Jihočeská Univerzita v Českých Budějovicích
Přírodovědecká fakulta

Zavlažovací robot s autonomním chováním

Bakalářská práce

Daniel Hrad

Vedoucí práce: Mgr. Jiří Pech, Ph.D.

České Budějovice 2016

Bibliografické údaje

Hrad D., 2016: Zavlažovací robot s autonomním chováním [Watering robot with autonomous behaviour. Bc.. Thesis, in Czech] - 43 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Anotace

Předmětem bakalářské práce je vytvoření pohyblivého robota na zalévání květin s autonomním chováním. Teoretická část poskytuje možnosti použití různých technologií na řešení problémů vztahující se k robotovi. Následuje praktické řešení, které popisuje jednotlivé použité součástky a odůvodňuje jejich výběr. Výsledkem práce je zrealizovaný robot umožňující samostatné zalévání květin a stanice na doplňování vody.

Klíčová slova

Arduino, autonomní, robot, zalévání, automatizace

Annotation

Subject of bachelor thesis is to create mobile robot with autonomous behaviour for watering flowers. Theory part provides options of using different technologies for solving a problems related to robot. Followed by practical part, which describes used components and reason for selection. Result of thesis is real constructed robot able to autonomous watering flowers and station for water refilling.

Keywords

Arduino, autonomous, robot, watering, automation

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury. Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích 22. 4. 2016

Daniel Hrad

Poděkování

Děkuji Mgr. Jiřímu Pechovi, Ph.D. za možnost psát práci na mnou zvolené téma, za velmi ochotný a pozitivní přístup a za rady, které mi poskytl v době zpracovávání této práce.

Obsah

1 Úvod

Současným trendem, který můžeme již delší dobu pozorovat, je automatizace i nejběžnějších každodenních činností. K tomu se využívá sofistikovaných elektronických systémů, případně - obecně řečeno - robotů. Právě odvětví robotiky se stává dostupnější i pro běžné uživatele v domácnostech díky poměrně nízkým cenám robotických součástek a jednoduchosti jejich použití. Faktem může být rozrůstající nabídka mikropočítačů, které mohou najít uplatnění například v zábavě ve formě konzolí.

Bakalářská práce se bude zabývat problematikou zautomatizování jedné z každodenních činnosti - zaléváním květin. Hlavním motivem pro zvolení tohoto tématu byl zájem o robotiku a možnost vyzkoušet si sestavení vlastního robota s mobilním systémem.

Hlavním cílem bude vývoj a reálné sestavení pohyblivého robota schopného samostatného zalévání květin. V teoretické části budou nejprve popsány jednotlivé prvky součástek včetně alternativ. Předmětem praktické části je pak konkrétní realizace, výčet použitých komponent a odůvodnění jejich výběru. Závěr praktické části je věnován logice robota a některých součástí včetně zhodnocení finanční náročnosti zvolené realizace.

2 Cíle

Cílem mé práce je návrh a realizování systému na zalévání květin, tedy návrh a realizace autonomního pojízdného robota, který dokáže zalít květiny. Tento cíl bude provázán s možným finančně nenáročným řešením. Cíl se skládá ze tří hlavních částí. První je konstrukce zmíněného robota, druhá část zahrnuje navigaci a třetí se zabývá stanicí na doplňování vody.

V rámci hlavního cíle jsou vyčleněny následující dílčí cíle, které je nutné splnit

- Návrh a konstrukce pojízdného autonomního robota
- Návrh a konstrukce navigačního systému
- Návrh a konstrukce stanice na doplňování vody

3 Výchozí stav

Bakalářská práce bude vycházet z teorie robotiky a fyziky. Budou nastíněny ostatní vývojové platformy a důvody jejich nezvolení nebo případné rozdíly mezi zvolenou platformou.

V bakalářské práci budou uvedeny všechny komponenty, které jsou použity včetně důvodu jejich zvolení. Částečnou inspirací pro konkrétní konstrukční řešení byla již hotová řešení společně nesouvisejících robotů a jejich navigačních systémů.

Rešeršní činnost byla téměř výhradně elektronická díky své dostupnosti a rychlosti. Jedním z hlavních zdrojů k sestavení a ověření návrhu robota byla práce *Plant Watering Autonomous Mobile Robot* [36], která se zabývá podobnou problematikou, nicméně řešení je odlišné. Mezi další práce patří různé návrhy robotů využívající totožnou vývojovou platformu nebo platformu podobných vlastností.

Při hledání stávajících řešení jsem se snažil výsledky omezit podle jejich ceny pořízení, neboť celý systém na dodávání vláhy rostlinám má být jedním z finančně dostupnějších zařízení, ačkoliv to není prioritou práce.

4 Teoretická část

V této kapitole bude teoreticky popsána část prvků, které byly nutné k realizaci řešení. Součástí bude také alternativní řešení. Výběr skutečně použitých součástek a jeho zdůvodnění bude dále řešeno v praktické části práce.

4.1 Řídicí jednotky

Základem každého samostatného systému, který na základě vstupů může vytvářet určité výstupy, je řídicí jednotka. V analogii s firemním prostředím tuto úlohu zastává ředitel, který na základě požadavků trhu (vstup) může upravit nabídku svých nabízených služeb (výstup). Stejný pohled můžeme použít na stolní počítač. Ten zobrazuje na monitoru pohyb myši (výstup), který je vkládán skrz pohyb fyzické myši na podložce přes USB kabel (vstup).

Lze říci, že tato myšlenka je v principu stejná pro jakékoliv elektronické zařízení.

Při návrhu autonomního robota je potřeba si ujasnit a definovat, která řídicí jednotka bude robota ovládat a řídit. Trh s řídicími jednotkami pro tyto systémy je pestrý a záleží jen na požadavcích, které má řídicí jednotka splňovat. V následující části se budu snažit stručně popsat různé řídicí jednotky (vývojové platformy) vhodné pro robotiku. Podkapitola vychází ze zdrojů [1], [2], [3], [4], [5], [6] a [7].

4.1.1 Typy vývojových platforem

Před každým projektem je nutné si ujasnit, co vyžadujeme. V podstatě se rozdělují vývojové platformy na dva druhy. Prvním z nich je mikrokontrolér (SoC[1]), druhým minipočítač (SBC[2]). Nejznámějším mikrokontrolérem je Arduino, v oblasti minipočítačů zastává roli Raspberry Pi. Na první pohled se může zdát, že se jedná o podobné systémy s výjimkou fyzické konstrukce. Je nutné uvést, že SoC může být pouze jednou částí jakéhokoliv počítače.

Srovnání platforem z hardware a software hlediska je klíčové pro rozhodnutí, kterou platformu si zvolit. Tabulka 1 Stručné porovnání platforem zobrazuje vlastnosti platformy Arduino Uno a Raspberry Pi Model B.

[1] SoC = System on Chip
[2] SBC = Single Board Computer

Tabulka 1 Stručné porovnání platforem

	Arduino Uno	**Raspberry Pi Model B**
Cena	30 USD	35 USD
Velikost paměti	2 KB	512 MB
Frekvence procesoru	16 MHz	700 MHz
Připojení sítě	žádné	Ethernet 10/100 Mbit
Multitasking	ne	ano
Vstupní napětí	7-12V	5V
Flash paměť	32 KB	SD karta (2-16GB)
USB	jeden, typ B	dva, typ A
Operační systém	žádný	Linux
Vývojové prostředí	Arduino	Scratch, cokoliv s Linuxovou podporou

Jediné, co lze srovnávat je cena platforem. To, co srovnatelné není, je uvnitř platforem.

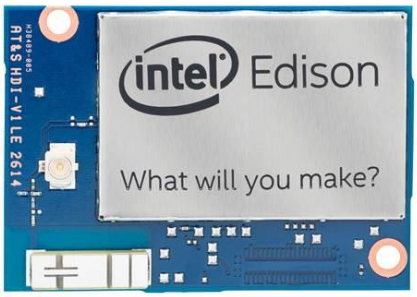
Raspberry Pi má několikanásobně větší frekvenci procesoru a několikanásobně větší paměť. Raspberry je nezávislý počítač, na kterém je možné používat operační systém Linux, u novějších modelů je zabudovaná podpora pro Windows 10. Dále podporuje multitasking a umožňuje se připojit přes Ethernet k síti. Po této stránce se může jevit jako naprosto nadřazený. Ovšem tyto výhody platí pouze pro software aplikace.

Naopak Arduino je nesrovnatelně lepší co se týče projektů využívající hardware. Arduino má flexibilitu a umožňuje pracovat s hardwarem, hlavně dokáže pracovat analogově bez nutnosti dalšího hardwaru. Naopak Raspberry k tomu potřebuje dodatečný hardware. Například pro zapnutí LED diody stačí napsat pár řádků kódu a nahrát na Arduino, kdežto Raspberry potřebuje již nějaké knihovny. Z tohoto porovnání lze usuzovat, že pro menší, spíše hardware projekty, které zahrnují ovládání motorů, řízení sensorů a podobně, je znatelně lepší platforma Arduino. Raspberry se tedy spíše hodí pro softwarové projekty, případně lze také spojit dohromady Arduino a Raspberry.

Mimo zmíněné rozdíly je také rozdíl ve fyzickém principu. SoC platformy, kam se řadí i Arduino, obsahují pouze jeden jediný čip, na kterém je vše a může se použít již samotný oproti Raspberry, který obsahuje zvlášť procesor, paměť a další hardware nutný k základnímu fungování.

4.1.2 Ostatní platformy

Za zmínku určitě stojí také Intel Edison, který stoupá na popularitě a je kompatibilní s platformou Arduino. Nelze opomenout ani mikrokontrolér Picaxe. Mezi minipočítače se řadí i produkty firmy BeagleBone.



Obrázek 1 Intel Edison
zdroj: http://www.intel.com/buy/us/en/product/emergingtechnologies/intel-edison-compute-module-iot-463633

4.2 Navigace

Základní schopností pohyblivého robota, a nejenom robota, je navigace v prostoru.

V této kapitole budou představeny základní typy navigace v prostoru pohyblivého robota. Podkapitola vychází ze zdrojů [8], [9], [22], [23], [24] a [25].

4.2.1 GPS

Mezi moderní způsoby navigace v prostoru je GPS[3]. Tato metoda funguje na bázi rádiových vln vysílaných satelity pohybující se na oběžné dráze Země.

Zjednodušený princip:

- Satelit vyšle rádiový signál, který obsahuje přesnou lokaci satelitu a přesný čas palubních atomových hodin.
- Tento rádiový signál cestuje skrz atmosféru rychlostí větší než rychlost světla.
- Přijímač tohoto signálu na Zemi tento signál zachytí a uloží si přesný čas doby zachycení, tento údaj použije k výpočtu vzdálenosti od emitoru k přijímači.
- Jakmile zachytí signál alespoň od 4 satelitů, přijímač může pomocí těchto hodnot určit svojí geografickou polohu na Zemi.

Přesnost této metody v civilním použití se pohybuje v 95 % případů od 3 - 15 metrů, což lze považovat za nevýhodu, pokud je přesnost klíčovým faktorem.

Výhoda systému je celosvětová funkčnost v otevřeném terénu.

---

[3] GPS = Global Positioning System

Přesnost lze zvýšit pomocí mobilní sítě nebo Wi-Fi, která umožňuje stáhnout přesné polohy satelitů a jejich palubních časů. Tento způsob zvaný asistované GPS (A-GPS) zrychluje lokalizaci.

4.2.2 Navigace pomocí dráhy

Směrovat roboty lze také s využitím charakteristik infračervených sensorů. Jelikož infračervené vlnění je téměř pohlceno černou barvou, lze tohoto jevu využít v navádění. Konkrétně lze sestrojit jakousi dráhu, po které se robot bude pohybovat. K základní funkčnosti je nutné nasadit alespoň 2 infračervená čidla.

Jako výhodu jednoznačně vidím nenáročnost logiky a implementace.

Nevýhoda samotného systému spočívá v neurčení přesné polohy - víme pouze, že je kdesi na dráze.

Ostatní lokalizační systémy využívají podobného stylu jako GPS. Hlavní rozdíl je pouze v použití jiného rádiového vlnění a staticky umístěných vysílačů například společně s 3D mapou oblasti.

4.3 Bezdrátová komunikace

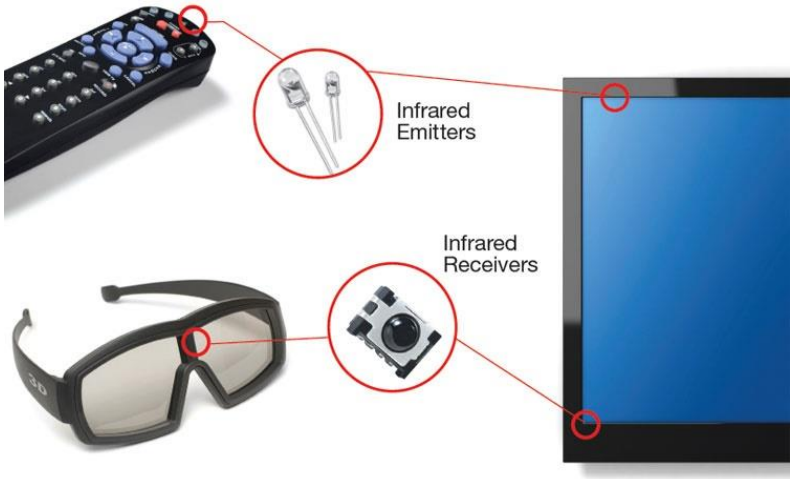
Návrh celého systému zahrnuje mimo jiné i bezdrátovou komunikaci.

Bezdrátová komunikace je přenos informací mezi dvěma elektrickými obvody, které nejsou navzájem fyzicky spojené. Pro přenos informací se využívá rádiových vln kromě infračervené komunikace.

4.3.1 Infračervené světlo

Logika způsobu dorozumívání je totožná v analogii s televizí a dálkovým ovladačem, kde dálkový ovladač představuje infračervený vysílač a čidlo na televizi infračervený přijímač.

Signál je přenášen pomocí vysílaného infračerveného světla s pomocí pulzně šířkové modulace, která umožňuje, aby pro různé příkazy vysílala jednoznačný vzor nul a jedniček, která je druhou stranou jednoznačně identifikována.



Obrázek 2 Časté využití infračervené komunikace
zdroj: http://electronicdesign.com/communications/infrared-still-dominates-short-range-wireless-communications

4.3.2 Bluetooth

Bluetooth se řadí mezi bezdrátové technologie využívající bezlicenčního pásma ISM[4]. Je definována standardem IEEE 802.15.1 a spadá do bezdrátových osobních sítí (WPAN - Wireless Personal Area Network). Pracuje na pásmu 2,4 GHz.

Technologie se řadí do tří tříd, které jsou rozdílné podle vysílacího výkonu a dosahu signálu. Spojení mezi prvky může přenášet data i hlasový signál.

S bluetooth se můžeme setkat například u přenosných reproduktorů, náhlavních sluchátek s mikrofonem či myší nebo klávesnicí.

Obrázek 3 Sluchátko a reproduktor využívající Bluetooth pro spojení s mobilním telefonem zdroj: https://www.plantronics.com/us/category/bluetooth-headsets

4.3.3 ZigBee

Tato technologie je definována standardem IEEE. 802.15.4. Cílem Zigbee je zajistit spolehlivost, rychlost a energetickou nenáročnost spojení. Omezený je datový přenos, proto se technologie hodí na dálkové ovládání.

4.3.4 Wi-Fi

Veřejnosti nejznámější bezdrátovou technologií pro přenos dat je Wi-Fi, definovaná standardem IEEE 802.11. Stejně jako Bluetooth využívá bezlicenčního pásma nejčastěji 2.4 GHz a 5 Ghz. Bylo vytvořeno již několik modulací, které vylepšují přenosové charakteristiky nebo spotřebu energie.

4.4 Pohyb a řízení pohybu

V této kapitole bude představeno řízení pohybu robota. Bude popsána také PID regulace, která souvisí s řízením pohybu. Podkapitola vychází ze zdrojů [10], [11], [17], [18],[26], [27]

4.4.1 Chůze

Napodobování lidské chůze roboty je každým dnem vyspělejší. Humanoidi, jak jsou tito roboti označováni, jsou navrženi a zkonstruováni tak, aby fyzickým vzhledem připomínali lidi. Za prvního vytvořeného humanoida lze považovat tzv. "A Steam Man" zkonstruovaného Zadoc P. Dederickem v roce 1868. Známým humanoidem je také Asimo od firmy Honda z roku 2000.

[4] ISM = Industrial Scientific Medicine

Mezi chodící princip by se dal zařadit i pavoučí typ.

Obrázek 4 pavoučí typ - Lynxmotion CH3-R
zdroj: http://www.robotshop.com/en/lynxmotion-ch3-r-hexapod-robot-kit-botboarduino-w--hs-645-servo-upgrade.html

4.4.2 Rotační pohyb

Nejzákladnějším a nejjednodušším pohybem je rotační pohyb kola. V robotice se nejčastěji můžeme setkat s elektromotory, na jejichž rotační části je umístěno kolo, které je poháněno pomocí silových účinků magnetického pole. Různé druhy se liší velikostí, rychlostí maximálního otáček za minutu (RPM[5]) a celkovou odolností vůči vnějším vlivům a výkonem.

V případě pohybu robota se sudým počtem elektromotorů s fixním upevněním ke konstrukci lze využít výhody řízení pomocí PID regulace, která dokáže měnit rychlost otáček.

PID regulace se zpětnou vazbou

"Cílem regulace je generovat akční veličinu $u(t)$ (napětí, proud) tak, aby se regulovaná veličina $y(t)$ (skutečné otáčky) chovala podle předem zadaného cíle, jenž je charakterizován žádanou veličinou $w(t)$ (žádané otáčky)."[18] K realizování jsou třeba tři složky - proporcionální, integrační a derivační:

$$u(t) = r_0 e(t) + r_1 \int_0^t e(t)dt + r_2 e'(t)$$

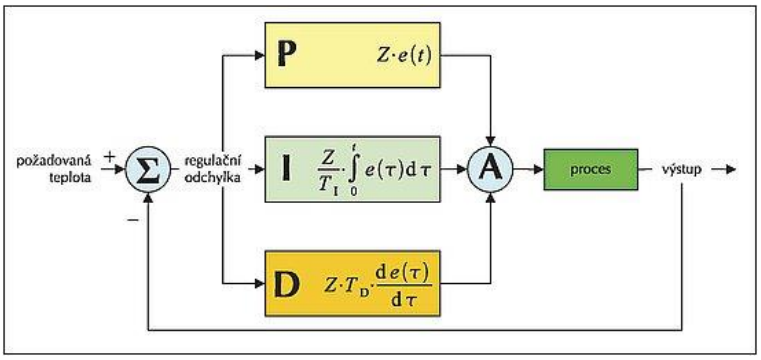
kde r_i jsou parametry: r_0 ... proporcionální zesílení

r_1 ... integrační zesílení

r_2 ... derivační zesílení

Není nutné vždy používat všechny tři složky, ale záleží na konkrétním případě užití regulátoru. Nulová hodnota složky způsobí její deaktivaci.

[5] RPM = Revolutions per Minute



Obrázek 5 Proces PID regulátoru
zdroj: http://www.stavebnictvi3000.cz/clanky/proporcionalni-pi-a-pid-regulatory-ve-vystavbe-a-j/

Vlastnosti složek:

- P složka je úměrná regulační odchylce. Vysoká hodnota může způsobit vzrůstající rozkmitání. Nízká hodnota způsobí trvalou regulační odchylku.
- I složka je úměrná časovému integrálu regulační odchylce. Pomáhá dlouhotrvající malou odchylku snížit na nulovou hodnotu.
- D složka je úměrná rychlosti časové změny regulační odchylky. Předvídá další vývoj regulační odchylky a reaguje s předstihem.

Vzájemným vyladěním hodnot jednotlivých složek se optimalizuje chování regulační odchylky s ohledem na rychlost reakce, přesnost regulace a překmity.

Nejběžněji se můžeme setkat s PID regulací u plynového pedálu automobilu. V tomto příkladě je PID regulátorem řidič, neboť on ovlivňuje plynulost a rychlost své jízdy a tím se snaží minimalizovat regulační odchylku (rozdíl požadované a skutečné rychlosti) od požadované rychlosti.

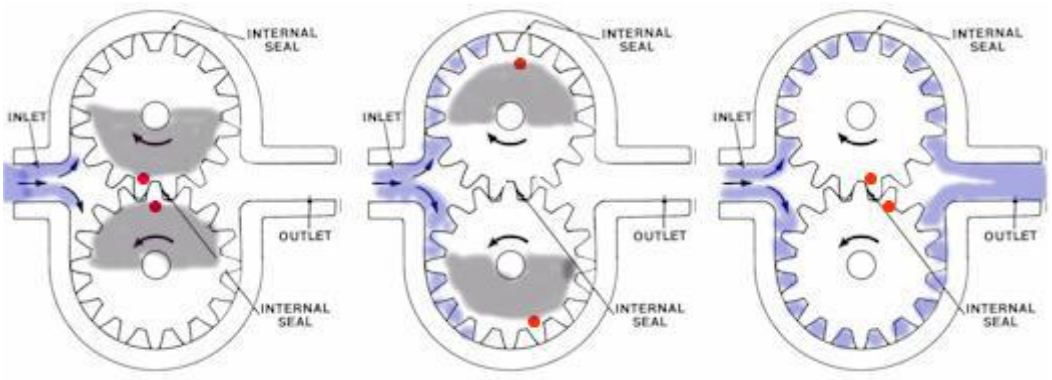
4.5 Řízení vody

V této části představím možnosti řízení vody. Konkrétně, jak a čím přemisťovat vodu z místa A do místa B pomocí nejužívanějších elektrických zařízení a dále možnosti měření hladiny vody. Podkapitola vychází ze zdrojů [12], [13], [14] a [15].

4.5.1 Přemisťování kapaliny

Základní možností, jak přesunout kapalinu, je pomocí čerpadla, nicméně existují různé druhy čerpadel podle konstrukce a systému. Výběr čerpadla závisí na několika faktorech. Může to být druh kapaliny, teplota kapaliny, rychlost čerpání a objem.

V automatizační technice se nejčastěji nasazuje zubové čerpadlo. Zubové čerpadlo přenáší tekutinu pomocí prostoru mezi zuby v uzavřené komoře. Hodí se také pro čerpání pevných materiálů.



Obrázek 6 Jednotlivé fáze funkce zubového čerpadla
zdroj: http://automatizace.hw.cz/principy-prumyslovych-cerpadel-1dil-zubova-cerpadla

Dalším průmyslově nasazovaným čerpadlem je Lobe pumpa. Často se využívá v potravinářském průmyslu, například v mlékárnách. Princip je založen na dvou rotorech točící se vzájemně v opačném směru. Nejčastější počet křídel je 2 - 3.

Obrázek 7 Jednotlivé fáze TRI-Lobe čerpadla
zdroj: http://automatizace.hw.cz/principy-prumyslovych-cerpadel-2dil-rotacni-lobe-pumpy

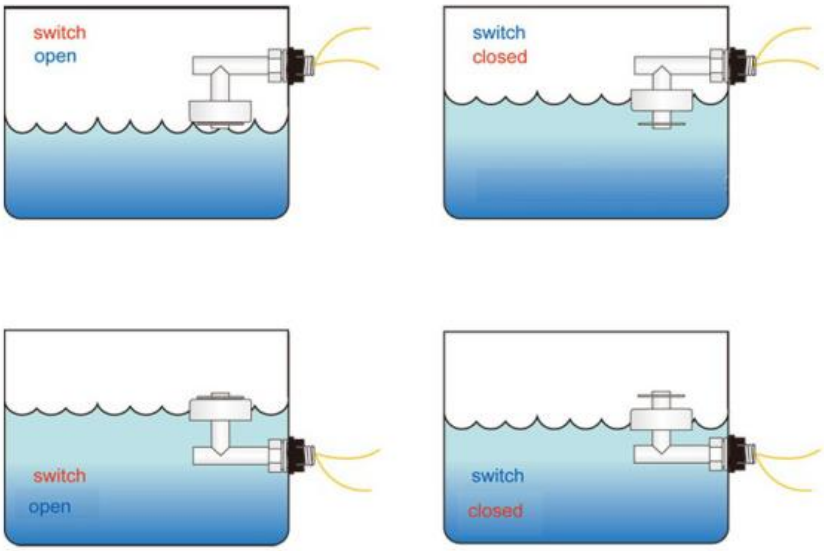
U benzínových nebo naftových motorů se můžeme setkat s lopatkovými čerpadly, které umožňuje čerpání paliva do motoru. Lopatkové čerpadlo se skládá z rotačního motoru, který je vyosen vzhledem ke středu kruhovité komory. V tomto prostoru proudí kapalina poháněná tímto motorem se zásuvnými lopatkami.

Obrázek 8 Lopatkové čerpadlo s vyoseným rotorem
zdroj: http://automatizace.hw.cz/principy-prumyslovych-cerpadel-3dil-lopatkova-cerpadla-vane-pumps

V průmyslu se používá dále například odstředivé čerpadlo, rotační pístové čerpadlo, elektromagnetické čerpadlo a další.

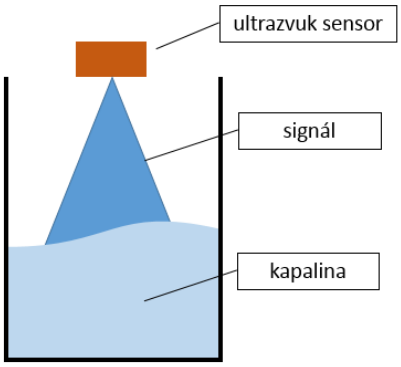
4.5.2 Měření kapaliny v nádobě

Samostatné měření kapaliny lze provést několika způsoby. První způsob, se kterým se můžeme setkat každodenně, se používá v nádobě na splachování u toalet. Zde se využívá plovák, který mechanicky sepne nebo vypne přítok vody do nádoby, pokud se jeho poloha dostane do určité výšky.



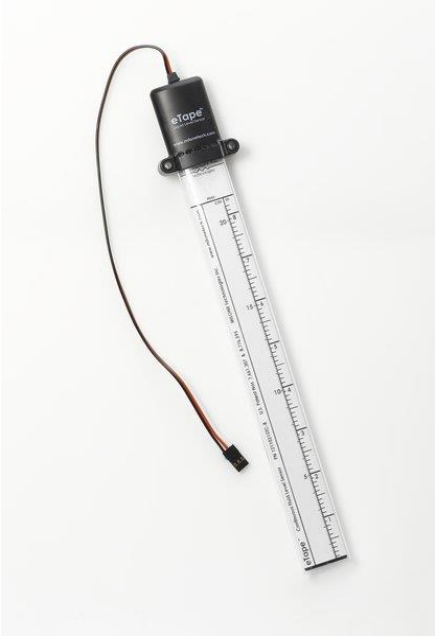
Obrázek 9 Princip plováku
zdroj: http://chioszrobots.com/2015/03/22/liquid-water-level-sensor-float-switch/

Měření kapaliny lze realizovat také pomocí ultrazvuku. Hladinu vody určuje čas, za který se zachytí vyslaný ultrazvuk přijímačem. Ultrazvukové zařízení je nutné mít kolmo k hladině vody. Tato metoda se využívá i u zachycování překážek na dráze, která byla popsána výše. Nevýhoda systému tkví k průměru nádoby, tj. pokud je příliš malý průměr nádoby, přijímač může zachycovat odrazy od stěn nádoby, nikoliv od hladiny.



Obrázek 10 Princip měření ultrazvuku
převzato z http://www.instructables.com/id/Measuring-water-level-with-ultrasonic-sensor/step2/Theory-behind-ultrasonic-level-sensor/

Posledním nejběžnějším měření kapaliny je za pomocí samotné vody. Ta přebírá v této metodě funkci vodiče. Princip závisí na spojení elektrického obvodu a tím se určí dosažený bod v nádobě. Toho lze docílit několika přístupy. První využívá zvolené úrovně a jakmile se voda dostane nad úroveň (spojí elektrický obvod), tak lze binárně určit úroveň hladiny. Druhá metoda funguje s analogovým výstupem a odporem. Tedy čím nižší je hladina vody, tím je vyšší odpor a naopak. Příkladem je výrobek eTape®.



Obrázek 11 eTape® - 20 cm
zdroj: http://milonetech.com/products/standard-etape-assembly

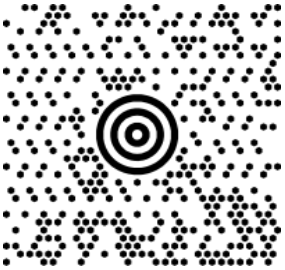
4.6 Typy čtení

Realizace cíle této práce si následně vyžádala systém, který určí pozici robota na určitém bodě. K určení jednoznačné pozice je nutný jednoznačný identifikátor, podle kterého bude jednotka schopna vykonat jedinečnou operaci. Tato podkapitola vychází ze zdrojů [19], [20] a [21].

4.6.1 Čárový kód

Čárový kód je tvořen černotiskem vytištěnými pruhy definovaných rozměrů, umožňující čtení speciálními čtečkami - snímačů čárových kódů. Jednorozměrný čárový kód je rozdělen na několik desítek rovnoměrných sloupců, od nichž se vysílaný laserový paprsek odrazí v případě bílých prostor. Tyto odrazy jsou poté převedeny čtečkou na 0 a 1, kde 0 znamená bílý prostor, 1 černý.

Na stejném principu fungují i dvourozměrné čárové kódy, známé jako QR[6].



Obrázek 12 MaxiCode (navržený pro americkou poštovní službu UPS)
zdroj http://www.carovy-kod.info/carovy-kod/2d-carove-kody_216.html

4.6.2 RFID

Radio frekvenční identifikace slouží k bezdrátovému čtení, zápisu a poskytování informací v reálném čase pomocí elektromagnetického vlnění.

[6] QR = Quick Response

Sestava se skládá z 1 aktivního prvku (RFID čtečka) a alespoň 1 pasivního prvku (RFID transpodér, nebo-li tag). RFID čtečka neustále vysílá elektromagnetické vlnění na určité frekvenci a pokud se tag dostane do určité blízkosti, toto vlnění je zachyceno a elektrická energie je přenesena bezdrátově z čtečky na tag, poté tag odpoví svým identifikátorem.

Díky různým frekvencím lze dosahovat jiných vlastností. Zatímco 125 KHz čtečky a tagy umožňují pouze čtení, 13.56 MHz umožňují i zápis.

Z RFID vychází technologie NFC[7], která bývá umístěna například na debetních kartách. Tato technologie pracuje na frekvenci 13.56 MHz. Výhodou NFC je vlastnost, která může udělat z čipu buď čtecí/zápisové zařízení nebo také zařízení, které se dokáže nechat číst.

Obrázek 13 Využití RFID na platebních terminálech
zdroj: http://www.atec.sk/rfid.php

4.7 Detekce překážek

Systém na detekci překážek počítá pozici překážky vzhledem k vysílači pomocí vzdálenosti. Vzdálenost může být zachycena sonarem (ultrazvuk) nebo systémem využívajícím světelné zdroje (infračervené světlo, laser - LiDAR). Tato podkapitola vychází ze zdroje [16], [28] a [29].

4.7.1 Ultrazvuk

Ultrazvukem lze měřit vzdálenost od vysílače k překážce za pomocí akustického vlnění. Zařízení se skládá z vysílače a přijímače vlnění. Vzdálenost od překážky se určí na základě času, který narůstá, dokud přijímač nezachytí vysílané vlnění.

4.7.2 Infračervené světlo

Měření vzdáleností lze i pomocí infračerveného světla. Skládá se z infračerveného vysílače a infračerveného přijímače. Princip spočívá ve vyslání infračerveného světla, které se odrazí od světlých ploch a infračervený přijímač toto světlo může zachytit, na rozdíl od ploch tmavých, kde se světlo pohltí a přijímač nic nezachytí. Princip se nejčastěji používá pro detekci překážek nebo sledování bílé/černé vodící čáry.

4.7.3 LiDAR

LiDAR[8] vypočítává vzdálenost také podle času. K tomu využívá laserový vysílač a přijímač. Signál se vysílá ve vlnové délce kolem 530 nm (zelená barva) a 1060 nm

[7] NFC = Near Field Communication
[8] LiDAR = Light Detection and Ranging

(téměř infračervené světlo). Tento systém využívají především letadla pro měření nadmořské výšky společně se systémem GPS.

4.8 Baterie

Návrh řešení zahrnuje také část, která má za cíl měřit napětí v baterii. Jednoduše to lze realizovat pomocí běžně dostupného voltmetru. Problém nastává v případě, když je baterie zabudovaná uvnitř robota a uživatel potřebuje, aby napětí dokázal změřit sám robot a podle hodnoty se určitým způsobem zachoval. Nejjednodušší způsob je pomocí odporového děliče napětí.

5 Praktická část

V následující kapitole se budu věnovat konkrétní realizaci celého systému. První část uvádí zvolené součástky pro konstrukci robota a důvody, pro jejich zvolení. Důležitou součástí bude také fyzické uspořádání a konstrukce dílů a součástek. V závěru bude popsán funkční algoritmus.

5.1 Metodika realizace

Na začátku zpracování práce bylo nutné vytvořit tři části systému, které jsem řešil nezávisle na sobě. První je samostatný pohyblivý robot, druhou částí je stanice na doplňování vody a třetí je dráha pro pohyb robota.

Návrh systému fungování byl první otázkou. Bylo nutné stanovit, jakým způsobem budou květiny zalévány. Řešením bylo upevnit nádobu na vodu na konstrukci robota, která bude sloužit jako zásobník vody, který je nutno doplňovat. Původně měl být robot sestrojen tak, aby mu vodu doplňoval člověk. Pro zvýšení samostatnosti robota jsem se však rozhodl vytvořit stanici, která bude doplňovat vodu do nádrže robota. Z tohoto konceptu jsem vycházel v průběhu celé realizace.

Před samotným sestavováním robota jsem si prostudoval, jaký mají vývoj a jaký základní hardware je k tomu potřeba. K pochopení mi pomohly základní tutoriály od Arduino a Raspberry Pi, které jsou uvedeny na webových stránkách. Také videa na webovém portálu YouTube nabízí mnoho tutoriálů od začátků, tzn. co je to Arduino, jak se s ním pracuje, jak se programuje a co je k programování potřeba. V oficiálním vývojovém prostředí Arduino IDE jsem využil několik zdrojových kódů, které mi pomohly s lepším pochopením programovacího jazyka. V tomto vývojovém prostředí jsem ze začátku realizoval zdrojový kód, nicméně po určité době jsem programoval v Microsoft Visual Studio s doplňkem pro Arduino.

Po prohloubení základních poznatků s Arduino jsem začal pořizovat první hardware, který sice přímo nesouvisel se samotným robotem, ale pomohl více porozumět platformě a ověřit si základní testovací zdrojové kódy uvedené v Arduino IDE. Oddělení robotiky mi v začátku vývoje vypůjčilo hardware, konkrétně Arduino Uno + motor-shield, dále Arduino Mega. Pomocí vyhledávače Google jsem si našel, jak tento motor-shield funguje, kam se zapojuje a jak se používá. Ještě než jsem se začal zabývat navigací, podle které bude robot jezdit, jsem si musel ujasnit alespoň přibližnou fyzickou konstrukci robota. Jak bude vypadat, jak bude velký, kolik kol bude mít.

5.2 Robot

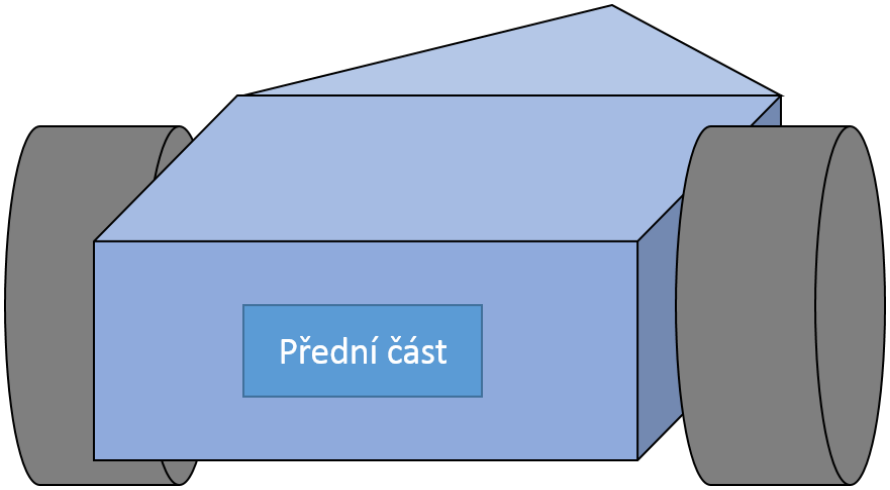
V této podkapitole chci popsat řešení robota a navigace, která je jeho nedílnou součástí. K fyzickému upevnění veškerých částí jsem využil stavebnici Merkur a samolepící elektrikářskou pásku z PVC materiálu, která sloužila také jako izolace elektrických obvodů.

5.2.1 Účel robota

Hlavním úkolem robota je zalít samostatně všechny květiny právě jednou a poté se vrátit na výchozí pozici.

5.2.2 Konstrukce a fyzický vzhled

Základem robota je jeho podvozek, neboli šasi, na kterém budu fyzicky umisťovat jednotlivé součástky a spojovací materiál. Oddělení robotiky mi za tímto účelem poskytlo k zapůjčení kovový podvozek. Ten byl zkonstruován se dvěma elektromotory s připevněnými koly a pomocným třetím kolem, které se mohlo otočit o 360 stupňů. Tento typ podvozku se nejčastěji využívá pro první prototypy. Maximální rychlost elektromotorů dosahuje u tohoto podvozku 64 otáček za minutu. Ačkoliv to může vypadat jako nízká hodnota, pro účely sestavení tohoto robota plně postačuje.



Obrázek 14 Ilustrace zapůjčeného modelu

5.2.3 Vývojová platforma

Při výběru platformy jsem se soustředil na jednoduchost a uživatelské prostředí, částečně i rozšiřitelnost a finanční náročnost. V tomto ohledu jsem si zvolil platformu Arduino. Výrobce Arduino nabízí ve svém portfoliu velké množství platforem na základě určení, parametrů a ceny.

Mezi nimi je platforma Arduino Mega 2560, která byla vybrána jako řešení autonomního robota z hlediska počtu vstupně-výstupních pinů a velikosti vnitřní paměti. Dalším kandidátem z portfolia byl Arduino Due, nicméně model nevyhovoval v technických parametrech, neboť umožňoval připojit pouze součástky s napětím do 3.3V, což se v průběhu řešení ukázalo jako nedostačující. Původně jsem se domníval, že pro realizaci postačí model Uno, nicméně byl nedostačující z hlediska počtu pinů a velikosti paměti.

Jelikož je Arduino open-source projekt, můžeme se setkat s klony, deriváty případně padělky.

Klon je totožný výrobek, má pouze jiné logo výrobce.

Derivát (odvozenina) je výrobek vycházející z originálu, nicméně obsahuje jiné periferie nebo komponenty. Konstrukční zpracování je téměř odlišné.

Padělkem se rozumí napodobeninou původního výrobku, který nese logo Arduino a má stejné technické parametry, rozdíly mohou být v jiné grafické úpravě a ceně.

Finanční náklady:

- Originál Arduino Mega 2560: 45 USD (prodejce: Adafruit, 3/2016)

- Klon Arduino Mega 2560: nejlevnější již od 9 USD
- Originál Arduino Uno: 25 USD (prodejce: Adafruit, 3/2016)
- Klon Arduino Uno: nejlevnější již od 5 USD

K realizování bylo využito nepájivé pole, které sloužilo ke spojení součástek.

5.2.4 Pohyb robota

Tato podkapitola vysvětluje výběr součástek pro pohyb robota a řízení. Podkapitola vychází ze zdrojů [30], [17], [22] a [35].

K ovládání pohybu kol je použit standardní stejnosměrný elektromotor. Jeho účelem je převést elektrickou energii na mechanickou práci, konkrétně na rotační pohyb. Využívá silových účinků magnetického pole.

Různé druhy se liší velikostí, maximální rychlostí otáček za minutu a celkovou odolností vůči vnějším vlivům a výkonem. Pro návrh robota jsem použil 12V elektromotory.

Obrázek 15 Příklad stejnosměrného elektromotoru
zdroj: http://rarecomponents.com/store/Johnson%20Electric

Finanční náklady:

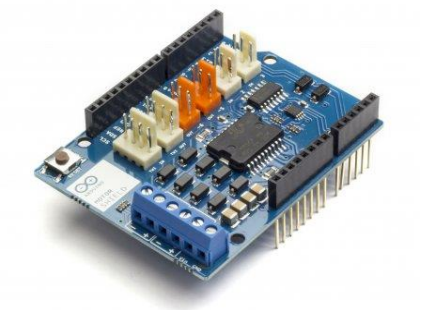
- Ceny jsou nejrůznější podle velikost a výkonu. Pro malé roboty, kteří nepotřebují vysoký výkon, lze sehnat 1 pár elektromotorů s koly již od 9 USD.

H-můstek (h-bridge) je elektrický obvod, bez kterého by nebylo možné bezpečně dodat elektrickou energii k stejnosměrným elektromotorům tak, aby nedošlo k nenávratnému poškození vývojové desky. Tento obvod umožňuje proudění napětí a proudu do motorů skrz tranzistory. Nejčastěji se tento typ obvodu používá pro ovládání elektromotorů.

Samotné ovládání elektromotorů je realizováno pulzně šířkovou modulací (PWM), která umožňuje analogový signál v rozsahu 0-255 transformovat do digitální podoby. "*Modulovaný signál nese informaci pomocí poměru doby trvání jedničky a nuly signálu v jednotlivých intervalech.*" [22]

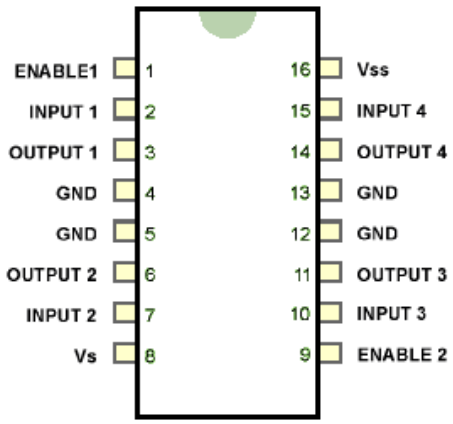
Na trhu lze sehnat několik těchto h-můstků. Rozlišuje se podle maximálního dovoleného počtu zapojených motorů a podle možnosti zapojení. Například motor-

shield lze jednoduše nainstalovat do pinů modelu Arduino Uno, kdežto H-můstek typ L298N (případě L293D) se musí spojit dráty. Rozdíl je také v programové části, kdy motor-shield má většinou striktně nastaveny piny pro funkčnost, u H-můstku toto omezení odpadá.



Obrázek 16 Příklad motor-shieldu Arduino
zdroj: https://store.arduino.cc/product/A000079

Rozhodl jsem se využít h-můstek L293D (čip). Důvod, proč jsem si nezvolil L298N byl takový, že ve fázi testování se projevil jako nespolehlivý a po chvilkové zátěži přestal spolehlivě fungovat. Cena čipu L293D byla činila pouze 0,7 USD. Instalace byla provedena do nepájivého pole."



Obrázek 17 Rozložení portů čipu L293D
zdroj: http://www.gadgetronicx.com/bidirectional-motor-controller-circuit-l293d/

Zapojení L293D čipu

- ENABLE 1 → digitální pin 3 (PWM) - slouží k ovládání rychlosti motoru 1
- INPUT 1 → digitální pin 24 - slouží k nastavení směru motoru 1
- INPUT 2 → digitální pin 25 - slouží k nastavení směru motoru 1
- OUTPUT 1 + OUTPUT 2 → slouží k zapojení vodičů k motoru 1
- ENABLE 2 → digitální pin 4 (PWM) - slouží k ovládání rychlosti motoru 2
- INPUT 3 → digitální pin 26 - slouží k nastavení směru motoru 2
- INPUT 4 → digitální pin 27 - slouží k nastavení směru motoru 2
- OUTPUT 3 + OUTPUT 4 → slouží k zapojení vodičů k motoru 2

- Vs → zde připojit pozitivní pól externího zdroje elektrické energie 12V, tímto pinem se napájí připojené motory
- Vss → zde připojit pozitivní pól (5V) z Arduino, slouží k napájení samotného čipu
- GND → uzemnění, všechna uzemnění vedou do Arduino

Pro správnou funkčnost je nutné hodnotu INPUT1/INPUT2 nenastavovat na stejnou hodnotu (LOW, HIGH), ale na hodnotu rozdílnou, jinak se připojený motor zastaví.

Finanční náklady:

- 1ks L293D: ~0,7 USD

Vzorový program, ze kterého jsem vycházel, je přiložen jako příloha č. 1.

5.2.5 Řízení pohybu - PID

Pro plynulé řízení pohybu jsem se rozhodl, že využiji princip PID regulátoru, který byl popsán v teoretické části. Implementace úzce souvisí s tvarem robota, pozicí kol, poloměrem kol, poloměrem otáčení, výkonem elektromotorů a hlavně s fyzickým umístěním navigačních sensorů - konkrétně infračervených sensorů. Právě infračervené sensory mají funkci vstupu do PID regulátoru.

Konfigurování jednotlivých složek PID regulátoru je věcí časově náročnou. Postupně se musí upravovat jednotlivé složky do té doby, než se robot začne pohybovat a reagovat tak, jak chceme. Implementace PID je možná několika způsoby. Například výrobce Pololu vytvořil ke svému čidlu QTR-8RC knihovnu, kterou lze využít pro Arduino platformu, případně si lze celý regulátor naprogramovat samostatně. Já jsem využil uživatelský kód kvůli přehlednosti bez nutnosti importovat do programu dodatečný software (knihovnu).

Implementace PID regulátoru má těsnou souvislost s čidly pro navigaci, které jsou popsány níže v podkapitole Navigace. PID regulátor využívá hodnoty z infračervených čidel. Pro testování jsem využil pouze složku proporcionální a derivační. Integrační složku jsem vynechal, neboť jsem ji nepotřeboval a při testování působila značné komplikace.

Koeficienty složek jsem nastavil na následující hodnoty:

- Kp = 12,
- Ki = 0,
- Kd = 1, při nastavení analogové hodnoty 170 (rychlost motorů).

Vzorový program k PID, ze kterého jsem vycházel, je přiložen jako příloha č. 2.

5.2.6 Navigace

V následující podkapitole představím model navigace. Ten je využíván robotem společně s dráhou. Podkapitola čerpá ze zdrojů [31] a [20].

Aby byl robot schopen dojet na určené místo, je důležité, aby mohl na místo určení dojet podle navigace. První myšlenkou byla navigace pomocí GPS. Po analýze vyšlo najevo, že by toto řešení bylo nejen finančně náročné, ale také přesnost běžného GPS lokalizátoru by se pohybovala mezi 3-15 metry, což se ukázalo jako značný problém, jelikož bylo třeba mít přesnost maximálně 15 cm.

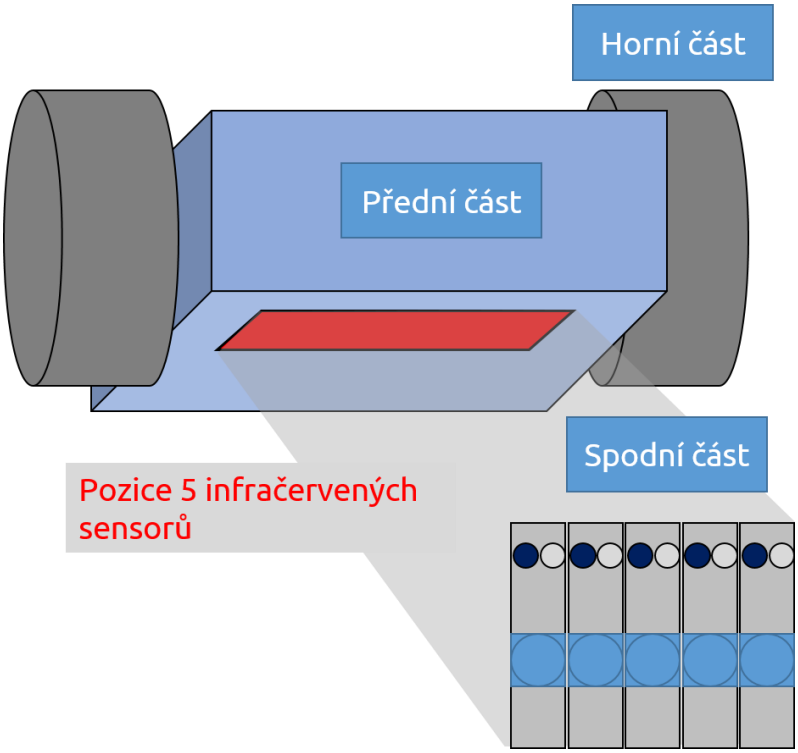
Další možností bylo řešení pomocí principu triangulace se 3 a více anténami - tedy zkonstruovat vysílače v testovacím terénu a lokalizovat svojí polohu pomocí síly signálu od těchto vysílačů. Toto řešení se ukázalo nevhodné z hlediska ceny a také z hlediska praktického nasazení.

Další z možností bylo využití infračerveného vlnění. Myšlenka spočívala v hlavní stanici, která by byla umístěna uprostřed oblasti a měla do všech stran nasměrované infračervené přijímače. Květiny by byly v tomto případě vybaveny infračervenými vysílači. Jakmile by květina potřebovala vodu, zasvítila by infračerveně na určité frekvenci, stanice by rozpoznala, jaké vlnění přijímá a kterým směrem a robotovi by předala informaci, kudy má jet ke květině. Robot by také měl na sobě přijímač, aby mohl udržovat směr emitujícího infračerveného vlnění. Tento nápad jsem posléze zavrhl kvůli ceně, nutnosti dodatečnému nákupu dalších součástek. Při slunečním svitu by také mohlo docházet k chybám, nehledě pak na fyzické natažení kabelů ke květinám a vysílačům infračerveného vlnění, případně nainstalování samostatných řídicích jednotek.

Nakonec jsem se zaměřil na běžně dostupná a levná řešení navigace. Po hledání na vyhledávači Google jsem zvolil řešení pomocí vodící čáry společně se sensory, které slouží robotovi jako jakési smysly, podle kterých vidí/nevidí černou čáru. Finanční zátěž byla v řádech desetikorun a navíc tento způsob lze snadno implementovat a nemůže se stát, že by robot jel jiným směrem než tím, který je mu vymezen pomocí černé pásky. Pro jasné umístění robota na černé pásce je lepší využít vysoký kontrast, konkrétně použit černé pásky uprostřed a bílého podkladu o šířce 15 cm z obou stran. Tím se vyvarujeme případné chybovosti v odrazech. K rozlišování a udržování směru je zapotřebí implementovat alespoň jeden pár infračervených sensorů. Na realizaci jsem jich použil právě 5, a to z důvodu lepší reakce na změny ve směru, které jsem umístil na spodní část povozku tak, aby čidla směřovala směrem k zemi. Umístění je vyobrazeno na obrázku 18 *Pozice infračervených sensorů*.

Infračervený sensor má za cíl vysílat a zachycovat odrazy. Skládá se z infračerveného vysílače a infračerveného přijímače. Princip spočívá ve vyslání infračerveného světla, které se odrazí od světlých ploch a infračervený přijímač toto vlnění může zachytit, na rozdíl od ploch tmavých, kde je světlo pohlceno a přijímač nic nezachytí. V obchodech lze najít několik druhů podle počtu těchto senzorů na jedné destičce od 1 páru (např. KY-033) až po několik párů senzorů (např. Pololu QTR-8RC). Běžnou součástí obvodu bývá i potenciometr, kterým lze ovlivňovat vzdálenost, na kterou čidlo bude nebo nebude reagovat. V mém případě jsem hodnotu nastavil na poloviční u všech 5 sensorů.

K fyzické konstrukci jsem využil díl ze stavebnice Merkur a 5 infračervených sensorů KY-033, které jsem umístil vedle sebe a přišrouboval k dílu. Ten jsem následně pomocí pásky přilepil zespod na povozek robota.



Obrázek 18 Pozice infračervených sensorů

Obrázek 19 Infračervený sensor KY-033
zdroj: http://www.banggood.com/KY033-Tracing-Black-White-Line-Hunting-Sensor-Module-For-Arduino-p-91854.html

Zapojení

- V+(+)/G(-) → připojení napájení (5V) a uzemnění
- S → analogové piny A1 - A5

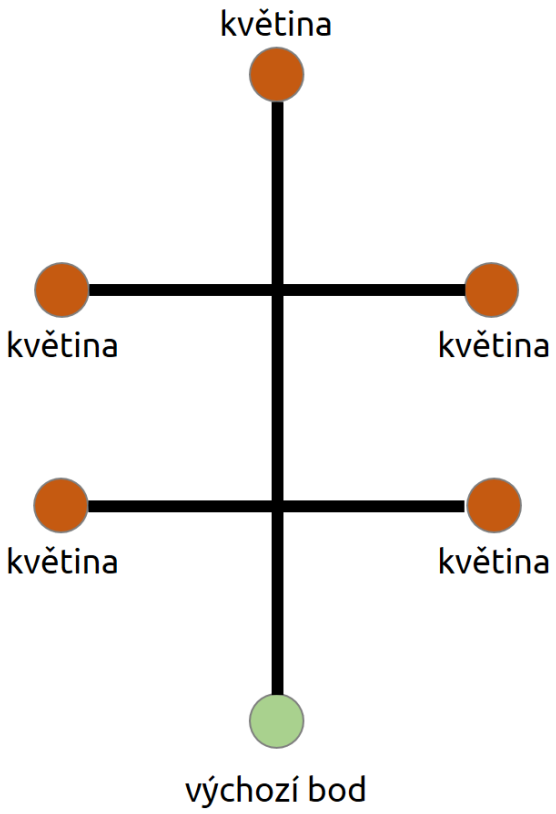
Finanční náklady:

- 1ks KY-033: ~ 0,8 USD

Vzorový program, ze kterého jsem vycházel, je přiložen jako příloha č. 3.

Ještě předtím, než jsem začal zkoušet navigaci pomocí černé pásky, jsem si musel určit trasu, kterou bude robot využívat. K tomu sloužil návrh testovací mapy (viz obrázek 20 *Testovací mapa*), ze které jsem vycházel po celou dobu zpracovávání práce.

Algoritmus, který popíši v závěru kapitoly, vychází právě z této navržené trasy, včetně 90 stupňových úhlů mezi trasami u křižovatek.



Obrázek 20 Testovací mapa

Z obrázku lze více či méně vypozorovat problém s jasnou navigací robota, tedy jak a podle čeho bude moci určit, kde se zhruba nachází. Původně jsem zamýšlel dát ke květinám jednoznačný identifikátor, nejlépe bez nutnosti elektrické energie.

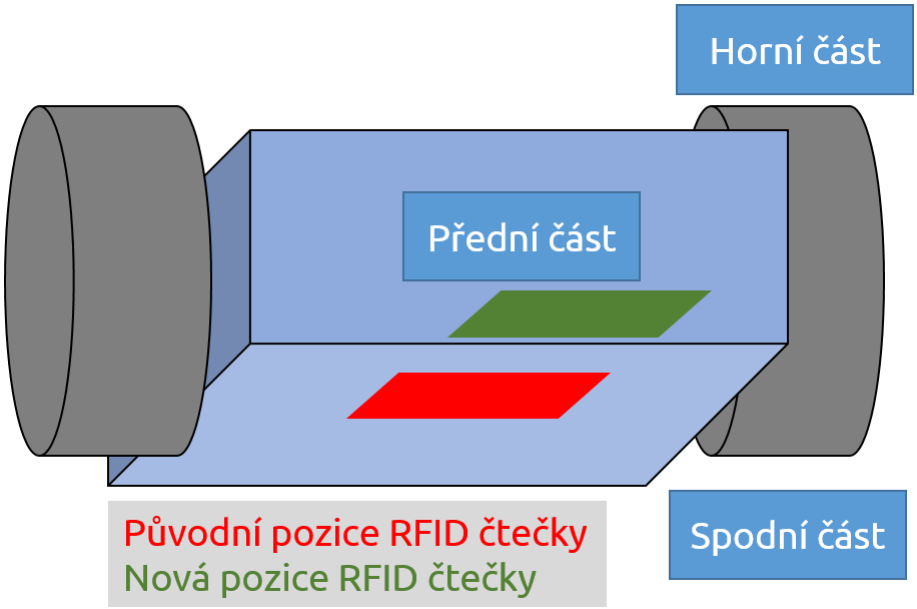
Jednorozměrný nebo dvourozměrný čárový kód byla první varianta. Ovšem, jak se později ukázalo, čtečka těchto kódů by byla řešením finančně nákladným. Navíc fyzické umístění takovéto čtečky by bylo, z mého pohledu, věcí nepraktickou a složitě by se na podvozek konstruovala. S tím souvisí i možné čištění kódů u květin, které by se mohly vlivem fyzikálních jevů znehodnotit nebo poškodit tak, že by nebylo možné jednoznačně přečíst čárový kód.

Po určité době hledání řešení problému na internetu jsem objevil potenciál v technologii RFID. Jak již bylo popsáno v teoretické části, tento systém se skládá ze čtečky a samotného čipu. Právě čip je většinou zapuštěn do celoplastového obalu, který chrání cívku před působením větru, vody, prachu, špíny a prachu, což se jevilo jako velká výhoda. Také finanční nákladnost čipů (RFID tagů) byla zanedbatelná - v řádu korun, čtečka se pohybovala v řádech desetikorun. Důležitým krokem výzkumu byla i vlastnost čteček a čipů v různých frekvencích. Díky různým frekvencím lze dosahovat jiných vlastností. Zatímco 125 KHz čtečky (např. RDM630) a tagy umožňují pouze čtení, 13.56 MHz (např. MFRC522) umožňují i zápis. NFC technologii jsem zamítl hlavně kvůli složitější implementaci než u RFID a také kvůli ceně, která byla vyšší než vybraná technologie.

Obrázek 21 RFID čtečka RDM630 s anténou
zdroj: http://www.ebay.com/itm/125-KHZ-EM4100-RFID-Card-Read-Module-RDM630-UART-compatible-Arduino-/370668442091

Obrázek 22 RFID Tag - klíčenka
zdroj: http://www.tme.eu/cz/details/mikroe-1475/moduly-rfid/mikroelektronika/rfid-tag-1356mhz-iso14443-a-standard/

RFID na frekvenci 125 kHz byla pro moje řešení dostačující z důvodu čtecí vzdálenosti, která se pohybuje v řádech centimetrů, ceně pořízení a nenutnosti zapisovat na RFID tagy jakákoliv data. Při testování přímo na robotovi se objevil konstrukční problém, který způsoboval nefunkčnost RFID antény u čtečky. To bylo způsobeno nesprávným fyzickým umístěním přímo na kovový podvozek. Řešením bylo posunout anténu z dosahu této kovové desky před robota (viz obrázek 23 *Změna pozice RFID čtečky*).

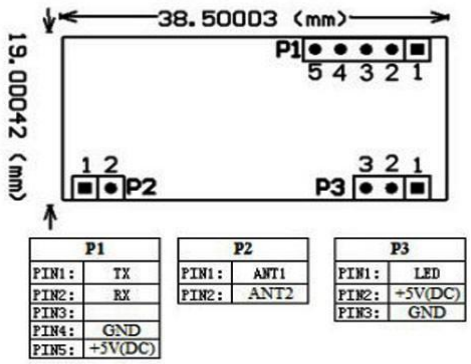


Obrázek 23 Změna pozice RFID čtečky

RFID anténu jsem upevnil na robota pomocí plastové silnější fólie a RFID tagy jsem umístil ke každé květině. Po standardizaci a představě algoritmu jsem se rozhodl, že RFID tag nainstaluji ke každému kritickému bodu. Kritickým bodem se rozumí jakýkoliv bod, kde robot nejede pouze jedním směrem, ale dochází zde k nějaké určité akci - například zalévání, rozhodování kudy jet a kudy ne. Kritickým bodem je stanice, květina a křižovatka. Dle obrázku 20 *Testovací mapa* jsem použil celkem 8 RFID tagů, které jsem přelepil černou vodící páskou. To z důvodu, aby infračervené sensory stále mohli "vidět" trasu. Odlišností byla konstrukce RFID tagů u květin a stanice. U květin jsem implementoval tyto tagy před samotné květiny tak, aby měl robot před sebou nějaký prostor. U stanice tento problém nehrozí, neboť bude stanice zkonstruována tak, aby před sebou robot nebyl jakkoliv stanicí omezován ve svém pohybu.

Při testování tohoto návrhu společně s pohybem robota se objevil problém, který zapříčinil neschopnost čtečky přečíst RFID tag umístěný na dráze kvůli rychlosti robota. Ačkoliv jsem rychlost snížil z původních 200 na 100 (PWM), stále to bylo příliš rychlé na sejmutí RFID signálu. Problém byl vyřešen za pomoci přerušení černé pásky před předpokládaným tagem, rozšíření černé pásky na 2 cm kvůli infračerveným sensorům a úpravou kódu, která způsobovala zastavení robota, jakmile infračervená čidla ztratí černou vodící čáru z dosahu. Délka přerušení, tedy odstranění černé pásky před předpokládaným tagem, je 5 cm. To umožnilo robotovi na konci černé pásky zastavit, a jakmile stál, byl schopen přečíst RFID tag a následně se rozhodnout, co udělat. Délka 5 cm je vzdálenost od infračervených snímačů k nainstalované RFID anténě.

Zapojení



Obrázek 24 Význam pinů RFID čtečky RDM630
zdroj: http://arduino8.webnode.cz/news/lekce-33-arduino-a-modul-ctecky-rfid/

- P1-PIN5(+)/P1-PIN4(-) → připojení napájení (5V) a uzemnění
- P1/1 (TX) → digitální pin 50, slouží k odesílání dat z čipu do řídicí jednotky
- P2-PIN1 (ANT1) + P2-PIN2 (ANT2) → slouží k připojení samotné antény

Finanční náklady:

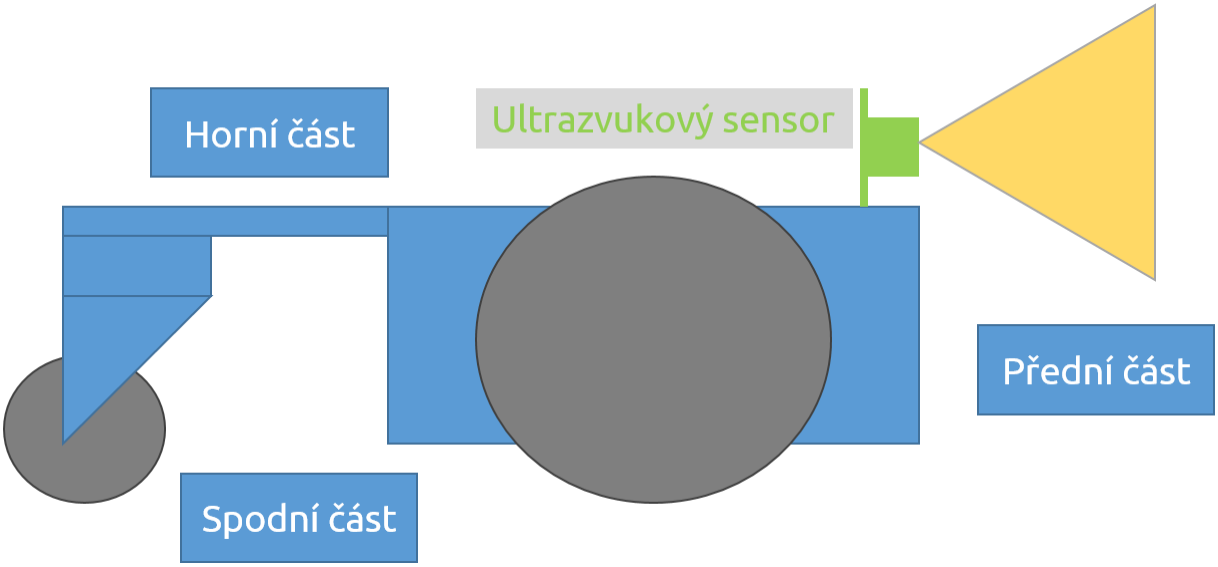
- RDM630 s anténou: ~ 2,5 USD
- 1 ks RFID tag 125 kHz: ~ 0,2 USD

Vzorový program, ze kterého jsem vycházel, je přiložen jako příloha č. 4.

5.2.7 Detekování překážek

Kromě infračervených čidel na detekci překážek se dále využívá technologie ultrazvuku. Zařízení se skládá z vysílače a přijímače ultrazvuku. Tato podkapitola vychází ze zdroje [32].

Nejdostupnějším čidlem je HC-SR04, které dokáže zpracovat objekty v různých vzdálenostech a má oddělený vysílač a přijímač. Rozhodujícím faktorem oproti jiným řešením byla cena a jednoduchost programového nasazení. Ultrazvukový sensor jsem umístil na přední část robota s určitým odstupem.



Obrázek 25 Pozice ultrazvukového sensoru

Obrázek 26 Ultrazvuk HC-SR04
zdroj: http://robu.in/product/hc-sr04-ultrasonic-range-finder/

Zapojení

- VCC(+)/GND(-) → připojení napájení (5V) a uzemnění
- Trig → digitální pin 34
- Echo → digitální pin 35

Finanční náklady:

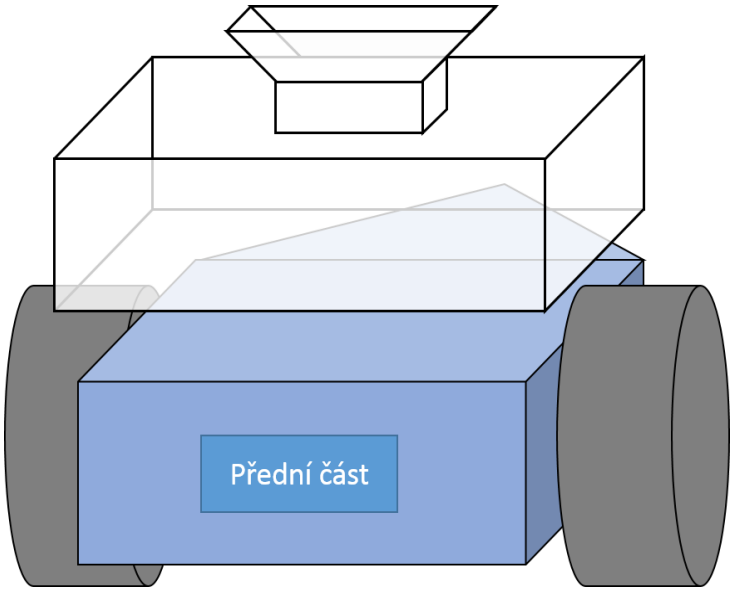
- 1 ks HC-SR04: ~1 USD

Vzorový program, ze kterého jsem vycházel, je přiložen jako příloha č. 5.

5.2.8 Řízení vody

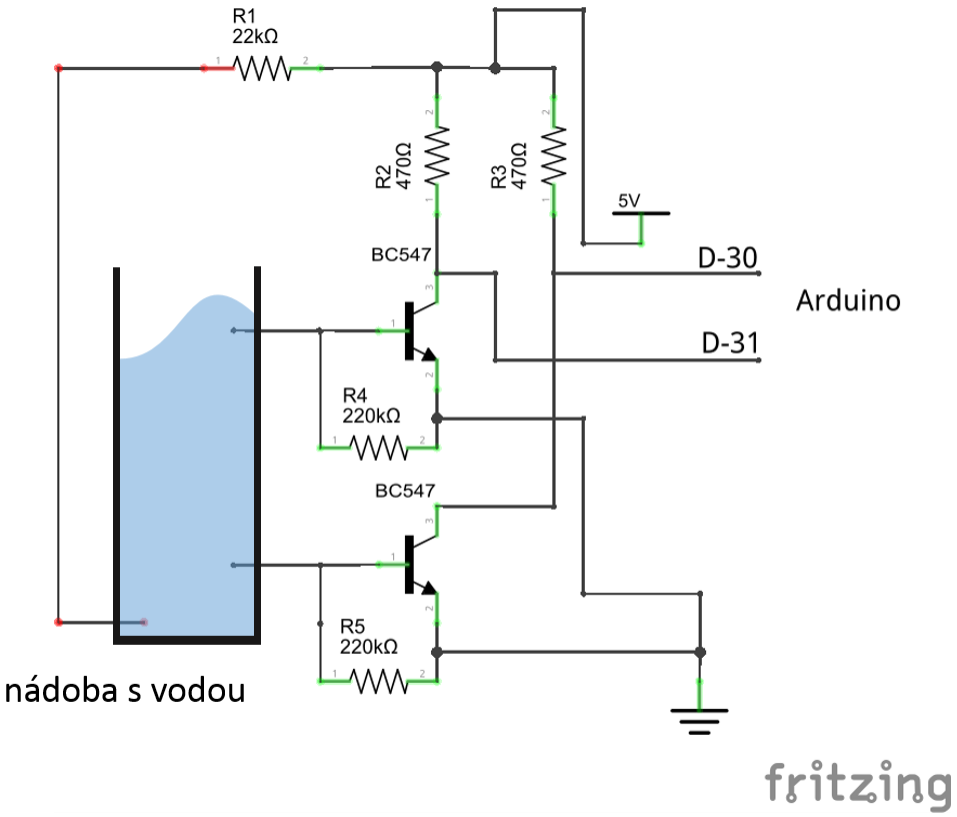
Řízení vody se skládá ze dvou hlavních částí. První z nich je samotný zásobník vody, který je napevno uchycen na kostře robota. Zásobník je kvádrovitého tvaru o vnitřních rozměrech 160 x 120 x 50 mm s otevřeným víkem o výšce 45 mm ve tvaru trychtýře a je vyroben z plexiskla. Po testovacím naplnění se do zásobníku vejde přibližně 900 ml tekutiny. Uchycen je na horní části robota.

Otevřený trychtýřovitý otvor je z důvodu plnění kapaliny. Tato podkapitola využívá zdroj [33].



Obrázek 27 Pozice zásobníku na vodu

Součástí zásobníku je i měřič hladiny. Úlohou elektrického obvodu je určit, kdy je kapaliny v nádobě málo a zároveň, kdy je kapaliny v nádobě dostatek. Ke splnění jsem využil několik transistorů, podle jejichž spínání lze určit, kde proudí elektrická energie, a tedy i úroveň, ve které se voda nachází. Vodičem je, kromě metalických drátů, voda. Zvolený obvod je připojený na několik vstupních pinů, přičemž pokud tekutina propojí obvod, na konkrétním pinu lze číst hodnotu '0'. Pokud tekutina nepropojí obvod, na konkrétním pinu lze číst hodnotu '1'.



Obrázek 28 Schéma a zapojení obvodu na měření kapaliny
zdroj: http://www.electroschematics.com/9964/arduino-water-level-indicator-controller/

Zapojení

- 5V/GND → připojení napájení (5V) a uzemnění
- D-30 → digitální pin 30 (1. úroveň)
- D-31 → digitální pin 31 (2. úroveň)

Finanční náklady:

- 1 ks odpor 22K ohm: pod 0,1 USD
- 2 ks odpor 470 ohm: pod 0,1 USD
- 2 ks odpor 220K ohm: pod 0,1 USD
- 2 ks NPN transistor BC547: pod 0,1 USD

Obvod jsem realizoval pomocí vytvořeného plošného obvodu za použití kuprexitu, leptací kyseliny a lihového fixu.

Vzorový program, ze kterého jsem vycházel, je přiložen jako příloha č. 6.

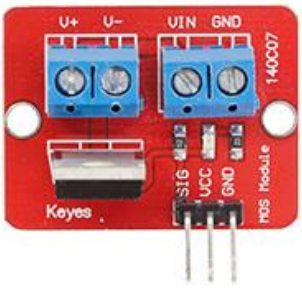
Pro přesun kapaliny z nádoby ke květině přes silikonové trubičky jsem využil zubové čerpadlo. Zubová část je poháněná elektromotorem. Celá tato součástka vypadá jako standardní elektromotor popsaný v teoretické části, nicméně dodává se s formou, která obsahuje zmíněná ozubená kola a komoru, kudy teče tekutina.

Obrázek 29 Čerpadlo s elektromotorem RS-360SH
zdroj: http://www.banggood.com/Mini-Micro-DC-9V-RS-360SH-Water-Priming-Pump-p-87577.html

Na realizaci jsem instaloval čerpadlo s elektromotorem RS-360SH, které bylo nejdostupnějším, finančně nenáročným řešením, které se v průběhu testování ukázalo jako dostačující.

V souvislosti s čerpadlem je nutné využít transistor (mosfet) podobný h-můstku s rozdílem v maximálním dovoleným počtem připojených elektromotorů. Jedním z mnoha transistorů je IRF520 nebo 30N06L.

Na trhu se nabízí buď samotný transistor anebo model na destičce (např. Keyes MOS module IRF520). Kvůli lepší spolehlivosti jsem vyměnil na destičce (viz Obrázek 14) transistor IRF520 za 30N06L. Zapojení a funkčnost je pro oba případy totožná.



Obrázek 30 Transistor 30N06L na destičce
zdroj: http://www.spikenzielabs.com/Catalog/index.php?main_page=product_info&products_id=1152

Zapojení elektromotoru a Keyes MOS module s 30N06L

- V+/V- → slouží k připojení vodičů elektromotoru
- VIN(+)/GND(-) → připojení napájení (12V) a uzemnění k externímu zdroji
- VCC(+)/GND(-) → připojení napájení (5V) a uzemnění k Arduino
- SIG → digitální pin 6 (PWM)

Finanční náklady:

- Čerpadlo RS-360SH: ~4 USD
- Keyes MOS module IRF520: ~1 USD
- Transistor 30N06L: ~0,4 USD

Vzorový program, ze kterého jsem vycházel, je přiložen jako příloha č. 7.

Zalévací trubičku jsem umístil nad ultrazvuk s přesahem 7 cm od kostry. Po otestování dokázalo toto čerpadlo na poloviční výkon odčerpat přibližně 18 ml za 1 sekundu. Měření sloužilo k získání informace o průtoku a následnou implementací do logiky zalévání.

Logika zalévání spočívala v kontrole hladiny vody vždy po 50 ml. V případě, že květina potřebuje více vody, než má robot v zásobníku, zbývající hodnotu k zalití snižuje po každém úspěšném cyklu. Tímto způsobem je ošetřené, že by květina byla zalita například pouze 100 ml vody místo původních 300 ml, jakmile by robot neměl dostatek vody.

5.2.9 Komunikace

Jelikož robot bude mít k dispozici stanici na doplňování vody, bylo potřeba vyřešit problém komunikace. Komunikační způsob jsem realizoval pomocí infračerveného vlnění, které je jednoduché na implementaci a zároveň je finančně výhodné. Tato podkapitola využívá zdroj [34].

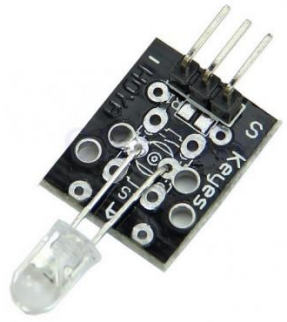
Je možné si pořídit samotnou infračerveně emitující diodu (např. IR LED 950nm) nebo již hotovou destičku s touto diodou (např. KEYES Infrared transmitter module).

Infračervený přijímač lze také pořídit v minimálně dvou variantách, tj. základní součástka (např. TL1838 VS1838B) nebo již hotový elektrický obvod na destičce (např. Keyes TSOP1838).

Správnost konfigurace závisí na vysílání a přijímání signálu ve stejných frekvencích (např. 38 kHz).

K jednoduchému ovládání byla nutná knihovna '`IRremote.h`', která usnadňuje poslání a přijímání kódů pomocí nejrůznějších protokolů (např. Sony, JVC, Samsung a další), které najdou uplatnění například v ovládání televizí. Součástí této knihovny je seznam nejrůznějších platformem, jejichž připojení z hlediska pinů, se liší.

Fyzicky byl vysílač upevněn na přední část zásobníku s vodou, natočení vysílače bylo směrem nahoru.



Obrázek 31 Keyes infračervený vysílač na destičce (IR Led)
zdroj: http://en.keyes-robot.com/productshow.aspx?id=202

Obrázek 32 Keyes module TSOP1838 (infračervený přijímač)
zdroj: http://en.keyes-robot.com/productshow.aspx?id=203

Vysílání bylo realizováno pomocí samostatné emitující led diody a na příjem jsem použil infračervený přijímač na destičce. Zapojení infračerveného přijímače budou dále popsáno v podkapitole Stanice.

Zapojení

- Delší vodič (+) + odpor (220 ohm) → digitální pin 9 (platí striktně pouze pro Arduino Mega)
- Kratší vodič (-) → uzemnění

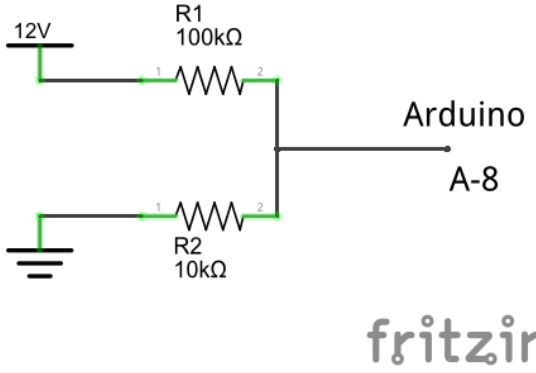
Finanční náklady:

- 1 ks infračervená led dioda: pod 0,1 USD
- 1 ks odpor 220 ohm: pod 0,1 USD

Vzorový program, ze kterého jsem vycházel, je přiložen jako příloha č. 8.

5.2.10 Měření baterie

Způsob měření napětí za pomocí napěťového děliče se zdála z hlediska sestrojení a implementace jednoduchá. Jeho cena je zároveň zanedbatelná. Tímto obvodem lze měřit napětí až do 55V. Oddělení robotiky mi v rámci práce zapůjčilo baterii Zippy 2800 30C, na které zároveň jsem testoval realizaci robota.



Obrázek 33 Schéma a zapojení měřiče napětí pomocí odporového děliče
zdroj: http://www.electroschematics.com/9351/arduino-digital-voltmeter/

Zapojení

- Spoj mezi odpory → analogový pin 8

Finanční náklady:

- 1 ks odpor 100k ohm: pod 0,1 USD
- 1 ks odpor 10k ohm: pod 0,1 USD

Vzorový program, ze kterého jsem vycházel, je přiložen jako příloha č. 9.

Hodnota, pod kterou nesmí klesnout napětí je zhruba 3,5 V na článek.

5.2.11 Dodatečný hardware

Pro zjištění chyb byly implementovány 2 LED diody a 1 bzučák. Zelená LED dioda signalizuje splnění úkolu = všechny květiny zalité. Červená LED dioda signalizuje nedostatek baterie. Bzučák funguje pro případ detekce překážky ultrazvukovým sensorem a pro případné zjištění nedostatku baterie, pokud se nachází ve stanici.

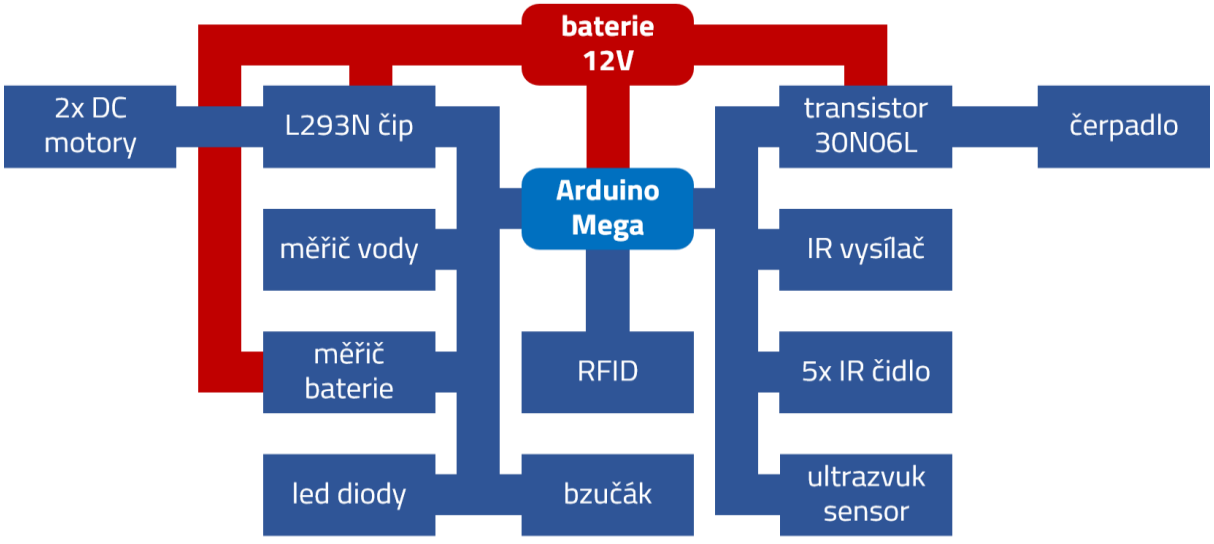
Zapojení

- Zelená LED dioda → delší vodič je nutné spojit s odporem 220 ohm, poté připojit na digitální pin 40
- Červená LED dioda → delší vodič je nutné spojit s odporem 220 ohm, poté připojit na digitální pin 41
- Bzučák (BPT 14X) → delší vodič připojit na pin 42
- Kratší vodič obou diod i bzučáku slouží k uzemnění k Arduino

Finanční náklady:

- 2x LED dioda: pod 0,1 USD
- 2x odpor 220 ohm: pod 0,1 USD
- 1x bzučák: ~0,5 USD

5.2.12 Zjednodušené schéma zapojení komponent



Obrázek 34 Zjednodušené schéma zapojení robota

5.3 Stanice

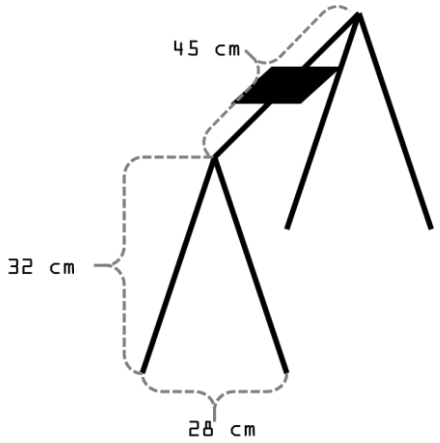
K fyzickému upevnění stanice na doplňování vody jsem využil stejný materiál jako u robota, tedy díly ze stavebnice Merkur a PVC pásku pro dodatečné uchycení.

5.3.1 Účel stanice

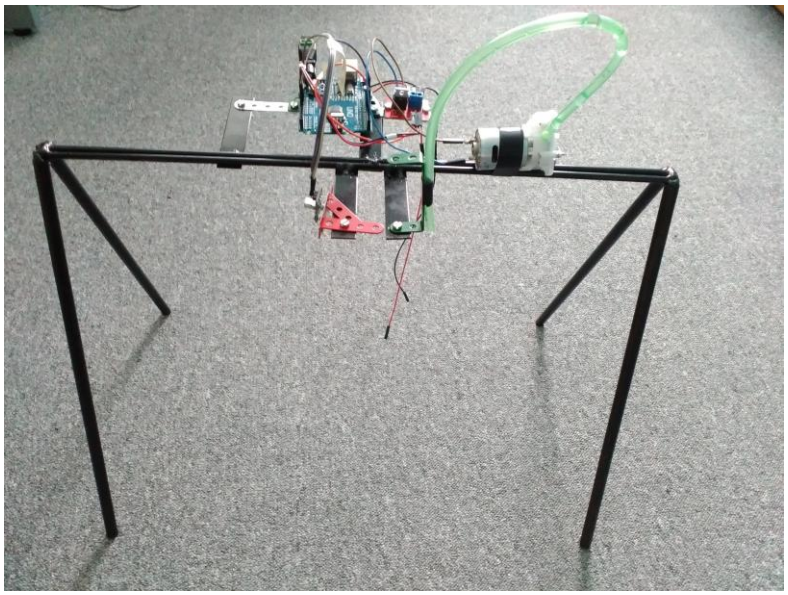
Účelem stanice je doplňovat vodu na zalévání do zásobníku umístěného na kostře robota. Stanice slouží také jako výchozí bod robota. Výchozím bodem se rozumí bod, ve kterém robot skončí, jakmile všechny květiny zalije dostatečným množstvím vody.

5.3.2 Fyzická kostra

Stanice byla navržena jako rám (železná konstrukce), do kterého se robot vejde se svými rozměry. Části byly svařeny k sobě a vytvořily kostru stanice. Na horní část spoje byly následně přivařeny malé plechové destičky, ke kterým jsem připevnil vývojovou platformu a další komponenty pomocí dílů ze stavebnice Merkur.



Obrázek 35 Vyrobená kostra stanice s rozměry



Obrázek 36 Realizovaná stanice

5.3.3 Vývojová platforma

Nejvhodnější z hlediska logiky systému a fungování bylo užití samostatné řídicí jednotky. V průběhu testování a zkoušení součástek robota byla pořízena platforma Funduino Uno, která je klonem Arduino Uno. Jednoznačnou výhodou byla cena, která byla 5 USD. Z hlediska pinů a velikosti paměti se jednalo o plně dostačující kontrolér, který může být použit i pro případné rozšíření.

K napájení celé stanice jsem využil adaptér snižující napětí z 230V na 12V, neboť není potřeba se stanicí v průběhu zalévání pohybovat a může využít stálý zdroj elektrické energie skrz elektrickou síť.

5.3.4 Doplňování vody

Představa vycházela z možnosti mít poblíž stanice velký zásobník vody, kterou člověk bude doplňovat. Z tohoto zásobníku by již stanice mohla pomocí zubového čerpadla čerpat vodu a plnit zásobník robota. Čerpadlo je totožné s čerpadlem nainstalovaným na robotovi včetně transistoru. Ovládání je realizováno pomocí PWM s rozdílem zapojení, které je na digitálním pinu 6.

5.3.5 Komunikace

Jak jsem naznačil v části realizace robota, je nutné implementovat komunikaci mezi stanicí a robotem. Smysl komunikace spočívá v přijetí povelu od robota ke stanici. Na základě čidla hladiny zásobníku robota bude požadovat po stanici, aby začala pumpovat vodu do zásobníku robota. Pro tento případ naprosto postačuje simplex komunikace. Přijímač infračerveného vlnění byl zvolen Keyes TSOP1838, který po otestování fungoval společně s emitující LED diodou umístěnou na kostře robota. Mimo ceny shledávám výhodu ve smyslu úhlu zachycení infračerveného vlnění. Na základě několika provedených testů přijímač zachycoval signál i v případě, že byl vysílač velmi nakloněn a nesměřoval přímo do přijímače. Fyzické umístění bylo přibližně vprostřed stropu hranice, přijímač směřoval směrem dolů.

Zapojení

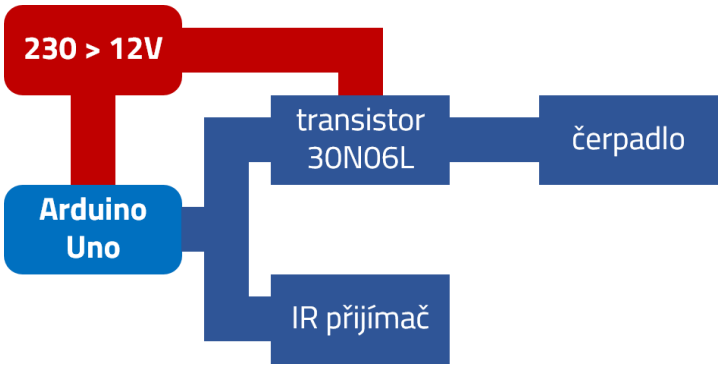
- S → digitální pin 11 (PWM)
- + (+)/- (-) → připojení napájení (5V) a uzemnění k Arduino

Finanční náklady:

- Keyes module TSOP1838: ~1 USD

Vzorový program, ze kterého jsem vycházel, je přiložen jako příloha č. 10.

5.3.6 Zjednodušené schéma zapojení stanice



Obrázek 37 Zjednodušené zapojení schéma stanice

5.4 Dráha

Jak již bylo zmíněno v předchozích kapitolách, dráha se skládá z kritických bodů. Těmito body jsou květiny, křižovatky a stanice. Jelikož RFID technologie umožňuje jednoznačnou identifikaci, rozhodl jsem se pro každý kritický bod zanést specifické softwarové vybavení. Ze zdrojového kódu robota, kde je veškerá logika,

jsem body rozdělil do tří částí podle toho, jaký tag robot sejme. V následující podkapitole společně s celou logikou a algoritmem popíši, jak RFID tagy ovlivňují a co způsobují.

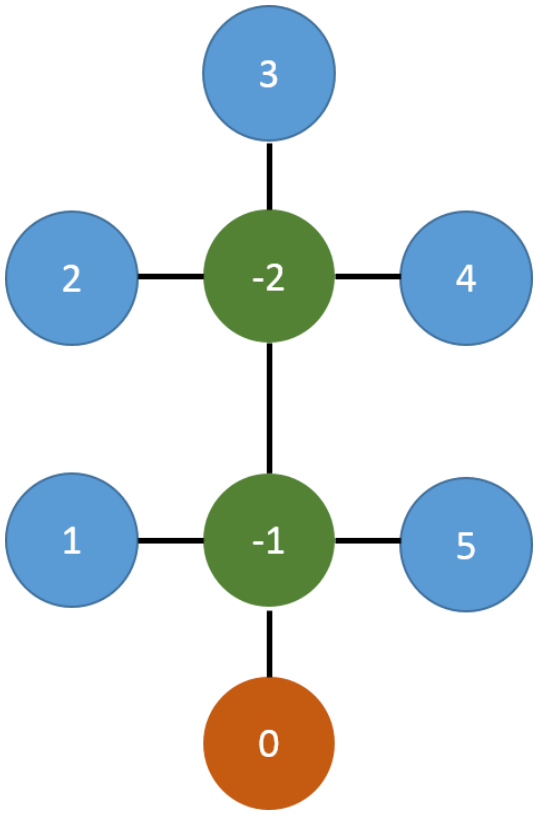
5.5 Algoritmus

Vývoj algoritmu může u podobných systémů zabrat několik týdnů i měsíců. Přistoupil jsem k řešení, které se přímo váže na testovací dráhu. Nelze tedy jednoduše rozšířit mapu bez softwarové úpravy tak, aby vše fungovalo.

5.5.1 Navigace

Hlavní složkou softwarového vybavení robota je navigace. Bez ní by nebylo možné splnit účel.

Logika softwaru spočívá v jednoduché smyčce, která neustále kontroluje signál RFID tagu a poté využívá softwaru pro řízení po černé vodící čáře, po které se robot pohybuje. K navigaci je označení bodů přiřazeno k RFID tagům následovně:



Obrázek 38 Opozicovaná testovací mapa

Přičemž 0 je označení pro stanici, negativní čísla jsou označením pro křižovatky a kladná čísla označují květiny.

5.5.2 Akce pro skupinu tagů

Stanice - zde robot kontroluje své zdroje, případně si nechá od stanice doplnit vodu pomocí infračerveného vlnění. Stanice signál zachytí a doplní vodu. Jakmile měřič tekutiny v zásobníku robota změří dostatek kapaliny, vyšle signál stanici na zastavení doplňování. Také zde probíhá kontrola, zda jsou všechny květiny zalité. Pokud ano, robot stojí.

Květina - zde robot přečte identifikátor, zjistí si hodnotu potřebnou k zalití a květinu zalije.

Křižovatka - po přečtení se robot rozhoduje na základě záznamu předchozího bodu, pokud záznam není (robot byl postaven ze začátku kamkoliv a není schopen určit, odkud přijel), pojede rovně po černé čáře. Pokud záznam existuje, vytvoří si pomocí pole mapu číselného označení bodů, která odpovídá jeho směru pohledu. Indexy této mapy mají staticky přiřazeny úhly otáčení k bodům. 1. index - otočení o 180°, 2. index - otočení o 90° doleva, 3. index - jet rovně, 4. index - otočení o 90° doprava. Probíhá i kontrola zdrojů, při nízké hodnotě si robot vyhledá z mapy nejvyšší číslo z intervalu $< -\infty, 0 >$. Tento krok zajistí nastavení směru do stanice.

Pro vyhledání květin v dosahu křižovatky hledá nejnižší číslo z intervalu $< 1, +\infty >$. Jakmile číslo najde, zkontroluje u něj zbývající hodnotu k zavlažení a v případě hodnoty nad 0 se natočí k tomuto bodu a dojede květinu zalít. Pokud jsou všechny květiny z dosahu křižovatky zalité, nastaví pro danou křižovatku hodnotu "zalito" a pokračuje do další křižovatky. Po zalití či doplnění vody se robot vždy otočí o 180°, aby zachytil směr jízdy a mohl jet po černé čáře k dalšímu bodu. Jakmile jsou všechny květiny zalité, zamíří robot do stanice.

5.5.3 Zalévání a doplňování vody

Jelikož každá květina má svůj jednoznačný identifikátor, bylo možné tuto vlastnost použít pro určení konkrétního množství vody na zalití. Lze tedy nastavit v rozsahu typu `int` potřebnou hodnotu na zalití různou pro každou květinu v dosahu v programu robota. Zadávaná hodnota představuje hodnotu v mililitrech. K tomu byla uzpůsobená i logika zalévání.

Program způsobuje při zalévání kontrolu vody v zásobníku po určitých cyklech a postupných snižováním zbývající hodnoty zalití. Pokud během zalévání dojde voda, robot přestane a dojede si doplnit vodu do stanice.

5.5.4 Detekce překážek

Algoritmus při každém pohybu kontroluje pomocí ultrazvuku vzdálenost překážky. Robot se zastaví a zapne se připojený bzučák, jakmile je zachycena překážka ve vzdálenosti 5 cm a menší.

5.5.5 Komunikace

Jakmile robot dorazí do stanice, zastaví a ještě jednou překontroluje měřič hladiny. Pokud robot potřebuje vodu, vyšle infračervenou diodou signál, konkrétně znak typu char 'A'. Při další iteraci cyklu vždy kontroluje měřič hladiny. Jakmile má robot dostatečně vody v zásobníku, vyšle znak typu char 'N'.

Stanice je naprogramována tak, že svým přijímačem kontroluje příchozí signál. Pokud zachytí signál a po rozkódování odpovídá znaku 'A', zapne čerpadlo, které pumpuje z připravené objemné nádoby vodu do zásobníku robota. Pokud zachytí signál odpovídající znaku 'N', vypne čerpadlo.

5.5.6 Souhrn použitých součástek a finanční zátěž

V tabulce 2 Seznam použitých komponent a jejich finanční náročnost je uvedeno finanční zátěž součástek. Zároveň představuje výčet všech použitých součástek. V ceně není zahrnuta kostra stanice, spojovací materiál, zásobník na vodu uchycený na robotovi a podvozek s elektromotory a černou PVC pásku na označení

dráhy. Oddělení robotiky mi pro potřeby práce zapůjčilo podvozek s elektromotory, dále vývojovou platformu Arduino Mega 2560 a H-můstek L293D na ovládání elektromotrů. Cena zásobníku je velmi ovlivněna materiálem, tvarem a rozměry. Cena použitého podvozku byla přibližně 150 USD.

Celková cena se dá případně snížit nákupem klona Arduino Mega nebo levnější baterií.

Tabulka 2 Seznam použitých komponent a jejich finanční náročnost

Hardware	Cena/ks	Počet kusů	Celkem
Stanice			
Funduino Uno (Arduino Uno klon)	5,0	1,0	5,0
Infračervený přijímač Keyes module TSOP1838	1,0	1,0	1,0
Čerpadlo s RS-360SH	4,0	1,0	4,0
Keyes MOS module s 30N06L	1,0	1,0	1,0
Pojízdný robot			
Arduino Mega 2560	45,0	1,0	45,0
Čerpadlo s RS-360SH	4,0	1,0	4,0
Keyes MOS module s 30N06L	1,0	1,0	1,0
Obvod pro měření baterie	1,0	1,0	1,0
Obvod pro měření kapaliny (2 úrovně)	1,0	1,0	1,0
Infračervený vysílač IR Led (940nm)	0,1	1,0	0,1
Infračervený sensor na sledování čáry KY-033	0,8	5,0	4,0
RFID vysílač RDM630 s anténou	2,5	1,0	2,5
Ultrazvukový sensor HC-SR04	1,0	1,0	1,0
H-můstek L293D	0,7	1,0	0,7
Nepájivé pole	1,0	1,0	1,0
Nepájivé pole 170pin	0,4	1,0	0,4
Baterie Zippy 2800 30C Li-Po	20,0	1,0	20,0
Nádrž na vodu			-
Podvozek s elektromotory			-
Bzučák BPT 14X	1,0	1,0	1,0
Led diody	0,1	2,0	0,2
Dráha			
RFID tag 125kHz	0,2	8,0	1,6
Černá PVC páska			
Celkem v dolarech			95,0
v korunách, přibližně			2300,0

Fotografie robota a stanice příloze č. 15.

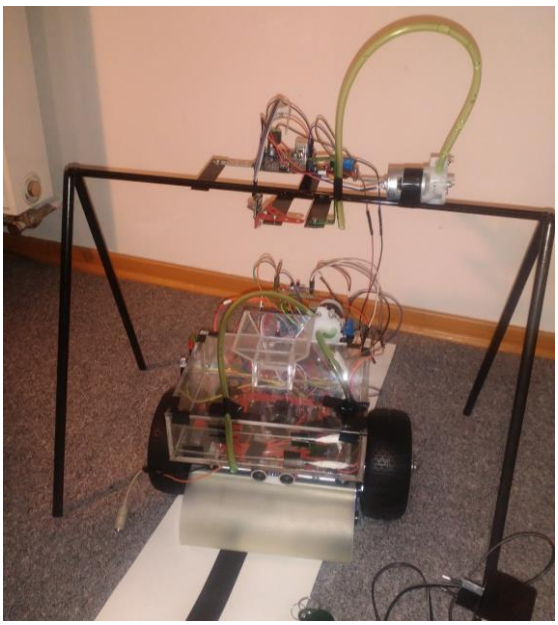
Robot se zapíná a vypíná přes mechanický spínač.

6 Závěr

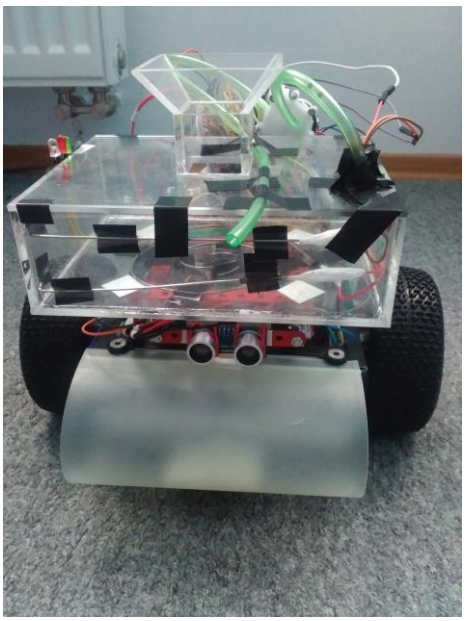
Pro naplnění cíle bakalářské práce byla provedena analýza dostupných technologií vzhledem k řešené problematice. Na základě těchto poznatků byl vytvořen návrh součástek, které celý systém bude využívat včetně zdůvodnění. Součástky poté byly řádně fyzicky umístěné na robota, stanici a dráhu. V průběhu byly jednotlivé způsoby otestovány z hardwarového i softwarového hlediska. Následně byly jednotlivé komponenty spojeny ve funkční celek doplněný a upravený o softwarovou část, která zajistila kompatibilitu a funkčnost robota a stanice. Současně byl vymyšlen a otestován algoritmus zajišťující samostatné zalévání květin. Během konstrukce byl robot průběžně testován z hlediska logiky a především pohybu. Výsledkem práce je reálně postavený robot schopný samostatného zalévání květin a stanice, která slouží jako výchozí bod a zároveň místo, kde si robot doplňuje vodu.

Řešení mimo jiné obsahuje 3 hlavní problémy, které mohou nastat při reálném nasazení. Prvním problémem je dobíjení baterie. Stanice bohužel není vybavena dokem či bezdrátovým nabíjecím adaptérem na doplnění energie baterie. V tomto případě se musí systém spoléhat na uživatele, který ručně musí dobít baterii. Druhé úskalí spočívá v celkové kontrole dráhy. Jakmile se na dráze objeví překážka, tak se robot zastaví a opět čeká na uživatele, aby překážku odstranil. Třetí možnou problematickou oblastí je softwarová úprava. Výsledný systém funguje pouze na zvolenou testovací mapu a v případě změny mapy by bylo nutné upravit zdrojový kód. S tím souvisí nutnost přehrát zdrojový kód v případě změny hodnot k zalití jednotlivých květin.

Přesto si myslím, že současné řešení, s případnou úpravou zmíněných problémů, je prakticky využitelné, zejména v případech, kdy by nebylo možné využít rozvod zavlažovacích trubiček. Jako jednoznačnou výhodu vidím mobilitu a jednoduchost řešení oproti staticky umístěnému zavlažovacímu systému včetně nenucenost dalších rozvodů přímo ke květinám.



Obrázek 39 Robot a stanice



Obrázek 40 Robot

7 Seznam použitých zdrojů

[1] *Invention Story and History of Development of Arduino* [online]. [cit. 2016-03-15]. Dostupné z: http://www.circuitstoday.com/story-and-history-of-development-of-arduino

[2] *Raspberry Pi Products - Where to Buy Raspberry Pi* [online]. [cit. 2016-03-15]. Dostupné z: https://www.raspberrypi.org/products/

[3] GOPAL, Kondaveeti Arun. *Beagle board* [online]. 2012 [cit. 2016-03-15]. Dostupné z: http://www.slideshare.net/kondaveetiarungopal/beagle-board

[4] BECHYNSKÝ, Štěpán. Raspberry Pi 2 a Windows 10 IoT Core. *Zdroják* [online]. 2015 [cit. 2016-03-15]. Dostupné z: https://www.zdrojak.cz/clanky/raspberry-pi-2-windows-10-iot-core/

[5] *Arduino - Products* [online]. [cit. 2016-03-15]. Dostupné z: https://www.arduino.cc/en/Main/Products

[6] *Raspberry Pi Products - Where to Buy Raspberry Pi* [online]. [cit. 2016-03-15]. Dostupné z: https://www.raspberrypi.org/products/

[7] *BeagleBoard.org - boards* [online]. [cit. 2016-03-15]. Dostupné z: http://beagleboard.org/boards

[8] How GPS Works - Poster. *GPS.gov* [online]. [cit. 2016-03-15]. Dostupné z: http://www.gps.gov/multimedia/poster/

[9] LAMANCE, Jimmy, Javier DESALAS a Jani JÄRVINEN. Assisted GPS A Low-Infrastructure Approach. *GPS World* [online]. 2002, , 51 [cit. 2016-03-15]. Dostupné z: http://www.gpsworld.com/wp-content/uploads/2012/09/gpsworld_Innovation_0302.pdf

[10] *History Makers - 1868 Dederick's Steam Man* [online]. 2007 [cit. 2016-03-15]. Dostupné z: http://davidbuckley.net/DB/HistoryMakers/1868DederickSteamMan.htm

[11] *Honda Worldwide | ASIMO | Evolution of ASIMO* [online]. [cit. 2016-03-15]. Dostupné z: http://world.honda.com/ASIMO/technology/index.html

[12] VOJÁČEK, Antonín. Principy průmyslových čerpadel: 1.díl – zubová čerpadla. *Automatizace.HW.cz* [online]. 2011 [cit. 2016-03-05]. Dostupné z: http://automatizace.hw.cz/principy-prumyslovych-cerpadel-1dil-zubova-cerpadla

[13] VOJÁČEK, Antonín. Principy průmyslových čerpadel: 2.díl – rotační lobe pumpy. *Automatizace.HW.cz* [online]. 2011 [cit. 2016-03-05]. Dostupné z: http://automatizace.hw.cz/principy-prumyslovych-cerpadel-2dil-rotacni-lobe-pumpy

[14] VOJÁČEK, Antonín. Principy průmyslových čerpadel: 3.díl – lopatková čerpadla. *Automatizace.HW.cz* [online]. 2011 [cit. 2016-03-05]. Dostupné z: http://automatizace.hw.cz/principy-prumyslovych-cerpadel-3dil-lopatkova-cerpadla-vane-pumps

[15] *Water Level Measurement with the Ping* [online]. Parallax INC [cit. 2016-03-15]. Dostupné z: http://www.robotshop.com/media/files/PDF/water-level-with-the-ping-28015.pdf

[16] WASSER, Leah A. The Basics of LiDAR. *National Ecological Observatory Network* [online]. 2014 [cit. 2016-03-15]. Dostupné z: http://neondataskills.org/self-paced-tutorial/1_About-LiDAR-Data-Light-Detection-and-Ranging_Activity1/

[17] NAVE, Carl Rod. *HyperPhysics: DC Electric Motors* [online]. Department of Physics and Astronomy, Georgia State University, Atlanta, 2012 [cit. 2016-03-05]. Dostupné z: http://hyperphysics.phy-astr.gsu.edu/hbase/magnetic/motdc.html

[18] KRŮŽELA, Miroslav. *Regulace otáček elektromotoru*. Zlín, 2006. Bakalářská práce. Univerzita Tomáše Bati ve Zlíně.

[19] *Čárový kód - Informace o čárovém kódu* [online]. [cit. 2016-03-15]. Dostupné z: http://www.carovy-kod.info/

[20] SOMMEROVÁ, Martina. *Základy RFID technologií: Výukový materiál* [online]. Vysoká škola báňská - Technická univerzita Ostrava, 32 [cit. 2016-03-05]. Dostupné z: http://rfid.vsb.cz/export/sites/rfid/cs/informace/RFID_pro_Logistickou_akademii.pdf

[21] WEIS, Stephen A. *RFID (Radio Frequency Identification): Principles and Applications* [online]. MIT: Computer Science and Artificial Intelligence Laboratory, , 23 [cit. 2016-03-05]. Dostupné z: http://www.eecs.harvard.edu/cs199r/readings/rfid-article.pdf

[22] ŠIŠKA, Martin. *IMPULZOVÉ MODULACE*. Brno, 2013. Diplomová práce. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ. Vedoucí práce Ing. RADIM ČÍŽ, Ph.D., 13 str.

[23] TKÁČ, Josef. Jak na Bluetooth v rekordním čase. 1. vyd. Praha : Grada Publishing, a.s., 2005. 81s. ISBN 80-247-1081-1

[24] FUCHS, Michal. *Řízení bezdrátové komunikace pomocí Zigbee*. Brno, 2008. Diplomová práce. Vysoké Učení Technické v Brně.

[25] VOHNOUT, R. *Počítačové sítě I: Fyzická vrstva (2. část)* (přednáška) České Budějovice: Jihočeská univerzita v Českých Budějovicích, Přírodovědecká fakulta, Ústav Aplikované Informatiky, 16.10.2014

http://moodle.prf.jcu.cz/pluginfile.php/15726/mod_resource/content/1/PSI-Fyz.vrstva-2-Vohnout.pdf

[26] Co je to PID regulace teploty? *Diskuse Elektrika.cz* [online]. [cit. 2016-03-22]. Dostupné z: http://diskuse.elektrika.cz/index.php/topic,13998.0.html

[27] ČECH, Zdeněk. *Metody seřizování PID regulátorů*. Pardubice, 2015. Bakalářská práce. Univerzita Pardubice.

[28] SINHA, Nishant, Pranit SAMARTH a Satish S. NAIR. *How does an Ultrasonic Sensor Work? - Lesson - teachengineering.com* [online]. [cit. 2016-03-22]. Dostupné z: https://www.teachengineering.org/view_lesson.php?url=collection/umo_/lessons/umo_sensorswork/umo_sensorswork_lesson06.xml

[29] VISHWAM, Aggarwal. How to build an IR Sensor. In: *MaxEmbedded.com* [online]. 2013 [cit. 2016-03-22]. Dostupné z: http://maxembedded.com/2013/08/how-to-build-an-ir-sensor/

[30] WILLIAMS, Al. *Microcontroller projects using the Basic Stamp*. 2nd ed. Lawrence, Kan.: CMP Books, c2002. ISBN 15-782-0101-2.

[31] *IR Sensor: What is an IR sensor?* [online]. Carnegie Mellon University, Carnegie Mellon's Robotics Academy [cit. 2016-03-05]. Dostupné z: http://education.rec.ri.cmu.edu/content/electronics/boe/ir_sensor/1.html

[32] KLUGER, Ondřej. *Autonomní robot pro měření uzavřených prostor*. České Budějovice, 2014. Bakalářská práce. Jihočeská univerzita v Českých Budějovicích.

[33] MARIAN, Popescu. Arduino Water Level Indicator + Controller. *Electronics Projects Circuits* [online]. [cit. 2016-03-06]. Dostupné z: http://www.electroschematics.com/9964/arduino-water-level-indicator-controller/

[34] *IRremote Arduino Library* [online]. [cit. 2016-03-20]. Dostupné z: https://github.com/z3t0/Arduino-IRremote/

[35] SRINIVAS, Samvrit. *PID Controller – Simply Explained* [online]. [cit. 2016-04-12]. Dostupné z: http://samvrit.tk/tutorials/pid-controller-simply-explained/

[36] N, Hema, Reema ASWANI a Monisha MALIK. *Plant Watering Autonomous Mobile Robot* [online]. India, 2012, **1**(3) [cit. 2016-01-13]. ISSN 2089-4856. Dostupné z: http://iaesjournal.com/online/index.php/IJRA/article/view/1305/682

8 Seznam obrázků

9 Seznam příloh

Součástí práce je CD, které obsahuje elektronickou kopii nezkrácené bakalářské práce, zdrojový kód robota a zdrojový kód stanice uložený ve formátu .ino.

Příloha č. 1

```
/*
 Popis: Vzorový program pro h-můstek L293D s 1 připojeným elektromotorem
 Zdroj: http://www.instructables.com/id/Control-your-motors-with-L293D-and-
Arduino/
 Datum: 6.3.2016
 Testovací platforma: Arduino Uno
 Testovací modul: L293D + elektromotor
 Test programu: Úspěšně proveden
*/
#define motor_A_speed 5 //digitální pin PWM na ovládání rychlosti
#define motor_A_dir1 7 //digitální pin na ovládání směru
#define motor_A_dir2 8 //digitální pin na ovládání směru

void setup()
{
	pinMode(motor_A_speed, OUTPUT); //nastavení pinu na výstupní vlastnost
	pinMode(motor_A_dir1, OUTPUT); //nastavení pinu na výstupní vlastnost
	pinMode(motor_A_dir2, OUTPUT); //nastavení pinu na výstupní vlastnost
}
void loop()
{
	digitalWrite(motor_A_dir1, LOW); //směr 1 vypnout
	digitalWrite(motor_A_dir2, LOW); //směr 2 vypnout
	digitalWrite(motor_A_dir1, HIGH); //zapnutí směru motoru - doprava/doleva
	//pro pohyb elektromotoru je nutné mít rozdílné hodnoty ve směrech
(LOW/HIGH, HIGH/LOW)
	int speedMotor = 123; //rychlost motoru v rozsahu 0-255 (0-100%)
	analogWrite(motor_A_speed, speedMotor); //nastavení rychlosti na 50%
	analogWrite(motor_A_speed, 0); //nastavení rychlosti na 0%
}
```

Příloha č. 2

```
/*
Popis: Kód pro uživatelský PID regulátor k ovládání motorů na základě vstupních
hodnot infračervených snímačů
Zdroj: http://samvrit.tk/tutorials/pid-control-arduino-line-follower-robot/
Datum: 6.3.2016
Testovací platforma: Arduino Uno
Testovací modul: Arduino, h-můstek L293D + motory
Test programu: Úspěšně proveden
*/

#define motor_A_speed 7 //digitální pin PWM na ovládání rychlosti - motor A
#define motor_B_speed 6 //digitální pin PWM na ovládání rychlosti - motor B
#define motor_A_dir1 40 //digitální pin na ovládání směru - motor A
#define motor_A_dir2 41 //digitální pin na ovládání směru - motor A
#define motor_B_dir1 34 //digitální pin na ovládání směru - motor B
#define motor_B_dir2 35 //digitální pin na ovládání směru - motor B


#define IR_1 A1 //Analogový pin pro infračervený sensor
#define IR_2 A2 //Analogový pin pro infračervený sensor
#define IR_3 A3 //Analogový pin pro infračervený sensor
#define IR_4 A4 //Analogový pin pro infračervený sensor
#define IR_5 A5 //Analogový pin pro infračervený sensor

float Kp = 0, Ki = 0, Kd = 0; //Konfigurační složky PID regulátoru
float error = 0, P = 0, I = 0, D = 0, PID_value = 0;
/* Jednotlivé hodnoty:
error: právě naměřená odchylka
P: výstupní hodnota regulátoru, složka P
I: výstupní hodnota regulátoru, složka I
D: výstupní hodnota regulátoru, složka D
PID_value: součet všech tří složek, tato hodnota upravuje rychlosti elektromotorů
*/

float previous_error = 0; //Předchozí odchylka

int sensor[5] = { 0, 0, 0, 0, 0 }; //Pole, do kterého se budou ukládat hodnoty
infračervených sensorů (0/1)
int initial_motor_speed = 150; //rychlost elektromotorů (rozsah 0-255)

void read_sensor_values(void); //metoda, která přečte vstupní hodnoty sensorů
a uloží je do pole
void calculate_pid(void); //metoda počítající hodnotu konečnou PID
void motor_control(void); //metoda aplikující PID na rychlost motorů


void setup()
{
	pinMode(motor_A_speed, OUTPUT); //nastavení pinu na výstupní vlastnost
	pinMode(motor_B_speed, OUTPUT); //nastavení pinu na výstupní vlastnost
	pinMode(motor_A_dir1, OUTPUT); //nastavení pinu na výstupní vlastnost
	pinMode(motor_A_dir2, OUTPUT); //nastavení pinu na výstupní vlastnost
	pinMode(motor_B_dir1, OUTPUT); //nastavení pinu na výstupní vlastnost
	pinMode(motor_B_dir2, OUTPUT); //nastavení pinu na výstupní vlastnost

	pinMode(IR_1, INPUT); //nastavení pinu na vstupní vlastnost
	pinMode(IR_2, INPUT); //nastavení pinu na vstupní vlastnost
	pinMode(IR_3, INPUT); //nastavení pinu na vstupní vlastnost
```

```
	pinMode(IR_4, INPUT); //nastavení pinu na vstupní vlastnost
	pinMode(IR_5, INPUT); //nastavení pinu na vstupní vlastnost

	Serial.begin(9600); //nastavení sériové komunikace s PC (Sériový monitor)
	//9600 = baud (modulační rychlost)

}

void loop()
{
	read_sensor_values(); //1. krok - získat hodnoty ze sensorů, výstupem je
odchylka
	calculate_pid(); //2. krok - na základě odchylky spočítat PID_value
	motor_control(); //3. krok - na základě hodnoty PID_value upravit rychlost
elektromotorů

}

void read_sensor_values()
{
	//přečtení výstupních hodnot pomocí metody digitalRead([pin])

	sensor[0] = digitalRead(IR_1);
	sensor[1] = digitalRead(IR_2);
	sensor[2] = digitalRead(IR_3);
	sensor[3] = digitalRead(IR_4);
	sensor[4] = digitalRead(IR_5);

	//výpočet odchylky

	if ((sensor[0] == 0) && (sensor[1] == 0) && (sensor[2] == 0) && (sensor[3]
== 0) && (sensor[4] == 1)) {
		error = 4;
	}
	else if ((sensor[0] == 0) && (sensor[1] == 0) && (sensor[2] == 0) &&
(sensor[3] == 1) && (sensor[4] == 1)) {
		error = 3;
	}
	else if ((sensor[0] == 0) && (sensor[1] == 0) && (sensor[2] == 0) &&
(sensor[3] == 1) && (sensor[4] == 0)) {
		error = 2;
	}
	else if ((sensor[0] == 0) && (sensor[1] == 0) && (sensor[2] == 1) &&
(sensor[3] == 1) && (sensor[4] == 0)) {
		error = 1;
	}
	else if ((sensor[0] == 0) && (sensor[1] == 0) && (sensor[2] == 1) &&
(sensor[3] == 0) && (sensor[4] == 0)) {
		error = 0;
	}
	else if ((sensor[0] == 0) && (sensor[1] == 1) && (sensor[2] == 1) &&
(sensor[3] == 0) && (sensor[4] == 0)) {
		error = -1;
	}
	else if ((sensor[0] == 0) && (sensor[1] == 1) && (sensor[2] == 0) &&
(sensor[3] == 0) && (sensor[4] == 0)) {
		error = -2;
	}
```

```
	else if ((sensor[0] == 1) && (sensor[1] == 1) && (sensor[2] == 0) &&
(sensor[3] == 0) && (sensor[4] == 0)) {
		error = -3;
	}
	else if ((sensor[0] == 1) && (sensor[1] == 0) && (sensor[2] == 0) &&
(sensor[3] == 0) && (sensor[4] == 0)) {
		error = -4;
	}
	else if ((sensor[0] == 0) && (sensor[1] == 0) && (sensor[2] == 0) &&
(sensor[3] == 0) && (sensor[4] == 0)) {

		if (error == -4) error = -5;
		else error = 5;
	}


}

void calculate_pid()
{
	P = error; //výpočet složky P
	I = I + error; //výpočet složky I
	D = error - previous_error; //výpočet složky D

	PID_value = (Kp*P) + (Ki*I) + (Kd*D); //výpočet PID hodnoty

	previous_error = error; //pro ovlivnění dalšího kroku je nutné si uložit
stávající odchylku
}

void motor_control()
{

	int left_motor_speed = initial_motor_speed - PID_value; //úprava rychlosti
levého motoru
	int right_motor_speed = initial_motor_speed + PID_value; //úprava rychlosti
pravého motoru

	//ošetření maximální a minimální hodnoty
	if (left_motor_speed > 255) {
		left_motor_speed = 255;
	}
	if (right_motor_speed > 255) {
		right_motor_speed = 255;
	}

	if (left_motor_speed < 0) {
		left_motor_speed = 0;
	}
	if (right_motor_speed < 0) {
		right_motor_speed = 0;
	}

	//nastavení směru elektromotorů
	digitalWrite(motor_A_dir1, LOW);
	digitalWrite(motor_A_dir2, HIGH);
	digitalWrite(motor_B_dir1, LOW);
	digitalWrite(motor_B_dir2, HIGH);
```

```
      analogWrite(motor_A_speed, left_motor_speed);   //nastavení rychlosti levého
motoru
      analogWrite(motor_B_speed, right_motor_speed);  //nastavení rychlosti
pravého motoru


}
```

Příloha č. 3

```
/*
 Popis: Vzorový program pro správnou funkčnost infračerveného sensoru
 Zdroj: http://www.instructables.com/id/DIY-Line-Follower-Sensor-Array/step4/Some-
code-to-start-out-with/
 Datum: 5.3.2016
 Testovací platforma: Arduino Uno
 Testovací modul: KY-033
 Test programu: Úspěšně proveden
*/
#define infraPin 8 //digitální/analogový pin
void setup() {
	Serial.begin(9600); //nastavení sériové komunikace s PC (Sériový monitor)
	pinMode(infraPin, INPUT); //nastavení pinu na vstupní vlastnost
	//9600 = baud (modulační rychlost)
}
void loop() {
	int stav_IR = digitalRead(infraPin);
//pomocí metody digitalRead([pin]) lze  číst vstupní signál směřující do platformy
	if (stav_IR == 0) { //pokud se vstupní hodnota rovná 0
		Serial.println("Nezachytil jsem vysilany signal. Neni zde prekazka/je
zde cerna cast drahy.");
	}
	else if (stav_IR == 1) { // pokud se vstupní hodnota rovná 1
		Serial.println("Zachytil jsem signal. Je zde prekazka/je zde bila
cast drahy.");
	}
	else { // pokud je vstupní hodnota jiná než 0 nebo 1
		Serial.println("Chyba: jina hodnota");
	}
}
```

Příloha č. 4

```
/*
 Popis: Vzorový kód pro čtečku RFID - 125 KHz
 Zdroj: http://arduino8.webnode.cz/news/lekce-33-arduino-a-modul-ctecky-rfid/
 Datum: 6.3.2016
 Testovací platforma: Arduino Uno
 Testovací modul: RDM630 s anténou + RFID tag
 Test programu: Úspěšně proveden
*/
#include <SoftwareSerial.h> //importování knihovny pro vytvoření komunikačního
portu kdekoliv na platformě
SoftwareSerial RFID(2, 3); //nastavení pro vstupní(Rx) a výstupní pin(Tx)
ve formátu ([RxPin],[TxPin])
//v tomto případě digitální pin 2 bude zapojen a spojen s RDM630, digitální pin 3
nebude použit
void setup() {
      RFID.begin(9600); //dle předchozího nastavení zapnout komunikaci
      Serial.begin(9600); //nastavení sériové komunikace s PC (Sériový monitor)
      //9600 = baud (modulační rychlost)
}
void loop() {
      if (RFID.available() > 0) { //pokud jsou nějaká data zachycena anténou
            Serial.println(RFID.read()); //tyto data zobrazit na sériový monitor
      }
}
```

Příloha č. 5

```
/*
 Popis: Vzorový kód pro ultrazvuk HC-SR04
 Zdroj: http://www.tautvidas.com/blog/2012/08/distance-sensing-with-ultrasonic-
sensor-and-arduino/
 Datum: 6.3.2016
 Testovací platforma: Arduino Uno
 Testovací modul: HC-SR04
 Test programu: Úspěšně proveden
*/
#define trigPin 7 //příklad výstupního digitálního pinu, spojí se s Trig
na testovacím modulu
#define echoPin 8 //příklad vstupního digitálního pinu, spojí se s Echo
na testovacím modulu
void setup()
{
	pinMode(trigPin, OUTPUT); //nastavení pinu na výstupní vlastnost
	pinMode(echoPin, INPUT); //nastavení pinu na vstupní vlastnost
	Serial.begin(9600); //nastavení sériové komunikace s PC (Sériový monitor)
	//9600 = baud (modulační rychlost)
}
void loop()
{
	int duration, distance;
	digitalWrite(trigPin, LOW); //vypnutí vysílače
	delayMicroseconds(2); //zdržení 2 mikrosekundy
	digitalWrite(trigPin, HIGH); //zapnutí vysílače
	delayMicroseconds(10); //zdržení 10 mikrosekund
	digitalWrite(trigPin, LOW); //vypnutí vysílače
	duration = pulseIn(echoPin, HIGH); //doba zachycení prvního vysílaného
signálu
	distance = (duration / 2) / 29.1; //převod doby do vzdálenosti
v centimetrech
	Serial.print("Zachycena vzdalenost: "); //formátování výstupu na sériový
monitor
	Serial.print(distance);
	Serial.println(" cm.");
}
```

Příloha č. 6

```
/*
 Popis: Vzorový kód na měřič kapaliny v nádobě
 Zdroj: http://www.electroschematics.com/9964/arduino-water-level-sensor/
 Datum: 6.3.2016
 Testovací platforma: Arduino Uno + nádrž s vodou
 Testovací modul: Obvod skládající se z transistorů a odporů dle schématu
 Test programu: Úspěšně proveden
*/
#define lowLevelPin 8 //příklad vstupního digitálního pinu
#define highLevelPin 9 //příklad vstupního digitálního pinu
const byte sensors = 2; //počet celkových sensorů v úrovních
int level = 0; //proměnná pro aktuální úroveň
//0 - pod nejnižší úrovní (nedostatek vody)
//1 - nad nejnižší úrovní, pod nejvyšší úrovní
//2 - nad nejvyšší úrovní (dostatek vody)

void setup() {
	pinMode(lowLevelPin, INPUT); //nastavení vstupní vlastnosti pinu
	pinMode(highLevelPin, INPUT); //nastavení vstupní vlastnosti pinu
	Serial.begin(9600); //nastavení sériové komunikace s PC (Sériový monitor)
	//9600 = baud (modulační rychlost)
}
void loop() {
	if (digitalRead(lowLevelPin) == LOW) { //pokud zde není signál => voda je
nad úrovní
		level++;
	}
	if (digitalRead(highLevelPin) == LOW) { //pokud zde není signál => voda je
nad úrovní
		level++;
	}

}
```

Příloha č. 7

```
/*
 Popis: Vzorový kód pro ovládání motoru přes Keyes MOS Module s 30N06L
 Zdroj: Arduino Examples/Basics/Fade
 Datum: 6.3.2016
 Testovací platforma: Arduino Uno
 Testovací modul: Keyes MOS module s 30N06L
 Test programu: Úspěšně proveden
*/
#define motor_speed 5 //digitální pin s PWM vlastností pro ovládání rychlosti
motoru

void setup()
{
      pinMode(motor_speed, OUTPUT); //nastavení pinu na výstupní vlastnost
}
void loop()
{
      int speedMotor = 123; //rychlost motoru v rozsahu 0-255 (0-100%)
      analogWrite(motor_speed, speedMotor); //nastavení rychlosti na 50%
}
```

Příloha č. 8

```
/*
 Popis: Vzorový kód pro infračervený vysílač (IR Led)
 Zdroj: http://www.instructables.com/id/Cheap-wireless-transmission-between-two-
Arduinos-w/step2/Wiring-the-emitter/
 Datum: 6.3.2016
 Testovací platforma: Arduino Uno
 Testovací modul: IR led
 Test programu: Úspěšně proveden
*/
#include <IRremote.h> //import knihovny, zdroj: https://github.com/z3t0/Arduino-
IRremote/
/*
 Připojení k pinu závisí na typu platformy (Uno/Mega/...).
 Arduino Uno - pin 3.
 Arduino Mega 2560 - pin 9.
*/
IRsend irsend; //typ proměnné, který obsahuje metody na odesílání
void setup()
{
	//není potřeba zde nic nastavovat
}
void loop()
{
	String message = "test"; //zpráva k odeslání
	for (int i = 0; i < message.length(); i++) { //odesílá zprávu po znacích
		irsend.sendRC5(message.charAt(i), 12); //odesílací protokol RC5
		delay(20); //zdržení mezi znaky
}
```

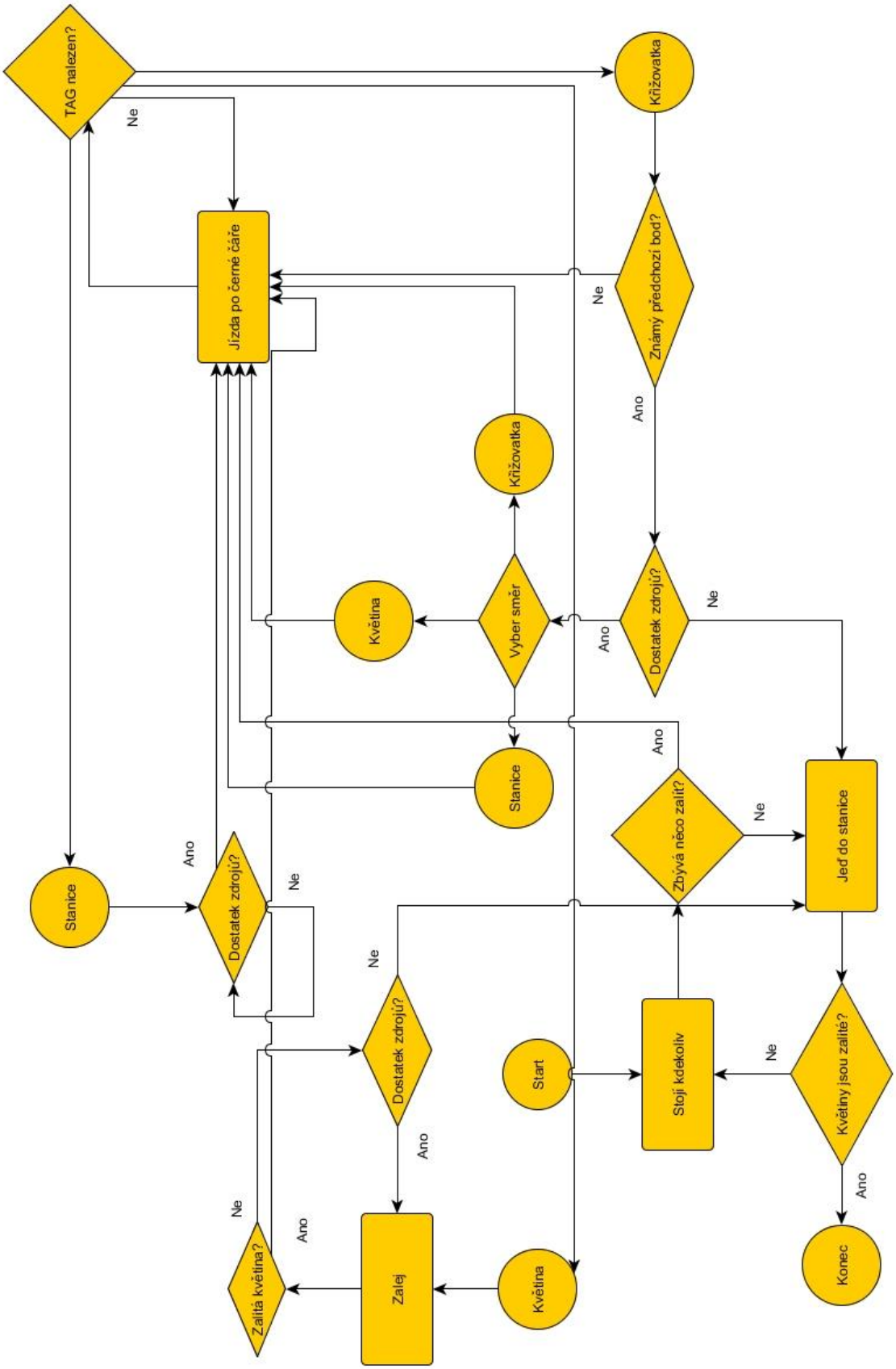
Příloha č. 9

```
/*
 Popis: Vzorový kód na měření voltů v baterii
 Zdroj: http://www.electroschematics.com/9351/arduino-digital-voltmeter/
 Datum: 6.3.2016
 Testovací platforma: Arduino Uno + baterie
 Testovací modul: Obvod skládající se z odporů dle schématu
 Test programu: Úspěšně proveden
 */
#define inputBatteryPin A1 //příklad vstupního analogového pinu
#define lowCriticalBatteryLevel 4 //nastavení dolní hranice vybití baterie
float vout = 0.0; // (voltage out)
float vin = 0.0; // = měřené napětí (voltage in)
float R1 = 100000.0; // odpor 100K
float R2 = 10000.0; // odpor 10K
int value = 0; //vstupní hodnota analogového pinu

void setup()
{
	pinMode(inputBatteryPin, INPUT); //nastavení vstupní vlastnosti
}

void loop()
{
	value = analogRead(A0);
	vout = (value * 5.0) / 1024.0;
	vin = vout / (R2 / (R1 + R2));
	if (vin < lowCriticalBatteryLevel) { //pokud je napětí pod zvolenou hranicí
	}
}
```

Příloha č. 10

```
/*
 Popis: Vzorový kód pro infračervený přijímač
 Zdroj: http://www.instructables.com/id/Cheap-wireless-transmission-between-two-
Arduinos-w/step3/Wiring-the-receiver/
 Datum: 6.3.2016
 Testovací platforma: Arduino Uno
 Testovací modul: Keyes module TSOP1838
 Test programu: Úspěšně proveden
*/
#include <IRremote.h> //import knihovny, zdroj: https://github.com/z3t0/Arduino-
IRremote/
#define RECV_PIN 11 //příklad vstupního digitálního pinu PWM
IRrecv irrecv(RECV_PIN); //Typ proměnné, v které je pomocí knihovny možné
dekódovat signál
decode_results results; //dekódovaný signál
void setup()
{
	irrecv.enableIRIn(); //metoda, která spouští zachytávání
	Serial.begin(9600); //nastavení sériové komunikace s PC (Sériový monitor)
	//9600 = baud (modulační rychlost)
}
void loop()
{
	if (irrecv.decode(&results)) { //je zachycen signál?
		Serial.print(char(results.value)); //tisk znaků na obrazovku
		irrecv.resume(); //metoda, která umožní vyprázdnit uloženou hodnotu
v modulu a zachytit další
	}
}
```

Příloha č. 11


TAG nalezen?
Ne
Křižovatka
Jízda po černé čáře
Známý předchozí bod?
Ne
Ano
Křižovatka
Květina
Výber směr
Dostatek zdrojů?
Ano
Ne
Stanice
Ano
Zbývá něco zalít?
Ne
Jeď do stanice
Stanice
Ano
Dostatek zdrojů?
Ne
Ne
Dostatek zdrojů?
Start
Stůj kdekoliv
Ne
Květiny jsou zalité?
Ano
Konec
Ne
Zalitá květina?
Ano
Zalej
Květina

Příloha č. 12

```
#include <SoftwareSerial.h>
#include <IRremote.h>

//rychlost vozidla//
#define normalSpeed 170 //0-255

//Hodnoty v ml potřebně k zalití - jednotlivé květiny
#define fl_1 300 //fl = flower/kvetina
#define fl_2 200
#define fl_3 100
#define fl_4 200
#define fl_5 190

//hodnoty potřebné vody - zbývající
int fl_1_remain = fl_1;
int fl_2_remain = fl_2;
int fl_3_remain = fl_3;
int fl_4_remain = fl_4;
int fl_5_remain = fl_5;
//pole zbývajících hodnot k zalití
int remainingWater[5] = { fl_1_remain, fl_2_remain, fl_3_remain, fl_4_remain,
fl_5_remain };

//Setup - Komunikace - infračervený vysílač
IRsend irsend; //vstupní pin D9 (striktně nastaveno v knihovně)

//Setup - měřič baterie
#define batteryPin A8 //vstupní analogový pin
float voltage_out = 0.0; // volty ven
float voltage_in = 0.0; // volty do
float resistor1 = 100000.0; // odpor 100K
float resistor2 = 10000.0; // odpor 10K
int batteryVin = 0; //naměřené volty

//Setup - piny k ovládání motorů (L293D)
#define motorR_in1 24 //D - směr R motoru: dopředu
#define motorR_in2 25 //D - směr R motoru: zpět
#define motorR_sp 3 //D-PWM (rychlost R-motoru)
#define motorL_in1 26 //D - směr L motoru: dopředu
#define motorL_in2 27 //D - směr L motoru: zpět
#define motorL_sp 4 //D-PWM (rychlost L-motoru)

//Setup - piny pro řízení směru - infračervené sensory
#define path_1 A1 //A
#define path_2 A2 //A
#define path_3 A3 //A
#define path_4 A4 //A
#define path_5 A5 //A
int sensors[5] = { 0,0,0,0,0 };

//Setup - ultrazvukový sensor
#define ul_echo 34 //D - echo pin
#define ul_trig 35 //D - trig pin

//Setup - měřič vodní hladiny
#define dc_pump 2 //D-PWM (připojení pumpy)
#define dc_pump_speed 123 //rychlost pumpy v analog 0-255
#define water_level_low 30 //D - vstup pro nizkou hodnotu
```

```
#define water_level_high 31 //D - vstup pro vysokou hodnotu

//Setup - RFID
#define rfid_rx 50 //receive data - rfid (použito) RFID čtečka
#define rfid_tx 51 //transmit data - rfid (nevyužito)
SoftwareSerial rfidReader(rfid_rx, rfid_tx);

//globální proměnné - start - ostatní
int encodedRFIDnumber; //rozkódované číslo
//globální proměnné - stop

//ostatní nastavení
#define greenLed 40 //zelená LED dioda
#define redLed 41 //červená LED dioda
#define buzzer 42 //bzučák

void setup() {

	//Setup motors - start
	pinMode(motorR_in1, OUTPUT); //nastavení výstupních pinů
	pinMode(motorR_in2, OUTPUT);
	pinMode(motorR_sp, OUTPUT);
	pinMode(motorL_in1, OUTPUT);
	pinMode(motorL_in2, OUTPUT);
	pinMode(motorL_sp, OUTPUT);
	//Setup motors - end

	//Setup IR path - start
	pinMode(path_1, INPUT); //nastavení vstupních pinů
	pinMode(path_2, INPUT);
	pinMode(path_3, INPUT);
	pinMode(path_4, INPUT);
	pinMode(path_5, INPUT);
	//Setup IR path - end

	//Setup ultrasonic sensor - start
	pinMode(ul_echo, INPUT); //nastavení vstupního pinu
	pinMode(ul_trig, OUTPUT); //nastavení výstupního pinu
	//Setup ultrasonic sensor - end

	//Setup water level w DC pump - start
	pinMode(dc_pump, OUTPUT); //nastavení výstupního pinu pro ovládání pumpy
	pinMode(water_level_low, INPUT); //nastavení vstupního pinu pro měřič -
nedostatek vody
	pinMode(water_level_high, INPUT); // nastavení vstupního pinu pro měřič -
dostatek vody
	//Setup water level w DC pump - end

	//Setup ostatní - start
	pinMode(greenLed, OUTPUT);
	pinMode(redLed, OUTPUT);
	pinMode(buzzer, OUTPUT);
	//Setup ostatní - end

	//Setup - RFID - start
	rfidReader.begin(9600); //zapnutí RFID čtečky
	//Setup - RIFD - end
}
```

```
void loop() {
	if (!watered_all) {
		rfidRead();
		if (!(getDistance() <= 5)) {
			turnBuzzer(false);
			goPID();
		}
		else {
			turnBuzzer(true);
		}
	}
}

//--------------------//
//BATTERY CHECKER-----//
//--------------------//

bool isBatteryLow() {
	batteryVin = analogRead(batteryPin);
	voltage_out = (batteryVin * 5.0) / 1024.0;
	voltage_in = voltage_out / (resistor2 / (resistor1 + resistor2));
	if (voltage_in <= 10.2) { //pokud je naměřená hodnota menší než 10.2 V
(použita 3článková baterie)
		return true;
	}
	else return false;
}

//--------------------//
//WATER LEVEL SENSOR--//
//--------------------//

int getWaterLevel() {
	int level = 0;
	if (digitalRead(water_level_low) == LOW) {
		level++;
		if (digitalRead(water_level_high) == LOW) {
			level++;
		}
	}
	return level; //return: 0-bez vody, 1-nad dolním měřičem, 2-dostatek vody
}

//--------------------//
//RFID SENSOR---------//
//--------------------//


int const offset = -2;
int const databaseSize = 8;
String rfidTags[databaseSize] = { //databáze RFID tagů a jejich umístění na dráze
	"0A00200315", //-2 - křižovatka
	"0E0066420F", //-1
	"0A0066C515", //0 - stanice
	"0400197136", //1 - květina
	"040018B577", //2
	"040018C3E5", //3
	"0B004DF422", //4
```

```
      "0C00554276"  //5
};

int lastPosition = NULL; //poslední pozice

int tagReaded[14]; //pole pro ukládání hodnot sejmutých z RFID tagu
String tagReceived = ""; //text RFID po rozkódování

void rfidRead() { //rozkódování RFID tagu
      if (rfidReader.available() > 0) {
            for (int i = 0; i < 14; i++) {
                  tagReaded[i] = rfidReader.read();
            }

            for (int j = 1; j < 11; j++) {
                  char ch = tagReaded[j];
                  tagReceived = tagReceived + ch;
            }
            delay(500);
            rfidReader.flush();
            bool exist = false;
            int encodedRFIDnumber = 0;
            for (int i = 0; i < databaseSize; i++) {
                  if (rfidTags[i] == tagReceived) {
                        exist = true;
                        encodedRFIDnumber = offset + i;
                        tagReceived = "";
                        break;
                  }
            }
            if (exist) { //pokud existuje sejmuté číslo v databázi
                  rfidDecide(encodedRFIDnumber);
            }
      }
}

void rfidDecide(int encodedRFIDnumber) {

      if (lastPosition == NULL) { //pokud je poslední pozice neznámá
            nullNavi(encodedRFIDnumber);
            return;
      }
      else { //pokud je poslední pozice známá
            notNullNavi(encodedRFIDnumber);
            return;
      }


}

//--------------------//
//NAVIGATION----------//
//--------------------//

bool arrayWatered[2] = { false,false }; //in.0 - prvni krizovatka, in.1 - druha
krizovatka (0-false, 1-true)
//false - nezalito, true - zalito
bool watered_all = false; //zalito vše?
```

```
int standardMap[4]; //staticky nastavená mapa
int arrangedMap[4]; //zpřeházená mapa podle pohledu robota v křižovatce

void doArrangedMap(int crossRoad, int lastPoint) {
	if (crossRoad == -1) { //mapa pro křižovatku -1
		standardMap[0] = 0; //bod zpět
		standardMap[1] = 1; //bod vlevo
		standardMap[2] = -2; //bod rovně
		standardMap[3] = 5; //bod vpravo
	}
	else { //mapa pro křižovatku -2
		standardMap[0] = -1; //bod zpět
		standardMap[1] = 2; //bod vlevo
		standardMap[2] = 3; //bod rovně
		standardMap[3] = 4; //bod vpravo
	}

	int startingIndex = -1;
	for (int i = 0; i < 4; i++) {
		if (standardMap[i] == lastPoint) {
			startingIndex = i;
		}
	}

	if (startingIndex < 0 || startingIndex > 4) {
		for (int i = 0; i < 4; i++) {
			arrangedMap[i] = standardMap[startingIndex++ % 4]; //nastavení
nové mapy, dle směru robota
		}
	}
}

void nullNavi(int readedNumber) { //chování bez přechozího bodu
	if (readedNumber < 0) {
		lastPosition = readedNumber;
		goStraight();
	}
	else {
		navigation(readedNumber);
    goBack();
		turn180();
	}
}

void notNullNavi(int readedNumber) { //chování se znalostí přechozího bodu
	if (readedNumber < 0) {
		doArrangedMap(readedNumber, lastPosition); //nastav mapu dle pohledu

		if (isBatteryLow() || getWaterLevel() == 0) { //pokud jsou zdroje
nízké > směr stanice
			int station =
findLargerstNegativeNumberIncludeZero(arrangedMap); //<-nekonečno,0>
			turnToNumber(arrangedMap, station);
		}
		else { //zdroje OK > najdi nejmenší hodnotu (1-5) z mapy, která
potřebuje zalít
			int number =
findLowestPositiveNumberWhichNeedsToBeWatered(arrangedMap, readedNumber);
```

```
                  if (number == -1) { //pokud v dosahu není květina nutná k
zalití
                        int num = findLowestNumber(arrangedMap);
                        int index = abs(num) - 1;
                        if (!arrayWatered[index]) { //jeď do další křižovatky,
pokud má hodnotu false = nezalito
                              turnToNumber(arrangedMap, num);
                              return;
                        } else { //jinak jeď do stanice
                              int station =
findLargerstNegativeNumberIncludeZero(arrangedMap);
                              turnToNumber(arrangedMap, station);
                              return;
                        }
                  }
                  turnToNumber(arrangedMap, number);
                  return;
            }
      }
      else {
            navigation(readedNumber);
    goBack();
            turn180();
      }
}

void turnGreenLight(bool on) { //zapínání/vypínání zelené LED diody
      if (on) {
            digitalWrite(greenLed, HIGH);
      }
      else {
            digitalWrite(redLed, LOW);
      }
}

void turnRedLight(bool on) { //zapínání/vypínání červené LED diody
      if (on) {
            digitalWrite(redLed, HIGH);
      }
      else {
            digitalWrite(redLed, LOW);
      }
}

void turnBuzzer(bool on) { //zapínání/vypínání bzučáku
      if (on) {
            digitalWrite(buzzer, HIGH);
      }
      else {
            digitalWrite(buzzer, LOW);
      }
}

void navigation(int readedNumber) { //chování pro stanici nebo květinu
      if (readedNumber == 0) { //
            if (arrayWatered[0] == true && arrayWatered[1] == true) {
                  watered_all = true;
                  turnGreenLight(true);
            }
```

```
			if (getWaterLevel() == 0) {
				fillTank();
			}
			while (isBatteryLow()) {
				turnBuzzer(true);
				turnRedLight(true);

				delay(10000); //pokud je baterie názká, upozornit a vyčkávat po
15sek. blocích
			}
			lastPosition = readedNumber;

			return;
		}

		if (readedNumber > 0) { //chování pro květinu
			lastPosition = readedNumber;
			waterFlower(readedNumber);

		}

}



int findLowestNumber(int *arrangedMap) { //najdi nejmenší číslo z mapy (-
INFINITY,+INFINITY) = číslo křižovatky
		int lowestNum = arrangedMap[0];
		for (int i = 0; i < 4; i++) {
			if (arrangedMap[i] <= lowestNum) {
				lowestNum = arrangedMap[i];
			}
		}
		return lowestNum;
}

int findLargerstNegativeNumberIncludeZero(int *arrangedMap) { //najdi největší
číslo z mapy (-INFINITY,0>
		int iter = 0;
		int helpMap[4];
		for (int i = 0;i < 4;i++) {
			if (arrangedMap[i] <= 0) {
				helpMap[iter] = arrangedMap[i];
				iter++;
			}
		}

		int largestNum = helpMap[0];
		for (int j = 0; j < iter; j++) {
			if (helpMap[j] > largestNum) {
				largestNum = helpMap[j];
			}
		}
		return largestNum;
}

int findLargestNegativeNumber(int *arrangedMap) { //najdi největší číslo z mapy (-
INFINITY,0)
		int iter = 0;
```

```
	int helpMap[4];
	for (int i = 0;i < 4;i++) {
		if (arrangedMap[i] < 0) {
			helpMap[iter] = arrangedMap[i];
			iter++;
		}
	}

	int largestNum = helpMap[0];
	for (int j = 0; j < iter; j++) {
		if (helpMap[j] > largestNum) {
			largestNum = helpMap[j];
		}
	}
	return largestNum;
}

int findLowestPositiveNumberWhichNeedsToBeWatered(int *arrangedMap, int
readedNumber) { //najdi nejmenší číslo květiny, která potřebuje vodu (0,+INFINITY)
	int iter = 0;
	int helpMap[4];
	for (int i = 0; i < 4;i++) {
		if (arrangedMap[i] > 0) {
			helpMap[iter] = arrangedMap[i];
			iter++;
		}
	}
	int index;
	int count = 0;
	for (int j = 0; j < iter; j++) {
		index = helpMap[j] - 1;
		if (remainingWater[index] > 0) {
			count++;
			return helpMap[j];
		}
	}

	if (count == 0) {
		arrayWatered[abs(readedNumber) - 1] = true;
		return -1;
	}
}


//--------------------//
//ULTRASONIC SENSOR---//
//--------------------//


int getDistance() { //vrať hodnotu v centimetrech sejmutou ultrazvukem
	int duration, distance;
	digitalWrite(ul_trig, LOW);
	delayMicroseconds(2);
	digitalWrite(ul_trig, HIGH);
	delayMicroseconds(10);
	digitalWrite(ul_trig, LOW);
	duration = pulseIn(ul_echo, HIGH);
	distance = (duration / 2) / 29.1;
	return distance;
```

```
}

//--------------------//
//IR SENDER-----------//
//--------------------//

void sendMessage(String message) { //posílání signálu skrz IR vysílač
	for (int i = 0; i < message.length(); i++) {
		irsend.sendRC5(message.charAt(i), 12);
	}
}

//--------------------//
//MOVEMENT (PID)------//
//--------------------//

//nastavení hodnot PID regulátoru - pohyb vozidla
float Kp = 12, Ki = 0, Kd = 1;
float error = 0, P = 0, I = 0, D = 0, PID_value = 0;
float previous_error = 0, previous_I = 0;
int sensor[5] = { 0, 0, 0, 0, 0 };
int initial_motor_speed = normalSpeed;

void goPID() {
	readValues();
	calculatePID();
	setMotors();
}

void turnToNumber(int* arrangedMap, int numberOfPoint) { //otoč se k danému číslu
	int index = 0;
	for (int i = 0; i < 4; i++) {
		if (numberOfPoint == arrangedMap[i]) {
			index = arrangedMap[i];
			break;
		}
	}

	switch (index) { //statické nastavení - je nutné mít úhly cest u křižovatek
přesně 90 stupňů
	case 0: turn180();
	case 1: turn90Left();
	case 2: goStraight();
	case 3: turn90Right();
	}
}

void turn90Left() { //otáčení o 90 stupňů doleva
	digitalWrite(motorR_in1, LOW);
	digitalWrite(motorR_in2, HIGH);
	digitalWrite(motorL_in1, HIGH);
	digitalWrite(motorL_in2, LOW);
	analogWrite(motorR_sp, 255);
	analogWrite(motorL_sp, 125);

	delay(1100);
}

void turn90Right() { //otáčení o 90 stupňů doprava
```

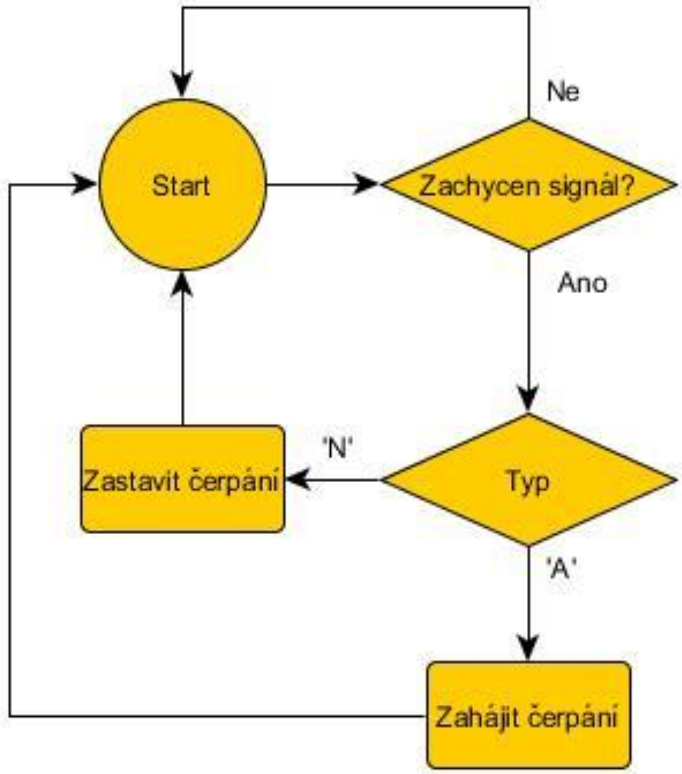
```
        digitalWrite(motorL_in1, LOW);
        digitalWrite(motorL_in2, HIGH);
        digitalWrite(motorR_in1, HIGH);
        digitalWrite(motorR_in2, LOW);

        analogWrite(motorL_sp, 255);
        analogWrite(motorR_sp, 100);
        delay(1000);
}

void turn180() { //otáčení o 180 stupňů - zpět
        digitalWrite(motorL_in1, LOW);
        digitalWrite(motorL_in2, HIGH);
        digitalWrite(motorR_in1, HIGH);
        digitalWrite(motorR_in2, LOW);
        analogWrite(motorL_sp, 255);
        analogWrite(motorR_sp, 255);
        delay(1400);
}

void goStraight() { //jízda rovně - 1 sekunda
        digitalWrite(motorL_in1, LOW);
        digitalWrite(motorL_in2, HIGH);
        digitalWrite(motorR_in1, LOW);
        digitalWrite(motorR_in2, HIGH);

        analogWrite(motorL_sp, 255);
        analogWrite(motorR_sp, 255);
        delay(800);

}

void stopMotors() {
        analogWrite(motorL_sp, 0);
        analogWrite(motorR_sp, 0);
        digitalWrite(motorL_in1, LOW);
        digitalWrite(motorL_in2, LOW);
        digitalWrite(motorR_in1, LOW);
        digitalWrite(motorR_in2, LOW);
}

void goBack() { //jízda zpět - 1 sekunda
        digitalWrite(motorL_in1, HIGH);
        digitalWrite(motorL_in2, LOW);
        digitalWrite(motorR_in1, HIGH);
        digitalWrite(motorR_in2, LOW);

        analogWrite(motorL_sp, 255);
        analogWrite(motorR_sp, 255);
        delay(800);

}

void readValues() { //přečtení sensorů a určení odchylky
        sensor[0] = digitalRead(path_1);
        sensor[1] = digitalRead(path_2);
        sensor[2] = digitalRead(path_3);
        sensor[3] = digitalRead(path_4);
        sensor[4] = digitalRead(path_5);
```

```
      if ((sensor[0] == 0) && (sensor[1] == 0) && (sensor[2] == 0) && (sensor[3]
== 0) && (sensor[4] == 1)) {
            error = 4;
      }
      else if ((sensor[0] == 0) && (sensor[1] == 0) && (sensor[2] == 0) &&
(sensor[3] == 1) && (sensor[4] == 1)) {
            error = 3;
      }
      else if ((sensor[0] == 0) && (sensor[1] == 0) && (sensor[2] == 1) &&
(sensor[3] == 1) && (sensor[4] == 1)) {
            error = 2;
      }
      else if ((sensor[0] == 0) && (sensor[1] == 0) && (sensor[2] == 0) &&
(sensor[3] == 1) && (sensor[4] == 0)) {
            error = 2;
      }
      else if ((sensor[0] == 0) && (sensor[1] == 0) && (sensor[2] == 1) &&
(sensor[3] == 1) && (sensor[4] == 0)) {
            error = 1;
      }
      else if ((sensor[0] == 0) && (sensor[1] == 0) && (sensor[2] == 1) &&
(sensor[3] == 0) && (sensor[4] == 0)) {
            error = 0;
      }
      else if ((sensor[0] == 0) && (sensor[1] == 1) && (sensor[2] == 1) &&
(sensor[3] == 1) && (sensor[4] == 0)) {
            error = 0;
      }
      else if ((sensor[0] == 0) && (sensor[1] == 1) && (sensor[2] == 1) &&
(sensor[3] == 0) && (sensor[4] == 0)) {
            error = -1;
      }
      else if ((sensor[0] == 0) && (sensor[1] == 1) && (sensor[2] == 0) &&
(sensor[3] == 0) && (sensor[4] == 0)) {
            error = -2;
      }
      else if ((sensor[0] == 1) && (sensor[1] == 1) && (sensor[2] == 1) &&
(sensor[3] == 0) && (sensor[4] == 0)) {
            error = -2;
      }
      else if ((sensor[0] == 1) && (sensor[1] == 1) && (sensor[2] == 0) &&
(sensor[3] == 0) && (sensor[4] == 0)) {
            error = -3;
      }
      else if ((sensor[0] == 1) && (sensor[1] == 0) && (sensor[2] == 0) &&
(sensor[3] == 0) && (sensor[4] == 0)) {
            error = -4;
      }
      else if ((sensor[0] == 0) && (sensor[1] == 0) && (sensor[2] == 0) &&
(sensor[3] == 0) && (sensor[4] == 0)) {

            if (error == -4) error = -5;
            else error = 5;
      }
}

void calculatePID() { //spočítání PID hodnoty
      P = error;
```

```
        I = I + error;
        D = error - previous_error;

        PID_value = (Kp*P) + (Ki*I) + (Kd*D);

        previous_I = I;
        previous_error = error;
}

void setMotors() { //na základě PID hodnoty upravit rychlost motorů
        int left_motor_speed = initial_motor_speed - PID_value;
        int right_motor_speed = initial_motor_speed + PID_value;


        if (left_motor_speed > 255) {
                left_motor_speed = 255;
        }
        if (right_motor_speed > 255) {
                right_motor_speed = 255;
        }

        if (left_motor_speed < 0) {
                left_motor_speed = 0;
        }
        if (right_motor_speed < 0) {
                right_motor_speed = 0;
        }

        digitalWrite(motorL_in1, LOW);
        digitalWrite(motorL_in2, HIGH);
        digitalWrite(motorR_in1, LOW);
        digitalWrite(motorR_in2, HIGH);

        if (error == abs(5)) {
                analogWrite(motorL_sp, 0);   //Left Motor Speed
                analogWrite(motorR_sp, 0);  //Right Motor Speed
        }
        else {
                analogWrite(motorL_sp, left_motor_speed);   //Left Motor Speed
                analogWrite(motorR_sp, right_motor_speed);  //Right Motor Speed
        }
}

//--------------------//
//WATERING------------//
//--------------------//

void waterFlower(int numberOfFlower) { //snížení čísla > index v poli
        numberOfFlower--;
        doWatering(remainingWater[numberOfFlower]);
}

void startWatering(void) { //zahájení zalévání
        analogWrite(dc_pump, dc_pump_speed);
}

void stopWatering(void) { //skončení zalévání
        analogWrite(dc_pump, 0);
}
```

```
void doWatering(int &mills) { //v parametru jsou mililitry
	//18ml/sekunda při 50% (Analog 123)
	int pauseByMillis = 50; //cykly po 50 ml
	int forcedDelay = (pauseByMillis * 1000) / 18; //trojclenka
	int delays = mills / 18;
	delays = delays * 1000; //jak dlouho v sekundach ma byt zaple cerpadlo
	startWatering();
	for (int i = 0; i < delays; i = i + forcedDelay) {
		if (getWaterLevel == 0) {
			stopWatering();
			break;
		}
		delay(forcedDelay);
		mills = mills - pauseByMillis;
	}

}

//--------------------//
//MANAGE WATER--------//
//--------------------//

void fillTank() { //naplnění zásobníku
	sendMessage("A");
	while (!(getWaterLevel() == 2)) {
		delay(20);
	}
	sendMessage("N");
}
```

Příloha č. 13


Ne
Start
Zachycen signál?
Ano
'N'
Zastavit čerpání
Typ
'A'
Zahájit čerpání

Příloha č.14

```
/*
 Popis: Zdrojový kód pro stanici
 Platforma: Funduino Uno (klon Arduino Uno)
*/

#include <IRremote.h> //knihovna na rozkódování IR signálu

int RECV_PIN = 11; //vstupní pin pro přijímač
int PUMP_PIN = 6; //výstupní pin pro ovládání čerpadla

IRrecv irrecv(RECV_PIN); //nastavení přijímače k určitému pinu
decode_results results; //signál

char START = 'A'; //znak pro zapnutí čerpadla
char STOP = 'N'; //znak pro vypnutí čerpadla

void setup() {
	irrecv.enableIRIn(); //povolení zahájení signálu
	pinMode(pumpPin, OUTPUT); //nastavení pinu k čerpadlu
}

void loop() {
	if (irrecv.decode(&results)) {
		if (char(results.value) == START) {
			irrecv.resume();
			startPump();
		}
		else if (char(results.value) == STOP) {
			irrecv.resume();
			stopPump();
		}
	}
}


void startPump() { //zapnutí čerpadla na poloviční výkon
	analogWrite(PUMP_PIN, 123);
}

void stopPump() { //vypnutí čerpadla
	analogWrite(PUMP_PIN, 0);
}
```

Příloha č. 15