LATVIJAS UNIVERSITĀTE

DATORIKAS FAKULTĀTE

KVADROTORA SADARBĪBA AR BEZVADU SENSORU TĪKLU

MAĢISTRA DARBS

Autors: **Klāvs Taube**

Studenta apliecības Nr.: kt10022

Darba vadītājs: pētnieks Mg. dat. Artis Gaujēns

RĪGA 2016

ANOTĀCIJA

Darbs ir par kiberfizikālajām sistēmām, kas ir strauji augoša datorikas apakšnozare. Pēdējos gados kvadrotoru un bezvadu sensoru tīklu tehnoloģijas ir vienas no nozīmīgākajām kiberfizikālajās sistēmās. To attīstības rezultātā ir radusies interese par šo tehnoloģiju apvienošanu vienā lietojumā. Darbā ir izpētīti dažādi kvadrotora un bezvadu sensoru tīkla sadarības piemēri, kā arī nepieciešamās tehnoloģijas to realizācijai. Darbs ir daļa no *Artemis R5-COP* projekta, un tas ir izstrādāts *LUMII* Reālā laika sistēmu laboratorijā. Darbā ir apskatītas *LUMII* pieejamā kvadrotora, bezvadu sensoru tīkla un *MATLAB Simulink* iespējas un dažādas lokalizācijas metodes. Darba ietvaros ir izstrādāts demonstrācijas testa piemērs balstoties uz izpētītajiem sadarbības piemēriem un tehnoloģijām.

Atslēgvārdi: kiberfizikālās sistēmas, kvadrotors, bezvadu sensoru tīkli, *MATLAB Simulink*

ABSTRACT

COLLABORATION BETWEEN QUADCOPTER AND WIRELESS SENSOR NETWORK

The subject of this work is referring to cyber-phyisical systems, which is a rapidly growing field of embedded computing. In the recent yeards quadcopter and wireless sensor network technologies have became one of the most significant cyber-physical systems. That is why interest in collaboration between quadcopter and wireless sensor networks has emerged. In this work various examples of collaboration between quadcopter and wireless sensor networks and necessary technologies for implementation of said examples have been researched. The work is a part of *Artemis* project *R5-COP* and it has been created in Real Time Systems Laboratory of *IMCS* of University of Latvia. The available quadcopter and wireless sensor network, various methods of localization, as well as *MATLAB Simulink* has been researched. A demonstration test example has been developed which relies on the researched examples of collaboration and technologies.

Keywords: cyber-physical systems, quadcopter, wireless sensor networks, *MATLAB Simulink*

AUTOREFERĀTS

Darbā ir izpētīti, apvienoti un izstrādāti nepieciešamie līdzekļi kvadrotora sadarbībai ar bezvadu sensoru tīklu, lai balstoties uz tiem nākotnē varētu nākotnē izstrādāt kādu reālu sadarbības piemēru.

Darbā ir izmantoti 30 literatūras avoti, no kuriem 17 ir no ražotāju un izstrādāju dokumentācijas, 2 ir projektu apraksti, 1 ir no Latvijas Republikas Tiesību aktu datubāzes, 10 ir zinātniskie raksti un grāmatas.

Darba problēmas un risinājumi ir aprakstīti detalizēti. Darbā veiktajā praktiskajā daļā ietilpst bezvadu sensoru tīkla tehnoloģijas vadības, un kvadrotora saskarņu vadības programmatūras projektēšana un izstrāde. Balstoties uz izstrādātajiem risinājumiem, tiks turpināta *R5-COP* projekta darbība. Risinājumi ir testēti izmantojot darbā izstrādāto demonstrācijas testa piemēru.

Darba struktūra un noforējums atbilst metodiskajām prasībām. Viss, kas ir aizgūts no citiem autoriem, ir atzīmēts ar attiecīgām literatūras atsaucēm.

SATURA RĀDĪTĀJS

APZĪMĒJUMU SARAKSTS

Apzīmējums	Apraksts
Kvadrotors	Lidaparāts, kura kustību nodrošina četri pa pāriem pretējos virzienos rotējoši propelleri.
Bezvadu sensoru tīkls	Telpā izvietots autonomu sensoru kopums, kuri veido tīklu.
Bezvadu sensoru tīkla mezgls	No angļu *node* - iekārta, kas ir daļa no bezvadu sensoru tīkla.
Bezvadu sensoru tīkla vārteja	No angļu *gateway* - iekārta, kura nodrošina bezvadu sensoru tīkla savienojumu ar citu tīklu, piemēram, internetu.
Unity3D	Vairāku platformu atbalstošs speļu dzinis
IQRF	*MICRORISC s.r.o* izstrādāta bezvadu sensoru tīklu platforma, kas ir paredzēta mazas jaudas, ātruma un datu apjoma saziņai.
Robotu operētājsistēma	Elastīgs satvars robotu programmatūras izstrādei. Dažādu rīku, bibliotēku un metodiku krājums robotikas problēmu risināšanai.
AscTec Pelican	*Ascending Technologies GmbH* ražots kvadrotors.
AscTec Mastermind	*Ascending Technologies GmbH* ražots kvadrotora borta dators.
Ubuntu Linux	Populāra *Linux* operētājsistēmas distribūcija.
IDE	No angļu *Integrated Development Environment* - integrētā izstrādes vide.
USB	No angļu *Universal Serial Bus* - universālā seriālā kopne.
UART	No angļu *Universal Asynchronous Receiver/Transmitter* - universālais asinhronais raiduztvērējs.

SPI	No angļu *Serial Peripheral Interface* - seriālā perifērijas saskarne.
Python	Plaši izmantota augsta līmeņa, universāla programmēšanas valoda.
CDC	USB komunikācijas iekārtas klase, no angļu *Communication Device Class.*
SIM	No angļu *Subscriber Identity Module* - abonenta identifikācijas modulis.
DPA protokols	IQRF platformas saziņas protokols, no angļu *Direct Peripheral Access.*
Datu sapludināšana	No angļu *Data Fusion.* Datu apstrādes process, ar kuru no dažādu avotu datiem var iegūt precīzākus, lietderīgākus un konsekventākus datus.
ROS	No angļu *Robot Operating System* - robotu operētājsistēma
BST	Bezvadu sensoru tīkls
HWP	No angļu *Hardware Profile*
GPS	No angļu *Global Positioning System*
RSSI	No angļu *Received Signal Strength Indication*
UAV	No angļu *Unmanned Aerial Vehicle*
UAS	No angļu *Unmanned Aircraft System*

RUAV	No angļu *Rotorcraft Unmanned Aerial Vehicle*
GCS	No angļu *Ground Control Station*
IMU	No angļu *Inertial Measurement Unit*
SDK	No angļu *Software Development Kit*

IEVADS

Darbs pieder kiberfizikālo sistēmu apakšnozarei, kura strauji attīstas sakarā ar dažādu mobilo tehnoloģiju attīstību. Kiberfizikālās sistēmām ir tādi pielietojumi kā transporta sistēmas, veselības aprūpes sistēmas, gudrās mājas un ēkas, sociālie tīkli un spēles, plānošana, enerģētikas sistēmu vadība, mākoņskaitļošana un datu centri, energotīkli, tīklošanas sistēmas, novērošanas sistēmas, industriālo procesu kontrole, aviācijas vadība, meklēšanas dziņi, vides monitorēšana, civilā inženierija, video apstrāde, ūdens sadale un robotika(1). Eiropas Savienības zinātnes ceļakartē kiberfizikālās sistēmas ir viens no nozīmīgākajiem virzieniem, tajā ir gaidāms liels finansējuma pieaugums, piemēram *ECSEL* kopuzņēmumam(2).

Pēdējos gados no kiberfizikālajām sistēmām strauji attīstas divas ļoti aktuālas tehnoloģijas - civilie bezpilota lidaparāti un bezvadu sensoru tīkli. Bezpilota lidaparātus var izmantot meklēšanas un glābšanas, ārkārtas reaģēšanas, infrastruktūras atbalstīšanas, aerofotogrāfēšanas, mežu ugunsgrēku monitorēšanas, vides monitorēšanas, elektrotīklu monitorēšanas un citās misijās(3). Savukārt, bezvadu sensoru tīklu tehnoloģijas tiek izmantotas tādās jomās kā, piemēram, vides monitorēšanā, karadarbībā, bērnu izglītošanā, novērošanā, mikroķirurģijā, lauksaimniecībā un citur(4).

Maģistra darbs ir izstrādās *LUMII* Reālā laika sistēmu laboratorijā. *LUMII* Reālā laika sistēmu laboratorija ir strādājusi vairākos projektos kas saistīti gan ar kvadrotoriem, gan bezvadu sensoru tīkliem. *Artemis* projekta *R5-COP* ietvaros ir paredzēts realizēt kvadrotora sadarbība ar bezvadu sensoru tīklu. *R5-COP* (*Reconfigurable ROS-based Resilient Reasoning Robotic Cooperating Systems*) tiek izstrādāti uz *ROS* bāzēti robotikas risinājumi sadarbojoties industrijas lielajiem uzņēmumiem un vadošajām Eiropas zinātniskajām iestādēm.

Šī tēma ir interesanta, jo kvadrotors var papildināt bezvadu sensoru tīkla darbu, piemēram, var veikt tīkla mezglu apkalpošanu - identificēt mezglu, iegūt mezgla atrašanās vietu, ja katram mezglam nav pieejams *GPS*, mezgla programmatūras kļūdas gadījumā restartēt to u.c. Pasaulē ir realizētas vairākas sistēmas, kurās tiek veikta sadarbība starp bezvadu sensoru tīkliem un kvadrotoriem. Piemēram, kvadrotora izmantošanā izolētu bezvadu sensoru tīklu daļu savienošanai, ir veidots bezvadu sensoru tīkls izmantojot kvadrotorus, ir izstrādāta tehnoloģija datu savākšanai no bezvadu sensoru tīkla ar kvadrotoru, šie piemēri detalizētāk ir aprakstīti 2. nodaļas 1. apakšnodaļā. Sadarbība starp kvadrotoru un bezvadu sensoru tīklu var tikt pielietota pielietojumi precīzajā lauksaimniecībā, kas interesē Reālā laika sistēmu laboratorijai.

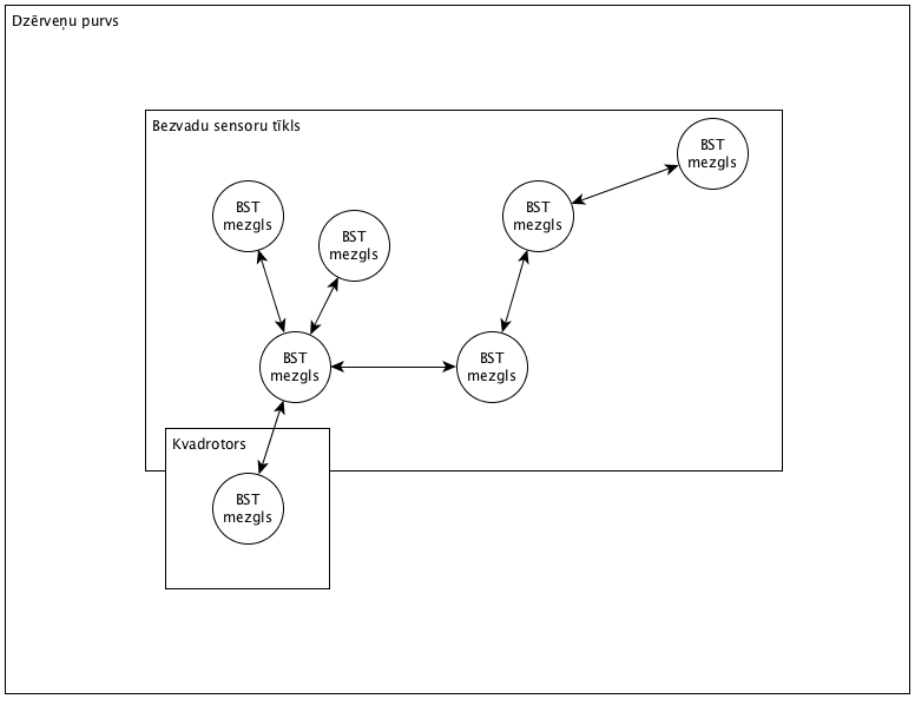
Darba uzdevums ir izveidot *RUAV*, kuru izmantojot varētu izstrādāt lietotnes sadarbībai ar bezvadu sensoru tīklu. Lai vienkāršotu izstrādi, uzdevums tiek realizēts iekštelpās, kā arī ir iekārtota vide testēšanai. Lai pārbaudītu darbību, ir izstrādāts demonstrācijas testa piemērs. Darbā ir izpētīts, kādas saskarnes ir nepieciešamas, lai šo piemēru realizētu un to atbalsts ir realizēts atbilstoši ražotāju specifikācijām, un tās ir pārbaudītss ar aprakstīto demonstrācijas piemēru.

Šajā darbā tiek apskatītas nepieciešamās komponentes, lai kvadrotors varētu sadarboties ar bezvadu sensoru tīklu. Kvadrotora misijas piemērā tas aplido bezvadu sensoru tīkla mezglus un atpazīst tos. Uzdevums ir paredzēts precīzās lauksaimniecības lietojumiem, kad bezvadu sensnoru tīkls ir ar ierobežotu fizisku pieejamību. Misija nodrošina iespēju vizuāli apskatīt grūti pieejamā vietā atrodošos bezvadu sensoru tīklu.

1. UZDEVUMA APRAKSTS

Tā kā *LUMII* Reālā laika sistēmu laboratoriju interesē kvadrotoru pielietojumi precīzajā lauksaimniecībā, noderīgs uzdevums būtu grūti pieejama tīkla apkalpošana. Piemēram, ja tīkls atrodas dzērveņu purvā un mezgli cilvēkam ir grūti sasniedzami, tad tos varētu sasniegt ar kvadrotoru, vizuāli atpazīt, pat nodrošināt darbības ar fizisku kontaktu. Lai varētu radīt *UAS*, kas šādu uzdevumu spētu izpildīt, pirmkārt, tiek veikta izstrāde un testēšana laboratorijas apstākļos iekštelpās. Lai to varētu izdarīt, tika secināts, ka no sākuma ir jāveic šādas darbības:

- jāveic kvadrotoru un bezvadu sensoru tīklu sadarbības risinājumu esošā stāvokļa izpēte;
- jāizpēta *LUMII* pieejamā kvadrotora, kā arī bezvadu sensoru tīkla iespējas un saskarnes;
- jāpiedāvā *LUMII* pieejamā kvadrotora un bezvadu sensoru tīkla savienošanas risinājums;
- jārealizē piedāvātais risinājums;
- jāizveido un jāpārbauda testa piemērs pieejamajos apstākļos. Tāpēc, ka āra apstākļos darbam ar kvadrotoru ir sezonāls raksturs, uzdevums tiek veikts iekštelpās. Tam ir izraudzīta un speciāli iekārtota telpa *LUMII* ēkā.



1. att. bezvadu sensoru tīkls un kvadrotors ārpus telpām

Projekta vadītāja definēts testa piemērs pārbaudei, kuram par pamatu ir ņemts *R5-COP* projekta plānotais demonstrācijas piemērs:

1. kvadrotors aizlido uz bezvadu sensoru tīkla mezglu pēc uzdotām *GPS* koordinātām;
2. kvadrotors veic plānotās darbības ar bezvadu sensoru tīklu.

Ierobežotajos iekštelpu apstākļos šis testa piemērs ir jāvienkāršo:

1. kvadrotors paceļas 1 metra augstumā virs zemes;
2. kvadrotors veic darbības ar bezvadu sensoru tīklu (tās varētu būt tādas, kuras ir viegli novērot no malas, piemēram, tīkla mezglu diožu ieslēgšana un izslēgšana).

Kritērijs veiksmīgai izpildei ir noteikums, lai kvadrotora un bezvadu sensoru tīkla vadība tiktu veikta ar vienu galveno algoritmu. *R5-COP* ietvaros vadības algoritms ir paredzēts *MATLAB Simulink Stateflow*, lai varētu izpildīt lielas sarežģītības lēmuma pieņemšanas (no angļu *Decision Making*) algoritmus.

Uzdevuma izpildē vajadzētu attūreties no sarežģītu kvadrotora lidojuma elementu izstrādes, jo tas ir cits *R5-COP* uzdevums, pie kura tiek strādāts paralēli, tāpēc šajā darbā sasniegtais kvadrotora vadības daļā maz der *R5-COP* izstrādē.

Lai risinātu nosprausto uzdevumu, ir nepieciešams apzināt sistēmas prasības. 1. att. ir redzama shēma ar kvadrotoru un bezvadu sensoru tīklu. Tajā ir redzams, ka kvadrotoram ir arī bezvadu sensoru tīkla mezgls – tas nozīmē, ka kvadrotors var komunicēt ar bezvadu sensoru tīklu. Tātad, lai būtu iespējams izpildīt šo uzdevumu, kvadrotoram ir jāspēj pārvietoties telpā, kā arī sazināties ar bezvadu sensoru tīklu. Šo uzdevumu ir iespējams pārcelt uz iekštelpām un sadalīt apakšuzdevumos. Uzdevums tiek uzskatīts par realizētu, ja sistēma ir spējīga:

- nodrošināt komunikāciju starp kvadrotoru un bezvadu sensoru tīklu;
- komunikācijai starp kvadrotoru un bezvadu sensoru tīklu ir jābūt nodrošinātai lidojuma laikā;

Uzdevuma apraksts iekštelpu testa piemēram:

- ir izveidots bezvadu sensoru tīkls;
- kvadrotors ir pievienots tīklam;
- kvadrotors spēj noturēt aptuvenu pozīciju telpas punktā;
- pa bezvadu sensoru tīklu ir iespējams sazināties ar kvadrotoru.

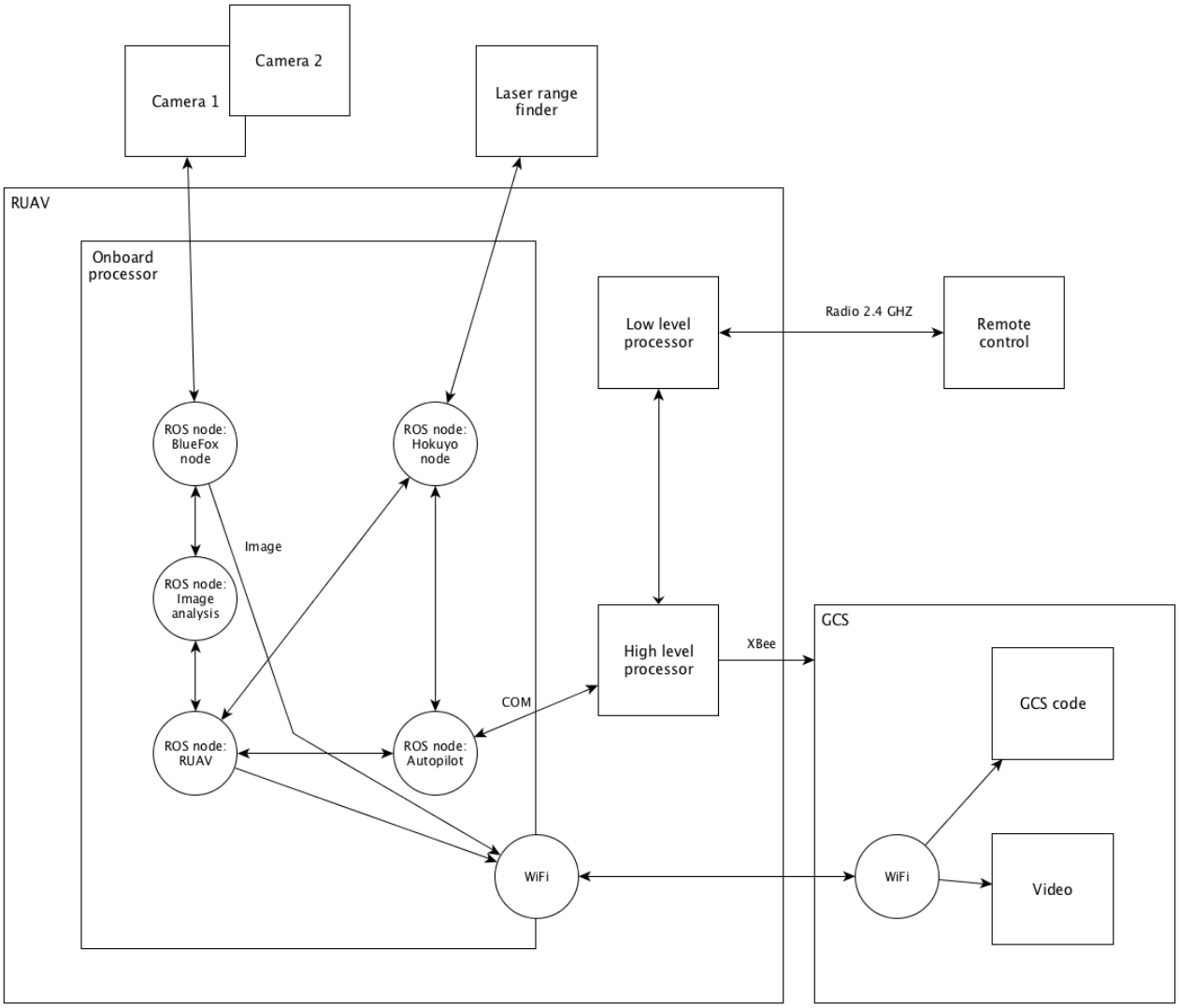
Uzdevuma pārbaudei ir izveidots šāds testa piemērs:

- kvadrotors paceļas 1 metra augstumā;
- izmantojot kvadrotora vadības programmu, pa vienai tiek ieslēgtas bezvadu sensoru tīkla gaismas diodes;
- pēc katras gaismas diodes ieslēgšanas, tiek uzņemts attēls;

- katra gaismas diode tiek izslēgta pirms tiek ieslēgta nākamā;
- kvadrotors nosēžas.

1.1. Uzdevuma saistība ar *R5-COP* projektu

Darbs ir veidots *R5-COP* projekta ietvaros. *R5-COP*, jeb *Reconfigurable ROS-based Resilient Reasoning Robotic Cooperating Systems* ir *Artemis* organizācijas projekts, kas

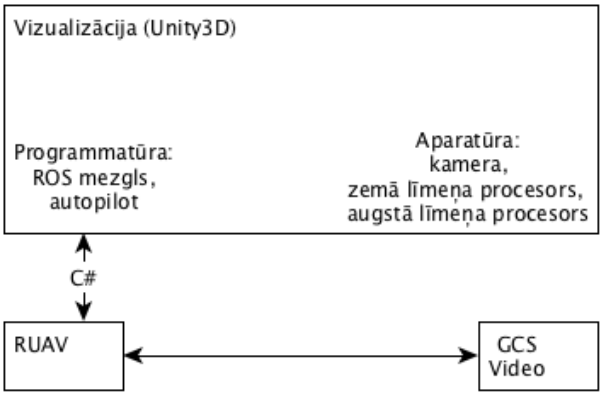


2. att. R5-COP RUAV shēma

koncentrējas uz spējām ražošanas paradigmām, specifiski uz modulārām robotikas sistēmām(5). Tā sākums bija 2014. gada februārī, beigas būs 2017. gada februārī, tā finansējums ir 13,1 miljoni eiro un tajā piedalās 30 partneri no 12 dažādām Eiropas valstīm(5). Viens no partneriem ir *LUMII*, ko pārstāv Reālā laika sistēmu laboratorija.

Reālā laika sistēmu laboratorijas uzdevuma pamatā ir izveidot autonomu sistēmu misijas vadības formāla apraksta izstrādes metožu un tehnoloģiju kopumu izmantojot *MATLAB Simulink* tehnoloģiju. Izstrādātās tehnoloģijas balstās uz *SIL* (no angļu *software-in-*

the-loop) metodoloģiju. Šobrīd aktīvi tiek strādāts pie misijas vadības formālā apraksta izstrādes, bet nākotnē, izmantojot to, ir plānots veidot demonstratoru, kurā būtu sadarbība starp bezvadu sensoru tīklu un kvadrotoru – līdz ar to tika izvēlēta šāda darba tēma. 2. Att. ir redzama kvadrotora vadības shēma, kāda tā tiek izstrādāta projekta *R5-COP* ietvaros, pagaidām vēl neņemot vērā sadarbību ar bezvadu sensoru tīklu. Lai veidotos sinerģija starp maģistra darbu un *R5-COP* projektu, tika nolemts, ka maģistra darbs tiks izstrādāts balstoties uz *R5-COP* izmantotajām tehnoloģijām un idejām.



3. att. SIL simulācija ar Unity3D vizualizāciju

Iepriekš tika izveidots bakalaura darbs šī paša projekta ietvaros, kurā tika izstrādāta vizualizācija kvadrotora misijas modeļa simulācijai. 3. Att. ir redzama aptuvena *SIL* modeļa shēma, kurā ir attēlota *Unity3D* izstrādātā vizualizācija un galvenās simulācijas daļas. Pagaidām bezvadu sensoru tīkls vēl nav ietverts *SIL* modelī, tāpēc nebija iespējams testēt šo izstrādi izmantojot to. Būtu bijis loģiski turpināt strādāt pie vizualizācijas un tajā integrēt bezvadu sensoru tīklu simulēšanas iespējas, bet projekta darbu loģistikas dēļ, ir bijis jāstrādā pie reālā kvadrotora saskarnēm. Vizualizācijas attīstīšana varētu būt viens no turpmākajiem darbiem šajā virzienā, tai varētu pievienot bezvadu sensoru tīklu simulāciju, kā arī *ROS* integrāciju.

Balstoties uz (skat. 2. att.) plānoto *R5-COP* shēmu, tika izstrādāta arhitektūra darba uzdevuma veikšanai. Ir secināts, ka vajadzīgie moduļi ir *ROS*. Tāpēc arī visi šajā darbā izstrādātie moduļi, ieskaitot bezvadu sensoru tīkla, ir veidoti *ROS*. Ir izveidoti šādi *ROS* risinājumi:

- jauns *ROS* mezgls bezvadu sensnoru tīkla vadībai;
- izstrādāts piemērs *ROS* vadības mezglam, kas veic uzdoto demonstrācijas testa piemēru;
- netika izstrādāts atsevišķs mezgls attēlapstrādei, jo vienkāršos uzdevumos vienkāršāk to ir iekļaut *ROS* vadības mezglā.

2. UZDEVUMA IZPĒTE

Darba mērķis ir izstrādāt līdzekļus, izpētīt un pielāgot jau esošus risinājumus kvadrotora sadarbībai ar bezvadu sensoru tīklu. Darba paredzētais rezultāts ir dažādu līdzekļu un metožu kopums kvadrotora sadarbībai ar bezvadu sensoru tīklu, piemēram, saziņas nodrošināšana ar bezvadu sensoru tīklu, kvadrotora vadība, dažādi attēlapstrādes paņēmieni, kas nepieciešami sadarbībai.

Lai kvadrotors varētu sazināties ar bezvadu sensoru tīklu, ir nepieciešams to padarīt par bezvadu sensoru tīkla daļu - mezglu. Tas nozīmē, ka tīklam ir jāatbalsta dinamiski (no angļu *mesh*) mezgli, jo kvadrotors kustēsies pa tīklu un tam ir jāvar sazināties ar citiem mezgliem.

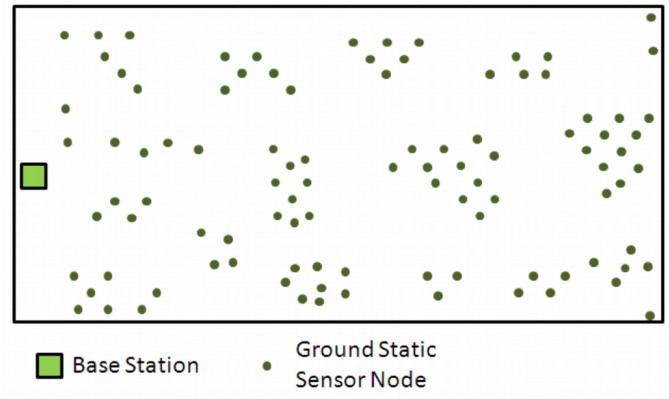
Kvadrotoram ir jāspēj autonomi pārvietoties pa telpu, kā arī tam ir jāspēj nonākt koordinātās, neizraisīt sadursmes ar apkārtējiem objektiem un pietiekami stabili jāspēj noturēties pozīcijā, lai varētu atpazīt bezvadu sensoru tīkla mezglus.

2.1. Esošie kvadrotoru un bezvadu sensoru tīklu sadarbības risinājumi

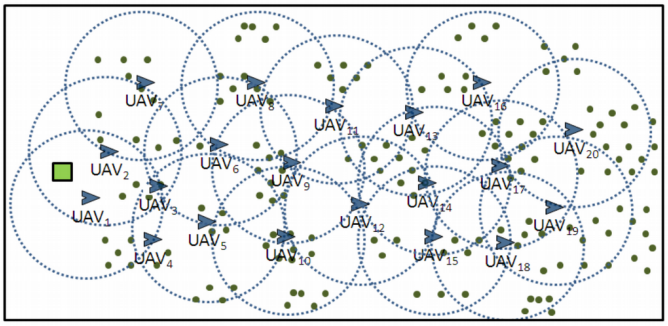
Izpētot, kas citur pasaulē ir veikts bezvadu sensoru tīklu un kvadrotoru sadarbības jomā, tiek secināts, ka tā ir populāra tēma. Ir daudz dažādu pielietojumu, kas risina reālas problēmas. Daži interesantākie uzdevumi un to risinājumi:

1. divu bezvadu sensoru tīklu savienošana ar kvadrotora palīdzību;
2. dinamiski jeb mobīli mezgli;
3. darbības attāluma palielināšana;
4. kvadrotors aizlido pie izolēta bezvadu sensoru tīkla, lai savāktu datus.

Pie bezpilota lidaparātu un bezvadu sensoru tīklu sadarbības ir strādājuši vairāki pētnieki. Kvadrotori ir izmantoti bezvadu sensoru tīkla daļu izolācijas jeb salu veidošanās problēmas risināšanā(6) (skat. 4., 5. att.). Šāds risinājums varētu būt visoperatīvākais datu savienojuma nodrošināšanai gadījumā, ja bezvadu sensoru tīkls sadalās (izveidojas kāda daļa,



4. att. Bezvadu sensoru tīkla piemērs ar izolētām mezglu salām(6)



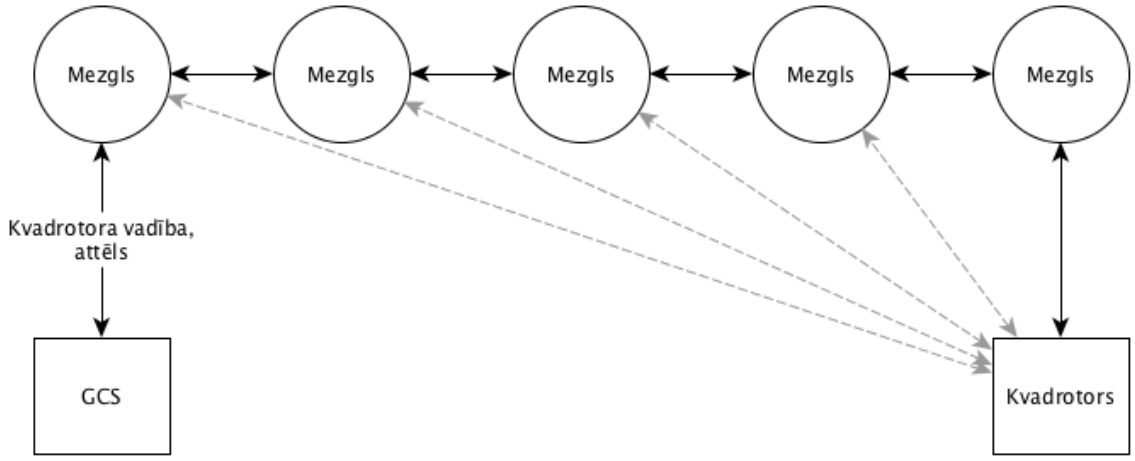
5. att. Mobīlu satekņu tīkls, kas atbalsta bezvadu sensoru tīkla savienojamību(6)

kura nespēj komunicēt ar citu bezvadu sensoru tīka daļu).

Ir veidots bezvadu sensoru tīkls, kurš pamatā sastāv no bezpilota lidaparātiem(7). Šajā gadījumā tiek runāts par *UAV* spietu (no angļu Swarm), kas var tikt realizēts kā mobīlu bezvadu sensoru tīkls (no angļu *Mobile Wireless Sensor Network*). *LUMII* Reālā laika sistēmu laboratorijas gadījumā šis piemērs pagaidām nav interesants, jo tiek izmantoti statiski bezvadu sensoru tīkli.

Ir izstrādāta tehnoloģija datu savākšanai no bezvadu sensoru tīkla ar kvadrotora palīdzību(8). Šis uzdevums ir līdzīgāks tam, ar ko saskaras *LUMII* Reālā laika sistēmu laboratorija. Bezvadu sensoru tīkls var atrasties grūti pieejamā vietā.

Vēl viens interesants bezvadu sensoru tīkla un kvadrotora sadarbības piemērs varētu būt kvadrotora vadības un attēla pārraidīšanas attāluma pagarināšana. Tīkla piemērs redzams 6.



6. att. Kvadrotora vadības attāluma pagarināšana izmantojot bezvadu sensoru tīklu

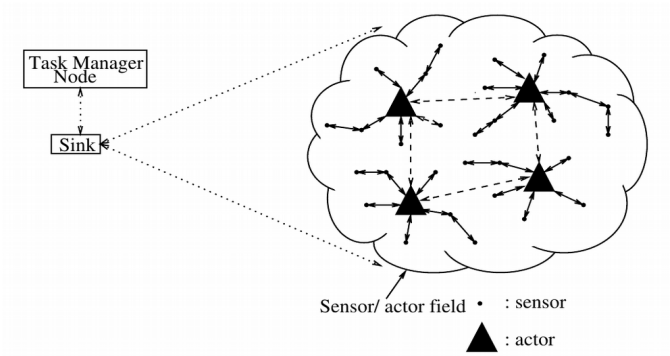
att. Tagadējā Latvijas Republikas likumdošana ierobežo kvadrotoru autonomos lidojumus. Kvadrotoru ir aizliegts izlaist no vizuālās kontroles, un tā pilotam visu laiku ir jābūt gatavam pārņemt vadību(9). Tas ir saprotams no drošības viedokļa, jo krītot kvadrotors var nodarīt bojājumus. Bieži kvadrotors tiek vadīts izmantojot *GCS* (no angļu *Ground Control Station*), kas varētu būt viedtelefons, planšetdators vai dators. Ja tiek saņemts vizuālais signāls uz *GCS,* kvadrotoru, var sūtīt autonomā misijā tālākā maršrutā. Sevišķi lauku mazapdzīvotos, grūti pieejamos apgabalos, kur kvadrotoram krītot radušos postījumu risks ir mazāks. Viens no faktoriem ir arī komunikācijas darbības attālums, viens no tā risinājumiem varētu būt bezvadu sensoru tīkla izmantošana komandas nodošanai kvadrotoram un atbildes saņemšanai no tā.

Reālā laika sistēmu laboratorijas rīcībā ir *AscTec Pelican* kvadrotors, kas ir aprīkots ar *AscTec Mastermind* borta datoru, zemā un augstā līmeņa plidojuma vadības procesoriem, žiroskopu, akselerometru, *GPS,* kamerām un lāzerskeneri, kā arī laboratorijas iepriekš izstrādāta bezvadu sensoru tīkla platforma, kura saziņai izmanto *IQRF* platformu. Risinājuma izstrāde un pārbaude balstās uz *Asctec Pelican* kvadrotoru un *IQRF* platformu, tāpēc tās tiek aprakstītas sīkāk.

2.2. Bezvadu sensoru tīkli

Bezvadu sensoru tīkli ir telpā izvietotu autonomu sensoru kopums, kas ir savienoti ar bezvadu savienojumu, lai veiktu dalītu vides parametru noteikšanu(10). Bezvadu sensoru tīkls sastāv no mezgliem, kā arī parasti tam ir satekne (no angļu *Sink*), kas bieži ir vārteja (no angļu *Gateway*) uz internetu vai mērījumu reģistrētājs (no angļu *Data Logger*). 7. att. ir redzama bezvadu sensoru un izpildītāju tīkla arhitektūra. Runājot par bezvadu sensoru tīkliem, piemin

arī bezvadu sensoru un izpildītāju tīklus, šādi tīkli sastāv no izpildītāju (no angļu *Actor*) mezgliem un sensoru mezgliem. Izpildītāja mezgls veic nostrādi (no angļu *Actuation*).



7. att. Bezvadu sensoru un izpildītāju tīkla fiziskā arhitektūra(7)

Bezvadu sensoru tīklus iedala dažādās topoloģijās, tie var būt vienkārši tīkli ar zvaigznes topoloģiju, vai sarežģītāk realizējami ar režģa topoloģiju. Šajā darbā nozīmīgi ir apskatīt bezvadu sensoru tīklus ar režģa topoloģiju, jo vismaz viens mezgls (tas, kurš ir uzmontēts uz kvadrotora) būs kustīgs, un tas maina savu atrašanās vietu relatīvi pret citiem bezvadu sensoru tīkla mezgliem, līdz ar to, zaudējot savienojumu ar vienu, tam ir jāspēj sazināties caur citu mezglu.

8. att. LUMII Reālā laika sistēmu laboratorijas bezvadu sensoru tīkla mezgls

LUMII reālā laika sistēmu laboratorija ir izstrādāts bezvadu sensoru tīkla risinājums, kas izmanto *MSP430* mikrokontrolierus un *FarmOS* operētājsistēmu. Lai mezgli būtu autonomi,

to barošana ir nodrošināta izmantojot saules paneļus, kuri veic *AAA NiMH* akumulatoru uzlādi. 8. Att. ir redzams *LUMII* Reālā laika sistēmu laboratorijas bezvadu sensoru tīkla mezgls, kas ir aprīkots ar sensoru komutatoru (8. att. apakšā), kas nodrošina vairāku sensoru piesēgumu pie vienas saskarnes un vairākiem temperatūras un mitruma sensoriem. Piedāvātajā risinājumā šie mezgli izmanto *IQRF* bezvadu sensoru tīkla platformu.

2.3. Pieejamā kvadrotora iespējas

Darbā ir izmantots *AscTec Pelican* kvadrotors, kurš ir piemērots dažādu perifērijas iekārtu savietošanai ar to, kā arī dažādiem pielietojumiem, kas saistīti ar datorredzi. Uzdevums ir izpētīt tos elementus, kuri ir nepieciešami testa piemēram. Lai saprastu, kā notiek darbs ar to, ir jāapskata galvenās kvadrotora komplektācijas komponentes - pats kvadrotors, kā arī *AscTec Mastermind* borta dators. 9. att. redzams *AscTec Pelican* kvadrotors.

9. att. AscTec Pelican

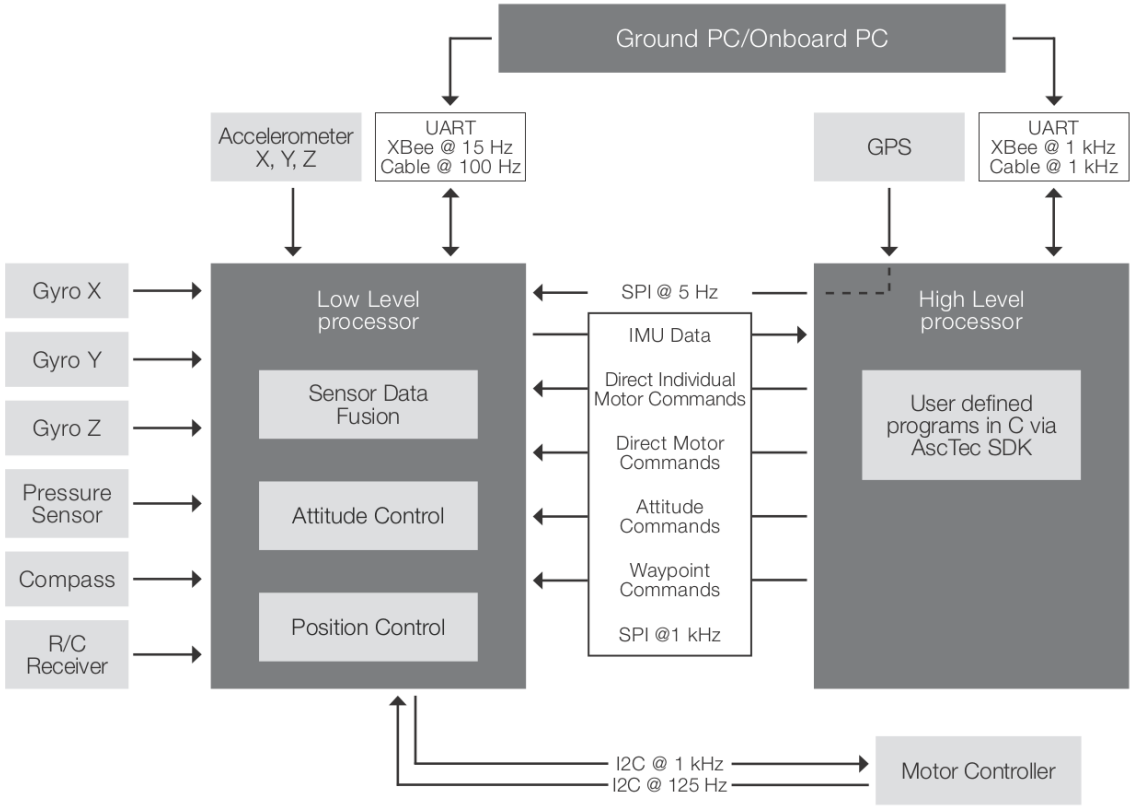
AscTec Pelican tehniskie parametri(11):

- Izmērs: 65,1 * 65,1 * 18,8 cm;
- Propelleru izmērs: 10";

- Dzinēji: 4 160W jaudas;
- Maksimālā normālslodze: 36N;
- Maksimālā derīgās kravas masa: 650g;
- Maksimālā kopējā masa: 1650g;
- Maksimālais lidojuma ātrums: 16m/s;
- Maksimālais lidojuma laiks: 30 minūtes bez derīgās kravas;
- Akumulātors: 6250mAh litija polimēru.

Darbā pieejamajā kvadrotora komplektācijā ietilpst *AscTec Mastermind* borta dators, ko var lietot tāpat kā jebkuru personālo datoru, bet tā forma un savietojamība ar ārējām iekārtām ir pielāgota tā, lai tas atrastos uz kvadrotora. Borta dators sazinās ar kvadrotora zemā līmeņa procesoru, kurš nodrošina kvadrotora pamatfunkciju vadību - lidojuma izlīdzināšanu, motoru kontroli u.c. *AscTec Mastermind* noklusēti ir operētājsistēma, kas bāzēta uz *Ubuntu Linux*, kā arī tam ir uzstādīti draiveri un programmu piemēri dažādu tā iekārtu lietošanai, piemēram, kameru, lāzerskanera, žiroskopa u.c.(12). A*scTec Mastermind* ir pieejamas arī *USB* pieslēgvietas.

AscTec Mastermind plaši izmanto *ROS* dažādu saskarņu realizācijai ar komponentēm, ieskaitot kvadrotora autopilotu. Tajā ir nodrošināti tādi *ROS* risinājumi kā autopilota sensoru, video kameras un lāzerskanera. Saziņas prototipa izstrādē interesē autopilota sensoru daļa, jo no tās var iegūt informāciju par kvadrotora lidojuma stāvokli. *ROS* autopilota sensoru programma iegūst datus no kvadrotora zemā līmeņa datora.



10. att. AscTec Pelican arhitektūra

Kvadrotora vadībai ir iespējams izmantot zemā līmeņa procesoru vai augstā līmeņa procesoru, 10. att. ir redzamas *AscTec Pelican* saskarnes. Lai varētu izmantot augstā līmeņa procesoru, ir nepieciešams izstrādāt lidojuma vadības programmatūru izmantojot *AscTec SDK(13,14)*. Tādēļ darba ietvaros ir vienkāršāk izmantot tikai zemā līmeņa procesoru, kura programmatūru nodrošina ražotājs. Nākotnē ir paredzēta pāreja uz vadību ar augstā līmeņa procesoru.

Bezvadu sensoru tīkla mezgls ir jāizstrādā *ROS*, jo tādā veidā būtu visvienkāršākā saskarne ar citiem moduļiem.

2.4. ROS apraksts

ROS ir elastīgs satvars dažādu robotikas uzdevumu risināšanai. *ROS* ir bibliotēku, rīku un metodikas vienots kopums, kura mērķis ir atvieglot sarežģītas un robustas robotu programmatūras izstrādi. Robotikā cilvēka intelektam vienkāršus uzdevumus mēdz būt ļoti sarežģīti realizēt(15). *ROS* ir paredzēts tikai darbam ar uz *Unix* bāzētām operētājsistēmām. *ROS* arhitektūrā ir trīs līmeņu koncepti(16):

- failu sistēmas līmenis;

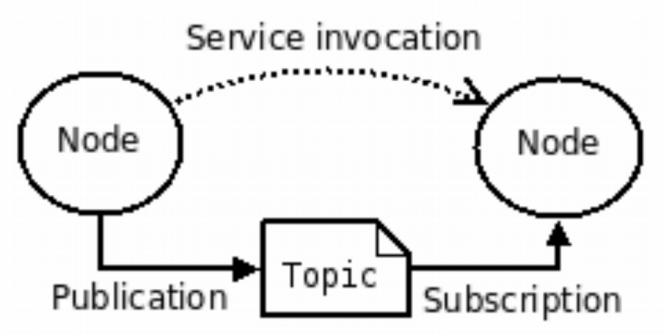
- aprēķinu grafa līmenis;
- komūnas līmenis.

Failu sistēmas līmenis ietver *ROS* resursus, kas atrodas uz diska(16):

- pakotnes - galvenā vienība *ROS* programmatūras organizēšanā, var saturēt izpildlaika procesus(mezglus), no *ROS* atkarīgas bibliotēkas, datu kopas, konfigurācijas failus u.c.;
- metapakotnes - specializētas pakotnes, kas veido vairāku pakotņu grupu;
- pakotņu deklarācijas - nodrošina pakotņu metadatus ieskaitot to nosaukumu, versiju, aprakstu, licenzes informāciju, atkarības no citām pakotnēm u.c.;
- repozitoriji - pakotņu kolekcijas, kurām ir vienota versiju vadības sistēma, kuras tiek izlaistas vienlaicīgi;
- ziņojumu tipi - ziņojumu apraksti, kas definē *ROS* izmantoto ziņojumu datu struktūras;
- servisu tipi - servisu apraksti, kas definē *ROS* servisu pieprasījumu un atbilžu datu struktūras.

Aprēķinu grafs ir vienādranga tīkls (no angļu *peer-to-peer* network), kas sastāv no *ROS* procesiem, kas kopīgi apstrādā datus. Vienkārši aprēķinu grafa koncepti ir:

- mezgli - procesi, kas nodarbojas ar aprēķiniem;
- vedēji(no angļu *master*) - nodrošina vārdu reģistrāciju un meklēšanu aprēķinu grafā, bez tā mezgli nevarētu atrast viens otru, apmainīties ar ziņojumiem vai izsaukt servisus;
- parametru serveris - nodrošina datu glabāšanu pēc atslēgas centrālā vietā, daļa no vedēja;
- ziņojumi - mezgli sazinās viens ar otru izmantojot ziņojumus, datu struktūra, kas sastāv no laukiem, kuriem piemīt tipi;
- tēmas - saziņas modelis, kurā ziņojumi tiek maršrutēti izmantojot transporta sistēmu ar izdevējiem un abonentiem, mezgli sūta ziņojumus izdodot tos noteiktā tēmā, saņem ziņojumus pierakstoties noteiktai tēmai;
- servisi - saziņas modelis, kurā ir pieprasījumi un atbildes, tos definē divas ziņojumu datu struktūras - viena pieprasījumam, viena atbildei;
- maisi - *ROS* ziņojumu saglabāšanas formāts, noderīgs grūti iegūstamu sensoru datu saglabāšanai, lai izstrādātu un testētu algoritmus.



11. att.: ROS saziņa

11. att. redzama saziņa starp diviem mezgliem izmantojot tēmu un servisu.

2.5. Pieejamā bezvadu sensoru tīkla iespējas

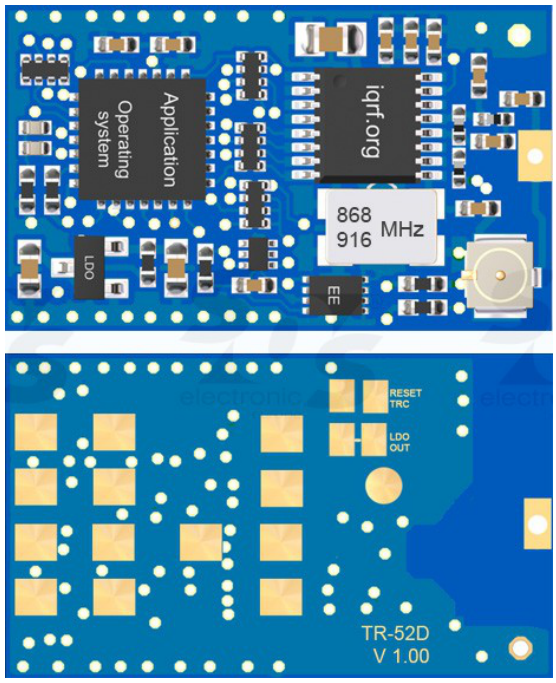
Pieejamais bezvadu sensoru tīkla risinājums izmanto *IQRF* platformu. *IQRF* bezvadu sensoru tīklu platforma ļauj vienkārši izveidot dažādas tīkla konfigurācijas(17).

Šī platforma ar viedo raiduztvērēju palīdzību nodrošina visas tīklam nepieciešamās funkcijas(17), lai veiksmīgi izveidotu tīklu un pārsūtītu datus, ir nepieciešams izpētīt šīs tehnoloģijas specifikāciju. Platforma nodrošina arī režģa (no angļu *mesh*) tīkla funkcionalitāti ar viņu izstrādāto protokolu *IQMESH(18)*, izmantojot to, ir vienkārši veidot dinamiskus bezvadu sensoru tīklus, kur ir iespējams veidot mezglus, kuri drīkst pārvietoties pa tīklu. Izmantojot *IQMESH,* datu paketes tiek piegādātas izmantojot pārpludināšanas (no angļu *Flooding*) mehānismu, kas tās nodot tālāk izmantojot citas bezvadu sensoru tīklā esošās iekārtas(18). Lai izmantotu *IQMESH*, ir nepieciešams no sākuma veikt tīkla mezglu apzināšanu (no angļu *Discovery*), kā arī maršrutētāju uzstādīšanu, bet šajā uzdevumā tas netiks veikts, jo iekštelpu piemērā var darboties ar tīklu, kuram ir zvaigznes topoloģija.

Darba izpildei ir pieejami šādi *IQRF* produkti:

- *DCTR-52* raiduztvērēji;
- *CK-USB-04A* programmatori;
- *DK-EVAL-04A* izmēģinājuma moduļi;
- *IQRF IDE*.

12. att. ir redzams *IQRF DCTR-52* raiduztvērējs. Tas ir fiziski pievienojams iekārtai

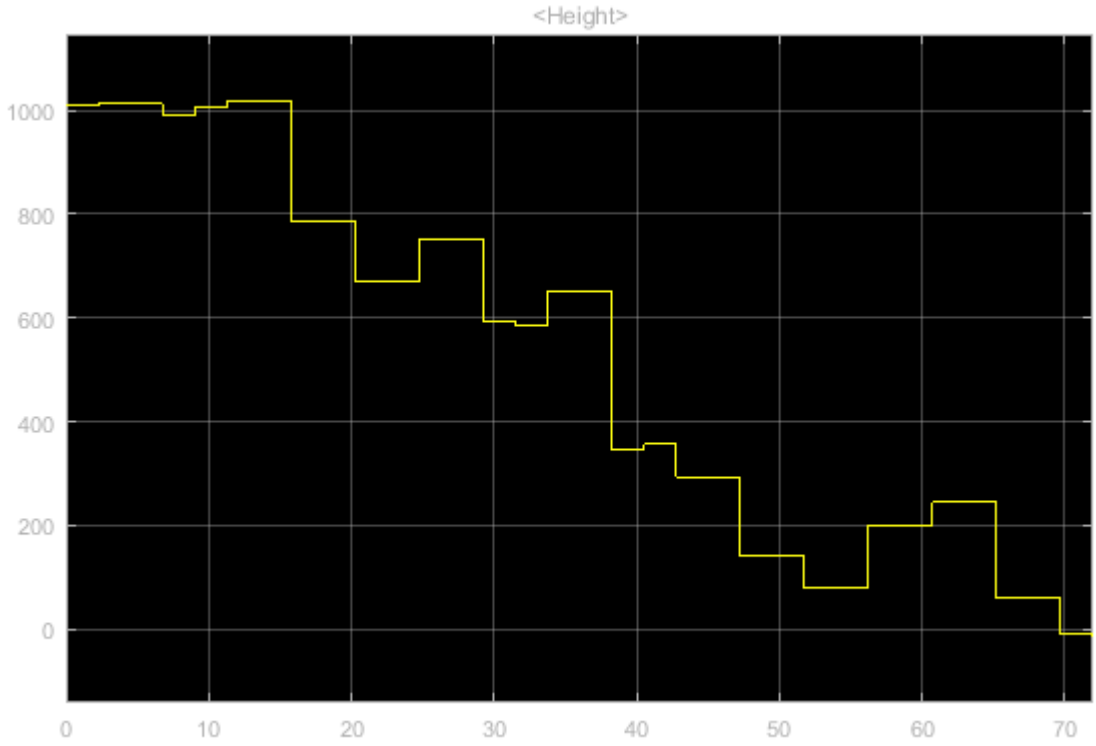


12. att. IQRF DCTR-52D raiduztvērējs

izmantojot *SIM* savienojumu. To ir iespējams kontrolēt izmantojot *UART* vai *SPI* saskarnes, bet prototipa izstrādē nav nepieciešams darboties šādā līmenī - programmatori nodrošina iespēju sazināties ar raiduztvērējiem izmantojot *USB* seriālo iekārtu(19).

2.6. Kvadrotora lokalizācija

Darba uzdevumā bija paredzēts, ka pacelšanās tiks regulēta *IMU* datiem (skat. 2. nodaļas 3. apakšnodaļu). Augstuma noteikšanai var tikt izmantots spiediena sensors, bet ar to nav iespējams noteikt kvadrotora atrašanās vietu telpas *x* un *y* koordinātās. Ārpus telpām tam varētu izmantot *GPS* režīmu, kurš, izmantojot datu sapludināšanu, no *IMU* un *GPS* datiem iegūst precīzu atrašanās vietu.



13. att. Kvadrotora aprēķinātais augstums no IMU_CALCDATA. x - laiks sekundēs; y - augstums milimetros

Darba gaitā ar kvadrotoru atklājās problēmas. Augstuma noteikšanai izmantojot spiediena sensoru, ir redzams, ka iekštelpās tas ir pārāk neprecīzs. Novērojot kvadrotora lidojuma rādījumus, ir redzama liela mērījumu kļūda pat atrodoties uz darba galda (skat. 13. att.).

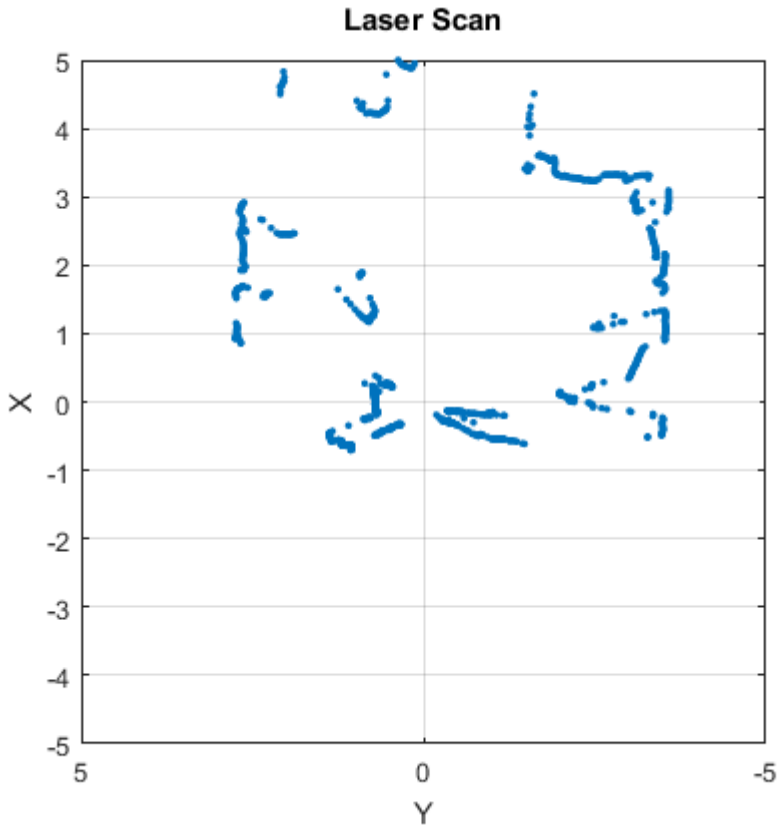
Lidojumos, kas notiek ārpus telpām, ir pieejams *GPS*. *AscTec Pelican* kvadrotoram ir pieejami trīs lidojuma režīmi – *GPS* režīms, augstuma režīms (no angļu *Height Mode*), kā arī rokas vadības (no angļu *Manual Mode*) režīms(13). Izmantojot *GPS* režīmu, ir vienkārši pārvietoties pa telpu, jo ieejas parametri ir telpas koordinātas(20), bet iekštelpās *GPS* nav pieejams, tādēļ iekštelpām atliek augstuma vai rokas vadības režīms. Izmēģinot lidojumu

iekštelpās ar augstuma lidojuma režīmu, ir redzams, ka tas nenodrošina stabilu noturēšanos telpas punktā.

Kvadrotora pozīcija telpā nav stabila, kā arī ir ļoti ierobežota testu izpildes vieta, kura ir aprakstīta 3. nodaļas 3. apakšnodaļā. Līdz ar to, lai veiksmīgi realizētu demonstrācijas testa piemēru, izpētes gaitā radās problēma - ir nepieciešams alternatīvs risinājums kvadrotora lokalizācijai. To varētu realizēt izmantojot:

- lāzerskeneri un/vai stereokameru attālumu noteikšanai līdz telpā esošajam šķērslim (sienām);
- imzantojot telpā izvietotas *UWB* bākas;
- lokalizācija izmantojot bezvadu sensoru tīklu.
- izmantojot videokameru un attēlapstrādi, atpazīt telpā atrodošos objektus un pēc tiem noteikt atrašanās vietu un šķēršļus;

UAV lokalizācijas principi ar lāzerskeneri un stereokameru ir līdzīgi. Tos pētnieki jau ir realizējuši citviet(21). Lāzerskenera gadījumā, daļa no stariem ir novirzīta lejup, lai noteiktu arī *UAV* augstumu telpā, kas ir svarīgi lidojošam objektam(21). Ar lāzerskeneri *MATLAB Simulink* vidē punktu mākoņa dati ir viegli iegūstami, skat. 14. att., no tā var veidot aizņemtības matricu. Punktu mākonis ir nepieciešams, lai izvairītos no objektiem un sienām, bet bez tehniskas pārbūves no šādiem datiem ir sarežģīti noteikt skenera augstumu virs zemes, ko iespējams varētu atrisināt ar augstuma atzīmēm uz sienām. Tā kā definētajā demonstrācijas testa piemērā ir nepieciešams noteikt kvadrotora augstumu, tad šā darba ietvaros tas netiks izmantots, jo būtu jāuzbūvē lāzera atstarošanas virsmas staru pavēršanai lejup vai jāizveido augstuma atzīmes uz sienām, kā arī izmantotie algoritmi ir pietiekami sarēžģīti realizējami.



14. att. lāzerskenera grafiks, x un y koordinātas metros

UWB bāku izvietošana ļautu iegūt kvadrotora pozīciju telpā(22). Lai to realizētu, ir nepieciešams izstrādāt vai iegūt šādas bākas, bet darba izstrādes laikā tādas nebija pieejamas, kā arī šis risinājums ļautu kvadrotoram veikt lokalizāciju tikai tādā telpā, kura ir iepriekš tam sagatavota. Šis risinājums arī liktu iepriekš sagatavot datus par telpu, lai kvadrotoram būtu pieejama informācija par tās parametriem un tajā esošiem šķēršļiem.

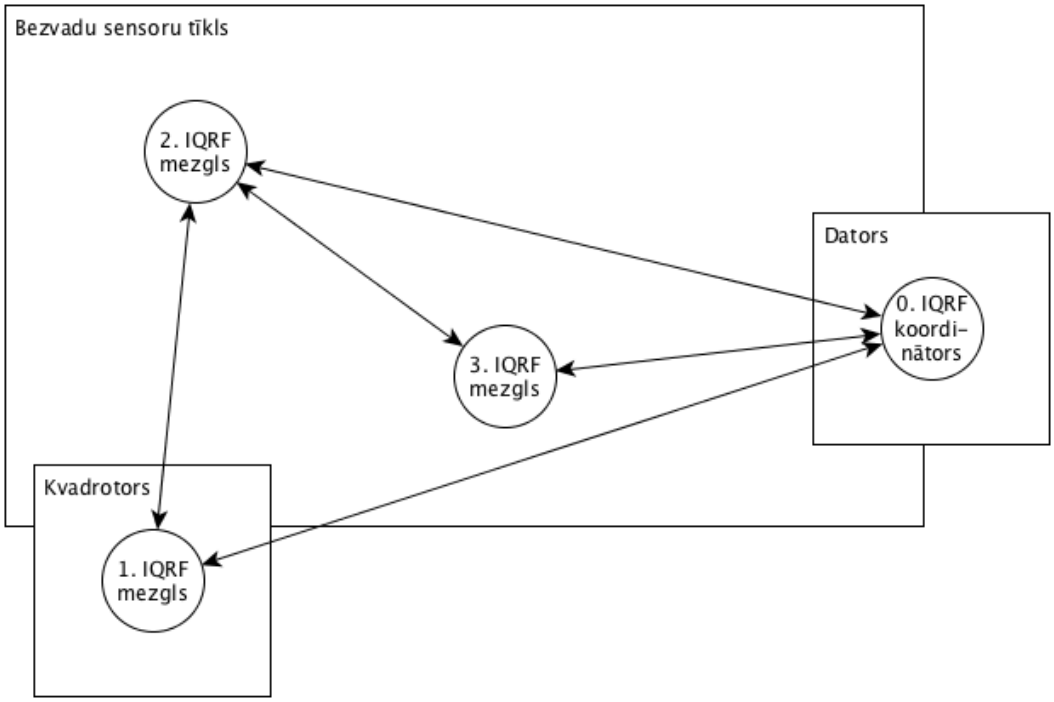
Tā kā darba tēma ir kvadrotora sadarbība ar bezvadu sensoru tīklu, loģiski būtu padomāt par lokalizācijas nodrošināšanu izmantojot to. Viens lokalizācijas variants izmantojot bezvadu sensoru tīklu, ja nav pieejams *GPS*, ir izmantojot *RSSI(23)*. Bet izmantojot šādu risinājumu rastos tāda pati problēma kā ar *UWB* bākām – būtu iepriekš jāsagatavo dati ar šķēršļu atrašanās vietām.

Ir iespējams realizēt kvadrotora lokalizāciju izmantojot kameru un attēlapstrādi. Lai to izdarītu, ir nepieciešama telpas iepriekšēja sagatavošana, bet vienkāršāka kā ar *UWB* bākām. Var, piemēram, uzzīmēt uz zemes apļus, noteikt to rādiusus un, atkarībā no tiem, noteikt kvadrotora pozīciju telpā. Šai metodei galvenais trūkums ir tas, ka lokalizēt kvadrotoru nav iespējams, ja atpazīstamie objekti ir ārpus kameras redzamības.

3. UZDEVUMA REALIZĀCIJA

3.1. Piedāvātā sistēmas arhitektūra

Tiek piedāvāts izveidot uz *IQRF* platformas bāzētu bezvadu sensoru tīklu ar vienu koordinātoru un trīs mezgliem (skat. 15. att.). Tā kā *IQRF* platforma atbalsta režģa tīklu veidošanu, mezglus var izvietot patvaļīgi ar nosacījumu, ka katrs mezgls sasniedz vismaz vienu citu mezglu vai koordinātoru. Viens no mezgliem atrodas uz kvadrotora, bet divi uz zemes. Uz kvadrotora mezgla vietā varētu atrasties koordinātors, bet tas nozīmētu, ka, ja kvadrotors izlidotu ārpus bezvadu sensoru tīkla darbības zonas, tīkls pārstātu funkcionēt. Kvadrotora vadības programma ir izstrādāta *MATLAB Simulink* vidē.

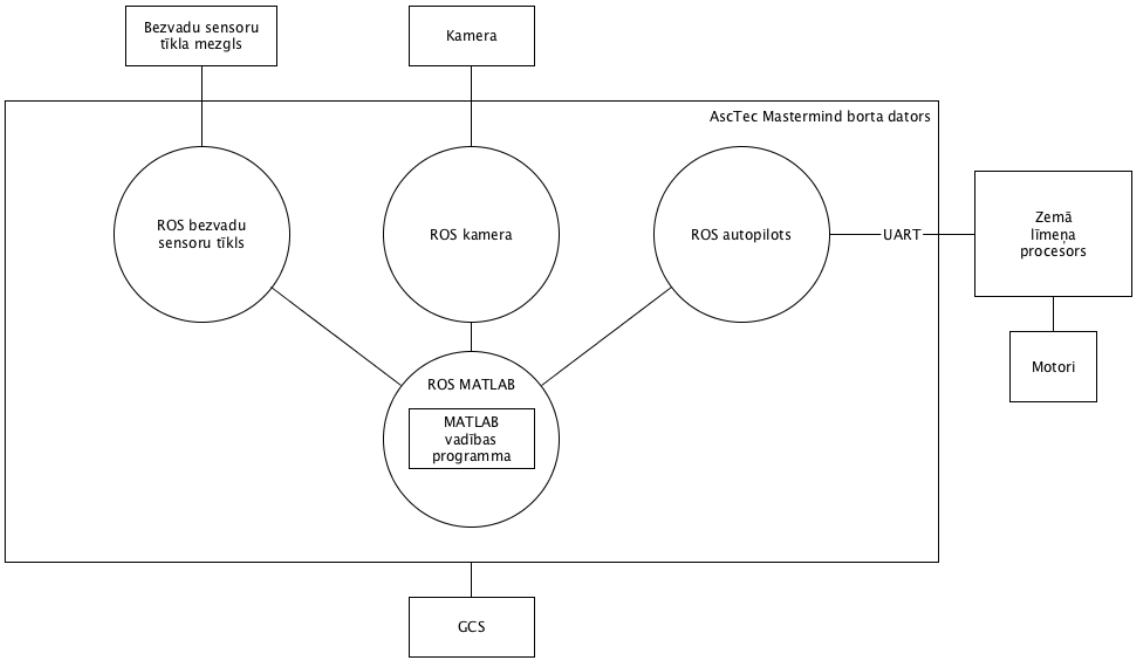


15. att. Demonstrācijas testa piemēra bezvadu sensoru tīkla shēma

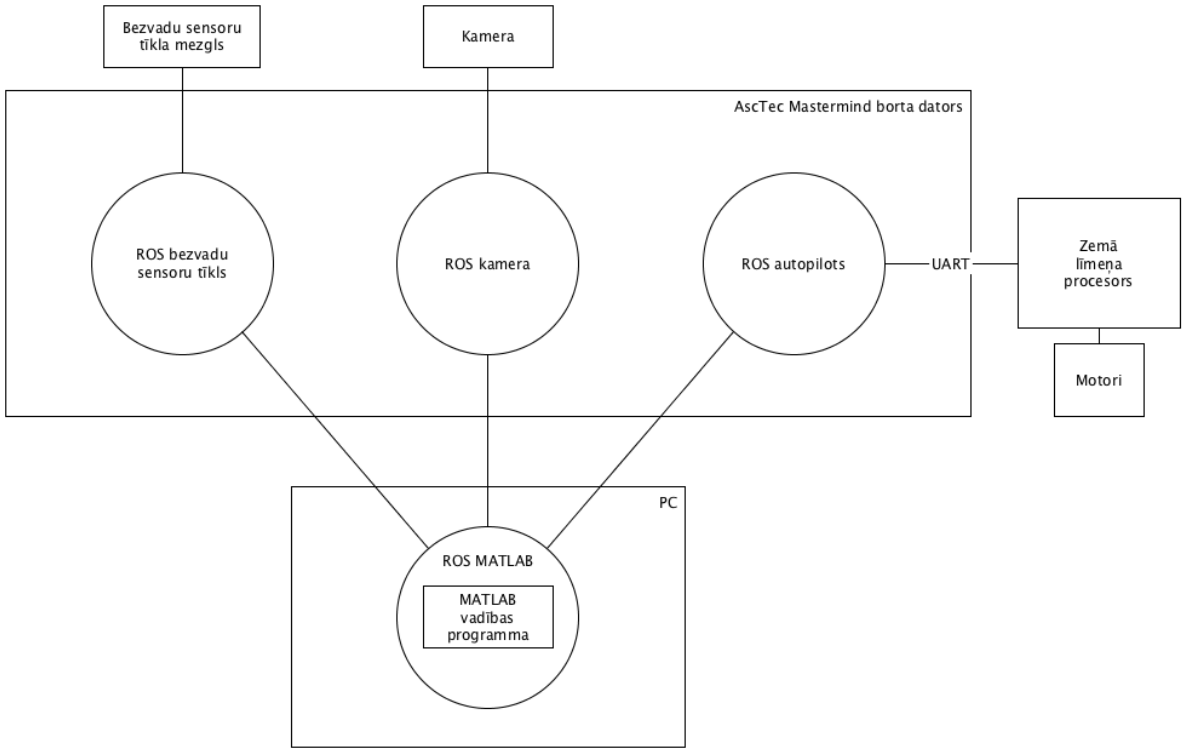
Tā kā sākotnējā sadarbības izstrāde starp bezvadu sensoru tīklu un kvadrotoru notiks iekštelpās, lokalizācijas uzdevums nav triviāls – nav pieejams *GPS*, kā arī augstuma noteikšana ar spiediena sensoru nav pietiekami precīza. Lokalizācijas nodrošināšanai tiks izmantota vizuāla objekta – uzzīmēta apļa atpazīšana un tā rādiusa noteikšana.

Izstrādes piemēra arhitektūra (skat. 16. att.) ir izstrādāta balstoties uz plānoto *R5-COP* arhitektūru (skat. 2. att.). Tā kā uzdevums ir paredzēts testēšanai iekštelpās, ir ieviesti vairāki vienkāršojumi, kuru rezultātā iegūta shēma, kas attēlota 17. att.:

1. no *MATLAB Simulink ROS* mezgla netiek ģenerēts *C* kods (tiek darbināts kā redzams 17. att., bet vēlāk ir paredzēts 16. att. redzamais variants), bet izmantojot *ROS* iespējas, tiek darbināts uz atsevišķa datora no *MATLAB Simulink* vides. Tas arī tiek darīts tādēļ, ka *R5-COP* projekta ietvaros šobrīd līdz galam nav izstrādāts reālā laika kods no *Stateflow* diagrammas;
2. nav vajadzības pēc *GCS* moduļa, jo kvadrotora vadības mezgls nodrošina pietiekamas testēšanas iespējas *MATLAB Simulink* vidē;
3. kvadrotora vadības piemērs *Stateflow* tiek speciāli izstrādāts šīm testam, lai saglabātu pārbaudes vienkāršību. Tālākā *R5-COP* attīstības posmā tā vietā tiks lietots pilnais vadības bloks, kurš pagaidām tiek izstrādāts izmantojot *SIL* modeli.



16. att. Kvadrotora perifērija un ROS mezgli ar vadības programmu uz borta datora



17. att. Kvadrotora perifērija un ROS mezgli ar vadības programmu MATLAB Simulink vidē

Ir izstrādāti trīs *ROS* mezgli:

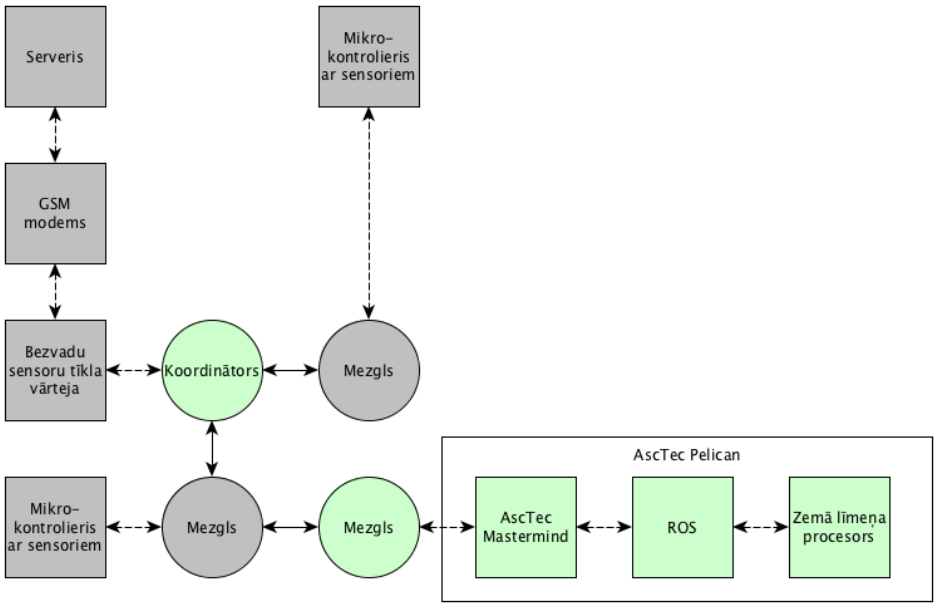
- *ROS* bezvadu sensoru tīkla mezgls, kas redzams 6. pielikumā. Tas gaida ziņu no koordinātora vienā mezgla, uz kuru tas atbild ar iepriekš pa *ROS* saņemtu ziņojumu, kurš satur *DPA* komandu;
- kvadrotora vadības mezgla piemērs tikai šim uzdevumam, kurā ietverta arī lokalizācija.

3.2. Piedāvātais saskarņu lietojums

Izmantotie *IQRF DCTR-52D* raiduztvērēji ir paredzēti darboties izmantojot *UART* vai *SPI*, bet, izmantojot *IQRF* programmātorus, ir iespējams tos kontrolēt izmantojot *USB* seriālo iekārtu(19). Šis risinājums ir derīgs prototipa izstrādei.

Kvadrotora vadībai tiek izmantots *ROS* mezgls *asctec_driver*, kas jau ir nodrošināts un tas izmanto saskarni starp *AscTec Mastermind* borta datoru un zemā līmeņa procesoru. Ir iespējams arī izmantot saskarni starp *AscTec Mastermind* borta datoru un augstā līmeņa procesoru, bet tad ir nepieciešams izstrādāt papildus programmatūru kvadrotora vadībai caur augstā līmeņa procesoru.

Savukārt, kvadrotora saskarnei ar bezvadu sensoru tīklu tiek izmantots *IQRF* programmātors, kas pieslēgts kā *USB* seriālā iekārta. Tas tiek vadīts izmantojot izstrādātu *ROS* mezglu *Python* programmēšanas valodā, kas redzams 6. pielikumā.



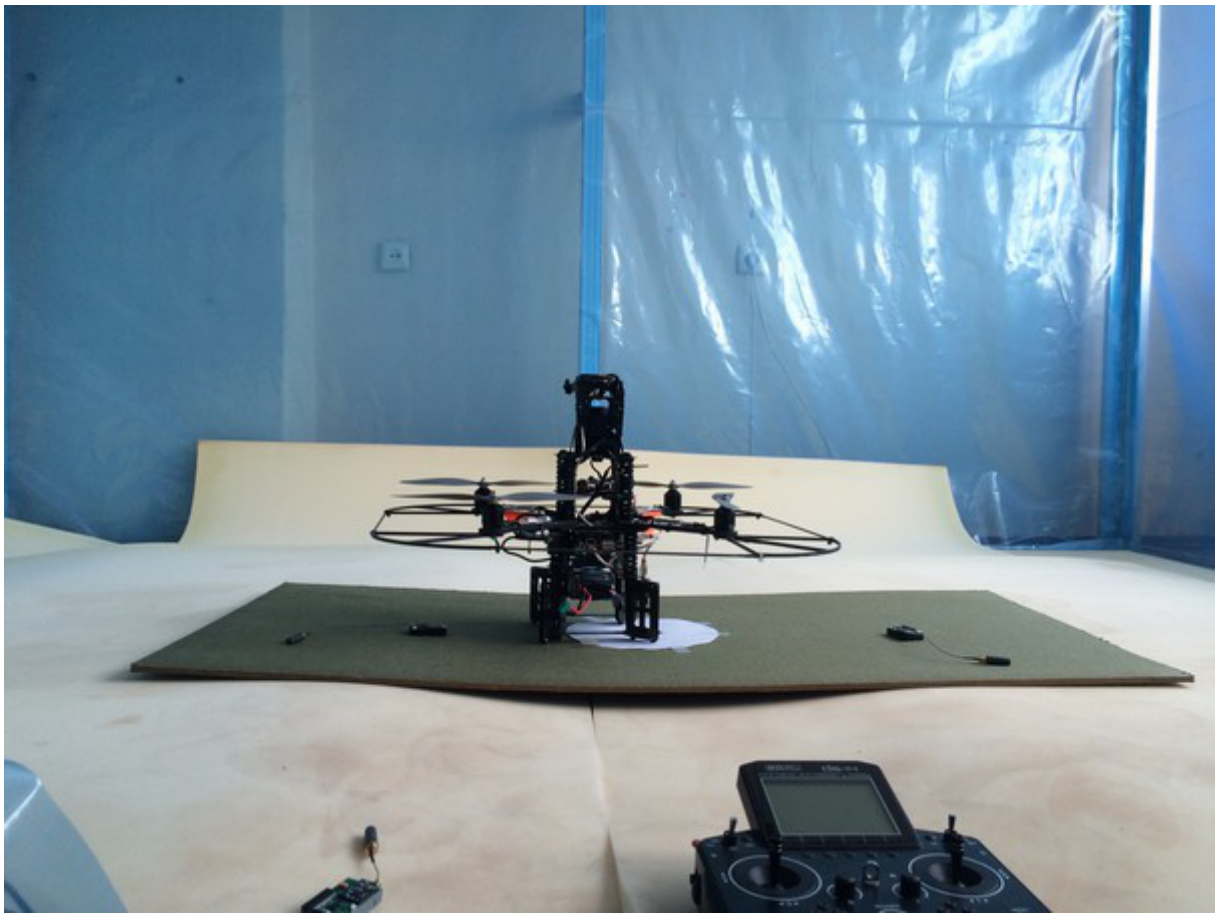
18. att. bezvadu sensoru tīkla modelis ar kvadrotoru

18. att. ir redzams bezvadu sensoru tīkla modelis, kurā ietilpst *AscTec Pelican* kvadrotors, tajā ir attēloti arī citi bezvadu sensoru tīkla mezgli un to mikrokontrolieri ar sensoriem, bet tos darba izpildei nav nepieciešams apskatīt, jo tīkla funkcionalitāti nodrošina *IQRF* platforma (pārējiem mikrokontrolieriem netiek paziņots par ziņojumiem, kuri nav adresēti tiem).

Kvadrotora vadībai tiek izmantots *MATLAB Simulink* un *Stateflow* modelis, kas veido bezvadu sensoru tīkla *ROS* mezglu, kā arī vēl vienu *ROS* mezglu, kas ar kvadrotora motoru vadību sazināsies izmantojot *ROS asctec_driver* mezglu.

3.3. Testa vietas iekārtošana

Kiberfizikālo sistēmu izstrādei un testēšanai ir nepieciešama tam piemērotas vides iekārtošanu, ne vienmēr pietiek tikai ar galda datoru. Tā kā tika nolemts, ka sākotnēji darbs tiks veikts iekštelpās, bija nepieciešams iekārtot testa vietu. Testa telpas galvenās prasības ir kvadrotora drošības nodrošināšana gadījumā, ja tiek zaudēta kontrole pār to. Telpa tika norobežota telpa ar profiliem, starp kuriem tika novilkta plēve. Uz grīdas tika izklāts porlons triecienu mīkstināšanai. Tādā veidā tika panākts, ka kontroles zaudēšana pār kvadrotoru un avārijas nerada tehnikas bojājumus. 19. Att. ir redzams kvadrotors un bezvadu sensoru tīkls iekārtotajā testa telpā.



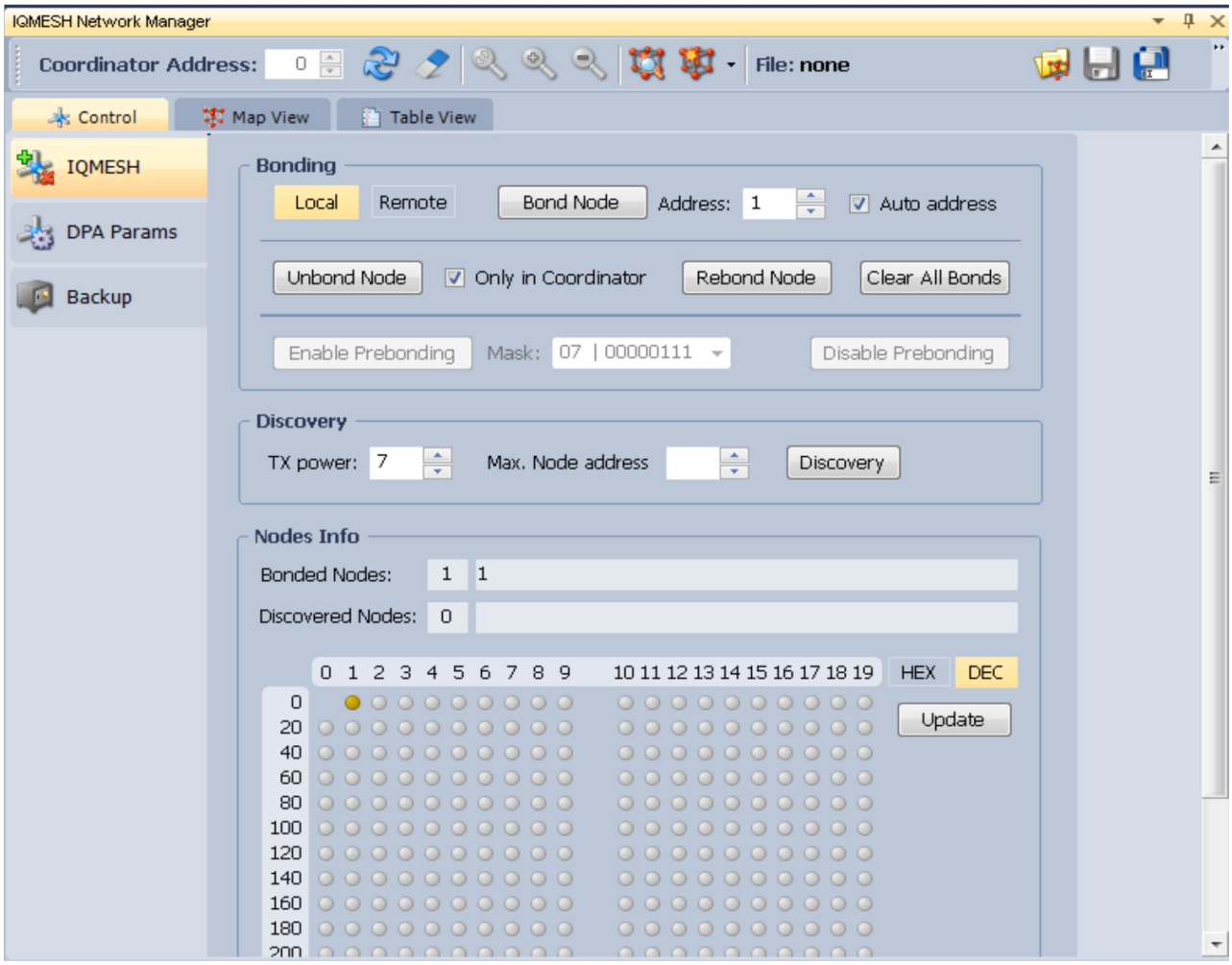
19. att.: AscTec Pelican kvadrotors un IQRF bezvadu sensoru tīkls iekārtotajā testa telpā

3.4. Bezvadu sensoru tīkla izveide

Lai izveidotu *IQRF* tīklu, ir nepieciešams veikt šādas darbības(24):

1. *IQRF IDE* instalācija;
2. koordinātora izveide;
3. mezgla izveide;
4. mezgla pievienošana konkrētā koordinātora tīklam.

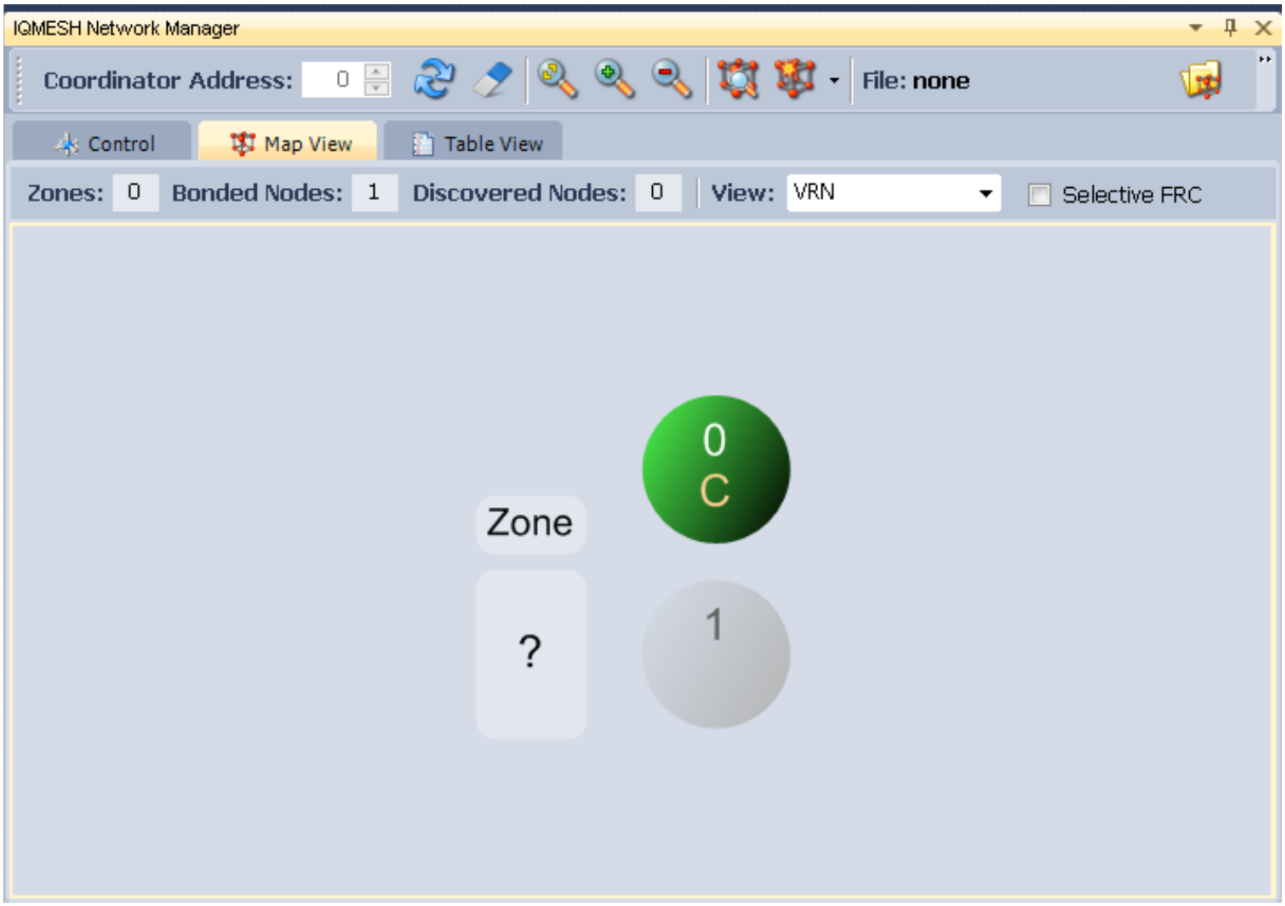
IQRF IDE instalācijas failu var iegūt no *IQRF* tīmekļa vietnes, instalācijas procesā tiek



20. att. IQRF tīkla konfigurācijas saskarne

uzinstalēti arī nepieciešamie draiveri. Pēc tam ir jāizveido projekts - var izmantot noklusēto konfigurāciju. Pēc tam projektam ir jāpievieno general *HWP*. *General HWP* satur programmatūru un konfigurācijas priekš *IQRF* koordinātora un mezgliem. Pēc tam ir nepieciešams tos augšupielādēt raiduztvērējos. Kad tas ir izdarīts, var pievienot mezgla raiduztvērēju koordinātoram. To ir iespējams izdarīt izmantojot 20. att. redzamo saskarni, no sākuma uzspiežot *"Clear All Bonds"*, lai pārliecinātos, ka koordinātoram nav pievienotu

mezglu. Pēc tam izmantojot *"Bond"* pogu var pievienot jaunu mezglu. 21. att. ir redzams izveidotais tīkls ar vienu koordinātoru un vienu mezglu.



21. att. IQRF tīkla statusa saskarne

3.5. Bezvadu sensoru tīkla vadība

Raiduztvērējus ir paredzēts vadīt izmantojot *UART* vai *SPI*, bet izmantojot *IQRF* programmātorus ir iespējams tos kontrolēt izmantojot *USB* seriālo iekārtu(19). Šis risinājums ir derīgs prototipa izstrādei. Lai pārslēgtu *IQRF* programmātoru uz *CDC* režīmu, ar ko iespējams to izmantot kā seriālo iekārtu saziņai ar raiduztvērēju, ir nepieciešams to uzstādīt izmantojot *IQRF IDE*, to var izdarīt izvēloties rīkjoslā *IQRF CDC* režīmu.

Pēc tam pievienojot programmātoru *Linux* datoram, tiek izveidota jauna seriālā iekārta, piemēram, *"/dev/ttyACM0"*. To arī ir iespējams pēc iekārtas un ražotāja identifikatora, šajā gadījumā programmātors ir atrodams *"/dev/serial/by-id/usb-MICRORISC_s.r.o._0002-if00"*.

CDC ziņojums izskatās šādi - ">[ķermenis][rakstatgrieze]"(19). Uzdevuma izpildīšanai ir nepieciešams izmantot:

- datu sūtīšanu;

- datu lasīšanu.

Datu sūtīšanai ir jāizmanto šāds ziņojums - *">DS[datu garums]:[dati] [rakstatgrieze]"*(19). Piemēram, *">DS[0x05]:Hello[rakstatgrieze]"*. Ja datu sūtīšana ir bijusi veiksmīga, tiek atgriezts "<DS:OK[rakstatgrieze]", ja sūtījums ir bijis nepareizs, iekārta atgriež ziņojumu *"<DS:ERR[rakstatgrieze]"*, bet, ja iekārta ir bijusi aizņemta, tiks atgriezts ziņoums *"<DS:BUSY[rakstagrieze]"*(19).

Datu lasīšanas ziņojums izskatās šādi - *"<DR[datu garums]:[dati][rakstatgrieze]"*(19). Veiksmīga *DPA* ziņojuma gadījumā no koordinātora uz mezglu, mezgls saņem šādu ziņojumu, uz kuru ir nepieciešams atbildēt pirms pagājis taimautā norādītais laiks.

Jāapskata 1. tabulā minētais *DPA* komunikācijas piemērs izmantojot *CDC*. Lai tādu *DPA* ziņojumu izsūtītu, *CDC* iekārtai ir jānosūta *">DS[0x08]:[0x01, 0x00, 0x08, 0x00, 0x00, 0x00, 0x01, 0x01][rakstatgrieze]"*. Ja komunikācija ir notikusi veiksmīgi, *CDC* iekārta, kurai ir pievienots mezgls, saņems *"<DR[0x01]:[0x01][rakstatgrieze]"*. Pēc tam no mezgla *CDC* iekārtas var nosūtīt ziņojumu *">DS[0x01]:[0x01][rakstagrieze]"*. Koordinātora *CDC* iekārtai vajadzētu saņemt ziņojumu *"<DR[0x08]:[0x01, 0x00, 0x08, 0x80, 0x00, 0x00, 0x00, 0x5B, 0x01][rakstatgrieze]"*, kura pēdējais baits ir no mezgla izsūtītais "*0x01"*. Ja šāds ziņojums ir saņemts, komunikācija starp koordinātoru un mezglu ir notikusi veiksmīgi.

Izmantojot *IQRF* raiduztvērējus, visvienkāršāk ir izmantot ražotāja piedāvāto *DPA* saziņas protokolu. *IQRF* platformā visas komunikācijas iniciē koordinātors - tādā veidā ir atrisināta problēma, ka varētu rasties datu pārraides sadursmes. 1. tabulā ir redzams *DPA* ziņojuma piemērs(25).

NADR[2B]	**PNUM[1B]**	**PCMD[1B]**	**HWPID[2B]**	**PDATA[0-56B]**
[0x01, 0x00]	[0x08]	[0x00]	[0x00, 0x00]	[0x01, 0x01]

1. tabula: DPA ziņojuma piemērs

NADR ir adresāta adrese, "*[0x00, 0x00]"* vienmēr ir koordinātors, pārējās vērtības ir mezgli(ja ar tādu *NADR* ir pievienoti koordinātoram). *PNUM* ir perifērijas numurs, piemērā redzamais ir *SPI* - tas nozīmē, ka saņēmējs ziņojumu sākot no *PDATA* 2. baita nosūtīs pa *SPI* iekārtai, kurai tas ir pievienots. *PCMD* ir perifērijas komanda, tās nozīme ir atkarīga no *PNUM*. Ja *PNUM* atbilst *SPI* un *PCMD* ir "0x00", tiks veikta datu pa *SPI* lasīšana un rakstīšana. *SPI* Lasīšanas un rakstīšanas gadījumā *PDATA* pirmais baits apzīmē taimautu - cik ilgi koordinātors gaidīs atbildi no mezgla. Taimautu apzīmē desmitos milisekunžu(25), piemērā koordinātors atbildi no mezgla gaidīs 10 milisekundes.

3.6. Kvadrotora datu nolasīšana *ROS*

Apskatot aprakstītos *ROS* konceptus un informāciju par kvadrotora autopilota pakotni, var secināt, ka uzdevuma izpildei ir nepieciešams abonēt tēmu ar nosaukumu *IMU_CALCDATA(26)*. Lai to izdarītu, ir nepieciešams:

1. izveidot *ROS* pakotni;
2. izveidot *ROS* mezglu;
3. izveidotajā *ROS* mezglā veikt tēmas ar nosaukumu *IMU_CALCDATA* abonēšanu.

Lai izveidotu *ROS* pakotni, ir jārīkojas šādi(27):

1. Linux terminālī jāpārvietojas uz *"~/catkin_ws/src"* direktoriju;
2. jāizsauc komanda *"catkin_create_pkg <pakotnes_nosaukums> [pakotnes_atkarības]"*, kvadrotora autopilota gadījumā, lai izveidotu pakotni ar nosaukumu wsn, ir jāizsauc komanda *"catkin_create_pkg wsn std_msgs rospy asctec_autopilot"*.

Nākamais solis ir izveidot *ROS* mezglu, kas abonē tēmu *IMU_CALCDATA*:

1. Linux terminālī ar *roscd* komandu jāpārvietojas uz pakotnes direktoriju - *"roscd wsn"*;
2. jāizveido skripts direktorijā, kurā tiks glabāti nepieciešamie *Python* skripti un jāpārvietojas uz to;
3. jāizveido skripts ar nosaukumu *"wsn_imu_calcdata_listener.py"*;
4. mezglu var izveidot ar šādu *Python* izsaukumu - *"rospy.init_node('wsn_imu_calcdata_listener', anonymous=True)"*, kur pirmais arguments ir jaunizveidotā mezgla nosaukums, bet otrais arguments nodrošina to, ka, ja tīklā parādīsies vēlviens mezgls ar tādu pašu nosaukumu, tas turpinās darboties;
5. pierakstīties tēmai var ar šādu *Python* izsaukumu - *"rospy.Subscriber('IMU_CALCDATA', IMUCalcData, callback)"*, kur pirmais parametrs ir tēmas nosaukums, otrais parametrs ir tēmas datu tips, bet trešais parametrs ir funkcija, kura tiks izsaukta, kad tiks saņemts ziņojums, kas pieder tēmai;
6. pēc tam pakotnes var nokompilēt izmantojot komandu *"catkin_make"*.

Jaunizveidotā *ROS* mezgla palaišanai nepieciešamās darbības ir:

1. vienā *Linux* terminālī jāizpilda komanda *"roscore"*, kas nodrošina *ROS* kodola darbību;
2. citā *Linux* terminālī ir jāizpilda komanda *"roslaunch asctec_autopilot autopilot.launch"*;
3. jāizpilda komanda *"rosrun wsn wsn_imu_calcdata_listener"*.

3.7. IQRF bezvadu sensoru tīkla vadīšana izmantojot *ROS* mezglu

Tika izstrādāts *ROS* mezgls, kas lasa no *IQRF* tīklu saņemtos datus un ir pierakstījies *"iqrf/CMD"* tēmai, kā arī publicē *"iqrf/STATUS"* tēmu (skat. 6. pielikumu). Saņemot datus no *"iqrf/CMD"* tēmas, mezgls pēc koordinātora pieprasījuma nodod komandu tam.

3.8. Kvadrotora vadības *ROS* mezgls

Centrālais *ROS* vadības mezgls ir realizēts (skat. 9. un 10. pielikumu) *MATLAB Simulink*, saskaņojoties ar *R5-COP* koncepciju kā *ROS* mezgls, lai sgalabātu vienotu arhitektūru ar *R5-COP* projektu. Kā tika minēts 1. nodaļas 2. apakšnodaļā, *R5-COP* ir balstīts uz *Simulink*, viens no tā lietošanas pamatojumiem ir uz *MBD* (no angļu *Model Based Design*) paradigmām balstītas valodas *Stateflow* pieejamība. Šāda tipa valodas atvieglo reālā laika sistēmu algoritmu pierakstu. Izstrādātā demonstrācijas testa piemēra centrālais vadības mezgls varētu būt arī izvietots *ROS* bezvadu sensoru tīkla mezglā vai citur, jo tas apzināti ir veidots pēc iespējas vienkāršāks. Tomēr, saglabājot pēctecību ar *R5-COP*, tas ir izstrādāts *MATLAB Simulink*.

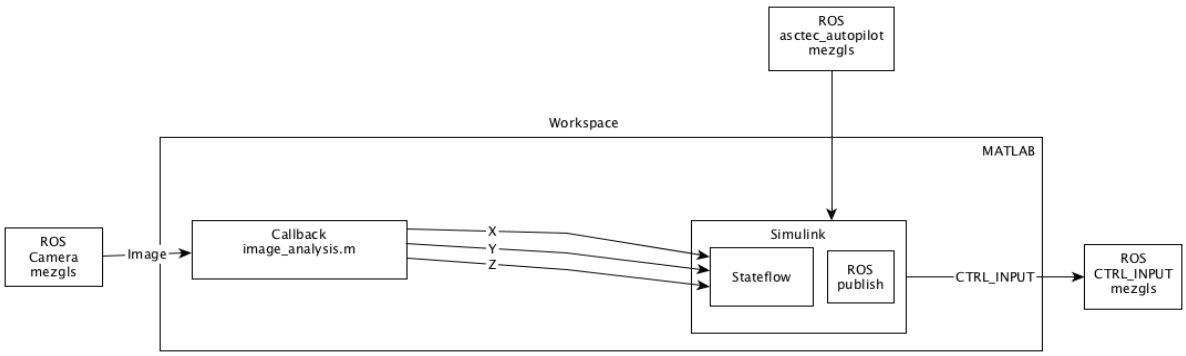
Lai atvieglotu testēšanu, tiek mēģināts *ROS* mezglu darbināt uz atsevišķa datora sazinoties izmantojot WiFi *MATLAB* vidē. Nākotnes risinājumā tiks uzģenerēts *C* kods no *MATLAB Simulink* modeļa, izmantojot *Simulink Real Time Toolbox*. Pēc tam tas tiks nokompilēts priekš *AscTec Mastermind* un darbosies lokāli uz kvadrotora borta datora.

Centrā ir *MATLAB Simulink* modelis (skat. 10. pielikumu). Tā uzdevums ir saņemt ārējos signālus no aparatūras un vides (pārsvarā tie ir organizēti *ROS*). Kvadrotora signālu struktūras ir parādītas 7. pielikumā. *Robotics Toolbox Simulink* modelim izstrādāta vienkārša *ROS* ziņojumu izsūtīšana un saņemšana (skat. 9. pielikumu). Lai kontrolētu motorus, svarīgākās kvadrotorā nodrošinātās *ROS* tēmas (no angļu *Topic*), ir *IMU_CALCDATA*, *LL_Status*, kuras ir redzamas 7. pielikumā. Savukārt, motoru vadībai tiek veidots *ROS* ziņojums *CTRL_INPUT* tēmai.

Simulink modelis tiek darbināts simulācijas režīmā, tas tiek darīts, lai atvieglotu šī darba izstrādi. Vēlāk tas tiks eksportēts uz *Asctec Mastermind* borta datoru kā realā laika programma. Tāpēc tiek ieviesta vienkāršota apmaiņas shēma - kad kvadrotora mezgls saņem *CTRL_OUTPUT ROS* ziņojumu, tas izsauc apstrādei vadības algoritmu, kas ir izstrādāts kā *Stateflow* diagramma (skat. 11. pielikumu), kas veido motora vadības signālu *CTRL_INPUT* un bezvadu sensoru tīkla vadības signālu *"iqrf/CMD"*.

Stateflow vadības piemērs apzināti ir veidots vienkāršs (skat. 11. pielikumu), bet lai varētu saskatīt demonstrācijas piemēra soļus.

Šajā demonstrācijas piemērā nebija paredzēts kameras attēla signāla apstrāde, bet rodoties problēmai ar lokalizāciju šaurajā testa telpā, tika ieviesta vienkārša tā apstrāde izmantojot atzvanīšanas funkciju (skat. 22. att).



22. att. MATLAB Simulink attēlapstrāde

3.9. MATLAB Simulink vides sagatavošana

Lai *MATLAB Simulink ROS* mezgls varētu komunicēt ar *asctec_driver ROS* mezglu, bija nepieciešams pievienot *asctec_msgs ROS* ziņojumu tipu grupu(26). Lai to izdarītu *MATLAB* vidē ir nepieciešams(28):

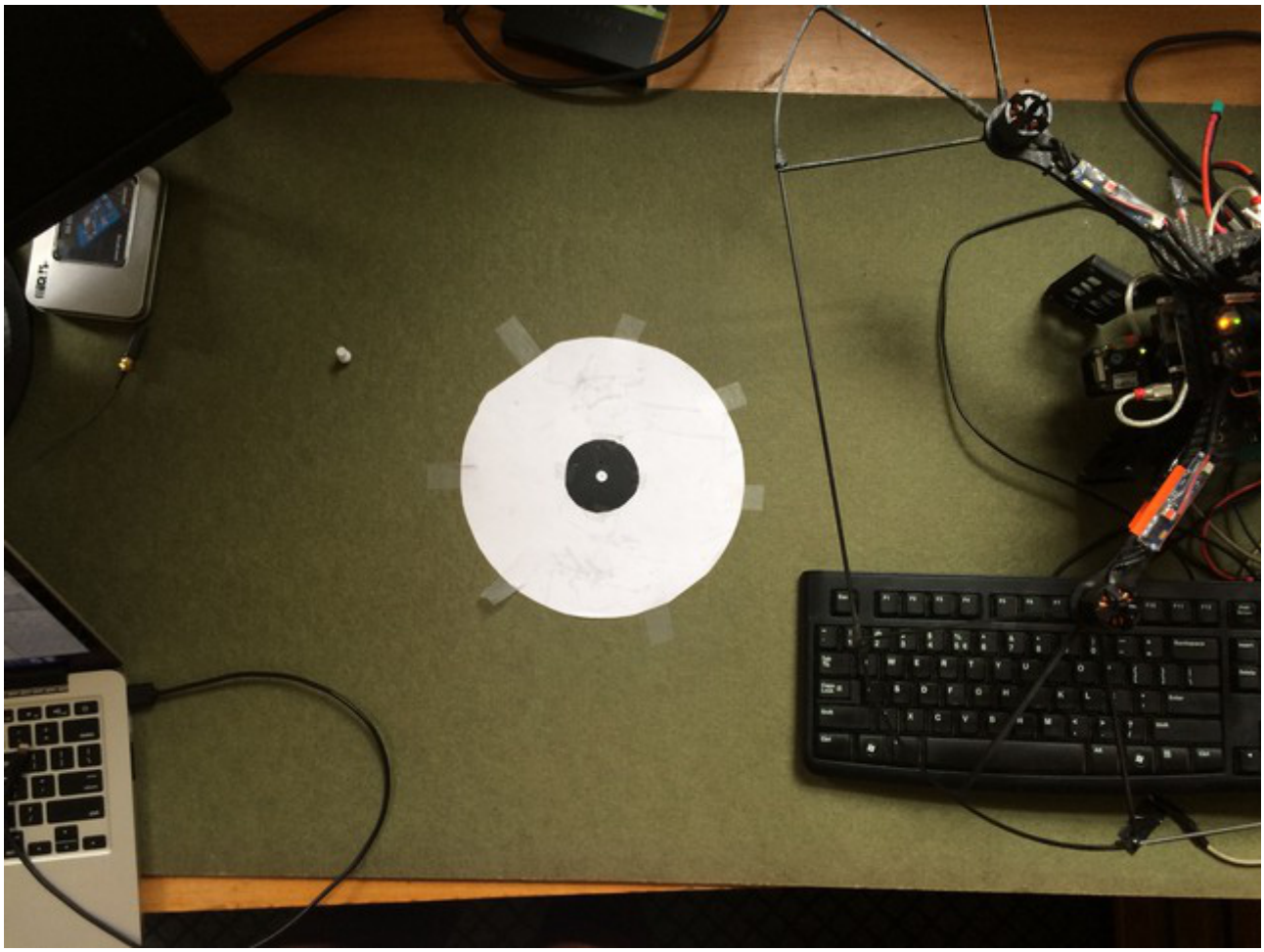
1. uzinstalēt *Robotics System Toolbox Add-ons(29)* ar komandu *"roboticsAddons"*;
2. pārkopēt *ROS* pakotni, kura satur pievienojamo ziņojumu uz noteiktu direktoriju, piemēram, *"c:\MATLAB\custom_msg" asctec_msgs* un izsaukt *rosgenmsg* funkciju ar šo direktoriju kā parametru;
3. pievienot *javaclasspath.txt* uzģenerēto *JAR* failu, kas atrodas iepriekš minētajā direktorijā;
4. pievienot *MATLAB* ceļam(no angļu *Path*) iepriekš minēto direktoriju ar *"addpath"* funkciju un saglabāt to ar *"savepath"* funkciju;
5. restartēt *MATLAB*;
6. pārliecināties, vai ir veiksmīgi pievienots jaunais ziņojums, var izmantojot *"rosmsg list"* funkcju.

3.10. Kvadrotora lokalizācija

Izpētes rezultātā (skat 2. nodaļas 6. apakšnodaļu) tika secināts, ka, lai nodrošinātu automātisku kustības kontroli mūsu testa telpā, ir nepieciešams veikt kvadrotora lokalizāciju izmantojot papildus līdzekļus.

Kvadrotora lokalizāciju iekštelpās tika nolemts risināt izmantojot parasto kameru un attēlapstrādi, jo tehniski šis risinājums tika atzīts par visvienkāršāko un tam bija nepieciešama vismazākā iepriekšējā sagatavošanās.

Parasti, lai veiktu lokalizāciju, tiek sapludināti dati (no angļu *Data Fusion*) no dažādiem



23. att. Kvadrotora lokalizācijas mērķis

sensoriem, piemēram, gaisa spiediena, *GPS* un *IMU*, lai iegūtu precīzāku rezultātu. Šī darba uzdevums nebija precīza lokalizācija, bet spēt lidot un neieskriet sienās vai stipri nemainīt augstumu, tāpēc tas netika izmantots, bet nākotnes attīstībā tas noteikti būs nepieciešams. Ir nolemts, ka vienkāršākais ir mērķa formas, zinot tā izmērus, ir vienkārši izrēķināt relatīvo augstumu pret tiem. Tas varēs noderēt, kad tiks izstrādātas misijas ar automātisko nosēšanos, izveidotais mērķis ir redzams 23. att. 22. Att. ir redzams vienkāršs attēlapstrādes modelis, tajā

ir attēlota vienkārša atzvanīšanas funkcija, kura izsauc *MATLAB* skriptu (skat. 8. pielikumu) tad, kad pienāk *ROS* ziņojums no kameras ar attēlu. Šāda no pārējo signālu(piemēram, *IMU_CALCDATA, LL_Status*) apstrādes atšķirīga apstrāde ir ieviesta, jo tā ir salīdzinoši laikietilpīga un, lai attēla apstrādes process notiktu paralēli vadības signāla apstrādei. Šajā skriptā tiek veikta attēla apstrāde un tiek atrasti tajā redzamie apļi, un kā rezultātes tiek noteikta kameras relatīvā pozīcija pret mērķa centru. Algoritms ir izvēlēts vienkāršs, jo tas ir izstrādāts tikai šī darba demonstrācijas piemēram, un tā precizitāte var sasniegt 0,5 metru robežu, bet tas pilnībā apmierina demonstrācijas piemēra prasības. Algoritms pie katra izsaukuma saglabā iepriekšējā izsaukuma reizē atrasto apļa centru un radiusu, un izrēķināto kameras augstumu virs mērķa. Tas ļauj prognozēt iespējamo rezultātu tekošajā attēlā, un neņemt vērā tos apļus, kurus atrod algoritms, bet tie nav daļa no mērķa. Atkarībā no iepriekšējā augstuma tiek noteikts, kuru mērķa apli meklēt. Jo augstāk atrodas kamera, jo algoritms meklē lielāku apli, ja neatrod, tad meklē mazāku. Attēla apstrādei tiek izmantota iebūvēta attēla apstrādes funkcija, kas balstīta uz *Circle Hough Transform(30)*.

3.11. Kvadrotora sagatavošana lidojumam

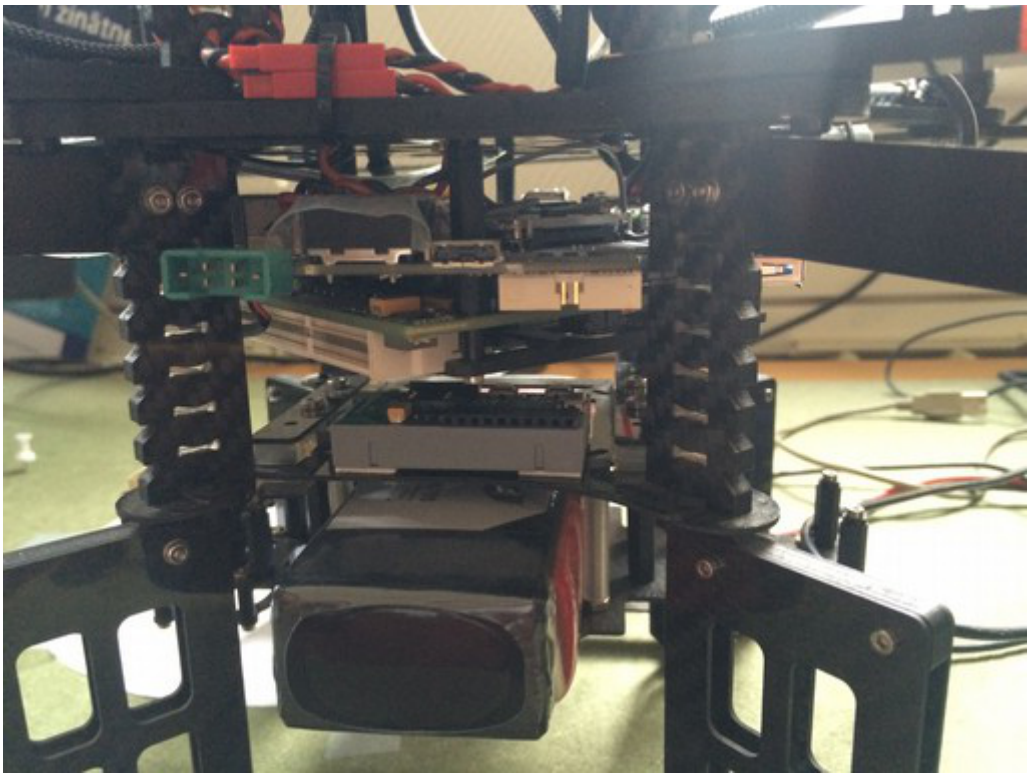
Tā kā saskarņu un *ROS* mezglu izstrāde ir notikusi pie darba galda un ar *AscTec Mastermind* var strādāt kā ar jebkuru personālo datoru, ir nepieciešams izveidotās daļas sagatavot lidojumam. Šajos darbos ietilpst:

- automātiska *ROS* mezglu palaišana ieslēdzot kvadrotora borta datoru;
- *IQRF* programmātora, kurš tiek izmantots bezvadu sensoru tīkla mezgla vadība, pievienošana un nostiprināšana pie kvadrotora korpusa.

Automātisko *ROS* mezglu palaišanu tika nolemts nodrošināt izmantojot *Linux* plaši pielietotās tehnoloģijas – *Cron* un *Screen*. Ir pievienots *Cron* darbs, kas izsauc *Bash* skriptu katru sistēmas startēšanas reizi,. Tas ir izdarīts izsaucot *"crontab -e"* un pēc tam pievienojot *"@reboot bash /home/asctec/bin/autostart_ros.sh"*. Skripts *"autostart_ros.sh"* satur komandas, kas izsauc *Screen* komandas ar parametriem *"-d -m"*, kas nozīmē, ka tiks izveidota jauna sesija un izsauktā funkcija tiks atvienota (no angļu *Detach*). Šis skripts ir redzams 12. pielikumā. 13. pielikumā ir redzams viens no caur *Screen* izsauktajiem skriptiem, kurš uzstāda *ROS* nepieciešamos parametrus un startē *ROS* procesus (šajā gadījumā *"roscore"*).

24. att. IQRF programmātors ar abpusējo līmlentu

Tā kā tika nolemts vadīt bezvadu sensoru tīkla mezglu izmantojot *IQRF* programmātoru un *USB*, to bija nepieciešams piestiprināt pie kvadrotora. Tas ir izdarīts izmantojot abpusējo



25. att. IQRF programmātors piestiprināts pie kvadrotora

līmlentu (skat. 24., 25. att.). Citi kvadrotora sensori un iekārtas no ražotāja ir piestiprinātas izmantojot plastmasas savilcējus vai plastmasas skrūves, bet prototipam šāds risinājums ir pieņemams.

REZULTĀTI UN SECINĀJUMI

Darbā izvirzītos mērķus ir izdevies izpildīt. Lielais uzdevums ir sadalīts daļās, un ir apskatītas dažādas tehnoloģijas un metodes sadarbības nodrošināšanai starp kvadrotoru un bezvadu sensoru tīklu. Ir apskatīta *IQRF* bezvadu sensoru tīkla platforma – tās konfigurācija un lietošana. Ir apskatīta *AscTec Pelican* platforma, kā arī *ROS*. Ir apskatītas metodes kvadrotora lokalizācijai iekštelpās. Ir izstrādāts demonstrācijas testa piemērs, kurā tiek pārbautītas apskatītās tehnoloģijas.

Risinot kvadrotora sadarbību ar bezvadu sensoru tīklu, parādījās daudz sīkāku tehnisku šķēršļu, kuru atrisināšana prasīja daudz laika.

Maģistra darba izstrādes laikā *R5-COP* projekta galvenais darbības virziens bija misijas izstrāde, kas tika darīts *SIL* modelī, tāpēc saistībā ar kvadrotora eksperimentiem, nebija iespējams paņemt gatavus rezultātus, bija nepieciešams iet *R5-COP* izstrādei pa priekšu.

Darba rezultātus, kas saistīti ar bezvadu sensoru tīklu un *ROS*, varēs iekļaut *R5-COP* paredzētajā demonstrācijas piemērā. Kvadrotora izpēte un mēģinājumi ierobežotajā telpā ir devusi daudz informācijas par turpmāk darāmajiem darbiem.

Pildot uzdevumu tika apgūtas un izmantotas tādas perspektīvas tehnoloģijas kā *ROS*, *IQRF* bezvadu sensoru tīkla platforma. Turpmākais darbs ar kvadrotoru noteikti ir jābalsta uz *ROS*. Turpmākajos darbos varētu ietilpt:

- lidojumi ārpus telpām, lai varētu tālāk attīstīt kvadrotora sadarbību ar bezvadu sensoru tīklu;
- pāriet uz datu apmaiņu ar augstā līmeņa procesoru kvadrotorā;
- lokalizācijas telpā uzlabošana – spēt lokalizēties nezināmā un iepriekš nesagatavotā telpā;
- senāk izstrādātās vizualizācijas attīstīšana – bezvadu sensoru tīkla un *ROS* integrācija.

Redzams, ka tēma ir aktuāla, un tai ir liela perspektīva, ir daudz iespēju to attīstīt turpmāk.

IZMANTOTĀ LITERATŪRA UN AVOTI

1. Khaitan SK, McCalley JD. Design Techniques and Applications of Cyberphysical Systems: A Survey. IEEE Syst J. 2015. gada jūnijā;9(2):350–65.

2. ECSEL JU. Multi-Annual Strategic Plan [Internets]. [citēts 2016. gada 21. maijā]. Iegūts no: http://www.ecsel-ju.eu/web/JU/ECSEL%20Work%20Plan.php

3. Valavanis KP, Vachtsevanos GJ. Handbook of Unmanned Aerial Vehicles. 2014. gada oktobrī; Iegūts no: http://dl.acm.org/citation.cfm?id=2755232

4. Puccinelli D, Haenggi M. Wireless sensor networks: applications and challenges of ubiquitous sensing. IEEE Circuits Syst Mag. 2005. gada;5(3):19–31.

5. Artemis. R5-COP at a glance [Internets]. Iegūts no: http://www.r5-cop.eu/en/project/r5-cop-glance/

6. de Freitas EP, Heimfarth T, Netto IF, Lino CE, Pereira CE, Ferreira AM, u.c. UAV relay network to support WSN connectivity. No: International Congress on Ultra Modern Telecommunications and Control Systems [Internets]. IEEE; 2010. lpp. 309–14. Iegūts no: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=5676621

7. White BA, Tsourdos A, Ashokaraj I, Subchan S, Zbikowski R. Contaminant Cloud Boundary Monitoring Using Network of UAV Sensors. IEEE Sens J. 2008. gada oktobrī;8(10):1681–92.

8. Ahir D, Patel T. Unmanned Aerial Vehicle Technology Using Wireless Sensor Network [Internets]. [citēts 2016. gada 21. maijā]. Iegūts no: http://www.ijetae.com/files/Volume4Issue12/IJETAE_1214_79.pdf

9. Ministru kabineta noteikumi Nr.656. Kārtība, kādā veicami bezpilota gaisa kuģu un tādu cita veida lidaparātu lidojumi, kuri nav kvalificējami kā gaisa kuģi.

10. Akyildiz IF, Kasimoglu IH. Wireless sensor and actor networks: research challenges. Ad Hoc Netw. 2004. gada oktobrī;2(4):351–67.

11. AscTec. AscTec Pelican [Internets]. [citēts 2016. gada 21. maijā]. Iegūts no: http://wiki.asctec.de/display/AR/AscTec+Pelican

12. AscTec. AscTec Mastermind [Internets]. [citēts 2016. gada 21. maijā]. Iegūts no: http://wiki.asctec.de/display/AR/AscTec+Mastermind

13. AscTec. AscTec Pelican Overview [Internets]. Iegūts no: http://www.asctec.de/en/uav-uas-drones-rpas-roav/asctec-pelican/

14. AscTec. SDK Manual [Internets]. Iegūts no: http://wiki.asctec.de/display/AR/SDK+Manual

15. ROS. About ROS [Internets]. [citēts 2016. gada 21. maijā]. Iegūts no: http://www.ros.org/about-ros/

16. ROS. ROS Concepts [Internets]. [citēts 2016. gada 21. maijā]. Iegūts no: http://wiki.ros.org/ROS/Concepts

17. MICRORISC. IQRF Technology [Internets]. [citēts 2016. gada 21. maijā]. Iegūts no: http://www.iqrf.org/technology

18. MICRORISC. IQMESH [Internets]. [citēts 2016. gada 21. maijā]. Iegūts no: http://www.iqrf.org/technology/iqmesh

19. MICRORISC. CDC Implementation in IQRF USB devices User's guide.

20. AscTec. Communicating with the Low Level Processor [Internets]. [citēts 2016. gada 21. maijā]. Iegūts no: http://wiki.asctec.de/display/AR/Communicating+with+the+Low+Level+Processor

21. Achtelik M, Bachrach A, He R, Prentice S, Roy N. Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments. No: Gerhart GR, Gage DW, Shoemaker CM, redaktors. SPIE Defense, Security, and Sensing [Internets]. International Society for Optics and Photonics; 2009. lpp. 733219–733219. Iegūts no: http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=778573

22. Scholtz RA. Ranging in a dense multipath environment using an UWB radio link. IEEE J Sel Areas Commun. 2002. gada decembrī;20(9):1677–83.

23. Barsocchi P, Lenzi S, Chessa S, Giunta G. A Novel Approach to Indoor RSSI Localization by Automatic Calibration of the Wireless Propagation Model. No: VTC Spring 2009 - IEEE 69th Vehicular Technology Conference [Internets]. IEEE; 2009. lpp. 1–5. Iegūts no: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=5073315

24. MICRORISC. IQRF DPA Quick Start Guide [Internets]. [citēts 2016. gada 21. maijā]. Iegūts no: http://www.iqrf.org/weben/ downloads.php?id=372

25. MICRORISC. IQRF DPA Framework Technical guide.

26. ROS. asctec_autopilot package [Internets]. [citēts 2016. gada 21. maijā]. Iegūts no: http://wiki.ros.org/asctec_autopilot

27. ROS. Creating a ROS Package [Internets]. [citēts 2016. gada 21. maijā]. Iegūts no: http://wiki.ros.org/ROS/Tutorials/CreatingPackage

28. MathWorks. Create Custom Messages from ROS Package [Internets]. [citēts 2016. gada 21. maijā]. Iegūts no: http://se.mathworks.com/help/robotics/ug/create-custom-messages-from-ros-package.html

29. MathWorks. Install Robotics System Toolbox Add-ons [Internets]. [citēts 2016. gada 21. maijā]. Iegūts no: http://se.mathworks.com/help/robotics/ug/install-robotics-system-toolbox-support-packages.html

30. MathWorks. imfindcircles [Internets]. [citēts 2016. gada 22. maijā]. Iegūts no: http://se.mathworks.com/help/images/ref/imfindcircles.html

PIELIKUMI

1. Pielikums cdc.py

```
import serial
import threading
import time

class CDC:
    BAUDRATE = 9600
    PARITY = serial.PARITY_NONE
    STOPBITS = serial.STOPBITS_ONE
    BYTESIZE = serial.EIGHTBITS

    def __init__(self, path, iqrf):
        self.ser = serial.Serial(
            port = path,
            baudrate = self.BAUDRATE,
            parity = self.PARITY,
            stopbits = self.STOPBITS,
            bytesize = self.BYTESIZE,
            timeout = 1
        )
        self.iqrf = iqrf
        self.read_thread = threading.Thread(target=self.read)
        self.read_thread.start()

    def read(self):
        message = []
        message_types = {"DR" : self.data_read}
        while(self.ser.isOpen() == True):
            byte = self.ser.read()
            if(len(byte) > 0):
                message.append(byte)
                if(byte == '\r'):
                    message_type = ''.join(message)[1:3]
                    if(message_type in message_types):
                        message_types[message_type](message)
                    message = []
```

```
    def write(self, data):
        time.sleep(0.01)
        message = ">DS" + chr(len(data)) + ':' + ''.join(chr(i) for i in data) + '\r'
        self.ser.write(message)

    def data_read(self, data):
        if ''.join(data[1:3]) == 'DR':
            self.iqrf.message_received(map(ord, data[5:-1]))

    def close(self):
        self.ser.close()
        self.read_thread.join()
```

2. Pielikums iqrf.py

```
from cdc import CDC
from abc import ABCMeta, abstractmethod

class IQRF:
    def __init__(self, path):
        self.cdc = CDC(path, self)

    @abstractmethod
    def message_received(self, data):
        return

    def close(self):
        self.cdc.close()
```

3. Pielikums coordinator.py

```
from iqrf import IQRF
import time
import threading
import numpy as np
import struct
```

```
from Queue import Queue

class Coordinator(IQRF):
    NODE = 0x01
    PERIPHERAL_NUMBER = 0x08
    PERIPHERAL_COMMAND = 0x00
    TIMEOUT = 0x32
    def __init__(self, path):
        IQRF.__init__(self, path)
        self.messages = Queue()
        self.running = True
        self.coordinate_thread = threading.Thread(target=self.coordinate)
        self.coordinate_thread.start()
        import time

    def decode(self, message, element_size):
        decoded = []
        i = 0
        while(i <= len(message) - element_size):
            element = ''.join(chr(j) for j in message[i:i+element_size])
            element = struct.unpack('!i', element)[0]
            decoded.append(element)
            i += element_size
        return decoded

    def message_received(self, data):
        if(data[3] == 0x80):
            decoded = self.decode(data[8:len(data)], 4)
            if(len(decoded) > 0):
                message_type = decoded[0]
                if(message_type == 1):
                    nick = decoded[1]
                    roll = decoded[2]
                    yaw = decoded[3]
                    print(nick)
                    print(roll)
                    print(yaw)
                elif(message_type == 2):
                    node = decoded[1]
                    pnum = decoded[2]
                    pcmd = decoded[3]
```

```
            header = Coordinator.compose_message_header(node, pnum, pcmd)
            self.messages.put(header)

    def coordinate(self):
        header = Coordinator.compose_message_header(self.NODE,
self.PERIPHERAL_NUMBER,
                                    self.PERIPHERAL_COMMAND)
        body = [self.TIMEOUT, 1]
        message = header + body
        while self.running == True:
            if(self.messages.empty() != True):
                self.cdc.write(self.messages.get_nowait())
            else:
                self.cdc.write(message)
            time.sleep(1)

    def close(self):
        self.running = False
        self.coordinate_thread.join()
        IQRF.close(self)

    @staticmethod
    def compose_message_header(node, peripheral_number, command):
        return [node & 0xFF, (node >> 8) & 0xFF, peripheral_number, command,
            0xFF, 0xFF]

coord = Coordinator("/dev/tty.usbmodem1411")
raw_input("Press Enter to continue...")
coord.close()
```

4. Pielikums node.py

```
from iqrf import IQRF

class Node(IQRF):
    def message_received(self, data):
        print(data)
        self.cdc.write([100])
```

```
node = Node("/dev/tty.usbmodem1411")
raw_input("Press Enter to continue...")
node.close()
```

5. Pielikums wsn_listener.py

```
import rospy
from std_msgs.msg import String
from iqrf import IQRF
import roslib
import struct
import ctypes

roslib.load_manifest('asctec_msgs')
from asctec_msgs.msg import IMUCalcData

class WSN_Listener(IQRF):
    def __init__(self):
        IQRF.__init__(self, "/dev/ttyACM0")
        self.angle_nick = 0
        self.angle_roll = 0
        self.angle_yaw = 0
        rospy.init_node('wsn_listener', anonymous=True)
        rospy.Subscriber('/asctec/IMU_CALCDATA', IMUCalcData, self.got_imu_data)
        rospy.spin()
    def got_imu_data(self, data):
        self.angle_nick = data.angle_nick
        self.angle_roll = data.angle_roll
        self.angle_yaw = data.angle_yaw
    def message_received(self, data):
        message = ''
        message += struct.pack("!i", self.angle_nick)
        message += struct.pack("!i", self.angle_roll)
        message += struct.pack("!i", self.angle_yaw)
        self.cdc.write(map(ord, message))

if __name__ == '__main__':
    WSN_Listener()
```

6. Pielikums iqrf_ros_node.py

```
import rospy
from std_msgs.msg import String
from iqrf import IQRF
import roslib
import struct
import ctypes

class Node(IQRF):
    def __init__(self):
        rospy.init_node('iqrf_node', anonymous=True)
        rospy.Subscriber('/iqrf/CMD', String, self.got_new_cmd)
        self.pub = rospy.Publisher('/iqrf/STATUS', String, queue_size=10)
        self.node = 0
        self.pnum = 0
        self.pcmd = 0
        self.message_type = 0
        IQRF.__init__(self, "/dev/serial/by-id/usb-MICRORISC_s.r.o._0002-if00")
        rospy.spin()

    def got_new_cmd(self, message):
        data = map(int, message.data.split(','))
        self.node = data[0]
        self.pnum = data[1]
        self.pcmd = data[2]
        self.message_type = 2

    def message_received(self, data):
        message = ''
        message += struct.pack("!i", self.message_type)
        message += struct.pack("!i", self.node)
        message += struct.pack("!i", self.pnum)
        message += struct.pack("!i", self.pcmd)
        self.cdc.write(map(ord, message))

        if(self.message_type != 0):
            self.message_type = 0
            output = str(self.node) + ',' + str(self.pnum) + ',' + str(self.pcmd)
            self.pub.publish(String(output))
```

```
if __name__ == '__main__':
    Node()
```

7. Pielikums kvadrotora datu struktūras

Pielikumā iekļautais programmatūras kods ir iegūts no kvadrotora ražotāja(20). Pielikums satur kvadrotora datu struktūras, kuras tiek izmantotas *ROS* saziņai.

```
struct CTRL_INPUT {

    short pitch; //pitch input: -2047..+2047 (0=neutral)
    short roll; //roll input: -2047..+2047 (0=neutral)
    short yaw; //(=R/C Stick input) -2047..+2047 (0=neutral)
    short thrust; //collective: 0..4095 = 0..100%
    short ctrl; /*control byte:
        bit 0: pitch control enabled
        bit 1: roll control enabled
        bit 2: yaw control enabled
        bit 3: thrust control enabled
        bit 4: height control enabled
        bit 5: GPS position control enabled
        */
short chksum;
};
struct CTRL_INPUT CTRL_Input;



struct LL_STATUS

{
    //battery voltages in mV
    short battery_voltage_1;
    short battery_voltage_2;
    //don't care
    short status;
    //Controller cycles per second (should be ?1000)
    short cpu_load;
    //don't care
    char compass_enabled;
    char chksum_error;
    char flying;
    char motors_on;
    short flightMode;
    //Time motors are turning
    short up_time;
```

```
};
struct IMU_RAWDATA
{
  //pressure sensor 24-bit value, not scaled but bias free
  int pressure;
  //16-bit gyro readings; 32768 = 2.5V
  short gyro_x;
  short gyro_y;
  short gyro_z;
  //10-bit magnetic field sensor readings
  short mag_x;
  short mag_y;
  short mag_z;
  //16-bit accelerometer readings
  short acc_x;
  short acc_y;
  short acc_z;
  //16-bit temperature measurement using yaw-gyro internal sensor
  unsigned short temp_gyro;
  //16-bit temperature measurement using ADC internal sensor
  unsigned int temp_ADC;
};
struct IMU_CALCDATA
{
  //angles derived by integration of gyro_outputs, drift compensated by data fusion;
  //-90000..+90000 pitch(nick) and roll, 0..360000 yaw; 1000 = 1 degree
  int angle_nick;
  int angle_roll;
  int angle_yaw;
  //angular velocities, raw values 16 bit but bias free
  int angvel_nick;
  int angvel_roll;
  int angvel_yaw;
  //acc-sensor outputs, calibrated: -10000..+10000 = -1g..+1g
  short acc_x_calib;
  short acc_y_calib;
  short acc_z_calib;
  //horizontal / vertical accelerations: -10000..+10000 = -1g..+1g
  short acc_x;
  short acc_y;
  short acc_z;
  //reference angles derived by accelerations only: -90000..+90000; 1000 = 1 degree
  int acc_angle_nick;
  int acc_angle_roll;
  //total acceleration measured (10000 = 1g)
  int acc_absolute_value;
  //magnetic field sensors output, offset free and scaled;
  //units not determined, as only the direction of the field vector is taken into account
  int Hx;
  int Hy;
  int Hz;
```

```
    //compass reading: angle reference for angle_yaw: 0..360000; 1000 = 1 degree
    int mag_heading;
    //pseudo speed measurements: integrated accelerations, pulled towards zero;
    //units unknown; used for short-term position stabilization
    int speed_x;
    int speed_y;
    int speed_z;
    //height in mm (after data fusion)
    int height;
    //diff. height in mm/s (after data fusion)
    int dheight;
    //diff. height measured by the pressure sensor mm/s
    int dheight_reference;
    //height measured by the pressure sensor mm
    int height_reference;
};
struct GPS_DATA
{
    //latitude/longitude in deg * 10^7
    int latitude;
    int longitude;
    //GPS height in mm
    int height;
    //speed in x (E/W) and y(N/S) in mm/s
    int speed_x;
    int speed_y;
    //GPS heading in deg * 1000
    int heading;
    //accuracy estimates in mm and mm/s
    unsigned int horizontal_accuracy;
    unsigned int vertical_accuracy;
    unsigned int speed_accuracy;
    //number of satellite vehicles used in NAV solution
    unsigned int numSV;
    // GPS status information; 0x03 = valid GPS fix
    int status;
};
struct GPS_DATA_ADVANCED
{
    //latitude/longitude in deg * 10^7
    int latitude;
    int longitude;
    //GPS height in mm
    int height;
    //speed in x (E/W) and y(N/S) in mm/s
    int speed_x;
    int speed_y;
    //GPS heading in deg * 1000
    int heading;
    //accuracy estimates in mm and mm/s
    unsigned int horizontal_accuracy;
```

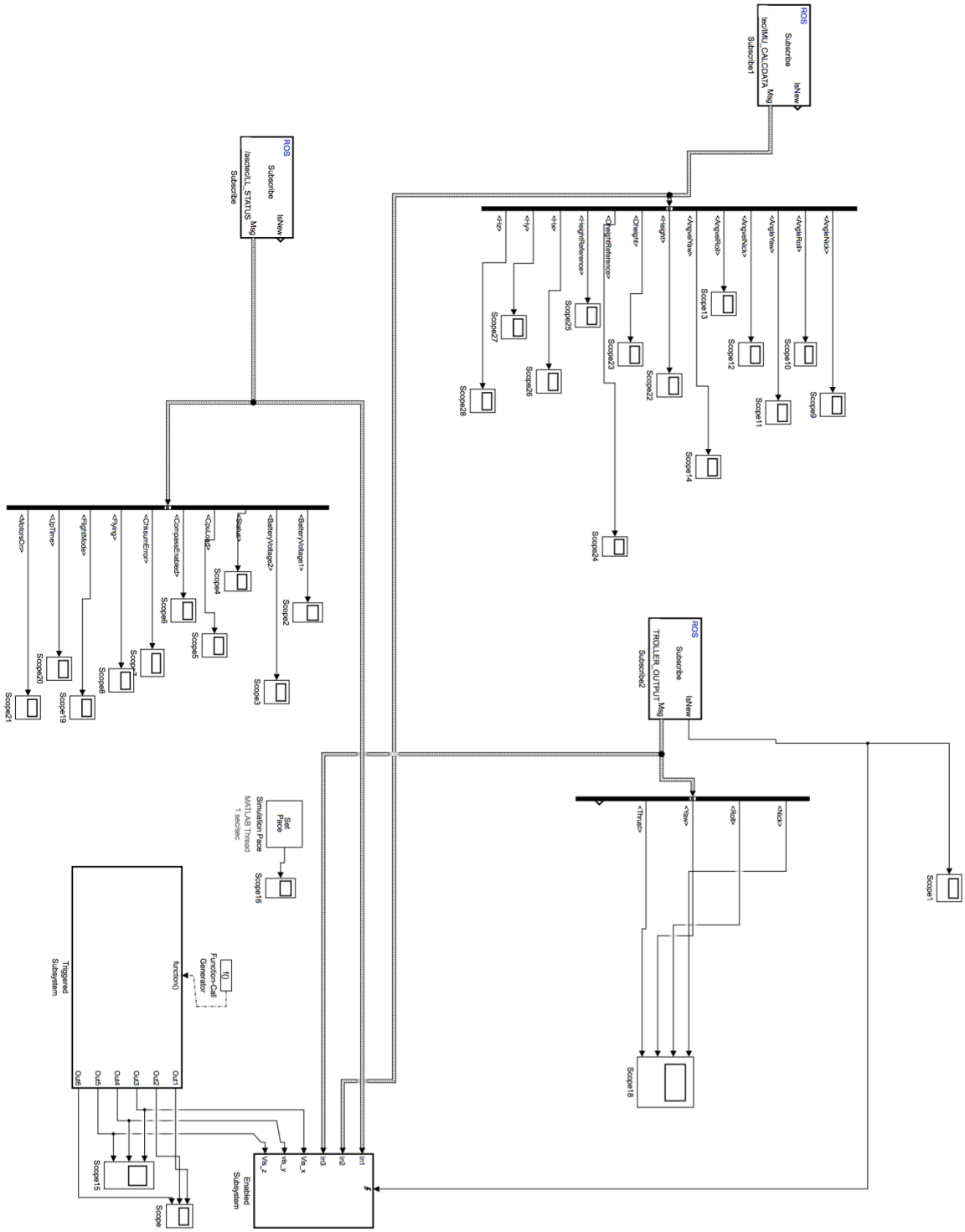
```
    unsigned int vertical_accuracy;
    unsigned int speed_accuracy;
    //number of satellite vehicles used in NAV solution
    unsigned int numSV;
    //GPS status information; 0x03 = valid GPS fix
    int status;
    //coordinates of current origin in deg * 10^7
    int latitude_best_estimate;
    int longitude_best_estimate;
    //velocities in X (E/W) and Y (N/S) after data fusion
    int speed_x_best_estimate;
    int speed_y_best_estimate;
};
struct RC_DATA
{
    //channels as read from R/C receiver
    unsigned short channels_in[8];
    //channels bias free, remapped and scaled to 0..4095
    unsigned short channels_out[8];
    //Indicator for valid R/C receiption
    unsigned char lock;
};
struct CONTROLLER_OUTPUT
{
    //attitude controller outputs; 0..200 = -100 ..+100%
    int nick;
    int roll;
    int yaw;
    //current thrust (height controller output); 0..200 = 0..100%
    int thrust;
};
```

8. Pielikums *MATLAB* attēla apstrādes skripts

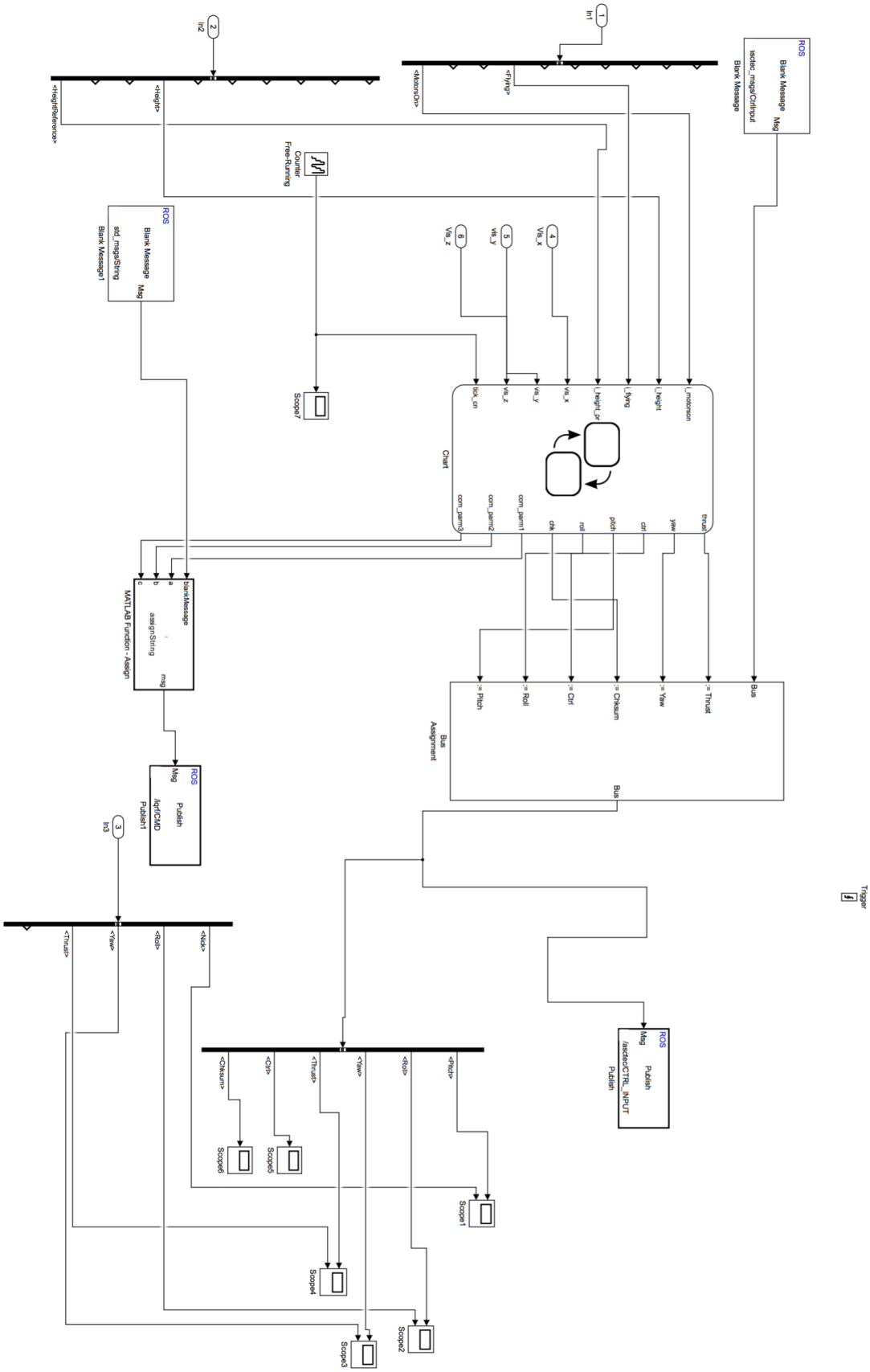
```
function subCallback(src,msg)

global imageFormatted;
global x;
global v_x;
global v_y;
global v_z;
global center_x;
global center_y;
global radiuss;
global initial_v;
global kluda;
global pedejais_redzetais;
global pirmais_mekl;

if (initial_v==0)
minr1=70; %80
maxr1=100; %100
cxmi=00;
cxma=580;
cymi=00;
cyma=500;
tiek_mekl=1;
else
 minr1=round(radiuss)-15;
 if (minr1<0)
    minr1=0;
 end;
 maxr1=round(radiuss)+15;
 cxmi=center_x-45;
cxma=center_x+45;
cymi=center_y-45;
cyma=center_y+45;
tiek_mekl=pirmais_mekl;
end

for j=drange(1:3);
if (tiek_mekl==1)
   radp=25;
   radcm=0.4;
   zz=25;
elseif (tiek_mekl==2)
    radcm=2.8;
    radp=150;
    zz=25;
else
    radcm=11;
    radp=150;
```

```
    zz=100;
end
imageFormatted2 = readImage(msg);

if (tiek_mekl==2)
[centers, radii] = imfindcircles(imageFormatted2,[minr1 maxr1],'ObjectPolarity',
'dark','Sensitivity',0.95);
else
[centers, radii] = imfindcircles(imageFormatted2,[minr1 maxr1],'ObjectPolarity',
'bright','Sensitivity',0.98);
end

ff=0;
for i=drange(1:length(radii));
   if (centers(i,1)>cxmi) && (centers(i,1)<cxma)
       if (centers(i,2)>cymi) && (centers(i,2)<cyma)
       center_x=double(centers(i,1));
       center_y=double(centers(i,2));
       radiuss=double(radii(i));
       ff=1;
      end
   end
end
    if (ff==1)
       if (initial_v==0)
          initial_v=1;
          pedejais_redzetais=1;
       end
      v_z=(zz*radp)/radiuss;
      v_x=(center_x-360)*(radcm/radiuss);
      v_y=(center_y-280)*(radcm/radiuss);
      kluda=0;
      if (v_z>40)
         pirmais_mekl=3;  %baltais aplis
      elseif (v_z>12)
          pirmais_mekl=2;  %melnais aplis
      else
          pirmais_mekl=1;
      end;
      break;
    else
    end
    if (tiek_mekl==1)
     kluda=kluda+1;
     break;
    else
       tiek_mekl=tiek_mekl-1;
    end;
end

x=x+1;
```

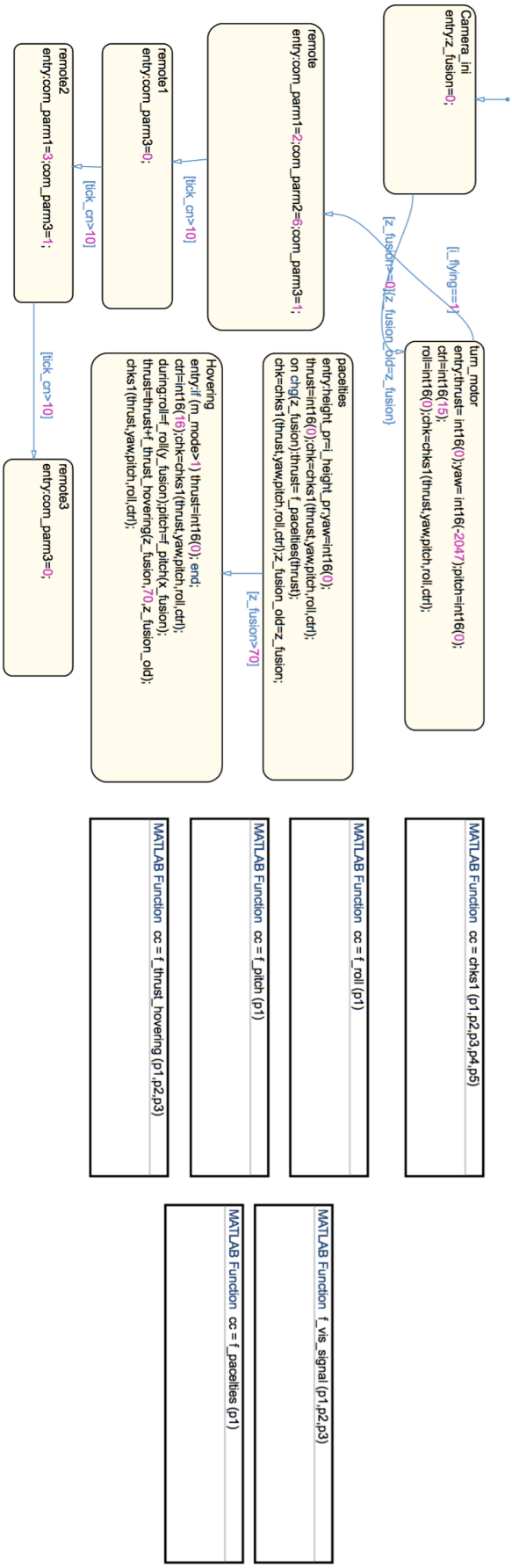
9. Pielikums *MATLAB Simulink ROS* datu ieguves modelis



10. Pielikums *MATLAB Simulink* demonstrācijas modelis



11. Pielikums MATLAB Simulink Stateflow misijas vadības modelis



12. Pielikums ros_autostart.sh

```
LOG_FILE=/home/asctec/log/ros_autostart.txt

set -e
set -v

{

screen -d -m bash /home/asctec/bin/start_roscore.sh
screen -d -m bash /home/asctec/bin/start_asctec_autopilot.sh
screen -d -m bash /home/asctec/bin/start_cameras.sh
screen -d -m bash /home/asctec/bin/start_iqrf.sh

} &>> ${LOG_FILE}
```

13. Pielikums start_roscore.sh

```
LOG_FILE=/home/asctec/log/start_roscore.txt

set -e

{
   source /opt/ros/jade/setup.bash
   source /home/asctec/catkin_ws/devel/setup.bash
   export ROS_WORKSPACE=/home/asctec/catkin_ws

   export ROS_MASTER_URI=http://localhost:11311/

   sleep 5

} &>> ${LOG_FILE}

set -v

{
   roscore

} &>> ${LOG_FILE}
```

DOKUMENTĀRĀ LAPA

Maģistra darbs: **Kvadrotora sadarbība ar bezvadu sensoru tīklu**

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: _____ / Klāvs Taube/
(Autora paraksts)

Ar savu parakstu apliecinu, ka esmu lasījis augstāk minēto maģistra darbu un atzīstu to par **p i e m ē r o t u / n e p i e m ē r o t u** (nevajadzīgo svītrot) aizstāvēšanai Latvijas Universitātes datorzinātņu maģistrantūrā.

Darba vadītājs: _____ / pētnieks Mg. dat. Artis Gaujēns /
(Vadītāja paraksts)

Darbs iesniegts **maģistrantūras sekretariātā** _____.
(Iesniegšanas datums)

Ar šo es apliecinu, ka darba elektroniskā versija ir augšupielādēta LU informatīvajā sistēmā.

Studiju metodiķe: _____.
(Metodiķes paraksts)

Recenzents: _____ / pētnieks Dr. dat. Artis Mednis/
(Akad.amats, zin.grāds, vārds, uzvārds)

Darbs aizstāvēts maģistra gala pārbaudījuma komisijas sēdē

_____ prot. Nr. _____

(Darba aizstāvēšanas datums)

Komisijas sekretārs: _____

(Sekretāra paraksts)