UAV common

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 AeroCoefficients Struct Reference

Aerodynamic coefficient.

`#include <aero_coefficients.hpp>`

Public Attributes

- double S
- double d
- double eAR
- Eigen::Vector< double, 6 > C0
- Eigen::Matrix< double, 6, 3 > Cpqr
- Eigen::Matrix< double, 6, 4 > Cab
- double stallLimit

4.1.1 Detailed Description

Aerodynamic coefficient.

4.1.2 Member Data Documentation

4.1.2.1 C0

`Eigen::Vector<double,6> AeroCoefficients::C0`

4.1.2.2 Cab

```
Eigen::Matrix<double,6,4> AeroCoefficients::Cab
```

4.1.2.3 Cpqr

```
Eigen::Matrix<double,6,3> AeroCoefficients::Cpqr
```

4.1.2.4 d

```
double AeroCoefficients::d
```

4.1.2.5 eAR

```
double AeroCoefficients::eAR
```

4.1.2.6 S

```
double AeroCoefficients::S
```

4.1.2.7 stallLimit

```
double AeroCoefficients::stallLimit
```

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

- src/components/aero_coefficients.hpp

4.2 AHRSParams Struct Reference

AHRS parameters.

```
#include <navi.hpp>
```

Public Attributes

- std::string type
- double alpha
- double Q
- double R

4.2.1 Detailed Description

AHRS parameters.

4.2.2 Member Data Documentation

4.2.2.1 alpha

```
double AHRSParams::alpha
```

4.2.2.2 Q

```
double AHRSParams::Q
```

4.2.2.3 R

```
double AHRSParams::R
```

4.2.2.4 type

```
std::string AHRSParams::type
```

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

- src/components/navi.hpp

4.3 Ammo Class Reference

`#include <loads.hpp>`

Inheritance diagram for Ammo:

Collaboration diagram for Ammo:

Public Member Functions

- Ammo ()=default
- Ammo (int ammount, double reload, Eigen::Vector3d offset, double mass, Eigen::Vector3d V0)
- Ammo & operator= (const Ammo &other)
- Eigen::Vector3d getV0 ()

  get start velocity of ammo when launched

Protected Attributes

- Eigen::Vector3d _V0

Additional Inherited Members

4.3.1 Constructor & Destructor Documentation

4.3.1.1 Ammo() [1/2]

```
Ammo::Ammo ( )  [default]
```

4.3.1.2 Ammo() [2/2]

```
Ammo::Ammo (
            int ammount,
            double reload,
            Eigen::Vector3d offset,
            double mass,
            Eigen::Vector3d V0 )
```

4.3.2 Member Function Documentation

4.3.2.1 getV0()

```
Eigen::Vector3d Ammo::getV0 ( )  [inline]
```

get start velocity of ammo when launched

Returns

start velocity vector

4.3.2.2 operator=()

```
Ammo & Ammo::operator= (
            const Ammo & other )
```

4.3.3 Member Data Documentation

4.3.3.1 _V0

```
Eigen::Vector3d Ammo::_V0  [protected]
```

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

- src/components/loads.hpp
- src/components/loads.cpp

4.4 Cargo Class Reference

```
#include <loads.hpp>
```

Inheritance diagram for Cargo:

Collaboration diagram for Cargo:

Public Member Functions

- Cargo ()=default
- Cargo (int ammount, double reload, Eigen::Vector3d offset, double mass)

Additional Inherited Members

4.4.1 Constructor & Destructor Documentation

4.4.1.1 Cargo() [1/2]

```
Cargo::Cargo ( )  [default]
```

4.4.1.2 Cargo() [2/2]

```
Cargo::Cargo (
            int ammount,
            double reload,
            Eigen::Vector3d offset,
            double mass )
```

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

- src/components/loads.hpp
- src/components/loads.cpp

4.5 ControlSurfaces Class Reference

Aircraft's control surfaces.

```
#include <control_surfaces.hpp>
```

Public Member Functions

- ControlSurfaces ()
- ControlSurfaces (int noOfSurfaces, Eigen::Matrix< double, 6,-1 > matrix, Eigen::VectorXd min, Eigen::VectorXd max, Eigen::VectorXd trim)

  Constructor.
- Eigen::Vector< double, 6 > getCoefficients () const
- bool setValues (Eigen::VectorXd new_values)
- void restoreTrim ()
- int getNoOfSurface () const
- Eigen::VectorXd getValues () const

4.5.1 Detailed Description

Aircraft's control surfaces.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 ControlSurfaces() [1/2]

```
ControlSurfaces::ControlSurfaces ( )
```

4.5.2.2 ControlSurfaces() [2/2]

```
ControlSurfaces::ControlSurfaces (
            int noOfSurfaces,
            Eigen::Matrix< double, 6,-1 > matrix,
            Eigen::VectorXd min,
            Eigen::VectorXd max,
            Eigen::VectorXd trim )
```

Constructor.

Parameters

noOfSurfaces	number of independent surfaces
matrix	coefficients matrix
min	vector of min angles
max	vector of max angles
trim	vector of trim angles

4.5.3 Member Function Documentation

4.5.3.1 getCoefficients()

```
Eigen::Vector< double, 6 > ControlSurfaces::getCoefficients ( ) const
```

4.5.3.2 getNoOfSurface()

```
int ControlSurfaces::getNoOfSurface ( ) const  [inline]
```

4.5.3.3 getValues()

```
Eigen::VectorXd ControlSurfaces::getValues ( ) const  [inline]
```

4.5.3.4 restoreTrim()

```
void ControlSurfaces::restoreTrim ( )
```

4.5.3.5 setValues()

```
bool ControlSurfaces::setValues (
            Eigen::VectorXd new_values )
```

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

- src/components/control_surfaces.hpp
- src/components/control_surfaces.cpp

4.6 Drive Struct Reference

Drive propelling aircraft.

```
#include <drive.hpp>
```

Inheritance diagram for Drive:

Collaboration diagram for Drive:

Public Attributes

- Eigen::Vector3d position
- Eigen::Vector3d axis
- int noOfHinges
- Hinge hinges [2]

4.6.1 Detailed Description

Drive propelling aircraft.

4.6.2 Member Data Documentation

4.6.2.1 axis

```
Eigen::Vector3d Drive::axis
```

4.6.2.2 hinges

```
Hinge Drive::hinges[2]
```

4.6.2.3 noOfHinges

```
int Drive::noOfHinges
```

4.6.2.4 position

```
Eigen::Vector3d Drive::position
```

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

- src/components/drive.hpp

4.7 EKFScalers Struct Reference

Scalers for EKF.

```
#include <navi.hpp>
```

Public Attributes

- double predictScaler
- double updateScaler
- double baroScaler
- double zScaler

4.7.1 Detailed Description

Scalers for EKF.

4.7.2 Member Data Documentation

4.7.2.1 baroScaler

```
double EKFScalers::baroScaler
```

4.7.2.2 predictScaler

```
double EKFScalers::predictScaler
```

4.7.2.3 updateScaler

```
double EKFScalers::updateScaler
```

4.7.2.4 zScaler

```
double EKFScalers::zScaler
```

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

- src/components/navi.hpp

4.8 Hinge Class Reference

Hinge connecting aircraft with drives.

```
#include <hinge.hpp>
```

Public Member Functions

- Hinge ()=default
- Hinge (Eigen::Vector3d axis, double max, double min, double trim)
- Hinge (const Hinge &old)
- Hinge & operator= (const Hinge &old)
- void updateValue (double newValue)

  set new angle on hinge
- const Eigen::Matrix3d getRot ()

  Get rotattion matrix of orientation change due to hinge.

4.8.1 Detailed Description

Hinge connecting aircraft with drives.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Hinge() [1/3]

```
Hinge::Hinge ( )  [default]
```

4.8.2.2 Hinge() [2/3]

```
Hinge::Hinge (
            Eigen::Vector3d axis,
            double max,
            double min,
            double trim )
```

4.8.2.3 Hinge() [3/3]

```
Hinge::Hinge (
            const Hinge & old )
```

4.8.3 Member Function Documentation

4.8.3.1 getRot()

```
const Eigen::Matrix3d Hinge::getRot ( )
```

Get rotattion matrix of orientation change due to hinge.

Returns

rotation matrix

4.8.3.2 operator=()

```
Hinge & Hinge::operator= (
            const Hinge & old )
```

4.8.3.3 updateValue()

```
void Hinge::updateValue (
            double newValue )
```

set new angle on hinge

Parameters

newValue	new angle of hinge

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

- src/components/hinge.hpp
- src/components/hinge.cpp

4.9 Jet Class Reference

Jet rocket engine.

`#include <drive.hpp>`

Inheritance diagram for Jet:

Collaboration diagram for Jet:

Public Member Functions

- bool start (double time)

  start jet engine
- double getThrust (double time)

  get thrust in specific time
- double getLastThrust ()

  get last calculated thrust

Public Attributes

- int phases
- Eigen::VectorXd thrust
- Eigen::VectorXd time

4.9.1 Detailed Description

Jet rocket engine.

4.9.2 Member Function Documentation

4.9.2.1 getLastThrust()

```
double Jet::getLastThrust ( )  [inline]
```

get last calculated thrust

Returns

last calculated thrust

4.9.2.2 getThrust()

```
double Jet::getThrust (
            double time )
```

get thrust in specific time

Parameters

time	timestamp

Returns

thrust value in Newtons

4.9.2.3 start()

```
bool Jet::start (
            double time )
```

start jet engine

Parameters

time	timestamp of start

Returns

true if start succesful, false if already started

4.9.3 Member Data Documentation

4.9.3.1 phases

```
int Jet::phases
```

4.9.3.2 thrust

```
Eigen::VectorXd Jet::thrust
```

4.9.3.3 time

```
Eigen::VectorXd Jet::time
```

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

- src/components/drive.hpp
- src/components/drive.cpp

4.10 Load Class Reference

Load of aircraft that can be droped or launched.

```
#include <loads.hpp>
```

Inheritance diagram for Load:

Public Member Functions

- double getMass ()
  get mass of load
- Eigen::Vector3d getOffset ()
  get offset of load
- int release (double time)
  Try to release load.

Protected Member Functions

- Load ()=default
- Load (int ammount, double reload, Eigen::Vector3d offset, double mass)
- Load & operator= (const Load &other)

4.10.1 Detailed Description

Load of aircraft that can be droped or launched.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Load() [1/2]

```
Load::Load ( )  [protected], [default]
```

4.10.2.2 Load() [2/2]

```
Load::Load (
            int ammount,
            double reload,
            Eigen::Vector3d offset,
            double mass )  [protected]
```

4.10.3 Member Function Documentation

4.10.3.1 getMass()

```
double Load::getMass ( )  [inline]
```

get mass of load

Returns

mass

4.10.3.2 getOffset()

```
Eigen::Vector3d Load::getOffset ( )  [inline]
```

get offset of load

Returns

offset vector

4.10.3.3 operator=()

```
Load & Load::operator= (
            const Load & other )  [protected]
```

4.10.3.4 release()

```
int Load::release (
            double time )
```

Try to release load.

Parameters

time	

Returns

leftover ammount of loads. Return -1 if load is not ready and -2 if out of load

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

- src/components/loads.hpp
- src/components/loads.cpp

4.11 Logger Class Reference

Log vector data with timestamp in file.

```
#include <logger.hpp>
```

Public Member Functions

- Logger (std::string path, std::string fmt="", uint8_t group=0)
  Constructor.
- ~Logger ()
  Deconstructor.
- void setFmt (std::string fmt)
  Set new format if was not known in constructor.
- void log (double time, std::initializer_list< Eigen::VectorXd > args)
  Log one row.
- void log (double time, std::initializer_list< double > args)
  Log one row.

Static Public Member Functions

- static void setLogDirectory (std::string subdirectory)

  Set global path that log should be created at. Path will be added to relative path of specific log instance.

4.11.1 Detailed Description

Log vector data with timestamp in file.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 Logger()

```
Logger::Logger (
            std::string path,
            std::string fmt = "",
            uint8_t group = 0 )
```

Constructor.

Parameters

path	relative path with log file name.
fmt	format - information about log structure. First line in log file
group	log group - log will be created only if group is in actual LOGGER_MASK

4.11.2.2 ~Logger()

```
Logger::~Logger ( )
```

Deconstructor.

4.11.3 Member Function Documentation

4.11.3.1 log() [1/2]

```
void Logger::log (
            double time,
            std::initializer_list< double > args )
```

Log one row.

Parameters

time	timestamp
args	list of doubles

4.11.3.2 log() [2/2]

```
void Logger::log (
            double time,
            std::initializer_list< Eigen::VectorXd > args )
```

Log one row.

Parameters

time	timestamp
args	list of double vectors

4.11.3.3 setFmt()

```
void Logger::setFmt (
            std::string fmt )
```

Set new format if was not known in constructor.

Parameters

fmt	new format

4.11.3.4 setLogDirectory()

```
void Logger::setLogDirectory (
            std::string subdirectory )  [static]
```

Set global path that log should be created at. Path will be added to relative path of specific log instance.

Parameters

subdirectory	new global log path

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

- src/logger/logger.hpp
- src/logger/logger.cpp

4.12 ODE Class Reference

Ordinal differencial equation solver.

`#include <ode.hpp>`

Inheritance diagram for ODE:

Public Types

- enum ODEMethod { Euler , Heun , RK4 , NONE }

  Supported solving method.

Public Member Functions

- ODE (int micro_steps)

  Constructor.
- virtual ~ODE ()

  Virtual deconstructor.
- virtual Eigen::VectorXd step (double t, Eigen::VectorXd y0, std::function< Eigen::VectorXd(double, Eigen::↩ VectorXd)> rhs_fun, double h)=0

  One step of explicit solving algorithm.
- int getMicrosteps () const

  Return microsteps - number of rhs function calls to calculate on step.

Static Public Member Functions

- static ODEMethod fromString (std::string str)

  Parse solving method from string.
- static std::unique_ptr< ODE > factory (ODEMethod method)

  Factory constructing ODE solvers.
- static int getMicrosteps (ODEMethod method)

  Get microsteps of given method.

4.12.1 Detailed Description

Ordinal differencial equation solver.

4.12.2 Member Enumeration Documentation

4.12.2.1 ODEMethod

`enum ODE::ODEMethod`

Supported solving method.

Enumerator

Euler	
Heun	
RK4	
NONE	

4.12.3 Constructor & Destructor Documentation

4.12.3.1 ODE()

```
ODE::ODE (
            int micro_steps )
```

Constructor.

4.12.3.2 ~ODE()

```
virtual ODE::~ODE ( )  [inline], [virtual]
```

Virtual deconstructor.

4.12.4 Member Function Documentation

4.12.4.1 factory()

```
std::unique_ptr< ODE > ODE::factory (
            ODEMethod method )  [static]
```

Factory constructing ODE solvers.

Parameters

method	type of desired method

Returns

instance of ODE solver

4.12.4.2 fromString()

```
ODE::ODEMethod ODE::fromString (
            std::string str )  [static]
```

Parse solving method from string.

Parameters

str	input string

Returns

solving method if parsed, NONE if unknown

4.12.4.3 getMicrosteps() [1/2]

```
int ODE::getMicrosteps ( ) const
```

Return microsteps - number of rhs function calls to calculate on step.

Returns

microsteps

4.12.4.4 getMicrosteps() [2/2]

```
int ODE::getMicrosteps (
            ODEMethod method )  [static]
```

Get microsteps of given method.

Parameters

method	method type

Returns

number of microstep in one algoritm step

4.12.4.5 step()

```
virtual Eigen::VectorXd ODE::step (
            double t,
            Eigen::VectorXd y0,
            std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun,
            double h )  [pure virtual]
```

One step of explicit solving algorithm.

Parameters

t	start time
y0	start variable
rhs_fun	right-hand-side function, calculation of derivative
h	time step

Returns

Implemented in ODE_RK4, ODE_Heun, and ODE_Euler.

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

- src/ode/ode.hpp
- src/ode/ode.cpp

4.13 ODE_Euler Class Reference

Explicit Euler algorithm.

`#include <ode_impl.hpp>`

Inheritance diagram for ODE_Euler:

Collaboration diagram for ODE_Euler:

Public Member Functions

- ODE_Euler ()
- Eigen::VectorXd step (double t, Eigen::VectorXd y0, std::function< Eigen::VectorXd(double, Eigen::Vector↩ Xd)> rhs_fun, double h) override

  One step of explicit solving algorithm.

Additional Inherited Members

4.13.1 Detailed Description

Explicit Euler algorithm.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 ODE_Euler()

```
ODE_Euler::ODE_Euler ( )  [inline]
```

4.13.3 Member Function Documentation

4.13.3.1 step()

```
Eigen::VectorXd ODE_Euler::step (
            double t,
            Eigen::VectorXd y0,
            std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun,
            double h )  [inline], [override], [virtual]
```

One step of explicit solving algorithm.

Parameters

t	start time
y0	start variable
rhs_fun	right-hand-side function, calculation of derivative
h	time step

Returns

Implements ODE.

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

- src/ode/ode_impl.hpp

4.14 ODE_Heun Class Reference

Second order explicit Heun algorithm.

`#include <ode_impl.hpp>`

Inheritance diagram for ODE_Heun:

Collaboration diagram for ODE_Heun:

Public Member Functions

- ODE_Heun ()
- Eigen::VectorXd step (double t, Eigen::VectorXd y0, std::function< Eigen::VectorXd(double, Eigen::Vector↩ Xd)> rhs_fun, double h) override

  One step of explicit solving algorithm.

Additional Inherited Members

4.14.1 Detailed Description

Second order explicit Heun algorithm.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 ODE_Heun()

```
ODE_Heun::ODE_Heun ( )  [inline]
```

4.14.3 Member Function Documentation

4.14.3.1 step()

```
Eigen::VectorXd ODE_Heun::step (
            double t,
            Eigen::VectorXd y0,
            std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun,
            double h )  [inline], [override], [virtual]
```

One step of explicit solving algorithm.

Parameters

t	start time
y0	start variable
rhs_fun	right-hand-side function, calculation of derivative
h	time step

Returns

Implements ODE.

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

- src/ode/ode_impl.hpp

4.15 ODE_RK4 Class Reference

Fourth order Runge Kutta algorithm.

`#include <ode_impl.hpp>`

Inheritance diagram for ODE_RK4:

Collaboration diagram for ODE_RK4:

Public Member Functions

- ODE_RK4 ()
- Eigen::VectorXd step (double t, Eigen::VectorXd y0, std::function< Eigen::VectorXd(double, Eigen::Vector↩ Xd)> rhs_fun, double h) override

  One step of explicit solving algorithm.

Additional Inherited Members

4.15.1 Detailed Description

Fourth order Runge Kutta algorithm.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 ODE_RK4()

```
ODE_RK4::ODE_RK4 ( )  [inline]
```

4.15.3 Member Function Documentation

4.15.3.1 step()

```
Eigen::VectorXd ODE_RK4::step (
            double t,
            Eigen::VectorXd y0,
            std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun,
            double h )  [inline], [override], [virtual]
```

One step of explicit solving algorithm.

Parameters

t	start time
y0	start variable
rhs_fun	right-hand-side function, calculation of derivative
h	time step

Returns

Implements ODE.

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

- src/ode/ode_impl.hpp

4.16 ODETest Class Reference

Inheritance diagram for ODETest:

Collaboration diagram for ODETest:

Protected Member Functions

- void SetUp () override
- void TearDown () override

4.16.1 Member Function Documentation

4.16.1.1 SetUp()

```
void ODETest::SetUp ( )  [inline], [override], [protected]
```

4.16.1.2 TearDown()

```
void ODETest::TearDown ( )  [inline], [override], [protected]
```

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

- src/ode/ode_test.cpp

4.17 PID Class Reference

PID discrete controller.

`#include <PID.hpp>`

Public Member Functions

- PID (double Kp, double Ki, double Kd, double min=std::numeric_limits< double >::min(), double max=std::numeric_limits< double >::max(), AntiWindUpMode antiWindUp=AntiWindUpMode::Clamping)
- ~PID ()
- void set_dt (double dt)
  Set new time step.
- double calc (double error)
  calc output of controller
- double calc (double error, double dt)
  calc output of controller with specific time step
- void clear ()
  clear internal state

4.17.1 Detailed Description

PID discrete controller.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 PID()

```
PID::PID (
            double Kp,
            double Ki,
            double Kd,
            double min = std::numeric_limits<double>::min(),
            double max = std::numeric_limits<double>::max(),
            AntiWindUpMode antiWindUp = AntiWindUpMode::Clamping )
```

4.17.2.2 ~PID()

```
PID::~PID ( )
```

4.17.3 Member Function Documentation

4.17.3.1 calc() [1/2]

```
double PID::calc (
            double error )
```

calc output of controller

Parameters

error	input of controller

Returns

output of controller

4.17.3.2 calc() [2/2]

```
double PID::calc (
            double error,
            double dt )
```

calc output of controller with specific time step

Parameters

error	input of controller
dt	time step

Returns

output of controller

4.17.3.3 clear()

```
void PID::clear ( )
```

clear internal state

4.17.3.4 set_dt()

```
void PID::set_dt (
            double dt )
```

Set new time step.

Parameters

dt	new time step

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

- src/PID/PID.hpp
- src/PID/PID.cpp

4.18 Rotor Struct Reference

Rotor engine with controlled speed.

`#include <drive.hpp>`

Inheritance diagram for Rotor:

Collaboration diagram for Rotor:

Public Attributes

- double forceCoff
- double torqueCoff
- int direction
- double timeConstant
- double maxSpeed
- double hoverSpeed

4.18.1 Detailed Description

Rotor engine with controlled speed.

4.18.2 Member Data Documentation

4.18.2.1 direction

```
int Rotor::direction
```

4.18.2.2 forceCoff

```
double Rotor::forceCoff
```

4.18.2.3 hoverSpeed

```
double Rotor::hoverSpeed
```

4.18.2.4 maxSpeed

```
double Rotor::maxSpeed
```

4.18.2.5 timeConstant

```
double Rotor::timeConstant
```

4.18.2.6 torqueCoff

```
double Rotor::torqueCoff
```

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

- src/components/drive.hpp

4.19 SensorParams Struct Reference

Base parameters of a sensor.

```
#include <navi.hpp>
```

Public Attributes

- std::string name
- double sd
- Eigen::Vector3d bias
- double refreshTime

4.19.1 Detailed Description

Base parameters of a sensor.

4.19.2 Member Data Documentation

4.19.2.1 bias

```
Eigen::Vector3d SensorParams::bias
```

4.19.2.2 name

```
std::string SensorParams::name
```

4.19.2.3 refreshTime

```
double SensorParams::refreshTime
```

4.19.2.4 sd

```
double SensorParams::sd
```

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

- src/components/navi.hpp

4.20 TimedLoop Class Reference

Simulation of real-time synchronized loop.

```
#include <timed_loop.hpp>
```

Public Member Functions

- TimedLoop (int periodInMs, std::function< void(void)> func, Status &status)
  Constructor.
- void go ()
  start infinite loop
- void go (uint32_t loops)
  start loop for specific cycle numbers

4.20.1 Detailed Description

Simulation of real-time synchronized loop.

4.20.2 Constructor & Destructor Documentation

4.20.2.1 TimedLoop()

```
TimedLoop::TimedLoop (
            int periodInMs,
            std::function< void(void)> func,
            Status & status )
```

Constructor.

Parameters

periodInMs	loop period in milliseconds
func	function that should be called in loop
status	reference to controlling status

4.20.3 Member Function Documentation

4.20.3.1 go() [1/2]

```
void TimedLoop::go ( )
```

start infinite loop

4.20.3.2 go() [2/2]

```
void TimedLoop::go (
            uint32_t loops )
```

start loop for specific cycle numbers

Parameters

loops	how many cycles should be done

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

- src/timed_loop/timed_loop.hpp
- src/timed_loop/timed_loop.cpp

4.21 UAVparams Struct Reference

Parsed UAV configuration from XML.

`#include <uav_params.hpp>`

Collaboration diagram for UAVparams:

Public Member Functions

- UAVparams ()

  Initialize default data.
- ~UAVparams ()
- void loadConfig (std::string configFile)
- Eigen::VectorXd getRotorTimeContants () const
- Eigen::VectorXd getRotorMaxSpeeds () const
- Eigen::VectorXd getRotorHoverSpeeds () const

Static Public Member Functions

- static const UAVparams * getSingleton ()

Public Attributes

- std::string name
- bool instantRun
- std::string initialMode
- Eigen::Vector3d initialPosition
- Eigen::Vector3d initialOrientation
- Eigen::Vector3d initialVelocity
- double m
- double Ix
- double Iy
- double Iz
- double Ixy
- double Ixz
- double Iyz
- int noOfRotors
- std::unique_ptr< Rotor[ ]> rotors
- int noOfJets
- std::unique_ptr< Jet[ ]> jets
- ControlSurfaces surfaces
- AeroCoefficients aero_coffs
- std::map< std::string, PID > pids
- std::vector< SensorParams > sensors
- AHRSParams ahrs
- EKFScalers ekf
- Eigen::MatrixX4d rotorMixer
- Eigen::MatrixX4d surfaceMixer
- int noOfAmmo
- std::unique_ptr< Ammo[ ]> ammo
- int noOfCargo
- std::unique_ptr< Cargo[ ]> cargo

4.21.1 Detailed Description

Parsed UAV configuration from XML.

4.21.2 Constructor & Destructor Documentation

4.21.2.1 UAVparams()

```
UAVparams::UAVparams ( )
```

Initialize default data.

4.21.2.2 ~UAVparams()

```
UAVparams::~UAVparams ( )
```

4.21.3 Member Function Documentation

4.21.3.1 getRotorHoverSpeeds()

```
Eigen::VectorXd UAVparams::getRotorHoverSpeeds ( ) const
```

4.21.3.2 getRotorMaxSpeeds()

```
Eigen::VectorXd UAVparams::getRotorMaxSpeeds ( ) const
```

4.21.3.3 getRotorTimeContants()

```
Eigen::VectorXd UAVparams::getRotorTimeContants ( ) const
```

4.21.3.4 getSingleton()

```
const UAVparams * UAVparams::getSingleton ( )  [static]
```

4.21.3.5 loadConfig()

```
void UAVparams::loadConfig (
            std::string configFile )
```

4.21.4 Member Data Documentation

4.21.4.1 aero_coffs

```
AeroCoefficients UAVparams::aero_coffs
```

4.21.4.2 ahrs

```
AHRSParams UAVparams::ahrs
```

4.21.4.3 ammo

```
std::unique_ptr<Ammo[]> UAVparams::ammo
```

4.21.4.4 cargo

```
std::unique_ptr<Cargo[]> UAVparams::cargo
```

4.21.4.5 ekf

```
EKFScalers UAVparams::ekf
```

4.21.4.6 initialMode

```
std::string UAVparams::initialMode
```

4.21.4.7 initialOrientation

```
Eigen::Vector3d UAVparams::initialOrientation
```

4.21.4.8 initialPosition

```
Eigen::Vector3d UAVparams::initialPosition
```

4.21.4.9 initialVelocity

```
Eigen::Vector3d UAVparams::initialVelocity
```

4.21.4.10 instantRun

```
bool UAVparams::instantRun
```

4.21.4.11 Ix

```
double UAVparams::Ix
```

4.21.4.12 Ixy

```
double UAVparams::Ixy
```

4.21.4.13 Ixz

```
double UAVparams::Ixz
```

4.21.4.14 Iy

```
double UAVparams::Iy
```

4.21.4.15 Iyz

```
double UAVparams::Iyz
```

4.21.4.16 Iz

```
double UAVparams::Iz
```

4.21.4.17 jets

```
std::unique_ptr<Jet[]> UAVparams::jets
```

4.21.4.18 m

```
double UAVparams::m
```

4.21.4.19 name

```
std::string UAVparams::name
```

4.21.4.20 noOfAmmo

```
int UAVparams::noOfAmmo
```

4.21.4.21 noOfCargo

```
int UAVparams::noOfCargo
```

4.21.4.22 noOfJets

```
int UAVparams::noOfJets
```

4.21.4.23 noOfRotors

```
int UAVparams::noOfRotors
```

4.21.4.24 pids

```
std::map<std::string,PID> UAVparams::pids
```

4.21.4.25 rotorMixer

```
Eigen::MatrixX4d UAVparams::rotorMixer
```

4.21.4.26 rotors

```
std::unique_ptr<Rotor[]> UAVparams::rotors
```

4.21.4.27 sensors

```
std::vector<SensorParams> UAVparams::sensors
```

4.21.4.28 surfaceMixer

```
Eigen::MatrixX4d UAVparams::surfaceMixer
```

4.21.4.29 surfaces

```
ControlSurfaces UAVparams::surfaces
```

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

- src/parser/uav_params.hpp
- src/parser/uav_params.cpp

Chapter 5

File Documentation

5.1 header/common.hpp File Reference

```
#include "../src/logger/logger.hpp"
#include "../src/ode/ode.hpp"
#include "../src/PID/PID.hpp"
#include "../src/timed_loop/timed_loop.hpp"
#include "../src/timed_loop/status.hpp"
#include "../src/parser/parser.hpp"
#include "../src/parser/uav_params.hpp"
#include "../src/components/components.hpp"
```

Include dependency graph for common.hpp:

5.2 src/components/aero_coefficients.hpp File Reference

```
#include <Eigen/Dense>
```

Include dependency graph for aero_coefficients.hpp: This graph shows which files directly or indirectly include this file:

Classes

- struct AeroCoefficients

  Aerodynamic coefficient.

5.3 src/components/components.hpp File Reference

```
#include "drive.hpp"
#include "control_surfaces.hpp"
#include "aero_coefficients.hpp"
#include "loads.hpp"
#include "navi.hpp"
```

Include dependency graph for components.hpp: This graph shows which files directly or indirectly include this file:

5.4 src/components/control_surfaces.cpp File Reference

```
#include "control_surfaces.hpp"
```
Include dependency graph for control_surfaces.cpp:

5.5 src/components/control_surfaces.hpp File Reference

```
#include <Eigen/Dense>
```
Include dependency graph for control_surfaces.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ControlSurfaces

  Aircraft's control surfaces.

5.6 src/components/drive.cpp File Reference

```
#include "drive.hpp"
```
Include dependency graph for drive.cpp:

5.7 src/components/drive.hpp File Reference

```
#include <Eigen/Dense>
#include "hinge.hpp"
```
Include dependency graph for drive.hpp: This graph shows which files directly or indirectly include this file:

Classes

- struct Drive

  Drive propelling aircraft.
- struct Rotor

  Rotor engine with controlled speed.
- class Jet

  Jet rocket engine.

5.8 src/components/hinge.cpp File Reference

```
#include "hinge.hpp"
```
Include dependency graph for hinge.cpp:

Functions

- Eigen::Matrix3d asSkewMatrix (Eigen::Vector3d v)

5.8.1 Function Documentation

5.8.1.1 asSkewMatrix()

```
Eigen::Matrix3d asSkewMatrix (
            Eigen::Vector3d v )
```

5.9 src/components/hinge.hpp File Reference

```
#include <Eigen/Dense>
#include <mutex>
#include <memory>
```

Include dependency graph for hinge.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class Hinge

  Hinge connecting aircraft with drives.

5.10 src/components/loads.cpp File Reference

```
#include "loads.hpp"
#include <limits>
```

Include dependency graph for loads.cpp:

5.11 src/components/loads.hpp File Reference

```
#include <Eigen/Dense>
#include <atomic>
```

Include dependency graph for loads.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class Load

  Load of aircraft that can be droped or launched.
- class Ammo
- class Cargo

5.12 src/components/navi.hpp File Reference

```
#include <Eigen/Dense>
```
Include dependency graph for navi.hpp: This graph shows which files directly or indirectly include this file:

Classes

- struct SensorParams

  Base parameters of a sensor.
- struct AHRSParams

  AHRS parameters.
- struct EKFScalers

  Scalers for EKF.

5.13 src/logger/logger.cpp File Reference

```
#include "logger.hpp"
#include <Eigen/Dense>
#include <iostream>
#include <fstream>
#include <initializer_list>
#include <string>
#include <filesystem>
```
Include dependency graph for logger.cpp:

Functions

- bool shouldLog (uint8_t group)

5.13.1 Function Documentation

5.13.1.1 shouldLog()

```
bool shouldLog (
            uint8_t group )
```

5.14 src/logger/logger.hpp File Reference

```
#include <Eigen/Dense>
#include <iostream>
#include <fstream>
#include <initializer_list>
#include <string>
#include <filesystem>
```
Include dependency graph for logger.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class Logger

  Log vector data with timestamp in file.

Macros

- #define LOGGER_MASK -1

5.14.1 Macro Definition Documentation

5.14.1.1 LOGGER_MASK

```
#define LOGGER_MASK -1
```

5.15 src/ode/ode.cpp File Reference

```
#include "ode.hpp"
#include "ode_impl.hpp"
```
Include dependency graph for ode.cpp:

5.16 src/ode/ode.hpp File Reference

```
#include <functional>
#include <memory>
#include <Eigen/Dense>
```
Include dependency graph for ode.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ODE

  Ordinal differencial equation solver.

5.17 src/ode/ode_impl.hpp File Reference

```
#include "ode.hpp"
```
Include dependency graph for ode_impl.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ODE_Euler

  Explicit Euler algorithm.
- class ODE_Heun

  Second order explicit Heun algorithm.
- class ODE_RK4

  Fourth order Runge Kutta algorithm.

5.18 src/ode/ode_test.cpp File Reference

```
#include "ode.hpp"
#include <gtest/gtest.h>
#include <numbers>
```

Include dependency graph for ode_test.cpp:

Classes

- class ODETest

Functions

- std::vector< ODE::ODEMethod > getMethodsToTest ()
- TEST_F (ODETest, FromStringTest)
- TEST_F (ODETest, FactoryTest)
- TEST_P (ODETest, TestConstFunction)
- TEST_P (ODETest, TestFirstOrder)
- TEST_P (ODETest, TestRHSCalls)
- INSTANTIATE_TEST_SUITE_P (TestDerivedClasses, ODETest, testing::ValuesIn(getMethodsToTest()))
- int main (int argc, char **argv)

5.18.1 Function Documentation

5.18.1.1 getMethodsToTest()

```
std::vector<ODE::ODEMethod> getMethodsToTest ( )
```

5.18.1.2 INSTANTIATE_TEST_SUITE_P()

```
INSTANTIATE_TEST_SUITE_P (
            TestDerivedClasses ,
            ODETest ,
            testing::ValuesIn(getMethodsToTest())  )
```

5.18.1.3 main()

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

5.18.1.4 TEST_F() [1/2]

```
TEST_F (
            ODETest ,
            FactoryTest  )
```

5.18.1.5 TEST_F() [2/2]

```
TEST_F (
            ODETest ,
            FromStringTest  )
```

5.18.1.6 TEST_P() [1/3]

```
TEST_P (
            ODETest ,
            TestConstFunction  )
```

5.18.1.7 TEST_P() [2/3]

```
TEST_P (
            ODETest ,
            TestFirstOrder  )
```

5.18.1.8 TEST_P() [3/3]

```
TEST_P (
            ODETest ,
            TestRHSCalls  )
```

5.19 src/parser/parser.cpp File Reference

```
#include "parser.hpp"
#include <Eigen/Dense>
#include <iostream>
#include <sstream>
```

Include dependency graph for parser.cpp:

Functions

- Eigen::MatrixXd parseMatrixXd (const std::string &input, int R, int C, char delimiter)

  Parse input string to double matrix of specific shape and delimiter.
- Eigen::VectorXd parseVectorXd (std::string str, int noOfElem, char delimiter)

  Parse input string to double vector of specific length and delimiter.

5.19.1 Function Documentation

5.19.1.1 parseMatrixXd()

```
Eigen::MatrixXd parseMatrixXd (
            const std::string & input,
            int R,
            int C,
            char delimiter = ' ' )
```

Parse input string to double matrix of specific shape and delimiter.

Parameters

input	input string
R	number of rows
C	number of columns
delimiter	delimiter

Returns

parsed matrix

5.19.1.2 parseVectorXd()

```
Eigen::VectorXd parseVectorXd (
            std::string str,
            int noOfElem,
            char delimiter = ' ' )
```

Parse input string to double vector of specific length and delimiter.

Parameters

str	input string
noOfElem	length of vector
delimiter	delimiter

Returns

parsed vector

5.20 src/parser/parser.hpp File Reference

`#include <Eigen/Dense>`
Include dependency graph for parser.hpp: This graph shows which files directly or indirectly include this file:

Functions

- Eigen::MatrixXd parseMatrixXd (const std::string &input, int R, int C, char delimiter=' ')
  Parse input string to double matrix of specific shape and delimiter.
- Eigen::VectorXd parseVectorXd (std::string str, int noOfElem, char delimiter=' ')
  Parse input string to double vector of specific length and delimiter.

5.20.1 Function Documentation

5.20.1.1 parseMatrixXd()

```
Eigen::MatrixXd parseMatrixXd (
            const std::string & input,
            int R,
            int C,
            char delimiter = ' ' )
```

Parse input string to double matrix of specific shape and delimiter.

Parameters

input	input string
R	number of rows
C	number of columns
delimiter	delimiter

Returns

parsed matrix

5.20.1.2 parseVectorXd()

```
Eigen::VectorXd parseVectorXd (
            std::string str,
            int noOfElem,
            char delimiter = ' ' )
```

Parse input string to double vector of specific length and delimiter.

Parameters

str	input string
noOfElem	length of vector
delimiter	delimiter

Returns

parsed vector

5.21 src/parser/uav_params.cpp File Reference

```
#include <Eigen/Dense>
#include "uav_params.hpp"
#include <iostream>
#include <fstream>
#include <filesystem>
#include <mutex>
#include "rapidxml/rapidxml.hpp"
#include "parser.hpp"
```

Include dependency graph for uav_params.cpp:

Functions

- void parseHinge (rapidxml::xml_node<> *hingeNode, Hinge *hinge)
- PID parsePID (rapidxml::xml_node<> *PIDNode)

5.21.1 Function Documentation

5.21.1.1 parseHinge()

```
void parseHinge (
            rapidxml::xml_node<> * hingeNode,
            Hinge * hinge )
```

5.21.1.2 parsePID()

```
PID parsePID (
            rapidxml::xml_node<> * PIDNode )
```

5.22 src/parser/uav_params.hpp File Reference

```
#include <Eigen/Dense>
#include <mutex>
#include <memory>
#include <map>
#include "rapidxml/rapidxml.hpp"
#include "../components/components.hpp"
#include "../PID/PID.hpp"
```

Include dependency graph for uav_params.hpp: This graph shows which files directly or indirectly include this file:

Classes

- struct UAVparams

  Parsed UAV configuration from XML.

5.23 src/PID/PID.cpp File Reference

```
#include "PID.hpp"
#include <limits>
#include <algorithm>
```

Include dependency graph for PID.cpp:

5.24 src/PID/PID.hpp File Reference

```
#include <limits>
```

Include dependency graph for PID.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class PID

  PID discrete controller.

Enumerations

- enum AntiWindUpMode { None , Clamping }

  Methods of handling windup in controller.

5.24.1 Enumeration Type Documentation

5.24.1.1 AntiWindUpMode

```
enum AntiWindUpMode
```

Methods of handling windup in controller.

Enumerator

None	
Clamping	

5.25 src/timed_loop/status.hpp File Reference

This graph shows which files directly or indirectly include this file:

Enumerations

- enum Status { idle = 1 , running = 2 , exiting = 3 , reload = 4 }
  status of timed loop. Control it's job

5.25.1 Enumeration Type Documentation

5.25.1.1 Status

```
enum Status
```

status of timed loop. Control it's job

Enumerator

idle	loop is ready to run
running	loop is running
exiting	loop will be break in next occasion.
reload	loop job should be reloaded

5.26 src/timed_loop/timed_loop.cpp File Reference

```
#include "timed_loop.hpp"
#include <stdint.h>
#include <chrono>
#include <thread>
#include "status.hpp"
#include <iostream>
```

Include dependency graph for timed_loop.cpp:

5.27 src/timed_loop/timed_loop.hpp File Reference

```
#include <stdint.h>
#include <functional>
#include "status.hpp"
```

Include dependency graph for timed_loop.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class TimedLoop

  Simulation of real-time synchronized loop.

Index