

nymeria\_ar drone

By Project Nymeria

Wed Jan 21 2015 10:36:53



# Contents

<b>1</b>	<b>Main Page</b>	<b>1</b>
<b>2</b>	<b>Hierarchical Index</b>	<b>5</b>
2.1	Class Hierarchy . . . . .	5
<b>3</b>	<b>Class Index</b>	<b>7</b>
3.1	Class List . . . . .	7
<b>4</b>	<b>File Index</b>	<b>9</b>
4.1	File List . . . . .	9
<b>5</b>	<b>Class Documentation</b>	<b>11</b>
5.1	Nymeria Class Reference . . . . .	11
5.1.1	Detailed Description . . . . .	13
5.1.2	Constructor & Destructor Documentation . . . . .	14
5.1.2.1	Nymeria . . . . .	14
5.1.2.2	Nymeria . . . . .	14
5.1.3	Member Function Documentation . . . . .	14
5.1.3.1	decreaseAngularSpeed . . . . .	14
5.1.3.2	decreaseLinearSpeed . . . . .	14
5.1.3.3	decreaseMaxAngularSpeed . . . . .	14
5.1.3.4	decreaseMaxLinearSpeed . . . . .	14
5.1.3.5	emergencyStop . . . . .	14
5.1.3.6	getAngularSpeed . . . . .	14
5.1.3.7	getLinearSpeed . . . . .	14
5.1.3.8	getMaxAngularSpeed . . . . .	15
5.1.3.9	getMaxLinearSpeed . . . . .	15
5.1.3.10	getParameter . . . . .	15
5.1.3.11	getSecurityDist . . . . .	15
5.1.3.12	increaseAngularSpeed . . . . .	15
5.1.3.13	increaseLinearSpeed . . . . .	15
5.1.3.14	increaseMaxAngularSpeed . . . . .	15
5.1.3.15	increaseMaxLinearSpeed . . . . .	15

5.1.3.16	<a href="#">init_move_msg</a>	16
5.1.3.17	<a href="#">init_publishers</a>	16
5.1.3.18	<a href="#">init_rosParams</a>	16
5.1.3.19	<a href="#">init_safeActions</a>	16
5.1.3.20	<a href="#">inRange</a>	16
5.1.3.21	<a href="#">isSafeAction</a>	16
5.1.3.22	<a href="#">keepSecurityDistance</a>	16
5.1.3.23	<a href="#">land</a>	16
5.1.3.24	<a href="#">moveBackward</a>	17
5.1.3.25	<a href="#">moveDown</a>	17
5.1.3.26	<a href="#">moveForward</a>	17
5.1.3.27	<a href="#">moveLeft</a>	17
5.1.3.28	<a href="#">moveRight</a>	17
5.1.3.29	<a href="#">moveUp</a>	17
5.1.3.30	<a href="#">obstaclePossible</a>	17
5.1.3.31	<a href="#">reactionRoutine</a>	17
5.1.3.32	<a href="#">setLinearSpeed</a>	17
5.1.3.33	<a href="#">setMaxAngularSpeed</a>	17
5.1.3.34	<a href="#">setMaxLinearSpeed</a>	18
5.1.3.35	<a href="#">setSecurityDist</a>	18
5.1.3.36	<a href="#">slowDown</a>	18
5.1.3.37	<a href="#">stop</a>	18
5.1.3.38	<a href="#">takeOff</a>	18
5.1.3.39	<a href="#">triggerAction</a>	18
5.1.3.40	<a href="#">turnLeft</a>	18
5.1.3.41	<a href="#">turnRight</a>	18
5.1.3.42	<a href="#">underSecurityDist</a>	19
5.1.3.43	<a href="#">validateStates</a>	19
5.1.4	<a href="#">Friends And Related Function Documentation</a>	19
5.1.4.1	<a href="#">Controller</a>	19
5.1.5	<a href="#">Member Data Documentation</a>	19
5.1.5.1	<a href="#">angularSpeed</a>	19
5.1.5.2	<a href="#">empty_msg</a>	19
5.1.5.3	<a href="#">lastCmd</a>	19
5.1.5.4	<a href="#">linearSpeed</a>	19
5.1.5.5	<a href="#">maxAngularSpeed</a>	19
5.1.5.6	<a href="#">maxLinearSpeed</a>	19
5.1.5.7	<a href="#">move_msg</a>	19
5.1.5.8	<a href="#">navData</a>	19
5.1.5.9	<a href="#">nh</a>	19

5.1.5.10	pub_cmd_land . . . . .	19
5.1.5.11	pub_cmd_move . . . . .	19
5.1.5.12	pub_cmd_reset . . . . .	19
5.1.5.13	pub_cmd_takeoff . . . . .	19
5.1.5.14	safeActions . . . . .	19
5.1.5.15	sub_navdata . . . . .	19
5.2	NymeriaCheckObstacle Class Reference . . . . .	20
5.2.1	Detailed Description . . . . .	21
5.2.2	Constructor & Destructor Documentation . . . . .	21
5.2.2.1	NymeriaCheckObstacle . . . . .	21
5.2.2.2	NymeriaCheckObstacle . . . . .	21
5.2.3	Member Function Documentation . . . . .	21
5.2.3.1	getSecurityDist . . . . .	21
5.2.3.2	getSensorMaxRange . . . . .	21
5.2.3.3	inputCurFrontDist . . . . .	21
5.2.3.4	PID . . . . .	22
5.2.3.5	pilotage . . . . .	22
5.2.3.6	rebouclage . . . . .	22
5.2.3.7	regulation . . . . .	22
5.2.3.8	saturationCommande . . . . .	23
5.2.3.9	saturationPente . . . . .	23
5.2.3.10	setSecurityDist . . . . .	23
5.2.3.11	setSensorMaxRange . . . . .	23
5.2.4	Member Data Documentation . . . . .	23
5.2.4.1	error . . . . .	23
5.2.4.2	nh . . . . .	23
5.2.4.3	sensorMaxRange . . . . .	23
5.2.4.4	sub_navdata . . . . .	24
5.3	NymeriaConstants Class Reference . . . . .	24
5.3.1	Detailed Description . . . . .	25
5.3.2	Constructor & Destructor Documentation . . . . .	25
5.3.2.1	NymeriaConstants . . . . .	25
5.3.3	Member Data Documentation . . . . .	25
5.3.3.1	ANTICIPATING_OBSTACLE_DISTANCE . . . . .	25
5.3.3.2	D_A_SPEED . . . . .	25
5.3.3.3	D_L_SPEED . . . . .	25
5.3.3.4	D_M_A_SPEED . . . . .	25
5.3.3.5	D_M_L_SPEED . . . . .	25
5.3.3.6	E_PARAM . . . . .	25
5.3.3.7	E_STOP . . . . .	25

5.3.3.8	I_A_SPEED	25
5.3.3.9	I_L_SPEED	25
5.3.3.10	I_M_A_SPEED	25
5.3.3.11	I_M_L_SPEED	25
5.3.3.12	INIT	25
5.3.3.13	LAND	25
5.3.3.14	M_BACKWARD	25
5.3.3.15	M_DOWN	25
5.3.3.16	M_FORWARD	26
5.3.3.17	M_LEFT	26
5.3.3.18	M_RIGHT	26
5.3.3.19	M_UP	26
5.3.3.20	O_FRONT	26
5.3.3.21	SLOW_DOWN	26
5.3.3.22	STOP	26
5.3.3.23	T_LEFT	26
5.3.3.24	T_RIGHT	26
5.3.3.25	TAKEOFF	26
5.4	NymeriaExceptions Class Reference	26
5.4.1	Detailed Description	27
5.4.2	Constructor & Destructor Documentation	27
5.4.2.1	NymeriaExceptions	27
5.4.2.2	~NymeriaExceptions	27
5.4.3	Member Function Documentation	27
5.4.3.1	what	27
5.4.4	Member Data Documentation	27
5.4.4.1	errMsg	27
5.5	NymeriaInvalidSecurityDistance Class Reference	27
5.5.1	Detailed Description	28
5.5.2	Constructor & Destructor Documentation	28
5.5.2.1	NymeriaInvalidSecurityDistance	28
5.5.2.2	~NymeriaInvalidSecurityDistance	28
5.5.3	Member Function Documentation	28
5.5.3.1	what	28
5.6	NymeriaMutex Class Reference	28
5.6.1	Constructor & Destructor Documentation	29
5.6.1.1	NymeriaMutex	29
5.7	NymeriaMutexCommand Class Reference	29
5.7.1	Detailed Description	30
5.7.2	Constructor & Destructor Documentation	30

5.7.2.1	~NymeriaMutexCommand	30
5.7.2.2	NymeriaMutexCommand	30
5.7.3	Member Function Documentation	30
5.7.3.1	getInstance	30
5.7.3.2	lock	30
5.7.3.3	unlock	30
5.7.4	Member Data Documentation	31
5.7.4.1	instanceFlag	31
5.7.4.2	locked	31
5.7.4.3	mutexDrone	31
5.8	NymeriaMutexObstacle Class Reference	31
5.8.1	Detailed Description	32
5.8.2	Constructor & Destructor Documentation	32
5.8.2.1	~NymeriaMutexObstacle	32
5.8.2.2	NymeriaMutexObstacle	32
5.8.3	Member Function Documentation	32
5.8.3.1	getInstance	32
5.8.3.2	lock	32
5.8.3.3	unlock	32
5.8.4	Member Data Documentation	33
5.8.4.1	instanceFlag	33
5.8.4.2	locked	33
5.8.4.3	mutexObstacle	33
5.9	NymeriaMutexSecurityDistance Class Reference	33
5.9.1	Detailed Description	34
5.9.2	Constructor & Destructor Documentation	34
5.9.2.1	~NymeriaMutexSecurityDistance	34
5.9.2.2	NymeriaMutexSecurityDistance	34
5.9.3	Member Function Documentation	34
5.9.3.1	getInstance	34
5.9.3.2	lock	34
5.9.3.3	unlock	34
5.9.4	Member Data Documentation	35
5.9.4.1	instanceFlag	35
5.9.4.2	locked	35
5.9.4.3	mutexSecDist	35
5.10	NymeriaParamExc Class Reference	35
5.10.1	Detailed Description	35
5.10.2	Constructor & Destructor Documentation	36
5.10.2.1	NymeriaParamExc	36

5.10.2.2	<a href="#">~NymeriaParamExc</a>	36
5.10.3	<a href="#">Member Function Documentation</a>	36
5.10.3.1	<a href="#">what</a>	36
5.11	<a href="#">NymeriaTest Class Reference</a>	36
5.11.1	<a href="#">Constructor &amp; Destructor Documentation</a>	36
5.11.1.1	<a href="#">NymeriaTest</a>	36
5.11.2	<a href="#">Member Function Documentation</a>	36
5.11.2.1	<a href="#">getNH</a>	36
5.11.2.2	<a href="#">loop</a>	36
5.11.3	<a href="#">Member Data Documentation</a>	37
5.11.3.1	<a href="#">nh</a>	37
<b>6</b>	<b><a href="#">File Documentation</a></b>	<b>39</b>
6.1	<a href="#">nymeria_ardrone/include/nymeria_ardrone/Nymeria.h File Reference</a>	39
6.2	<a href="#">nymeria_ardrone/include/nymeria_ardrone/NymeriaCheckObstacle.h File Reference</a>	39
6.2.1	<a href="#">Function Documentation</a>	40
6.2.1.1	<a href="#">stateDroneCallback</a>	40
6.3	<a href="#">nymeria_ardrone/include/nymeria_ardrone/NymeriaConstants.h File Reference</a>	41
6.4	<a href="#">nymeria_ardrone/include/nymeria_ardrone/NymeriaExceptions.h File Reference</a>	41
6.5	<a href="#">nymeria_ardrone/include/nymeria_ardrone/NymeriaInvalidSecurityDistance.h File Reference</a>	41
6.6	<a href="#">nymeria_ardrone/include/nymeria_ardrone/NymeriaMutex.h File Reference</a>	41
6.7	<a href="#">nymeria_ardrone/include/nymeria_ardrone/NymeriaMutexCommand.h File Reference</a>	41
6.8	<a href="#">nymeria_ardrone/include/nymeria_ardrone/NymeriaMutexObstacle.h File Reference</a>	42
6.9	<a href="#">nymeria_ardrone/include/nymeria_ardrone/NymeriaMutexSecurityDistance.h File Reference</a>	42
6.10	<a href="#">nymeria_ardrone/include/nymeria_ardrone/NymeriaParamExc.h File Reference</a>	42
6.11	<a href="#">nymeria_ardrone/README.md File Reference</a>	42
6.12	<a href="#">nymeria_ardrone/src/exception/NymeriaExceptions.cpp File Reference</a>	42
6.12.1	<a href="#">Detailed Description</a>	43
6.13	<a href="#">nymeria_ardrone/src/exception/NymeriaInvalidSecurityDistance.cpp File Reference</a>	43
6.14	<a href="#">nymeria_ardrone/src/exception/NymeriaParamExc.cpp File Reference</a>	43
6.15	<a href="#">nymeria_ardrone/src/Nymeria.cpp File Reference</a>	43
6.16	<a href="#">nymeria_ardrone/src/NymeriaCheckObstacle.cpp File Reference</a>	43
6.16.1	<a href="#">Function Documentation</a>	43
6.16.1.1	<a href="#">stateDroneCallback</a>	43
6.16.2	<a href="#">Variable Documentation</a>	44
6.16.2.1	<a href="#">droneState</a>	44
6.16.2.2	<a href="#">pitch</a>	44
6.17	<a href="#">nymeria_ardrone/src/NymeriaConstants.cpp File Reference</a>	44
6.17.1	<a href="#">Detailed Description</a>	44
6.18	<a href="#">nymeria_ardrone/src/NymeriaMutex.cpp File Reference</a>	44



---

6.19	<a href="#">nymeria_ardrone/src/NymeriaMutexCommand.cpp File Reference</a>	44
6.20	<a href="#">nymeria_ardrone/src/NymeriaMutexObstacle.cpp File Reference</a>	44
6.21	<a href="#">nymeria_ardrone/src/NymeriaMutexSecurityDistance.cpp File Reference</a>	44
6.22	<a href="#">nymeria_ardrone/test/NymeriaTest.cpp File Reference</a>	45
6.22.1	Function Documentation	45
6.22.1.1	main	45
 <b>Index</b>		 <b>47</b>



# Chapter 1

## Main Page

nymeria\_ardrone is a [ROS](#) package for [Parrot AR-Drone](#) quadcopter. 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](#)
- [Installation](#)
- [How to run it](#)
- [How does it work](#)
- [Some examples](#)

### 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 :

```
$ roslaunch 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(cutValue);* Replace the 'cutValue' variable by the current distance of the front sensor of your drone. Once done, run the sensor\_interface node :

```
$ roslaunch 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 :

```
$ roslaunch 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 detailed 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

```
$ rosrn 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 kept, 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 library 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 *DroneUDPServer.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:

exception	
NymeriaExceptions . . . . .	26
NymeriaInvalidSecurityDistance . . . . .	27
NymeriaParamExc . . . . .	35
Nymeria . . . . .	11
NymeriaCheckObstacle . . . . .	20
NymeriaConstants . . . . .	24
NymeriaMutex . . . . .	28
NymeriaMutexCommand . . . . .	29
NymeriaMutexObstacle . . . . .	31
NymeriaMutexSecurityDistance . . . . .	33
NymeriaTest . . . . .	36





## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">Nymeria</a>	Definitions of the class <a href="#">Nymeria</a> , that declares all functionalities in order to allow for drone navigation with obstacle detection and avoidance . . . . .	11
<a href="#">NymeriaCheckObstacle</a>	Definition of the class <a href="#">NymeriaCheckObstacle</a> , that declares all functionalities in order to allow for obstacle detection . . . . .	20
<a href="#">NymeriaConstants</a>	Declaration of the class <a href="#">NymeriaConstants</a> , that defines all constants necessary to define both commands and states of the drone and obstacles . . . . .	24
<a href="#">NymeriaExceptions</a>	Declaration of the class <a href="#">NymeriaExceptions</a> , that declares the base class for all exceptions particular to <a href="#">Nymeria</a> . . . . .	26
<a href="#">NymeriaInvalidSecurityDistance</a>	Declaration of the class <a href="#">NymeriaParamExc</a> , that declares the exception thrown when the ROS parameter requested does not exist or was misspelled . . . . .	27
<a href="#">NymeriaMutex</a>	. . . . .	28
<a href="#">NymeriaMutexCommand</a>	Defintion of the class <a href="#">NymeriaMutexCommand</a> , which manages access to the ROS Parameter <code>nymeriaCommand</code> . . . . .	29
<a href="#">NymeriaMutexObstacle</a>	Defintion of the class <a href="#">NymeriaMutexObstacle</a> , which manages access to the ROS Parameter <code>nymeriaStateObstacle</code> . . . . .	31
<a href="#">NymeriaMutexSecurityDistance</a>	Defintion of the class <a href="#">NymeriaMutexSecurityDistance</a> , which manages access to the ROS Parameter <code>nymeriaSecurityDistance</code> . . . . .	33
<a href="#">NymeriaParamExc</a>	Declaration of the class <a href="#">NymeriaParamExc</a> , that declares the exception thrown when the ROS parameter requested does not exist or was misspelled . . . . .	35
<a href="#">NymeriaTest</a>	. . . . .	36



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

nymeria_drone/include/nymeria_drone/Nymeria.h . . . . .	39
nymeria_drone/include/nymeria_drone/NymeriaCheckObstacle.h . . . . .	39
nymeria_drone/include/nymeria_drone/NymeriaConstants.h . . . . .	41
nymeria_drone/include/nymeria_drone/NymeriaExceptions.h . . . . .	41
nymeria_drone/include/nymeria_drone/NymeriaInvalidSecurityDistance.h . . . . .	41
nymeria_drone/include/nymeria_drone/NymeriaMutex.h . . . . .	41
nymeria_drone/include/nymeria_drone/NymeriaMutexCommand.h . . . . .	41
nymeria_drone/include/nymeria_drone/NymeriaMutexObstacle.h . . . . .	42
nymeria_drone/include/nymeria_drone/NymeriaMutexSecurityDistance.h . . . . .	42
nymeria_drone/include/nymeria_drone/NymeriaParamExc.h . . . . .	42
nymeria_drone/src/Nymeria.cpp . . . . .	43
nymeria_drone/src/NymeriaCheckObstacle.cpp . . . . .	43
nymeria_drone/src/NymeriaConstants.cpp . . . . .	
Definition of the class <a href="#">NymeriaConstants</a> . . . . .	44
nymeria_drone/src/NymeriaMutex.cpp . . . . .	44
nymeria_drone/src/NymeriaMutexCommand.cpp . . . . .	44
nymeria_drone/src/NymeriaMutexObstacle.cpp . . . . .	44
nymeria_drone/src/NymeriaMutexSecurityDistance.cpp . . . . .	44
nymeria_drone/src/exception/NymeriaExceptions.cpp . . . . .	
Definition of the class <a href="#">NymeriaExceptions</a> , that defines the base class for all exceptions particular to <a href="#">Nymeria</a> . . . . .	42
nymeria_drone/src/exception/NymeriaInvalidSecurityDistance.cpp . . . . .	43
nymeria_drone/src/exception/NymeriaParamExc.cpp . . . . .	43
nymeria_drone/test/NymeriaTest.cpp . . . . .	45



## 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

<i>n</i>	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**

<i>str</i>	- 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, nymeriaSecurityDist.

**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**

<i>min</i>	left border of interval.
<i>max</i>	right border of interval.
<i>value</i>	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**

<i>cmd</i>	- 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**

<i>speed</i>	- 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

<i>speed</i>	- 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

<i>speed</i>	- 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

<i>secDist</i>	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

<i>cmd</i>	- incoming command.
<i>factor</i>	- 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 original 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

<i>n</i>	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

<i>cfd</i>	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	<i>lastError</i>	
----	------------------	--

## Returns

regulated command

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

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

## Parameters

in	<i>dist_To_Obstacle</i>	distance of the drone from the obstacle
in	<i>securityDist</i>	security distance
in	<i>userCmd</i>	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	<i>estimatedAngle</i>	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 original command and the estimated pitch of the drone.

## Parameters

<i>userCmd</i>	original command sent by user. Represented as a double corresponding to a linear speed factor
----------------	---



<i>drone</i>	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

<i>in, out</i>	<i>cmd,value</i>	to saturate
----------------	------------------	-------------

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

Saturation of the derivative.

## Parameters

<i>lastCmd</i>	last value of the variable to saturate
<i>currentCmd</i>	current value of the variable to saturate
<i>param_saturation</i>	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

<i>secDist</i>	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

<i>range</i>	- 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:

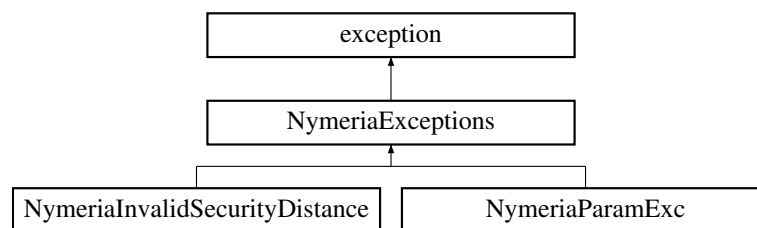
- [nymeria\\_ar drone/include/nymeria\\_ar drone/NymeriaConstants.h](#)
- [nymeria\\_ar drone/src/NymeriaConstants.cpp](#)

## 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:

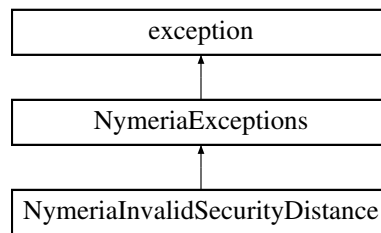
- `nymeria_ardrone/include/nymeria_ardrone/NymeriaExceptions.h`
- `nymeria_ardrone/src/exception/NymeriaExceptions.cpp`

## 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](#).

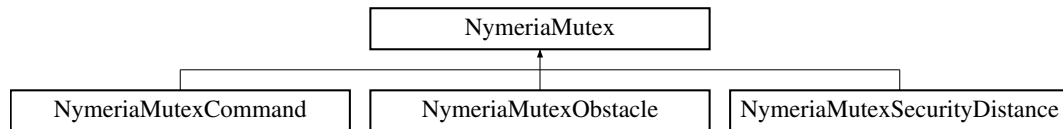
The documentation for this class was generated from the following files:

- [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:

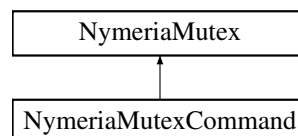
- [nymeria\\_ardrone/include/nymeria\\_ardrone/NymeriaMutex.h](#)
- [nymeria\\_ardrone/src/NymeriaMutex.cpp](#)

## 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:

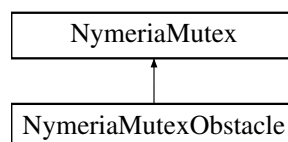
- [nymeria\\_ardrone/include/nymeria\\_ardrone/NymeriaMutexCommand.h](#)
- [nymeria\\_ardrone/src/NymeriaMutexCommand.cpp](#)

## 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:

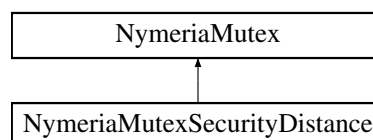
- [nymeria\\_ardrone/include/nymeria\\_ardrone/NymeriaMutexObstacle.h](#)
- [nymeria\\_ardrone/src/NymeriaMutexObstacle.cpp](#)

## 5.9 NymeriaMutexSecurityDistance Class Reference

Definition of the class [NymeriaMutexSecurityDistance](#), which manages access to the ROS Parameter `nymeriaSecurityDistance`.

```
#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:

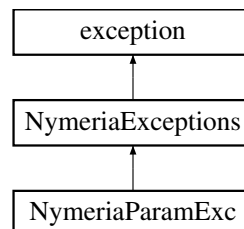
- [nymeria\\_ardrone/include/nymeria\\_ardrone/NymeriaMutexSecurityDistance.h](#)
- [nymeria\\_ardrone/src/NymeriaMutexSecurityDistance.cpp](#)

## 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

<code>msg</code>	- 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_drone/include/nymeria_drone/NymeriaParamExc.h`
- `nymeria_drone/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\_ar drone/include/nymeria\_ar drone/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_ar drone/NymeriaConstants.h>
#include <nymeria_ar drone/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\_ar drone/include/nymeria\_ar drone/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

<i>data</i>	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 `nymeriaSecurityDistance`.

## 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\_ar drone/src/exception/NymeriaInvalidSecurityDistance.cpp File Reference

```
#include <nymeria_ar drone/NymeriaInvalidSecurityDistance.h>
```

## 6.14 nymeria\_ar drone/src/exception/NymeriaParamExc.cpp File Reference

```
#include <nymeria_ar drone/NymeriaParamExc.h>
```

## 6.15 nymeria\_ar drone/src/Nymeria.cpp File Reference

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

## 6.16 nymeria\_ar drone/src/NymeriaCheckObstacle.cpp File Reference

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

### Functions

- void [stateDroneCallback](#) (const ar drone\_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 ar drone\_autonomy::Navdata & data )

callback function for the subscriber sub\_navdata gets the pitch of the drone and its state

## Parameters

<i>data</i>	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\_ar drone/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_ar drone/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

- ~NymeriaExceptions
  - NymeriaExceptions, [27](#)
- ~NymeriaInvalidSecurityDistance
  - NymeriaInvalidSecurityDistance, [28](#)
- ~NymeriaMutexCommand
  - NymeriaMutexCommand, [30](#)
- ~NymeriaMutexObstacle
  - NymeriaMutexObstacle, [32](#)
- ~NymeriaMutexSecurityDistance
  - NymeriaMutexSecurityDistance, [34](#)
- ~NymeriaParamExc
  - NymeriaParamExc, [36](#)
- ANTICIPATING\_OBSTACLE\_DISTANCE
  - NymeriaConstants, [25](#)
- angularSpeed
  - Nymeria, [19](#)
- Controller
  - Nymeria, [19](#)
- D\_A\_SPEED
  - NymeriaConstants, [25](#)
- D\_L\_SPEED
  - NymeriaConstants, [25](#)
- D\_M\_A\_SPEED
  - NymeriaConstants, [25](#)
- D\_M\_L\_SPEED
  - NymeriaConstants, [25](#)
- decreaseAngularSpeed
  - Nymeria, [14](#)
- decreaseLinearSpeed
  - Nymeria, [14](#)
- decreaseMaxAngularSpeed
  - Nymeria, [14](#)
- decreaseMaxLinearSpeed
  - Nymeria, [14](#)
- droneState
  - NymeriaCheckObstacle.cpp, [44](#)
- E\_PARAM
  - NymeriaConstants, [25](#)
- E\_STOP
  - NymeriaConstants, [25](#)
- emergencyStop
  - Nymeria, [14](#)
- empty\_msg
  - Nymeria, [19](#)
- errMsg
  - NymeriaExceptions, [27](#)

- error
  - NymeriaCheckObstacle, [23](#)
- getAngularSpeed
  - Nymeria, [14](#)
- getInstance
  - NymeriaMutexCommand, [30](#)
  - NymeriaMutexObstacle, [32](#)
  - NymeriaMutexSecurityDistance, [34](#)
- getLinearSpeed
  - Nymeria, [14](#)
- getMaxAngularSpeed
  - Nymeria, [14](#)
- getMaxLinearSpeed
  - Nymeria, [15](#)
- getNH
  - NymeriaTest, [36](#)
- getParameter
  - Nymeria, [15](#)
- getSecurityDist
  - Nymeria, [15](#)
  - NymeriaCheckObstacle, [21](#)
- getSensorMaxRange
  - NymeriaCheckObstacle, [21](#)
- I\_A\_SPEED
  - NymeriaConstants, [25](#)
- I\_L\_SPEED
  - NymeriaConstants, [25](#)
- I\_M\_A\_SPEED
  - NymeriaConstants, [25](#)
- I\_M\_L\_SPEED
  - NymeriaConstants, [25](#)
- INIT
  - NymeriaConstants, [25](#)
- inRange
  - Nymeria, [16](#)
- increaseAngularSpeed
  - Nymeria, [15](#)
- increaseLinearSpeed
  - Nymeria, [15](#)
- increaseMaxAngularSpeed
  - Nymeria, [15](#)
- increaseMaxLinearSpeed
  - Nymeria, [15](#)
- init\_move\_msg
  - Nymeria, [15](#)
- init\_publishers
  - Nymeria, [16](#)
- init\_rosParams

- Nymeria, 16
- init\_safeActions
  - Nymeria, 16
- inputCurFrontDist
  - NymeriaCheckObstacle, 21
- instanceFlag
  - NymeriaMutexCommand, 31
  - NymeriaMutexObstacle, 33
  - NymeriaMutexSecurityDistance, 35
- isSafeAction
  - Nymeria, 16
- keepSecurityDistance
  - Nymeria, 16
- LAND
  - NymeriaConstants, 25
- land
  - Nymeria, 16
- lastCmd
  - Nymeria, 19
- linearSpeed
  - Nymeria, 19
- lock
  - NymeriaMutexCommand, 30
  - NymeriaMutexObstacle, 32
  - NymeriaMutexSecurityDistance, 34
- locked
  - NymeriaMutexCommand, 31
  - NymeriaMutexObstacle, 33
  - NymeriaMutexSecurityDistance, 35
- loop
  - NymeriaTest, 36
- M\_BACKWARD
  - NymeriaConstants, 25
- M\_DOWN
  - NymeriaConstants, 25
- M\_FORWARD
  - NymeriaConstants, 25
- M\_LEFT
  - NymeriaConstants, 26
- M\_RIGHT
  - NymeriaConstants, 26
- M\_UP
  - NymeriaConstants, 26
- main
  - NymeriaTest.cpp, 45
- maxAngularSpeed
  - Nymeria, 19
- maxLinearSpeed
  - Nymeria, 19
- move\_msg
  - Nymeria, 19
- moveBackward
  - Nymeria, 16
- moveDown
  - Nymeria, 17
- moveForward
  - Nymeria, 17
- moveLeft
  - Nymeria, 17
- moveRight
  - Nymeria, 17
- moveUp
  - Nymeria, 17
- mutexDrone
  - NymeriaMutexCommand, 31
- mutexObstacle
  - NymeriaMutexObstacle, 33
- mutexSecDist
  - NymeriaMutexSecurityDistance, 35
- navData
  - Nymeria, 19
- nh
  - Nymeria, 19
  - NymeriaCheckObstacle, 23
  - NymeriaTest, 37
- Nymeria, 11
  - angularSpeed, 19
  - Controller, 19
  - decreaseAngularSpeed, 14
  - decreaseLinearSpeed, 14
  - decreaseMaxAngularSpeed, 14
  - decreaseMaxLinearSpeed, 14
  - emergencyStop, 14
  - empty\_msg, 19
  - getAngularSpeed, 14
  - getLinearSpeed, 14
  - getMaxAngularSpeed, 14
  - getMaxLinearSpeed, 15
  - getParameter, 15
  - getSecurityDist, 15
  - inRange, 16
  - increaseAngularSpeed, 15
  - increaseLinearSpeed, 15
  - increaseMaxAngularSpeed, 15
  - increaseMaxLinearSpeed, 15
  - init\_move\_msg, 15
  - init\_publishers, 16
  - init\_rosParams, 16
  - init\_safeActions, 16
  - isSafeAction, 16
  - keepSecurityDistance, 16
  - land, 16
  - lastCmd, 19
  - linearSpeed, 19
  - maxAngularSpeed, 19
  - maxLinearSpeed, 19
  - move\_msg, 19
  - moveBackward, 16
  - moveDown, 17
  - moveForward, 17
  - moveLeft, 17
  - moveRight, 17
  - moveUp, 17
  - navData, 19

- nh, [19](#)
- Nymeria, [14](#)
- obstaclePossible, [17](#)
- pub\_cmd\_land, [19](#)
- pub\_cmd\_move, [19](#)
- pub\_cmd\_reset, [19](#)
- pub\_cmd\_takeoff, [19](#)
- reactionRoutine, [17](#)
- safeActions, [19](#)
- setLinearSpeed, [17](#)
- setMaxAngularSpeed, [17](#)
- setMaxLinearSpeed, [18](#)
- setSecurityDist, [18](#)
- slowDown, [18](#)
- stop, [18](#)
- sub\_navdata, [19](#)
- takeOff, [18](#)
- triggerAction, [18](#)
- turnLeft, [18](#)
- turnRight, [18](#)
- underSecurityDist, [18](#)
- validateStates, [19](#)
- nymeria\_ar drone/README.md, [42](#)
- nymeria\_ar drone/include/nymeria\_ar drone/Nymeria.h, [39](#)
- nymeria\_ar drone/include/nymeria\_ar drone/Nymeria↔CheckObstacle.h, [39](#)
- nymeria\_ar drone/include/nymeria\_ar drone/Nymeria↔Constants.h, [41](#)
- nymeria\_ar drone/include/nymeria\_ar drone/Nymeria↔Exceptions.h, [41](#)
- nymeria\_ar drone/include/nymeria\_ar drone/Nymeria↔InvalidSecurityDistance.h, [41](#)
- nymeria\_ar drone/include/nymeria\_ar drone/Nymeria↔Mutex.h, [41](#)
- nymeria\_ar drone/include/nymeria\_ar drone/Nymeria↔MutexCommand.h, [41](#)
- nymeria\_ar drone/include/nymeria\_ar drone/Nymeria↔MutexObstacle.h, [42](#)
- nymeria\_ar drone/include/nymeria\_ar drone/Nymeria↔MutexSecurityDistance.h, [42](#)
- nymeria\_ar drone/include/nymeria\_ar drone/Nymeria↔ParamExc.h, [42](#)
- nymeria\_ar drone/src/Nymeria.cpp, [43](#)
- nymeria\_ar drone/src/NymeriaCheckObstacle.cpp, [43](#)
- nymeria\_ar drone/src/NymeriaConstants.cpp, [44](#)
- nymeria\_ar drone/src/NymeriaMutex.cpp, [44](#)
- nymeria\_ar drone/src/NymeriaMutexCommand.cpp, [44](#)
- nymeria\_ar drone/src/NymeriaMutexObstacle.cpp, [44](#)
- nymeria\_ar drone/src/NymeriaMutexSecurityDistance.↔cpp, [44](#)
- nymeria\_ar drone/src/exception/NymeriaExceptions.↔cpp, [42](#)
- nymeria\_ar drone/src/exception/NymeriaInvalid↔SecurityDistance.cpp, [43](#)
- nymeria\_ar drone/src/exception/NymeriaParamExc.cpp, [43](#)
- nymeria\_ar drone/test/NymeriaTest.cpp, [45](#)
- NymeriaCheckObstacle, [20](#)
  - error, [23](#)
  - getSecurityDist, [21](#)
  - getSensorMaxRange, [21](#)
  - inputCurFrontDist, [21](#)
  - nh, [23](#)
  - NymeriaCheckObstacle, [21](#)
  - PID, [22](#)
  - pilotage, [22](#)
  - rebouclage, [22](#)
  - regulation, [22](#)
  - saturationCommande, [23](#)
  - saturationPente, [23](#)
  - sensorMaxRange, [23](#)
  - setSecurityDist, [23](#)
  - setSensorMaxRange, [23](#)
  - sub\_navdata, [23](#)
- NymeriaCheckObstacle.cpp
  - droneState, [44](#)
  - pitch, [44](#)
  - stateDroneCallback, [43](#)
- NymeriaCheckObstacle.h
  - stateDroneCallback, [40](#)
- NymeriaConstants, [24](#)
  - ANTICIPATING\_OBSTACLE\_DISTANCE, [25](#)
  - D\_A\_SPEED, [25](#)
  - D\_L\_SPEED, [25](#)
  - D\_M\_A\_SPEED, [25](#)
  - D\_M\_L\_SPEED, [25](#)
  - E\_PARAM, [25](#)
  - E\_STOP, [25](#)
  - I\_A\_SPEED, [25](#)
  - I\_L\_SPEED, [25](#)
  - I\_M\_A\_SPEED, [25](#)
  - I\_M\_L\_SPEED, [25](#)
  - INIT, [25](#)
  - LAND, [25](#)
  - M\_BACKWARD, [25](#)
  - M\_DOWN, [25](#)
  - M\_FORWARD, [25](#)
  - M\_LEFT, [26](#)
  - M\_RIGHT, [26](#)
  - M\_UP, [26](#)
  - NymeriaConstants, [25](#)
  - O\_FRONT, [26](#)
  - SLOW\_DOWN, [26](#)
  - STOP, [26](#)
  - T\_LEFT, [26](#)
  - T\_RIGHT, [26](#)
  - TAKEOFF, [26](#)
- NymeriaExceptions, [26](#)
  - ~NymeriaExceptions, [27](#)
  - errMsg, [27](#)
  - NymeriaExceptions, [27](#)
  - what, [27](#)
- NymeriaInvalidSecurityDistance, [27](#)
  - ~NymeriaInvalidSecurityDistance, [28](#)
  - NymeriaInvalidSecurityDistance, [28](#)

- what, 28
- NymeriaMutex, 28
  - NymeriaMutex, 29
- NymeriaMutexCommand, 29
  - ~NymeriaMutexCommand, 30
  - getInstance, 30
  - instanceFlag, 31
  - lock, 30
  - locked, 31
  - mutexDrone, 31
  - NymeriaMutexCommand, 30
  - unlock, 30
- NymeriaMutexObstacle, 31
  - ~NymeriaMutexObstacle, 32
  - getInstance, 32
  - instanceFlag, 33
  - lock, 32
  - locked, 33
  - mutexObstacle, 33
  - NymeriaMutexObstacle, 32
  - unlock, 32
- NymeriaMutexSecurityDistance, 33
  - ~NymeriaMutexSecurityDistance, 34
  - getInstance, 34
  - instanceFlag, 35
  - lock, 34
  - locked, 35
  - mutexSecDist, 35
  - NymeriaMutexSecurityDistance, 34
  - unlock, 34
- NymeriaParamExc, 35
  - ~NymeriaParamExc, 36
  - NymeriaParamExc, 36
  - what, 36
- NymeriaTest, 36
  - getNH, 36
  - loop, 36
  - nh, 37
  - NymeriaTest, 36
- NymeriaTest.cpp
  - main, 45
- O\_FRONT
  - NymeriaConstants, 26
- obstaclePossible
  - Nymeria, 17
- PID
  - NymeriaCheckObstacle, 22
- pilotage
  - NymeriaCheckObstacle, 22
- pitch
  - NymeriaCheckObstacle.cpp, 44
- pub\_cmd\_land
  - Nymeria, 19
- pub\_cmd\_move
  - Nymeria, 19
- pub\_cmd\_reset
  - Nymeria, 19
- pub\_cmd\_takeoff
  - Nymeria, 19
- reactionRoutine
  - Nymeria, 17
- rebouclage
  - NymeriaCheckObstacle, 22
- regulation
  - NymeriaCheckObstacle, 22
- SLOW\_DOWN
  - NymeriaConstants, 26
- STOP
  - NymeriaConstants, 26
- safeActions
  - Nymeria, 19
- saturationCommande
  - NymeriaCheckObstacle, 23
- saturationPente
  - NymeriaCheckObstacle, 23
- sensorMaxRange
  - NymeriaCheckObstacle, 23
- setLinearSpeed
  - Nymeria, 17
- setMaxAngularSpeed
  - Nymeria, 17
- setMaxLinearSpeed
  - Nymeria, 18
- setSecurityDist
  - Nymeria, 18
  - NymeriaCheckObstacle, 23
- setSensorMaxRange
  - NymeriaCheckObstacle, 23
- slowDown
  - Nymeria, 18
- stateDroneCallback
  - NymeriaCheckObstacle.cpp, 43
  - NymeriaCheckObstacle.h, 40
- stop
  - Nymeria, 18
- sub\_navdata
  - Nymeria, 19
  - NymeriaCheckObstacle, 23
- T\_LEFT
  - NymeriaConstants, 26
- T\_RIGHT
  - NymeriaConstants, 26
- TAKEOFF
  - NymeriaConstants, 26
- takeOff
  - Nymeria, 18
- triggerAction
  - Nymeria, 18
- turnLeft
  - Nymeria, 18
- turnRight
  - Nymeria, 18

underSecurityDist  
    Nymeria, [18](#)  
unlock  
    NymeriaMutexCommand, [30](#)  
    NymeriaMutexObstacle, [32](#)  
    NymeriaMutexSecurityDistance, [34](#)  
  
validateStates  
    Nymeria, [19](#)  
  
what  
    NymeriaExceptions, [27](#)  
    NymeriaInvalidSecurityDistance, [28](#)  
    NymeriaParamExc, [36](#)