UNIVERSITATEA TITU MAIORESCU
FACULTATEA: INFORMATICĂ
DEPARTAMENT: INFORMATICĂ
Programa de studii: INFORMATICĂ
DISCIPLINA: INTELIGENȚĂ ARTIFICIALĂ

IA - *Testul de evaluare nr.* 19

Payload roboți mobili

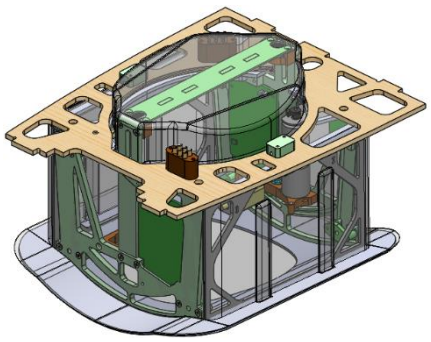
Grupa	Numele și prenumele	Semnătură student	Notă evaluare

Data: ____/____/_____

CS-I dr.ing.
Lucian Ștefăniță GRIGORE

Conf.dr.ing
Iustin PRIESCU

Ș.L.dr.ing.
Dan-Laurențiu GRECU



Cuprins

1. INTRODUCERE

"Payload"-urile pot fi de două tipuri:

a) nedispensabile: senzori, camere video, etc., care rămân cu aeronava,

b) ***dispensabile***, cum ar fi livrarea de materiale în zone cu teren dificil sau a unor kit-uri de urgență.

Implicarea in industria logisticii a UAV-urilor cu payload dispensabil:

- *In domeniul urban*. De cele mai multe ori urbanizarea unui oraș nu poate tine pasul cu ritmul de dezvoltare a acestuia. De aici apare nevoia de dezvoltare a unor sisteme de transport alternative. De exemplu, compania de curierat aerian Flirtey intenționează să introducă primul UAV comercial pentru livrare. Clienții vor primi o notificare smartphone care le va permite să urmări coletul prin GPS, și să primească coletul direct la o locație in aer liber. Odată ce UAV ajunge la destinația de livrare, acesta plutește și cu grijă coboară coletul printr-un mecanism de livrare care este atașat la un cablu retractabil

Fig. 1-1 Textbook delivery service via UAV; **Source:** Web2Carz

Fig. 1-2 The DHL Paketcopter delivering a parcel in Bonn

- *In domeniul Rural*.

Potențialul tehnologiei UAV este evidentă și în zonele rurale cu infrastructură slabă sau provocatoare din punct de vedere a condițiilor geografice.

Recent, Google a dezvăluit cele mai recente progrese ale programul său numit proiectul Wing system de livrare autonom, sistem capabile de a aduce colete in timp foarte mic. Google lucrează la acest proiect de aproape doi ani si este deja testează în prezent în Queensland, Australia.

Fig. 1-3 Google‘s drone deliveries; **Source:** Engadget

Fig. 1-4 DHL Paketkopter flies to German isle; **Source:** Stern

Un UAV numit DHL Paketkopter a fost sub controlul manual al unui operator, în orice moment, pentru a îndeplini cerințele impuse dar din punct de vedere tehnic, ar fi putut operat cu autonomie deplină urmând puncte intermediare GPS. Acest vehicul a fost echipat cu un mecanism de eliberare care să îi permită să pună jos coletul prin telecomandă sau pre-programat

Principalele probleme ce trebuie analizate:

A. ***Metoda de lansare a „ payload ”-ului***

B. ***Zona de dispunere a „ payload ”-ului in cadrul sistemului UAV***

C. ***Sistem de fixare/eliberare „ payload ”***

D. ***„ Payload ”-ul*** (forma carcasa, greutate utila si totala, materiale carcasa, sisteme suplimentare de protecție)

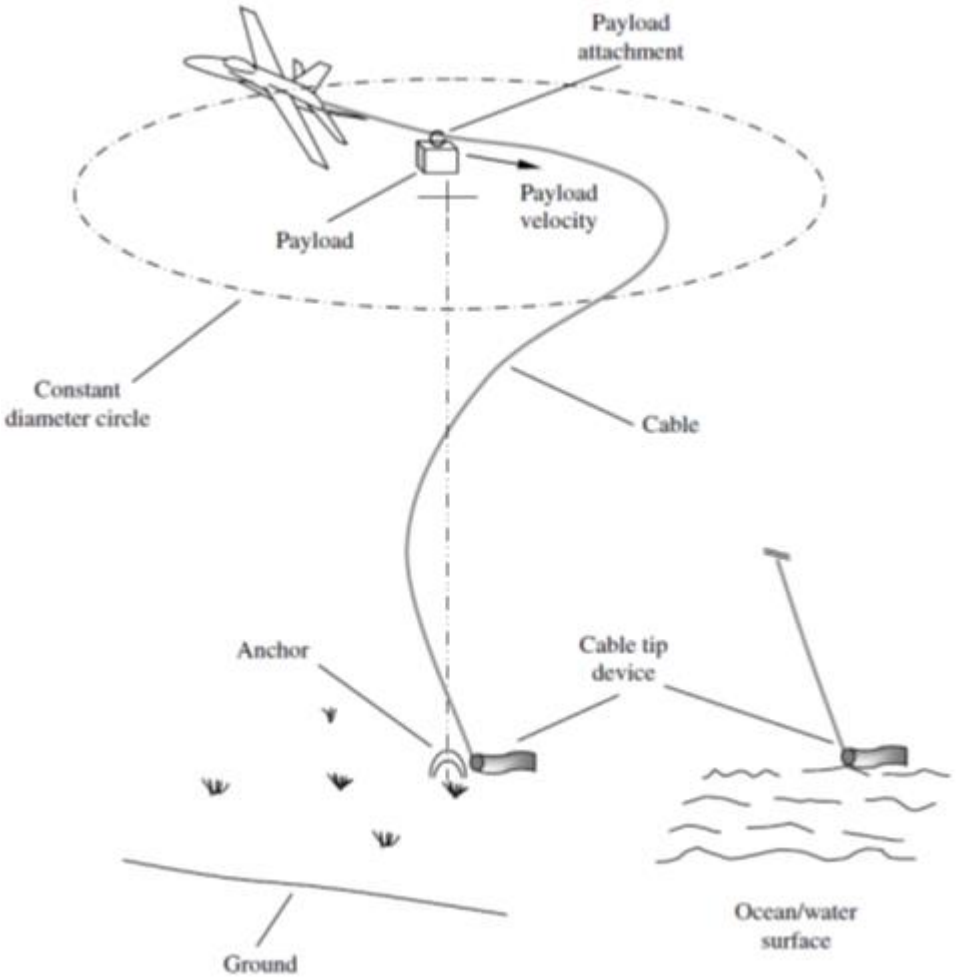


Fig. 1-5 Glisare pe fir

A. *Metoda de lansare a „ payload ”-ului*

Principalele metode de lansare a "payload"-ului pot fi împărțite în două grupe:

a) neghidate

b) ghidate.

Principalele metode de lansare neghidate sunt:

a1) Cădere liberă fără parașuta

Această metodă este de departe cea mai simplă. Este nevoie de nimic altceva decât de un dispozitiv care să deconecteze sarcina utilă de UAV și controlul acestui dispozitiv. Asta înseamnă că există mai puțin echipament care se poate defecta și care trebuie să fie întreținut. Principalele dezavantaje ale acestei metode sunt necesitatea determinării cu mare precizie a punctului de eliberarea a încărcăturii iar la impactul cu solul există posibilitatea de deteriorare a încărcăturii.

a2) Cădere liberă cu parașuta

Această metodă are simplitatea metodei precedente , dar oferă un mecanism care previne ca încărcătura să fie distrusă de cădere, și anume parașuta. Parașuta micșorează viteza de coborâre a încărcăturii prin creșterea rezistenței aerului și astfel micșorează semnificativ forța de impact cu solul. Principalele dezavantaje ale acestei metode sunt micșorarea spațiului de transport (spațiul ocupat de parașută) și prin faptul ca este mărită suprafața de contact cu aerul micșorarea semnificativă a preciziei de lansare a încărcăturii.

Principalele metode de lansare ghidate sunt: glisare pe un fir în spiral sau folosirea unui fir derulat de pe un tambur

Aceste metode sunt mult mai complexe decât precedentele, deoarece necesită ca firul să fie eliberat și un alt dispozitiv care eliberează sarcina utilă pe fir. În plus este necesară și o eventuală ancorare a firului la sol sau o greutate mare pe capătul acestuia.

B. *Zona de dispunere a „ payload ”-ului in cadrul sistemului UAV*

Capacitatea de transport a “payload”-ului unui UAV este important. Cu toate acestea, poziția acesteia este și mai importanta iar aceasta variază în funcție de tipul de “payload”:

Studiu literaturii oferă 2 soluții pentru montarea unui sistem de distribuție a unor încărcături, și anume:

a) În interiorul aeronavei

b) În exteriorul aeronavei

- Poziționarea la nivelul abdomenului
- Poziționare la nivelul nasului
- Poziționare la nivelul aripilor

- Pozitionare la nivelul cozii

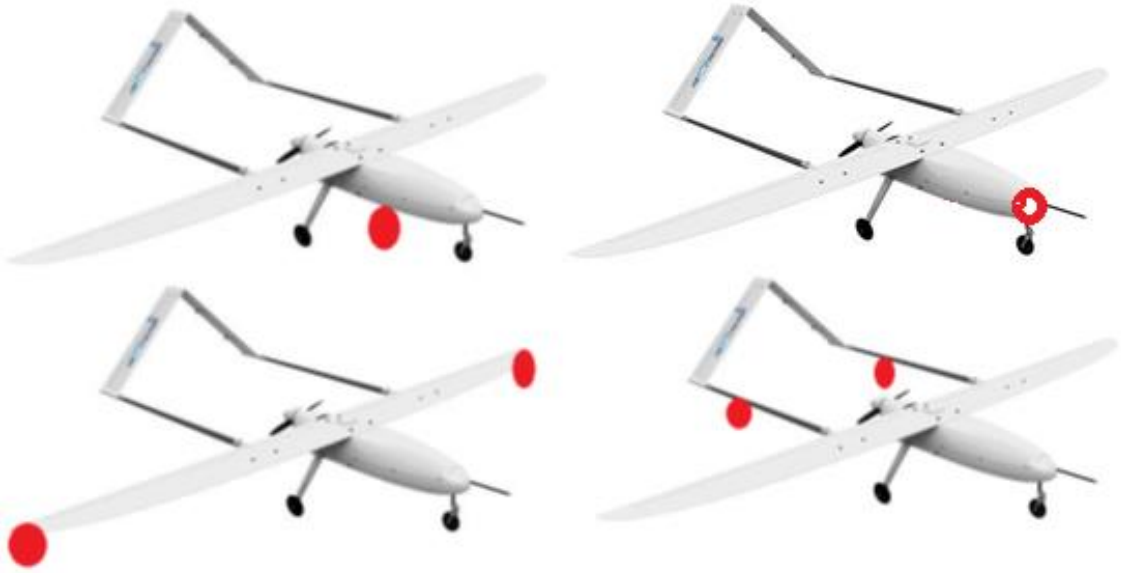


Fig. 1-6 Sisteme de distribuție a încărcăturii pe UAVs

Plasarea in interiorul aeronavei prezinta unele avantaje: nu modifică dinamica zborului dar are și numeroase dezavantaje: necesită un mecanism de escamotare înainte de eliberare, limitează forma și greutatea încărcăturii la spațiul disponibil în interiorul aeronavei și dacă nu există un spațiu suplimentar trebuie să înlocuiască payloadul senzorial.

In schimb prin plasarea in exteriorul aeronavei multe din aceste inconveniente dispar dar metoda de lansare a aeronavei poate fi decisiva in alegerea dispunerii paylodului in cadrul aeronavei.

A. *Mecanismul de eliberare "payload"*

În cazul în care UAV a urmat traiectoria optimă și a ajuns la locul lansării aceasta trebuie să fie în măsură să elibereze "payload"-ul. În cazul în care UAV nu are un mecanism de eliberare a "payload"-ul trebuie atașat unul extern.

Cerințele pentru un astfel de mecanism sunt:

1. Trebuie să elibereze "payload"-ul la momentul potrivit și fără să afecteze ireversibil mișcarea UAV-ului
2. Nu trebuie să fie prea greu sau un consumator mare de energie
3. Ar trebui să fie ușor de montat si controlat

Modele comerciale de mecanisme de eliberare "payload"

EFLA405 Servoless Payload Release are un design care îl face ușor de montat și folosit. Se compune din două module, unul mai mare care conține partea electronică si unul mai mic (pasiv) care se va monta pe „payload". Ambele module au găuri în corpurile lor, ceea ce le face foarte ușor

de montat. Cele două module sunt legate printr-un știft de metal. Când mecanismul de eliberare primește comanda tip PWM în intervalul între 0-1 V, știftul se retrage din partea pasivă iar cele două module sunt deconectate.

Fig. 1-7 EFLA Servoless Payload Release
http://www.horizonhobby.com/product/airplanes/airplane-accessories/other-accessories/servoless-payload-release-efla405

Legătura cu sistemul UAV se face prin trei cabluri, alimentare 5 V ,masă și semnalul de control. EFLA405 Servoless Payload poate transporta pana la 340 g ca sarcină utilă.

Fig. 1-8 Quanum RTR Bomb System
http://www.hobbyking.com/hobbyking/store/__10624__Quanum_RTR_Bomb_System_1_6_scale_Plug_n_Drop.html

Principiul de funcționare este asemănător cu cel precedent diferind numai prin forma si prin faptul ca se folosește de cârlige pentru a rigidiza si elibera încărcătura.

Acest lucru poate fi folosit ca un mecanism de lansare deși nu acesta a fost scopul initial. O ilustrare a acestui mecanism este prezentat în figura următoare.

Fig. 1-9 Tinder Rocketry Peregrine

Când iîncărcătura de CO2 este eliberata arcul comprimat se destinde si desprinde carcasa.

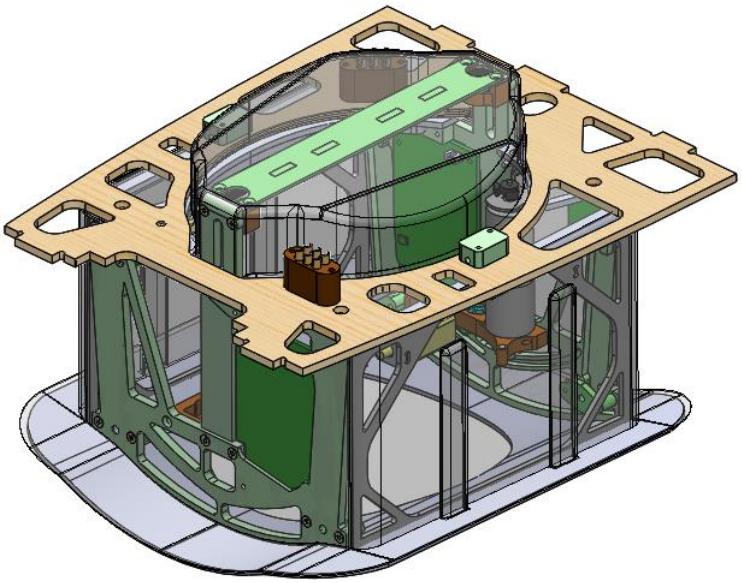
C. ***„ Payload ”-ul***

- *forma carcasa* - pentru poziționarea in interior determinata de spațiul existent
- *greutate utila* – determinata de aerodinamica zborului si carcasa,
- *materiale carcasa* – materialele din care este confecționată carcasa trebuie sa fie cat mai ușoare dar destul de rezistente la impactul cu solul
- *sisteme suplimentare de protecție* – daca se optează pentru varianta cu parașuta trebuie determinata cu precizie locul de dispunere si modalitatea de deschidere a acesteia

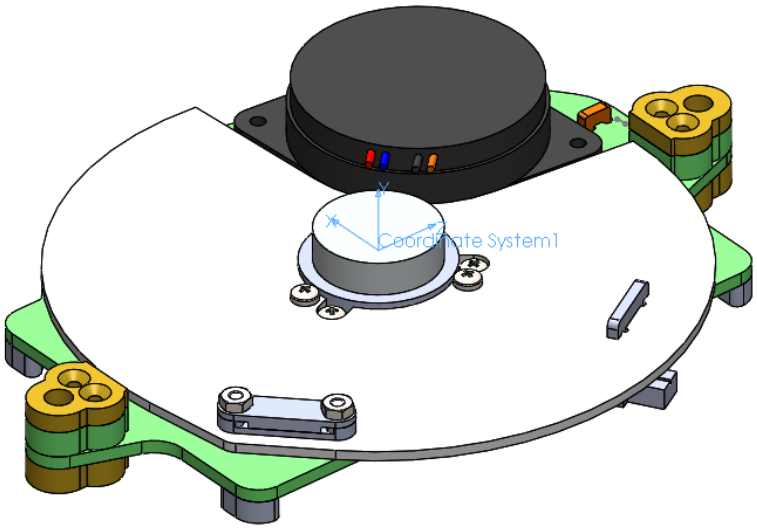
2. SIMULARE VIRTUALĂ PAYLOAD HIRRUS-V1

Părțile componente ale ansamblului

Carcasa – partea fixa



Platforma fixa a camerei



Mass = 159.84 grams

Center of mass: (millimeters)

X = 8.74

Y = -5.55

Z = 9.19

Principal axes of inertia and principal moments of inertia: (grams * square millimeters)

Taken at the center of mass.

Ix = (0.19, 0.06, 0.98)	Px = 57629.60
Iy = (0.98, 0.09, -0.19)	Py = 128340.80
Iz = (-0.10, 0.99, -0.04)	Pz = 177028.84

Moments of inertia: (grams * square millimeters)

Taken at the center of mass and aligned with the output coordinate system.

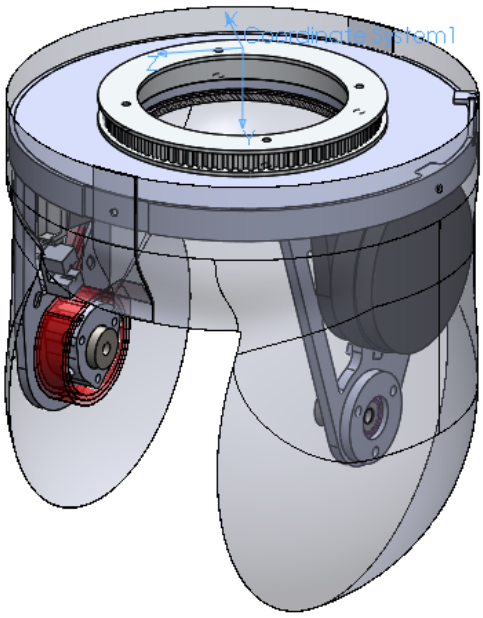
Lxx = 126293.64	Lxy = 5676.67	Lxz = 12958.68
Lyx = 5676.67	Lyy = 176218.54	Lyz = 5903.24
Lzx = 12958.68	Lzy = 5903.24	Lzz = 60487.07

Moments of inertia: (grams * square millimeters)

Taken at the output coordinate system.

Ixx = 144730.71	Ixy = -2075.07	Ixz = 25801.64
Iyx = -2075.07	Iyy = 201937.76	Iyz = -2253.45
Izx = 25801.64	Izy = -2253.45	Izz = 77615.62

Platforma mobila a camerei



Mass = 146.12 grams

Volume = 58618.13 cubic millimeters

Surface area = 106660.57 square millimeters

Center of mass: (millimeters)

X = -7.07

Y = 39.99

Z = -4.56

Principal axes of inertia and principal moments of inertia: (grams * square millimeters)

Taken at the center of mass.

Ix = (-0.42, -0.61, -0.67)	Px = 119317.52
Iy = (-0.51, 0.77, -0.39)	Py = 225661.16
Iz = (0.75, 0.17, -0.64)	Pz = 262147.05

Moments of inertia: (grams * square millimeters)

Taken at the center of mass and aligned with the output coordinate system.

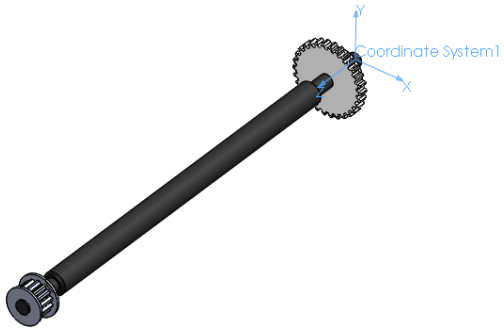
Lxx = 227145.83	Lxy = 22955.26	Lxz = 47498.04
Lyx = 22955.26	Lyy = 186694.73	Lyz = 47462.53
Lzx = 47498.04	Lzy = 47462.53	Lzz = 193285.18

Moments of inertia: (grams * square millimeters)

Taken at the output coordinate system.

Ixx = 463809.58	Ixy = -18333.92	Ixz = 52203.32
Iyx = -18333.92	Iyy = 197025.81	Iyz = 20838.29
Izx = 52203.32	Izy = 20838.29	Izz = 434211.85

Surub escamotare



Mass = 12.94 grams

Volume = 2407.47 cubic millimeters

Surface area = 2512.79 square millimeters

Center of mass: (millimeters)

X = 0.00

Y = 0.00

Z = 51.45

Principal axes of inertia and principal moments of inertia: (grams * square millimeters)

Taken at the center of mass.

Ix = (-0.00, 0.00, 1.00)	Px = 49.51
Iy = (-0.00, -1.00, 0.00)	Py = 9756.79
Iz = (1.00, -0.00, 0.00)	Pz = 9756.81

Moments of inertia: (grams * square millimeters)

Taken at the center of mass and aligned with the output coordinate system.

Lxx = 9756.81	Lxy = 0.00	Lxz = -0.96
Lyx = 0.00	Lyy = 9756.79	Lyz = 0.00
Lzx = -0.96	Lzy = 0.00	Lzz = 49.51

Moments of inertia: (grams * square millimeters)

Taken at the output coordinate system.

Ixx = 44022.13	Ixy = 0.00	Ixz = 0.06
Iyx = 0.00	Iyy = 44022.11	Iyz = -0.00
Izx = 0.06	Izy = -0.00	Izz = 49.51

Pasul surubului de escamotare este de 0,8 mm. Roata dințată de la capatul surubului are 30 de dinți și angrenează cu o roată cu același număr de dinți, raporul de transmitere fiind 1/1.

Camera cu sistemul de prindere

Mass = 424.49 grams

Volume = 330585.19 cubic millimeters

Surface area = 138022.51 square millimeters

Center of mass: (millimeters)

X = 6.94

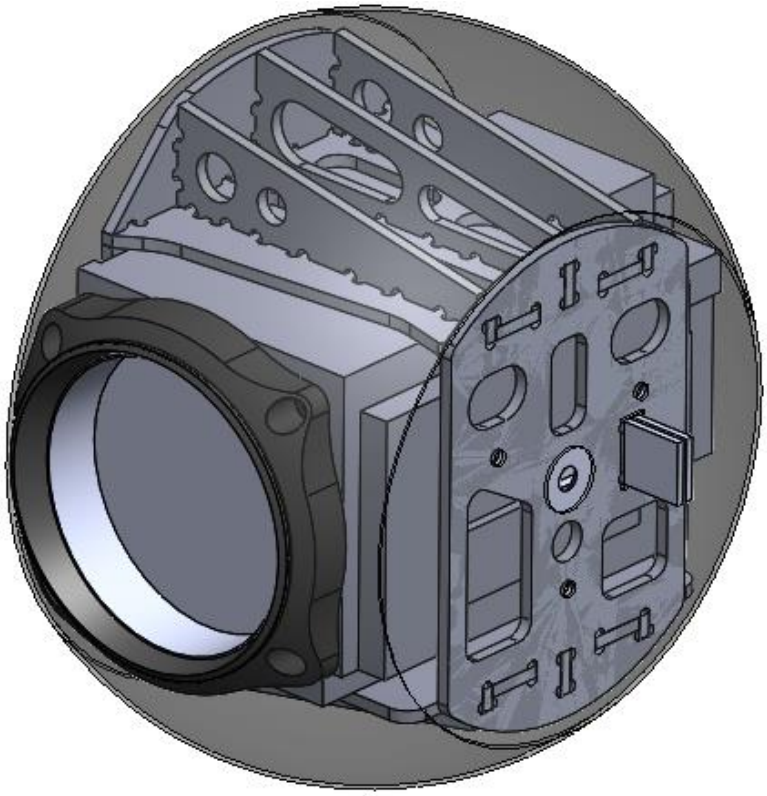
Y = -52.19

Z = 9.58

Principal axes of inertia and principal moments of inertia: (grams * square millimeters)

Taken at the center of mass.

Ix = (0.19, -0.25, 0.95)	Px = 235778.04
Iy = (0.56, -0.77, -0.32)	Py = 319523.00
Iz = (0.81, 0.59, -0.00)	Pz = 332186.11



Moments of inertia: (grams * square millimeters)

Taken at the center of mass and aligned with the output coordinate system.

Lxx = 324876.99	Lxy = -10027.16	Lxz = 14967.25
Lyx = -10027.16	Lyy = 318428.54	Lyz = -20226.98
Lzx = 14967.25	Lzy = -20226.98	Lzz = 244181.62

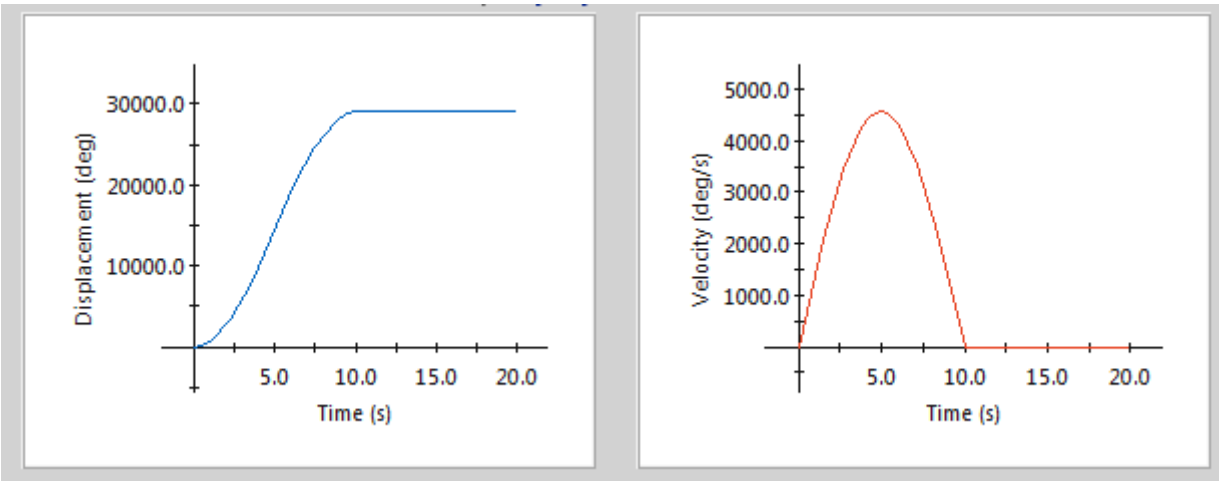
Moments of inertia: (grams * square millimeters)

Taken at the output coordinate system.

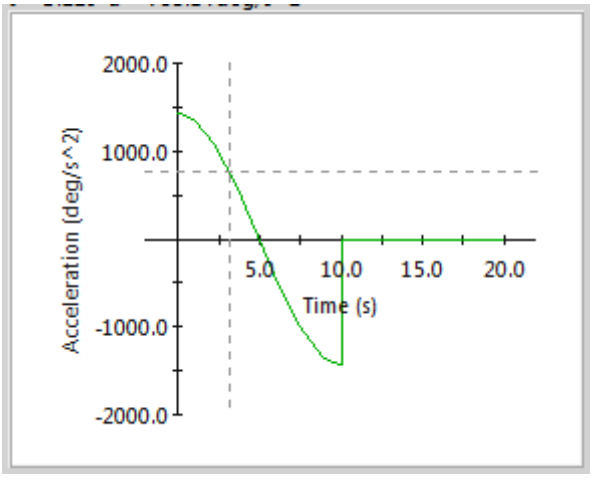
Ixx = 1519930.89	Ixy = -163753.71	Ixz = 43173.07
Iyx = -163753.71	Iyy = 377790.25	Iyz = -232354.66
Izx = 43173.07	Izy = -232354.66	Izz = 1420754.69

2.1 Motorul de escamotare:

Legile de variatie ale motorului de actionare a mecanismului de escamotare

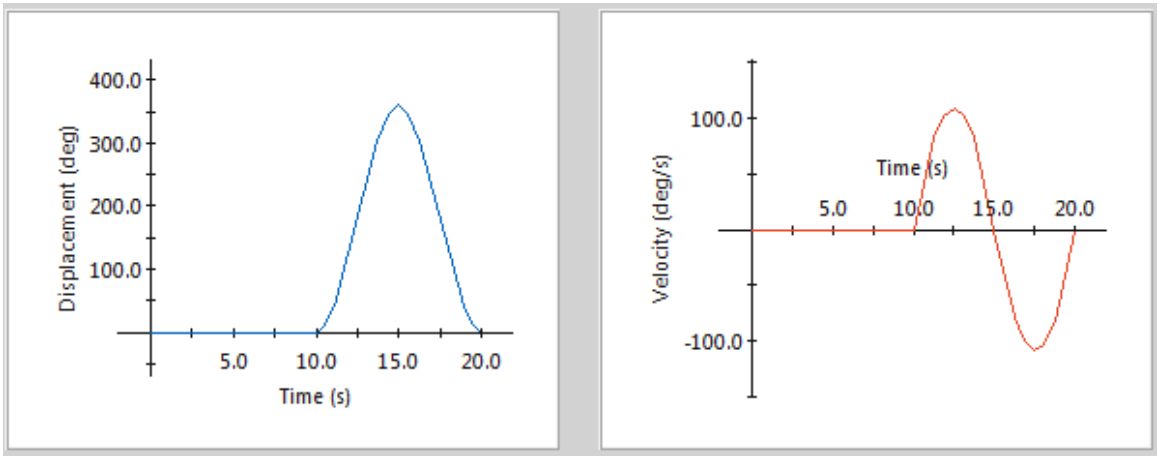


Deplasarea unghiulara Viteza unghiulara

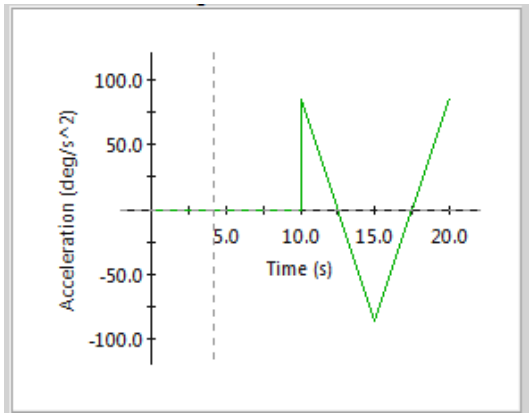


Acceleratia unghiulara

2.2 Motor pentru rotatia platformei camerei

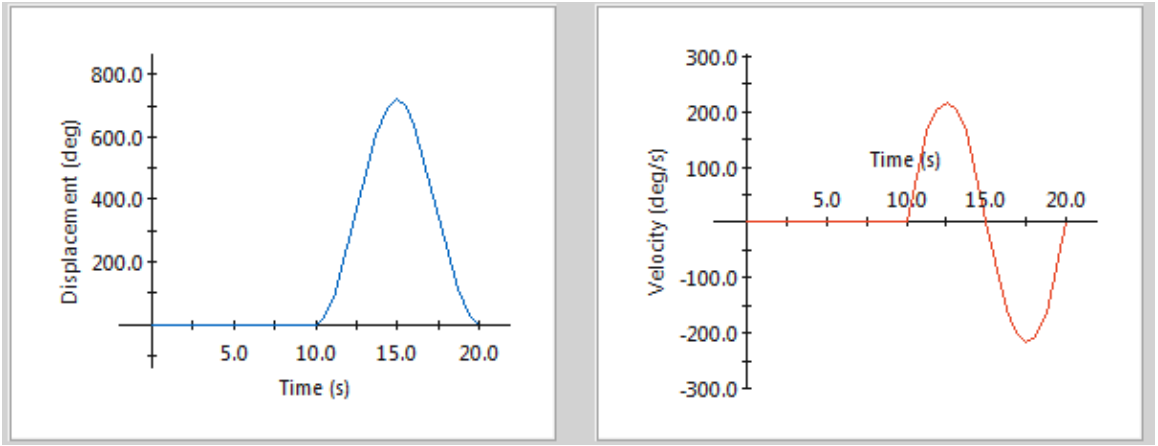


Deplasarea unghiulara Viteza unghiulara

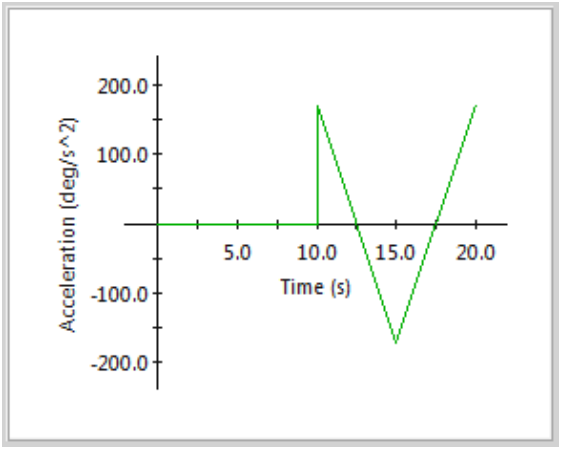


Acceleratia unghiulara

2.3 Motor pentru rotatia camerei



Deplasarea unghiulara Viteza unghiulara



Acceleratia unghiulara

3. SOFTWARE[1]

https://github.com/HKUST-Aerial-Robotics/AutoTrans/blob/main/Utils/odom_visualization/CMakeLists.txt

```
#cmake_minimum_required(VERSION 2.4.6)
#include($ENV{ROS_ROOT}/core/rosbuild/rosbuild.cmake)

# Set the build type.  Options are:
#  Coverage       : w/ debug symbols, w/o optimization, w/ code-coverage
#  Debug          : w/ debug symbols, w/o optimization
#  Release        : w/o debug symbols, w/ optimization
#  RelWithDebInfo : w/ debug symbols, w/ optimization
#  MinSizeRel     : w/o debug symbols, w/ optimization, stripped binaries
#set(ROS_BUILD_TYPE RelWithDebInfo)

#rosbuild_init()

#set the default path for built executables to the "bin" directory
#set(EXECUTABLE_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/bin)
#set the default path for built libraries to the "lib" directory
#set(LIBRARY_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/lib)

#uncomment if you have defined messages
#rosbuild_genmsg()
#uncomment if you have defined services
#rosbuild_gensrv()

#common commands for building c++ executables and libraries
#rosbuild_add_library(${PROJECT_NAME} src/example.cpp)
#target_link_libraries(${PROJECT_NAME} another_library)
#rosbuild_add_boost_directories()
#rosbuild_link_boost(${PROJECT_NAME} thread)
```

[1] https://github.com/HKUST-Aerial-Robotics/AutoTrans

```
#rosbuild_add_executable(example examples/example.cpp)
#target_link_libraries(example ${PROJECT_NAME})

#rosbuild_add_executable(odom_visualization src/odom_visualization.cpp)
#target_link_libraries(odom_visualization pose_utils)

#---------------------------------
cmake_minimum_required(VERSION 2.8.3)
project(odom_visualization)

set(CMAKE_BUILD_TYPE "Release")
set(CMAKE_CXX_FLAGS "-O3 -Wall -g")
ADD_COMPILE_OPTIONS(-std=c++11 )
ADD_COMPILE_OPTIONS(-std=c++14 )

## Find catkin macros and libraries
## if COMPONENTS list like find_package(catkin REQUIRED COMPONENTS xyz)
## is used, also find other catkin packages
find_package(catkin REQUIRED COMPONENTS
  roscpp
  sensor_msgs
  nav_msgs
  visualization_msgs
  quadrotor_msgs
  tf
  pose_utils
)
find_package(Eigen3 REQUIRED)
## System dependencies are found with CMake's conventions
# find_package(Boost REQUIRED COMPONENTS system)



## Uncomment this if the package has a setup.py. This macro ensures
## modules and global scripts declared therein get installed
## See http://ros.org/doc/api/catkin/html/user_guide/setup_dot_py.html
```

```
# catkin_python_setup()

################################################
## Declare ROS messages, services and actions ##
################################################

## To declare and build messages, services or actions from within this
## package, follow these steps:
## * Let MSG_DEP_SET be the set of packages whose message types you use in
##   your messages/services/actions (e.g. std_msgs, actionlib_msgs, ...).
## * In the file package.xml:
##   * add a build_depend and a run_depend tag for each package in MSG_DEP_SET
##   * If MSG_DEP_SET isn't empty the following dependencies might have been
##     pulled in transitively but can be declared for certainty nonetheless:
##     * add a build_depend tag for "message_generation"
##     * add a run_depend tag for "message_runtime"
## * In this file (CMakeLists.txt):
##   * add "message_generation" and every package in MSG_DEP_SET to
##     find_package(catkin REQUIRED COMPONENTS ...)
##   * add "message_runtime" and every package in MSG_DEP_SET to
##     catkin_package(CATKIN_DEPENDS ...)
##   * uncomment the add_*_files sections below as needed
##     and list every .msg/.srv/.action file to be processed
##   * uncomment the generate_messages entry below
##   * add every package in MSG_DEP_SET to generate_messages(DEPENDENCIES ...)

## Generate messages in the 'msg' folder
#add_message_files(
#   FILES
#   MultiOccupancyGrid.msg
#   MultiSparseMap3D.msg
#   SparseMap3D.msg
#   VerticalOccupancyGridList.msg
  #plan_cmd.msg
#)
```

```
## Generate services in the 'srv' folder
# add_service_files(
#   FILES
#   Service1.srv
#   Service2.srv
# )

## Generate actions in the 'action' folder
# add_action_files(
#   FILES
#   Action1.action
#   Action2.action
# )

## Generate added messages and services with any dependencies listed here
#generate_messages(
#   DEPENDENCIES
#   geometry_msgs
#   nav_msgs
#)

####################################
## catkin specific configuration ##
####################################
## The catkin_package macro generates cmake config files for your package
## Declare things to be passed to dependent projects
## INCLUDE_DIRS: uncomment this if you package contains header files
## LIBRARIES: libraries you create in this project that dependent projects also need
## CATKIN_DEPENDS: catkin_packages dependent projects also need
## DEPENDS: system dependencies of this project that dependent projects also need
catkin_package(
#  INCLUDE_DIRS include
#  LIBRARIES irobot_msgs
#  CATKIN_DEPENDS geometry_msgs nav_msgs
```

```
#  DEPENDS system_lib
)

###########
## Build ##
###########
find_package(Armadillo REQUIRED)
include_directories(${ARMADILLO_INCLUDE_DIRS})



## Specify additional locations of header files
## Your package locations should be listed before other locations
include_directories(include)
include_directories(
  ${catkin_INCLUDE_DIRS}
  ${EIGEN3_INCLUDE_DIR}
)

## Declare a cpp library
# add_library(irobot_msgs
#   src/${PROJECT_NAME}/irobot_msgs.cpp
# )

## Declare a cpp executable
add_executable(odom_visualization src/odom_visualization.cpp src/CameraPoseVisualization.cpp)

## Add cmake target dependencies of the executable/library
## as an example, message headers may need to be generated before nodes
# add_dependencies(multi_map_visualization multi_map_server_messages_cpp)

## Specify libraries to link a library or executable target against
target_link_libraries(odom_visualization
   ${catkin_LIBRARIES}
   ${ARMADILLO_LIBRARIES}
   pose_utils
```

```
)

#############
## Install ##
#############

# all install targets should use catkin DESTINATION variables
# See http://ros.org/doc/api/catkin/html/adv_user_guide/variables.html

## Mark executable scripts (Python etc.) for installation
## in contrast to setup.py, you can choose the destination
# install(PROGRAMS
#   scripts/my_python_script
#   DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
# )

## Mark executables and/or libraries for installation
# install(TARGETS irobot_msgs irobot_msgs_node
#   ARCHIVE DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
#   LIBRARY DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
#   RUNTIME DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
# )

## Mark cpp header files for installation
# install(DIRECTORY include/${PROJECT_NAME}/
#   DESTINATION ${CATKIN_PACKAGE_INCLUDE_DESTINATION}
#   FILES_MATCHING PATTERN "*.h"
#   PATTERN ".svn" EXCLUDE
# )

## Mark other files for installation (e.g. launch and bag files, etc.)
# install(FILES
#   # myfile1
#   # myfile2
#   DESTINATION ${CATKIN_PACKAGE_SHARE_DESTINATION}
```

```
# )

#############
## Testing ##
#############

## Add gtest based cpp test target and link libraries
# catkin_add_gtest(${PROJECT_NAME}-test test/test_irobot_msgs.cpp)
# if(TARGET ${PROJECT_NAME}-test)
#   target_link_libraries(${PROJECT_NAME}-test ${PROJECT_NAME})
# endif()
```

==========

https://github.com/HKUST-Aerial-Robotics/AutoTrans/blob/main/Utils/odom_visualization/src/CameraPoseVisualization.cpp

```
/*******************************************************
 * Copyright (C) 2019, Aerial Robotics Group, Hong Kong University of Science and Technology
 *
 * This file is part of VINS.
 *
 * Licensed under the GNU General Public License v3.0;
 * you may not use this file except in compliance with the License.
 *******************************************************/

#include "CameraPoseVisualization.h"

const Eigen::Vector3d CameraPoseVisualization::imlt = Eigen::Vector3d(-1.0, -0.5, 1.0);
const Eigen::Vector3d CameraPoseVisualization::imrt = Eigen::Vector3d( 1.0, -0.5, 1.0);
const Eigen::Vector3d CameraPoseVisualization::imlb = Eigen::Vector3d(-1.0,  0.5, 1.0);
const Eigen::Vector3d CameraPoseVisualization::imrb = Eigen::Vector3d( 1.0,  0.5, 1.0);
const Eigen::Vector3d CameraPoseVisualization::lt0 = Eigen::Vector3d(-0.7, -0.5, 1.0);
const Eigen::Vector3d CameraPoseVisualization::lt1 = Eigen::Vector3d(-0.7, -0.2, 1.0);
const Eigen::Vector3d CameraPoseVisualization::lt2 = Eigen::Vector3d(-1.0, -0.2, 1.0);
const Eigen::Vector3d CameraPoseVisualization::oc = Eigen::Vector3d(0.0, 0.0, 0.0);
```

```
void Eigen2Point(const Eigen::Vector3d& v, geometry_msgs::Point& p) {
    p.x = v.x();
    p.y = v.y();
    p.z = v.z();
}

CameraPoseVisualization::CameraPoseVisualization(float r, float g, float b, float a)
    : m_marker_ns("CameraPoseVisualization"), m_scale(0.2), m_line_width(0.01) {
    m_image_boundary_color.r = r;
    m_image_boundary_color.g = g;
    m_image_boundary_color.b = b;
    m_image_boundary_color.a = a;
    m_optical_center_connector_color.r = r;
    m_optical_center_connector_color.g = g;
    m_optical_center_connector_color.b = b;
    m_optical_center_connector_color.a = a;
}

void CameraPoseVisualization::setImageBoundaryColor(float r, float g, float b, float a) {
    m_image_boundary_color.r = r;
    m_image_boundary_color.g = g;
    m_image_boundary_color.b = b;
    m_image_boundary_color.a = a;
}

void CameraPoseVisualization::setOpticalCenterConnectorColor(float r, float g, float b, float a) {
    m_optical_center_connector_color.r = r;
    m_optical_center_connector_color.g = g;
    m_optical_center_connector_color.b = b;
    m_optical_center_connector_color.a = a;
}

void CameraPoseVisualization::setScale(double s) {
    m_scale = s;
```

```
}
void CameraPoseVisualization::setLineWidth(double width) {
    m_line_width = width;
}
void CameraPoseVisualization::add_edge(const Eigen::Vector3d& p0, const Eigen::Vector3d& p1){
    visualization_msgs::Marker marker;

    marker.ns = m_marker_ns;
    marker.id = m_markers.size() + 1;
    marker.type = visualization_msgs::Marker::LINE_LIST;
    marker.action = visualization_msgs::Marker::ADD;
    marker.scale.x = 0.005;

    marker.color.g = 1.0f;
    marker.color.a = 1.0;

    geometry_msgs::Point point0, point1;

    Eigen2Point(p0, point0);
    Eigen2Point(p1, point1);

    marker.points.push_back(point0);
    marker.points.push_back(point1);

    m_markers.push_back(marker);
}

void CameraPoseVisualization::add_loopedge(const Eigen::Vector3d& p0, const Eigen::Vector3d&
p1){
    visualization_msgs::Marker marker;

    marker.ns = m_marker_ns;
    marker.id = m_markers.size() + 1;
    marker.type = visualization_msgs::Marker::LINE_LIST;
    marker.action = visualization_msgs::Marker::ADD;
```

```
    marker.scale.x = 0.04;
    //marker.scale.x = 0.3;

    marker.color.r = 1.0f;
    marker.color.b = 1.0f;
    marker.color.a = 1.0;

    geometry_msgs::Point point0, point1;

    Eigen2Point(p0, point0);
    Eigen2Point(p1, point1);

    marker.points.push_back(point0);
    marker.points.push_back(point1);

    m_markers.push_back(marker);
}


void CameraPoseVisualization::add_pose(const Eigen::Vector3d& p, const Eigen::Quaterniond& q)
{
    visualization_msgs::Marker marker;

    marker.ns = m_marker_ns;
    marker.id = m_markers.size() + 1;
    marker.type = visualization_msgs::Marker::LINE_STRIP;
    marker.action = visualization_msgs::Marker::ADD;
    marker.scale.x = m_line_width;

    marker.pose.position.x = 0.0;
    marker.pose.position.y = 0.0;
    marker.pose.position.z = 0.0;
    marker.pose.orientation.w = 1.0;
    marker.pose.orientation.x = 0.0;
    marker.pose.orientation.y = 0.0;
```

```
marker.pose.orientation.z = 0.0;


geometry_msgs::Point pt_lt, pt_lb, pt_rt, pt_rb, pt_oc, pt_lt0, pt_lt1, pt_lt2;

Eigen2Point(q * (m_scale *imlt) + p, pt_lt);
Eigen2Point(q * (m_scale *imlb) + p, pt_lb);
Eigen2Point(q * (m_scale *imrt) + p, pt_rt);
Eigen2Point(q * (m_scale *imrb) + p, pt_rb);
Eigen2Point(q * (m_scale *lt0 ) + p, pt_lt0);
Eigen2Point(q * (m_scale *lt1 ) + p, pt_lt1);
Eigen2Point(q * (m_scale *lt2 ) + p, pt_lt2);
Eigen2Point(q * (m_scale *oc  ) + p, pt_oc);

// image boundaries
marker.points.push_back(pt_lt);
marker.points.push_back(pt_lb);
marker.colors.push_back(m_image_boundary_color);
marker.colors.push_back(m_image_boundary_color);

marker.points.push_back(pt_lb);
marker.points.push_back(pt_rb);
marker.colors.push_back(m_image_boundary_color);
marker.colors.push_back(m_image_boundary_color);

marker.points.push_back(pt_rb);
marker.points.push_back(pt_rt);
marker.colors.push_back(m_image_boundary_color);
marker.colors.push_back(m_image_boundary_color);

marker.points.push_back(pt_rt);
marker.points.push_back(pt_lt);
marker.colors.push_back(m_image_boundary_color);
marker.colors.push_back(m_image_boundary_color);
```

```
    // top-left indicator
    marker.points.push_back(pt_lt0);
    marker.points.push_back(pt_lt1);
    marker.colors.push_back(m_image_boundary_color);
    marker.colors.push_back(m_image_boundary_color);

    marker.points.push_back(pt_lt1);
    marker.points.push_back(pt_lt2);
    marker.colors.push_back(m_image_boundary_color);
    marker.colors.push_back(m_image_boundary_color);

    // optical center connector
    marker.points.push_back(pt_lt);
    marker.points.push_back(pt_oc);
    marker.colors.push_back(m_optical_center_connector_color);
    marker.colors.push_back(m_optical_center_connector_color);



    marker.points.push_back(pt_lb);
    marker.points.push_back(pt_oc);
    marker.colors.push_back(m_optical_center_connector_color);
    marker.colors.push_back(m_optical_center_connector_color);

    marker.points.push_back(pt_rt);
    marker.points.push_back(pt_oc);
    marker.colors.push_back(m_optical_center_connector_color);
    marker.colors.push_back(m_optical_center_connector_color);

    marker.points.push_back(pt_rb);
    marker.points.push_back(pt_oc);
    marker.colors.push_back(m_optical_center_connector_color);
    marker.colors.push_back(m_optical_center_connector_color);

    m_markers.push_back(marker);
}
```

```
void CameraPoseVisualization::reset() {
	m_markers.clear();
}

void CameraPoseVisualization::publish_by( ros::Publisher &pub, const std_msgs::Header &header )
{
	visualization_msgs::MarkerArray markerArray_msg;

	for(auto& marker : m_markers) {
		marker.header = header;
		markerArray_msg.markers.push_back(marker);
	}

	pub.publish(markerArray_msg);
}
```

4. BIBLIOGRAFIE

1. Feng, L., Borenstein, J. and Everett, H.R. Sensors and Methods for Autonomous Mobile Robot Positioning. [ed.] Edited and compiled by J. Borenstein. Technical Report UM-MEAM-94-21. University of Michigan for the Oak Ridge Nat, December 1994, UM-MEAM-94-21.
2. Fields, M.A. Modifying ModSAF Terrain Databases to Support the Evaluation of Small Weapons Platforms in Tactical Scenarios,. Army Research Laboratory. ARL-TR-1996, AUGUST 1999.
3. A Simulation Model For The Prediction Of The Ground Pressure Distribution Under Tracked Vehicles. Gigler, J.K. and Ward, S.M. no.6, Journal of Theramechanics, Vol. vol.30.
4. Gordon, McComb. The Robot Builder's Bonanza, Second Edition. New York : McGraw-Hill, 2001.
5. Grigore, L.Ș. "Principii privind funcţionarea traductoarelor necesare cercetării experimentale a proceselor de injecţie". Bucureşti : Academia Tehnică Militară, 1997. Referat nr.3.
6. Handraluca, V. and ș.a. Roboţi. Cluj-Napoca : Dacia, 1996.
7. Mobility Analisys of Small, Lightweight Robotic Vehicles. Haueisen, B. April 25, 2003. 6807 9990.
8. Helmick, Daniel M. and ș.a. Multi-Sensor, High Speed Autonomous Stair Climbing. NASA - Jet Propulsion Laboratory, NASA. California Institute of Technology, Pasadena, CA : NASA.
9. Iorga, V., Jora, B. and ș.a. Programare numerică. București : Tora, 1996.
10. LAUGHERY, S., GERHART, G. and MUENCH, P. Evaluating Vehicle Mobility Using Bekker's Equations. Warren : US Army TARDEC. MI 48397-5000.
11. Morel, Yannick. Applied Nonlinear Control of Unmanned Vehicles with Uncertain Dynamics. Faculty of the Mechanical Engineering, Virginia Polytechnic Institute and State University. Blacksburg : s.n., April 17th, 2009. p. 226, Dissertation submitted in partial ful¯llment of the requirements for the degree of Doctor of Philosophy.
12. Roland, I. R. and Siegwart, N. Introduction to Autonomous Mobile Robots, A Bradford Book,. Massachusetts London, England : The MIT Press Cambridge,.
13. Sandin, E. P. Robot mechanisms and mechanical devices illustrated. New-York : McGraw-Hill, 2003.
14. AUVs: In Space, Air, Water, and on the Ground. Schoenwald, David A. [ed.] The author (daschoe@sandia.gov) is with Sandia National Laboratories. Albuquerque, NM 87185, U.S.A., P.O. Box 5800, MS 1004, : IEEE Control Systems Magazine, December 2000. Autonomous Unnmed Vehicles. 0272-1708/00/$10.00©2000IEEE.

15. Overview of cold regions mobility modeling at CRREL,. Shoop, S.A., Richmond, P.W. and Lacombe, J. Hanover, USA : Cold Regions Research and Engineering Laboratory,, US Army Engineering Research and Development Center.
16. Vestut, J. and Coiffet, Ph. Les robots. Tom 3A. Teleoperation evolution des tehnologies. France, Hermes Publishing. 1984.