nymeria_ardrone

By Project Nymeria

Wed Jan 21 2015 10:36:53

Contents

Chapter 1

Main Page

nymeria_ardrone is a ROS package for Parrot AR-Drone quadrocopter. It acts as a layer and filters drone commands sent from an external controller. It helps the drone determine if movement orders are safe or not depending on the trajectory of an obstacle and, if so, to move accordingly. In practice it contains three main modules. The first, linked to sensors, allows the drone to detect an obstacle. The second gets drone commands and the last makes the link between them. User defines radius of an obstacle and drone is controlled by Nymeria to slow down and stop in front of it. The driver supports AR-Drone 2.0.

Table of Contents

Requirements

- *ROS*: Robot Operating System
- *ardrone_autonomy*: Driver for Ardrone 1.0 & 2.0
- *Sensor*: any kind of tool enabling to retrieve range between drone and front obstacles

Installation

The first step is to install ROS following the (Robot Operating System installation tutorial). We have successfully tested two versions : hydro and indigo.

Then create a ROS workspace.

In order to communicate with the drone you will need to download ardrone_autonomy which provide the ardrone_driver. Follow the instruction in the installation section.

Navigate to your catkin_workspace sources repository.

```
$ cd ~/catkin_ws/src
```

Download the nymeria_ardrone package using the following command in a terminal.

```
$ git clone https://github.com/ProjectNymeria/nymeria_ardrone
```

You might prefer to reach nymeria_ardrone webpage and download and unpack the nymeria_ardrone package.

Go back to your root workspace repository.

```
$ cd ~/catkin_ws
```

Use the catkin_make command to compile

```
$ catkin_make
```

How to run it

First switch on Wifi on your computer and connect it to your Ardrone 2.0.

You must launch the master node. Navigate to your catkin_workspace (`$ cd ~/catkin_ws`) and type the following command :

```
$ roscore
```

Then launch the ardrone_autonomy driver's executable node. You can use :

```
$ rosrun ardrone_autonomy ardrone_driver
```

Or put it in a custom launch file with your desired parameters.

Navigate to ~/catkin_ws/src/nymeria_ardrone/src/SensorInterface.cpp and find the line *nco.inputCurFrontDist(cut↩ Value);* Replace the 'cutValue' variable by the current distance of the front sensor of your drone. Once done, run the sensor_interface node :

```
$ rosrun nymeria_ardrone nymeria_sensor_interface
```

By default the security distance is 100 cm. To change it just call the setSecurityDist(double secDist) from the class NymeriaCheckObstacle.

```
double getSecurityDist();
void setSecurityDist(double secDist);
```

By default the sensor max range is 350 cm. To change it just call the setSensorMaxRange(double range) from the class NymeriaCheckObstacle.

```
double getSensorMaxRange();
void setSensorMaxRange(double range);
```

Launch the nymeria_command executable node using :

```
$ rosrun nymeria_ardrone nymeria_command
```

This node is the interface between you as a user who wish to send orders and the drone. Command are sent from keystroke detailled below.

- *ENTER* : LAND / TAKE OFF
- *Z* : move forward
- *S* : move backward

- *Q* : rotate left
- *D* : rotate right
- *UP* : move up
- *DOWN* : move down
- *i* : move down
- *k* : move down
- *o* : move down
- *l* : move down
- *p* : move down
- *m* : move down
- *SPACE* : stop

The last step consists to run the launch the Controller node

```
$ rosrun nymeria_ardrone controller
```

You are ready to go. Just stroke the appropriate key from the nymeria_command interface. Your drone will naturally keep the inputed security distance between any front obstacle and itself.

How does it work

As explain in the introduction above, nymeria_ardrone uses various nodes.

- *roscore* is the implicit one. It is a ROS requirement allowing nodes to communicate with each others.
- *ardrone_driver* allows communicating with the drone.
- *nymeria_sensor_interface* runs constantly, and obtains data from any kind of tool that enables you to retrieve the distance between the drone and obstacles in front.

  It also gets the pitch which is the degree of inclination of the drone and represents its speed.

  This component then provides a regulated speed factor for the drone.
- *nymeria_command* is the interface with the user. From there, he sends navigation commands to the drone.
- *nymeria_controller* takes into account all the parameters and determines whether or not, the user's command is safe for the drone in terms of obstacle& detection.

  Just as the sensor interface, it runs constantly since dangers can arise anytime.

  The first action of the controller is to check the user's command :

  If the command includes a rotating move, or the modification of configuration parameters such as the speed then it processes it immediately.

  If the command represents a linear move, it might not be safe because, in that case, the drone can run into an obstacle. So there are two choices : either the security distance provided by the user has already been violated, or we can still anticipate the obstacle.

  In this last case we apply a slow down algorithm which adapts the speed factor periodically based on the current distance from an obstacle in front : if the obstacle is far from the drone, the speed factor is highest. If the obstacle is close to the drone the speed factor is lowest.

  This allows for smoothly stopping at the desired security distance.

  If the security distance has already been violated then nymeria quickly reacts by stopping the drone and moving backwards in order to keep the security distance.

The latter behavior conducts the drone in every critical situation : it must keep a security distance. That means that if the drone is running an unsafe action such as moving forward, and an obstacle is approaching it, then the drone must react and move backwards in order to keep the security distance. It also means that, if the user inputs a new command while the security distance is not keeped, the command will be ignored until the security distance is reestablished.

Nymeria is a library providing navigation commands which can be for instance accessed by the controller, the user and so on.

The graph below summarizes the behavior of the *nymeria_ardrone* package.

Some examples

In this package you will find the *sensor.ino* file in the *arduino/* repository. In order to test our librairy we used an arduino nano coupled with an ultrasonic sensor and linked by the USB port to the drone. The simple code in *sensor.ino* enables us to retrieve the distance between the drone and obstacles in front. Belows is a part of it.

```
[...]

// convert the time into a distance
cm = microsecondsToCentimeters(duration);

// cast distance into a string
String dist = String(cm);

// send distance
bytesSent = Serial.print(dist);

// send separator
Serial.print('x');

[...]
```

You will also find the *embedded/* repository. Still for test purposes we needed to embed a server on the drone so that the sensor_interface, by simply making request to it, can retrieve the distance from the sensor. The *DroneU↩ DPServer.elf* is the compiled version of the *DroneUDPServer.cpp*. If you want to run your own version of the server please refer to the Cross compilation tutorial we followed. It first connects to the USB serial then send the distance read.

```
[...]

/* Open serial port
try to open ttyUSB0 first. If failed, try to open ttyUSB1
*/
fd = open("/dev/ttyUSB0", O_RDONLY | O_NOCTTY | O_NDELAY);

[...]

// Open UDP server //
UDPServer server("192.168.1.1", 7777);

[...]

bsent = server.send(sendBuffer, 4);

[...]
```

By default the UDP port is 7777. If you would like to change it, do not forget to make the update in the Nymeria↩ CheckObstacle too.

On the other

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Class Documentation

5.1 Nymeria Class Reference

Definitions of the class Nymeria, that declares all functionalities in order to allow for drone navigation with obstacle detection and avoidance.

`#include <Nymeria.h>`

Public Member Functions

- Nymeria ()

  Default empty constructor.
- Nymeria (ros::NodeHandle *n)

  Constructor in order to create a meaningful object of the type Nymeria.
- void moveForward ()

  Command in order to move drone forward.
- void moveBackward ()

  Command in order to move drone backward.
- void moveLeft ()

  Command in order to make drone rotate to the left.
- void moveRight ()

  Command in order to make drone rotate to the right.
- void moveUp ()

  Command in order to move drone upward, i.e.
- void moveDown ()

  Command in order to move drone downward, i.e.
- void turnLeft ()

  Command in order to move drone to the left.
- void turnRight ()

  Command in order to move drone to the right.
- void stop ()

  Command in order to stop the drone's movement, i.e.
- void takeOff ()

  Command in order to make the drone take off.
- void land ()

  Command in order to make the drone land, i.e.
- void emergencyStop ()

  Command in order to make drone stop and immediately land.

- void increaseMaxLinearSpeed ()

  Command in order to increase the maximum linear speed by 10%.
- void decreaseMaxLinearSpeed ()

  Command in order to decrease the maximum linear speed by 10%.
- void increaseMaxAngularSpeed ()

  Command in order to increase the maximum angular speed by 10%.
- void decreaseMaxAngularSpeed ()

  Command in order to decrease the maximum angular speed by 10%.
- void increaseLinearSpeed ()

  Command in order to increase the linear speed by 10%.
- void decreaseLinearSpeed ()

  Command in order to decrease the linear speed by 10%.
- void increaseAngularSpeed ()

  Command in order to increase the angular speed by 10%.
- void decreaseAngularSpeed ()

  Command in order to decrease the angular speed by 10%.
- double getSecurityDist ()

  Getter function for security distance, in order to permit the user to retain its current value.
- void setSecurityDist (double secDist)

  Setter function for security distance, in order to permit the user to change its value.
- double getMaxLinearSpeed ()

  Getter function for maximum linear speed, in order to permit the user to retain its current value.
- void setMaxLinearSpeed (double speed)

  Setter function for maximum linear speed, in order to permit the user to change its value.
- double getLinearSpeed ()

  Getter function for current linear speed.
- void setLinearSpeed (double speed)

  Setter function for current linear speed, in order to permit the user to change its value.
- double getMaxAngularSpeed ()

  Getter function for maximum angular speed, in order to permit the user to retain its current value.
- void setMaxAngularSpeed (double speed)

  Setter function for maximum angular speed, in order to permit the user to change its value.
- double getAngularSpeed ()

  Getter function for angular speed, in order to permit the user to retain its current value.

Private Member Functions

- void init_safeActions ()

  Helper function in order to initialize the array of safe actions.
- void init_rosParams ()

  Helper function in order to initialize ROS parameters nymeriaCommand, nymeriaStateObstacle, nymeriaSecurityDist.
- void init_move_msg ()

  Helper function in order to initialize move_msg.
- void init_publishers ()

  Helper function in order to initialize publishers.
- double getParameter (char *str)

  Helper function in order to access ROS parameters (read access).
- int validateStates ()

  Entry point of obstacle detection and avoidance.
- bool isSafeAction (int cmd)

Does the user's command belong to the list of safeActions, i.e.

- bool obstaclePossible ()

  Is it possible, that there is an obstacle in front?
- bool underSecurityDist ()

  Is there an obstacle in front closer than the given security distance?
- int triggerAction (int cmd, double factor=1.0)

  Forward command to drone.
- void reactionRoutine ()

  Routine in order to make drone stop in front of obstacle and keep the security distance.
- void keepSecurityDistance ()

  Method in order to keep security distance by moving backward if necessary.
- void slowDown ()

  Method in order to initiate slowing down.
- bool inRange (double min, double max, double value)

  Helper functions in order to determine, whether a given value is in a given interval.

Private Attributes

- ros::NodeHandle * nh
- ros::Publisher pub_cmd_takeoff
- ros::Publisher pub_cmd_land
- ros::Publisher pub_cmd_move
- ros::Publisher pub_cmd_reset
- ardrone_autonomy::Navdata navData
- ros::Subscriber sub_navdata
- std_msgs::Empty empty_msg
- geometry_msgs::Twist move_msg
- int lastCmd
- double maxLinearSpeed
- double maxAngularSpeed
- double linearSpeed
- double angularSpeed
- int safeActions [20]

Friends

- class Controller

5.1.1 Detailed Description

Definitions of the class Nymeria, that declares all functionalities in order to allow for drone navigation with obstacle detection and avoidance.

Author

Team-Nymeria

Version

0.2

Date

18th of January 2015

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Nymeria::Nymeria ( )

Default empty constructor.

5.1.2.2 Nymeria::Nymeria (ros::NodeHandle * *n*)

Constructor in order to create a meaningful object of the type Nymeria.

Meaningful in terms of functionality: It provides all navigation commands for the drone whilst ensuring obstacle protection and avoidance.

Parameters

n	NodeHandle permitting to relate ROS-node.

5.1.3 Member Function Documentation

5.1.3.1 void Nymeria::decreaseAngularSpeed ( )

Command in order to decrease the angular speed by 10%.

5.1.3.2 void Nymeria::decreaseLinearSpeed ( )

Command in order to decrease the linear speed by 10%.

5.1.3.3 void Nymeria::decreaseMaxAngularSpeed ( )

Command in order to decrease the maximum angular speed by 10%.

5.1.3.4 void Nymeria::decreaseMaxLinearSpeed ( )

Command in order to decrease the maximum linear speed by 10%.

5.1.3.5 void Nymeria::emergencyStop ( )

Command in order to make drone stop and immediately land.

5.1.3.6 double Nymeria::getAngularSpeed ( )

Getter function for angular speed, in order to permit the user to retain its current value.

Returns

angular speed

5.1.3.7 double Nymeria::getLinearSpeed ( )

Getter function for current linear speed.

Returns

current linear speed.

5.1.3.8 double Nymeria::getMaxAngularSpeed ( )

Getter function for maximum angular speed, in order to permit the user to retain its current value.

Returns

maximum angular speed

5.1.3.9 double Nymeria::getMaxLinearSpeed ( )

Getter function for maximum linear speed, in order to permit the user to retain its current value.

Returns

maximum linear speed.

5.1.3.10 double Nymeria::getParameter (char * *str*) [private]

Helper function in order to access ROS parameters (read access).

Parameters

str	- name of parameter.

Returns

read parameter value, -1 if no parameter is found.

5.1.3.11 double Nymeria::getSecurityDist ( )

Getter function for security distance, in order to permit the user to retain its current value.

Returns

security distance.

5.1.3.12 void Nymeria::increaseAngularSpeed ( )

Command in order to increase the angular speed by 10%.

5.1.3.13 void Nymeria::increaseLinearSpeed ( )

Command in order to increase the linear speed by 10%.

5.1.3.14 void Nymeria::increaseMaxAngularSpeed ( )

Command in order to increase the maximum angular speed by 10%.

5.1.3.15 void Nymeria::increaseMaxLinearSpeed ( )

Command in order to increase the maximum linear speed by 10%.

5.1.3.16 void Nymeria::init_move_msg ( ) [private]

Helper function in order to initialize move_msg.

5.1.3.17 void Nymeria::init_publishers ( ) [private]

Helper function in order to initialize publishers.

5.1.3.18 void Nymeria::init_rosParams ( ) [private]

Helper function in order to initialize ROS parameters nymeriaCommand, nymeriaStateObstacle, nymeriaSecurity↩ Dist.

5.1.3.19 void Nymeria::init_safeActions ( ) [private]

Helper function in order to initialize the array of safe actions.

5.1.3.20 bool Nymeria::inRange (double *min,* double *max,* double *value*) [private]

Helper functions in order to determine, whether a given value is in a given interval.

Parameters

min	left border of interval.
max	right border of interval.
value	value to be tested.

Returns

true: yes, value is in given interval. false: no, value is not in given interval.

5.1.3.21 bool Nymeria::isSafeAction (int *cmd*) [private]

Does the user's command belong to the list of safeActions, i.e.

can the command be safely forwarded to the drone?

Parameters

cmd	- incoming command.

Returns

true: yes, command can be forwarded. false: no, check for obstacles is necessary.

5.1.3.22 void Nymeria::keepSecurityDistance ( ) [private]

Method in order to keep security distance by moving backward if necessary.

5.1.3.23 void Nymeria::land ( )

Command in order to make the drone land, i.e.

underneath current position.

5.1.3.24 void Nymeria::moveBackward ( )

Command in order to move drone backward.

5.1.3.25 void Nymeria::moveDown ( )

Command in order to move drone downward, i.e.

decrease altitude.

5.1.3.26 void Nymeria::moveForward ( )

Command in order to move drone forward.

5.1.3.27 void Nymeria::moveLeft ( )

Command in order to make drone rotate to the left.

5.1.3.28 void Nymeria::moveRight ( )

Command in order to make drone rotate to the right.

5.1.3.29 void Nymeria::moveUp ( )

Command in order to move drone upward, i.e.

increase altitude.

5.1.3.30 bool Nymeria::obstaclePossible ( ) [private]

Is it possible, that there is an obstacle in front?

Returns

true: yes, obstacle anticipated. false: no, no obstacle to be likely in front.

5.1.3.31 void Nymeria::reactionRoutine ( ) [private]

Routine in order to make drone stop in front of obstacle and keep the security distance.

5.1.3.32 void Nymeria::setLinearSpeed (double *speed*)

Setter function for current linear speed, in order to permit the user to change its value.

Parameters

speed	- linear speed.

5.1.3.33 void Nymeria::setMaxAngularSpeed (double *speed*)

Setter function for maximum angular speed, in order to permit the user to change its value.

Parameters

speed	- maximum angular speed.

5.1.3.34 void Nymeria::setMaxLinearSpeed (double *speed*)

Setter function for maximum linear speed, in order to permit the user to change its value.

Parameters

speed	- maximum linear speed.

5.1.3.35 void Nymeria::setSecurityDist (double *secDist*)

Setter function for security distance, in order to permit the user to change its value.

Parameters

secDist	security distance.

5.1.3.36 void Nymeria::slowDown ( ) [private]

Method in order to initiate slowing down.

5.1.3.37 void Nymeria::stop ( )

Command in order to stop the drone's movement, i.e.

stay at current position.

5.1.3.38 void Nymeria::takeOff ( )

Command in order to make the drone take off.

5.1.3.39 int Nymeria::triggerAction (int *cmd,* double *factor* = 1.0) [private]

Forward command to drone.

Parameters

cmd	- incoming command.
factor	- regulating speed factor for slow down, 1 by default.

Returns

constant representing cmd processed.

5.1.3.40 void Nymeria::turnLeft ( )

Command in order to move drone to the left.

5.1.3.41 void Nymeria::turnRight ( )

Command in order to move drone to the right.

5.1.3.42 bool Nymeria::underSecurityDist ( ) [private]

Is there an obstacle in front closer than the given security distance?

Returns

true: yes, obstacle in front too close. false: no, security distance still kept.

5.1.3.43 int Nymeria::validateStates ( ) [private]

Entry point of obstacle detection and avoidance.

Algorithm analyzes sensor data in the form of distances and decides whether to (1) either stop the drone immediately and let it move backward if applicable (2) or let the drone slow down (3) or process the user's command without acting.

Returns

constant representing processed command or -1, when there has been an obstacle.

5.1.4 Friends And Related Function Documentation

5.1.4.1 friend class Controller [friend]

5.1.5 Member Data Documentation

5.1.5.1 double Nymeria::angularSpeed [private]

5.1.5.2 std_msgs::Empty Nymeria::empty_msg [private]

5.1.5.3 int Nymeria::lastCmd [private]

5.1.5.4 double Nymeria::linearSpeed [private]

5.1.5.5 double Nymeria::maxAngularSpeed [private]

5.1.5.6 double Nymeria::maxLinearSpeed [private]

5.1.5.7 geometry_msgs::Twist Nymeria::move_msg [private]

5.1.5.8 ardrone_autonomy::Navdata Nymeria::navData [private]

5.1.5.9 ros::NodeHandle* Nymeria::nh [private]

5.1.5.10 ros::Publisher Nymeria::pub_cmd_land [private]

5.1.5.11 ros::Publisher Nymeria::pub_cmd_move [private]

5.1.5.12 ros::Publisher Nymeria::pub_cmd_reset [private]

5.1.5.13 ros::Publisher Nymeria::pub_cmd_takeoff [private]

5.1.5.14 int Nymeria::safeActions[20] [private]

5.1.5.15 ros::Subscriber Nymeria::sub_navdata [private]

The documentation for this class was generated from the following files:

- nymeria_ardrone/include/nymeria_ardrone/Nymeria.h
- nymeria_ardrone/src/Nymeria.cpp

5.2 NymeriaCheckObstacle Class Reference

Definition of the class NymeriaCheckObstacle, that declares all functionalities in order to allow for obstacle detection.

`#include <NymeriaCheckObstacle.h>`

Public Member Functions

- NymeriaCheckObstacle ()

  Default constructor.
- NymeriaCheckObstacle (ros::NodeHandle *n)

  Constructor for the NymeriaCheckObstacle class Contains the navdata subscriber, sets the security distance to 100.0 and the speed factor to 1.0 by default.
- void inputCurFrontDist (int cfd)

  Update the distance between the drone and the obstacle, this value is stored in a ROS param named /nymeria↩ StateObstacle.
- double getSecurityDist ()

  Getter function for security distance, in order to permit the user to retain its current value.
- void setSecurityDist (double secDist)

  Setter function for security distance, in order to permit the user to change its value.
- double getSensorMaxRange ()

  Getter function for sensor max range, in order to permit the user to retain its current value.
- void setSensorMaxRange (double range)

  Setter function for sensor max range, in order to permit the user to change its value.

Private Member Functions

- void regulation (double angleEstimated, double userCmd)

  Regulation method Updates the speed factor stored in the ROS param "nymeriaFactor" according to the user orignial command and the estimated pitch of the drone.
- double pilotage (const double &distToObstacle, const double &securityDist, const double &userCmd)

  First regulataion of the speed factor command regarding the drone distance to obstacle.
- double PID (const double lastError, const double estimatedCmd)

  PID part of the regulation.
- double rebouclage (const double &angleEstimated)

  Conversion between the pitch of the drone and the speed factor.
- double saturationPente (const double lastCmd, const double param_saturation, double &currentCmd)

  Saturation of the derivative.
- void saturationCommande (double &cmd)

  Saturate the value of a variable to 1.0.

Private Attributes

- double error
- double sensorMaxRange
- ros::Subscriber sub_navdata
- ros::NodeHandle * nh

5.2.1 Detailed Description

Definition of the class NymeriaCheckObstacle, that declares all functionalities in order to allow for obstacle detection.

Author

Team-Nymeria

Version

0.2

Date

18th of January 2015

5.2.2 Constructor & Destructor Documentation

5.2.2.1 NymeriaCheckObstacle::NymeriaCheckObstacle ( )

Default constructor.

5.2.2.2 NymeriaCheckObstacle::NymeriaCheckObstacle (ros::NodeHandle * *n*)

Constructor for the NymeriaCheckObstacle class Contains the navdata subscriber, sets the security distance to 100.0 and the speed factor to 1.0 by default.

Parameters

n	Node handle for ROS

5.2.3 Member Function Documentation

5.2.3.1 double NymeriaCheckObstacle::getSecurityDist ( )

Getter function for security distance, in order to permit the user to retain its current value.

Returns

security distance.

5.2.3.2 double NymeriaCheckObstacle::getSensorMaxRange ( )

Getter function for sensor max range, in order to permit the user to retain its current value.

Returns

sensor max range.

5.2.3.3 void NymeriaCheckObstacle::inputCurFrontDist (int *cfd*)

Update the distance between the drone and the obstacle, this value is stored in a ROS param named /nymeria↩ StateObstacle.

Parameters

cfd	Current distance to the obstacle

5.2.3.4 double NymeriaCheckObstacle::PID (const double *lastError,* const double *cmd*) [private]

PID part of the regulation.

Parameters

in	*lastError*	

Returns

regulated command

5.2.3.5 double NymeriaCheckObstacle::pilotage (const double & *dist_To_Obstacle,* const double & *securityDist,* const double & *userCmd*) [private]

First regulataion of the speed factor command regarding the drone distance to obstacle.

Parameters

in	*dist_To_↩ Obstacle*	distance of the drone from the obstacle
in	*securityDist*	security distance
in	*userCmd*	initial user command (as speed factor)

Returns

regulated command

5.2.3.6 double NymeriaCheckObstacle::rebouclage (const double & *estimatedAngle*) [private]

Conversion between the pitch of the drone and the speed factor.

Parameters

in	*estimatedAngle*	drone pitch

Returns

speed factor

5.2.3.7 void NymeriaCheckObstacle::regulation (double *estimatedAngle,* double *userCmd*) [private]

Regulation method Updates the speed factor stored in the ROS param "nymeriaFactor" according to the user orignial command and the estimated pitch of the drone.

Parameters

userCmd	original command sent by user. Represented as a double corresponding to a linear speed factor

drone	pitch given by sensors on drone

Returns

void

5.2.3.8 void NymeriaCheckObstacle::saturationCommande (double & *cmd*) [private]

Saturate the value of a variable to 1.0.

Parameters

in,out	*cmd,value*	to saturate

5.2.3.9 double NymeriaCheckObstacle::saturationPente (const double *lastCmd,* const double *param_saturation,* double & *currentCmd*) [private]

Saturation of the derivative.

Parameters

lastCmd	last value of the variable to saturate
currentCmd	current value of the variable to saturate
param_↩ saturation	set the derivative limit

Returns

new saturated value

5.2.3.10 void NymeriaCheckObstacle::setSecurityDist (double *secDist*)

Setter function for security distance, in order to permit the user to change its value.

Parameters

secDist	security distance.

5.2.3.11 void NymeriaCheckObstacle::setSensorMaxRange (double *range*)

Setter function for sensor max range, in order to permit the user to change its value.

Parameters

range	- sensor max range.

5.2.4 Member Data Documentation

5.2.4.1 double NymeriaCheckObstacle::error [private]

5.2.4.2 ros::NodeHandle* NymeriaCheckObstacle::nh [private]

5.2.4.3 double NymeriaCheckObstacle::sensorMaxRange [private]

5.2.4.4 ros::Subscriber NymeriaCheckObstacle::sub_navdata [private]

The documentation for this class was generated from the following files:

- nymeria_ardrone/include/nymeria_ardrone/NymeriaCheckObstacle.h
- nymeria_ardrone/src/NymeriaCheckObstacle.cpp

5.3 NymeriaConstants Class Reference

Declaration of the class NymeriaConstants, that defines all constants necessary to define both commands and states of the drone and obstacles.

`#include <NymeriaConstants.h>`

Public Member Functions

- NymeriaConstants ()

  Constructor in order to create an object of the class NymeriaConstants.

Static Public Attributes

- static const double E_PARAM = -2.0
- static const int O_FRONT = -1
- static const int INIT = 0
- static const int M_FORWARD = 1
- static const int M_BACKWARD = 2
- static const int M_LEFT = 3
- static const int M_RIGHT = 4
- static const int M_UP = 5
- static const int M_DOWN = 6
- static const int T_LEFT = 7
- static const int T_RIGHT = 8
- static const int STOP = 9
- static const int TAKEOFF = 10
- static const int LAND = 11
- static const int E_STOP = 12
- static const int I_M_L_SPEED = 13
- static const int D_M_L_SPEED = 14
- static const int I_M_A_SPEED = 15
- static const int D_M_A_SPEED = 16
- static const int I_L_SPEED = 17
- static const int D_L_SPEED = 18
- static const int I_A_SPEED = 19
- static const int D_A_SPEED = 20
- static const int SLOW_DOWN = 21
- static const double ANTICIPATING_OBSTACLE_DISTANCE = 150.0

5.3.1 Detailed Description

Declaration of the class NymeriaConstants, that defines all constants necessary to define both commands and states of the drone and obstacles.

Author

Team-Nymeria

Version

0.2

Date

18th of January 2015

5.3.2 Constructor & Destructor Documentation

5.3.2.1 NymeriaConstants::NymeriaConstants ( )

Constructor in order to create an object of the class NymeriaConstants.

5.3.3 Member Data Documentation

5.3.3.1 const double NymeriaConstants::ANTICIPATING_OBSTACLE_DISTANCE = 150.0 [static]

5.3.3.2 const int NymeriaConstants::D_A_SPEED = 20 [static]

5.3.3.3 const int NymeriaConstants::D_L_SPEED = 18 [static]

5.3.3.4 const int NymeriaConstants::D_M_A_SPEED = 16 [static]

5.3.3.5 const int NymeriaConstants::D_M_L_SPEED = 14 [static]

5.3.3.6 const double NymeriaConstants::E_PARAM = -2.0 [static]

5.3.3.7 const int NymeriaConstants::E_STOP = 12 [static]

5.3.3.8 const int NymeriaConstants::I_A_SPEED = 19 [static]

5.3.3.9 const int NymeriaConstants::I_L_SPEED = 17 [static]

5.3.3.10 const int NymeriaConstants::I_M_A_SPEED = 15 [static]

5.3.3.11 const int NymeriaConstants::I_M_L_SPEED = 13 [static]

5.3.3.12 const int NymeriaConstants::INIT = 0 [static]

5.3.3.13 const int NymeriaConstants::LAND = 11 [static]

5.3.3.14 const int NymeriaConstants::M_BACKWARD = 2 [static]

5.3.3.15 const int NymeriaConstants::M_DOWN = 6 [static]

5.3.3.16 const int NymeriaConstants::M_FORWARD = 1 [static]

5.3.3.17 const int NymeriaConstants::M_LEFT = 3 [static]

5.3.3.18 const int NymeriaConstants::M_RIGHT = 4 [static]

5.3.3.19 const int NymeriaConstants::M_UP = 5 [static]

5.3.3.20 const int NymeriaConstants::O_FRONT = -1 [static]

5.3.3.21 const int NymeriaConstants::SLOW_DOWN = 21 [static]

5.3.3.22 const int NymeriaConstants::STOP = 9 [static]

5.3.3.23 const int NymeriaConstants::T_LEFT = 7 [static]

5.3.3.24 const int NymeriaConstants::T_RIGHT = 8 [static]

5.3.3.25 const int NymeriaConstants::TAKEOFF = 10 [static]

The documentation for this class was generated from the following files:

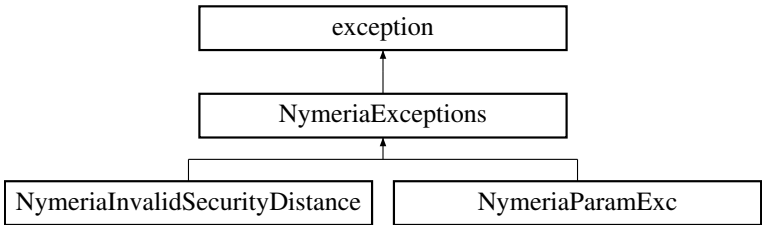
- nymeria_ardrone/include/nymeria_ardrone/NymeriaConstants.h
- nymeria_ardrone/src/NymeriaConstants.cpp

5.4 NymeriaExceptions Class Reference

Declaration of the class NymeriaExceptions, that declares the base class for all exceptions particular to Nymeria.

`#include <NymeriaExceptions.h>`

Inheritance diagram for NymeriaExceptions:



Public Member Functions

- NymeriaExceptions (string msg)

  Constructor in order to create an object of type NymeriaException.
- virtual ~NymeriaExceptions (void) throw ()
- virtual const char * what () const throw ()

  Overriding what() function from standard Exception.

Private Attributes

- string errMsg

5.4.1 Detailed Description

Declaration of the class NymeriaExceptions, that declares the base class for all exceptions particular to Nymeria.

Author

Team-Nymeria

Version

0.2

Date

18th of January 2015

5.4.2 Constructor & Destructor Documentation

5.4.2.1 NymeriaExceptions::NymeriaExceptions (string *msg*)

Constructor in order to create an object of type NymeriaException.

5.4.2.2 NymeriaExceptions::~NymeriaExceptions (void) throw) [virtual]

5.4.3 Member Function Documentation

5.4.3.1 const char * NymeriaExceptions::what ( ) const throw) [virtual]

Overriding what() function from standard Exception.

Reimplemented in NymeriaInvalidSecurityDistance, and NymeriaParamExc.

5.4.4 Member Data Documentation

5.4.4.1 string NymeriaExceptions::errMsg [private]

The documentation for this class was generated from the following files:

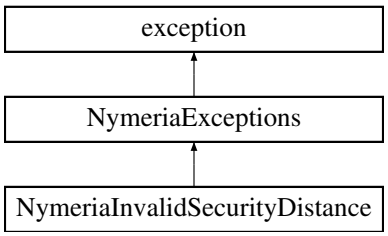
- nymeria_ardrone/include/nymeria_ardrone/NymeriaExceptions.h
- nymeria_ardrone/src/exception/NymeriaExceptions.cpp

5.5 NymeriaInvalidSecurityDistance Class Reference

Declaration of the class NymeriaParamExc, that declares the exception thrown when the ROS parameter requested does not exist or was misspelled.

`#include <NymeriaInvalidSecurityDistance.h>`

Inheritance diagram for NymeriaInvalidSecurityDistance:



Public Member Functions

- NymeriaInvalidSecurityDistance (void)

  Definition of the class NymeriaInvalidSecurityDistance, that defines the exception thrown when the an invalid security distance is entered.
- virtual ~NymeriaInvalidSecurityDistance (void) throw ()
- virtual const char * what () const throw ()

  Overriding what() function from standard Exception.

5.5.1 Detailed Description

Declaration of the class NymeriaParamExc, that declares the exception thrown when the ROS parameter requested does not exist or was misspelled.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 NymeriaInvalidSecurityDistance::NymeriaInvalidSecurityDistance (void)

Definition of the class NymeriaInvalidSecurityDistance, that defines the exception thrown when the an invalid security distance is entered.

5.5.2.2 NymeriaInvalidSecurityDistance::~NymeriaInvalidSecurityDistance (void) throw) [virtual]

5.5.3 Member Function Documentation

5.5.3.1 const char * NymeriaInvalidSecurityDistance::what ( ) const throw) [virtual]

Overriding what() function from standard Exception.

Reimplemented from NymeriaExceptions.

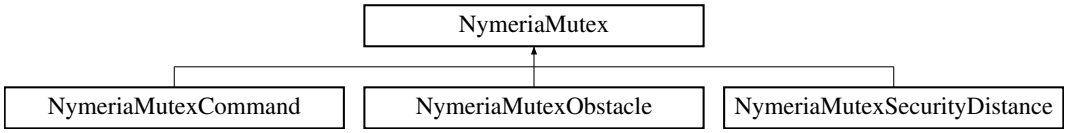
The documentation for this class was generated from the following files:

- nymeria_ardrone/include/nymeria_ardrone/NymeriaInvalidSecurityDistance.h
- nymeria_ardrone/src/exception/NymeriaInvalidSecurityDistance.cpp

5.6 NymeriaMutex Class Reference

`#include <NymeriaMutex.h>`

Inheritance diagram for NymeriaMutex:



Public Member Functions

- NymeriaMutex ()

  Defintion of the class NymeriaMutex, which serves as the parent class for all mutexes used in the context of Nymeria.

5.6.1 Constructor & Destructor Documentation

5.6.1.1 NymeriaMutex::NymeriaMutex ( )

Defintion of the class NymeriaMutex, which serves as the parent class for all mutexes used in the context of Nymeria.

Default constructor in order to create an object of type NymeriaMutex.

The documentation for this class was generated from the following files:

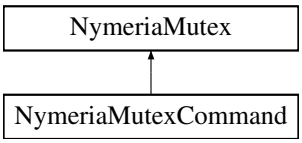
- nymeria_ardrone/include/nymeria_ardrone/NymeriaMutex.h
- nymeria_ardrone/src/NymeriaMutex.cpp

5.7 NymeriaMutexCommand Class Reference

Defintion of the class NymeriaMutexCommand, which manages access to the ROS Parameter nymeriaCommand.

`#include <NymeriaMutexCommand.h>`

Inheritance diagram for NymeriaMutexCommand:



Public Member Functions

- ~NymeriaMutexCommand ()

  Destructor resetting class attributes.

Static Public Member Functions

- static NymeriaMutexCommand * getInstance ()

  Function in order to get an instance of NymeriaMutexCommand.
- static void lock ()

  Method in order to lock or acquire resource.
- static void unlock ()

  Method in order to unlock or release resource.

Private Member Functions

- NymeriaMutexCommand ()

  Defintion of the class NymeriaMutexCommand, which manages access to the ROS Parameter nymeriaCommand.

Static Private Attributes

- static bool locked = false

  Attribute that marks whether or not the resource has been acquired yet.
- static bool instanceFlag = false

  Flag in order to make sure there is only one instance of the class.
- static NymeriaMutexCommand * mutexDrone = NULL

  First declaration of instance of type NymeriaMutex.

5.7.1 Detailed Description

Defintion of the class NymeriaMutexCommand, which manages access to the ROS Parameter nymeriaCommand.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 NymeriaMutexCommand::~NymeriaMutexCommand ( )

Destructor resetting class attributes.

5.7.2.2 NymeriaMutexCommand::NymeriaMutexCommand ( ) [private]

Defintion of the class NymeriaMutexCommand, which manages access to the ROS Parameter nymeriaCommand.

Default constructor in order to create an object of type NymeriaMutexCommand.

5.7.3 Member Function Documentation

5.7.3.1 NymeriaMutexCommand * NymeriaMutexCommand::getInstance ( ) [static]

Function in order to get an instance of NymeriaMutexCommand.

Returns

useful, i.e. not NULL object of type NymeriaMutex.

5.7.3.2 void NymeriaMutexCommand::lock ( ) [static]

Method in order to lock or acquire resource.

Resource can not be acquired by any other object while being locked.

5.7.3.3 void NymeriaMutexCommand::unlock ( ) [static]

Method in order to unlock or release resource.

Resource can be acquired by an other object after being released.

5.7.4 Member Data Documentation

5.7.4.1 bool NymeriaMutexCommand::instanceFlag = false [static], [private]

Flag in order to make sure there is only one instance of the class.

true - has been already instantiated once. false - hasn't been instantiated yet.

5.7.4.2 bool NymeriaMutexCommand::locked = false [static], [private]

Attribute that marks whether or not the resource has been acquired yet.

true - has been already acquired. false - hasn't been acquired yet.

5.7.4.3 NymeriaMutexCommand * NymeriaMutexCommand::mutexDrone = NULL [static], [private]

First declaration of instance of type NymeriaMutex.

The documentation for this class was generated from the following files:

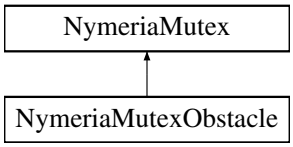
- nymeria_ardrone/include/nymeria_ardrone/NymeriaMutexCommand.h
- nymeria_ardrone/src/NymeriaMutexCommand.cpp

5.8 NymeriaMutexObstacle Class Reference

Defintion of the class NymeriaMutexObstacle, which manages access to the ROS Parameter nymeriaStateObstacle.

`#include <NymeriaMutexObstacle.h>`

Inheritance diagram for NymeriaMutexObstacle:



Public Member Functions

- ~NymeriaMutexObstacle ()

  Destructor resetting class attributes.

Static Public Member Functions

- static NymeriaMutexObstacle * getInstance ()

  Function in order to get an instance of NymeriaMutexObstacle.
- static void lock ()

  Method in order to lock or acquire resource.
- static void unlock ()

  Method in order to unlock or release resource.

Private Member Functions

- NymeriaMutexObstacle ()

  Defintion of the class NymeriaMutexObstacle, which manages access to the ROS Parameter nymeriaStateObstacle.

Static Private Attributes

- static bool locked = false

  Attribute that marks whether or not the resource has been acquired yet.
- static bool instanceFlag = false

  Flag in order to make sure there is only one instance of the class.
- static NymeriaMutexObstacle * mutexObstacle = NULL

  First declaration of instance of type NymeriaMutex.

5.8.1 Detailed Description

Defintion of the class NymeriaMutexObstacle, which manages access to the ROS Parameter nymeriaStateObstacle.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 NymeriaMutexObstacle::~NymeriaMutexObstacle ( )

Destructor resetting class attributes.

5.8.2.2 NymeriaMutexObstacle::NymeriaMutexObstacle ( ) [private]

Defintion of the class NymeriaMutexObstacle, which manages access to the ROS Parameter nymeriaStateObstacle.

Default constructor in order to create an object of type NymeriaStateObstacle.

5.8.3 Member Function Documentation

5.8.3.1 NymeriaMutexObstacle * NymeriaMutexObstacle::getInstance ( ) [static]

Function in order to get an instance of NymeriaMutexObstacle.

Returns

useful, i.e. not NULL object of type NymeriaMutex.

5.8.3.2 void NymeriaMutexObstacle::lock ( ) [static]

Method in order to lock or acquire resource.

Resource can not be acquired by any other object while being locked.

5.8.3.3 void NymeriaMutexObstacle::unlock ( ) [static]

Method in order to unlock or release resource.

Resource can be acquired by an other object after being released.

5.8.4 Member Data Documentation

5.8.4.1 bool NymeriaMutexObstacle::instanceFlag = false [static], [private]

Flag in order to make sure there is only one instance of the class.

true - has been already instantiated once. false - hasn't been instantiated yet.

5.8.4.2 bool NymeriaMutexObstacle::locked = false [static], [private]

Attribute that marks whether or not the resource has been acquired yet.

true - has been already acquired. false - hasn't been acquired yet.

5.8.4.3 NymeriaMutexObstacle * NymeriaMutexObstacle::mutexObstacle = NULL [static], [private]

First declaration of instance of type NymeriaMutex.

The documentation for this class was generated from the following files:

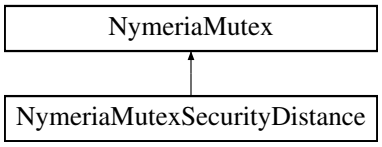
- nymeria_ardrone/include/nymeria_ardrone/NymeriaMutexObstacle.h
- nymeria_ardrone/src/NymeriaMutexObstacle.cpp

5.9 NymeriaMutexSecurityDistance Class Reference

Defintion of the class NymeriaMutexSecurityDistance, which manages access to the ROS Parameter nymeria↩ SecurityDistance.

`#include <NymeriaMutexSecurityDistance.h>`

Inheritance diagram for NymeriaMutexSecurityDistance:



Public Member Functions

- ~NymeriaMutexSecurityDistance ()

  Destructor resetting class attributes.

Static Public Member Functions

- static NymeriaMutexSecurityDistance * getInstance ()

  Function in order to get an instance of NymeriaMutexSecurityDistance.
- static void lock ()

  Method in order to lock or acquire resource.
- static void unlock ()

  Method in order to unlock or release resource.

Private Member Functions

- NymeriaMutexSecurityDistance ()

  Defintion of the class NymeriaMutexSecurityDistance, which manages access to the ROS Parameter nymeria↩ SecurityDistance.

Static Private Attributes

- static bool locked = false

  Attribute that marks whether or not the resource has been acquired yet.
- static bool instanceFlag = false

  Flag in order to make sure there is only one instance of the class.
- static NymeriaMutexSecurityDistance * mutexSecDist = NULL

  First declaration of instance of type NymeriaMutexSecurityDistance.

5.9.1 Detailed Description

Defintion of the class NymeriaMutexSecurityDistance, which manages access to the ROS Parameter nymeria↩ SecurityDistance.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 NymeriaMutexSecurityDistance::~NymeriaMutexSecurityDistance ( )

Destructor resetting class attributes.

5.9.2.2 NymeriaMutexSecurityDistance::NymeriaMutexSecurityDistance ( ) [private]

Defintion of the class NymeriaMutexSecurityDistance, which manages access to the ROS Parameter nymeria↩ SecurityDistance.

Default constructor in order to create an object of type NymeriaMutexSecurityDistance.

5.9.3 Member Function Documentation

5.9.3.1 NymeriaMutexSecurityDistance * NymeriaMutexSecurityDistance::getInstance ( ) [static]

Function in order to get an instance of NymeriaMutexSecurityDistance.

Returns

useful, i.e. not NULL object of type NymeriaMutexSecurityDistance.

5.9.3.2 void NymeriaMutexSecurityDistance::lock ( ) [static]

Method in order to lock or acquire resource.

Resource can not be acquired by any other object while being locked.

5.9.3.3 void NymeriaMutexSecurityDistance::unlock ( ) [static]

Method in order to unlock or release resource.

Resource can be acquired by an other object after being released.

5.9.4 Member Data Documentation

5.9.4.1 bool NymeriaMutexSecurityDistance::instanceFlag = false [static], [private]

Flag in order to make sure there is only one instance of the class.

true - has been already instantiated once. false - hasn't been instantiated yet.

5.9.4.2 bool NymeriaMutexSecurityDistance::locked = false [static], [private]

Attribute that marks whether or not the resource has been acquired yet.

true - has been already acquired. false - hasn't been acquired yet.

5.9.4.3 NymeriaMutexSecurityDistance * NymeriaMutexSecurityDistance::mutexSecDist = NULL [static], [private]

First declaration of instance of type NymeriaMutexSecurityDistance.

The documentation for this class was generated from the following files:

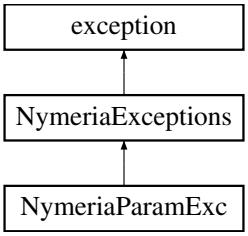
- nymeria_ardrone/include/nymeria_ardrone/NymeriaMutexSecurityDistance.h
- nymeria_ardrone/src/NymeriaMutexSecurityDistance.cpp

5.10 NymeriaParamExc Class Reference

Declaration of the class NymeriaParamExc, that declares the exception thrown when the ROS parameter requested does not exist or was misspelled.

`#include <NymeriaParamExc.h>`

Inheritance diagram for NymeriaParamExc:



Public Member Functions

- NymeriaParamExc (string msg="")

  Definition of the class NymeriaParamExc, that defines the exception thrown when the ROS parameter requested does not exist or was misspelled.
- virtual ~NymeriaParamExc (void) throw ()

  Method in order to throw exception.
- virtual const char * what () const throw ()

  Overriding what() function from standard Exception.

5.10.1 Detailed Description

Declaration of the class NymeriaParamExc, that declares the exception thrown when the ROS parameter requested does not exist or was misspelled.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 NymeriaParamExc::NymeriaParamExc (string *msg* = " ")

Definition of the class NymeriaParamExc, that defines the exception thrown when the ROS parameter requested does not exist or was misspelled.

Constructor in order to create an object of type NymeriaParamExc.

Parameters

msg	- message to be shown, when exception is thrown.

5.10.2.2 NymeriaParamExc::~NymeriaParamExc (void) throw) [virtual]

Method in order to throw exception.

5.10.3 Member Function Documentation

5.10.3.1 const char * NymeriaParamExc::what ( ) const throw) [virtual]

Overriding what() function from standard Exception.

Reimplemented from NymeriaExceptions.

The documentation for this class was generated from the following files:

- nymeria_ardrone/include/nymeria_ardrone/NymeriaParamExc.h
- nymeria_ardrone/src/exception/NymeriaParamExc.cpp

5.11 NymeriaTest Class Reference

Public Member Functions

- NymeriaTest ()
- ros::NodeHandle * getNH ()
- void loop (ros::NodeHandle *n)

  Central functionality of the NymeriaTest: trigger Nymeria in order to actiate obstacle detection and avoidance.

Private Attributes

- ros::NodeHandle nh

5.11.1 Constructor & Destructor Documentation

5.11.1.1 NymeriaTest::NymeriaTest ( )

5.11.2 Member Function Documentation

5.11.2.1 ros::NodeHandle * NymeriaTest::getNH ( )

5.11.2.2 void NymeriaTest::loop (ros::NodeHandle * *n*)

Central functionality of the NymeriaTest: trigger Nymeria in order to actiate obstacle detection and avoidance.

5.11.3 Member Data Documentation

5.11.3.1 ros::NodeHandle NymeriaTest::nh [private]

The documentation for this class was generated from the following file:

- nymeria_ardrone/test/NymeriaTest.cpp

Chapter 6

File Documentation

6.1 nymeria_ardrone/include/nymeria_ardrone/Nymeria.h File Reference

```
#include "ros/ros.h"
#include "std_msgs/Empty.h"
#include "geometry_msgs/Twist.h"
#include "std_msgs/UInt8.h"
#include "std_msgs/String.h"
#include <ardrone_autonomy/Navdata.h>
#include <nymeria_ardrone/NymeriaConstants.h>
#include <nymeria_ardrone/Controller.h>
```

Classes

- class Nymeria

  Definitions of the class Nymeria, that declares all functionalities in order to allow for drone navigation with obstacle detection and avoidance.

6.2 nymeria_ardrone/include/nymeria_ardrone/NymeriaCheckObstacle.h File Reference

```
#include "ros/ros.h"
#include <ardrone_autonomy/Navdata.h>
```

Classes

- class NymeriaCheckObstacle

  Definition of the class NymeriaCheckObstacle, that declares all functionalities in order to allow for obstacle detection.

Functions

- void stateDroneCallback (const ardrone_autonomy::Navdata &data)

  callback function for the subscriber sub_navdata gets the pitch of the drone and its state

6.2.1 Function Documentation

6.2.1.1 void stateDroneCallback (const ardrone_autonomy::Navdata & *data*)

callback function for the subscriber sub_navdata gets the pitch of the drone and its state

Parameters

data	variable where the value is stored, must be const

6.3 nymeria_ardrone/include/nymeria_ardrone/NymeriaConstants.h File Reference

Classes

- class NymeriaConstants

  Declaration of the class NymeriaConstants, that defines all constants necessary to define both commands and states of the drone and obstacles.

6.4 nymeria_ardrone/include/nymeria_ardrone/NymeriaExceptions.h File Reference

```
#include <exception>
#include <string>
```

Classes

- class NymeriaExceptions

  Declaration of the class NymeriaExceptions, that declares the base class for all exceptions particular to Nymeria.

6.5 nymeria_ardrone/include/nymeria_ardrone/NymeriaInvalidSecurityDistance.h File Reference

```
#include <nymeria_ardrone/NymeriaExceptions.h>
```

Classes

- class NymeriaInvalidSecurityDistance

  Declaration of the class NymeriaParamExc, that declares the exception thrown when the ROS parameter requested does not exist or was misspelled.

6.6 nymeria_ardrone/include/nymeria_ardrone/NymeriaMutex.h File Reference

Classes

- class NymeriaMutex

6.7 nymeria_ardrone/include/nymeria_ardrone/NymeriaMutexCommand.h File Reference

```
#include <nymeria_ardrone/NymeriaMutex.h>
```

Classes

- class NymeriaMutexCommand

  Defintion of the class NymeriaMutexCommand, which manages access to the ROS Parameter nymeriaCommand.

6.8 nymeria_ardrone/include/nymeria_ardrone/NymeriaMutexObstacle.h File Reference

```
#include <nymeria_ardrone/NymeriaMutex.h>
```

Classes

- class NymeriaMutexObstacle

  Defintion of the class NymeriaMutexObstacle, which manages access to the ROS Parameter nymeriaStateObstacle.

6.9 nymeria_ardrone/include/nymeria_ardrone/NymeriaMutexSecurityDistance.h File Reference

```
#include <nymeria_ardrone/NymeriaMutex.h>
```

Classes

- class NymeriaMutexSecurityDistance

  Defintion of the class NymeriaMutexSecurityDistance, which manages access to the ROS Parameter nymeria↩ SecurityDistance.

6.10 nymeria_ardrone/include/nymeria_ardrone/NymeriaParamExc.h File Reference

```
#include <nymeria_ardrone/NymeriaExceptions.h>
```

Classes

- class NymeriaParamExc

  Declaration of the class NymeriaParamExc, that declares the exception thrown when the ROS parameter requested does not exist or was misspelled.

6.11 nymeria_ardrone/README.md File Reference

6.12 nymeria_ardrone/src/exception/NymeriaExceptions.cpp File Reference

Definition of the class NymeriaExceptions, that defines the base class for all exceptions particular to Nymeria.

```
#include <nymeria_ardrone/NymeriaExceptions.h>
```

6.12.1 Detailed Description

Definition of the class NymeriaExceptions, that defines the base class for all exceptions particular to Nymeria.

6.13 nymeria_ardrone/src/exception/NymeriaInvalidSecurityDistance.cpp File Reference

```
#include <nymeria_ardrone/NymeriaInvalidSecurityDistance.h>
```

6.14 nymeria_ardrone/src/exception/NymeriaParamExc.cpp File Reference

```
#include <nymeria_ardrone/NymeriaParamExc.h>
```

6.15 nymeria_ardrone/src/Nymeria.cpp File Reference

```
#include <nymeria_ardrone/Nymeria.h>
#include <nymeria_ardrone/NymeriaParamExc.h>
#include <nymeria_ardrone/NymeriaInvalidSecurityDistance.h>
#include <nymeria_ardrone/NymeriaMutexCommand.h>
#include <nymeria_ardrone/NymeriaMutexObstacle.h>
#include <nymeria_ardrone/NymeriaMutexSecurityDistance.h>
#include <string.h>
```

6.16 nymeria_ardrone/src/NymeriaCheckObstacle.cpp File Reference

```
#include <nymeria_ardrone/NymeriaCheckObstacle.h>
#include <nymeria_ardrone/NymeriaParamExc.h>
#include <nymeria_ardrone/NymeriaInvalidSecurityDistance.h>
#include <nymeria_ardrone/NymeriaMutexObstacle.h>
#include <nymeria_ardrone/NymeriaMutexSecurityDistance.h>
```

Functions

- void stateDroneCallback (const ardrone_autonomy::Navdata &data)

  callback function for the subscriber sub_navdata gets the pitch of the drone and its state

Variables

- double pitch = 0.0
- int droneState = 0

6.16.1 Function Documentation

6.16.1.1 void stateDroneCallback (const ardrone_autonomy::Navdata & *data*)

callback function for the subscriber sub_navdata gets the pitch of the drone and its state

Parameters

data	variable where the value is stored, must be const

6.16.2 Variable Documentation

6.16.2.1 int droneState = 0

6.16.2.2 double pitch = 0.0

6.17 nymeria_ardrone/src/NymeriaConstants.cpp File Reference

Definition of the class NymeriaConstants.

```
#include <nymeria_ardrone/NymeriaConstants.h>
```

6.17.1 Detailed Description

Definition of the class NymeriaConstants.

6.18 nymeria_ardrone/src/NymeriaMutex.cpp File Reference

```
#include <nymeria_ardrone/NymeriaMutex.h>
```

6.19 nymeria_ardrone/src/NymeriaMutexCommand.cpp File Reference

```
#include <nymeria_ardrone/NymeriaMutexCommand.h>
#include <iostream>
```

6.20 nymeria_ardrone/src/NymeriaMutexObstacle.cpp File Reference

```
#include <nymeria_ardrone/NymeriaMutexObstacle.h>
#include <iostream>
```

6.21 nymeria_ardrone/src/NymeriaMutexSecurityDistance.cpp File Reference

```
#include <nymeria_ardrone/NymeriaMutexSecurityDistance.h>
#include <iostream>
```

6.22 nymeria_ardrone/test/NymeriaTest.cpp File Reference

```
#include <signal.h>
#include <termios.h>
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <string.h>
#include "ros/ros.h"
#include "std_msgs/Empty.h"
#include "geometry_msgs/Twist.h"
#include "std_msgs/UInt8.h"
#include <ardrone_autonomy/Navdata.h>
#include <nymeria_ardrone/Nymeria.h>
```

Classes

- class NymeriaTest

Functions

- int main (int argc, char **argv)

6.22.1 Function Documentation

6.22.1.1 int main (int *argc*, char ** *argv*)

Index