Středoškolská technika 2014

Setkání a prezentace prací středoškolských studentů na ČVUT

Chronos – robotická platforma

Matouš Hýbl

Klvaňovo gymnázium a Střední odborná škola zdravotnická a sociální
Komenského 549, Kyjov

Prohlášení

Prohlašuji, že jsem svou práci vypracoval samostatně, použil jsem pouze podklady (literaturu, SW atd.) uvedené v přiloženém seznamu a postup při zpracování a dalším nakládání s prací je v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V Kyjově dne ..... podpis: .....

Poděkování

Děkuji především mé rodině za podporu a pomoc při řešení různých problémů. Dále děkuji Mikuláši Michalovi za poskytnutí ultrazvukových senzorů. V neposlední řadě bych chtěl poděkovat Klvaňovu gymnáziu za poskytnutí 3D tiskárny, bez které by nebylo možné tuto práci dokončit.

Anotace

Tato práce pojednává o konstrukci a programování diferenciální robotické platformy s názvem Chronos. Platforma slouží k experimentům s umělou inteligencí, je založena na řídicích jednotkách Arduino© a RaspberryPi© a byla navržena s důrazem na jednoduchost a také tak, aby byla lehce sestavitelná.

Klíčová slova: UMĚLÁ INTELIGENCE, ARDUINO, RASPBERRYPI, ROBOTICKÁ PLATFORMA, KONSTRUKCE, PROGRAMOVÁNÍ

Annotation

This thesis is about construction and programming of a simple differential-driven robotic platform named Chronos. Platform is used for experiments with artificial intelligence, it is based on Arduino© and RaspberryPi©. It was designed to be simple and easy to assembly.

Keywords: ARTIFICIAL INTELLIGENCE, ARDUINO, RASPBERRYPI, ROBOTIC PLATFORM, CONSTRUCTION, PROGRAMMING

Obsah

1 Úvod

Robotika se stává stále populárnějším vědním oborem, který v sobě kombinuje poznatky z matematiky, fyziky, elektrotechniky, informačních technologií a psychologie. Mobilní roboty budou v budoucnosti sloužit lidstvu jako pomocníci ve všech odvětvích lidské činnosti, zejména se uplatní v průmyslové automatizaci, jako průzkumníci vesmíru a budou provádět většinu nebezpečných operací, např.: manipulaci s jaderným palivem nebo průzkum zamořených oblastí. V budoucnosti se také objeví první androidi - roboti, kteří budou vypadat úplně jako lidé a ti budou schopni nahradit člověka ve všech jeho činnostech.

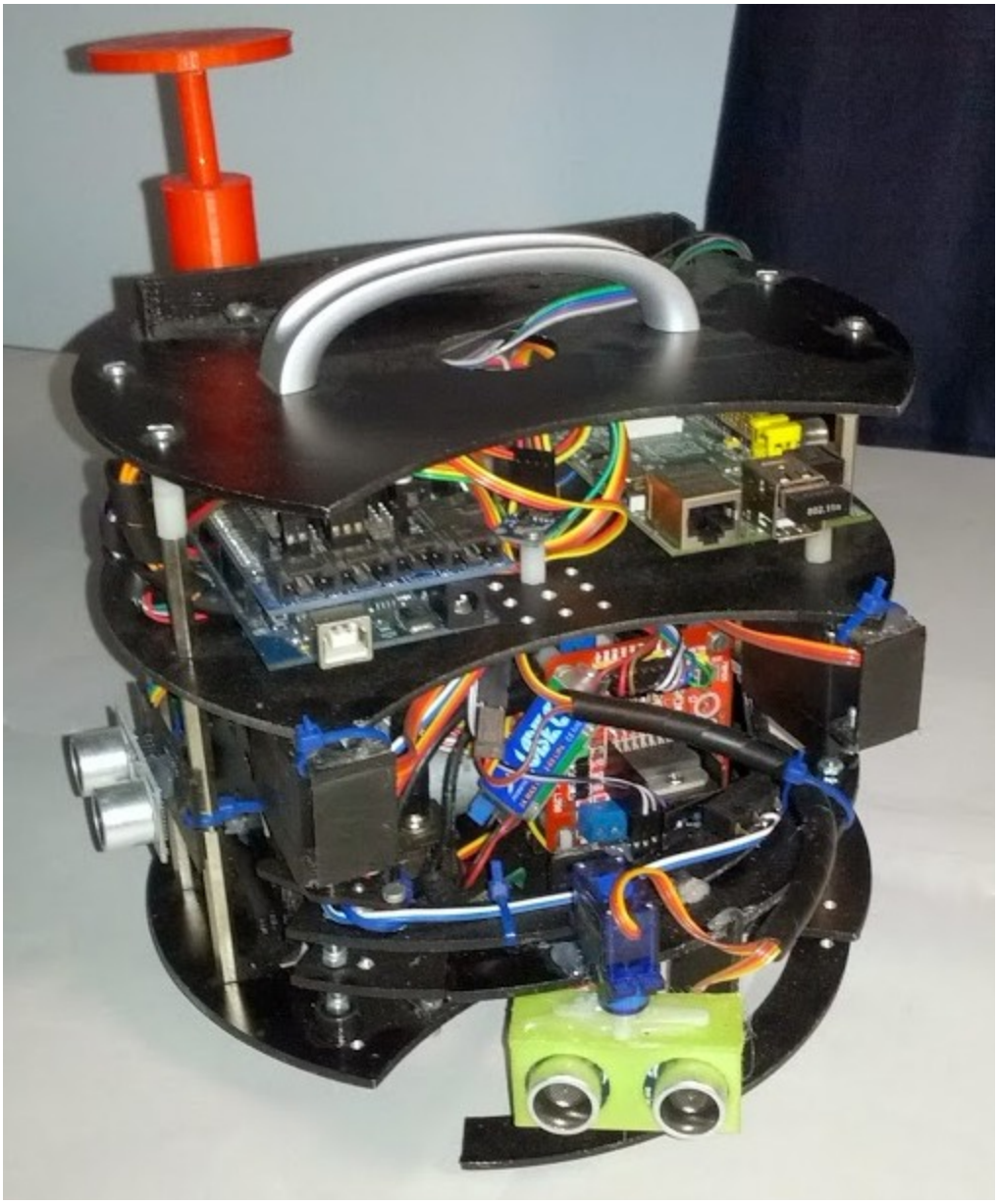
V České republice se každoročně koná několik robotických soutěží. Jednou z nejvýznamnějších je Robotický den v Praze pořádaný na přelomu dubna a května Matematicko-fyzikální fakultou Univerzity Karlovy. Tato soutěž má několik kategorií – sledování čáry, mini sumo nebo Bear Rescue. V kategorii Bear Rescue má robot za úkol najít v bludišti ztraceného plyšového medvěda a dovézt jej zpět na start. Tato kategorie je rozdělena do dvou podkategorií – dálkově ovládaný robot a zcela autonomní robot.

Tato práce pojednává o konstrukci složitějšího robota, založeného na platformách Arduino© a RasberryPi©. Robot by měl být schopen účasti v obou podkategoriích kategorie Bear Rescue. Robot byl navržen tak, aby byl snadno rozšířitelný o další senzory i výstupní zařízení.

2 Hardware

Robot Chronos je robot s kruhovým půdorysem založený na diferenciálním řízení, což znamená, že má dvě řízená kola a dvě stabilizační kuličky, díky čemuž se může na rozdíl od Ackermanova řízení, používanému například v autech, otáčet na místě. Pro dané úlohy robota by bylo možná vhodnější použít všesměrové řízení, ale došlo by tím ke zvýšení náročnosti řízení, zvýšení chybovosti a v neposlední řadě by také došlo, vzhledem k ceně všesměrových kol, ke zvýšení nákladů na stavbu robota.

Robot byl sestaven z hliníkových desek propojených distančními sloupky. Hliníkové desky jsou doplněny 3D tištěnými díly, které souží k uchycení komponent – například senzorů.



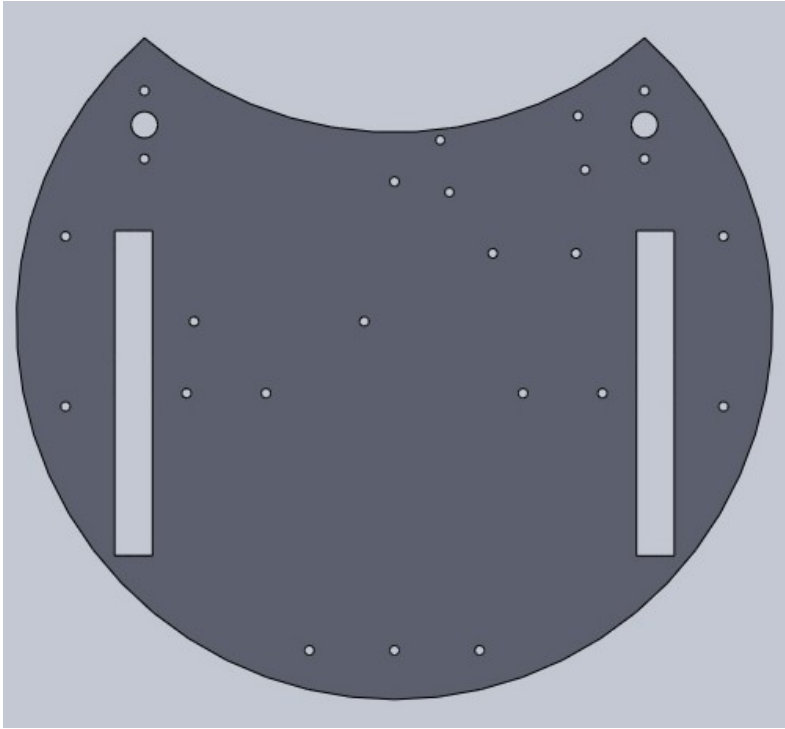
2.1 Hliníkové desky

Základní konstrukce robota je tvořená kruhovými hliníkovými deskami o průměru 200 mm propojenými distančními sloupky. Desky byly vyřezány z desek slitiny hliníku $AlMg_3$ o tloušťce 2,5 mm pomocí vodního paprsku.

Ilustrace 1: Surová hliníková deska

2.1.1 Spodní deska

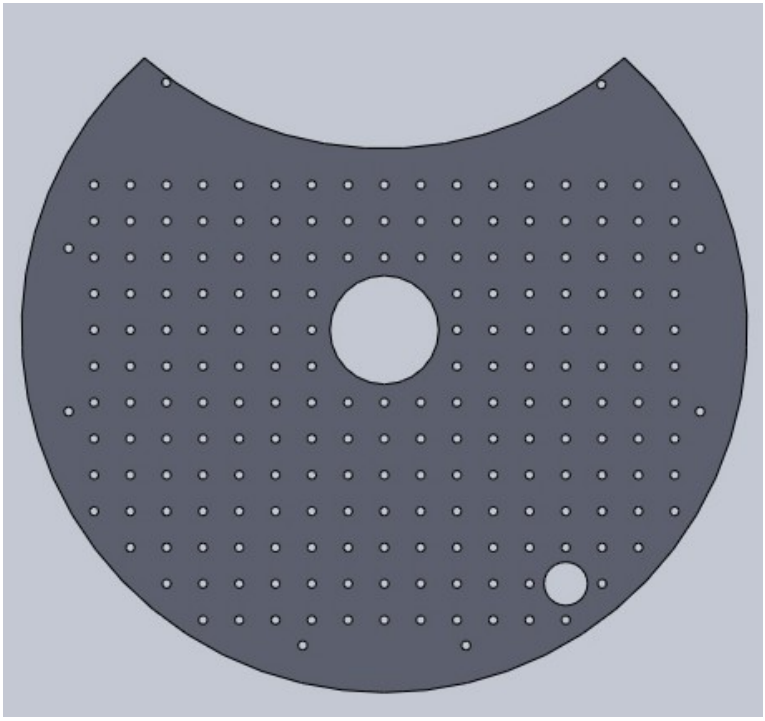
Spodní deska robota tvoří základ robota, jsou na ní umístěné motory, ovládací elektronika motorů, některé ze senzorů, akumulátory, držák napájecích konektorů a držák nabíjecích konektorů. Také se na ní nachází domeček ložiska 625, který slouží k uchycení uchopovacího mechanismu ke spodní desce. Veškeré přídavné díly jsou na desce uchyceny pomocí šroubů a samojistných matic rozměru M3. Deska má kruhový tvar s kruhovým výřezem v přední části, do desky byly také vyřezány otvory pro kola a deska byla navrtána dírami o průměru 2,4 mm, do kterých byl vyřezán M3 závit, dále byly do desky navrtány dvě díry o průměru 7 mm, kterými byl prostrčen šroub sloužící jako osa uchopovacího mechanismu. Ke spodní desce je pomocí devadesáti milimetrového kovového distančního sloupku přichycena první horní deska.



Ilustrace 2: Render spodní hliníkové desky

2.1.2 Horní desky

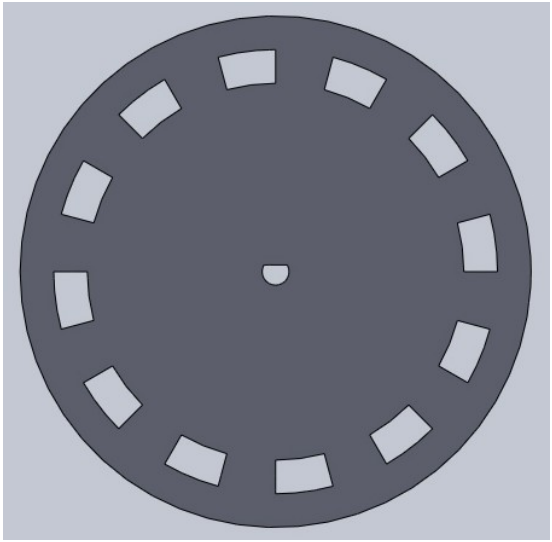
Horní desky jsou téměř stejné jako spodní deska, došlo u nich pouze k vypuštění otvorů pro kola a os uchopovacího mechanismu. Do první horní desky byly navrtány díry pro distanční sloupky sloužící k uchycení elektroniky a servomechanismů uchopovacího mechanismu. Do druhé vrchní desky byly navrtány díry pro uchycení madla určeného k snazšímu přenášení robota a díry, kterými je k desce přichycen držák mobilního telefonu. Horní desky jsou k sobě připevněny pomocí distančních sloupků o délce 50 mm.



Ilustrace 3: Render horní hliníkové desky

2.2 Kola

Kola robota mají průměr 76 mm a byla vyřezána pomocí vodního paprsku z hliníkové desky o tloušťce 5 mm. Uprostřed kol je výřez tvaru písmene D o poloměru 2 mm sloužící k nasazení kol na motory. Ve vzdálenosti 5 mm od okraje kol jsou vyřezány otvory sloužící k přerušování paprsku enkodérů, těchto otvorů je po obvodu kola 12 a jsou rozmístěny v 15 stupňovém intervalu. Po obvodu kol byl nalepen linoleový pásek zajišťující dostatečně velký součinitel smykového tření.



Ilustrace 4: Render kola - bez pneumatiky

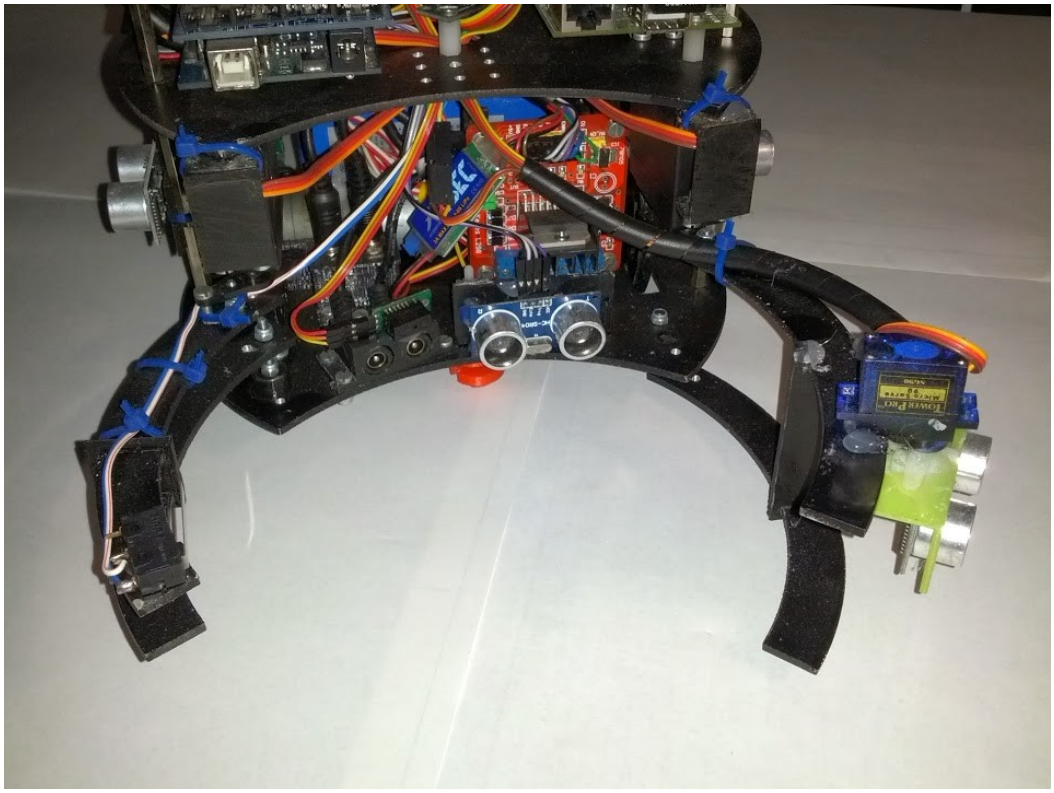
2.2.1 Enkodéry

Enkodéry robota jsou složeny z modulů infračervených závor a jejich držáků vytištěných na 3D tiskárně. Modul infračervené závory má označení HC-020K a byl zakoupen z Číny, jedná se o modul, který má vyjma samotné závory také analogový komparátor, díky kterému je výstupem senzoru binární informace. Vzhledem k různé hloubce senzoru a vzdálenosti mezi okrajem kola a otvoru pro enkodér v kole bylo nutné odstranit plastovou přepážku mezi infračervenou diodou a fototranzistorem citlivým na infračervené záření.

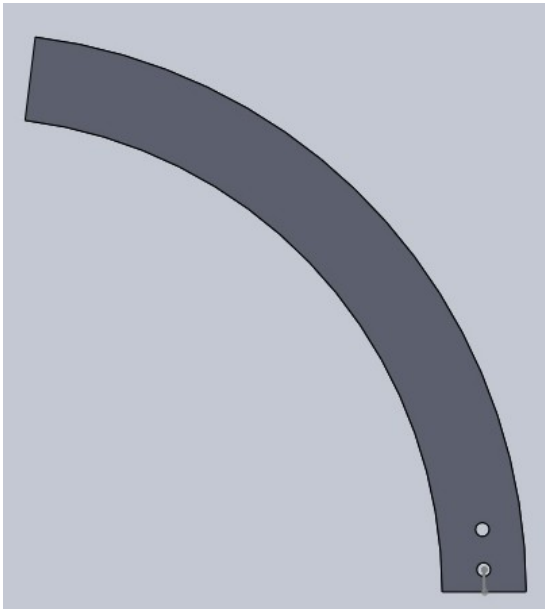
Ilustrace 5: Modul enkodéru, zdroj http://dx.com

2.3 Uchopovací mechanismus

Uchopovací mechanismus robota se skládá ze čtyř hliníkových desek a dvou servomechanismů. Hliníkových desek uchopovacího mechanismu jsou dva typy lišící se různými vnitřními a vnějšími poloměry podle toho na které straně jsou namontovány - na levé straně jsou desky s většími poloměry a také s větší vzdáleností mezi deskami, na pravé straně jsou desky s menšími poloměry a také s menší vzdáleností mezi deskami. Na levé straně uchopovacího mechanismu je umístěn radar a na pravé straně je umístěn koncový spínač, který přeruší pohyb mechanismu v případě, že byl sepnut.



Ilustrace 6: Uchopovací mechanismus na robotovi



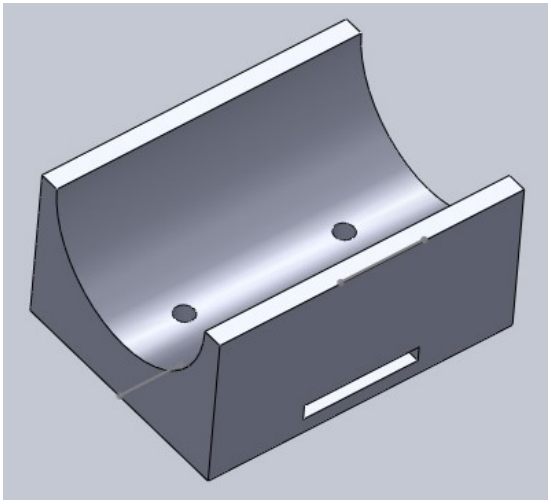
Ilustrace 7: Render hliníkové desky uchopovacího mechanismu

2.4 Tištěné díly

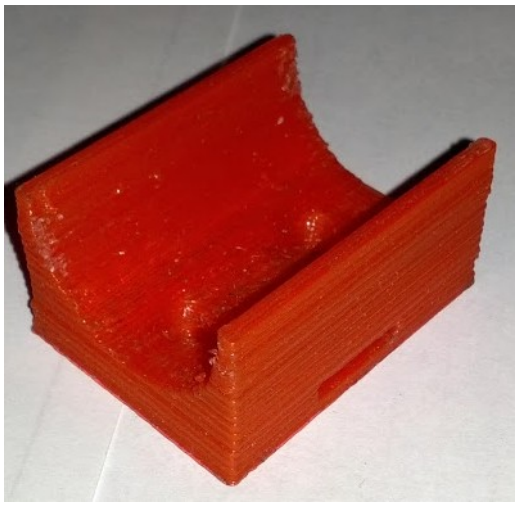
Ke konstrukci robota byly vyjma hliníkových desek použity také díly vytištěné pomocí 3D tiskárny. Jedná se o tiskárnu Prusa Mendel, což je 3D tiskárna vycházející z projektu RepRap vyvinutá Josefem Průšou. Díly byly vytištěny z černého nebo červeného PLA plastu.

2.4.1 Držáky motorů

Držáky motorů jsou pravděpodobně nejdůležitějšími tištěnými díly na robotovi, jedná se o kvádry, které mají z jedné strany vyřezaný půlkruhový otvor pro vložení motoru. Motor je v nich upevněn pomocí tavného lepidla a dvou stahovacích pásků, provlečených otvorem ve spodní části držáků, které zaručují téměř absolutní upevnění motorů v držácích. Ke spodní desce jsou držáky připevněny pomocí dvou šroubů M3x10, které jsou i se svými zahloubeními umístěny v nejnižším bodě půlkruhového zahloubení držáků.



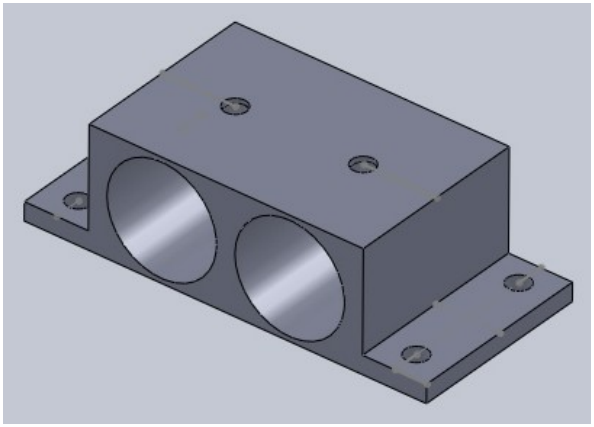
Ilustrace 8: Render držáku motoru



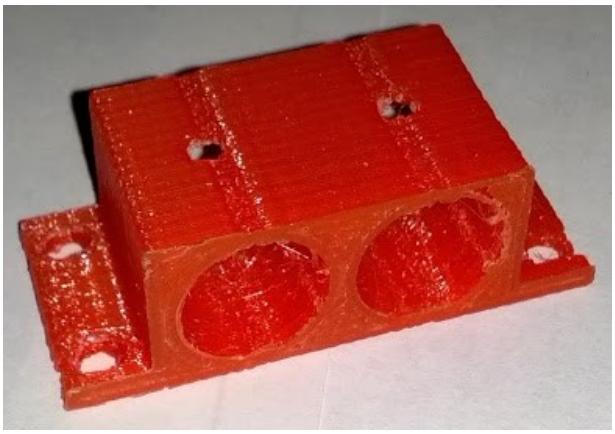
Ilustrace 9: Vytištěný držák motoru

2.4.2 Držáky konektorů akumulátorů

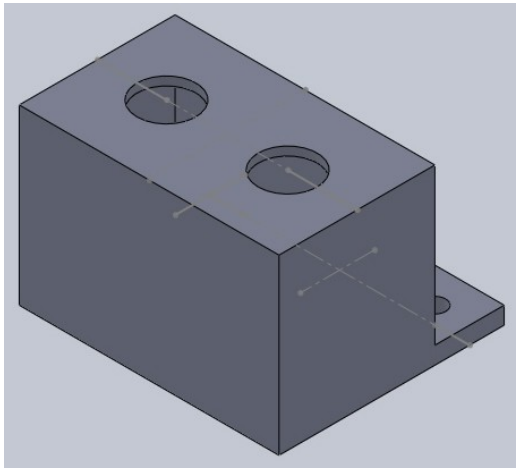
Akumulátory použité na robotovi mají dva konektory, jeden vstupní (nabíjecí) a jeden výstupní (napájecí), které byly pro snazší manipulaci namontovány do dvou držáků. Nabijecí držák je kostka obsahující 2 díry pro vložení nabíjecích konektorů, které jsou v držáku navíc upevněny pomocí přítlačných šroubů M3x6. Nabijecí držák je dále tvořen dvěma postranními výstupky, které obsahují díry pro namontování držáku ke spodní desce. Napájecí držák je tvořen pláštěm kvádru, ve kterém jsou umístěny dva otvory pro montáž dvou konektorů sloužících k připojení akumulátoru, otvor pro vedení kabelů od konektorů, na plášti je dále umístěn výstupek obsahující díry pro šrouby M3x10, které slouží k namontování držáku ke spodní desce.



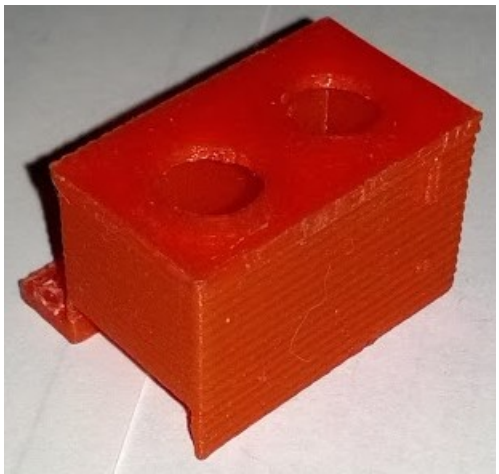
Ilustrace 10: Render držáku nabijecích kabelů



Ilustrace 11: Vytištěný držák nabijecích kabelů



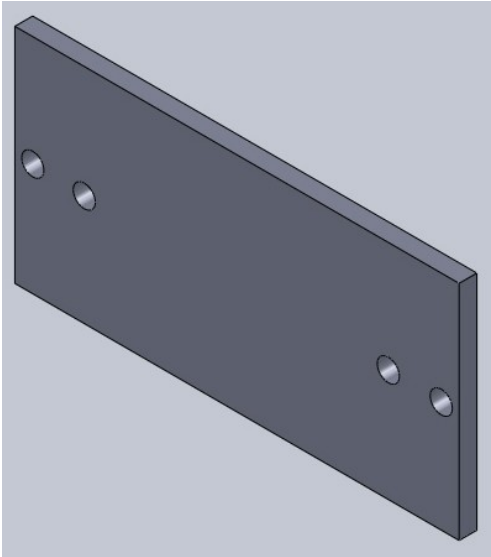
Ilustrace 13: Render držáku napájecích kabelů



Ilustrace 12: Vytištěný držák napájecích kabelů

2.4.3 Držáky ultrazvukových senzorů

K distančním sloupkům spojujícím spodní hliníkovou desku a první horní desku byly namontovány držáky ultrazvukových senzorů. Jedná se o obdélníkové destičky, které mají při obou kratších stranách dva otvory pro provlečení stahovacího pásku, pomocí kterého jsou k distančním sloupkům připevněny. Samotné ultrazvukové senzory jsou pak k držákům nalepeny pomocí tavného lepidla.



Ilustrace 14: Render držáku ultrazvukových senzorů

Ilustrace 15: Držák se senzorem namontovaný na robotovi

2.4.4 Stabilizátory

Za účelem maximální stability robota byly do přední a zadní části namontovány dva stabilizátory, jedná se o kruhové destičky s kopulí, do kterých byl přidán otvor pro uchycení šroubu M3x20, kterým se stabilizátory připojují ke spodní desce, kde jsou přišroubovány pomocí dvou samojistných matic, díky kterým lze regulovat výšku stabilizátoru.

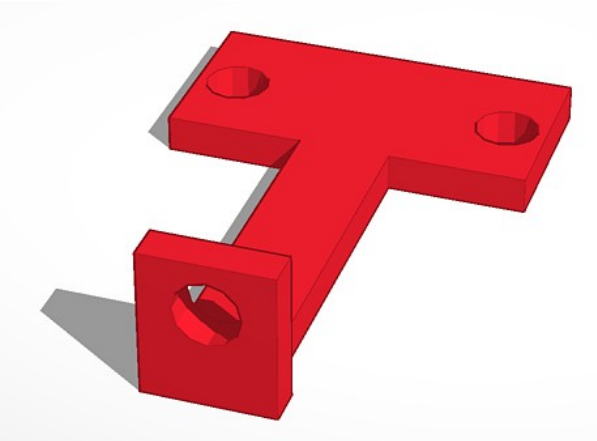
2.4.5 Nouzový vypínač

Na robotických soutěžích větších robotů je povinný nouzový vypínač, který v případě nutnosti odpojí motory od napájení, tím zůstanou zachována data v robotově paměti a je podle nich možné analyzovat chybu, bohužel komerční nouzové vypínače jsou moc velké a drahé, proto byl na robota Chronos vytvořen zjednodušený nouzový vypínač, skládá se ze dvou dílů – válce, ve kterém je umístěn otvor pro vedení bodce s čepičkou, který se dotýká jednoduchého tlačítkového vypínače ON-OFF.

Ilustrace 16: Nouzový vypínač namontovaný na robotovi

2.4.6 Držáky enkodérů

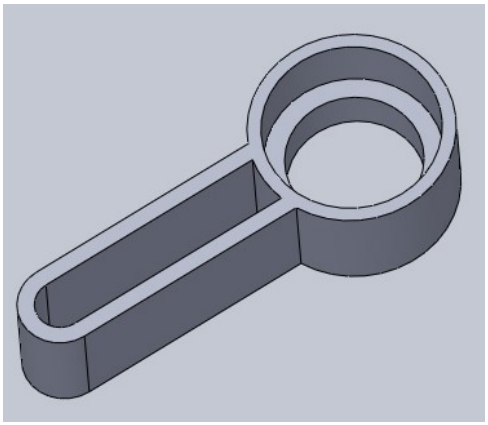
Ke správné a relativně přesné funkci enkodérů je nutné aby byly nasunuty na kola přibližně ve výšce osy kol, k tomu slouží držák enkodéru. Jedná se o destičku tvaru písmene T, která je dole zakončena destičkou kolmou k základní destičce a ve které je umístěna díra pro uchycení k základní desce. Enkodéry jsou k držáku přichyceny pomocí distančních sloupků našroubovaných do horní části držáku.



Ilustrace 17: Render držáku enkodéru

2.4.7 Držák ložiska 625

Součástí uchopovacího mechanismu je ložisko 625, které je ke spodní desce uchyceno pomocí jednoduchého držáku. Jedná se o kruhový prstenec, ke kterému je připojeno očko pomocí kterého je držák uchycen ke spodní desce. Ložisko bylo do držáku nalepeno pomocí vteřinového lepidla.



Ilustrace 18: Render držáku ložiska

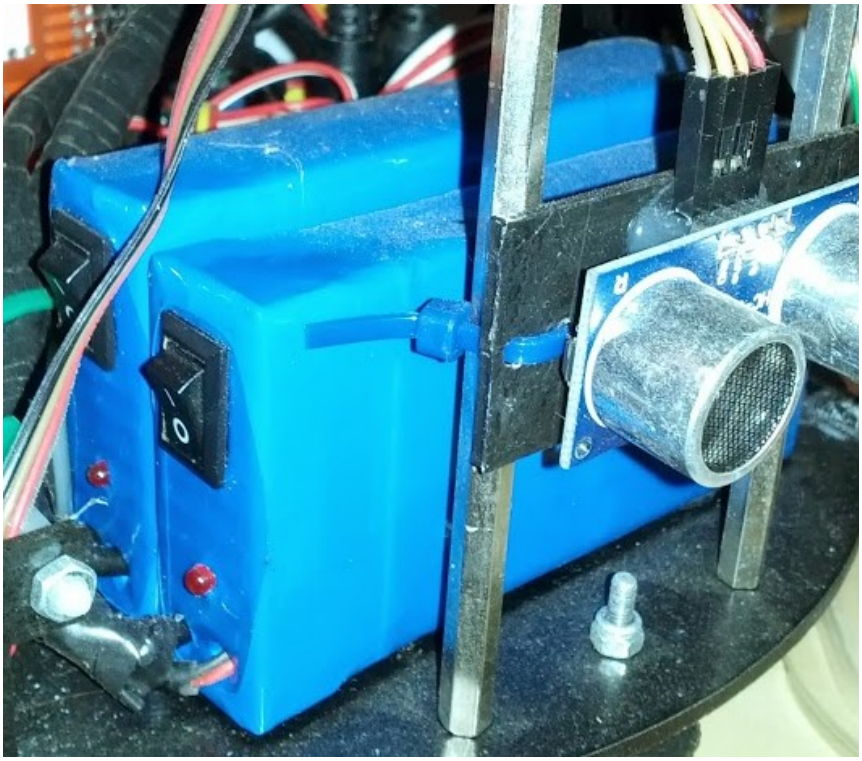
Ilustrace 19: vytištěný držák ložiska

2.5 Napájení

Na robotovi jsou použity dvě napěťové úrovně – 12 V úroveň pro motory a 5 V úroveň, kterými je napájena veškerá elektronika. Zdrojem elektrické energie jsou dva 12 V akumulátory, z nichž jeden byl použit na napájení motorů a druhý na napájení elektroniky, kde musel být požít stabilizátor ke stabilizaci napětí z 12 V na 5 V.

2.5.1 Akumulátory

Jako zdroj elektrické energie byly použity dva 12 V akumulátory o kapacitě 4800 mAh, tyto akumulátory se nabíjí pomocí externí nabíječky, která se připojí k akumulátoru pomocí jednoho z kabelů. Jedno nabíjení trvá přibližně 5 hodin. Na obou akumulátorech je integrován vypínač, kterým se připojí napětí k výstupu a tím se celý robot zapne.



Ilustrace 20: Dva akumulátory namontované na robotovi

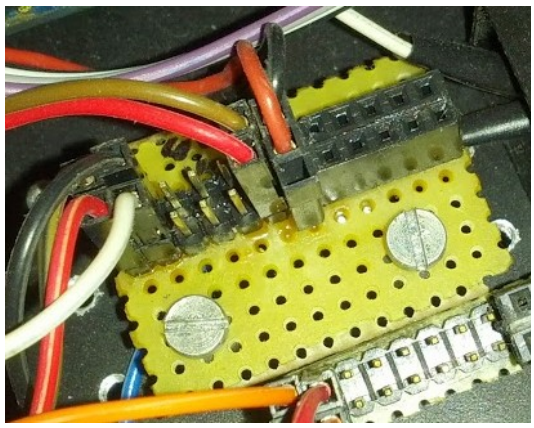
2.5.2 Stabilizátor napětí

Vzhledem k rozdílnému napětí akumulátoru a napětí potřebnému pro chod řídicích jednotek a servomechanismů bylo nutné napětí akumulátoru snížit pomocí stabilizátorů. Jako stabilizátory byly použity UBEC stabilizátory firmy HobbyWing, které jsou schopné dodávat proud až 5 A.

Ilustrace 21: Stabilizátor napětí, zdroj http://dx.com

2.5.3 Modul pro rozvod napětí elektroniky

Vzhledem k faktu, že ke stabilizátoru napětí lze připojit jen jedno zařízení bylo nutné vytvořit modul, díky kterému bude možno napájet více zařízení. Jedná se o dvouřadý konektor připájený na univerzální plošný spoj, díky kterému je možné napájet více zařízení najednou.



Ilustrace 22: Modul pro rozvod napětí elektroniky

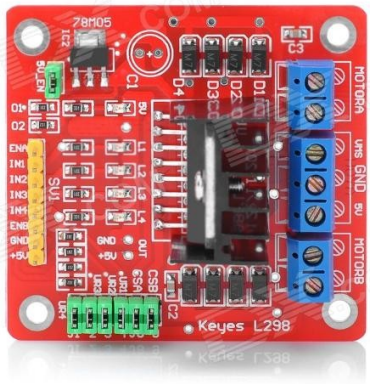
2.6 Motory

Ke svému pohybu používá robot dva stejnosměrné elektromotory pracující na napětí 12V. Motory mají rychlost 60 otáček za minutu a tah 4 kg/cm při napětí 12 V.

Ilustrace 23: Motor použitý na robotovi, zdroj http://dx.com

2.6.1 Řízení motorů

Motory jsou řízeny pomocí modulu dvojitého H-můstku založeném na integrovaném obvodu L298. Tento modul je schopen obousměrně řídit dva motory až do napětí 46 V a maximálního proudu 2 A na motor. Každý motor je ovládán pomocí tří pinů – dva z nich určují směr otáčení a třetí určuje rychlost motoru (jedná se o pin ENABLE, na který je přiveden PWM výstup řídicí jednotky).



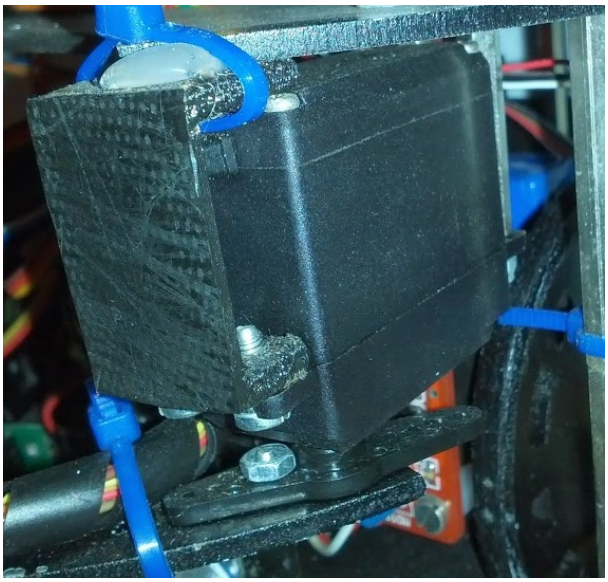
Ilustrace 24: H-můstek k řízení motorů, zdroj http://dx.com

2.7 Servomechanismy

Servomechanismus je zařízení, které umožňuje otočení výstupní osy o přesný úhel. Na robotovi Chronos byly použity dva typy servomechanismů – jeden typ pro pohyb uchopovacího mechanismu a druhý typ pro ovládání radaru.

2.7.1 Servomechanismy uchopovacího mechanismu

Pro pohyb ramen uchopovacího mechanismu byly použity dva servomechanismy typu SG5010 vyráběné firmou TowerPro. Tyto servomechanismy mají rychlost 60°/0,2 s a jejich tah je 5,5 kg/cm. Na osy servomechanismů byly nasazeny kotouče, do kterých byly navrtány díry pro připevnění desek uchopovacího mechanismu. Vzhledem k vysokému odběru elektrického proudu bylo nutné pro servomechanismy uchopovacího mechanismu přidat samostatný stabilizátor napětí a připojit jej k akumulátoru, který napájí motory.



Ilustrace 25: Servomechanismus uchopovacího mechanismu

2.7.2 Servomechanismus radaru

Pro pohyb radaru byl použit menší a méně výkonný servomechanismus SG90 rovněž vyráběný firmou TowerPro, jeho specifikace jsou – 60°/0,12 s a tah 2 kg/cm.

2.8 Senzory

Pro interakci robota s okolím byl robot vybaven různými senzory, například ultrazvukovými dálkoměry, infračerveným detektorem překážky a kompasem.

2.8.1 Ultrazvukové dálkoměry

Ultrazvukové dálkoměry slouží k měření vzdálenosti mezi robotem a překážkou, na robotovi Chronos jsou umístěny po jednom na bocích a vzadu a vepředu jsou dva přičemž jeden z nich slouží jako radar. Jedná se o moduly HC-SR04, které jsou schopné detekovat překážku vzdálenou až 4 metry.

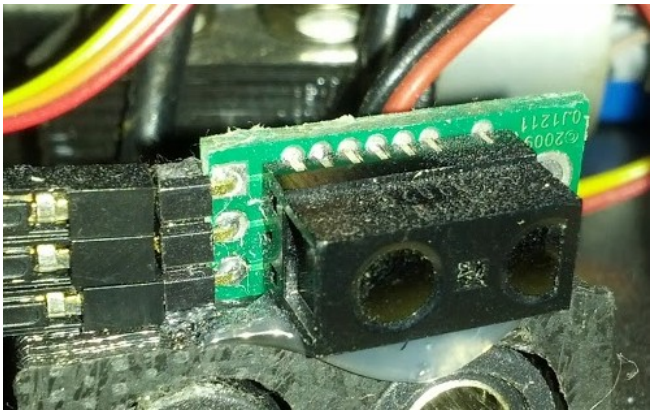
Ilustrace 26: Ultrazvukový senzor umístěný v zadní části robota

2.8.1.1 Radar

Radar je zvláštní druh ultrazvukového dálkoměru, a to hlavně proto, že je umístěn na servomechanismu a díky tomu je možné proskenovat plochu před robotem až do úhlu 120°.

2.8.2 Infračervený detektor překážky

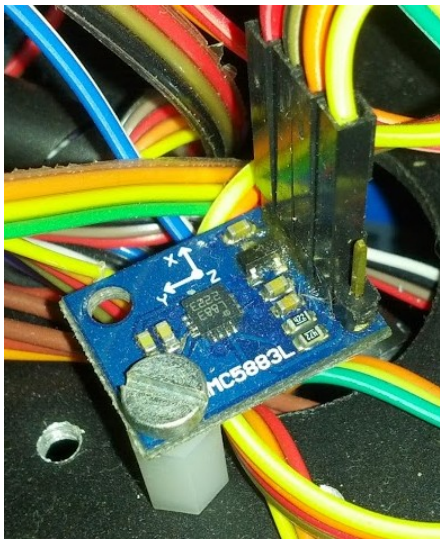
Infračervený detektor překážky byl na robota namontován do přední části, kde slouží jako senzor toho, že robot správně najel k nákladu a uchopovací mechanismus jej může uchopit. Jedná se o senzor GP2Y0D810 vyráběný firmou Sharp, doplněný o destičku plošných spojů Pololu 1133 s filtračními kondenzátory a konektorem pro připojení senzoru k řídicí jednotce.



Ilustrace 27: Infračervený detektor překážky

2.8.3 Kompas

K určení natočení robota vzhledem k magnetickému poli země a také k přesné rotaci byl na robota namontován modul kompasu HCM5883L. Jedná se o trojosý senzor magnetického pole, který s řídicí jednotkou komunikuje pomocí sběrnice I^2C.



Ilustrace 28: Modul kompasu

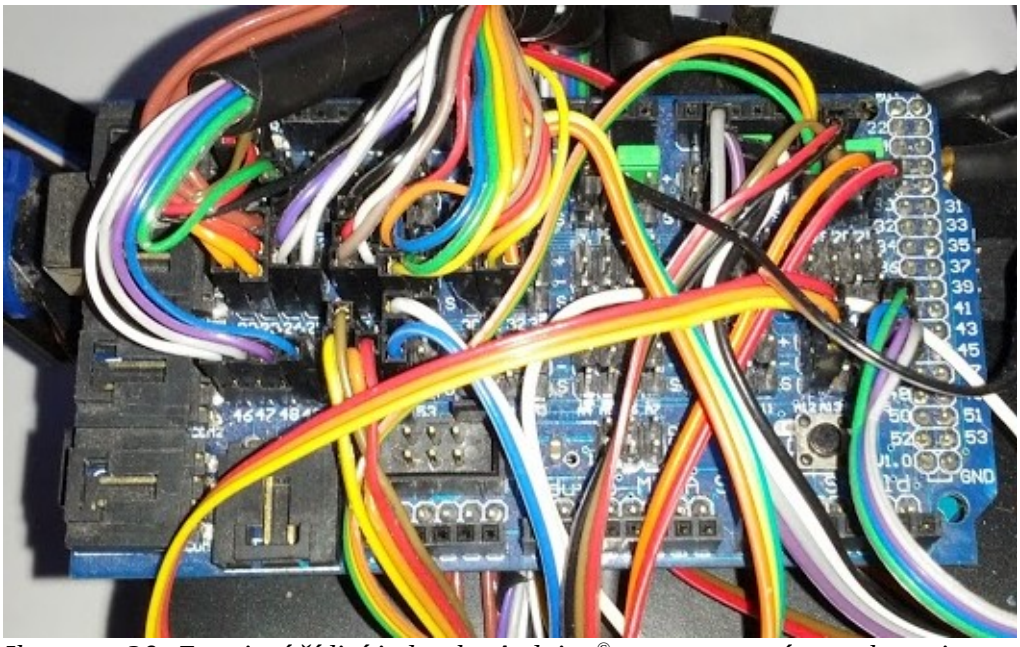
2.9 Řídicí jednotky

Na robotovi Chronos byly použity 2 typy řídicích jednotek – pro nízkoúrovňové operace 8bitové Arduino© Mega 2560 a 32bitové RaspberryPi© Model B pro výpočetně náročné operace, jako je například rozpoznávání obrazu.

2.9.1 Arduino© Mega 2560

Arduino© Mega 2560 je jedna z nejvybavenějších řídicích jednotek Arduino© založených na jádře AVR. Tato řídicí jednotka je založena na mikrokontroléru ATmega2560 vyráběném firmou Atmel, tento mikrokontrolér má k dispozici paměť FLASH o velikosti 256 kB, 8 kB paměti SRAM a 4 kB paměti EEPROM. Arduino© se programuje pomocí standardního USB kabelu A-B, na Arduinu© je k tomuto účelu naletován ještě mikrokontrolér Atmega16U2, který funguje jako CDC most. Arduino© je taktováno na 16 MHz a uživatel má k dispozici 70 I/O pinů, z nichž může být 16 použito jako analogové vstupy, 15 jako analogové výstupy (pomocí pulsně šířkové modulace), na zbylých pinech se nacházejí další periferie jako například USART, I^2C, SPI atd. Pro snazší připojení senzorů a dalších modulů byl na Arduino© nasazen tzv. shield (jedná se o rozšiřující desku plošných spojů, která přidává Arduinu© určitou funkcionalitu např. přístup k rozhraní Ethernet), který funguje jako redukce samic konektorů pinů na samce. Arduino© je pomocí USB kabelu připojeno také k RaspberryPi©.

Ilustrace 29: Řídicí jednotka Arduino© Mega 2560 se shieldem



Ilustrace 30: Zapojení řídicí jednotka Arduino© namontovaná na robotovi

2.9.2 RaspberryPi©

RaspberryPi© je malý počítač založený na SoC vyráběném firmou Broadcom, tento SoC je vybaven procesorem s jádrem ARM taktovaným na 700 MHz (volitelně lze přetaktovat až na 1 GHz), 512 MB operační paměti a grafickým procesorem VideoCoreIV. Z počítače je vyvedeno několik konektorů – konektor RJ45 pro připojení k Ethernetu, dva USB konektory, HDMI konektor, koaxiální konektor, 3,5 Jack pro připojení např. sluchátek, napájecí USB konektor a patice s GPIO piny sloužící k připojení různých periferií. Jako paměťové médium slouží RaspberryPi© SD karta, na které je uložen operační systém a veškerá uživatelská data. Na RaspberryPi© lze používat celou řadu operačních systémů, ale nejběžnější je některá z linuxových distribucí upravená pro běh na RaspberryPi© např. Raspbian založený na Debianu, vyjma linuxových distribucí lze použít také RISC OS, což je operační systém přímo vyvinutý pro procesory s redukovanou instrukční sadou a na RaspberryPi© tedy funguje nejrychleji. Díky použití linuxové distribuce Raspbian lze RaspberryPi© programovat téměř v libovolném programovacím jazyce.

Ilustrace 31: Řídicí jednotka RaspberryPI© namontovaná na robotovi

3 Software

Software robota byl naprogramován v několika programovacích jazycích v závislosti na typu zařízení – Arduino© bylo programováno v jazyce C++, RaspberryPi© bylo programováno v jazyce C a Java™ a mobilní telefon s operačním systémem Android© byl naprogramován v jazyce Java™ a XML.

3.1 Software pro Arduino©

Software pro Arduino© byl naprogramován v jazyce C++, naprogramováno bylo vyjma hlavního programu také několik knihoven, které značně zjednodušují ovládání robota. Arduino© bylo programováno v prostředí Eclipse© s použitím speciálního pluginu pro programování řídicích jednotek Arduino©.

3.1.1 Základní práce s Arduinem©

Arduino© má v základu předprogramovánu většinu nízkoúrovňových metod, které uživatel potřebuje, aby mohl programovat, bohužel je to částečně vykoupeno tím, že se některé parametry nedají upravovat – např. frekvence PWM je pevně stanovena na 500 Hz.

3.1.1.1 Základní struktura programu pro Arduino©

Programy pro Arduino© obsahují 2 základní metody – setup a loop. Setup je metoda, ve které je umístěn zdrojový kód pro inicializaci a základní nastavení periferií. Metoda loop obsahuje zdrojový kód hlavního programu, který je prováděn v nekonečné smyčce.

```
void setup(){
      // zdrojový kód prováděný pouze jednou, po zapnutí řídicí jednotky
}
void loop(){
      // zdrojový kód prováděný neustále v nekonečné smyčce
}
```

3.1.1.2 Přístup k I/O pinům

Pro přístup k pinům používá Arduino© tři základní metody, pokud se jedná o piny podporující PWM, nebo obsahující ADC přidává Arduino© ještě dvě další metody.

První metodou je pinMode, jedná se o metodu, která nastaví pin na vstup, výstup, nebo mu zapne interní PULL-UP rezistor.

```
pinMode(9, OUTPUT); // nastaví pin 9 jako výstup
pinMode(9, INPUT); // nastaví pin 9 jako vstup
pinMode(9, INPUT_PULLUP); // zapne na pinu 9 interní PULLUP rezistor
```

Druhou metodou je digitalWrite, která řídí výstupní tranzistor pinu a určuje logickou úroveň pinu – LOW/HIGH.

```
digitalWrite(9, HIGH); // nastaví pin 9 jako HIGH tzn. na pinu bude napětí 5V
digitalWrite(9, LOW); // nastaví pin 9 jako LOW tzn. na pinu bude napětí 0V
```

Třetí metodou je metoda digitalRead, která čte logickou hodnotu z určitého pinu, který je metodě předáván jako parametr, metoda vrací číslo 0, nebo 1 v závislosti na logické hodnotě pinu.

```
int result=digitalRead(9); // přečte logickou hodnotu pinu 9 a zapíše ji do
                           proměnné result
```

Další metodou je analogWrite, jedná se o metodu, která je schopná na vybraných pinech nastavit pomocí střídy signálu PWM nastavit určité napětí. Výstupní napětí lze vypočítat pomocí vzorce $V = \frac{param}{255} \times VIn$, kde param je hodnota předávaná funkci jako argument a VIn je napájecí napětí procesoru.

```
analogWrite(9, 255); // nastaví napětí pinu na 5V pomocí PWM
analogWrite(9, 127); // nastaví napětí pinu na 2,5V pomocí PWM
analogWrite(9, 0); // nastaví napětí pinu na 0V pomocí PWM
```

Poslední metodou je metoda analogRead, která souží ke čtení napětí na pinech s prefixem A. Tato metoda vrací 10 bitové číslo v závislosti na napětí připojeném k pinu. Napětí na pinu lze vypočítat pomocí vzorce $V = \frac{result}{1024} \times VIn$, kde result je hodnota navrácena metodou analogRead a VIn je napájecí napětí procesoru (za předpokladu, že se nevyužije pin AREF, díky kterému lze měřit jiný rozsah napětí se stejnou přesností).

```
int result = analogRead(A0); // změří napětí na pinu A0 a výsledek zapíše do
                                proměnné result
```

3.1.2 Obsluha základních knihoven

Arduino© má v základu naprogramováno několik knihoven, které značně zjednodušují práci s interními periferiemi procesoru např. pro přístup ke sběrnici I2C má Arduino© knihovnu Wire.

3.1.2.1 Obsluha USART pomocí knihovny Serial

Knihovna Serial je jedna ze základních knihoven Arduina©, jedná se o knihovnu schopnou číst a posílat data přes rozhraní USART. Na jednoduchých jednotkách Arduino©, jako je například Arduino© UNO, je předem definován jen jeden objekt této knihovny s názvem Serial, protože tato řídicí jednotka má pouze jedno rozhraní USART, ale Arduino© Mega 2560 použité na robotovi má díky tomu, že je vybaveno čtyřmi rozhraními UART, definovány celkem 4 objekty – Serial, Serial1, Serial2 a Serial3. Knihovna má 4 základní metody – begin, read, write a available.

Metoda begin slouží k inicializaci rozhraní USART, je jí předáván jeden parametr a to BaudRate, což je přenosová rychlost udávaná v bps.

```
Serial.begin(115200); // metoda inicializuje USART a nastaví mu přenosovou
                         rychlost 115200bps
```

Další metodou je metoda read, tato metoda vrací první nepřečtený byte ze vstupního bufferu, pokud nejsou dostupná žádná data vrací metoda -1. K přečtení určitého počtu bytu a zamezení zbytečného provádění metody read bývá tato metoda používána společně s metodou available, která vrací počet bytů, které se do vstupního bufferu načetly po posledním čtení.

```
if( Serial.available() > 2 ){ // pokud jsou ve vstupním bufferu více než dva
                                 byty přečti je
      int prvni_byte=Serial.read();
      int druhy_byte=Serial.read();
}
```

Poslední metodou je metoda write, která přes rozhraní USART pošle jeden byte, který je metodě předán jako parametr.

```
Serial.write(55); // odešle přes USART jeden byte s číslem 55
```

Na metodě write jsou pak v knihovnách Arduino© postaveny další metody, například metoda print, která přes rozhraní USART odešle řetězec znaků.

```
Serial.print(“ahoj!“); // pošle řetězec “ahoj!“ přes rozhraní USART
```

3.1.2.2 Obsluha knihovny Servo

Knihovna Servo je určena k ovládání servomechanismů. Knihovna je založena na přerušeních časovačů, pomocí kterých vytváří pulzy pro ovládání servomechanismů. Tato knihovna je schopna ovládat na Arduino© Mega 2560 až 48 servomechanismů avšak za cenu ztráty funkcionality PWM na některých pinech. Jednotlivé servomotory jsou pak deklarovány jako objekty třídy a jsou k nim přiřazeny piny pomocí metody attach. Servomechanismy jsou ovládány pomocí metody write, která na servo mechanismu nastaví požadovaný úhel, který je předáván metodě jako parametr. Knihovna Servo je do programu linkována pomocí příkazu #include <Servo.h>.

```
Servo servo; // vytvoří objekt servomechanismu
servo.attach(5); // nastaví pin servomechanismu na pin číslo 5
servo.write(90); // nastaví úhel servomechanismu na 90°
```

3.1.2.3 Obsluha knihovny Wire

Knihovna Wire slouží k přístupu ke sběrnici I²C. Ke sběrnici se přistupuje pomocí již předem vytvořeného objektu Wire, výjimkou je nejnovější Arduino© Due, kde lze přistupovat ke dvěma sběrnicím, tudíž lze vyjma objektu Wire použít i objekt Wire1. Inicializace sběrnice probíhá voláním metody begin, která má dvě přetížení – bez argumentu, toto přetížení nastaví řídicí jednotku jako Master, a s argumentem, který nastaví řídicí jednotce adresu, díky čemuž se pak řídicí jednotka chová jako Slave. Před použitím knihovny musí být knihovna nalinkována pomocí příkazu #include <Wire.h>.

3.1.2.3.1 Řídicí jednotka jako Slave

V případě, že je řídicí jednotka nastavena metodou begin jako Slave, používájí se k obsluze komunikace s Master řídicí jednotkou 2 metody onReceive a onRequest, které jako argument dostávají metodu bez návratového typu, která je volána vždy, když Master požaduje, nebo posílá data. Čtení a posílání dat přes I²C probíhá pomocí metod write a read. Rovněž v případě knihovny Wire je vhodné použít metodu read v kombinaci s metodou available, která zajistí, že ve vstupním bufferu bude požadovaný počet bytů.

```
Wire.begin(55); // inicializuje rozhraní I2C jako Slave a nastaví řídicí
                   jednotce adresu 55
void data_request(){ // metoda volaná v případě, že Master žádá o data
     Wire.write(50); // pošle Masteru číslo 50
};
void data_receive(int howmany){ // metoda volaná v případě, že Master posílá
                                   data, jejím parametrem je počet přijatých
                                   bytů
     int data=Wire.read(); // přečte celé číslo poslané Masterem
};
Wire.onRequest(data_request); // nastaví metodu, která se má provést, když
                                 Master žádá o data
Wire.onReceive(data_receive); // nastaví metodu, které se má provést, když
                                 Master posílá data
```

3.1.2.4 Řídicí jednotka jako Master

Pokud je řídicí jednotka nastavena jako Master používá se k posílání dat přes sběrnici I^2C metoda beginTransmission, které se jako argument předává adresa Slave zařízení, v kombinaci s metodou write, po odeslání všech bytů je nutné ukončit přenos pomocí metody endTransmission.

```
Wire.beginTransmission(50); // metoda začne přenos dat do zařízení s adresou
                            50
Wire.write(100); // metoda odešle jeden byte obsahující číslo 100
Wire.endTransmission(); // metoda ukončí přenos
```

Řídicí jednotka nastavená jako Master si žádá o data od zařízení Slave pomocí metody requestFrom, které jsou předávány dva argumenty, prvním je adresa zařízení a druhým je počet bytů, které jsou po Slave zařízení požadovány. Data jsou následně čtena z bufferu pomocí metody read kombinované s metodou available.

```
Wire.requestFrom(50,1); // zažádá o jeden byte od Slave zařízení s adresou 50
while(!Wire.available()); // čeká dokud nejsou ve vstupním bufferu žádná data
int data=Wire.read(); // přečte jeden byte z bufferu a zapíše jej do proměnné
                      data
```

3.1.3 Obsluha dodatečných knihoven

Vzhledem k tomu, že knihovny dodávané přímo Arduinem© neobsahují a nemohou obsahovat knihovny pro veškerý hardware, který je možné k Arduinu© připojit, bylo nutné přidat další knihovny vytvořené lidmi z komunity vytvořené kolem této řídicí jednotky. Jedná se například o knihovnu pro obsluhu kompasu nebo ultrazvukových senzorů.

3.1.3.1 Knihovna Ultrasonic

Jednou z nejdůležitějších dodatečných knihoven je knihovna Ultrasonic, jedná se o knihovnu vytvořenou firmou ITead studio. Tato knihovna obsahuje všechny funkce potřebné pro čtení vzdálenosti z ultrazvukových senzorů a také pro převod délky pulsu, který určuje vzdálenost sensoru od překážky, na centimetry, nebo palce. Třída Ultrasonic má jeden konstruktor, ve kterém se objektu předává číslo trigger a echo pinu. Samotné měření probíhá pomocí metody Ranging, které se předává jako argument jednotka, ve které má být vrácen výsledek. Argument má dvě možné hodnoty 0 a 1 reprezentované konstantami CM a IN. Metoda Ranging vrací naměřenou vzdálenost jako datový typ long.

```
Ultrasonic senzor(1, 2); // vytvoří objekt třídy Ultrasonic, přičemž
                            ultrazvukový senzor je připojený k pinům 1 a 2
long range=senzor.Ranging(CM); // změří vzdálenost mezi senzorem a překážkou,
                                  výsledek převede na centimetry a zapíše jej
                                  do proměnné range
```

3.1.3.2 Knihovna HCM5883L

Na robotovi je pro precizní otáčení o určitý úhel nainstalován senzor magnetického pole HCM5883L, o jehož ovládání a čtení dat z něj se stará knihovna HMC5883L. Tato knihovna je založena na knihovně Wire, přistupuje se k ní pomocí objektu třídy HMC5883L. Každý objekt musí být nastaven pomocí metod SetScale a SetMeasurementMode, první jmenovaná slouží k nastavení citlivosti senzoru a jako parametr se jí předává celé číslo například 1,3, které určuje citlivost v Gaussech, druhá metoda nastavuje mód měření – jestli bude měření probíhat nepřetržitě, nebo se zastaví po jednom měření.

```
HMC5883L compass; // vytvoří objekt senzoru magnetického pole
compass = HMC5883L(); // inicializuje objekt
compass.SetScale(1.3); // nastaví citlivost senzoru na 1.3 Gauss
compass.SetMeasurementMode(Measurement_Continuous); // nastaví mód měření na
                                                     nepřetržité měření
```

Vlastní měření probíhá tak, že se pomocí metody ReadScaledAxis přečtou honoty všech tří os a ty jsou poté zapsány do objektu MagnetoMeterScaled, z těchto hodnot se poté vypočítá natočení senzoru vůči jižnímu magnetickému pólu Země. Ke zjednodušení výpočtu natočení byla vytvořena metoda getRobotHeading, které je jako parametr předávána hodnota, která určuje, jestli bude výsledek v radiánech, nebo stupních. Metoda navrací desetinné číslo datového typu float obsahující výsledek výpočtu.

```
#define RAD 0 // možné módy výsledku – stupně, nebo radiány
#define DEG 1
float getRobotHeading(int mode) {
      MagnetometerScaled scaled = compass.ReadScaledAxis(); // přečte data
      float heading; // proměnná do které se zapíše výsledek
      heading = atan2(raw.YAxis, raw.XAxis); // spočítá úhel
      heading += 0.067; // k úhlu přičte úhel magnetické deklinace Země
      if (heading < 0) { // převede výsledek z intervalu (-180;180) na <0;360)
            heading += 2 * M_PI;
      }
      if (mode == RAD) { // podle módu výsledku vrátí buď výsledek v
                         radiánech, nebo vrátí výsledek ve stupních
            return heading;
      } else {
            return (heading * 180 / M_PI); // přepočítá radiány
      }
}
```

3.1.4 Obsluha knihoven zjednodušujících rutiny na robotovi

Pro zjednodušení rutin na robotovi, jako je například ovládání motorů, uchopovacího mechanismu atd., byly vytvořeny knihovny, pomocí kterých je lze efektivněji provádět. Jedná se o knihovny Motor, sloužící k ovládání dvou stejnosměrných motorů, knihovna Grabbers, sloužící k ovládání uchopovacího mechanismu, knihovna Radar, zjednodušující ovládání radaru a čtení z něj, a knihovna Joystick, sloužící ke čtení dat z analogového joysticku a jejich vyhodnocování.

3.1.4.1 Knihovna Motor

Knihovna Motor slouží k ovládání dvou stejnosměrných motorů, pomocí jakéhokoliv H- můstku s ovládáním pomocí jednoho pinu určujícího rychlost a dvou pinů určujících směr otáčení. Třída Motor má jeden konstruktor, ve kterém jsou instanci třídy předávány čísla pinů Arduina©, ke kterým je připojen H-můstek. Motory jsou poté ovládány pomocí metod, jejichž názvy korespondují s pohyby robota – například metoda forward slouží k pohybu robota rovně. Knihovna také obsahuje metodu writeDirectCommand, které se předává číslo příkazu, které je získáno například čtením dat z joysticku. Rychlost motorů lze nastavit pomocí tří metod, první metodou je metoda setGlobalSpeed, která nastaví rychlost, která je metodě předávána jako parametr, oběma motorům, druhé dvě metody jsou metody setLeftSpeed a setRightSpeed, které nastaví rychlost předávanou jako parametr každému motoru zvlášť.

```
Motor mot(11, 12, 21, 22, 23, 24); // vytvoří instanci třídy Motor, nastaví piny
                                   rychlosti na 11 a 12 a řídicí piny na 21, 22,
                                   23 a 24
mot.setGlobalSpeed(255); // nastaví rychlost obou motorů na 255 - maximální
mot.forward(); // robot pojede vpřed
mot.left(); // robot se bude otáčet vlevo
mot.writeDirectCommand(RIGHT); // robot se bude otáčet vpravo
```

3.1.4.2 Knihovna Grabbers

Knihovna Grabbers slouží k ovládání uchopovacího mechanismu, má dva konstruktory, první konstruktor předává knihovně čísla pinů, ke kterým jsou připojeny servomechanismy uchopovacího mechanismu, druhý konstruktor předává knihovně vyjma čísel pinů, ke kterým jsou připojeny servomechanismy uchopovacího mechanismu, také číslo pinu, ke kterému je připojen koncový spínač. Po vytvoření instance třídy Grabbers je nutné nastavit mezní hodnoty pro každý servomechanismus – určit ve který pozicích servomechanismů bude uchopovací mechanismus otevřen a ve kterých bude zavřen. K otevření uchopovacího mechanismu se používá metoda release a k uzavření mechanismu se používá metoda grab nebo metoda grabWithEndstop v závislosti na tom, jestli je použit koncový spínač, nebo ne.

```
Grabbers uMech(22, 23, 52); // vytvoří instanci třídy Grabbers, nastaví čísla
                               pinů servomechanismů na 22, 23 a číslo pinu
                               koncového spínače na 52

uMech.leftGrab = 130; // nastaví koncové hodnoty servomechanismů

uMech.leftRelease = 10;
uMech.rightGrab = 70;
uMech.rightRelease = 180;
uMech.grab(); // zavře uchopovací mechanismus
uMech.release(); // otevře uchopovací mechanismus
```

3.1.4.3 Knihovna Radar

Knihovna Radar slouží k ovládání radaru umístěného na pravé části uchopovacího mechanismu, jedná se o rozšíření a spojení knihoven Servo a Ultrasonic. Třída radar má jeden konstruktor, pomocí kterého se instanci předávají čísla pinů, ke kterým je připojen ultrazvukový senzor a číslo pinu, ke kterému je připojen servomechanismus. K nastavení úhlu radaru slouží metoda setAngle, které se jako parametr předá úhel natočení radaru. Čtení vzdálenosti probíhá pomocí metody getDistance, která vrací naměřenou vzdálenost v centimetrech, popřípadě lze použít metodu getDistanceAt, která změní úhel natočení radaru předaný jako parametr a vrátí naměřenou vzdálenost v tomto konkrétním úhlu natočení.

```
Radar rad(50, 51, 32); // vytvoří instanci třídy Radar, nastaví čísla pinů
                          senzoru a číslo pinu servomechanismu
rad.setAngle(90); // nastaví úhel servomechanismu na 90 stupňů
int dist = rad.getDistanceAt(91); // změří vzdálenost mezi senzorem a
                                     překážkou při úhlu natočení 91°
```

3.1.4.4 Knihovna Joystick

Knihovna Joystick slouží ke čtení dat z analogového Joysticku a jejich následnému vyhodnocování. Třída Joystick má jeden konstruktor, ve kterém jsou definovány ADC piny ke kterým je joystick připojen a pin, ke kterému je připojeno tlačítko, které slouží k detekci stisknutí joysticku. Ke čtení pozice joysticku byla vytvořena metoda getPosition, která vrací pozici joysticku pomocí čísel, která korespondují s čísly, pomocí kterých lze ovládat motory za použití metody writeDirectCommand. Chování třídy Joystick lze konfigurovat pomocí metody setDeadZone, které se jako parametr předává číslo, které určuje zónu na každé ose, ve které bude joystick neaktivní a ve které bude metoda getPosition vracet nulu. Ke zjištění zda-li je stisknuto tlačítko joysticku slouží metoda isPressed, která vrací 1, nebo nula v závislosti na tom, jestli je tlačítko stisknuto.

```
Joystick joy(A14, A15, 3); // vytvoří instanci třídy Joystick a nastaví piny,
                           na kterých je joystick připojen
joy.setDeadZone(50); // nastaví hodnotu netečné zóny na 50
mot.writeDirectCommand(joy.getPosition()); // hýbe s motory v závislosti na
                                              výstupu z joysticku
int pressed = joy.isPressed(); // zjistí, jestli je joystick stisknut
```

3.2 Hlavní program robota

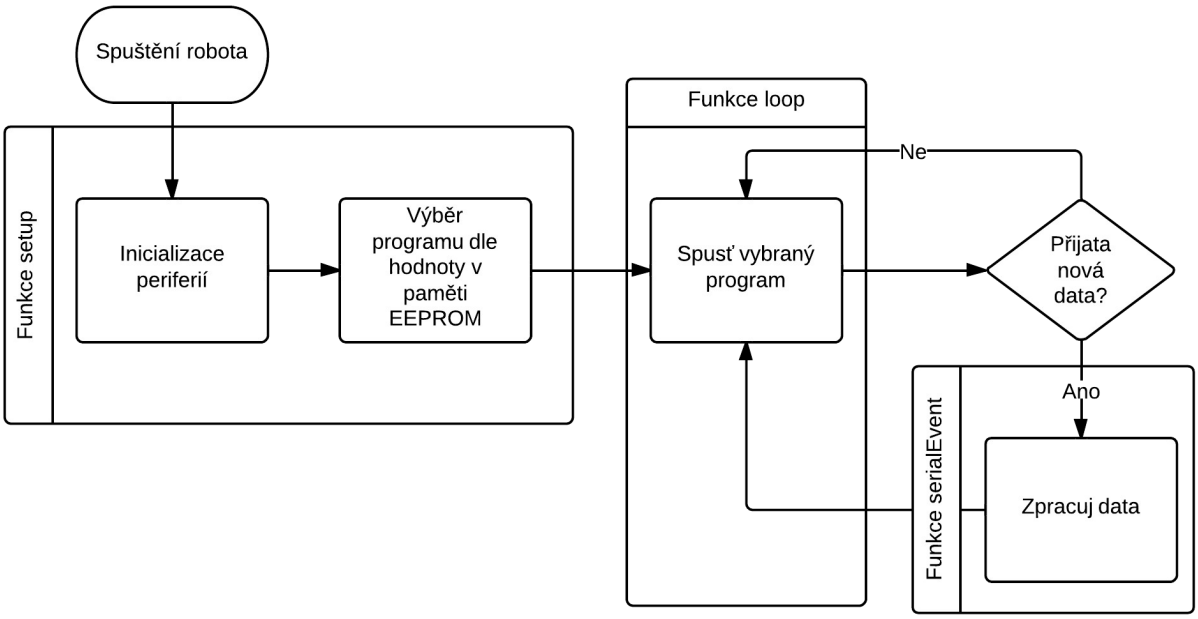
Hlavní program robota se stará o inicializaci všech periferií (senzorů, motorů, servomechanismů), spouštění programů, zpracování dat přicházejících po sériové lince z Bluetooth® modulu a RaspberryPI©.

Nejprve ve funkci setup dojde k inicializaci periferií a poté dojde k výběru programu, který se má spustit, pomocí hodnoty zapsané v paměti EEPROM.

Následně se spustí nekonečná smyčka, ve které je umístěna metoda loop a metody serialEvent, které se volají pouze v případě, že jsou v bufferu některé ze sériových linek nová data. V metodě loop se podle toho, který program je zvolen, provede volání příslušné metody obsluhující činnost robota. V metodách serialEvent dojde k analýze příchozích dat a jejich následnému umístění do příslušného bufferu, se kterým pracuje konkrétní metoda volaná z metody loop.

```
void loop() {
      if (FLAG_JOY) { //ovládání pomocí joysticku
            joystick_control();
            if (FLAG_joy_press) {
                  if (grab.isGrabbed()) {
                        grab.release();
                  } else {
                        grab.grabWithEndstop();
                  }
                  FLAG_joy_press = false;
            }
      } else if (FLAG_BEAR) { //autonomní mód hledání medvěda
            bear_mode();
      } else if (FLAG_TEST) { //testovací algoritmy

      }
}
```

```
void serialEvent2() {
      if (FLAG_FIRST_BYTE_RCVD == false) {
            FLAG_JOY = false;
      }
      FLAG_FIRST_BYTE_RCVD = true;
      BT_buffer[BT_buffer_index] = BT.read(); //čti příchozí byte
      BT_buffer_index++;
      if (BT_buffer_index == BT_buffer_length) {
            BT_buffer_index = 0;
            Serial.println(BT_buffer[1], DEC);
            switch (BT_buffer[1]) { //vyber program
            case BTP: //ovládání pomocí bluetooth
            {
                  FLAG_BT = true;
                  switch (BT_buffer[2]) {
                  case SIMPLE_COMMAND:
                        BT_control(BT_buffer[3], 0); //vykonej příkazy přijaté
                                                      přes bluetooth
                        break;
                  case COUMPOUND_COMMAND:
                        BT_control(BT_buffer[3], BT_buffer[4]);
                        break;
                  }
            }
                  break;
            case JOY:
                  FLAG_JOY = true;
                  break;
            case BEAR:
                  FLAG_BEAR = true;
                  break;
            case DIAG:
                  FLAG_TEST = true;
                  break;
            case KILL:
                  FLAG_JOY = false; //stop program
                  FLAG_BT = false;
                  FLAG_TEST = false;
                  FLAG_BEAR = false;
                  break;
            }
      }
}
```



Ilustrace 32: Vývojový diagram funkce robota

3.3 Software pro RaspberryPI©

RaspberryPI© lze díky tomu, že je založeno na linuxové distribuci programovat v téměř libovolném programovacím jazyce, přičemž výrobce vyzdvihuje zejména skriptovací jazyk Python©, pro který existuje v současné době asi nejvíc knihoven umožňujících přístup k periferiím RaspberryPI©. Řídicí jednotka RaspberryPI© použitá na robotovi byla naprogramována pomocí jazyka Java™. Pro přístup k periferiím Raspberry Pi© byl použit soubor knihoven pi4j, které obsahují již předvytvořené třídy pro přístup k GPIO, sériovým portům a sběrnici I2C.

Pro zjednodušení navázání spojení přes sériovou linku byla naprogramována třída Robot., jedná se o jednoduchou třídu, která v konstruktoru vytvoří spojení přes sériový port a nastaví mu listener, který se stará o zpracování dat, který se předává jako parametr konstruktoru. Třída dále obsahuje metodu pro zápis, metodu pro rekonstrukci 16 bitového integeru ze dvou bytů a metodu pro konverzi proměnné typu String na pole bytů.

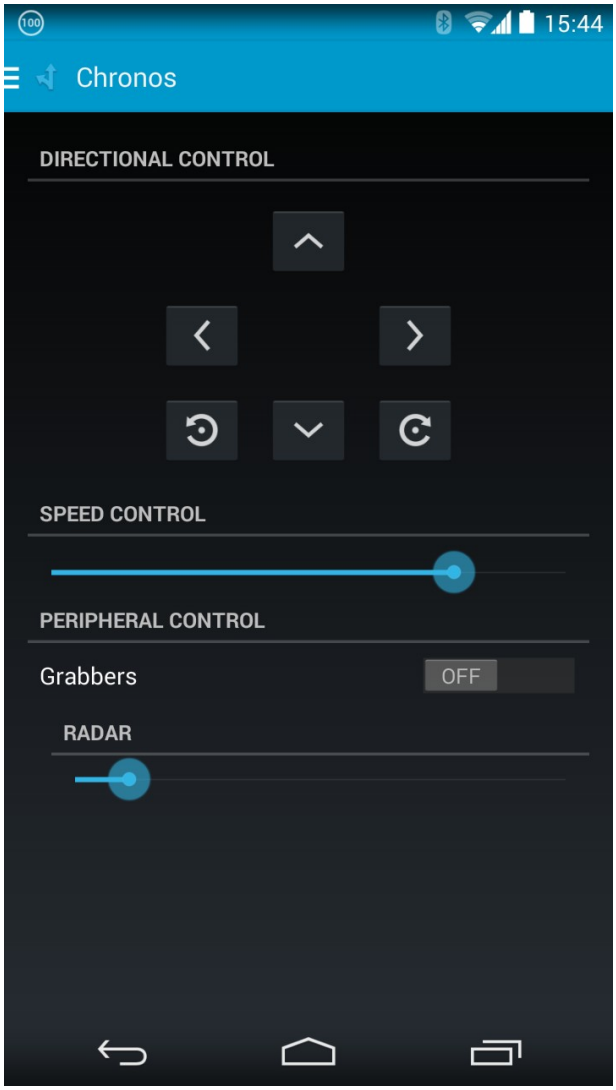
```
import com.pi4j.io.serial.Serial;
import com.pi4j.io.serial.SerialDataListener;
import com.pi4j.io.serial.SerialFactory;
import com.pi4j.io.serial.SerialPortException;

public class Robot {

    private static final int SERIAL_BAUD = 115200;
    private static final String SERIAL_PORT = "/dev/ttyACM0";

    private Serial serial;

    public Robot(SerialDataListener serialDataListener) {
        serial = SerialFactory.createInstance();
        try {
            serial.open(SERIAL_PORT, SERIAL_BAUD);
        } catch (SerialPortException e) {
            e.printStackTrace();
            System.exit(-1);
        }
        serial.addListener(serialDataListener);
    }

    public byte[] getByteData(String data) {
        return data.getBytes();
    }

    public void write(byte[] data) {
        serial.write(data);
    }

    public static int parse16BitInteger(byte one, byte two) {
        if (one < 0) {
            one += 255;
        }
        return (two << 8) + one;
    }

}
```

3.4 Software pro mobilní telefon s OS Android©

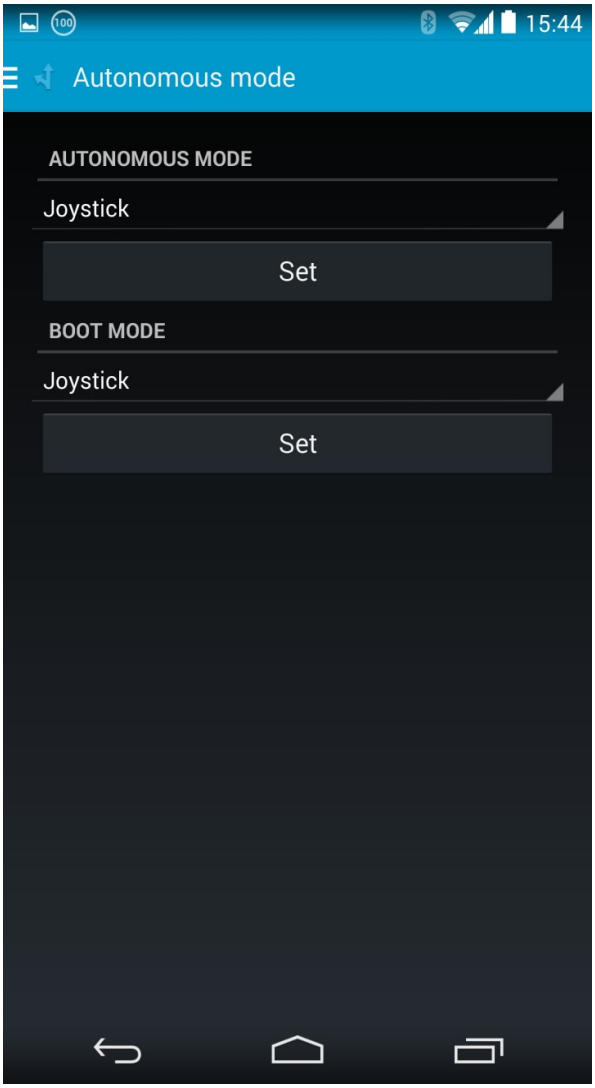
Robota lze manuálně ovládat dvěma způsoby, a to pomocí připojeného joysticku, který ale umožňuje ovládání pouze směru pohybu a otevírání uchopovacího mechanismu, a pomocí aplikace Chronos, která umožňuje pokročilé ovládání robota pomocí rozhraní Bluetooth® – díky ní je možné nejen ovládat směr pohybu robota a pohyb uchopovacího mechanismu, ale také zjišťovat aktuální hodnoty senzorů, ovládat radar, spouštět autonomní úlohy robota a nastavovat, co bude robot dělat po zapnutí.

Aplikace byla naprogramována v programovacích jazycích Java™ a XML, za použití IDE AndroidStudio©. Pro zpřehlednění ovládací aplikace bylo použito vysunovací menu SlidingDrawer, díky kterému se po kliknutí na položku v něm otevře Fragment obsahující příslušné ovládací funkce. Aplikace má v tomto menu celkem tři položky, a to Basic control, jejíž Fragment umožňuje ovládání robota pomocí šipek, Diagnosis, jejíž Fragment zobrazuje stav senzorů a umožňuje ovládání radaru a položka Autonomous mode, jejíž Fragment umožňuje zapnout některý z módu robota a nastavit program, který se zapne při spuštění robota.

Pro zjednodušení práce s rozhraním Bluetooth® byly vytvořeny dvě pomocné třídy Robot a ChronosRobot. Třída Robot se stará o připojení k robotovi a základní I/O operace, zatímco třída ChronosRobot je potomkem třídy Robot a implementuje komunikační protokol robota.



Ilustrace 33: Náhled aplikace - řídicí Fragment



Ilustrace 34: Náhled aplikace - Fragment pro výběr programu

4 Řešení robotických úloh

Robot Chronos je navržen tak, aby byl schopen řešit nejrůznější robotické úlohy – od těch relativně triviálních, jako je například jízda rovně nebo vyhýbání se překážkám, až po složité úlohy vyžadující velký výpočetní výkon nebo strojové vidění, jako například soutěž BearRescue Advanced.

4.1 Jednodušší robotické úlohy

Mezi jednodušší robotické úlohy patří veškeré úlohy, k jejichž řešení lze na robotovi použít pouze řídicí jednotku Arduino©, aniž by došlo k přerušení činnosti robota kvůli nedostatku výpočetního výkonu.

4.1.1 Jízda rovně

Jízdu rovně lze realizovat buď pomocí senzoru magnetického pole použitého jako kompas, nebo pomocí enkodérů. U obou způsobů lze využít dva způsoby implementace – PD regulátor a implementaci pomocí podmínky.

4.1.1.1 Jízda rovně s použitím enkodérů

Robotickou úlohu jízdy rovně pomocí enkodérů lze řešit dvěma způsoby a to, že řídicí jednotka robota měří doby mezi změnami signálů přicházejících z enkodéru a podle těchto dob pak nastavuje korekci rychlostí motorů, nebo měří počty změn signálů obou enkodérů a tímto určí vzdálenost, kterou ujelo každé kolo, a nastaví korekci motorů. Měření doby změn signálů je vhodné zejména u nižších rychlostí a měření ujeté dráhy u větších, vzhledem k tomu že vysokofrekvenční signály není možné efektivně zpracovat. Oba způsoby lze implementovat pomocí podmínek nebo pomocí PD regulátoru, díky kterému nedochází k tak velkým překmitům jako za použití implementace pomocí podmínek, ale je mnohem složitější jej odladit.

4.1.1.1.1 Jízda rovně za použití měření dob změny a implementace pomocí podmínky

Při použití tohoto způsobu řešení je nutné měřit dobu mezi jednotlivými změnami signálu přicházejícího z enkodérů. Tohoto je docíleno pomocí funkce millis(), která je implementovaná v core funkcích řídicích jednotek Arduino© a vrací v milisekundách čas od zapnutí a inicializace řídicí jednotky. Implementace metody, která se provede po přerušení navázaném na změnu signálu přicházejícího z enkodérů pak vypadá takto:

```
int left_tick_time; //proměnná obsahující čas mezi změnami signálu
int last_left_tick_time; // proměnná obsahující čas minulé změny


last_left_tick_time=millis();//inicializace proměnné volaná před začátkem
                             pohybu


void left_interrupt_routine() { //metoda přerušení volaná při změně signálu
     int current_millis=millis();
     left_tick_time=current_millis-last_left_tick_time; //výpočet času mezi
                                                        změnami signálu
     last_left_tick_time=current_millis; //uložení času poslední změny
}
```

Algoritmus tohoto řešení funguje tak, že od sebe odečte čas změny levého enkodéru a čas změny pravého enkodéru a u tohoto výsledku poté zjišťuje, zda-li je menší, větší nebo roven nule a podle toho pak upraví rychlosti motorů.

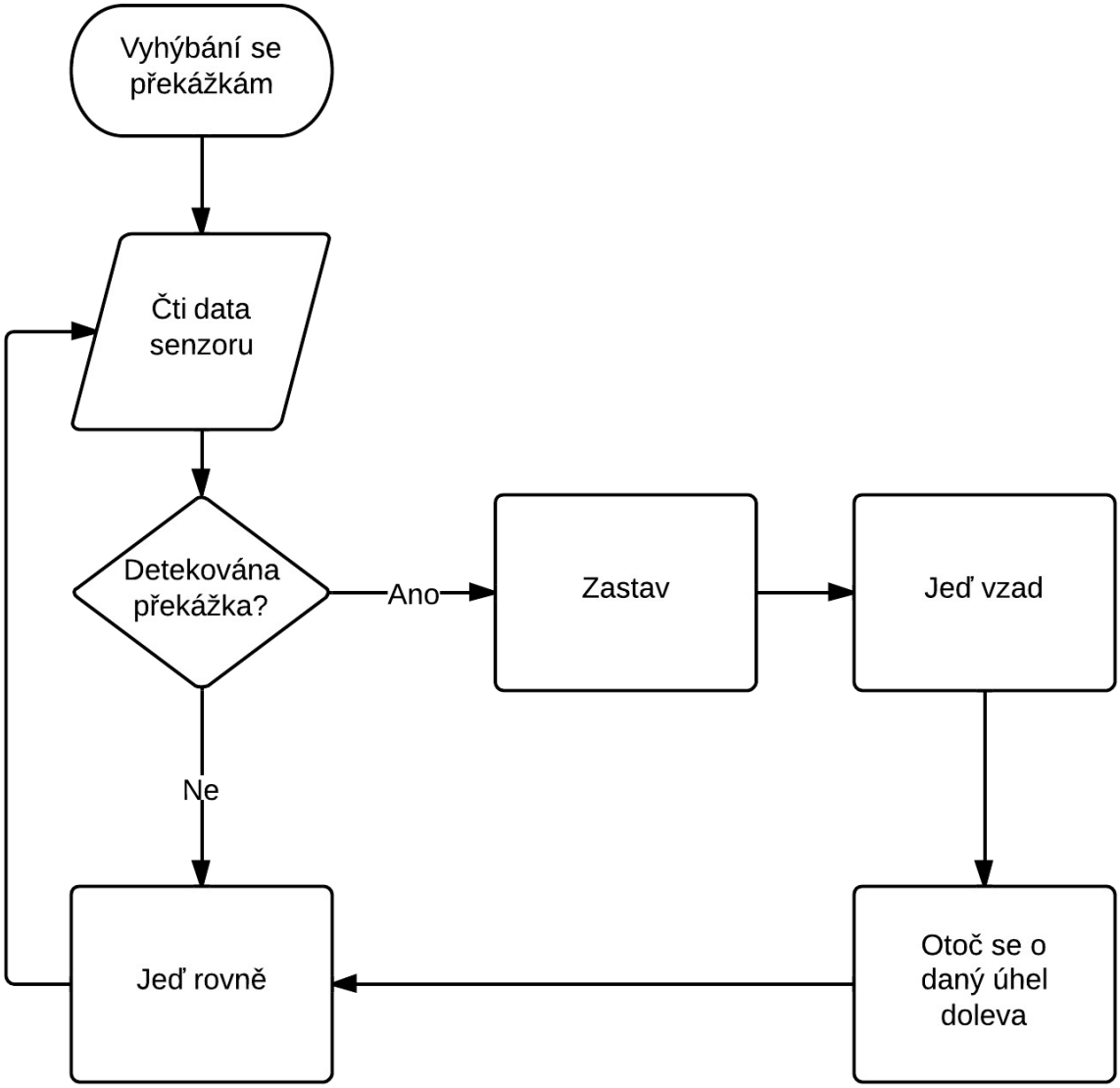
```
void go_straight_condition_time(){
     int diff=left_tick_time-right_tick_time; //vypočítá rozdíl mezi časy
                                              změn signálu
     if(diff>0){ //pokud je čas změny levého enkodéru větší než pravého sniž
                 ryclost pravého motoru
          eng.setLeftSpeed(200);
          eng.setRightSpeed(130);
     }else if(diff<0){//pokud je čas změny levého enkodéru menší než pravého
                      sniž ryclost levého motoru
          eng.setLeftSpeed(130);
          eng.setRightSpeed(200);
     }else{ //pokud jsou si časy rovny nastav rychlost obou motorů na stejnou
            hodnotu
          eng.setGlobalSpeed(200);
     }
}
```

4.1.2 Jízda robota bez kolize s předměty

Další jednoduchou robotickou úlohou implementovanou na robotovi je jízda, během které se robot snaží vyhnout překážkám, které jsou před ním a hrozila by mu s nimi kolize. Při této úloze robot kontinuálně, pomocí radaru, snímá prostor před sebou a v případě, že senzor radaru detekuje předmět ve kratší vzdálenosti od robota, než je „bezpečná“ vzdálenost, zastaví robot, popojede o určitou vzdálenost vzad, otočí se o předem daný úhel doleva a pokračuje v pohybu vpřed, dokud senzor radaru opět nedetekuje překážku před robotem.

Jedná se o extrémně jednoduchou úlohu, která by mohla být rozšířena například tak, že by robot po detekci překážky zjistil, zda-li je další překážka vlevo nebo vpravo a podle toho by zvolil směr svého otočení, tak aby se vyhnul i další překážce.

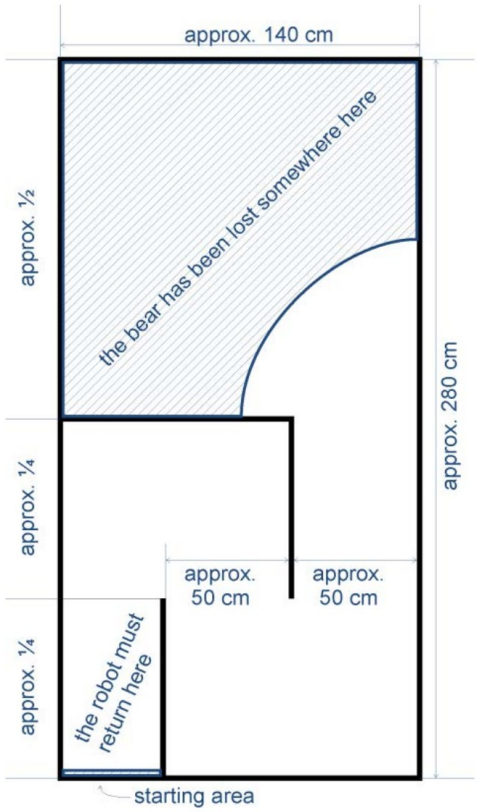
```
void obstacle_avoiding(){
      if(rad.getDistance()<20){ //pokud je vzdálenost mezi překážkou a robotem
                                  menší než 20cm
            eng.stop(); //zastav
            eng.backward(); //jeď vzad
            delay(1500);
            eng.rotateLeft();//otoč se vlevo
            delay(500);
      }else{
            eng.forward(); //pokud je vzdálenost větší než 20cm jeď rovně
      }
}
```



Ilustrace 35: Vývojový diagram jizdy bez kolize s předměty

4.2 Robotické úlohy pro soutěž Bear rescue

Soutěž Bear rescue je jedna ze soutěží, kterou pořádá Matematicko-fyzikální fakulta Univerzity Karlovy v rámci Robotického dne. Úlohou robota při této soutěži je projet jednoduchým bludištěm, najít plyšového medvěda v určitém prostoru a přivézt jej na start.



Ilustrace 36: Aréna pro soutěž Bear rescue

4.2.1 Projíždění bludištěm

Než se robot dostane k oblasti, kde je medvěd, musí nejprve projet bludištěm. Bludištěm projede tak, že jede rovně dokud není v určité vzdálenosti od protější stěny, vyhodnotí, kde se nachází a podle toho se otočí o 90° doprava nebo doleva. Toto se opakuje několikrát, podle počtu zatáček. Sofistikovanějším a univerzálnějším řešením je, že robot po zastavení před protější zdí vyhodnotí data z pravého a levého senzoru, porovná je a podle toho, kterým směrem je větší volný prostor, tak tím směrem se otočí a pokračuje v algoritmu.

```
for (int i = 0; i < 4; i++) {
            motors.forward();
            while (radar.getDistance() > 10) {
                  delay(20);
            }
            motors.stop();
            if (i < 2) {
                  motors.rotateRight();
            } else {
                  motors.rotateLeft();
            }
            while (preciseTickCount < ENCODER_90_DEGREES) {
                  NINETY_LEFT_FLAG = true;
                  delay(20);
            }
            preciseTickCount = 0;
            NINETY_LEFT_FLAG = false;
            motors.stop();
      }
```

4.2.2 Hledání medvěda

Medvěda může robot hledat různými způsoby, nejpoužívanější způsoby jsou obvykle strojové vidění, nebo laserový skener, tyto metody jsou ale náročné na software i hardware. Jedním z nejjednodušších způsobů hledání medvěda je ten, že robot pojede kolem pravé stěny arény a bude měřit vzdálenost naměřenou levým ultrazvukovým senzorem a pokud se objeví odchylka od skutečné vzdálenosti ode zdi větší než například 10 cm lze předpokládat, že medvěd byl nalezen. Robot se poté otočí na místě o 90°, otevře uchopovací mechanismus, pojede k medvědovi, nabere ho, otočí se o 180°, pojede zpět, otočí se o 90° doprava a projede bludištěm.

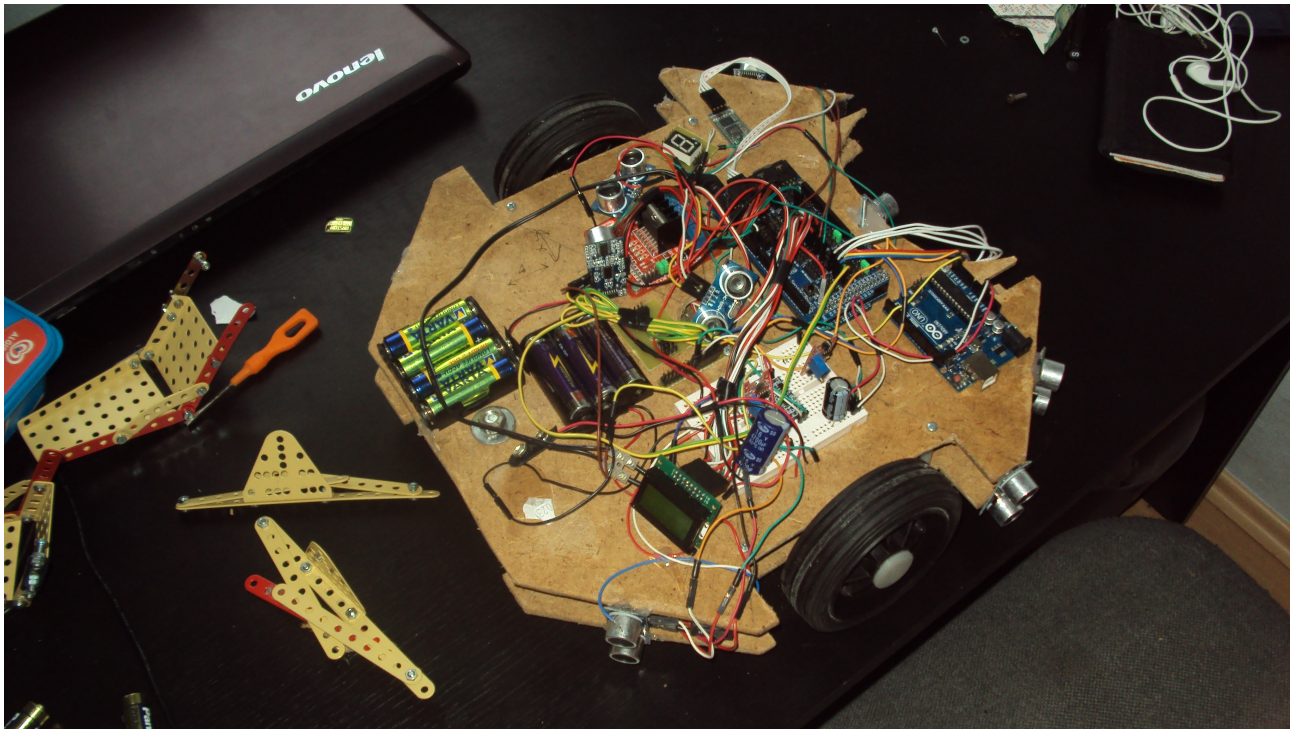
5 Možnosti dalšího vývoje robota

V době psaní této práce nebyl robot ani zdaleka hotov, chybělo dokončit některé robotické algoritmy. Možností dalšího vývoje robota existuje velké množství, jako nejdůležitější bych viděl výměnu akumulátorů za akumulátory s vyšší kapacitou, vzhledem k tomu, že použité akumulátory se při současné zátěži relativně rychle vybíjejí. V dalším vývoji by také mělo dojít k instalaci laserového skeneru, jako náhrady za stávající radar, vzhledem k jeho vysoké nepřesnosti, dále by mělo dojít k vytvoření bezdrátového ovladače s joystickem, který by umožňoval rychlejší řízení, než je stávající řešení pomocí mobilního telefonu.

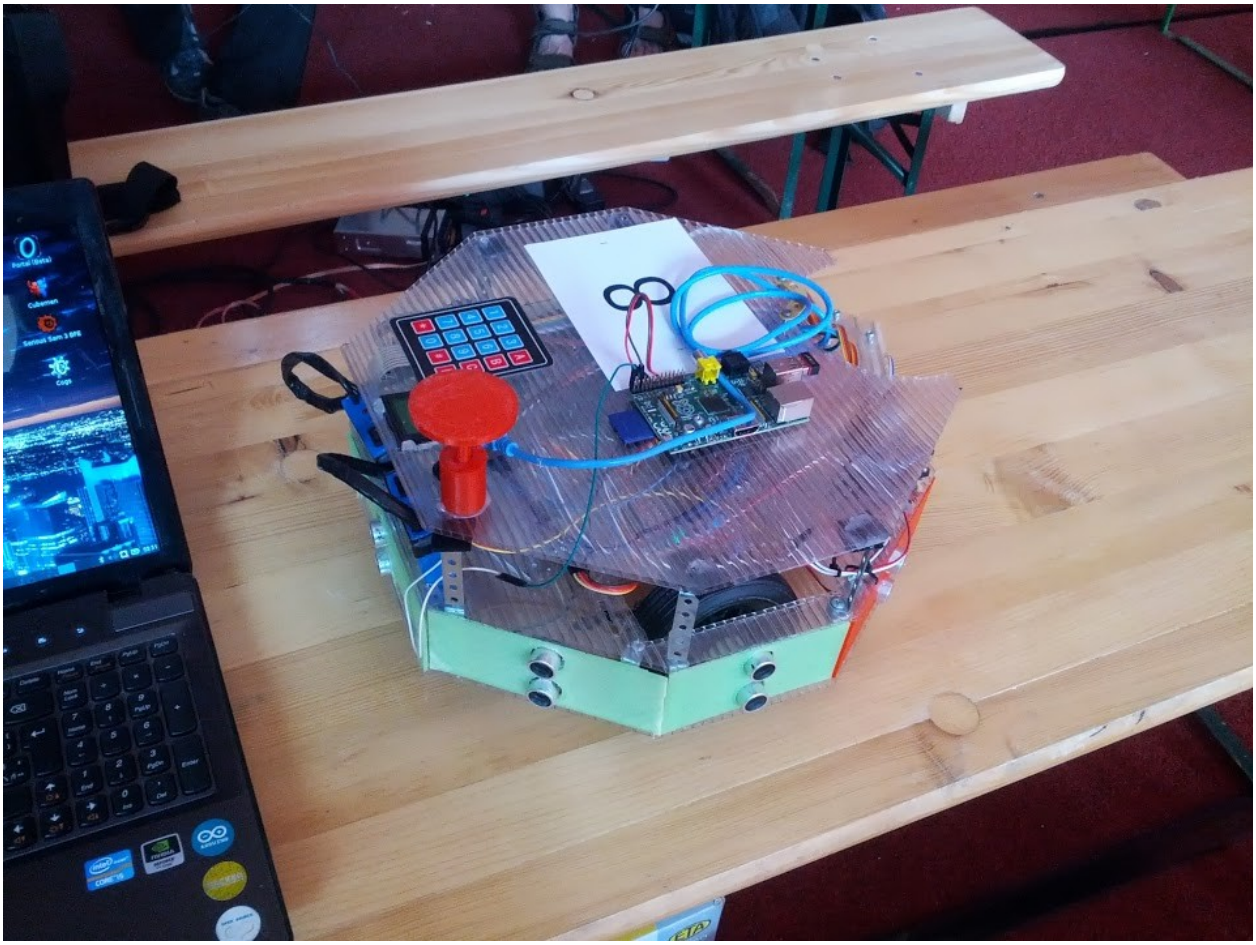
6 Závěr

Vývoj robota trval dosud téměř dva roky a ještě zdaleka není u konce, prošel přibližně třemi verzemi hardwaru a bylo naprogramováno mnoho verzí softwaru, přičemž ty nejdůležitější změny na robotovi se odehrály v létě roku 2013, kdy došlo ke kompletnímu přepracování celé platformy. Díky robotovi jsem se naučil pracovat s CAD softwarem a zlepšil si své znalosti týkající se programování a elektroniky. Robota budu nadále vylepšovat, aby byl schopen účastnit se soutěže BearRescue Advanced, protože předchozí verze byla schopná účastnit se pouze kategorie Beginner, ve které se díky mým chabým řidičským schopnostem a své těžkopádnosti umístil na čtrnáctém místě.

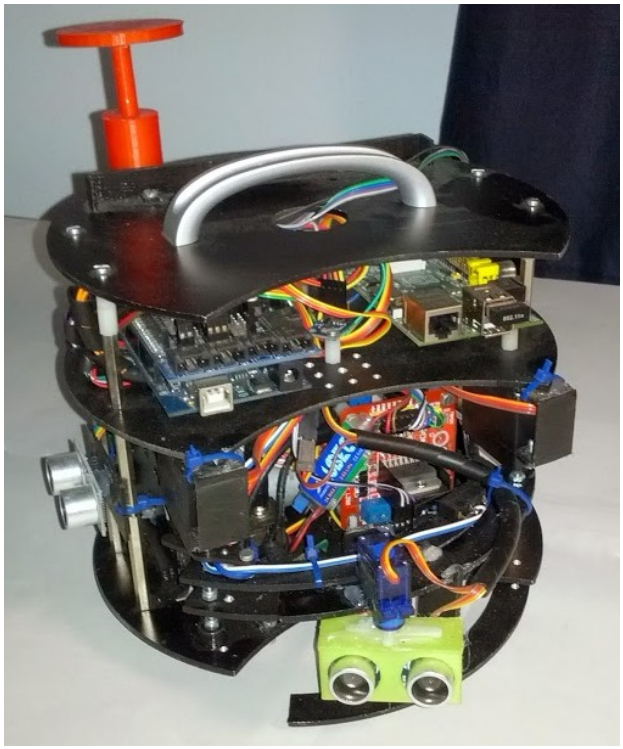
6.1 Fotografie předchozích verzí



Ilustrace 37: První verze robota



Ilustrace 38: Druhá verze na Robotickém dni 2013



Ilustrace 39: Třetí, poslední verze

7 Použitá literatura

1. STANĚK, Ondřej Mgr. PocketBot: robot do krabičky od zápalek. Robot revue: magazín ze světa robotiky. Praha: RCR, 2009-2011, roč. 2011, č. 1. DOI: 1804-056X.
2. Arduino: ArduinoBoardMega2560. ARDUINO SA. Arduino [online]. 2013 [cit. 2013-11-17]. Dostupné z: http://arduino.cc/en/Main/ArduinoBoardMega2560
3. ARDUINO SA. Arduino: MainPage [online]. 2013 [cit. 2013-11-17]. Dostupné z: http://www.arduino.cc/
4. RASPBERRYPI FOUNDATION. Raspberry Pi [online]. 2012 [cit. 2013-11-17]. Dostupné z: http://www.raspberrypi.org/
5. MATEMATICKO-FYZIKÁLNÍ FAKULTA UNIVERZITY KARLOVY V PRAZE. Robotický den 2014 [online]. 2013 [cit. 2013-11-17]. Dostupné z: http://www.robotickyden.cz/
6. GOOGLE INC. Android Developers [online]. 2009 [cit. 2013-11-17]. Dostupné z: http://developer.android.com/index.html
7. Raspberry Pi | Wiring | Gordons Projects. HENDERSON, Gordon. Gordons Projects [online]. 2012 [cit. 2013-11-17]. Dostupné z: https://projects.drogon.net/raspberry-pi/wiringpi/
8. Arduino Library For Ultrasonic Ranging Module HC-SR04 | ITead Intelligent Systems Blog. ITEAD STUDIO.ITead Intelligent Systems Blog | Just another WordPress site [online]. 2010 [cit. 2013-11-17]. Dostupné z: http://blog.iteadstudio.com/arduino-library-for-ultrasonic-ranging-module-hc-sr04/
9. PI4J. Pi4j.com [online]. 2012-2013 [cit. 2014-01-01]. Dostupné z: http://pi4j.com/
10. MFF UK. Bear rescue [online]. 2013 [cit. 2014-01-18]. Dostupné z: http://www.roboticday.org/2014/rules/2014-Bear_Rescue-ENv1.pdf
11. HMC5883L Compass Tutorial with Arduino Library - Tutorials - Love Electronics. *Love Electronics* [online]. 2011 [cit. 2014-02-08]. Dostupné z: http://www.loveelectronics.co.uk/Tutorials/8/hmc5883l-tutorial-and-arduino-library