nymeria_ardrone

Generated by Doxygen 1.8.9.1

Mon Jan 19 2015 11:19:25

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/jdufant/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

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.

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::getSecurityDist ( )

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

Returns

security distance.

5.1.3.11 void Nymeria::increaseAngularSpeed ( )

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

5.1.3.12 void Nymeria::increaseLinearSpeed ( )

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

5.1.3.13 void Nymeria::increaseMaxAngularSpeed ( )

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

5.1.3.14 void Nymeria::increaseMaxLinearSpeed ( )

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

5.1.3.15 void Nymeria::land ( )

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

5.1.3.16 void Nymeria::moveBackward ( )

Command in order to move drone backward.

5.1.3.17 void Nymeria::moveDown ( )

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

decrease altitude.

5.1.3.18 void Nymeria::moveForward ( )

Command in order to move drone forward.

5.1.3.19 void Nymeria::moveLeft ( )

Command in order to make drone rotate to the left.

5.1.3.20 void Nymeria::moveRight ( )

Command in order to make drone rotate to the right.

5.1.3.21 void Nymeria::moveUp ( )

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

increase altitude.

5.1.3.22 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.

5.1.3.23 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.

5.1.3.24 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.

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

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

Parameters

secDist	security distance.

5.1.3.26 void Nymeria::stop ( )

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

stay at current position.

5.1.3.27 void Nymeria::takeOff ( )

Command in order to make the drone take off.

5.1.3.28 void Nymeria::turnLeft ( )

Command in order to move drone to the left.

5.1.3.29 void Nymeria::turnRight ( )

Command in order to move drone to the right.

5.1.4 Friends And Related Function Documentation

5.1.4.1 friend class Controller `[friend]`

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

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

5.2 NymeriaCheckObstacle Class Reference

`#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.

5.2.1 Constructor & Destructor Documentation

5.2.1.1 NymeriaCheckObstacle::NymeriaCheckObstacle ( )

Default constructor.

5.2.1.2 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

5.2.2 Member Function Documentation

5.2.2.1 double NymeriaCheckObstacle::getSecurityDist ( )

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

Returns

security distance.

5.2.2.2 double NymeriaCheckObstacle::getSensorMaxRange ( )

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

Returns

sensor max range.

5.2.2.3 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

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

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

Parameters

secDist	security distance.

5.2.2.5 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.

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

- include/nymeria_ardrone/NymeriaCheckObstacle.h
- 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 ()
  Definition 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.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 NymeriaConstants::NymeriaConstants ( )

Definition of the class 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:

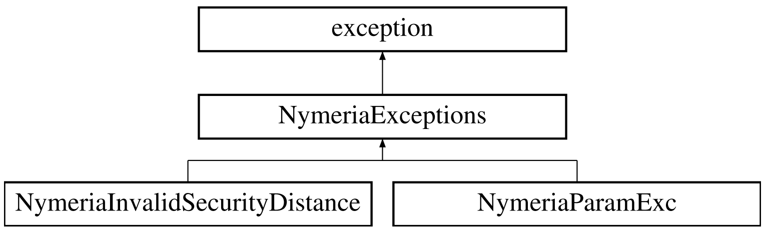
- include/nymeria_ardrone/NymeriaConstants.h
- src/NymeriaConstants.cpp

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)

  Definition of the class NymeriaExceptions, that defines the base class for all exceptions particular to Nymeria.
- virtual ∼NymeriaExceptions (void) throw ()
- virtual const char ∗ what () const throw ()

5.4.1 Detailed Description

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

5.4.2 Constructor & Destructor Documentation

5.4.2.1 NymeriaExceptions::NymeriaExceptions (string *msg*)

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

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]

Reimplemented in NymeriaInvalidSecurityDistance, and NymeriaParamExc.

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

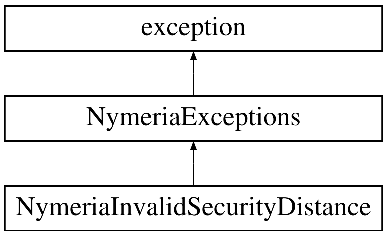
- include/nymeria_ardrone/NymeriaExceptions.h
- src/exception/NymeriaExceptions.cpp

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 ()

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]

Reimplemented from NymeriaExceptions.

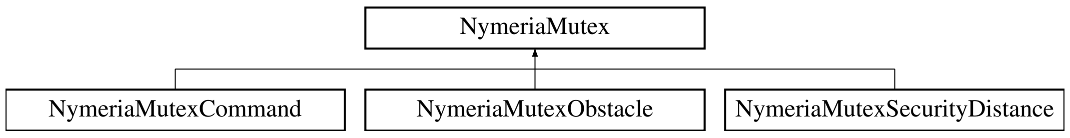
The documentation for this class was generated from the following files:

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

5.6 NymeriaMutex Class Reference

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

Inheritance diagram for NymeriaMutex:



Public Member Functions

- NymeriaMutex ()

5.6.1 Constructor & Destructor Documentation

5.6.1.1 NymeriaMutex::NymeriaMutex ( )

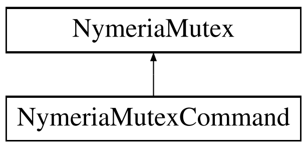
The documentation for this class was generated from the following files:

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

5.7 NymeriaMutexCommand Class Reference

```
#include <NymeriaMutexCommand.h>
```

Inheritance diagram for NymeriaMutexCommand:



Public Member Functions

- ∼NymeriaMutexCommand ()

Static Public Member Functions

- static NymeriaMutexCommand ∗ getInstance ()
- static void lock ()
- static void unlock ()

5.7.1 Constructor & Destructor Documentation

5.7.1.1 NymeriaMutexCommand::∼NymeriaMutexCommand ( )

5.7.2 Member Function Documentation

5.7.2.1 NymeriaMutexCommand ∗ NymeriaMutexCommand::getInstance ( ) [static]

5.7.2.2 void NymeriaMutexCommand::lock ( ) [static]

5.7.2.3 void NymeriaMutexCommand::unlock ( ) [static]

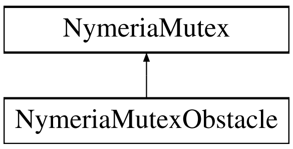
The documentation for this class was generated from the following files:

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

5.8 NymeriaMutexObstacle Class Reference

`#include <NymeriaMutexObstacle.h>`

Inheritance diagram for NymeriaMutexObstacle:



Public Member Functions

- ∼NymeriaMutexObstacle ()

Static Public Member Functions

- static NymeriaMutexObstacle ∗ getInstance ()
- static void lock ()
- static void unlock ()

5.8.1 Constructor & Destructor Documentation

5.8.1.1 NymeriaMutexObstacle::∼NymeriaMutexObstacle ( )

5.8.2 Member Function Documentation

5.8.2.1 NymeriaMutexObstacle ∗ NymeriaMutexObstacle::getInstance ( ) [static]

5.8.2.2 void NymeriaMutexObstacle::lock ( ) [static]

5.8.2.3 void NymeriaMutexObstacle::unlock ( ) [static]

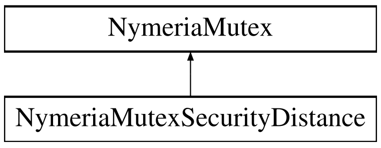
The documentation for this class was generated from the following files:

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

5.9 NymeriaMutexSecurityDistance Class Reference

```
#include <NymeriaMutexSecurityDistance.h>
```

Inheritance diagram for NymeriaMutexSecurityDistance:



Public Member Functions

- ∼NymeriaMutexSecurityDistance ()

Static Public Member Functions

- static NymeriaMutexSecurityDistance ∗ getInstance ()
- static void lock ()
- static void unlock ()

5.9.1 Constructor & Destructor Documentation

5.9.1.1 NymeriaMutexSecurityDistance::∼NymeriaMutexSecurityDistance ( )

5.9.2 Member Function Documentation

5.9.2.1 NymeriaMutexSecurityDistance ∗ NymeriaMutexSecurityDistance::getInstance ( ) `[static]`

5.9.2.2 void NymeriaMutexSecurityDistance::lock ( ) `[static]`

5.9.2.3 void NymeriaMutexSecurityDistance::unlock ( ) `[static]`

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

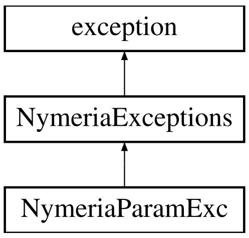
- include/nymeria_ardrone/NymeriaMutexSecurityDistance.h
- src/NymeriaMutexSecurityDistance.cpp

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 ()
- virtual const char ∗ what () const throw ()

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.

5.10.2.2 NymeriaParamExc::∼NymeriaParamExc (void) throw) `[virtual]`

5.10.3 Member Function Documentation

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

Reimplemented from NymeriaExceptions.

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

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

Chapter 6

File Documentation

6.1 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 include/nymeria_ardrone/NymeriaCheckObstacle.h File Reference

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

Classes

- class NymeriaCheckObstacle

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 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 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 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 include/nymeria_ardrone/NymeriaMutex.h File Reference

Classes

- class NymeriaMutex

6.7 include/nymeria_ardrone/NymeriaMutexCommand.h File Reference

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

Classes

- class NymeriaMutexCommand

6.8 include/nymeria_ardrone/NymeriaMutexObstacle.h File Reference

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

Classes

- class NymeriaMutexObstacle

6.9 include/nymeria_ardrone/NymeriaMutexSecurityDistance.h File Reference

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

Classes

- class NymeriaMutexSecurityDistance

6.10 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 README.md File Reference

6.12 src/exception/NymeriaExceptions.cpp File Reference

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

6.13 src/exception/NymeriaInvalidSecurityDistance.cpp File Reference

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

6.14 src/exception/NymeriaParamExc.cpp File Reference

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

6.15 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 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 src/NymeriaConstants.cpp File Reference

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

6.18 src/NymeriaMutex.cpp File Reference

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

6.19 src/NymeriaMutexCommand.cpp File Reference

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

6.20 src/NymeriaMutexObstacle.cpp File Reference

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

6.21 src/NymeriaMutexSecurityDistance.cpp File Reference

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

Index