Proteínový princ – ISTROBOT 2015

(Dokumentácia k tímovému projektu)

Vypracovali: Martin Ďurček, Jaromír Čukan, Tomáš Jurnický

Obsah

Úvod

Účasť tímu Proteínový princ na súťaži ISTROBOT 2015 je podmienená záujmom členov týmu o robotiku a chuťou pracovať na reálnom projekte a vlastnými rukami.

Po úvodnom uvažovaní sme sa rozhodli pre discpiplínu „V sklade kečupu“. Lákali nás aj disciplíny „Stopár“ a „Bludisko“ no po rozprave všetkých členov tímu sa súhlasilo no kečupoch. Hlavné fakty proti Stopárovi boli fádnosť a neoriginalita disciplíny, stokrát opakované podobné algoritmy všade po svete a celkový stereotyp disciplíny. Naopak „Bludisko“ nám prišla až moc softvérová disciplína, uvažovali sme, že po namotnovaní kamery alebo iného primárneho snímača by už nebola možnosť/potreba ďalšieho experimentovania. Na rozdiel od predchádzajúcich disciplín je „Sklad kečupu“ metóda, kde sa stretajú úplne odlišné stavby robotov, používajú sa úplne odlišné senzory a mechanizmy a celkovo sa nám zdala táto disciplína najzaujímavejšia a najzábavnejšia.

Hlavné body a charakter tímu je priblížený na našej webovej stránke
http://ap.urpi.fei.stuba.sk/protein/Motivation.html#

Počas semestra sme sa mnohému naučili, či už z technického alebo organizačného hladiska. Spoznali sme problémy, ktoré môžu nastať pri práci na projekte v tíme a snažili sme sa nájsť ich riešenie. Práca na projekte nás bavila a plánujeme pokračovať aj po konci semestra.

Metodika práce

Počas úvodných týždňov sa postupne prirodzene vykryštalizovali úlohy a schopnosti jednotlivých členov a tak sme oficiálne rozdelili úlohy v týme nasledovne:

- Martin Ďurček – supreme leader, vývoj softvéru a správca webstránky
- Jaromír Čukan – návrh architektúry a údržba robota, doplnkový vývoj softvéru
- Tomáš Jurnický – vývoj softvéru, doplnkové hárdverové úlohy

Počas úvodných týždňov sa pracovalo na vystavaní základného modelu robota. Bol použitý podvozok s viacmerovými kolieskami z bakalárskej práce Jara Čukana. Charakter viacsmerových kolies bol však vymenený za pásy podobné tanku. Popri práci do prvotnej podobe robota si členovia zaoberajúci sa vývojom softvéru brali domov rôzne snímače a zariadenia a snažili sa naučiť s nimi pracovať. Celý tím sa stretával počas celého týždňa na rôznych školských aj mimoškolských príležitostiach, takže bola téma neustále aktívne rozoberaná. Aj napriek tomu sme si zvolili oficiálnu schôdzku raz za týždeň – a to v utorok od 12:00 do 15:00 v budove D na ôsmom poschodí.

V neskoršom štádiu semestra si bral robota so sebou vždy len jeden „vývojár“ no komunikácia prebiehala denne, či už na báze osobného kontaktu alebo cez internet.

Na zdieľanie dokumentov sme zvolili najprv dynamicky sa rozvíjajúcu aplikáciu Slack, no po pár dňoch a problémoch s inštaláciou aplikácie do našich lokálnych počítačov sme zvolili ako komunikačný kanál na transfer súborov Dropbox.

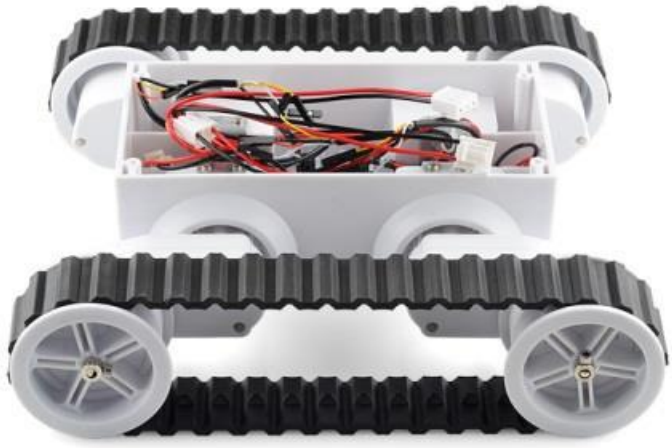
Chronologický prehľad postupu práce

1. Získanie viacsmerného podvozku od ÚRK FEI STU
2. Namontovanie pásov na kolesá
3. Montáž radlice
4. Vytvorenie web stránky
5. Namontovanie 7bodového snímača čiary
6. Vytvorenie algoritmu na 180-stupňové skenovanie okolia pomocou ultrazvukového snímača a serva
7. Výmena 8bodového snímača čiary za 3 samostatné bodové
8. Vytvorenie základných funkcií kompasu v arduine
9. Vytvorenie linefollower algoritmu
10. Výmena snímača čiary za 128b jednoriadkovú kameru (problém s rozoznaním T a X križovatky)
11. Vytvorenie a pripevnenie LED pásika pre lepšie osvetlenie terénu a spoľahlivejší výstup z kamery
12. Podbitie batérie, zakúpenie a objednanie nových dvoch batérií aj druhej nabíjačky
13. Vytvorenie regulátora uhla, pomocou ktorého sa robot vracia smerom naspäť na začiatočnú čiaru
14. Spätné namontovanie 8bodového infračerveného snímača
15. Nakrúcanie propagačného videa

Hardvér

Podvozok Rover 5

Jedná sa o štvorkolesový podvozok s rozmermi 245x225x74 mm, kde každé koleso má svoj vlastný elektromotor a optický snímač otáčok. Jedná sa o krúžok so štyrmi čiernymi a štyrmi bielymi pásikmi, a fotodiódou napájanou napätím 5 V. Snímač informuje o otáčkach aj o smere otáčania, neudáva však priamo otáčky kolesa, ale prevodového kolesa pred samotným hnaným kolesom. Frekvencia otáčok prevodového kolesa nesúhlasí s frekvenciou otáčok hnaného kolesa, tento pomer však výrobca neudáva. Výrobca udáva len celkový prevodový pomer ktorý je 87:1 (do pomala) a maximálnu rýchlosť ktorá je 1 km/h. Motory sú napájané napätím 7.2 V pri prúde 2.5 A. Podvozok je možné osadiť všemserovými kolesami alebo hladkými kolesami spojenými pásmi ako je zobrazené na obrázku. Teoreticky by pre pohon podvozku s pásmi stačilo ovládať na každej strane iba jeden motor. Avšak pásy nemajú vodiacu drážku, do ktorých by sa kolesá mohli zapierať. Pohyb kolies na pásy sa prenáša iba pomocou trenia a pokiaľ je poháňané iba jedno koleso na každej strane, kolesá občas prešmykujú. Testami sme zistili, že pre kvalitný pohyb podvozku s pásmi musia byť ovládané všetky štyri kolesá, čiže sa musia využívať všetky štyri motory.



Obr.1 – Podvozok Rover 5

Karta MD25

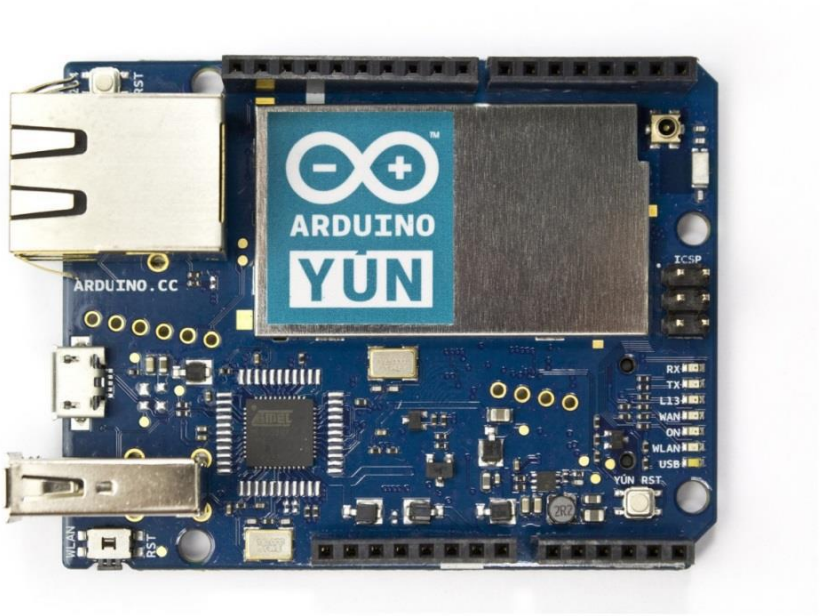
Karty MD25 slúžia na ovládanie motorov, jedna karta dokáže obslúžiť 2 motory. Vstupné dáta do kariet prichádzajú cez sériovú linku alebo I^2C zbernicu. Zbernica je adresovateľná a preto nie je problém pripojiť viac zariadení, v tomto prípade dve rovnaké MD25 karty s odlišnými adresami. Karty

sú napájané 7,2 V napätím a dokážu transformovať vstupné napätie na hodnotu 5 V, ktoré ďalej využívam na napájanie riadiacej karty Yún ako aj optických snímačov na motoroch. Rýchlosť a smer otáčania motora sa ovláda pomocou PWM modulácie. Princíp PWM modulácie spočíva v nastavovaní dĺžky impulzu v rámci pevne danej periódy, t.j. mení sa strieda impulzu. Rýchlosť i smer sa riadi zápisom dát, jedného bajtu do registra 0 pre prvý motor, respektíve registra 1 pre druhý motor na karte. Decimálna hodnota bajtu 128 zodpovedá zastaveniu motorov a krajné hodnoty 0 a 255 zodpovedajú maximálnym rýchlostiam v rozdielnych smeroch. I keby sa zdalo, že je k dispozícii 127 možných rýchlostí na obe strany otáčania, nie je to tak. Pri nízkych hodnotách v rozmedzí 129-133, respektíve 123-127, t.j. ±4 jednotky od pomyslenej nuly sa elektromotory správajú ako krokové motory. Nevykonávajú plynulý pohyb. Za najpomalšiu rýchlosť som teda zvolil hodnoty 135, respektíve 121, t.j. ±7 jednotiek od 128. Na obmedzenia som narazil i z druhej strany stupnice rýchlostí. Rozlišovacia schopnosť motorov končí pri hodnote pod 58, resp. nad 198 t.j. ±70 jednotiek od 128. Vyššie hodnoty sú už motormi ignorované. Karta samotná je schopná generovať signály PWM z celého rozsahu 0-255, ale obmedzenia spôsobujú samotné motory. Karta disponuje i vlastnými regulátormi rýchlostí. Tie prijímajú údaje z optických sond a akčnými zásahmi regulujú riadiacu PWM moduláciu.

Obr.2 – Karta MD25

Karta Arduino Yún

Karta Yún je prvým produktom skupiny Arduino, ktorá ma v sebe plne implementovanú bezdrôtovú technológiu wifi. Pomenovanie Yún pochádza z čínštiny a znamená oblak. Práve oblak má symbolizovať spojenie s internetom a cloud úložiskami.



Obr.3 – Arduino Yún

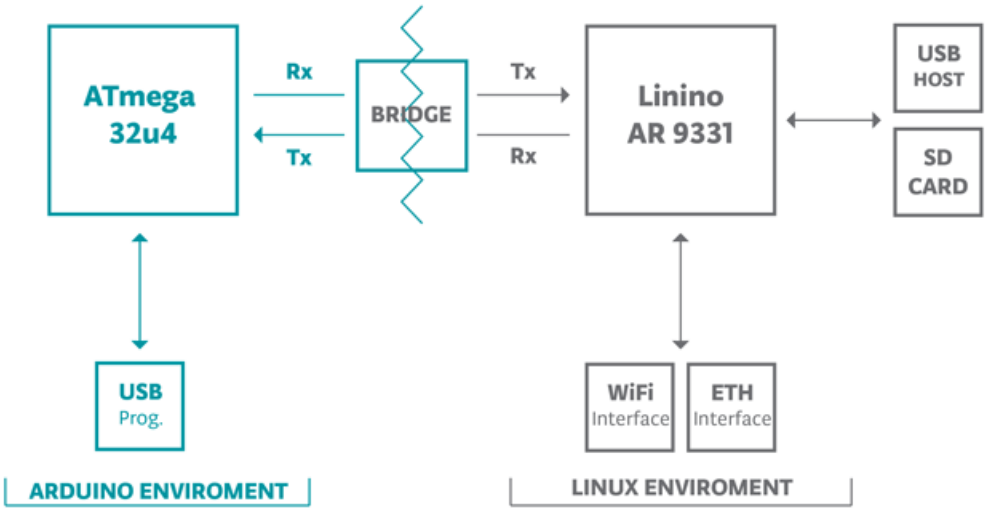
Riadiaca karta je napájaná 5 V a je schopná pretransformovať vstupné napätie na ďalej využiteľné napätie 3,3 V. K dispozícii je 20 programovateľných vstupov / výstupov.

Karta je osadená dvoma procesormi. Menší ATmega32u4 obsluhuje jednotlivé analógové a digitálne vstupy / výstupy a je programovateľný cez micro USB. Výkonnejší Atheros AR9331 sa stará o komunikáciu s okolím prostredníctvom wifi rozhrania, USB alebo Ethernet portu. Pamäť systému je možné rozšíriť o microSD kartu, prístupnú z procesora Atheros. V nasledujúcej tabuľke prebranej z oficiálnej stránky výrobcu sú uvedené základné parametre oboch procesorov.

Microcontroller	**ATmega32u4**
Digital I/O pins	20
Flash memory	32 KB
SRAM	2,5 KB
EEPROM	1 KB
Processor	**Atheros AR9331**
Architecture	MIPS @400 Mhz
Ethernet	IEEE 802.3 10/100 Mbit/s
WiFi	IEEE 802.11 b/g/n
USB Type-A	2.0 Host/Device
Card reader	microSD only
RAM	64 MB DDR2
Flash	16 MB

Procesory sú navzájom prepojené a komunikujú pomocou knižnice Bridge.h vytvorenej výrobcom. Zatiaľ čo ATmega32u4 je programovateľný cez viac menej štandardný C jazyk v prostredí vyvinutom

združením Arduino, Atheros beží na distribúcii Linuxu nazývanej Linino, ktorá je programovateľná v jazyku Python.



Obr.4 – Štruktúra Arduina Yún

Zbernica I^2C

Zbernica, vyvinutá spoločnosťou Philips, je štandardom pre nízko rýchlostnú komunikáciu medzi perifériami a riadiacim obvodom. Jej výhodou je adresovateľnosť, môže obslúžiť teoreticky 112 klientov a fakt, že k správnej funkcii stačia dva vodiča. V jednom sú z hlavného zariadenia (ďalej len „master“) do podriadeného zariadenia (ďalej len „slave“) vysielané samotné údaje (SDA linka). V druhom vodiči je vysielaný hodinový signál (SCL linka) potrebný pre synchronizáciu pre správne interpretovanie údajov. Každá stanica pripojená na zbernicu má pridelenú 7 alebo 10 bitovú adresu v závislosti na použitom zariadení. Master najskôr vyšle adresu stanice, na ktorú sa prepne stanica so súhlasnou adresou do režimu načúvania. Potom sú vysielané master dáta. V mojom prípade sa vysiela najskôr bajt s číslom registra do ktorého chcem zapisovať a následne samotný bajt s hodnotou rýchlosti. Komunikácia sa ukončí odoslaním podmienky stop. Štandardná rýchlosť zbernice je 100 kbit/s. Nakoľko integrované obvody s podporou I2C používajú na výstupe tranzistory s otvoreným kolektorom, je obvykle potrebné doplniť medzi linku na napájanie tzv. „Pull-up“ rezistory. Bez týchto rezistorov je linka nespoľahlivá alebo nefunguje vôbec. V našom prípade karty MD25 bez pull-up rezistorov vôbec nefungovali a preto sme použil pull-up rezistory s hodnotami 8,2 kΩ.

Zdroj energie

Ako zdroj energie predpisuje výrobca 8 ks batérií veľkosti AA. V priestore pre batérie je však množstvo káblov, takže batérie sa tam zmestia len veľmi ťažko. Ich výmena je potom veľmi problematická. Naviac majú malú kapacitu, čo ďalej komplikuje situáciu. Pôvodne sme na napájanie použili akumulátor typu Li-pol s kapacitou 1300 mAh a napätím 7,4V. Avšak častým testovaním

robota a prácou s ním ani táto kapacita nebola postačujúca. Zaobstarali sme teda podobnú 2 článkovú Li-pol batériu s napätím 7,4V no so zvýšenou kapacitou až 2100 mAh.

Obr.5 – Nami používaná Li-pol batéria

Konštrukcia robota

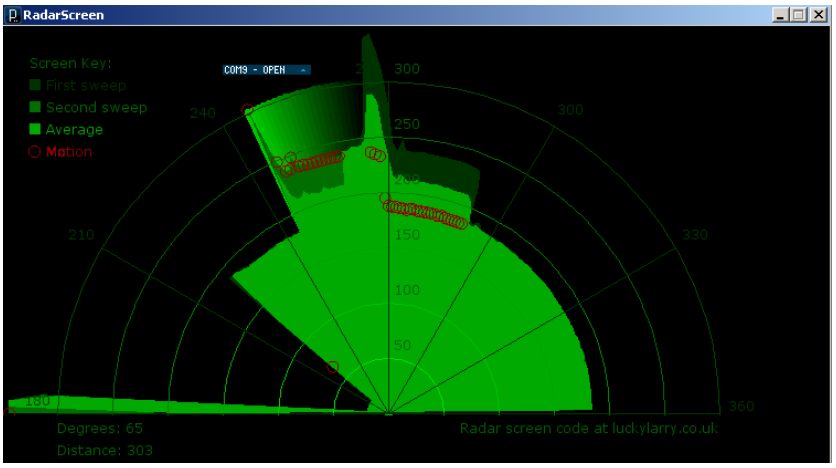
Na podvozok osadení riadiacimi kartami MD25 a ovládacou kartou Arduino Yún sme pripevnili pomocou štyroch skrutiek konštrukciu zo stavebnice Merkur. Vytvorili sme akoby druhé poschodie, kde sme umiesnili breadboard. Ďalšou výraznou časťou konštrukcie je predná radlica, ktorá je dostatočne veľká na nabratie dvoch plechoviek. Zvolili sme stratégiu zobrať dve plechovky naraz na domovskú čiaru a ušetriť tým čas, pretože nám to výkon podvozku aj jeho rozmery dovoľujú. V prednej časti radlice je namontovaná závora, ktorá bude spustená dolu iba pri otáčaní robota, aby sa zabránilo vypadnutiu plechovky. O rozpoznávanie plechoviek a prípadnú detekciu priestoru pred robotom, a zabráneniu zrážke sa stará ultrazvuk nainštalovaní nad radlicou. Rozmery podvozku spolu s prednou radlicou sa približujú maximálne povoleným rozmerom robota v súťaži, no spĺňajú ich s rezervou niekoľko milimetrov.

Práca na perifériách a senzoroch robota

Radar

Na vytvorenie radaru, ktorý plánujeme použiť na zisťovanie pozície plechoviek sme použili ultrazvukový senzor PING))) od Parallax a Servo TowerPro SG-5010. Algoritmus bol relatívne jednoduchý. Po nastavení určitého kroku sme vždy vyžiadali informáciu z ultrazvukového seznoru. Najvačší problem tu nastal na začiatku práce – používali sme knižnicu Servo.h, ktorá obsahuje funkciu Servo.write(). Argument tejto funkcie je číslo, ktoré v závoslosti od typu serva hovorí buď o rýchlosti kontinuálneho otáčania alebo o uhle, na ktorý sa má servo nastaviť. Keďže sme doteraz pracovali len so servom, ktoré chápalo argument ako rýchlosť, no TowerPro SG-5010 argument chápe ako uhol, bol zozačiatku problém s jeho „nepochopiteľným správaním“. Najlepšie radar fungoval pri skoku uhla o 5 stupňov.

Vyskúšali sme aj prácu s programom RadarScreen uvedenom na stránke ku predmetu Robotika. Program fungoval čiastočne – komunikácia fungovala a vzdialenosť aj momentálny uhol sa vypisoval, no nefungovala vizualizácia vzdialenosti predmetov.



Obr.6 – Aplikácia RadarScreen

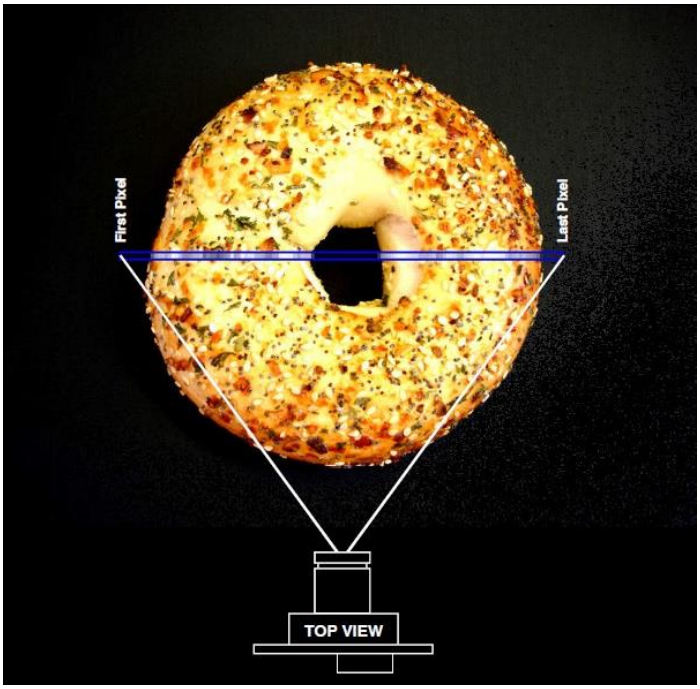
Ukážka kódu:

```
for(int i=0;i<=180;i=i+5){
  RadarServo.write(i);
  ReadDistance(i);
  if(i!=180){
   delay(200);}}
```

Riadková kamera TSL 1401-DB

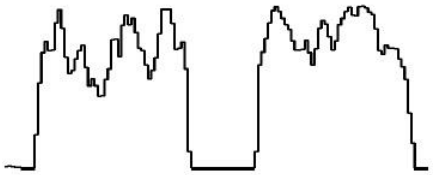
Táto kamera je schopná v jednom rozmere snímat **128 pixelov**. Potrebné napájacie napätie je 3V3 alebo 5V. Celkové zorné pole kamery so 7,9mm objektívom sa rovná vzdialenosti od objektu. To znamená, že v prípade vzdialenosti kamery jeden meter od objektu, bude kamera snímať jeden meter široký pas. Najčastejšie aplikácie tejto kamery sú:

- Zmeranie výšky, šírky, hrúbky
- Vyhľadanie objektov ako čiary, hrany, medzery, diery.
- Prečítania jednoduchých čiarových kódov



Obr. 7 - Príklad snímania TSL1401R modulu

Výstup z každého nasnímaného pixelu je analógové napätie úmerné intenzite sveta. Nasledujúci obrázok znázorňuje príklad nasnímanej intenzity svetla z Obr.1.



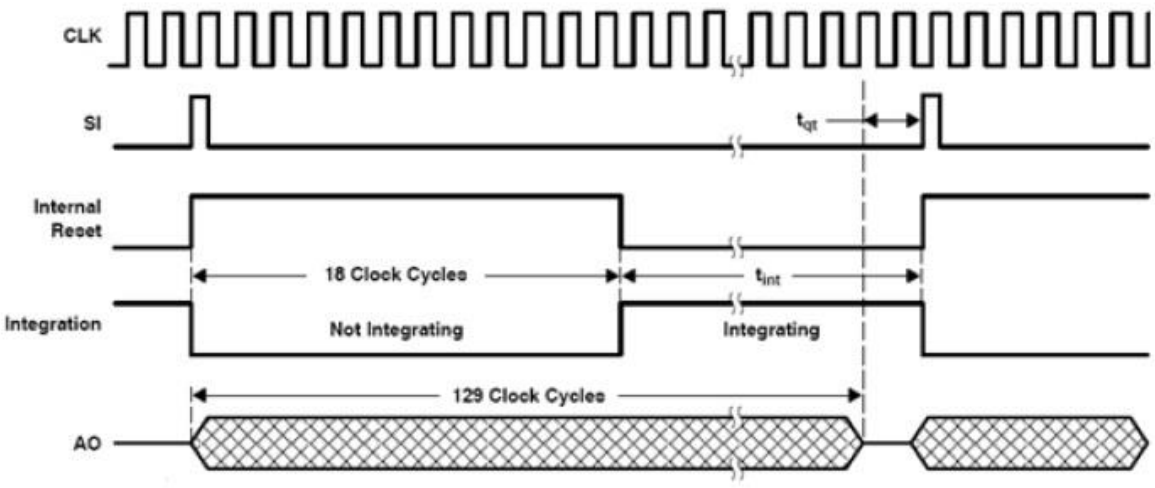
Obr. 8 - Krivka intenzity svetla nasnímanej kamerou

Zaostrovanie objektu sa vykonáva zaskrutkovaním alebo odskrutkovaním objektívu. Ak chceme zaostriť bližšie predmety musíme šošovku vyskrutkovať.

Pre funkčnosť kamery je potrebné zapojiť 5 pinov

- Ucc(3V3 alebo 5V)
- GND
- CLK(clock)
- SI (serial input)
- A0 (analog output)

Kde piny **SI** a **CLK** sú digitálne vstupy kamery a pin **A0** je analógový výstup kamery. Úpravou frekvencie generovania signálov **CLK** a **SI** vieme zrýchliť alebo spomaliť proces. Príklad komunikácie s kamerou je znázornený na nasledujúcom obrázku.



Obr. 9 - Príklad komunikácie s kamerou

Signály **CLK, SI, A0** musia byť synchronizované tak ako znázorňuje obrázok komunikácie. Pre správne fungovanie kamery musia mať hodiny **CLK** a impulz **SI** rovnakú šírku. Analógový signál **SI** je začiatočný impulz, ktorým začína čítanie z kamery.

Naša aplikácia kamery

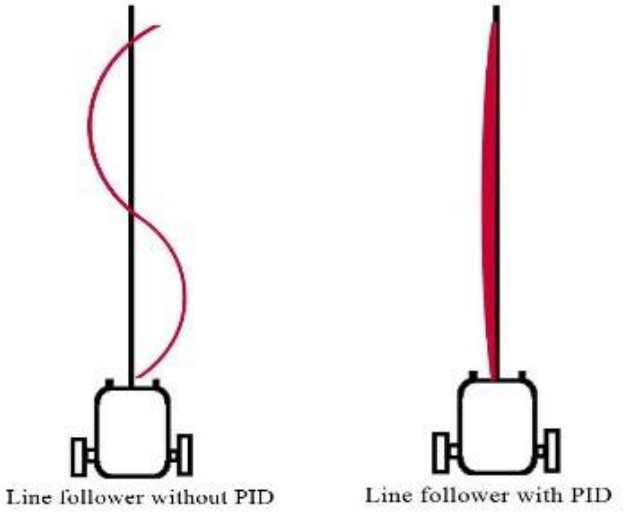
Túto kameru sme spočiatku využili na sledovanie čiernej čiary. Veľkou výhodou kamery je, že poskytuje až 128 pixelov resp. ako keby „senzorov“ na sledovanie čiary. Čo umožňuje ľahké sledovanie čiary a zaručene presné určenie križovatky. Šírku impulzov sme nastavili na dve mikrosekundy. Nevýhodou použitia kamery pre sledovanie čiary je, že závisí od svetelných podmienok. Z tohto dôvodu bolo potrebné vyrobiť led pásik, ktorý by zaručil dobré svetelné podmienky. Ďalšou nevýhodou je, že spomaľuje hlavné main vlákno programu. Keďže účelom nášho robota nie je iba sledovanie čiary, ale aj aplikovanie ostatných algoritmov na hľadanie plechovky a navigáciu robota, rozhodli sme sa zmeniť stratégiu a využiť infračervené snímače čiary.

Obr. 10 - Naša aplikácia led osvetlenia pre kameru

Sledovanie čiary (line-follower)

V súčasnosti na sledovanie čiary a detekovanie križovatky využívame osem infračervených senzorov. Tieto senzory sú zapojené na digitálne piny, čiže nám dávajú digitálnu hodnotu 0/1. Ak sa senzor nachádza nad čiarou jeho hodnota je jedna.

Na sledovanie čiary využívame **PID algoritmus s anti-windup efektom**. PID je široko používaný algoritmus, ktorý sa väčšinou používa v priemysle, robotike a ďalších oblastiach. Tento algoritmus nám zaručí plynulé sledovanie čiary oproti riešeniu line followera iba pomocou podmienok. Porovnanie výsledkov je znázornené obrázkom.



Obr. 11 - Porovnanie výsledkov sledovania čiary

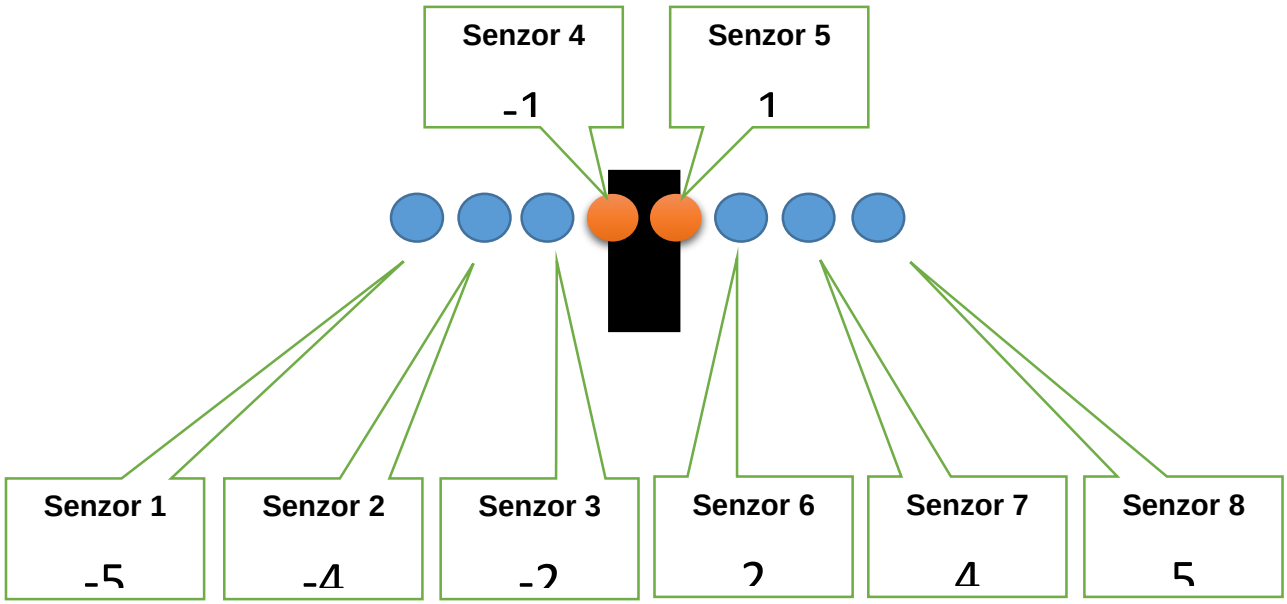
Nastavenie PID regulátora:

Set-Point

Ako žiadanú hodnotu (Set-Point) nastavíme pozíciu kedy je robot dokonale na čiare. V našom prípade sme žiadanú hodnotu nastavili na nula. Táto hodnota je fixná a nemôže sa stať aby počas vykonávania programu bola zmenená.

Process-Variable

Hodnotu PV vypočítame z aktuálnej polohy robota resp. infračervených senzorov nasledovne:



Príklad výpočtu hodnoty PV podľa obrázka:

***PV**=((-5*s1)+(-4*s2)+(-2*s3)+(-1*s4)+(1*s5)+(2*s6)+(4*s7)+(5*s8));*

Ak senzor 4 aj senzor 5 sú na čiare potom oba majú hodnotu jedna a ostatné dávajú hodnotu nula potom:

PV**=((-5*0)+(-4*0)+(-2*0)+(-11**)+(1***1**)+(2*0)+(4*0)+(5*0));*

***PV**=0*

Dostali sme výsledok nula čo znamená, že robot je na želanej hodnote a regulačná odchýlka by bola nulová.

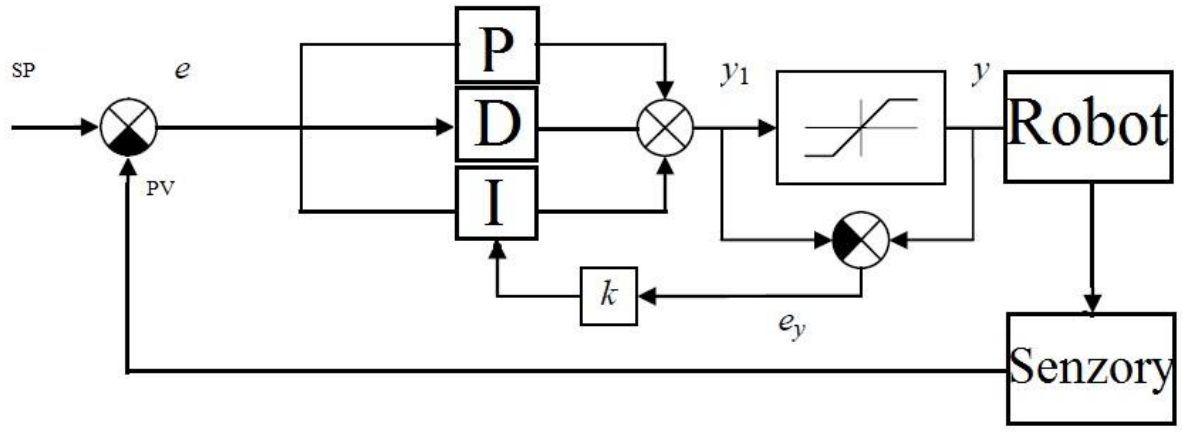
Vzorec pre výpočet PID

$$\mathbf{u(t)} = \mathbf{k_p} e(t) + \mathbf{k_i} \int_0^t e(t) + \mathbf{k_d} \frac{d}{dt} e(t)$$

Konštanty **kp, ki, kd** sme nastavili experimentálnou metódou.

Ovládanie motorov:

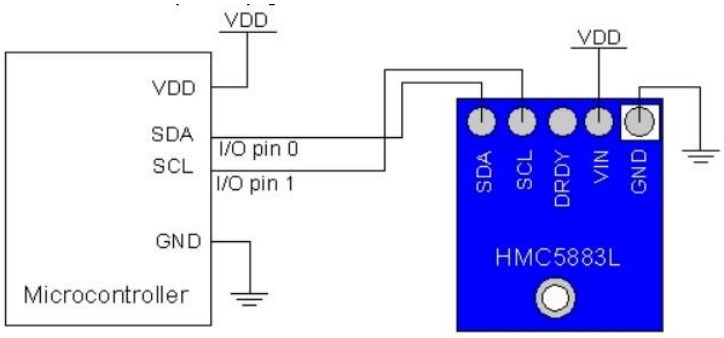
Motory sú ovládané pomocou spomínaných MD25 kariet, ktoré komunikujú s arduino YUN doskou cez I2C zbernicu. Ak sa danému motoru pošle hodnota 128 tak sa zastaví. Ak sa pošle hodnota 68 motory pôjdu na plný výkon, to znamená že akčná veličina by nemala presiahnuť +70 alebo -70 (ak by išiel dozadu). Aby sme predišli tomu, že by sa motory začali točiť do opačnej strany je potrebné zavrieť obmedzenie akčnej veličiny. Akčná veličina sa začne meniť až potom, keď sa dostane medzi hranice obmedzenia. Toto nám vnieslo do nášho regulačného obvodu oneskorenie a tým zhoršenie kvality riadenia. Vyriešením tejto situácie bolo vyradenie spätnej väzby, ak sa akčná veličina dostane mimo obmedzenie – zavedenie anti-windup efektu. Hodnotu parametra „k“ sme nastavili experimentálnou metódou.



Obr. 12- PID algoritmus

Kompas HMC5883L

Kompas HMC5883L je trojosí magnetometer. Pre funkčnosť kompasu musíme zapojiť štyri konektory. Dva napájacie konektory a dva konektory pre komunikáciu s kompasom. Komunikácia prebieha pomocou I2C zbernice. SDA sú dáta a SCL hodiny. Pri zapojení kompasu nepotrebuje pull-up odpory, pretože sú v ňom už integrované. Napájacie napätie kompasu je 3V3 alebo 5V. Kompas podporuje rýchlosti 100kHz a 400kHz – nepodporuje fast-speed mód (3,4GHz).



Obr. 13- Zapojenie kompasu HMC5883L

Na prácu s týmto kompasom sme využili zadarmo stiahnuteľné knižnice tretích strán pre Arduino z GitHubu: https://github.com/jarzebski/Arduino-HMC5883L. Po zapojení kompas neukazoval správne hodnoty natočenia. Kompas potreboval kalibráciu.

Hodnoty získané z kompasu bez kalibrácie:

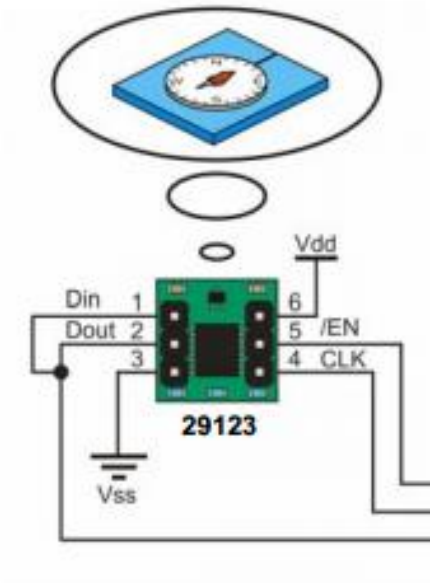
Fyzické natočenie kompasu	Odčítaná hodnota
0°	0°
90°	62°
180°	218°
270°	259°
360°	0,45°

Kalibrácia prebiehala nasledovne. Kompasom bolo potrebné hýbať na všetky strany a točiť sním, aby sme získali najväčšie a najmenšie hodnoty z kompasu resp. aby sme získali hraničné hodnoty z magnetometra. Tieto hodnoty prepočítame na úplný rozsah a na základe toho vypočítame správny uhol. No napriek tomu, po kalibrácii kompas stále ukazoval odchýlku až do 10°. Preto sme sa rozhodli použiť jednoduchší kompas HM55B, ktorý dával presnejšie hodnoty.

Kompas HM55B

Po úvodných problémoch sme sa rozhodli pri práci s uhlom a orientáciou robota použiť kompas HM55B od Parallaxu. Je to dvojosí senzor magnetického poľa. Je kompatibilný pre 3 voltové ako aj 5 voltové napájanie. Je relatívne rýchly, keďže na inicializáciu do prvého merania potrebuje len cca 40 milisekúnd.

Jeho zapojenie je jednoduché, po pripojení na zem a do napájania ostáva zapojit zvyšné 4 piny do 3 arduino pinov.



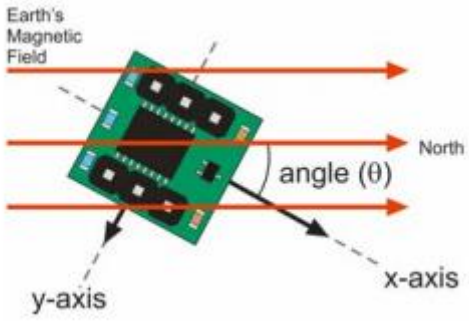
Obr.14 – Piny kompasu

Piny:

- 1 (Din) – Serial data input
- 2(Dout) – Serial data output
- 3 (Vss) - zem
- 4 (CLK) – hodiny
- 5 (/EN) – active-low device enable
- 6 (Vcc) – 5V napájanie

Po použití a miernej úprave vzorového kódu nájdeného na internete bolo možné vyupočítať uhol otočenia podľa nasledujúceho vzorca

$$uhol = \arctan(\frac{-y}{x})$$



Obr. 15 – Výpočet uhla natočenia

Celý zdrojový kód ku kompasu možno nájsť medzi zdrojovými kódmi, tu spomíname len najdôležitejšie funkcie:

- HM55B_ReadCommand() – vyžiadanie dát z kompasu
- X_Data = ShiftIn(11) – sila poľa v smere x
- Y_Data = ShiftIn(11) – sila poľa v smere y

Regulátor uhla

Na vytvorenie regulátora uhla bol ako vzor použitý kód na PID regulator na ovládanie motorov pri sledovaní čiary. Na začiatku celého programu sa odčíta uhol a zistí sa hodnota o 180 stupňov opačná – treba dať pozor aby uhol neprekročil 360.

```
If (initialAngle > 0)
        sp = initialAngle - 180;
else
sp = initialAngle + 180;
```

Výpočet akčnej veličiny, kde kp, ki a kd sú konštanty regulátora:

```
u = kp * e + ki * integracna + kd * derivacna;
```

Aplikácia akčnej veličiny na motory (základná, akčný zásah je ešte ručne trochu pri určitých podmienkach upravovaný):

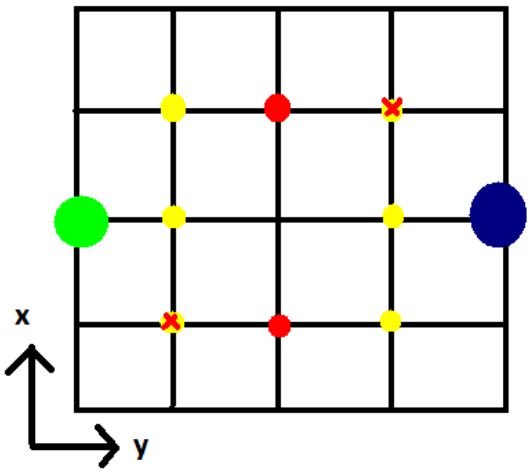
```
if((u<0)){
        pohyb(90,90-u);
}
else if((u>0)){
        pohyb(90+u,90);
}
else if((u==0)){
        pohyb(90,90);
}
```

Všeobecná navigácia

Začali sme pracovať aj na kóde všeobecnej navigácie robota v sklade. Na tej chceme pokračovať aj v budúcnosti, hneď po vyriešení pretrvávajúcich problémov s odlišovaním T a X križovatky. Na fungovanie tohoto algoritmu bude treba takisto aj namontovať ultrazvukový „radar“ vytvorený v prvej časti práce na robotovi. Ten po zaregistrovaní plechovky odošle dáta do navigácie a vďaka rozoznaniu čiar a križovatiek pri pohybe bude robot schopný prijsť k cieľu a neustále vedieť kde sa nachádza. To je potrebné aj pri návrate na základnú čiaru pomocou regulátora uhla – musí vedieť koľko čiar má prejsť kým zastaví.

Navigation (int MyX, int MyY,int FinX, int FinY) – vstupné sú 4 parametre (aktuálne a želané súradnice)

Na začiatku sa robot bude snažiť dostať na úroveň plechovky na ypsilonovej úrovni – bude sa pohybovať po čiare x. Zistí, či je želaná úroveň xovej súradnice väčšia alebo menšia ako aktuálna a podľa toho sa otočí doľava alebo doprava. Ďalej sa bude hýbať po čiare pomocou PID regulátora a zaznamenávať počet prekročených T križovaniek (keďže novú plechovku bude hľadať po prinesení starej, sme si istý, že križovatky budú mať charakter T). Počet potrebných prekročených čiar je absolútna hodnota rozdielu prvotnej a žiadanej pozície. V momente, keď robot dosiahne žiadaný počet prekročení, zastaví a otočí sa smerom do skladu. Potom vykoná podobnú logiku aj smerom po ypsilonovej čiare (sledovanie čiary a zaznamenávanie prekročených križovatiek). Keď robot zoberie plechovku, ideálne by bolo, keby sa naspäť dokázal vrátiť znova podľa PID regulátora uhla.



Obr.16 – Orientácia v sklade

Záver

Počas práce na tomto projekte sme sa veľa naučili. Museli sme pracovať v tíme a okrem riešenia technických problémov, na ktoré sme zvyknutí aj z iných predmetov sme sa museli naučiť komunikovať, hľadať kompromisy a prispôsovať sa jeden druhému. Nie vždy to bolo lahké, no prácu na tímovom projekte sme si vybrali práve preto. Niektorí z nás videli stavbu robotického zariadenia od úplnej nuly vôber prvýkrát vživote, ba čo viac, sami sa na nej podielali.

Práca na projekte nás bavila a v relatívne krátkom čase sme sa toho mnoho naučili. Napriek tomu sme však projekt na záverečnú súťaž ISTROBOT pripraviť nestihli. Plánujeme však na robotovi ďalej pracovať tak, aby sme na súťaži predstavili najlepšiu a najfunkčnejšiu verziu akú sme schopní v danom čase zvládnuť.

Na záver sa chceme poďakovať prof. Richardovi Balgohovi, za odborné vedenie, poskytnuté konzultácie ako aj zapožičanie veľkej väčšiny potrebných komponentov na výstavbu robota.