



**УНИВЕРЗИТЕТ „СВ. КИРИЛ И МЕТОДИЈ“ ВО
СКОПЈЕ**



**МАШИНСКИ ФАКУЛТЕТ – СКОПЈЕ
ПРВ ЦИКЛУС ЧЕТИРИГОДИШНИ СТУДИИ**

Марко Галевски

**Реализација на Автономно Возило-Робот со Функција
за Следење на Човек**

ДИПЛОМСКА РАБОТА

СКОПЈЕ, 2018

Ментор:

Проф. д-р Виктор Гаврилоски
Машински факултет – Скопје

Членови на комисијата:

Проф. д-р Виктор Гаврилоски
Машински факултет – Скопје

Проф. д-р Златко Петрески
Машински факултет – Скопје

Доц. Д-р Јована Јованова
Машински факултет – Скопје

Дата на одбрана:

29.06.2018

Марко Галевски

Реализација на Автономно Возило-Робот со Функција за Следење на Човек

АПСТРАКТ:

Предмет на оваа дипломска работа е изработката на автономен возило-робот со способност да препознае повеќе личности и да одлучи која од нив да ја следи според претходно зададени критериуми. Целта е да се постигне робустно решение со висока брзина на препознавање и следење.

Постигнати се речиси сите цели, но значителна мана се појави во тоа што употребениот алгоритам за препознавање има одредени пропусти при задвижување на Kinect камерата.

Идна работа на темата би вклучило испитување на алтернативни, поробустни решенија за човеково препознавање – дали тие начини би барале нови сензори или нови алгоритми.

КЛУЧНИ ЗБОРОВИ:

Мобилна Роботика, Машински Вид, LabVIEW, Kinect, Човеково Препознавање

Marko Galevski

Realisation of an Autonomous Vehicle/Robot with Human Following Functionality

ABSTRACT:

The subject of this diplom thesis is the realisation of an autonomous robot/vehicle with the capability to recognise multiple people and to decide which of them to follow based on previously provided criteria. The goal is to achieve a robust solution with a high speed of following and detection.

The goals were by and large achieved, although a significant flaw arose in the detection algorithm - which would fail upon significant motion of the Kinect.

Future work on the subject matter would include the inspection of alternative, more robust solutions for human recognition – whether those methods require new sensors or new algorithms.

KEY WORDS:

Mobile Robotics, Machine Vision, LabVIEW, Kinect, Human Recognition

Содржина

1 Вовед	1
2 Kinect v2 за Windows	2
2.1 Time of Flight камери	2
2.1.1 Директен ToF	2
2.1.2 Имплицитен ToF	3
2.2 Споредба и Анализа на Kinect-от	4
3 Примери од SDK-то на Kinect-от	7
3.1 Начин на отчитување во C++	7
3.2 Примери од SDK-то	8
4 DaNI Роботот	9
4.1 Актуатори	9
4.1.1 DC мотори	9
4.1.2 Анализа на измерени вредности при движење	12
4.1.3 Серво мотор	14
4.1.4 PWM	14
4.2 Сензори	15
4.2.1 Енкодер	16
4.2.2 Ултразвучен сензор	17
4.3 NI sbRIO 9632	19
4.3.1 FPGA	19
5 Теоретска Позадина	21
5.1 Генерален Опис на Системот	21
5.2 Блок Дијаграм на Системот	22
5.3 Мобилна Роботика	23
5.3.1 Холономност и диференцијален погон	23
5.3.2 Вектор модел и ПД управување	24
5.4 Теорија на Боја за Компјутерски Системи	25
6 Софтверско управување со LabVIEW	26
6.1 Robotics модул: Starter Kit 2.0	26
6.2 LabVIEW MakerHub модули	28
6.2.1 Kinect модул	28
6.2.2 PS4 контролер модул	29
6.3 Специјално изработени Виртуелни Инструменти и Контроли	31
6.3.1 humanDetect	31
6.3.2 colourDist	31
6.3.3 differentialDrive	34
6.3.4 Телеоперација	34
6.3.5 TypeDefs	34
6.4 Образложение на Софтверот	36
7 Заклучок	45
Користена литература	50

1 Вовед

Индустријата бурно и брзо има претрпено големи промени во минатата деценија, и се очекува дека таа брзина на промена сеуште ќе се зголемува во наредните години. Два фактори кои се дел од причината за новата индустриска револуција се напредувањата во сензорската и процесорската технологија.

Подобрувањето во сензорската технологија доведува до две главни придобивки: пософистицирани сензори и драстично поевтини сензори. Овие два факти се очигледни кај главниот сензорски склоп на кој што се потпираа оваа дипломска работа - Kinect-от од Microsoft. Сензорот спојува обична бојна камера (RGB) со интегрирана инфрацрвена матрица, односно длабинска камера (D), создавувајќи еден софистициран композитен сензор (RGB-D Камера) чија технологија поседува голем потенцијал за примена во роботика, оптимизација на производни процеси, безбедносната индустриска, а дури и во автономните возила и медицинската индустриска.

Ова унапредување во технички можности не е само поради сензорскиот развој, туку има тука и една голема улога подобрувањето на сериската комуникација - односно USB комуникацијата. Суровата информација од самата камера е прилично обемна. Прецисно кажано, за секоја слика $2,4 \times 10^6$ пиксели требат да стигнат од камерата до компјутерот, $2,1 \times 10^6$ од кои што се 4-бајтни вредности (YUYV формат), додека останатите 300.000 се 1-бајтни вредности. Ова изнесува $8,6 \times 10^6$ байти, или 8,6MB по слика. Со фреквенција на семплирање од 30Hz, брзината на пренос на информација е 258MB/s. Оваа брзина не би била ни замислива пред 30 години, но денес е секојдневие со USB 3.0.

Иако е завршено пренесувањето на информација, од тука следи усогласувањето на сликите од двете засебни камери (*matching*) и, во наш случај, наоѓањето на 1-6 луѓе и нивното симултано положбено следење преку 26 зглобни врски. Тоа што се потенцира со овие пресметки и примери е фактот дека да не беше за денешните пресметковни можности, идеите како оваа дипломска работа би останеле само на идеја.

Темата на оваа дипломска работа е робот кој што може да препознае една или повеќе личности, и според некој критериум/и да одлучи кого (или никој) да го следи. Главниот дел од оваа дипломска работа ќе биде составен од образложение на управувањето во LabVIEW на еден модифициран робот од National Instruments опремен со Kinect v2 камера. Целта на управувањето е да се постигне робустно и надежно следење на човек, одржувајќи го растојанието зададено од човекот преку гестикулации на рацете.

2 Kinect v2 за Windows

Како што е описано во [20], Kinect-от од Microsoft претставува композитен сензор составен од една обична RGB (*Red-Green-Blue*) камера и една матрица на инфрацрвени предаватели/приемници, која служи како D (*Depth* - Длабинска) камера. Оваа D камера се нарекува и Time of Flight (ToF) камера, односно камера што го пресметува времето на изминатиот пат за секој од нејзините оддадени фотони. Заедно создават еден уред кој што може да се смета концептуално како една RGB-D камера. Kinect-от ја игра улогата на главен сензор во оваа дипломска; преку алгоритми создадени од машинско учење и преку фузија на двата сензори може да ја препознае скелетната положба на 6 луѓе истовремено, како и отвореноста/затвореноста на нивните раце.

Првата верзија на Kinect-от (v1) била создадена од Microsoft во 2010г. како периферен уред за нивниот систем за видео игри - Xbox 360. Првиот Kinect има способност да следи само 2 скелетни положби, и не може да разликува отворени/затворени раце. Во 2013г., Microsoft ја изваде Kinect v2, која ги нуди способностите наведени погоре. Иако беше наменет како уред за играње и забава, Kinect-от брзо стана клучен дел од репетоарот на инженерите кои развиваат системи за машински вид поради неговата поволна цена и готови библиотеки. На крајот на ова поглавје е прикачена споредбена табела на двете верзии на Kinect-от.

2.1 Time of Flight камери

Time of Flight (ToF) технологиите се стремуваат да го пресметаат растојанието на секоја набљудена точка од центарот на камерата. Главниот принцип на работа на ToF камерите е набљудувањето на времетраењето на летот на предадените фотони. ToF технологијата се дели понатаму во „директен“ (*pulsed*) ToF и „имплицитен“ (*continuous wave*) ToF [17]. Kinect-от се базира на имплицитен ToF.

2.1.1 Директен ToF

Методот на директен ToF едноставно се базира на принципите на рефлекција на фотони и мерењето на времето потребно за изминување на целата траекторија. Сите равенки и информации се од [17].

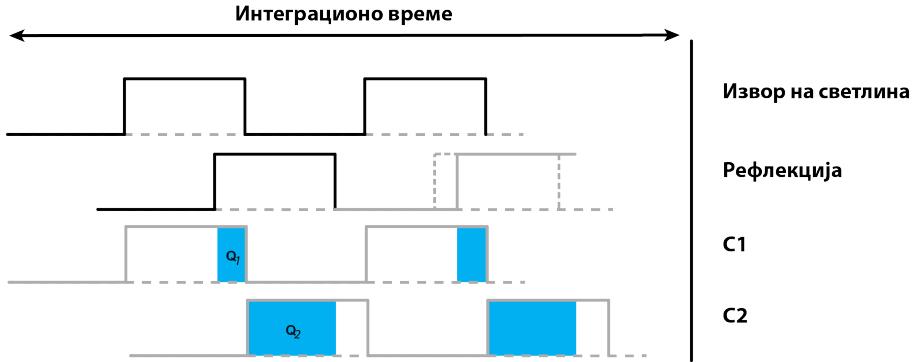
За еден многу краток период Δt се напојува изворот на светлина. Рефлектираната енергија за секој пиксел се прима, симултано, во по два приемници C_1 и C_2 кои се со иста фреквенција на семплирање, но се со фазна разлика од 180° . Се акумулираат два електрични полнежи Q_1 и Q_2 , кои се користат за да се пресмета изминатиот пат на фотонот што припаѓа на i -тиот пиксел, односно:

$$d = \frac{1}{2} c \Delta t \frac{Q_2}{Q_1 + Q_2} \quad (1)$$

Каде што:

- d е изминатиот пат на фотонот (m)
- c е брзината на светлина ($3 \times 10^8 \text{ ms}^{-1}$)
- Δt е времетраењето на светлосниот импулс (s)
- Q_1 и Q_2 се семплираните електрични полнежи.

Овој процес е прикажан во слика 2.1.



Слика 2.1: Приказ на мерниот протокол на директен ToF од [17]

2.1.2 Имплицитен ToF

Методот на имплицитен ToF се разликува од директниот на еден клучен начин: Имплицитниот метод го мери фазното отстапување помеѓу примениот и оддадениот зрак, кој што непрекинато се испраќа, додека директиот метод работи со директно мерење на време и дискретни импулси. Од оваа добиена фазна разлика, потребни се уште неколку пресметки за да се добие време на изминат пат.

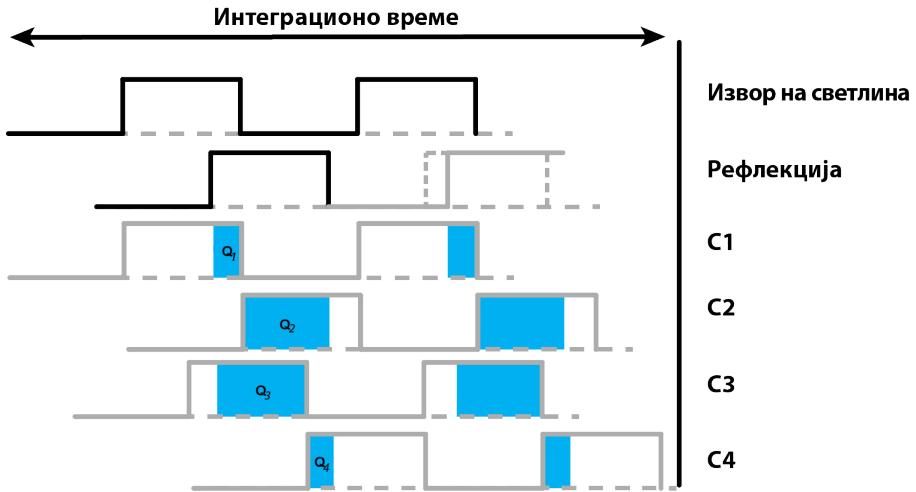
За разлика од начинот на семплирање кај директниот метод, имплицитниот метод семплира четири пати при секое мерење, со фазно отстапување 90° . Со оваа техника, фазниот агол помеѓу оддаден и рефлектиран бран ϕ и од тоа растојанието d следат:

$$\begin{aligned}\phi &= \text{atan}\left(\frac{Q_3 - Q_4}{Q_1 - Q_2}\right) \\ d &= \phi \frac{c}{4\pi f}\end{aligned}\quad (2)$$

Каде што c е брзината на светлината.

Од тука, следат и пресметките на интезитетот на пикселот A и нултото отстапување (*bias*-от) B :

$$\begin{aligned}A &= \frac{\sqrt{(Q_1 - Q_2)^2 + (Q_3 - Q_4)^2}}{2} \\ B &= \frac{Q_1 + Q_2 + Q_3 + Q_4}{2}\end{aligned}\quad (3)$$



Слика 2.2: Приказ на мерниот протокол на имплицитен ToF од [17]

Имплицитните равенки се посложени со причина - всушност одземувањето во равенката за A го амортизираат влијанието на нултите отстапувања, додека делењето во фазната пресметка ги анулира ефектите од било кои системски или рефлектирани засилувања. Следната равенка ја опишува варијацијата на пресметана длабинска информација како функција на A и B , како и фреквенцијата на оддадената светлина f и модулациониот контраст c_d , каде што модулационен контраст е способноста на сензорот да ги разделе фотоелектроните - односно чистота на отчитувањето на секој пиксел:

$$\sigma = \frac{c}{4\sqrt{2\pi f}} \cdot \frac{\sqrt{A + B}}{c_d A} \quad (4)$$

Од равенката може да извлече заклучокот дека надежноста на измерените длабини се подобрува со повиска работна фреквенција, висок модулационен контраст, и при јак светлосен интензитет, додека се влошува прецизноста при високо нулто отстапување. Една многу важна појава која треба да се земе во предвид е појавата на алиасинг. Бидејќи имплицитниот метод се базира на фазни разлики кои се повторуваат, ќе се појави растојание на двосмисленост d_{dvo} кое се дефинира според равенката:

$$d_{dvo} = \frac{c}{2f} \quad (5)$$

Сензорот е неупотреблив на растојанија поголеми од d_{dvo} , бидејќи не се разликува измерената вредност z од било која вредност $z + kd_{dvo}$ | $k \in \mathbb{N}$.

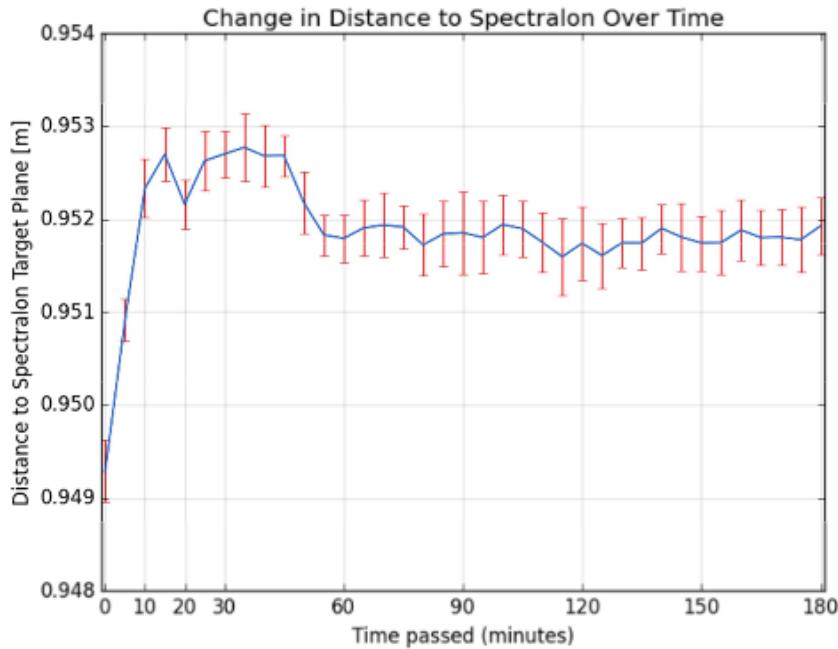
2.2 Споредба и Анализа на Kinect-от

Табелата 2.2 ги споредува важните карактеристики на двете верзии на Kinect сензорот.

Табела 1: Споредбена табела помеѓу Kinect v1 и v2

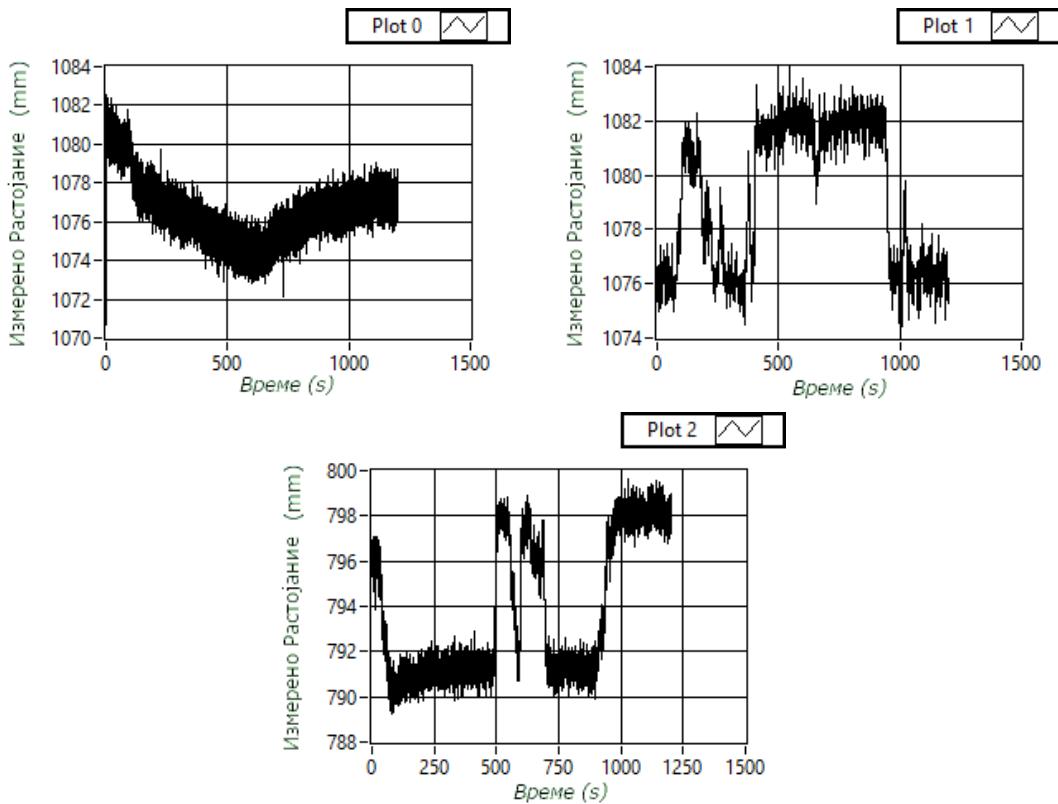
Особина	Kinect v1	Kinect v2
Бојна Камера	640 x 480 (30Hz)	1920 x 1080 (30Hz)
Длабинска Камера	320 x 240	512 x 424
Максимална Длабина	4.5m	4.5m
Минимална Длабина	40cm	50cm
Хоризонтално Видно Поле	57°	70°
Вертикално Видно Поле	43°	60°
Мотор за Аголно Помесетување	Да	Не
Зголбна Резолуција	20 зглобови	26 зглобови
Истовремено детектирани скелети	2	6
USB стандард	2.0	3.0
Оперативен Систем	\geq Win7	\geq Win8
Цена (USD)	50	200

Во текот на истражувањето за Kinect-от, наминавме на неколку трудови и статии во кои се дискутира темата на временската ненадежност поради загревање на Kinect-от при долготрајна употреба. Трудови како [1] покажуваат варијација на растојанието измерено од Kinect-от во максимален опсег од приближно 4 милиметри (сл. 2.2).



Слика 2.3: Мерења од страна на Џереми Стјуард [1]

Три слични тестирања беа изведени на три различни денови, и со нивните резултати прикажани во сл. 2.4:



Слика 2.4: Испитув ања на варијација на отчитување според време

Испитувањата открија стохастична варијација на отчитувањата во опсег од приближно 10 милиметри. Целната задача на оваа дипломска работа не е одговорна и нема посебно барање за милиметарска прецизност, односно резултатите се сепак задоволителни.

3 Примери од SDK-то на Kinect-от

Ова поглавје се состои од два делови: еден општ пример во C++ што ќе служи за опис на начинот на отчитување од Kinect-от, а во вториот дел ќе се прикажани неколку слики што се добиваат на излез од примерните програми зададени во SDK-то (*Software Development Kit*) за Kinect, како и кратки описи на другите можности на SDK-то.

3.1 Начин на отчитување во C++

Програмирањето на Kinect-от, односно неговото отчитување, не е сосема едноставно и неколку чекори се потребни за да се стигне до добивањето на самата слика.



Слика 3.1: Општ протокол за отчитување на Kinect-от

Причината за оваа сложеност е поради големата количина на можности пружена од интерфејсот за програмирање од Microsoft. За секој член од отчитувачкиот процес постои барем една опција која би влијаела на крајната слика.

Со следната програма се детектираат до 6 луѓе, и врз нивните тела се цртаат точки во сите 26 пресметани зглобови. Истовремено се испраќа состојбата на нивните раце и се прикажува таа состојба со обоени кругови околу рацете. Бојата се менува во зависност од тоа дали тие се отворени, затворени, или во „лассо“ состојби (два кренати прсти). Секој од зглобовите има свои координати во декартов систем со центар во средината на самиот Kinect. Пристапот употребен во овој пример е инспириран од примерите дадени и од Microsoft и од Nancy Owen [21].

```
#include "stdafx.h"
#include <iostream>
#include "util.h"
#include <thread>
#include <chrono>
#include "ppl.h"
#include "Kinect2_Tools.h"
int main(int argc, char ** argv){
    try {
        Kinect kinect;
        kinect.initializeSensor();
        kinect.initializeColor();
        kinect.initializeBody();

        while(true){
            kinect.updateColor();
            kinect.updateBody();
            kinect.drawColor();
            kinect.drawBody();
            kinect.showBody();
        }
    }
}
```

```

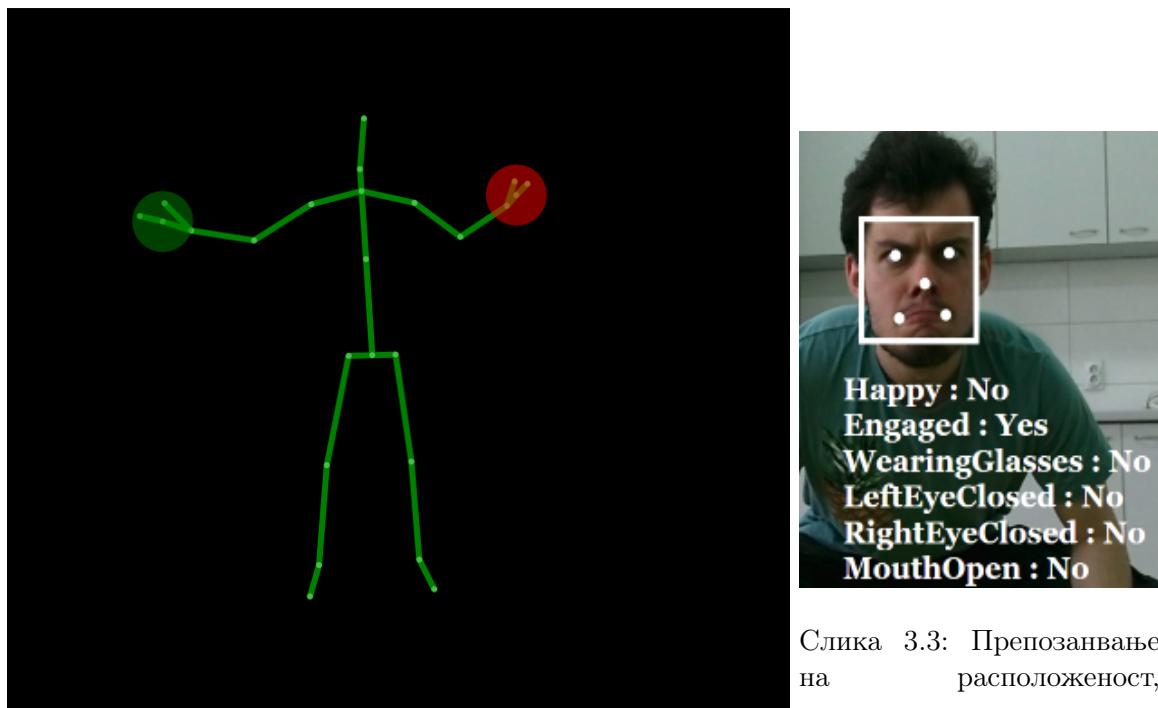
    catch (std::exception &except)
    {
        std::cout << except.what() << std::endl;
    }
}

```

Кодот е скратен со помош на функциите кои се напишани и вградени во проектот преку `include "Kinect2_Tools.h"` и `Kinect2_Tools.cpp` фајловите. Во анекс II е прикачен линк кон целосниот проект на Github, кој слободно може да се спушти, промени, и прошири.

3.2 Примери од SDK-то

Примерите дадени во SDK-то се разновидни, и пружат моќни можности. Најосновниот пример е препознавање на скелет и неговото прикажување, како што е прикажано во сл. 3.2:



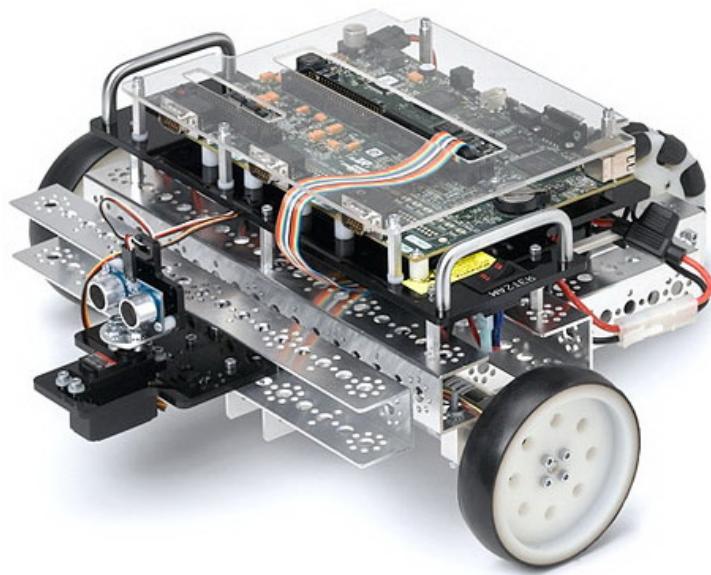
Слика 3.3: Препозанвање расположеност, заинтересираност, и др.

Слика 3.2: Детектиран скелет со прикажана отворена лева рака и затворена десна рака

SDK-то содржи примери кои го употребуваат препознавањето за посложени задачи како што се тродимензионална реконструкција и постигнување ефект на зелена завеса (*green screen*). Исто содржи и програми што ги користат другите способности на API-то - препознавање на гестикулации и на лицето и на рацете, а дури и препознавање на говор е можно со помош од соодветните пакети од Microsoft.

На сликата 3.3 е прикажан пример од SDK-то што ги користи пакетите за препознавање на лице за да направи неколку заклучоци за сликаното лице - односно дали е расположен или заинтересиран, помеѓу други работи.

4 DaNI Роботот



Слика 4.1: Роботот DaNI

National Instruments (NI) LabVIEW комплетот за роботика се состои од DaNI 2.0 (сл.4.1), кој содржи:

- Pitsco Education 12V DC мотори со 152 rpm и 21.6 kg-cm вртежен момент
- Оптички квадратурни енкодери со 400 импулси/револуција (0.9° резолуција)
- PING))) ултразвучен сензор за мерење на растојанија помеѓу 2cm и 3m
- Два Pitsco Education TETRIX 10cm тркала и едно омни тркало за насочување
- sbRIO единица и соодветни кабли за поврзување

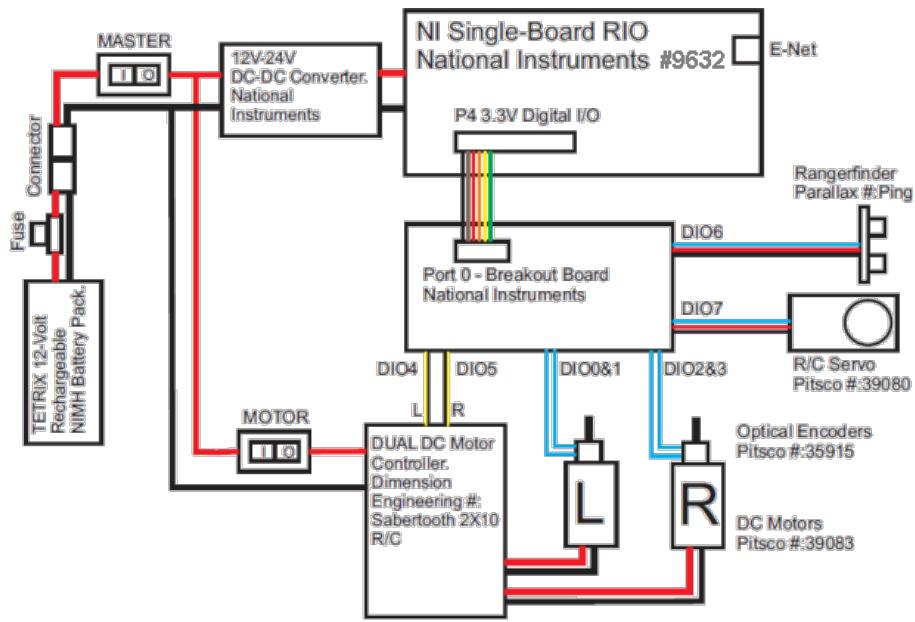
Хардверот може да биде проучуван, обратно инженериран, и модифициран. Самиот комплет има за цел да им овозможи на студенти да учат за актуелни концепти во модерната роботика како што се автономност, локализација и мапирање, и роботска перцепција. Сите потребни алгоритми се имплементирани во LabVIEW софтвер развиен на одделен персонален компјутер кои се спуштаат на роботскиот еднопложен компјутер. Електричната блок шема на DaNI роботот е прикажана во сл. 4.2.

4.1 Актуатори

Роботот DaNI поседува два DC мотори со вградени редуктори, и еден серво мотор. DC моторите се користат за погонување на роботот, додека серво моторот има улога прецизно да го насочува PING))) ултразвучниот сензор. Подолу се наведени нивните карактеристики.

4.1.1 DC мотори

DC мотор со четкици претставува внатрешно комутиран електричен мотор дизајниран да биде напојуван со извор на еднонасочна струја. Неговата брзина може да се менува или со директна промена на доведениот напон или со индиректна промена на доведениот



Слика 4.2: Блок дијаграм на DaNI

напон употребувајќи техники како PWM (*Pulse Width Modulation*), разгледан во поглавје 4.1.4. Во DaNI DC моторите се изработени од PITSCO, поточно моделот TETRIX MAX DC мотори(сл.4.3).



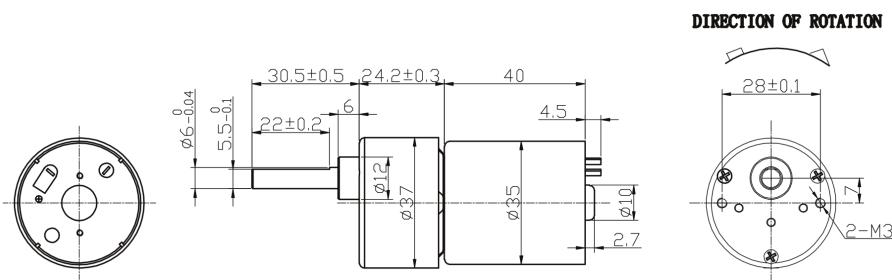
Слика 4.3: Tetrix Max DC мотор

Табела 2: Карактеристики на DC моторите

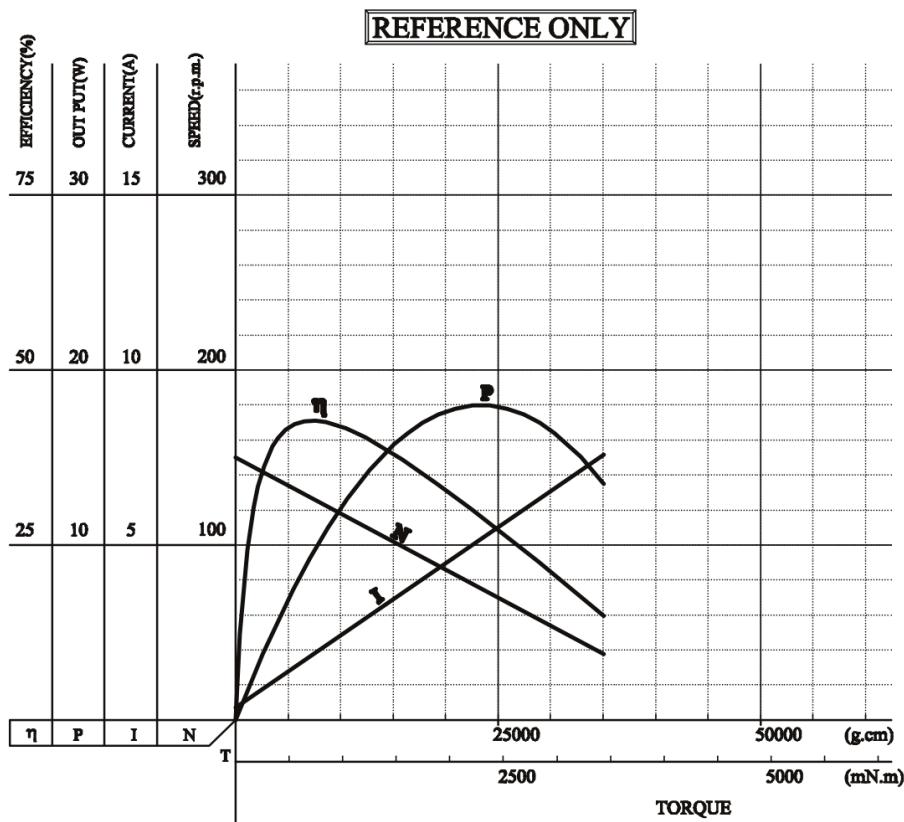
При нормални услови:		
	Номинален Напон	6 – 13.8V
	Температура на Околина	-10 ± 60°C
Услови на Испитување:		
	Напон	12V
	Температура на Околина	28°C
	Влажност на Воздухот	44%.
Електрични Особини:		
Неоптоварен:		
	Брзина	150 ± 10 vrt/s
	Струја	0.34 (0.68Amax.)
Оптоварен:		
	Вртежен Момент	0.382Nm
	Струја	0.91(1.37max.)
	Брзина	137.5 ± 10% vrt/s

Двата мотори се управувани со помош на Sabertooth Dual 10A Motor Driver. Тој може да напојува два DC мотори со четкици, доведувајќи струја до 10A на секој мотор. Но, управувачот може да истрпи максимална струја од 15A при кратки временски периоди. Поради својата ултразвучна фреквенција (32kHz) на уклучување и изгаснување на своите транзистори, Sabertooth управувачот има скоро безшумна операција. Sabertooth-от исто поседува карактеристика на регенеративност, каде што својата регенеративната топологија му овозможува на управувачот да ги полни батериите при забавување или обратно вртење на моторите. Вклучува внатрешно напојување од 5V со кое што може да напојува микроконтролер или R/C приемник.

Наредно се зададени техничката скица(сл.4.4) на моторот со неговите димензии, како и графикот на карактеристики на моторот за различно оптоварување(сл.4.5). Овој график ги дава целокупните карактеристики, т.е. карактеристиките на моторот заедно со редукторот.



Слика 4.4: Техничка скица на DC моторот

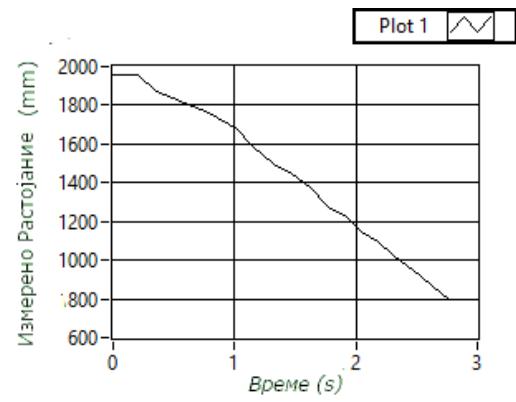
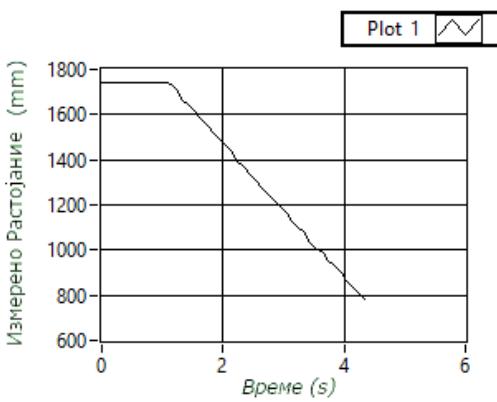
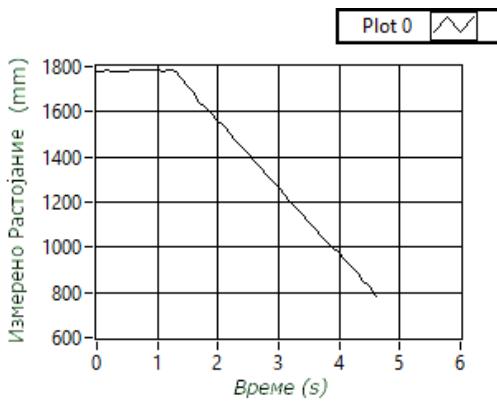


Слика 4.5: Карактеристични вредности на DC моторот

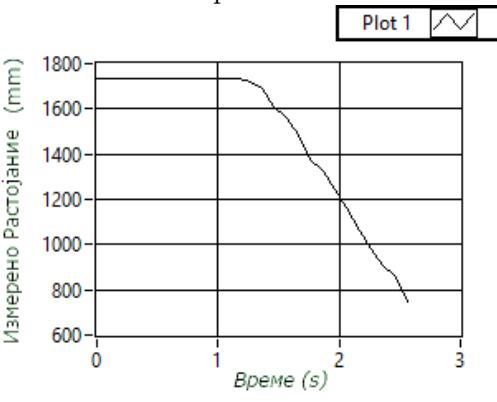
4.1.2 Анализа на измерени вредности при движење

Во средишната фаза на изработка на дипломската работа се појави штетно влијание од вибрации и немирно движење при високи брзини на моторите. Оваа хаотична вибрација доведуваше до губењето на личноста која се следи, и роботот често започнуваше па нагло запираше, бајкји ја личноста која ја следеше. Се извршле неколку испитувања - споредувајќи ја отчитаната длабина од Kinect-от при разни брзини на движење на моторите.

Како што се гледа од сликите подолу (сл. 4.10), при брзини поголеми од 0.3 ms^{-1} , веќе почнува значително да варира измерената вредност. Независно дали ова влијание потекнува од осцилации од моторите и конструкцијата или од грешки создадени при задвижување на Kinect-от, ова влијание сигурно игра голема улога при детектирањето на човекот, и штетно влијае врз операцијата на целиот автономен робот.



Слика 4.6: Брзина 0.3 ms^{-1}



Слика 4.7: Брзина 0.5 ms^{-1}



Слика 4.8: Брзина 0.8 ms^{-1}

Слика 4.9: Брзина 1 ms^{-1}

Слика 4.10: Измерена длабина при разни брзини на движење

4.1.3 Серво мотор

Серво мотор е ротационен актуатор кој дозволува прецизна контрола на аголната позиција на својот излез, т.е. осовина. Тој се состои од високо редуциран мотор поврзан со енкодер што дава повратна информација за моменталната положба на осовината. Овој склоп има и потреба од електронски серво драјвер, кој улога да прими наредбен сигнал од главниот управувачки систем, да го засили сигналот, и да создаде соодветен напон врз серво моторот со цел да го произведе побараното движење. За оваа цел го користи енкодерот, кој со повратна врска што ја создава му овозможува на моторот прецизно да ја зафати било која позиција во својот опсег.

Единствена функција на серво моторот во оваа конструкција е ротација на PING))) сензорот, поради што тој не е изложен на значителни оптоварувања.

Табела 3: Карактеристики на серво моторот

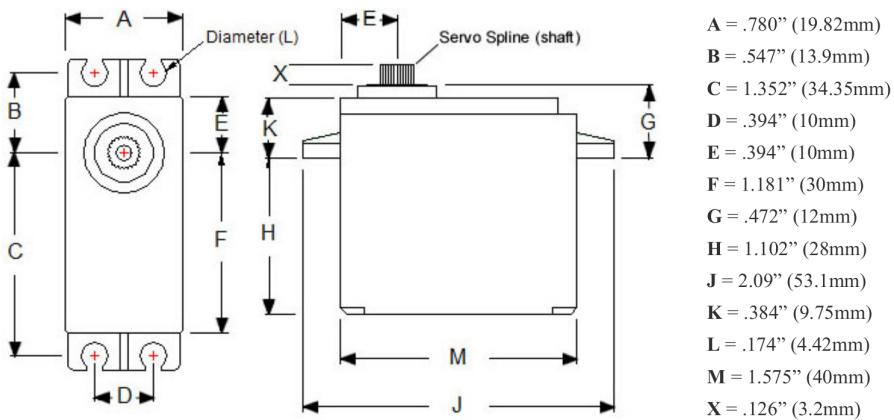
Димензии	39.88 x 19.81 x 37.85mm
Тежина	45g
Ранг на напон	4.8V - 6.0V
Брзина без оптоварување (4.8V)	0.22sec/60°
Брзина без оптоварување (6.0V)	0.18sec/60°
Вртежен момент на запирање (4.8V)	4.8kg.cm
Вртежен момент на запирање (6.0V)	6.0kg.cm
Максимален ранг на PWM	553-2425μsec
Поминат агол по μsec	.102°/μsec
Максимален пат	190.5°
Амплитуда на импулс	3-5V
Работна температура	-20°C до +60°C
Потрошувачка на струја - неактивен (4.8V)	8mA
Потрошувачка на струја - неактивен (6V)	8.8mA
Потрошувачка на струја - без оптоварување (4.8V)	150mA
Потрошувачка на струја - без оптоварување (6V)	180mA
Можност за континуирана ротација	+
Насока со зголемување на PWM сигнал	Во насока на часовата стрелка
Вид на запченици	Запченици со прави заби
Материјал на запченици	Карбонит

4.1.4 PWM

PWM (*Pulse Width Modulation* или модулација со промена на ширина на импулси) е начин на аналогно управување со употреба на чисто дигитални сигнали. При една претходно одредена фреквенција на операција, имаме една соодветна периода, $T(s)$. Во рамките на една период, можеме да ја дефинираме врската помеѓу времетраењето на активноста (ON) и времетраењето на неактивноста (OFF) на еден дигитален излез. Добиениот „аналоген“ излез (ефективниот напон) се пресметува со следната равенка:

$$V_{PWM} = V_{dig} \cdot \frac{T_{ON}}{T_{ON} + T_{OFF}} = V_{dig} \cdot \frac{T_{ON}}{T} \quad (6)$$

На пример, ако имаме работна фреквенција 32kHz, имаме периода $31.25\mu s$. Нека дигиталниот излез биде 12V (т.е. ON = 12V, OFF = 0V). Ако за $20\mu s$ од секоја периода

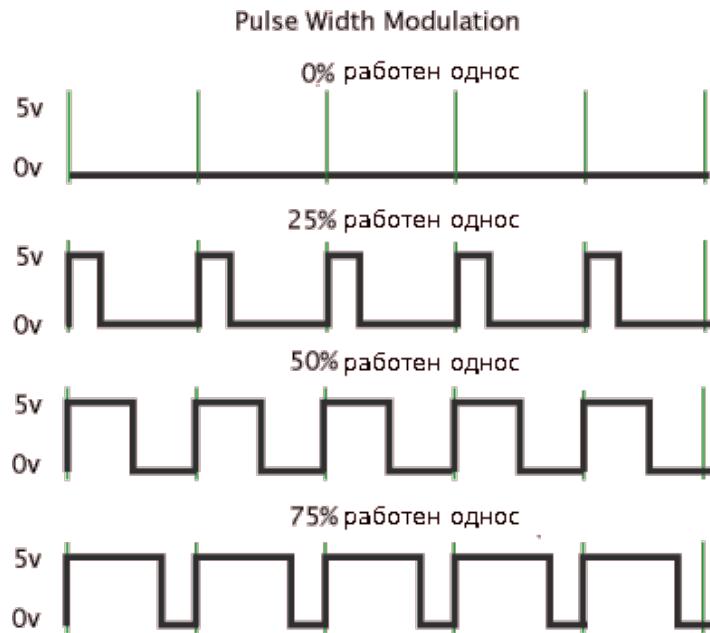


Слика 4.11: Димензии на серво моторот

задаваме ON сигнал, а за останатите $11.25\mu s$ задаваме OFF сигнал, на излез ќе го добиеме ефективен напон:

$$V_{PWM} = 12 \cdot \frac{20}{20 + 11.25} = 12 \cdot 0.64 = 7.68V$$

Во сл. 4.12 е прикажан принципот на PWM графички:



Слика 4.12: Графички приказ на PWM

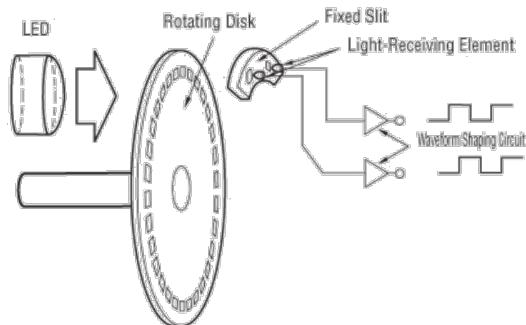
4.2 Сензори

Во најширока дефиниција, сензор претставува уред, модул или потсистем чија задача е да препознае настани или промени во својата физичка околина, често преку електрофизички настани, и да испрати согласна информација кон останатата електроника/управувачка единица (најчесто компјутерски процесор).

Роботот DaNI поседува два вида на сензори: енкодер и ултразвучен сензор.

4.2.1 Енкодер

Ротационен енкодер е електромеханички уред кој ја претвора аголната позиција на вратилото на кое што е наместено во аналоген или дигитален сигнал. Оваа информација понатаму може да се манипулира математички - на пример да се бараат повеќе изводи - за да се добие информација за брзината на моторите, забрзувањето на моторите, и линиските брзини и забрзувања на самата конструкција.



Слика 4.13: Шема на оптички енкодер

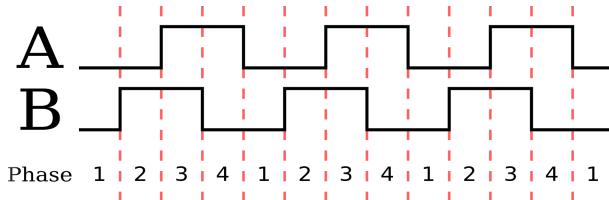
Постојат два главни видови на енкодер: апсолутни и инкрементални (релативни). Апсолутниот енкодер има можност да ја следи апсолутната аголна позиција на вратилото во делови на степен, што значи дека енкодерот претставува директен трансдусер за аголна позиција. Релативните енкодер можат само да измерат точно колку е заротирано вратилото од момент до момент, односно даваат само информација за движењето.

Ротационите енкодери се користат во многу области на апликации кои имаат потреба од прецизна контрола и неограничена ротација на вратило, вклучувајќи индустриска автоматизација и контрола, роботика, фотографски леѓи за специјални примени, компјутерски влезни уреди, и ротациони радарни платформи.

Енкодерите на DaNI роботот се инкрементални квадратурни енкодери(сл. 4.13), кои користат два излеза A и B кои се наоѓаат на 90° фазно поместување еден од друг(сл. 4.14). Исто така дисковите на енкодерите имаат по 400 отвори, односно енкодерите имаат резолуција од 0.9° . Табелата 4 и сл. 4.14 го опишуваат енкодерскиот систем.

Табела 4: Зголемување на фазата имплицира ротација во насока на стрелките на часовникот

Фаза	A	B
1	0	0
2	0	1
3	1	1
4	1	0

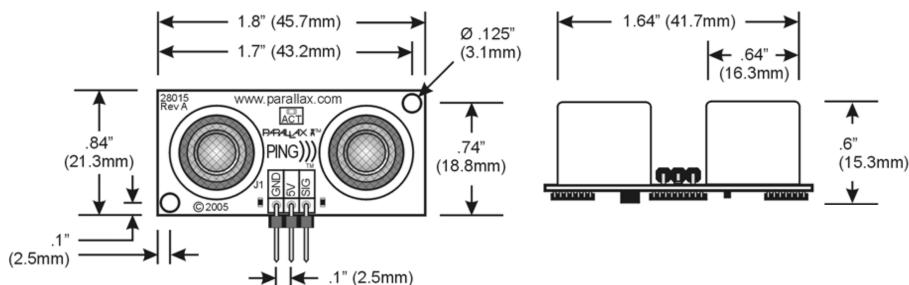


Слика 4.14: Два квадратни бранови во квадратура (ротација во насока на стрелките на часовникот)

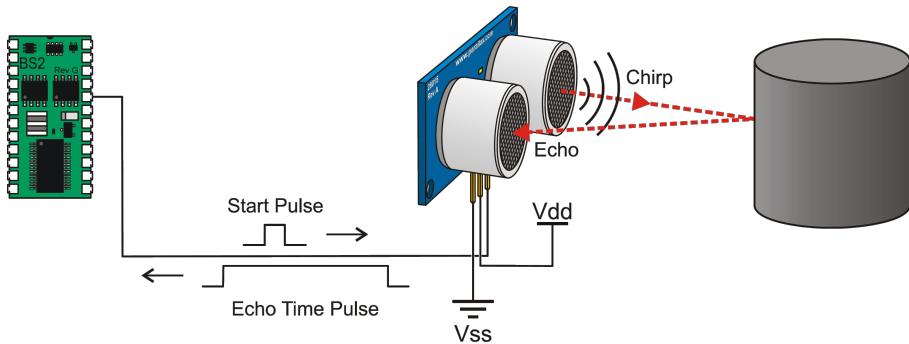
4.2.2 Ултразвучен сензор

Ултразвучниот сензор PING)) од Parallax (сл. 4.15) е способен да извршува прецизни мерења на растојанија од 2cm до 3m. Лесно се приклучува на микроконтролери како BASIC Stamp, Propeller chip, или Arduino, користејќи само еден дигитален пин влез-излез. Се напојува со 5V и влечи 30mA.

PING))) сензорот работи со пренесување на ултрасоничен ($\geq 20kHz$) збир или импулс (*burst*) на бранови, и произведува излезен пулс кој соодветствува на времето потребно за ехото на брановите да се врати до сензорот. Со мерење на ширината на излезниот пулс, лесно може да се пресмета растојанието до објектот.



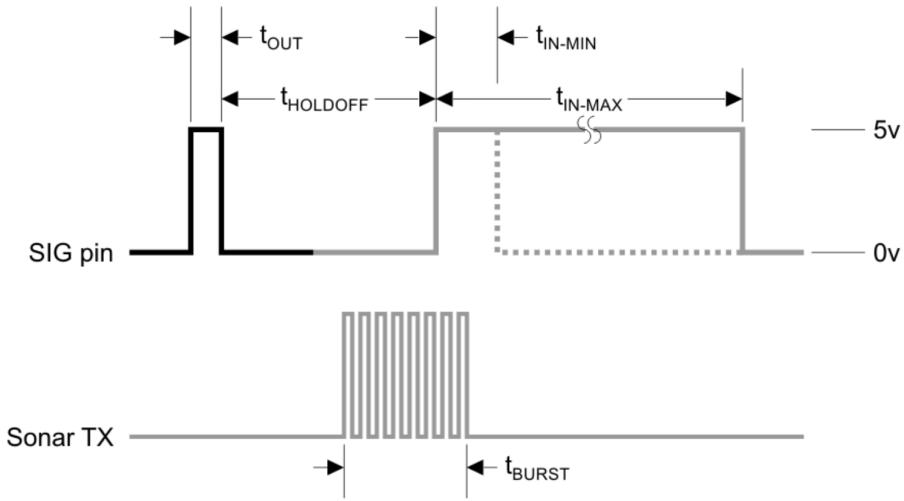
Слика 4.15: Димензии на PING))) сензорот



Слика 4.16: Шема и принцип на дејство на ултразвучен сензор

Протокол на комуникација

Под контрола на главен микроконтролер (пулс на активација), сензорот оддава краток сигнал со фреквенција 40kHz. Овој сигнал патува низ воздухот, се судира со објект и се одбива назад кон сензорот. Меѓувремено сензорот испраќа излезни пулсови до микроконтролерот кој ќе прекине да ги испраќа штом се детектира ехото. Со оваа информација може да се пресмета изминатиот пат на збирот, односно да се пресмета растојанието до предметот (сл. 4.16).



Слика 4.17: Изглед на сигналот

Табела 5: Карактеристики на сигналот

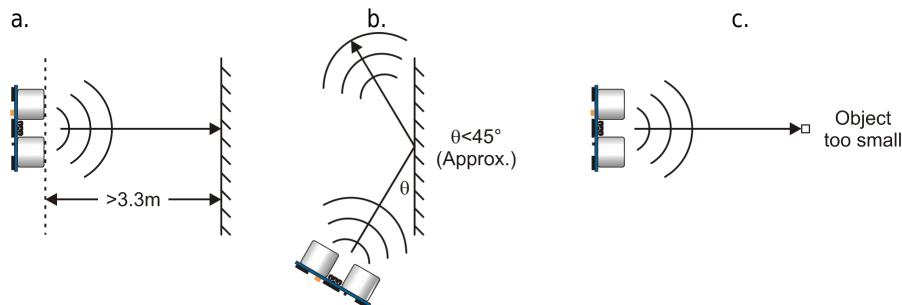
Темна линија	Главен Уред	Влезен пулс на активација	t_{OUT}	$2\mu s$ (min), $5\mu s$
Светла линија	PING))) Сензор	Задржување на ехо Фреквенција на сигналот (burst) Ехо повратен сигнал минимум Ехо повратен сигнал максимум Пауза пред следно мерење	$t_{HOLDOFF}$ t_{BURST} t_{IN-MIN} t_{IN-MAX}	$750\mu s$ $200\mu s @ 40kHz$ $115\mu s$ $18.5\mu s$ $200\mu s$

Позиционирање на објектот

PING))) сензорот не може прецизно да измери растојание до објект кој што:

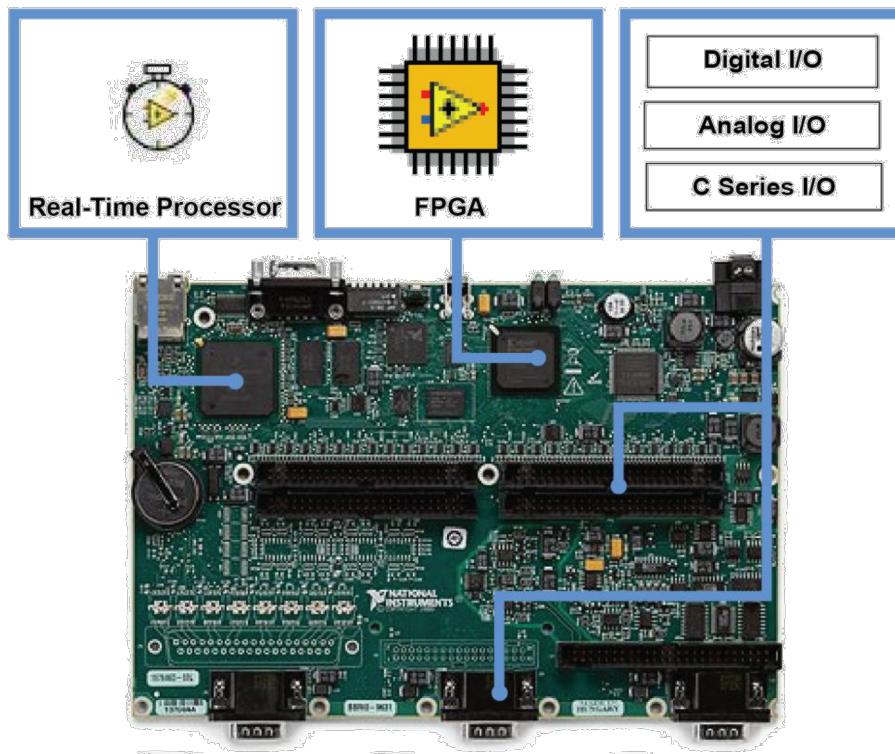
- се наоѓа подалеку од 3m
- има мал агол на рефлектирачката површина со што сигналот нема да биде одбиен назад кон сензорот
- е премногу малечок за да рефлектира доволно звук назад кон сензорот

Дополнително, доколку PING))) сензорот е поставен на ниска позиција на уредот, има можност да детектира звук кој се одбива од подот.



Слика 4.18: Позиционирање на објектот

4.3 NI sbRIO 9632



Слика 4.19: sbRIO со обележани компоненти од [14]

sbRIO (single-board Reconfigurable Input/Output) управувачките единици од National Instruments претставуваат компјутер монитран на една плочка (*single board*) наменета за случаи каде што е потребно решение со управување во реално време (*Real-Time*). Оперативен систем кој работи во реално време (*RTOS: Real Time Operating System*) е изработен со посебната цел да извршува функции кои барат големо ниво на временска прецизност и висок степен на надежност. За некој систем да се смета за RTOS, тој мора да има познато максимално време на извршување за секоја од неговите клучни операции. Системите кои можат сигурно да обезбедат максимален временски одсив се вели дека работат во потполно реално време, додека системите кои можат само понекогаш да обезбедат максимален временски одсив се вели дека работат во делумно реално време.

Потребната брзина се постигнува со истовремето користење на FPGA чип и една обична микропроцесорска единица.

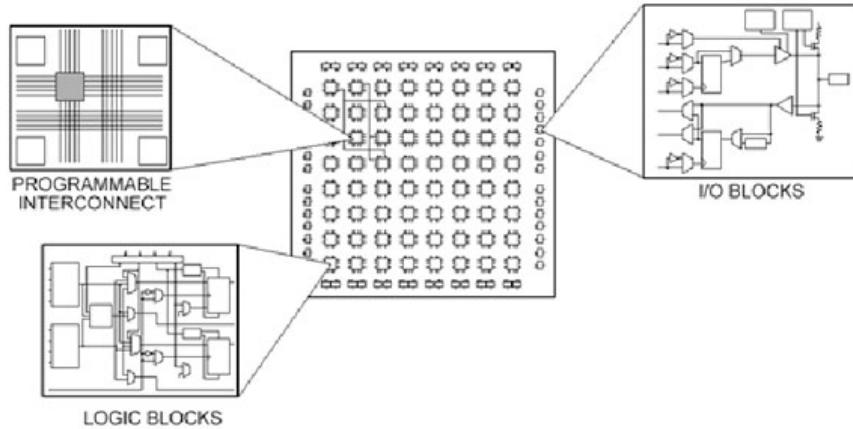
sbRIO поседува 110 дигитални влезови-излези, 100 од кои се оспособени за PWM (4.1.4), а 10 од кои се наменети за ниско-фреквентна намена. sbRIO поседува и 32 аналогни влезови, и 4 аналогни излези со ранг на излез од $\pm 10 V$ со 0.1% грешка при типична употреба на $25^{\circ}C \pm 5^{\circ}C$ [11].

sbRIO содржи и 3 сериски „C“ портови за поврзувањето на надворешни модули од NI за проширување на можностите на sbRIO да опфатат и актуација и аквизиција на податоци.

4.3.1 FPGA

Како што е објаснето во [24], FPGA, или „Field Programmable Gate Array“, претставува репограмабилно интегрирано коло кој поседува голем број на програмабилни логични порти. Кај обичните микроконтролери, логиката за управување се пишува и компајлира од некои програмски јазик како C, BASIC или некој графички јазик како G (LabVIEW). Во текот на овој процес, напишаните програми и потрутини се сублимираат во процесорски

наредби/инструкции како ADD и MOV. Низата на можните наредби е единствена за секој микропроцесор. Кај FPGA колата, програмата не се сублимира на наредби, туку самата внатрешна архитектура на колото се подредува со електромагнетни полиња. Како резултат, се добива конфигурација на логични порти која ќе ја извршува задачата описана во првобитната програма.



Слика 4.20: Приказ на често сретнати термини кај FPGA кола од [14]

FPGA колото што се наоѓа во sbRIO-то е моделот Xilinx Spartan 6, кој поседува 6 милиони репрограмабилни логични порти. Бидејќи FPGA-та нудат хардверско решение и имаат природно поголема можност за паралелни пресметувања, нивната операција е често многу пати побрзо од работењето на обичните микроконтролери, но се подрагоценi. FPGA-та наоѓаат најголема употреба во малосериска/ограничена продукција со помалку од 500-1000 изработени парчиња. При поголема серија на производство, ASIC (*Application Specific Integrated Circuits*) чиповите се поисплатливи, а и репрограмабилноста на FPGA-то веќе нема значење при општа употреба.

5 Теоретска Позадина

5.1 Генерален Опис на Системот

Започнувајќи од основната задача на системот, тој има цел да ги препознае личностите пред себе и според некои критериуми наложени од корисникот да одреди која личност најдобро соодветствува на тие критериуми. По тоа, роботот ќе ја следи избраната личност, одржувајќи се на некое одредено растојание. Брзината на моторите треба да биде доволно голема да се следи човекот со задоволителна брзина, а доволно мала за да не се појавуваат отскокнувања, нагли запирања, или осцилаторни движења.



Слика 5.1: Крајната форма на DaNI

Системот се состои од длабинско-бојната камера Kinect од Microsoft и еден Robotics Starter Kit управуван со sbRIO едно-плочен компјутер, набавени од NI. Како главна процесорска единица се користи лаптоп кој го води целиот процес во LabVIEW. Како што се гледа при споредба на сл. 5.1 и сл. 4.1, роботот е прилично променет, и е веќе прилагоден кон задачата.

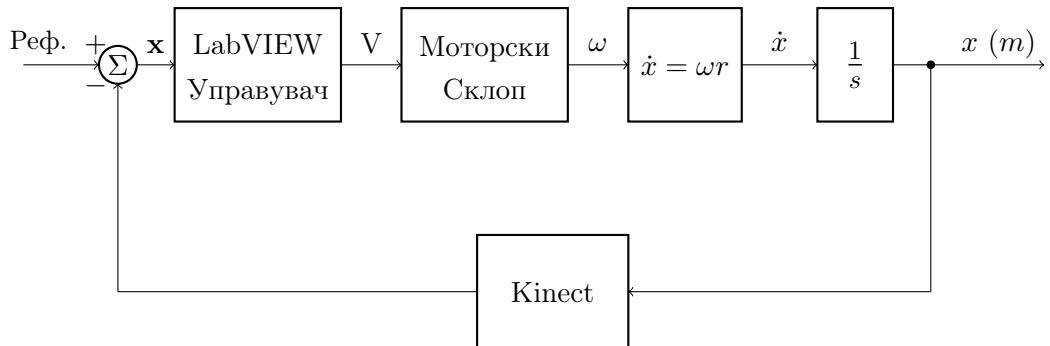
Подолу се описаны во општи црти етапите на функционирање:

- a. Kinect-от слика и испраќа бојна и длабинска информација за сцената пред себе со фреквенција од 30Hz кон главната процесорска единица.
- б. Со употреба на алгоритмите за човеково препознавање на Microsoft, се препознаваат луѓето кои се наоѓаат во сцената.

- в. Според критериумот за избор, една (или ниту една) од препознатите личности ќе биде избрана за следење, и нејзините декартови координати во однос на камерата се даваат на излез.
- г. Координатите се претворуваат во поларни координати и потоа уште еднаш се претворуваат во облик на општ информациски пакет и се испраќаат до едноплечниот компјутер на роботот.
- Важно е да се опомени дека во меѓувреме разновидни процеси работаат паралелно, како што се читачи за промени во засилувањата на системот, посаканото растојание на роботот, и пораки за итно запирање на роботот.
- д. Роботот ги прима и толкува пораките, одредувајќи кој вид на информација го содржи тој пакет и која вредност ја носи. Во тек на обична употреба, роботот ќе ги прими поларните координати и ќе ги доведе во свој два паралелни ПД управувачи. Подетално ќе е разгледан двојниот-ПД систем во наредно поглавје.
- ѓ. Излезите од ПД управувачите се претворуваат во дигитални PWM излези и се доведуваат до моторските управувачи.

5.2 Блок Дијаграм на Системот

Горенаведените елементи сочинуваат еден значително сложен систем, со развиени софтверски компоненти за управување на електромеханичниот склоп што го претставува роботот. Иако системот е прилично сложен и со безбројно многу меѓув зависности помеѓу информационите, електричните, и механичките потсистеми, за наша употреба самиот робот може да се упрости и да се апроксимира како механички систем од втор ред кој е линеарен и временски независен. Во сл. 5.2 е прикажан усвоениот упростен систем:



Слика 5.2: Блок дијаграм на усвоениот систем

Каде што:

- Реф. е референтното растојание на роботот кое што е побарано од корисникот.
- Векторот \mathbf{x} ги претставува пропорционалните и диференцијалните компоненти на промената на растојанието.
- Блокот **LabVIEW Управувач** го претставува алгоритамот изведен во LabVIEW.
- V е излезниот напон кој што се доведува до моторите.
- Излезот од моторите е $\omega = \theta s^{-1}$ во радијани.
- Излезот на претворувачкиот блок е брzinата на роботот во ms^{-1} .
- По интегрирање со членот $\frac{1}{s}$ се добива изминатиот пат на роботот во метри.

5.3 Мобилна Роботика

Стационарните индустриски роботи во вид на роботски раце се употребуваат во големо-сериското производство на автомобили дури од 1970тите [25], каде што драстично ја подобрија продуктивноста во средини каде што работата е повторлива, бара голема прецизност, и/или е опасна.

Мобилните роботи би нашле - и наоѓаат ден денеска - примена во разновидни средини поради слични причини. Интелигентни и мобилни роботи можат да најдат примена во земјоделство, производство, болници, обезбедување, мерења во средини кои се штетни за човекот, како и безброј воени употреби [5].

Постојат разновидни елементи кои се применуваат за задвижување на роботите, како што се: нозе, тркала, и траки. Благодарение на модерните производни технологии и материјали, изводливи се начини на задвижување кои се моделирани на појави во природата (*biomimicry*), како што се: перки, крилја, и нозе.

5.3.1 Холономност и диференцијален погон

Изборот на начинот на задвижување не се врши само според условите на средината во која ќе функционира роботот, туку и според пресметковните ограничувања и потребите за агилно движење и степените на слобода. Тука се воведува концептот на **холономност**. За робот се вели дека е холономен кога тој поседува можност за директно управување во секој од своите степени на слобода [3]. Односно, еден рамнински робот (како DaNI) со три степени на слобода (θ , x , y) е холономен ако тој има директна можност за задвижување по оска x , y , и да се ротира околу својата оска θ . DaNI роботот, како и повеќето на роботи во класата на *rovers* ја немаат оваа можност, бидејќи не можат директно да се преместуваат по оската x . Овие се нарекуваат **нехолономни**. За вакво хоризонтално движење, тие мораат да се ротираат, па по своја y -оска да се преместуваат додека не ја стигнат посаканата точка, и повторно да се заротираат кон точната ориентација.

Бидејќи DaNI роботот има два мотори кои независно еден од друг се управуваат, имаме случај на диференцијален погон, каде што линеарната брзина (\dot{y}) е функција на големината на аголните брзини на моторите, додека аголната брзина ($\dot{\theta}$) е функција на разликата на брзина помеѓу двета мотори. Во случај брзините на десното и левото тркало да се исти по големина а спротивни по знак, роботот би се ротирал во место, додека еднакви брзини предизвикуваат задвижување напред. Секоја друга варијација на брзини предизвикува криволиниско движење по лак со радиус R со центар наречен „Моментален Центар на Ротација“ (МЦР) (Англиски: *Instantaneous Centre of Curvature*). Равенките за пресметка на R , МЦР, и аголната брзина околу МЦР ω , добиени од [3], гласат:

$$R = \frac{d}{2} \frac{v_d + v_l}{v_d - v_l} \quad (7)$$

$$MCR = [x - R\sin(\theta), y + R\cos(\theta)] \quad (8)$$

$$\omega = \frac{v_d - v_l}{d} \quad (9)$$

Каде што:

- d е меѓусно растојание на тркалата во m
- v_d е аголната брзина на десното тркало во $rads^{-1}$
- v_l е аголната брзина на левото тркало во $rads^{-1}$
- θ е аголот создаден помеѓу осовините на тркалта и X-оската во rad

5.3.2 Вектор модел и ПД управување

Откако камерата го препознае телото, таа ги наоѓа неговите карактеристичните точки, и ги преобразува нивните податоци од длабинската камера во x, y, z декардни координати во t каде што $0, 0, 0$ е центарот на инфрацрвената камера, x е горизонталното растојание (лево и десно), y е вертикалното растојание (горе и долу), и z е нормалното растојание од камерата (напред и назад). Преку овие податоци, пресметуваме два вектори, т.е поларните координати - **аргумент** и **модулус**, каде што модулусот (mod) е растојанието до карактеристичната точка на следеното лице, како што е прикажано на сликата и пресметката подолу.

$$\begin{aligned} mod &= \sqrt{x^2 + z^2} \\ \tan(arg) &= \frac{x}{z} \\ arg &= \tan^{-1}\left(\frac{x}{z}\right) \end{aligned} \quad (10)$$

Пожелно е личноста која што ја следи роботот да биде во центарот на видното поле на камерата, и согласно аргументот треба да се стреми кон вредност 0, додека модулусот треба да се стреми кон вредноста зададена од личноста која што се следи. Управувањето на овие два параметри е постигнато со два паралелни ПД управувачи. Едниот ПД управувач е одговорен за оддржување на аргументот на 0, и на свој излез дава две инверзни брзини потребни за посаканата ротација, додека другиот ПД управувач е одговорен за оддржување на соодветен модулус и на излез дава две истонасочни брзини потребни за посаканото праволиниско движење.

Поради апроксимацијата дека системот е линеарен и временски независен, двата излези од ПД управувачите можат едноставно да се соберат аритметички и да се доведе соодветниот напонски сигнал до моторите. Комбинацијата на чиста ротација и чисто праволиниско движење го произведува потребното криволиниско движење. Равенките за двата ПД управувачи и нивното суперпонирање гласат:

$$\begin{aligned} v_d^{rot} &= K_p^{rot}(a_{ref} - a_i) + K_d^{rot}(a_i - a_{i-1}) \\ v_l^{rot} &= -v_d^{rot} \\ v_d^{lin} &= v_l^{lin} = K_p^{lin}(m_{ref} - m_i) + K_d^{lin}(m_i - m_{i-1}) \\ v_d &= v_d^{rot} + v_d^{lin} \\ v_l &= v_l^{rot} + v_l^{lin} \end{aligned} \quad (11)$$

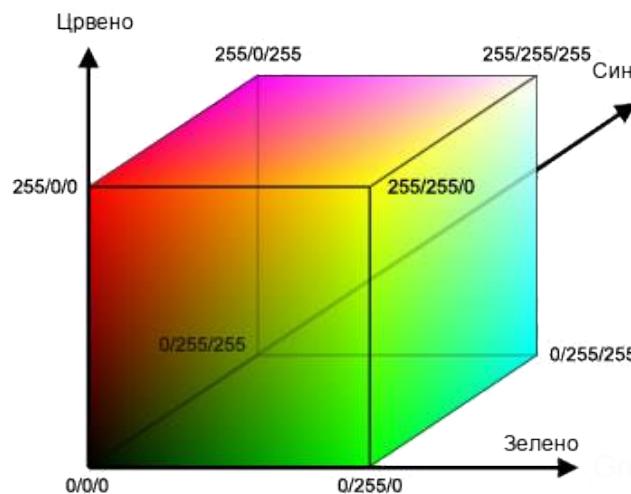
Каде што:

- a_i и m_i се сегашните вредности на аргументот и модулусот
- a_{i-1} и m_{i-1} се претходните вредности на аргументот и модулусот
- a_{ref} и m_{ref} се референтите вредности за аргументот и модулусот
- v_d^{rot} и v_l^{rot} се ротацијоните брзини на десното и левото тркало
- v_d^{lin} и v_l^{lin} се линиските брзини на десното и левото тркало
- K_p^{lin} и K_p^{rot} се пропорционалните засилувања на линискиот и ротациониот ПД управувач
- K_d^{lin} и K_d^{rot} се диференцијалните засилувања на линискиот и ротациониот ПД управувач
- v_d и v_l се крајните брзини на десното и левото тркало

5.4 Теорија на Боја за Компјутерски Системи

За да може да се објаснуваат соодветно бојните алгоритми што следат во поглавје 6.3.2, важно е читачот да има посевуда едно посебно разбирање за поимот боја. Во секојдневието, човекот ретко размислува за поимот боја. Објектите си имаат своја боја, и тоа е често крај на приказната. Кога ќе се разгледа основната физика на бојата, поимот боја се разбира од гледна точка на електромагнетни бранови со посебни фреквенции. Оваа дефиниција е сосема задоволителна и исправна за инженерско размислување, но кога се дискутира за поимот на процесирање на слики станува многу важна уште една перспектива: претставувањето на една боја како три-димензионален вектор во еден конечен простор.

Трихроматичната теорија на вид се појавила на почетокот на 18-тиот век, кога Томас Јанг преложил дека човековата способност да гледа бои била поради присуството на три различни видови на фотоприемнички клетки во очите [29]. Теоријата била понатаму разработена од Џејмс Максвел и Херман вон Хелмхолц пред да биде физиолошки докажана од Гуннар Сваетчин во 1956-та година [27].



Слика 5.3: Конечниот боен простор од [6]

Секоја од овие клетки има способност да прима светлина со ниска (црвена), средна (зелена), или висока (сина) фреквентна содржина. Со суперпонирањето на информациите што ги добиваат сите овие клетки се добива целосната бојна информација. Според овој модел, секоја боја може да се разреди во три елементарни компоненти: нејзината црвена (R), зелена (G), и сина (B) компонента. Овие компоненти можат да се ставаат во еден вектор $V = [R \ G \ B]$. Човековото око може да препознае од прилика 10 милиони бои [16], што значи дека просторот во кој овој вектор би постоел е конечен. Ако се смета дека секоја од оските на овој простор носи иста тежина, тие би имале една иста максимална вредност. Ако се направи обид секоја компонента да се опише со еден бајт, произлегува максимална вредност во било која насока да е 255. Употребувајќи по еден бајт за секоја компонента, испаѓа дека целата боја се опишува со еден 24 битен број. Еден 24 битен број може да создаде $2^{24} = 16,777,216$ комбинации - што се доволно комбинации да ги опишат сите бои што би можел еден човек да види. Од тука произлегува дека конечниот простор е коцка со димензии $a = R_{max} = G_{max} = B_{max} = 255$. Овој конечен боен простор е прикажан во сл. 5.3

6 Софтверско управување со LabVIEW

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) е софтверски пакет наменет за програмирањето на виртуелни уреди (инструменти) за мониторинг и управување на физички уреди. Инструментите што се програмираат во LabVIEW можат да се компајлираат и да се издаваат како комплетно независни програми кои можат да се монтираат како главен управувачки софтвер на мехатронички уреди и машини. LabVIEW инструментите се програмираат користејќи го нивниот сопствен графички програмски јазик, „G“.

Основниот пакет на LabVIEW поседува многу од стандардните можности што се очекуваат од било кој традиционален програмски јазик, како што се логички/булови операции, математички операции, и пристап до алатки за визуелизација на податоци.

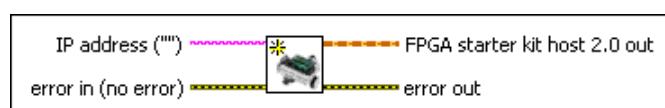
LabVIEW поддржува (и понекогаш бара) проширување со додатни пакети. На пример, овие пакети можат да содржат готови под-инструменти (sub-VIs) за обработка на сигнали (*Signal Processing Module*), или да овозможуваат соработка помеѓу LabVIEW и некои други технологии (*FPGA module*). Од овие пакети во оваа дипломска работа се употребени Real Time, FPGA, Vision Development, и Robotics пакетите. Првите два пакети го оспособуваат LabVIEW да може да програмира FPGA чипови и да програмира надворешни вградени системи (*Embedded Systems*) со оперативни системи кои управуваат во реално време, Vision Development го олеснува оперирањето со слики, додека Robotics пакетот содржи во себе голем број на готови инструменти за класична и инверзна кинематика, отчитување од сензори, и праќање наредби кон мотори.

Robotics пакетот исто поседува едно потмножество на инструменти наменети само за DaNI 2.0, и со тие е управуван DaNI роботот. Исто така, за посебна хардверска поддршка за Kinect и *bluetooth* контролер за PlayStation се употребени библиотеките од *LabVIEW MakerHub*, еден посебен систем за модули направени од публиката со отворен извор на код.

Да се спомени дека во контекстот на LabVIEW, термините *функција* и *блок* се земени за еднакви по значење, односно за меѓусебно заменливи.

6.1 Robotics модул: Starter Kit 2.0

Во Robotics модулот на LabVIEW, од интерес се готовите VIs за Starter Kit 2.0, кои содржат во нив потпрограми за иницијализација и деиницијализација на роботот и задавање на брзина на ротација на моторите. Тука се наброени и објаснети овие блокови.

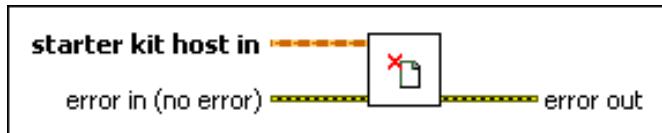


Слика 6.1: VI за иницијализација

Иницијализација:

Првиот блок служи за иницијализација на роботот. Како влез се доведува IP адресата на DaNI роботот, и според зададената адреса започнува комуникацијата со DaNI. При иницијализација на DaNI се создава „објект“ кој го претставува роботот. Програмерскиот термин „објект“ дефинира конгломерат на податоци и функции кој го опишува некој абстрактен предмет. Овие предмети се често неопипливи, но во случаи како овој, овој „објект“ е еден кибернетски претставнички меѓуслој кој овозможува комуникација со физичкиот систем. Како излез на оваа функција е самиот објект-претставник на DaNI, што претставува предуслов за употребата на било која од другите функции. *Error-in* и *Error-out* портите на VI-то се за заштита при некоја грешка во системот. Ако грешката е

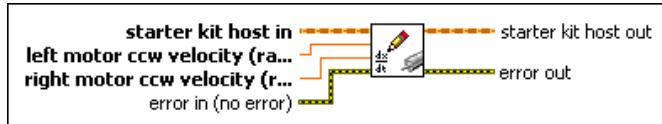
1 (има грешка), целата функцијата на иницијализација се заменува со куса врска, и нема излезен објект.



Слика 6.2: VI за деиницијализација

Деиницијализација:

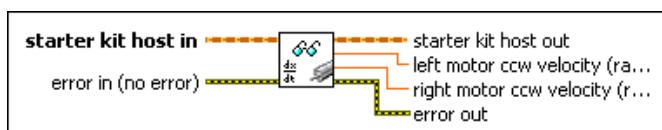
Овој блок се става на крај на програма за да се избрише објектот создаден од иницијализацијата. Без оваа крајна функција, можно е грешки се појавуваат при повторно лансирање на програмата и/или на роботот.



Слика 6.3: VI за дефинирање на брзина

Управување на моторите:

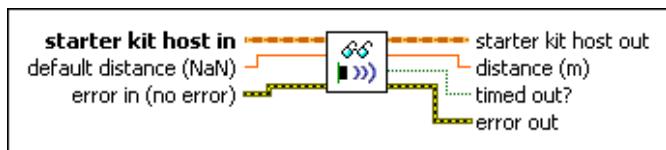
Овој блок е задолжен за управувањето на брзината на двата DC мотори на DaNI. За функционирањето на блокот потребен е самиот објект на DaNI, како и две вредности за врзините на моторите во rad/s . За движење во било која насока, важно е да се опомени дека брзините на моторите морат да бидат со обратен предзнак поради нивната обратна поставеност. Моторите имаат максимална брзина од $15.7 rad/s$.



Слика 6.4: VI за отчитување од енкодерите

Отчитување од енкодерите:

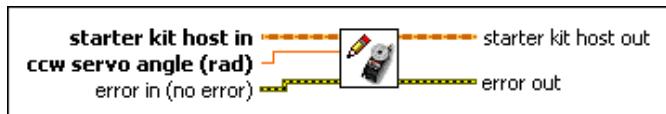
Со овој блок се отчитуваат моменталните вредности од енкодерите на двата мотори на DaNI. Излезните вредности се во rad/s и имаат *double* формат, и заради тоа можат директно да се прикажат на *indicator*, во *chart* да се претставаат, или да се вклучат во други пресметки или контролери.



Слика 6.5: VI за отчитување од ултразвучниот сензор

Отчитување од ултразвучниот сензор:

Овој блок служи за отчитување на информацијата дадена од ултразвучниот сензор, монтиран врз серво мотор на предниот дел од DaNI. Отчитаните вредности се исто во формат *double* и се во мерна единица метри.



Слика 6.6: VI за насочување на ултразвучниот сензор

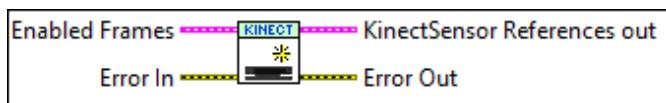
Насочување на ултразвучниот сензор:

Со овој блок се управува серво моторот на DaNI, односно се одредува поставеноста на ултразвучниот сензор. Блокот како влезови ги прими објектот на DaNI и посаканиот агол во радијани. Право пред роботот се смета за 0, позитивни вредности го вртат сензорот на лево, додека негативни вредности го вртат сензорот на десно. Доменот на сензорот е $\pm \frac{\pi}{2}$, или $\pm 90^\circ$.

6.2 LabVIEW MakerHub модули

6.2.1 Kinect модул

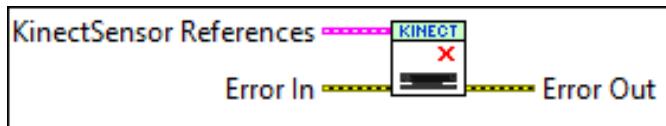
Модулот од NI MakerHub за отчитување од Kinect-от во LabVIEW ги сублимира во LabVIEW блокови сите функции за аквизиција на податоци кои што обично би се програмирале во јазик како C++ или C#. Во оваа дипломска работа, од значење се следните функции:



Слика 6.7: VI за Иницијализација на Kinect-от

Иницијализација:

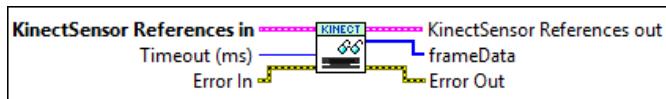
Слично како и кај блокот за иницијализација за роботот, оваа функција на излез дава референции кон објектот што ја претставува самата камера, и сигнал на грешка. Во случајот на Kinect-от, нема потреба за доведување на било какви идентификациони информации ако се употребува само една камера.



Слика 6.8: VI за Деиницијализација на Kinect-от

Деиницијализација:

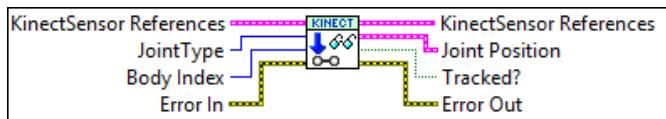
Потполно исто со роботската деиницијализација, оваа функција ги прима референциите кон објектот и ги анулира, затворајќи ја комуникационата врска со камерата.



Слика 6.9: VI за Отчитување на информација во рамка

Отчитување од Рамка:

Овој потинструмент се нарекува и „полиморфски“ инструмент, бидејќи флексибилен е во однос на излезниот сигнал, без било какви промени на влезот. Оваа функција му дозволува на корисникот да одбери од која „рамка“ да отчитува. Под рамка се подразбира вид на информација, односно избира корисникот да оберби бојна, длабинска, инфрацрвена, или „тесесна“ слика. Важно е да се спомени дека телесната слика на излез не дава никаква слика или матрица, туку низа со димензија 1×6 чиј секој елемент е податок од тип *body cluster* (телесна соединица) кој ги содржи координатите на телото во 3 разни координатни системи - бојна камера (т.е. пиксели), длабинска камера (т.е. пиксели), и вистинската камера (т.е. метри), како и некои други корисни мета-информации.



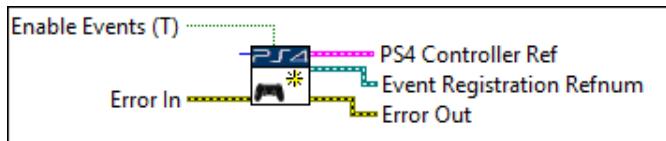
Слика 6.10: VI за Отчитување на зглобните координати на озбраната личност

Отчитување на Зглобни Координати:

Функцијата за отчитување на координатите на самите зглобови ги прима како влез индексот од 0 - 5 на телото и една енумерирана вредност која го претставува самиот зглоб. Иако не се доведува низата што се генерира од отчитувањето на телесната слика, оваа функција сепак има пристап до зглобовите. На излез го дава соодветниот *cluster* на посаканиот зглоб.

6.2.2 PS4 контролер модул

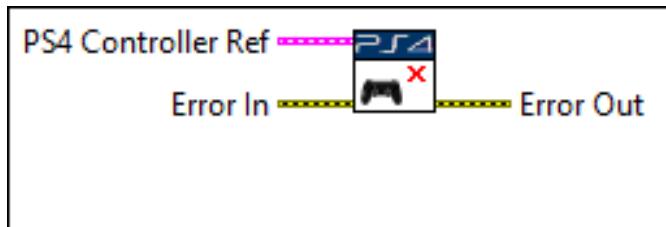
MakerHub интерфејсот за PS4 контролерот овозможува лесно отчитување на податоци од сите копчиња и оски од PS4 контролер. Функционира според стандардниот модел за физички уреди на LabVIEW - има блок за иницијализација, отчитување и деиницијализација, а има и вградена можност за детекција на настани, т.е. промена на состојбата на копчињата.



Слика 6.11: VI за Иницијализација на *bluetooth* контролерот

Иницијализација:

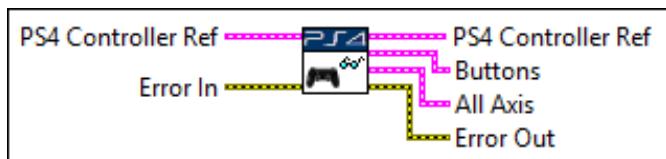
Стандарден блок за иницијализација на уред, во овој случај PS4 контролерот. По желба може да се избере дали да се иницијализира првиот конектиран уред, или во случај на повеќе уреди, може да се избере специфичен контролер со автоматски зададен индекс. Како излез од блокот повторно се добиваат референции кон објектот и сигнал на грешка.



Слика 6.12: VI за Деиницијализација на *bluetooth* контролерот

Деиницијализација:

Иста функција и влезови како и горенаведените блокови за деиницијализација.

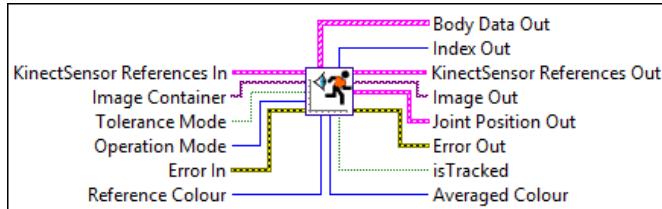


Слика 6.13: VI за отчитување на наредби од контролерот

Отчитување:

Оваа функција за отчитување има две излезни порти: првата порта има ги дава отчитуваните вредности на копчињата, а втората порта се користи за детекција на промени (*Event Detection*) во состојбата на голем број од влезовите. Оваа порта понатаму се поврзува со настанска структура (*Event Structure*) и го поедноставува процесот на интегрирање на контролата во програмата.

6.3 Специјално изработени Виртуелни Инструменти и Контроли



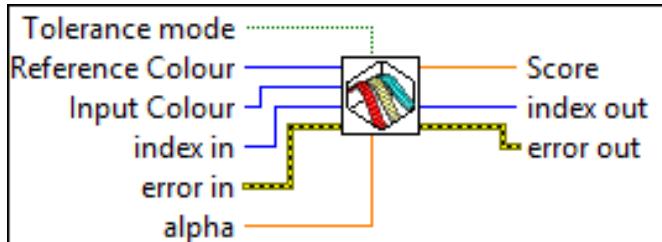
Слика 6.14: Главен пресметковен VI за човеково препознавање и избор

6.3.1 humanDetect

Функцијата *humanDetect* служи за препознавање на човек и има два режими на операција. Првиот режим, наречен *desperate* или **очаен**, ги дава на излез декартовите координати на првата личност која ќе биде детектирана.

Вториот режим, наречен *Colour Match* или **пребарување според боја**, врши избор на личност за следење според анализа на бојата во една претходно одредена област или регион. Областа, во овој случај, претставува посебен четриаголник на пиксели што припаѓа на секоја од личностите. Оваа област може да се наоѓа било каде и да има било кој облик; во оваа функција, таа област се наоѓа во горниот дел (градниот кош) на човекот. Се проверува сличноста на овие области со некоја референта боја, и се избира за следење тој човек со најсоодветна обоена област. Анализата на овие области ќе биде подетално покриена во описите за други потинструменти.

6.3.2 colourDist



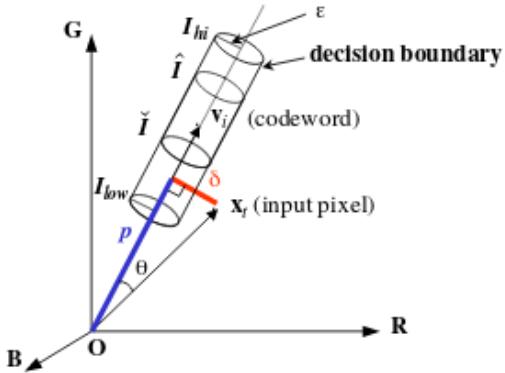
Слика 6.15: Потфункцијата *colourDist* за обработка на бојна информација

Подфункцијата *colourDist* се наоѓа во инструментот за препознавање *humanDetect* и ја врши анализата на обоените области. Оваа функција исто има два режими на работа со два вида на толерантни полиња на боја. На влез е доведена просечната боја на една област, и функцијата проверува дали оваа просечна боја припаѓа на едно од претходно дефинирите толерантни полиња, кое се наоѓа во дискретниот тридимензионален боен декартов систем (поглавје 5.4). Толерантното поле проверува за сличност според хроматичност (состав на боја), но може да е и проширено за да прифаќа еден поширок опсег на осветлувања.

Режим - Толерантен цилиндар:

Првиот режим на работа вклучува толерантно поле во облик на цилиндар, каде што радиусот на цилиндарат претставува „разлика на боен состав“ (хроматично отстапување), додека неговата висина го претставува опсегот на прифатени интензитети на осветленост. Толерантниот цилиндар како начин на споредба на бои е земен од трудот на К. Ким

[2]. Сликовито е покажен овој режим во сл. 6.16. Равенките за пресметка на хроматично отстапување, добиени од [2], гласат:



Слика 6.16: Графички приказ на толерантниот цилиндар од [2]

отстапување, добиени од [2], гласат:

$$\begin{aligned} \|\mathbf{x}_t\|^2 &= R^2 + G^2 + B^2, \\ \|\mathbf{v}_{ref}\|^2 &= R_{ref}^2 + G_{ref}^2 + B_{ref}^2, \\ (\mathbf{x}_t \cdot \mathbf{v}_{ref})^2 &= R R_{ref} + G G_{ref} + B B_{ref} \\ colourDist(\mathbf{x}_t, \mathbf{v}_{ref}) &= \delta = \sqrt{\|\mathbf{x}_t\|^2 - \frac{(\mathbf{x}_t \cdot \mathbf{v}_{ref})^2}{\|\mathbf{v}_{ref}\|^2}} \end{aligned} \quad (12)$$

Додека равенката за осветлувачко отстапување гласи:

$$\Delta I = \|\mathbf{x}_t\| - \|\mathbf{v}_{ref}\| \quad (13)$$

Условот за припаѓање на некоја обоена област i е:

$$\delta_i < \epsilon \text{ и } I_{min} \leq \Delta I_i \leq I_{max} \quad (14)$$

Каде ϵ , I_{min} , и I_{max} се константи зададени - т.е. нагодувани - од програмерот.

Доколку точката што ја претставува бојата (во декартовиот боен систем) припаѓа на овој цилиндар, личноста на која ѝ припаѓа оваа боја е ставена во низата на можни кандидати за следење.

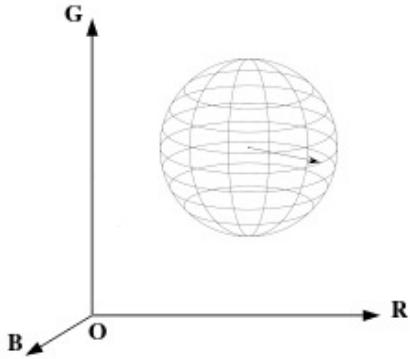
Толерантниот цилиндар има мала толеранција при хроматични отстапувања, но е мошне толерант на разлики во осветлување, што значи дека подобро функционира во случаи каде осветлувањето може да се менува.

Режим - Толеранта сфера:

Вториот режим на работа користи толерантно поле во облик на сфера (сл. 6.3.2), каде што нејзиниот волумен ги зафаќа и дозволените бојни состави и интензитети. Се пресметува тридимензионален вектор од просечната боја на доведената област до референтната боја, односно до центарот на сферата. Доколку овој вектор е целосно опфатен во сферата, личноста на која ѝ припаѓа оваа боја станува кандидат за следење.. Пресметката за релативниот вектор гласи:

$$\mathbf{V}_i^r = \mathbf{V}_r - \mathbf{V}_i = [R_r - R_i, G_r - G_i, B_r - B_i] \quad (15)$$

Каде \mathbf{V}_i е векторско прикажување на влезната боја, а \mathbf{V}_r е векторското претставување на референтната боја. Должина на релативниот вектор V_i^r се пресметува со нормата на



Слика 6.17: Графички приказ на сферичното толерантно поле

истиот ($\|\mathbf{V}_i^r\|$):

$$\|\mathbf{V}_i^r\| = \sqrt{R_i^r{}^2 + G_i^r{}^2 + B_i^r{}^2} \quad (16)$$

Ако релативният вектор е помал или еднаков со радиусот на сферата:

$$\|\mathbf{V}_i^r\| \leq \epsilon \quad (17)$$

Каде што ϵ е радијусот на сферата, точаш се вели дека векторот целосно припаѓа во сферата и соодветната личност станува кандидат за следење.

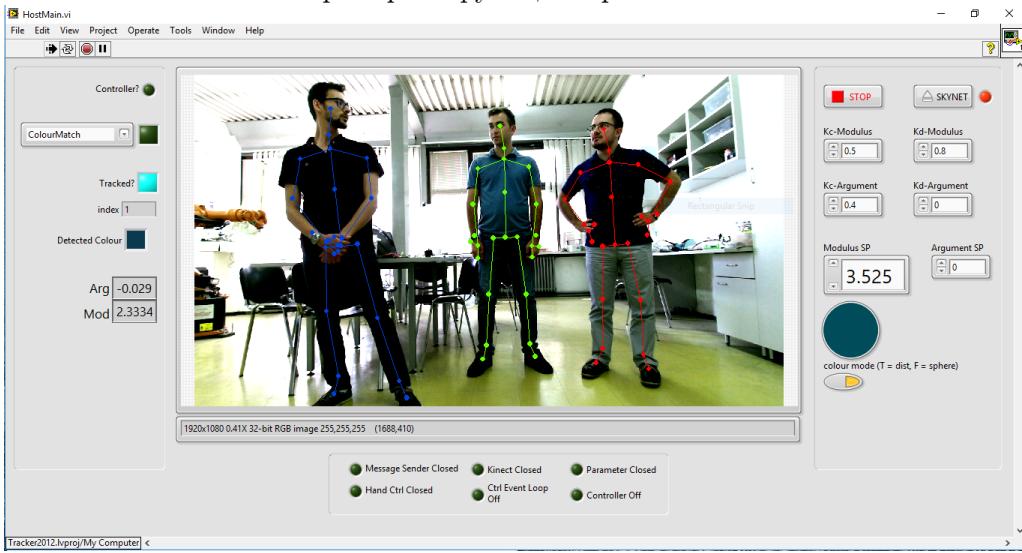
По споредување на бојните области на сите детектирани личности, сите кандидати за следење се сублимирани во една низа. Според видот на анализа, се наоѓа најзадоволителната (т.е. минималната) вредност на низата, и личноста на која ѝ припаѓа оваа вредност е избрана за следење. Во случај анализата да била цилиндрична, параметарот на сортирање би бил хроматично отстапување, односно близина до оската на цилиндартот занемарувајќи го осветлувањето. Доколку видот на анализа да бил сферичен, параметарот кој се споредува би бил долнината на векторот помеѓу центарот на сферата и влезната боја.

Како излез на функцијата е индексот на личноста која треба да се следи.

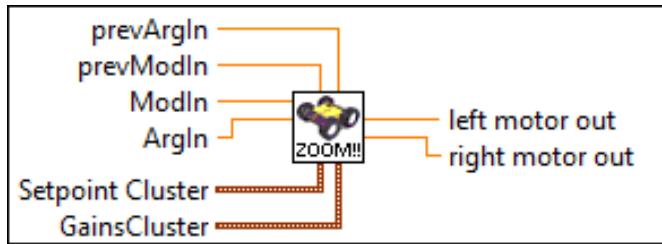
Толерантата сфера, за разлика од цилиндартот, не е соодветен за употреба во ситуации со променливо осветлување. Бидејќи сферата сешири во сите три димензии, таа има средна толеранција за осветлувачки разлики и средна толеранција на хроматично отстапување. Поради оваа, сферата е погодна за детектирање на сложени нијанси на бои, особено при непрецизно човеково нагодување на бојата.

Во сл. 6.18 се прикажани три личности со разни бои на горна облека. Модусот на операција е colourMatch, што се гледа во горно левиот агол на еcranот. Толерантниот волумен е цилиндар, чии избор е прикажан и управуван од најдолното копче од десната страна, доколку самата боја на препознавање се нагодува со кругот над толерантното копче. Kinect-от ги има препознаено трите личности, но има одлучено да го следи човекот во средината. Оваа информација е пренесена до корисникот со светилката *Tracked?*, бројниот покажуваш *index*, и бојата прикажана во покажувачот *Detected Colour*. Во случај да немало задоволителна личност во сцената, *Tracked?* би била исклучена, и *index* би покажувал код на грешка -2. Доколку ниту една личност да не е детектирана, *index* би вратил вредност -1.

Слика 6.18: Пример на функционирањето на colourMatch



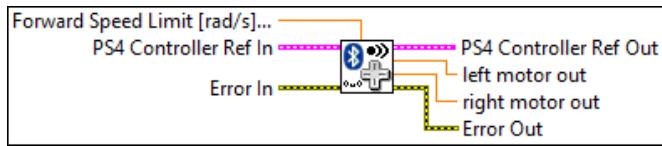
6.3.3 differentialDrive



Слика 6.19: Управувачот *differentialDrive*

Блокот *differentialDrive* го претставува главниот контролер во самиот систем, и го поседува двојниот ПД управувач претставен во поглавје 5.3.2. Тој на влез ги прима поларните координати на избраната личност и на излез ги дава потребните брзини што требат да се впишат во моторите за да се постигне саканата положба, односно да конвергираат референтниот и реалниот вектор еден кон друг.

6.3.4 Телеоперација



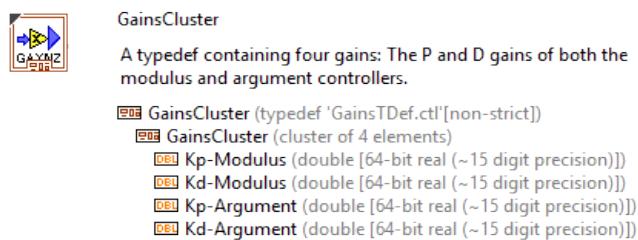
Слика 6.20: Блок за управување преку *bluetooth*

Блокот за телеоперација има интерфејс со блокот за отчитување на *bluetooth* контролерот, кој на излез ги дава положбите на аналогните влезови и состојбите на бинарните влезови. Овие влезови се претворени преку аналоген-дигитален претворувач во 16 битен број, и потоа преку два специјално изведени управувачи во линиска брзина и сооднос на распределување на оваа брзина помеѓу двета мотори, односно овозможување

на попрецизна контрола и движење со влезови различни од минималните и максималните вредности.

6.3.5 Typedefs

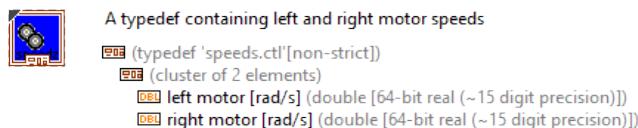
Во програмата се појави потреба на засебно групирање на податоци во определени структури (односно *clusters*). Овие структури претставуваат логични собири на податоци, како на пример информации за засилувањата или брзините на моторите. Исто е важно да овие структури да се строго дефинирани, унифицирани преку целиот софтвер. Согласно овие барања произлзе изборот за употреба на *typedefs*, односно композитни информациони структури со одредени дефиниции кои можаат да се повикаат на сличен начин како обичните типови на податоци како што се `int32` и `double` [28].



Слика 6.21: Typedef за засилувања

Засилувања:

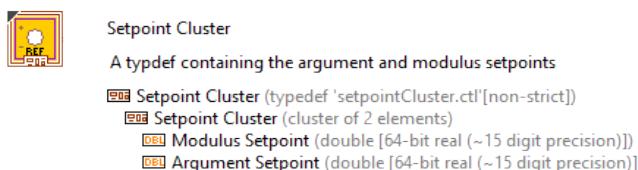
Typedef-от за засилувањата на системот ги содржи четирите засилувања потребни за двојниот ПД управувач: пропорционални и диференцијални засилувања за управување и на модулусот и на аргументот.



Слика 6.22: Typedef за брзини

Брзини:

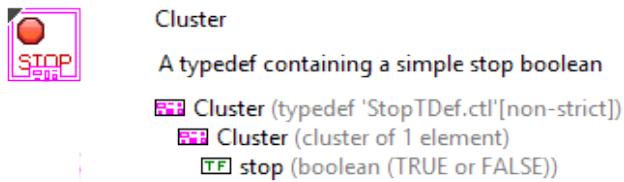
Овој typedef ги содржи конкретните брзини на моторите, кои се праќаат до роботот при режим на телеоперација.



Слика 6.23: Typedef за референтни вредности

Референтни Вредности:

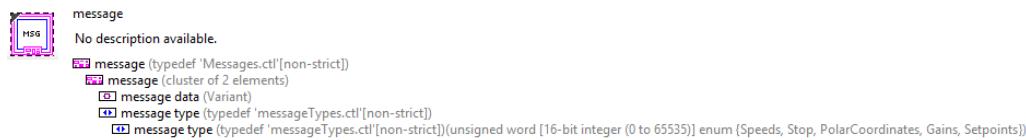
Референтните вредности се содржат во облик на горенаведениот `typedef`.



Слика 6.24: `typedef` за сопирање на роботот

Сопирање на Роботот:

`typedef`-от за сопирање на роботот содржи само еден `boolean`, но сепак е корисен при употребата на мрежен тек (поглавје 6.4).

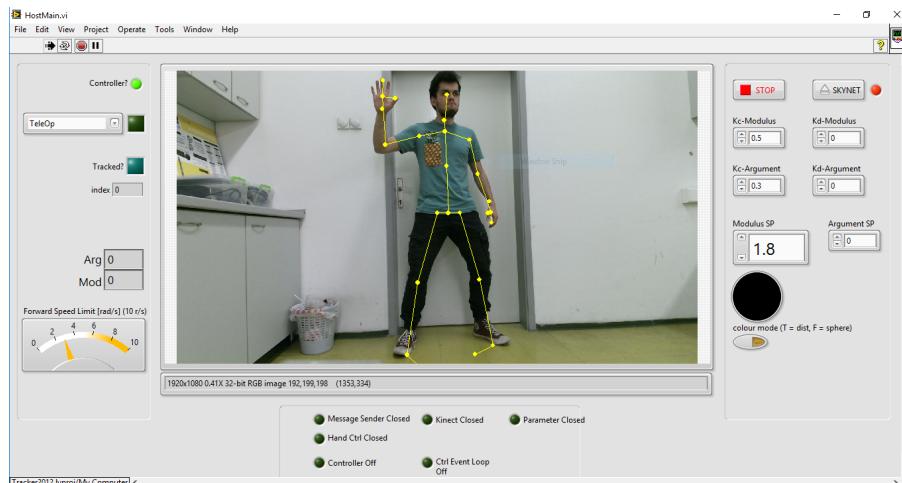


Слика 6.25: `typedef` за пораки

Порака:

Можеби најважниот `typedef`, пораката содржи два елементи: еден блок општа меморија кој може да содржи било кој вид на податок (*variant*, и етикета која кажува кој вид на информација е во варијантата. Ова е многу корисно бидејќи ја отстранува потребата за повеќе од еден мрежен тек - понатаму образложено во поглавје 6.4.

6.4 Образложение на Софтверот

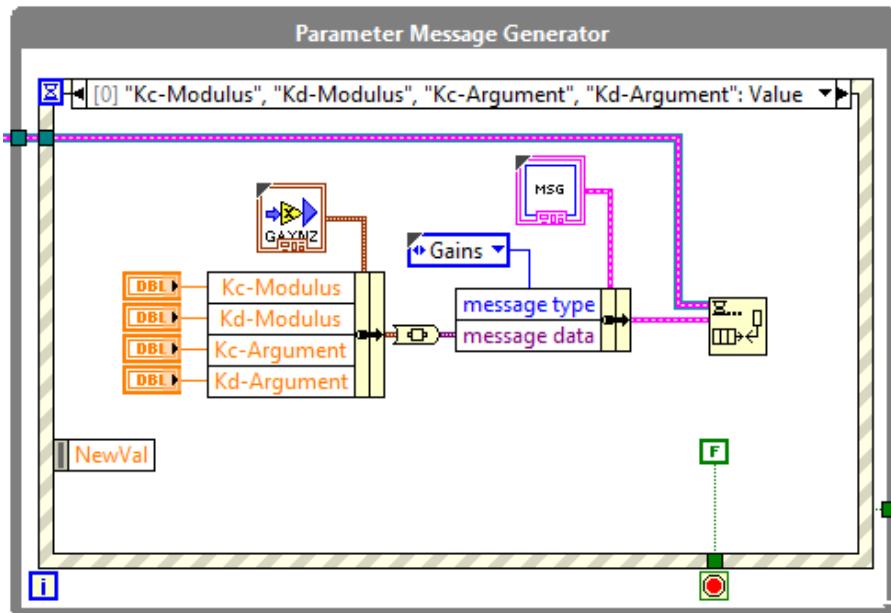


Слика 6.26: Главниот интерфејс за корисници на програмата, со видео од што гледа роботот

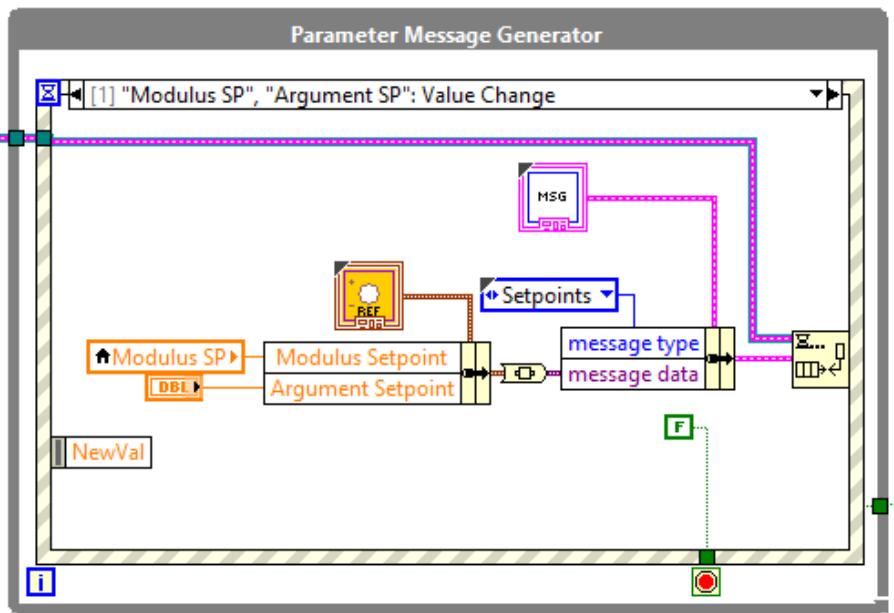
Следи изведбата во LabVIEW на етапите на функционирање описани во поглавје 5.1. Целиот софтвер се состои од два програми кои работат паралелно на две соседни машини: персоналниот компјутер (главна процесорска единица) и вградениот sbRIO компјутер на DaNI роботот. Како што се гледа погоре во слика 6.4, софтверот има целосен графички интерфејс за корисникот кој пружи корисна информацијата за моменталната состојба на програмата и роботот. На главната единица работат повеќе паралелни *while* рутини, како што се:

Произведувачи на Параметри:

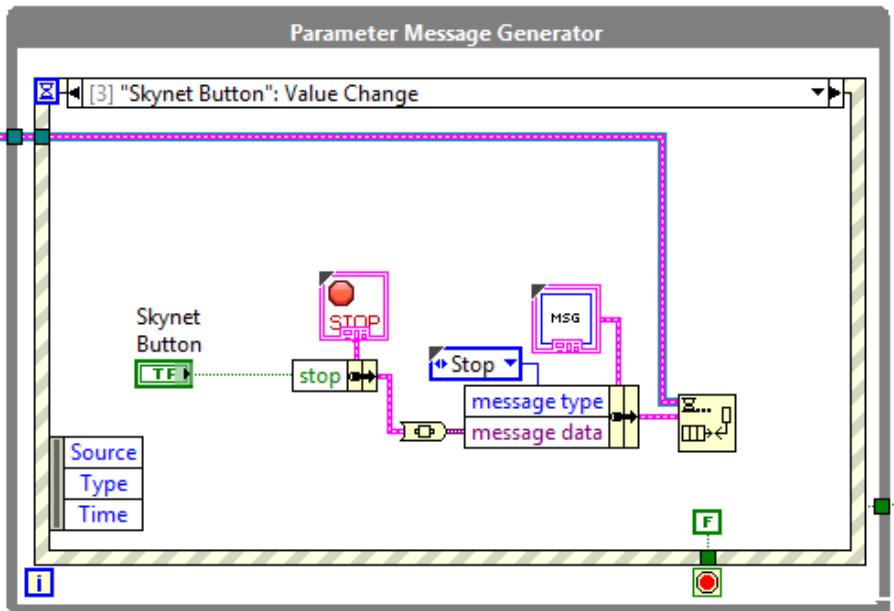
Три произведувачи на параметарски пораки се состојат во една настанска структура (сл. 6.27 - 6.29), чекајќи да се зададе промена на некој параметар од страна на корисникот и да се постави новата информација на челото на редот за пораки кои ќе се праќаат до роботот.



Слика 6.27: Произведувач на Пораки - Засилувања



Слика 6.28: Произведувач на Пораки - Рефентни Вредности

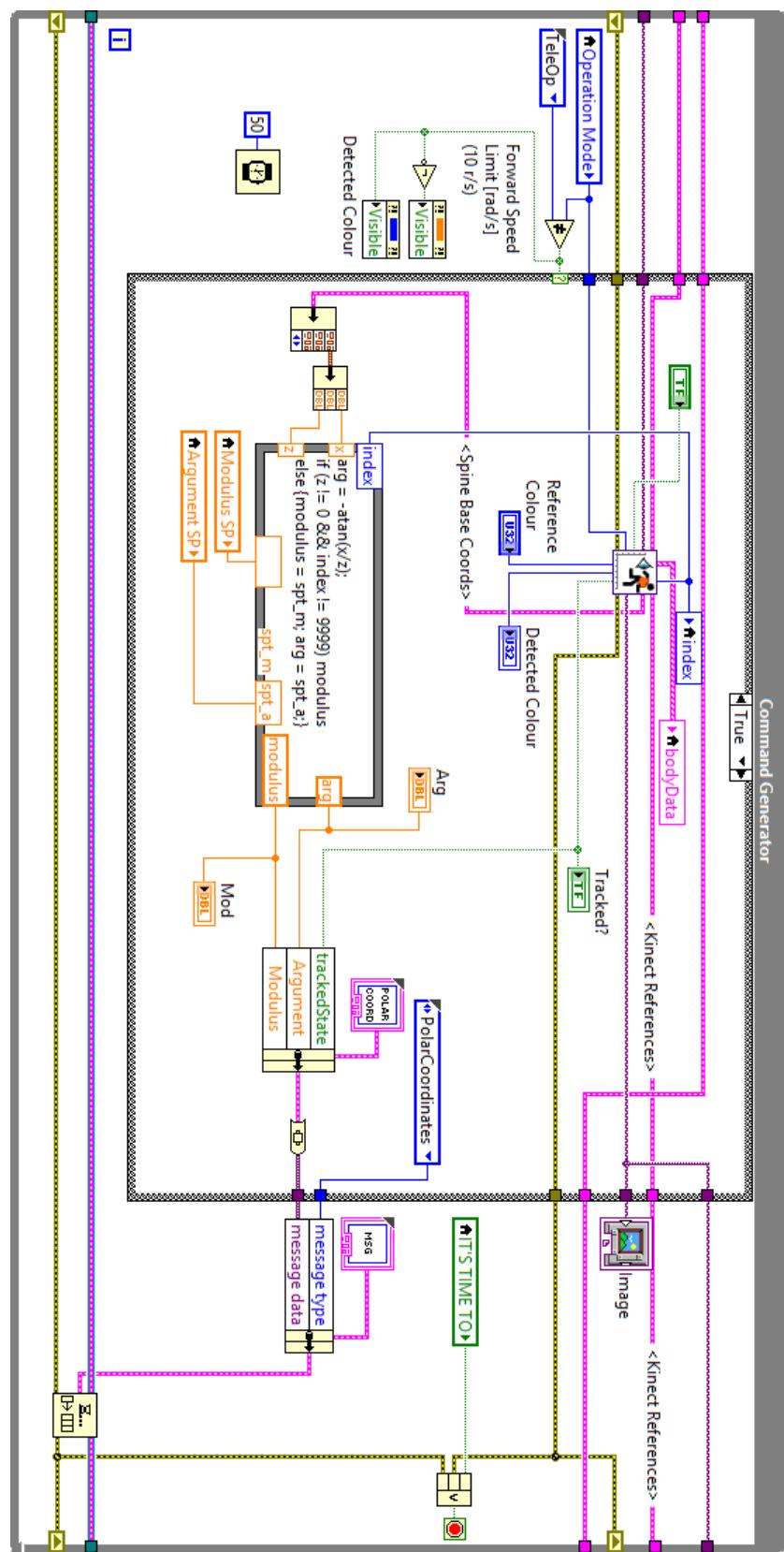


Слика 6.29: Произведувач на Пораки - Запирање на Роботот

Овие произведувачи работат во настанска структура (*event structure*) [28], односно реагираат само кога ќе има промена во таа вредност за која се одговорни.

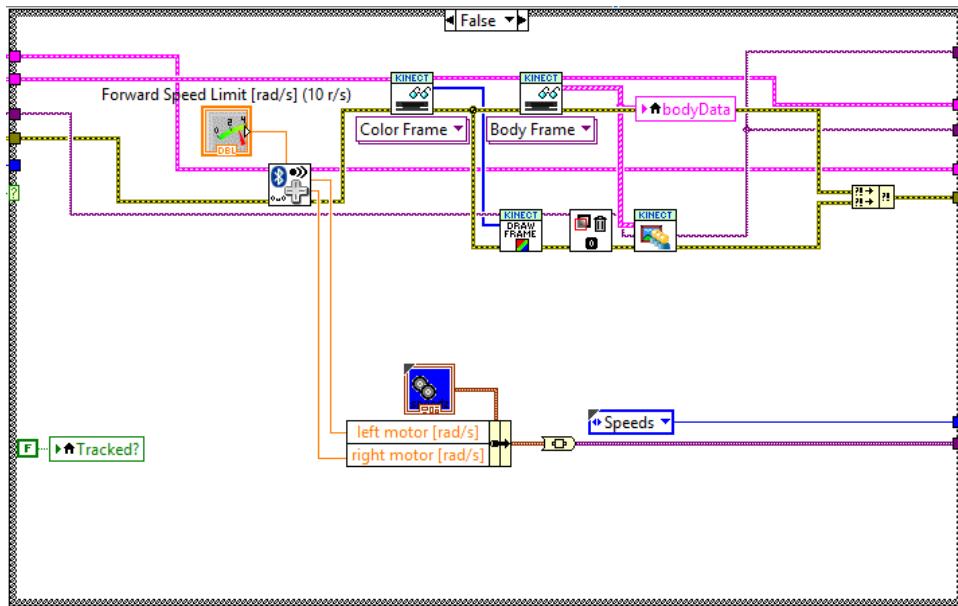
Произведувач на Движечки Наредби:

Произведувачот на движечки наредби е главниот дел од програмата, и во зависност од режимот на работа одбран од корисникот, тој функционира на различни начини. Доколку корисникот избере режим на телеоперација, роботот се управува со *bluetooth* контролерот како било кое друго возило со радио управување. Доколку корисникот избере еден од режимите на автономна работа, роботот ќе ја изврши соодветната функција (поглавје 6.3), при која на излез ќе добие соодветни наредби за движење.



Слика 6.30: Произведувач на наредби при автономно движење

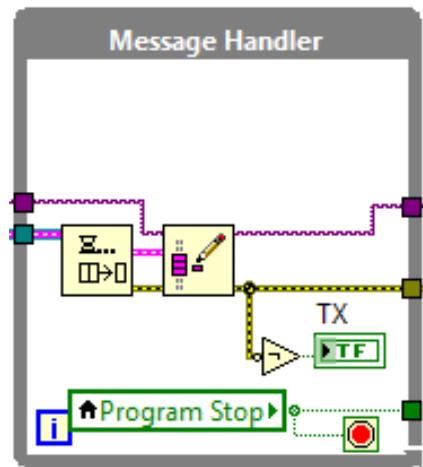
Независно од тоа како се добиени овие наредби, тие се пакуваат во формата на општата порака и се поставуваат во редот за пораки.



Слика 6.31: Произведувач на наредби при телеоперација

Управувач на Пораки:

Сите пораки се ставени во еден ред на податоци, создаден со *Queue* функцијата. Секој произведувач ги става своите пораки во овој ред. Наредбите за движење се ставаат на крајот на редот, додека другите се сметаат за итни и се ставаат на челото на редот. Структурата за раководење со пораките (сл. 6.32) има за задача да ги отчитува пораките од челото на редот и да ги испраќа преку мрежниот тек (*Network Stream*) до роботот. Мрежниот тек е комуникациски интерфејс кој припаѓа на LabVIEW, и поради тоа тој има поддршка за сите негови видови на податоци, вклучувајќи и *typedefs*. Тоа значи дека, за разлика од поопшти комуникациски протоколи како TCP/IP и UDP, овој протокол нема потреба од припрема на податоци пред испраќање, ниту посебно толкување при примање.



Слика 6.32: Управувачот на пораки

Но, еден недостаток е тоа што секој мрежен тек може да пренесува само еден вид на податок, пр. `uint16` или `string`. За ова да се заобиколи, се употребува специјалниот **typedef** **порака** (поглавје 6.3.5). Поради структурата податок-етикета, пораката може да се прилагоди за пренесување на било кој вид на податок, вклучувајќи и други *typedefs*.

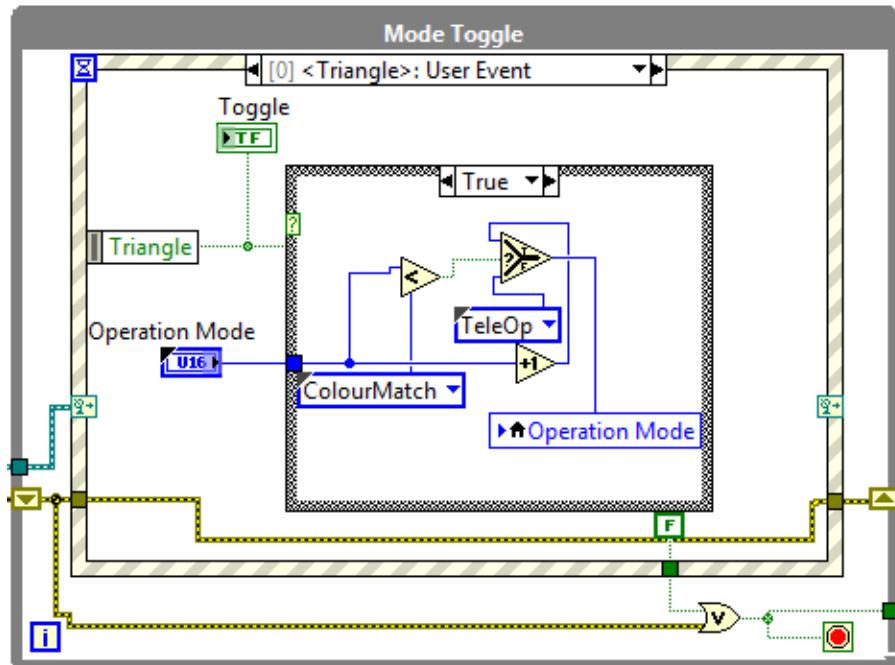
доколку примателот има копија од дефиницијата на испратениот typedef, и на овој начин со еден мрежен тек се пренесуваат сите посакани податоци.

Толкувач на *Bluetooth* Наредби:

Со употреба на погоре споменатите библиотеки и една настанска структура, се толкуваат податоците од контролерот, како што се промената на режим и итното застанување и прекинување на роботот.

Со притискање на триаголник копчето на контролерот, во настанската структура се препознава промена на состојбата на тоа копче, и со тоа, преку логиката се врши инкрементација на енумератор-от задолжен за пратење на режимот на работа. При тоа:

- TeleOp соодветствува на вредност 0 и претставува почетна (односно *default* режим),
- Desperate соодветствува на вредност 1,
- ColourMatch соодветствува на вредност 2.



Слика 6.33: Толкувач на *Bluetooth* Наредби

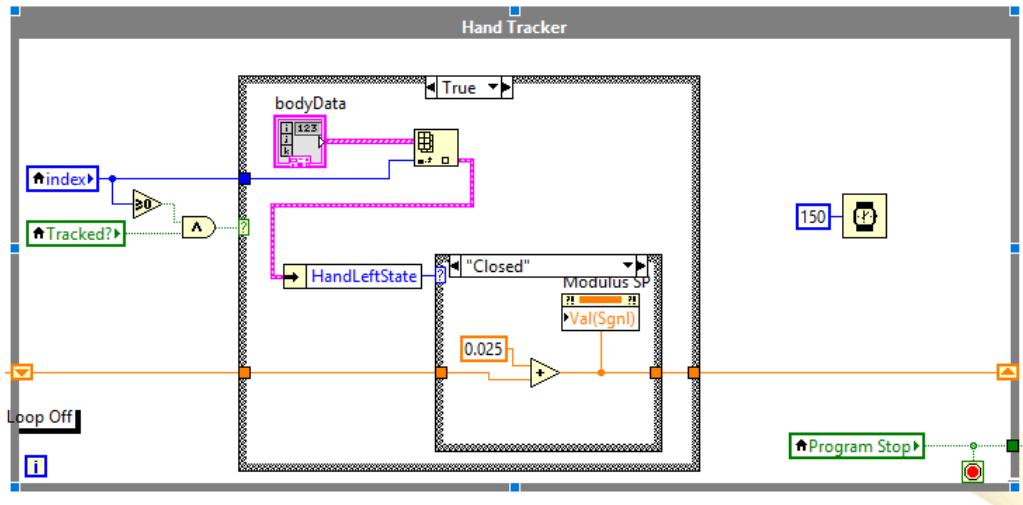
Кога енумераторот (во овој случај пратен со локална променлива) достигне вредност која соодветствува на ColourMatch, тогаш се ресетира на почетната вредност.

Исто така, во настанската структура се вклучени и две други копчиња, *Share* и *Options*, кои соодветно се користат за сопирање на програмата на компјутерот и програмата на роботот. При тоа треба да се внимава роботот да биде првиот којшто ќе биде запрен, бидејќи командата за запирање се испраќа преку мрежниот тек.

Читач на рачни гестикулации:

Рутината за отчитување на рачни гестикулации е активна во секој циклус, меѓутоа врши отчитување само во циклуси во кои е препознана и се следи личност. Индексот на таа личност се користи како показател за да се добијат информациите за состојбата на телото на таа личност. Од нив, се користи само податокот за состојбата на левата дланка, т.е.

HandLeftState. Овој податок има три можни вредности, но во оваа рутина се користат само отворена (*Open*) и затворена (*Closed*) состојба. Читачот за рачни гестикулации е прикажан во 6.34.



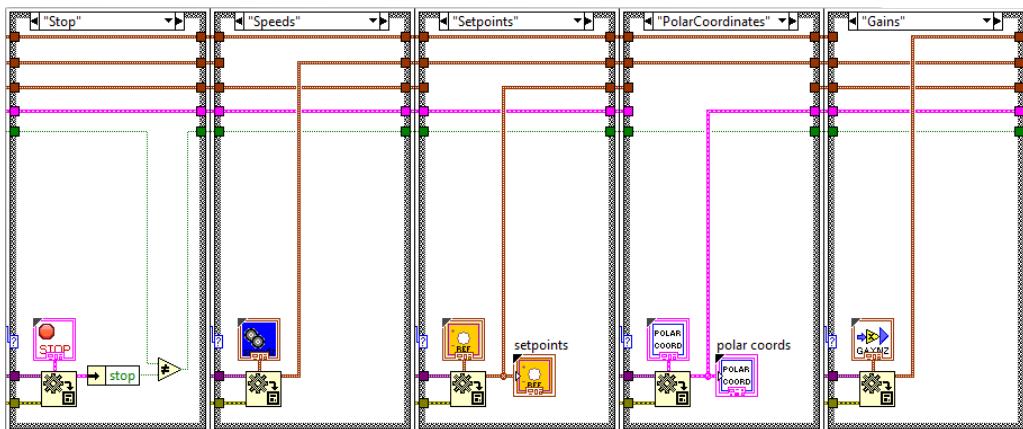
Слика 6.34: Читач на рачни гестикулации

Во зависност од состојбата на левата дланка, се врши или инкрементација или декрементација на референтното растојание кое треба да се оддржува до личноста која се следи. Инкрементацијата и декрементацијата се со вредности од $0.025m$ при секој успешен циклус.

Во оваа работа, роботот всушност ги има полесните задачи. Задачата на роботот е да ги прими пораките што му стигнуваат преку мрежниот тек, да ги толкува, и да одлучи како да се однесува според содржината на тие пораки.

Читач на пораки:

Читачот на пораки има задача да ги прими пораките, кои се во формат на „порака“ `typedef` (поглавје 6.3.5), и да ги разреди во своите две компоненти - етикетата и содржината.



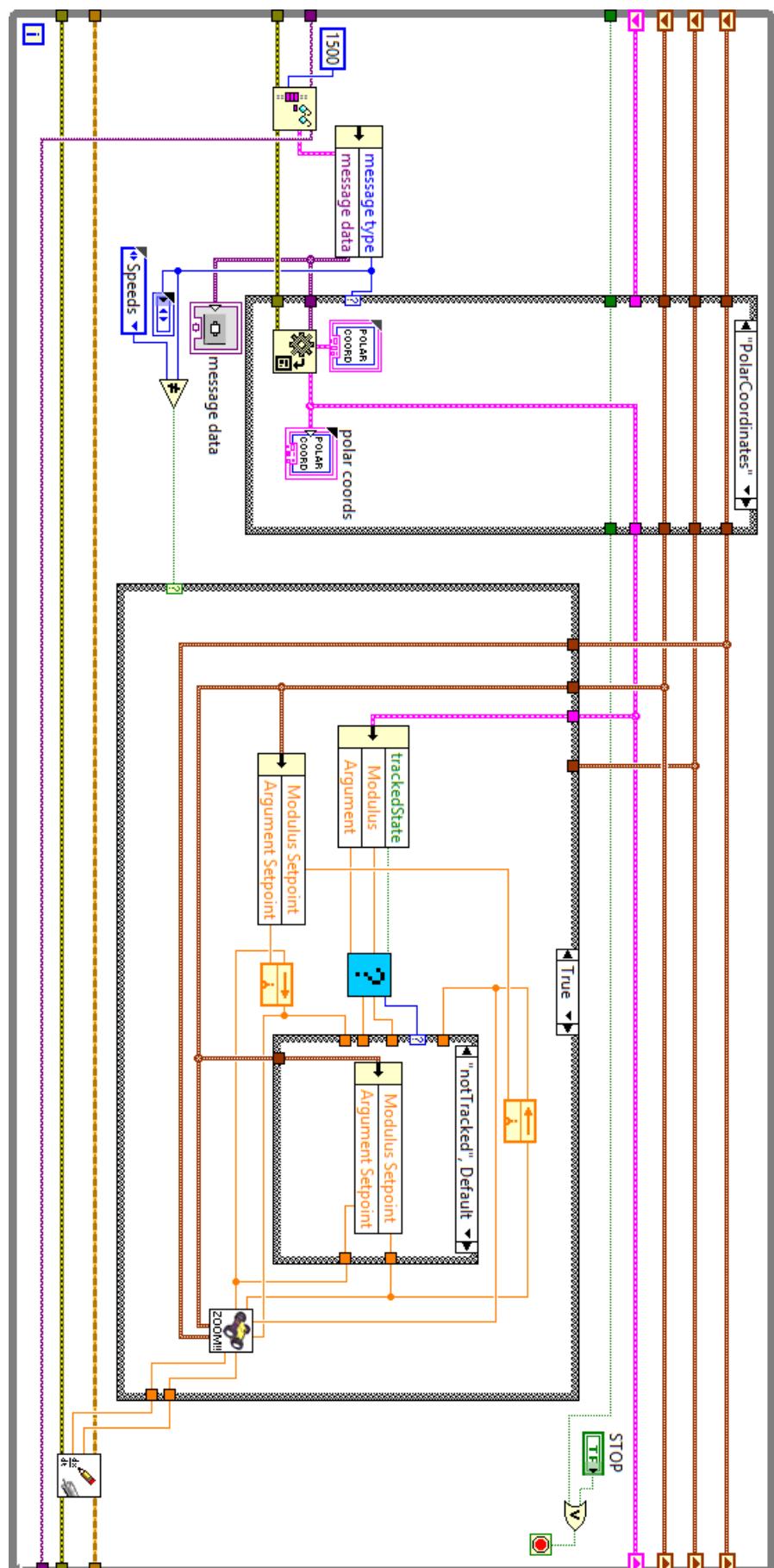
Слика 6.35: Читач на пораки

Етикетата на пораката е енумерирана вредност што е исто еден `typedef` - односно флексибилно се додаваат и одземуваат етикети. Енумерацијата содржи во неа по една етикета/вредност за секој можен вид на податок што може да стигне од компјутерот до

роботот. Оваа етикета се доведува до одлучувачот (*case selector*) на читачот - прикажан на сл. 6.35 - и според тоа функцијата *variant to data* ќе ја толкува содржината на доведената информација на соодветен начин.

Главен Циклус на Роботот:

Алгоритамот што работи на роботот е, како претходно споменато, прилично едноставен. Тој има улога да ги прими и толкува пораките со читачот, и доколку добие порака со координати на некоја личност - дали пораката е исправна или не употребувајќи ја специјално изработената функција *logicBox*. Резултатот од оваа проверка се испраќа до двојниот ПД регулатор *differentialDrive* и брзините се „впишуваат“ до моторите со соодветната функција од NI Robotics модулот. Главниот работен циклус на роботот е прикажен во сл. 6.36.



Слика 6.36: Главниот циклус на роботот

7 Заклучок

При завршувањето на проектот, може да се извлече заклучок дека системот речиси ги задоволува барањата наведени како цел. Пресметковните барања се задоволени - брзото препозавање на личности и нивно филтрирање според боја беа постигнати - а системот е и теоретски исправен со добра брзина на реагирање на наредбите од корисникот. Но, системот поседува една клучна маана којашто ја руши можноста за реалистична и робустна операција на истиот - при нагло или брзо движење на Kinect-от, алгоритамот за препозавање не успева да ја толкува состојбата на поместената сцена пред себе и се „губи“ личноста која се следи (како што е споменато во поглавје 4.1.2). Како последица, роботот застанува и повторно се обидува да препознае и да избере личност за следење.

Како резултат на оваа маана, операцијата е раскршена а дури и разочарувачка. Идна работа на овој проект би го вклучило оттргнувањето на овој проблем. Можностите за оттргнување лежат најверојатно во полнијата на сензори и софтвер. Во полето на софтвер проблемот со губење на личност можеби би се решил со употребата на некои филтрирачки алгоритми или некој сосем нов алгоритам за препознавање, додека во полето на сензори може да се подобри функционирањето со употребата на еден поинаков и посоодветен сензор. На пример, би можело да се употреби стерео-камерата ZED од StereoLabs, која добива слична длабинска слика на Kinect-от преку метод на споредба на сликите извадени од две соседни камери со позната константна оддалеченост.

Од позитивна страна, решенијата пружени од оваа дипломска - препознавањето според бојна сфера (поглавје 6.3.2) и паралелниот ПД контролер (поглавје 5.3.2) - сосема добро функционираат, и можеби би нашле примена во идна работа.

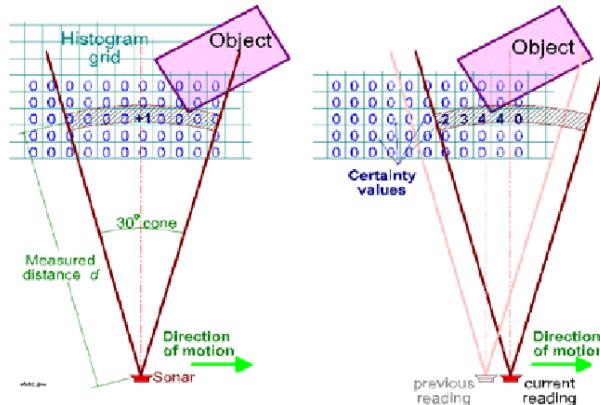
Анекс I - Избегнување пречки со хистограм на векторски-полиња

Методот на хистограм на векторски полиња (ХВП) се состои од намалување (кратење) на податоци во две фази, наспроти постарите техники кои користат само еден чекор во кратење на податоци. Постојат три нивоа на претставување на податоци:

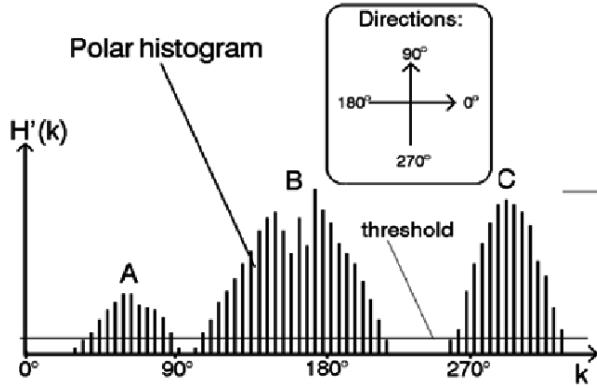
- a. Највисокото ниво содржи детален опис за околината во која се наоѓа роботот. Во ова ниво, дводимензионалниот декартов (*Cartesian*) хистограм **C** се надополнува со нови податоци во реално време.
- b. Второто, средишно ниво е едно димензионален поларен хистограм (**H**) кој се гради околу моменталната локација на роботот. **H** содржи n аголни сектори со одредена ширина.
- c. Последното ниво, најниското ниво, од претставување на податоци е излезот од ХВП алгоритамот.

За разлика од други имплементации, имплементацијата на ХВП во LabVIEW користи само локален поларен хистограм, а не декартов хистограм. Овој недостаток е исправен со вградување на енкодерот од кој што се добиваат информации за далечина и правец.

Најпрво се создава дводимензионален хистограм околу роботот, што ги претставува пречките, чии положбени информации се добиваат од сензорот. Вториот чекор е претворувањето на дводимензионалниот хистограм во еднодимензионален хистограм и потоа во поларен хистограм. Најпосле, алгоритамот го означува широкиот сектор со најмало поларно отстапување од средишната линија, и ги пресметува аголот и брзината потребни за да се стигне до тој сектор.

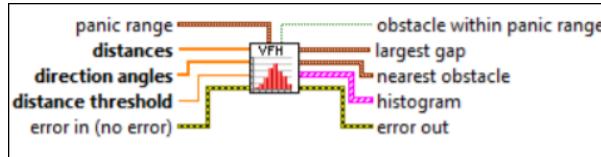


Слика 7.1: Конструкција на 2D хистограм карта од [30]



Слика 7.2: Репрезентација на 1D и поларен хистограм од [30]

Принципот на работа е всушност избирањето на најголемата празнина (сектор) преку податоците од поларниот хистограм, од каде што се пресметуваат правецот, насоката и оддалеченоста до таа празнина.

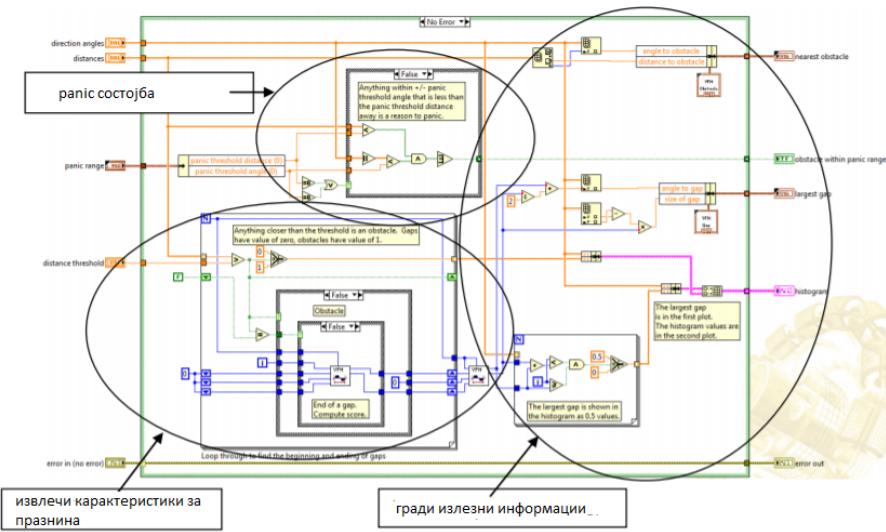


Слика 7.3: ХВП блок во LabVIEW

Влезовите *Distances* (растојанија) и *Direction angles* (агли на насока) се добиваат од податоците кои што ги отчитува сензорот. *Distances* е низа од податоци за далечините на пречките. *Direction angles* аналогно е низа со агли кои што започнуваат од центарот на сензорот до секоја од далечините (аглите десно од центарот се означени како позитивни а оние лево од центарот како негативни вредности). *Distance threshold* е влез кој што дефинира незначителни пречки односно роботот игнорира објекти кои што се на далечина поголема од зададениот *distance threshold*.

При тоа, објектите кои што ги детектира сензорот кои се наоѓаат на растојание и агол помали од *panic threshold distance* и *panic threshold angle* се идентификуваат како пречки кои што треба итно да се избегнаат.

Излезот *nearest obstacle* е структура од податоци која се добива како група од два елементи - агол и растојание до најблиската пречка. Блок дијаграмот на ХВП е даден на сл.7.4.



Слика 7.4: Блок дијаграм на Алгоритам за Избегнување

За да се одреди дали одредена далечина претставува празнина или пречка, секој елемент се става во *for loop* со индексирање при што се врши споредба со *distance threshold*. На крај излегува булова вредност (1 или 0) при што 0 се доделува на далечини кои што се поголеми од *distance threshold*, а 1 за помали далечини. Резултатот е низа со нули и единици која што се менува како што роботот се движи во својата околина. Споредбата на временски соседни излези е влез во една изборна структура ги споредува сите празнини да ја најди таа со најдобра „оценка“. Ако вредноста на моменталната празнина е подобра од минатата, се јавува **true** со што новата празнина станува најдобра и е новиот излез, а ако е обратно добиваме **false** каде што претходните податоци на големина и офсет (реден број на вредноста во низа) се најдобра празнина и се сеуште излез.

На крај на оваа пребраување се добиваат соодветните информации за најдобрата празнина, и роботот се насочува и движи кон неа.

Анекс II - Линкови до Github страни на изворните кодови

На следниот линк може заинтересираниот читач да го најде целосиот код за примерот од поглавје 3. Кодот е напишан во C++ во Visual Studio 2017 Community и има потреба од openCV библиотеката, како и SDK-то на Kinect-от.

https://github.com/markogalevski/DaNI_Kinect2_Tracker

Целата програма за дипломската работа напишана во LabVIEW 2012 може да се спушти од следниот линк.

<https://github.com/markogalevski/Bachelors-LabVIEW>

Потребно е корисникот да ги има следните пакети:

- OS: \geq Windows 8
- LabVIEW 2012 (32bit)
- LabVIEW Robotics модул
- LabVIEW FPGA модул
- LabVIEW Real-Time модул
- LabVIEW Vision Development модул
- LabVIEW NI RIO Drivers
- \geq .NET4.0
- Kinect 2.0 SDK
- VI Package Manager - MakerHub Kinect 2 и MakerHub PS4 Controller модули

Користена литература

- [1] J. Steward et al. "Performance Assessment and Calibration of the Kinect 2.0 Time-of-Flight Range Camera for Use in Motion Capture Applications". In: (2015).
- [2] K. Kim et al. "Real-time foreground-background segmentation using codebook model". In: (2005).
- [3] Peter Kirby Allen. "Differential Drive Robotics". In: () .
- [4] Tim Bower. *Robotics Programming Study Guide*. URL: http://faculty.salina.k-state.edu/tim/robotics_sg/.
- [5] Peter Corke. *Robotics, Vision, and Control*. Springer, 2011.
- [6] Roque Trindade Deise Maia. "Face Detection and Recognition in Color Images under Matlab". In: (2016).
- [7] PITSCO Educational. *Tetrix Max DC Motor*. URL: <https://www.pitsco.com/TETRIX-DC-Gear-Motor>.
- [8] Dimension Engineering. *Sabertooth Dual 10A Motor Driver*. URL: <https://www.dimensionengineering.com/products/sabertooth2x10>.
- [9] Harry Fairhead. *Kinect's AI breakthrough explained*. URL: <http://www.i-programmer.info/news/105-artificial-intelligence/2176-kinects-ai-breakthrough-explained.html>.
- [10] Michael Jenkin Gregory Dudek. *Computational Principles of Mobile Robotics*. Cambridge University Press.
- [11] National Instruments. *NI sbRIO 96xx User Guide*. URL: <http://www.ni.com/pdf/manuals/375052c.pdf>.
- [12] National Instruments. *Starter Kit 2.0 VIs*. URL: http://zone.ni.com/reference/en-XX/help/372983B-01/lvrobovi/starter_kit_2_pal/.
- [13] Y. Koren J. Borenstein. *The Vector Field Histogram - Fast Obstacle Avoidance For Mobile Robots*.
- [14] Dr. Robert King. *Mobile Robotics Experiments with DaNI*. Colorado School of Mines.
- [15] Donald E. Knuth. "Fundamental Algorithms". In: Addison-Wesley, 1973. Chap. 1.2.
- [16] Jennifer Leong. *Number of Colors Distinguishable By The Human Eye*. URL: <https://hypertextbook.com/facts/2006/JenniferLeong.shtml>.
- [17] Larry Li. "Time-of-Flight Camera - An Introduction". In: (2014).
- [18] Araki M. "Control Systems, Robotics, and Automation Vol. II". In: 2009. Chap. PID Control.
- [19] Microsoft. *Kinect for Windows SDK*. URL: [https://docs.microsoft.com/en-us/previous-versions/windows/kinect/dn799271\(v%3dieb.10\)](https://docs.microsoft.com/en-us/previous-versions/windows/kinect/dn799271(v%3dieb.10)).
- [20] Didier Stricker Oliver Wasenmueller. "Comparison of Kinect v1 and v2 Depth Images in Terms of Accuracy and Precision". In: () .
- [21] Nancy Owen. *Kinect for Windows SDK v2 Sample Program*. URL: <https://github.com/UnaNancyOwen/Kinect2Sample>.
- [22] Parallax. *PING))) Ultrasonic Distance Sensor*. URL: <https://www.parallax.com/sites/default/files/downloads/28015-PING-Sensor-Product-Guide-v2.0.pdf>.
- [23] Dushan Petrovich. *Automatskog Upravlivanja Interna Skripta I PPT Prezentacija*.
- [24] Raveon. *FPGA-Field Programmable Gate Array*. URL: <http://www.rfneulink.com/FPGA-whitearticle.htm>.

- [25] International Federation of Robotics. *Robot History*. URL: <https://ifr.org/robot-history>.
- [26] Servocity. *HS-485HB Servo*. URL: <https://www.servocity.com/hs-485hb-servo>.
- [27] Gunnar Svaetchin. “Spectral response curves from single cones”. In: (1956).
- [28] Jeffrey Travis. “LabVIEW for Everyone”. In: 2006. Chap. 7, 13.
- [29] Thomas Young. “Bakerian Lecture: On the Theory of Light and Colours”. In: (1802).
- [30] Muhammad Zohaib et al. “Control Strategies for Mobile Robot With Obstacle Avoidance”. In: 3 (June 2013), pp. 1027–1036.