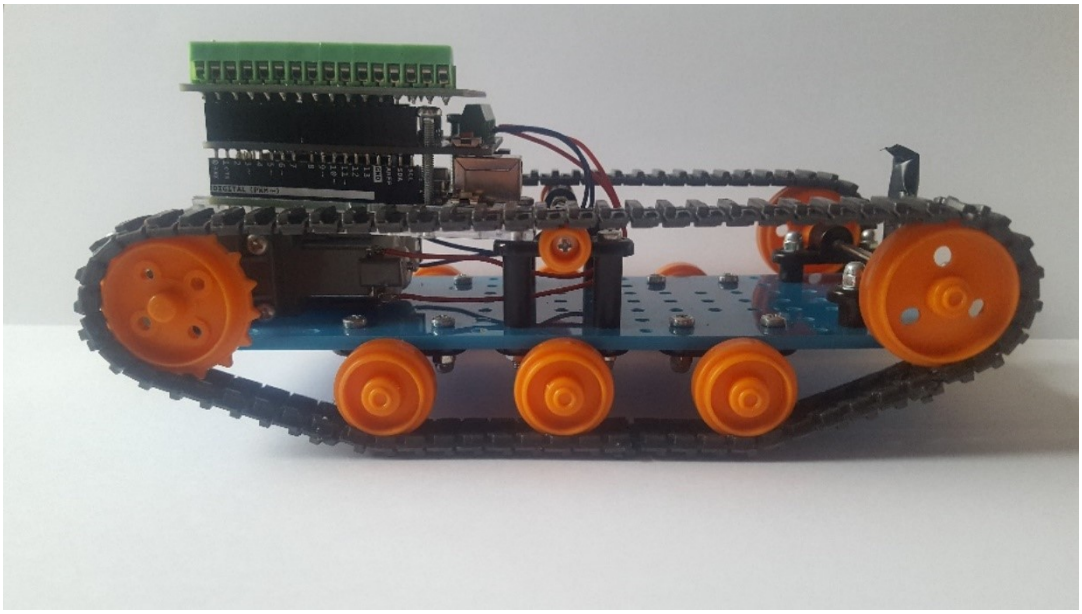
Mezi použité výstražné prvky byl použit bzučák, tedy bzučák. Který upozorní na to, že auto couvá. Jedno pípnutí probíhá opět v každé otočce smyčky mikropočítače. V případě, že bzučák vydává jen cvakavé zvuky, znamená to, že má nedostatek energie na vydání písklavého zvukového signálu.

9.5 Testování

Výsledný mobilní robot se chová tak, že poslouchá příkazy přicházející ze softwarového joysticku a dodržuje jejich pohyb. Mezi tím v neustálém toku dat přichází informace o vzdálenosti z ultrazvukového senzoru. Při zatáčení na strany problikávají příslušné LED diody. Při zpětném chodu je naopak vybuzen bzučák. V případě přiblížení přední části robota k velké překážce, se robot zastaví a je možné pouze couvání.

Rychlost tanku krom samotného nastavení závisí i na intenzitě proudu. Čím je baterie slabší, tím klesá proud a robot jede pomaleji.

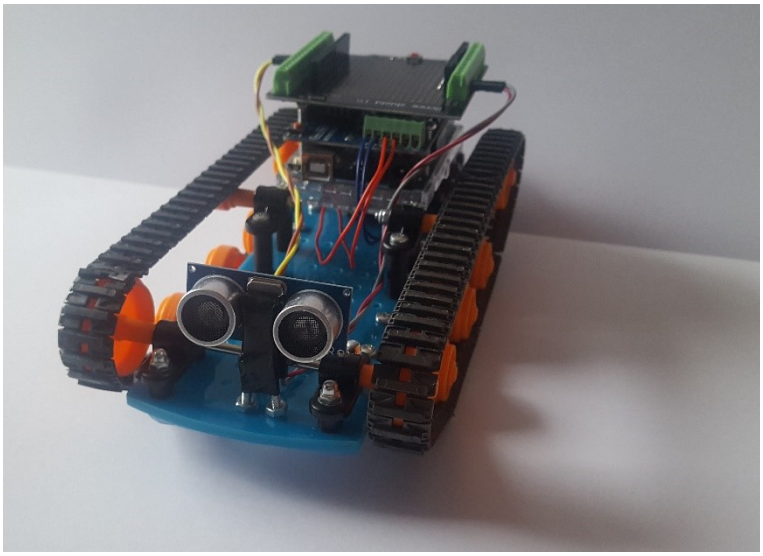
Díky tvaru podvozku a síle motorů je s robotem možné přejet i obtížnější překážky.



Obrázek 31 Mobilní robot – pohled z boku

Samotné shieldy jsou pro robota úsporným řešením z pohledu přehlednosti a místa. Na obrázcích můžeme vidět, jak vypadají shieldy úhledně.

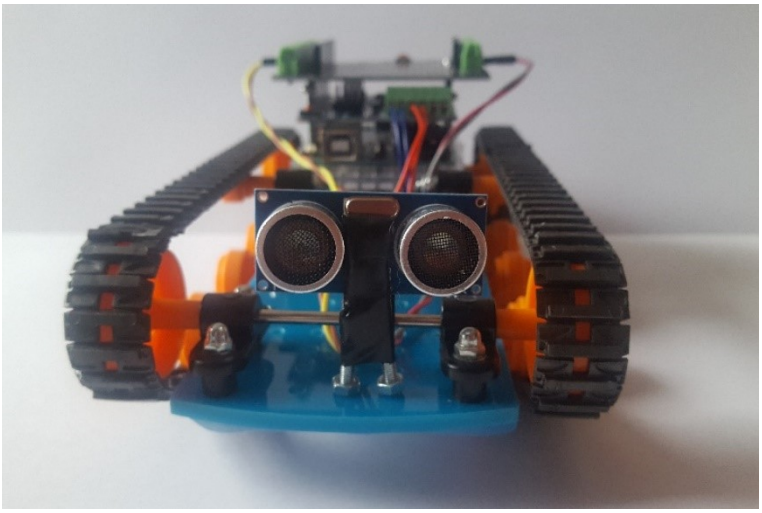
Pro větší bytelnost robota jsou shieldy k podvozku připevněny šrouby a piny jsou namísto dutinkové lišty připevněny svorkami. Jednotlivé drátky jsou u motoru připájeny (lze vidět na obrázku č. 32) a na druhém konci jsou připojeny svorkami u motor shieldu (lze vidět na obrázku č. 33). S pevným upevněním součástek je snazší pohyb bez omezení z důvodu vypojení.



Obrázek 32 Mobilní robot – pohled na celý model

Možným zdrojem problémů při provozu robota jsou napájecí baterie. Kdy při nízkém proudu není schopný uvést do pohybu motory. Nízký proud může způsobit i to, že není dostatečně vyuzen bzučák a tím místo vysokého tónu vzniká jen praskání.

Další nevýhodou je ultrazvukový senzor, při kterém mohou vznikat problémy v přehlédnutí objektu. Funkce ultrazvukového senzoru není dokonalá, odraz se nemusí vždy vrátit zpět do senzoru. Může docházet k tomu, že se ultrazvuková vlna odrazí o kulatý, či nepravidelný předmět špatným směrem, nebo že paprsek úplně zanikne.



Obrázek 33 Mobilní robot – pohled zepředu

ZÁVĚR

Cílem práce byl návrh a realizace modelu mobilního robota, který je ovládám pomocí mobilního zařízení jako je tablet či mobilní telefon. Bezdrátový obousměrný přenos byl uskutečněn za pomoci rozhraní Bluetooth. Celý model je směřován k tomu, aby byl zajímavý a jednalo se oživení výuky programování ve srovnání s běžnými programy pro osobní počítač s výstupem pouze na obrazovku. Také byla provedena rešerše již existujících komerčních programovatelných interaktivních stavebnic a porovnání jejich vlastností a cen.

Model robota byl řešen za pomoci Open-Source stavebnice Arduino. Robot se skládá z pásového podvozku poháněného motory skrze převodovku. Celý tento mechanizmus je pak řízen pomocí rozšiřující desky, tzv. motor shield. Na modelu je také použit ultrazvukový senzor vzdálenosti, který robotu umožňuje vyhýbat se překážkám. Dále jsou osazeny LED a bzučák pro indikaci směru pohybu robota.

Programové vybavení pro řídicí mikropočítač bylo napsáno v jazyce C/C++ ve vývojovém prostředí Arduino IDE.

Mobilní aplikace pro ovládání robota byla vytvořena v jazyce Java v prostředí Android Studio. Tato aplikace zajišťuje příjem a zobrazení dat z ultrazvukového senzoru a ovládání modelu pomocí joysticku na obrazovce.

Výsledkem práce je mobilní robot řízený vývojovou deskou s mikropočítačem Arduino, který je možno ovládat pomocí chytrého telefonu nebo tabletu s operačním systémem Android. Ovládání je realizováno prostřednictvím rozhraní Bluetooth. Robot je postaven na pásovém podvozku a je schopen vykonávat příkazy přicházející z mobilního zařízení a odesílat data z ultrazvukového senzoru do tohoto zařízení.

SEZNAM POUŽITÉ LITERATURY

[1] CATSOULIS, John. Designing embedded hardware. 2nd ed. Sebastopol, CA: O\symbol{39 Reilly, 2005, xvi, 377 p. ISBN 0596007558.

[2] GRIFFITHS, Dawn a David GRIFFITHS. Head first Android development. Sebastopol: O\symbol{39Reilly, \matsymb{lbrack2015\matsymb{rbrack. Head first series. ISBN 1449362184.

[3] MARGOLIS, Michael. Arduino cookbook. 2nd ed. Sebastopol, Calif.: O\symbol{39Reilly, 2012, xx, 699 p. ISBN 1449313876.

[4] NOVÁK, Petr. Mobilní roboty: pohony, senzory, řízení. Praha: BEN - technická literatura, 2005. ISBN 80-7300-141-1.

[5] PINKER, Jiří. Mikroprocesory a mikropočítače. 1. vyd. Praha: BEN - technická literatura, 2004, 159 s. ISBN 80-7300-110-1. \end{arab

[6] Elektronika [online]. Praha: Univerzita Karlova, 1998, 2018-10-03 [cit. 2019-04-27]. Dostupné z: https://physics.mff.cuni.cz/kfpp/skripta/elektronika/kap9/jednocpoc.html

[7] JANEČEK, Vladislav. Mikropočítač Raspberry Pi. Živě [online]. 12-09-2012 [cit. 2019-04-27]. Dostupné z: https://www.zive.cz/clanky/vyzkouseli-jsme-mikropocitac-raspberry-pi/sc-3-a-165391/default.aspx

[8] Tamiya 70168 Double Gearbox Kit: Pololu [online]. [cit. 2019-04-27]. Dostupné z: https://www.pololu.com/product/114

[9] VODA, Zbyšek. PROGRAMUJEME ARDUINO. Arduino.cz [online]. 14-10-2014 [cit. 2019-04-27]. Dostupné z: https://arduino.cz/programujeme-arduino/

[10] ARDUINO UNO REV3 [online]. 2018 [cit. 2019-04-27]. Dostupné z: https://store.arduino.cc/arduino-uno-rev3

[11] KENDALL, Brad. Getting started with Arduino a beginner's guide [online]. USA: MakeUseOf, 2013 [cit. 2019-04-27]. Dostupné z: www.bradkendall.ca

[12] VODA, Zbyšek. Průvodce světem Arduina [online]. Praha. Arduino Open-Source Comunity, 2013 [cit. 2019-04-27]. Dostupné zhttps://www.robotikabrno.cz/docs/arduino/Pr%C5%AFvodce-sv%C4%9Btem-Arduina-CZ.pdf

[13] ADA, Lady. Arduino Tips, Tricks, and Techniques [online]. 2019 [cit. 2019-04-27]. Dostupné z: https://cdn-learn.adafruit.com/downloads/pdf/arduino-tips-tricks-and-techniques.pdf

[14] ADA, Lady. Rydepier Blog Spot: List of Keyes Boards/Sensors for Arduino [online]. 2015 [cit. 2019-04-27]. Dostupné z: https://rydepier.wordpress.com/2015/06/13/list-of-keyes-boardssensors-for-ardui-no/?fbclid=IwAR1DsmYsQUrqV61kfVbndjz5KF7OHX32wxaKoEkevGhi8btYCAYzH22PC4s

[15] STUDT, Jim a Derek YERGER. Electronic Projects: Components Available Worldwide [online]. PJRC, 2007 [cit. 2019-04-27]. Dostupné z: https://www.pjrc.com/teensy/td_libs_OneWire.html?fbclid=IwAR2Dc_hIP6QvnWNqlCqgYQuYftgU6e_aKU-cHd_mst5cbU0-NotpBxhj5NE

[16] Nejlepší programovací jazyk pro robotiku [online]. Brno: Elektro průmysl, 2017 [cit. 2019-04-27]. Dostupné z: https://www.elektroprumysl.cz/software/jaky-je-nejlepsi-programovaci-jazyk-pro-robotiku

[17] TROJOVSKÝ, Pavel. Výukové aplikace pro mobilní zařízení [online]. Univerzita Hradec Králové katedra Kybernetiky, 2017 [cit. 2019-04-27]. Dostupné z: https://theses.cz/id/t9kowj/STAG87794.pdf. Diplomová práce. Univerzita Hradec Králové. Vedoucí práce Doc. RNDr. Štěpán Hubálovský, Ph.D.

[18] Merkur Robot: Arduino – jak ovládat „napětí“ [online]. Praha: Ladislav Vohralík, 2016 [cit. 2019-04-27]. Dostupné z: http://merkurrobot.cz/?p=1402

[19] Microchip: ATmega328P [online]. Chandler, Arizona: Microchip Technology, 2019 [cit. 2019-04-27]. Dostupné z: https://www.microchip.com/wwwproducts/en/ATmega328P

[20] NGUYEN, Nam. Essential Cyber Security Handbook [online]. Google Commerce, 2018 [cit. 2019-04-27]. Dostupné z: https://play.google.com/books/reader?id=Pr9TDwAAQBAJ&hl=cs&pg=GBS.PP1

[21] LOVELESS, Mark. DECIPHER: UNDERSTANDING BLUETOOTH SECURITY [online]. DUO, 2018 [cit. 2019-04-27]. Dostupné z: https://duo.com/decipher/understanding-bluetooth-security

[22] VOJÁČEK, Antonín. Magnetické senzory přiblížení - 2. díl: Základní teorie Hallova jevu [online]. 8. Leden 2018 [cit. 2019-04-28]. Dostupné z: https://automatizace.hw.cz/magneticke-senzory-priblizeni-2-dil.html

[23] PETERKA, Jiří. Bity za sekundu vs. baudy [online] Computerworld. 1992, čís. 44. Dostupné z: http://www.earchiv.cz//a92/a244c120.php3 [cit. 2012-02-20].

[24] VOJÁČEK, Antonín. Magnetické senzory s Hallovým efektem - 1. díl: Základní teorie Hallova jevu [online]. 7. Leden 2007 [cit. 2019-04-28]. Dostupné z: https://automatizace.hw.cz//magneticke-senzory-s-hallovym-efektem-1-princip

[25] Technické prostředky informatiky a automatizace: (úvod, popis funkce, konstrukce a aplikace). Fakulta aplikované informatiky. Ve Zlíně: Univerzita Tomáše Bati, 2007. ISBN 978-80-7318-535-0.

[26] HRUŠKA, František. Operační zesilovače v automatizační technice: (úvod, popis funkce, konstrukce a aplikace). Ve Zlíně: Univerzita Tomáše Bati, 2006. ISBN 80-731-8473-7.

[27] HRUŠKA, František. Technické prostředky automatizace II: (úvod, popis funkce, konstrukce a aplikací). Vyd. 2. Ve Zlíně: Univerzita Tomáše Bati, Fakulta aplikované informatiky, 2006. ISBN 80-731-8397-8.

[28] HRUŠKA, František. Senzory v systémech informatiky a automatizace. Zlín: Univerzita Tomáše Bati ve Zlíně, 2007. ISBN 978-80-7318-630-2.

[29] SELECKÝ, Matúš. Arduino: uživatelská příručka. Přeložil Martin HERODEK. Brno: Computer Press, 2016. ISBN 978-80-251-4840-2.

[30] ALLEN, Grant. Android 4: průvodce programováním mobilních aplikací. Brno: Computer Press, 2013. ISBN 978-80-2513-782-6.

[31] SCHILDT, Herbert. Mistrovství - Java. Brno: Computer Press, 2014. Mistrovství. ISBN 978-80-251-4145-8.

[32] KREIDL, Marcel. Měření teploty: senzory a měřící obvody. Praha: BEN - technická literatura, 2005. Senzory neelektrických veličin. ISBN 80-7300-145-4.

[33] BOLGER, Fergus. DPS: Používání C++ v embedded systémech [online]. 3/2013 [cit. 2019-05-08]. Dostupné z: https://www.dps-az.cz/vyvoj/id:4479/pouzivani-c-v-embedded-systemech

[34] WINTER, Jan. Bezpečné propojení počítačů [online]. VUT Fakulta informačních technologií, 2017 [cit. 2019-05-08]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=115892. Diplomová práce. Vysoké učení technické. Vedoucí práce Ing. Ján Kubek.

[35] VODA, Zbyšek. ARDUINO POD POKLIČKOU: JAK FUNGUJE MIKROKONTROLÉR. : ATMEGA328, AVR, MIKROKONTROLÉR [online]. 2019. [cit. 2019-05-08]. Dostupné z: https://arduino.cz/arduino-pod-poklickou-jak-funguje-mikrokontroler/

[36] HORÁČEK, OLDŘICH. ARDUBLOCK – PROGRAMUJTE ARDUINO GRAFICKY: ARDUBLOCK, ARDUINO, ARDUINO PROGRAMOVÁNÍ [online]. 15.8.2014 [cit. 2019-05-08]. Dostupné z: https://arduino.cz/ardublock-graficke-programovani-v-arduino/

[37] HORÁČEK, OLDŘICH. TOP 10 FREE NÁSTROJŮ, ABY SE NAŠE DĚTI NAUČILY PROGRAMOVAT: GRAFICKÉ PROGRAMOVÁNÍ, PRO DĚTI [online]. 8.11.2018 [cit. 2019-05-08]. Dostupné z: https://arduino.cz/top-10-free-nastroju-aby-se-nase-deti-naucily-programovat/

[38] ČÍŽEK, Jakub. Vybrali jsme 16 programovatelných hraček a stavebnic pro děti. A vlastně i pro vás [online]. 7. prosince 2018, , 16 [cit. 2019-05-08]. Dostupné z: https://www.zive.cz/clanky/nejlepsi-pocitacove-stavebnice/sc-3-a-195794/default.aspx#part=1

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

LED	Light-Emitting Diode
RISC	Reduced Instruction Set Computer
CISC	Complex Instruction Set Computer
OS	Operating System
USB	Universal Serial Bus
RX	Receiver
TX	Trasmitter
A/D	Analog/Digital
PWM	Pulse width modulation
LCD	Liquid Crystal Display
GSM	Global System for Mobile communications
VDC	Vehicle Dynamics Control
DHT	Dihydrotestosterone
IrDA	Infrared Data Association
L2CAP	List of Bluetooth protocols
SDP	service data point
MAC	Media Access Control
MITM	Man in the middle
NFC	Near-field communication
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
WWW	World Wide Web
PHP	Hypertext Preprocessor
SQL	Structured Query Language
VHDL	VHSIC Hardware Description Language

SMS	Short Message Service
RFCOMM	List of Bluetooth protocols

SEZNAM OBRÁZKŮ

SEZNAM SCHÉMAT

SEZNAM UKÁZEK KÓDU

SEZNAM TABULEK