UAV drop physic

Generated by Doxygen 1.9.1

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

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

Namespace Documentation

5.1 controllers Namespace Reference

Classes

- class BangBang
- class DoubleSetpoint
- class PID
- class PID_Discrete
- class ZTransformStatic
- class ZTransform

5.2 def Namespace Reference

Controller constants.

Variables

- const int INFO_PERIOD = 2

  How often send demands in response to stick command.

5.2.1 Detailed Description

Controller constants.

5.2.2 Variable Documentation

5.2.2.1 INFO_PERIOD

```
const int def::INFO_PERIOD = 2
```

How often send demands in response to stick command.

Chapter 6

Class Documentation

6.1 Accelerometer Class Reference

Representation of accelerometer.

```
#include <sensors.hpp>
```

Inheritance diagram for Accelerometer:

Collaboration diagram for Accelerometer:

Public Member Functions

- Accelerometer (Environment &env, double sd, Eigen::Vector3d bias, double refreshTime)
- void update () override

  Update sensor state. Measured value is updated if sensor is ready for next read.

Static Public Attributes

- static const Eigen::Vector3d g = Eigen::Vector3d(0.0,0.0,9.81)

Additional Inherited Members

6.1.1 Detailed Description

Representation of accelerometer.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Accelerometer()

```
Accelerometer::Accelerometer (
             Environment & env,
             double sd,
             Eigen::Vector3d bias,
             double refreshTime )
```

6.1.3 Member Function Documentation

6.1.3.1 update()

```
void Accelerometer::update ( ) [override], [virtual]
```

Update sensor state. Measured value is updated if sensor is ready for next read.

Implements Sensor< Eigen::Vector3d >.

6.1.4 Member Data Documentation

6.1.4.1 g

```
const Eigen::Vector3d Accelerometer::g = Eigen::Vector3d(0.0,0.0,9.81) [static]
```

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

- src/navigation/sensors.hpp
- src/navigation/sensors.cpp

6.2 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

6.2.1 Detailed Description

Aerodynamic coefficient.

6.2.2 Member Data Documentation

6.2.2.1 C0

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

6.2.2.2 Cab

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

6.2.2.3 Cpqr

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

6.2.2.4 d

```
double AeroCoefficients::d
```

6.2.2.5 eAR

```
double AeroCoefficients::eAR
```

6.2.2.6 S

```
double AeroCoefficients::S
```

6.2.2.7 stallLimit

```
double AeroCoefficients::stallLimit
```

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

- lib/UAV_common/src/components/aero_coefficients.hpp

6.3 AHRS Class Reference

Attitude and heading reference system.

```
#include <AHRS.hpp>
```

Inheritance diagram for AHRS:

Collaboration diagram for AHRS:

Public Member Functions

- AHRS (Environment &env)

  Constructor.
- ∼AHRS ()

  Deconstructor.
- Eigen::Vector3d getOri ()

  Returns estimatied orientation vector (roll, pitch, yaw)
- virtual Eigen::Vector3d getGyroBias ()

  Returns estimatied gyroscope bias.
- virtual Eigen::Matrix3d rot_bw ()=0

  Returns rotation matrix from body to world frame.
- virtual void update (Eigen::Vector3d gyro, Eigen::Vector3d acc, Eigen::Vector3d mag)=0

Protected Attributes

- Eigen::Vector3d ori_est
- std::mutex mtxOri
- Environment & env
- Logger logger

6.3.1 Detailed Description

Attitude and heading reference system.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 AHRS()

```
AHRS::AHRS (
            Environment & env )
```

Constructor.

Parameters

env	eference to environment, where AHRS works

6.3.2.2 ∼AHRS()

```
AHRS::∼AHRS ( )
```

Deconstructor.

6.3.3 Member Function Documentation

6.3.3.1 getGyroBias()

```
Eigen::Vector3d AHRS::getGyroBias ( )  [virtual]
```

Returns estimatied gyroscope bias.

Returns

gyroscope bias

Reimplemented in AHRS_EKF.

6.3.3.2 getOri()

```
Eigen::Vector3d AHRS::getOri ( )
```

Returns estimatied orientation vector (roll, pitch, yaw)

Returns

estimatied orientation

6.3.3.3 rot_bw()

```
virtual Eigen::Matrix3d AHRS::rot_bw ( )  [pure virtual]
```

Returns rotation matrix from body to world frame.

Returns

rotation matrix

Implemented in AHRS_EKF, and AHRS_complementary.

6.3.3.4 update()

```
virtual void AHRS::update (
            Eigen::Vector3d gyro,
            Eigen::Vector3d acc,
            Eigen::Vector3d mag )  [pure virtual]
```

Implemented in AHRS_EKF, and AHRS_complementary.

6.3.4 Member Data Documentation

6.3.4.1 env

```
Environment& AHRS::env  [protected]
```

6.3.4.2 logger

```
Logger AHRS::logger  [protected]
```

6.3.4.3 mtxOri

```
std::mutex AHRS::mtxOri  [protected]
```

6.3.4.4 ori_est

```
Eigen::Vector3d AHRS::ori_est  [protected]
```

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

- src/navigation/AHRS.hpp
- src/navigation/AHRS.cpp

6.4 AHRS_complementary Class Reference

Implementation of AHRS based on Complementary Filter.

```
#include <AHRS_complementary.hpp>
```

Inheritance diagram for AHRS_complementary:

Collaboration diagram for AHRS_complementary:

Public Member Functions

- AHRS_complementary (Environment &env, double alpha)
- ∼AHRS_complementary ()
- Eigen::Matrix3d rot_bw () override

  Returns rotation matrix from body to world frame.
- void update (Eigen::Vector3d gyro, Eigen::Vector3d acc, Eigen::Vector3d mag) override

Protected Attributes

- const double alpha

6.4.1 Detailed Description

Implementation of AHRS based on Complementary Filter.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 AHRS_complementary()

```
AHRS_complementary::AHRS_complementary (
            Environment & env,
            double alpha )
```

6.4.2.2 ∼AHRS_complementary()

```
AHRS_complementary::∼AHRS_complementary ( )
```

6.4.3 Member Function Documentation

6.4.3.1 rot_bw()

```
Eigen::Matrix3d AHRS_complementary::rot_bw ( )  [override], [virtual]
```

Returns rotation matrix from body to world frame.

Returns

rotation matrix

Implements AHRS.

6.4.3.2 update()

```
void AHRS_complementary::update (
             Eigen::Vector3d gyro,
             Eigen::Vector3d acc,
             Eigen::Vector3d mag )  [override], [virtual]
```

Implements AHRS.

6.4.4 Member Data Documentation

6.4.4.1 alpha

```
const double AHRS_complementary::alpha  [protected]
```

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

- src/navigation/AHRS/AHRS_complementary.hpp
- src/navigation/AHRS/AHRS_complementary.cpp

6.5 AHRS_EKF Class Reference

Implementation of AHRS based on Extended Kalman Filter.

```
#include <AHRS_EKF.hpp>
```

Inheritance diagram for AHRS_EKF:

Collaboration diagram for AHRS_EKF:

Public Member Functions

- AHRS_EKF (Environment &env, double Q_scaler, double R_scaler)
- ∼AHRS_EKF ()
- Eigen::Vector3d getGyroBias () override

  Returns estimatied gyroscope bias.
- Eigen::Matrix3d rot_bw () override

  Returns rotation matrix from body to world frame.
- void update (Eigen::Vector3d gyro, Eigen::Vector3d acc, Eigen::Vector3d mag) override

Protected Member Functions

- Eigen::Vector4d q ()
- Eigen::Vector3d quaterionToRPY (Eigen::Vector4d q)
- Eigen::Vector4d RPYToQuaterion (Eigen::Vector3d RPY)

Protected Attributes

- Eigen::Vector< double, 7 > x
- Eigen::Matrix< double, 7, 7 > P
- Eigen::Matrix< double, 7, 7 > Q
- Eigen::Matrix< double, 6, 6 > R

6.5.1 Detailed Description

Implementation of AHRS based on Extended Kalman Filter.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 AHRS_EKF()

```
AHRS_EKF::AHRS_EKF (
            Environment & env,
            double Q_scaler,
            double R_scaler )
```

6.5.2.2 ∼AHRS_EKF()

```
AHRS_EKF::∼AHRS_EKF ( )
```

6.5.3 Member Function Documentation

6.5.3.1 getGyroBias()

```
Eigen::Vector3d AHRS_EKF::getGyroBias ( )  [override], [virtual]
```

Returns estimatied gyroscope bias.

Returns

gyroscope bias

Reimplemented from AHRS.

6.5.3.2 q()

```
Eigen::Vector4d AHRS_EKF::q ( )  [protected]
```

6.5.3.3 quaterionToRPY()

```
Eigen::Vector3d AHRS_EKF::quaterionToRPY (
             Eigen::Vector4d q )  [protected]
```

6.5.3.4 rot_bw()

```
Eigen::Matrix3d AHRS_EKF::rot_bw ( )  [override], [virtual]
```

Returns rotation matrix from body to world frame.

Returns

rotation matrix

Implements AHRS.

6.5.3.5 RPYToQuaterion()

```
Eigen::Vector4d AHRS_EKF::RPYToQuaterion (
             Eigen::Vector3d RPY )  [protected]
```

6.5.3.6 update()

```
void AHRS_EKF::update (
             Eigen::Vector3d gyro,
             Eigen::Vector3d acc,
             Eigen::Vector3d mag )  [override], [virtual]
```

Implements AHRS.

6.5.4 Member Data Documentation

6.5.4.1 P

```
Eigen::Matrix<double,7,7> AHRS_EKF::P  [protected]
```

6.5.4.2 Q

```
Eigen::Matrix<double,7,7> AHRS_EKF::Q  [protected]
```

6.5.4.3 R

```
Eigen::Matrix<double,6,6> AHRS_EKF::R  [protected]
```

6.5.4.4 x

```
Eigen::Vector<double,7> AHRS_EKF::x  [protected]
```

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

- src/navigation/AHRS/AHRS_EKF.hpp
- src/navigation/AHRS/AHRS_EKF.cpp

6.6 AHRSParams Struct Reference

AHRS parameters.

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

Public Attributes

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

6.6.1 Detailed Description

AHRS parameters.

6.6.2 Member Data Documentation

6.6.2.1 alpha

```
double AHRSParams::alpha
```

6.6.2.2 Q

```
double AHRSParams::Q
```

6.6.2.3 R

```
double AHRSParams::R
```

6.6.2.4 type

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

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

- lib/UAV_common/src/components/navi.hpp

6.7 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

6.7.1 Constructor & Destructor Documentation

6.7.1.1 Ammo() [1/2]

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

6.7.1.2 Ammo() [2/2]

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

6.7.2 Member Function Documentation

6.7.2.1 getV0()

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

get start velocity of ammo when launched

Returns

start velocity vector

6.7.2.2 operator=()

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

6.7.3 Member Data Documentation

6.7.3.1 _V0

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

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

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

6.8 controllers::BangBang Class Reference

```
#include <bang_bang.hpp>
```

Inheritance diagram for controllers::BangBang:

Collaboration diagram for controllers::BangBang:

Public Member Functions

- BangBang (double high, double low, double delta=0.0)

  Constructor with all Bang-bang controller parameters.
- BangBang (rapidxml::xml_node<> *controller_node)

  Construct controller with parameters from xml.
- double calc (double desired, double actual, [[maybe_unused]] double dt) override

  calc output of controller with specific time step
- void clear () override

  clear internal state
- std::unique_ptr< Controller > clone () const override

  virtual clone method

Additional Inherited Members

6.8.1 Constructor & Destructor Documentation

6.8.1.1 BangBang() [1/2]

```
controllers::BangBang::BangBang (
            double high,
            double low,
            double delta = 0.0 )
```

Constructor with all Bang-bang controller parameters.

Parameters

high	output when error is positive
low	output when error is negative
delta	histeresis symetrical to zero

6.8.1.2 BangBang() [2/2]

```
controllers::BangBang::BangBang (
            rapidxml::xml_node<> * controller_node )
```

Construct controller with parameters from xml.

Parameters

controller_node	xml node with controller params

6.8.2 Member Function Documentation

6.8.2.1 calc()

```
double controllers::BangBang::calc (
            double desired,
            double actual,
            [[maybe_unused] ] double dt )  [override]
```

calc output of controller with specific time step

Parameters

desired	input of controller, desired value
actual	measured actual value
dt	time step

Returns

output of controller

6.8.2.2 clear()

```
void controllers::BangBang::clear ( ) [override], [virtual]
```

clear internal state

Implements Controller.

6.8.2.3 clone()

```
std::unique_ptr< Controller > controllers::BangBang::clone ( ) const [override], [virtual]
```

virtual clone method

Implements Controller.

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

- lib/UAV_common/src/controllers/impl/bang_bang.hpp
- lib/UAV_common/src/controllers/impl/bang_bang.cpp

6.9 Barometer Class Reference

Representation of barometer.

```
#include <sensors.hpp>
```

Inheritance diagram for Barometer:

Collaboration diagram for Barometer:

Public Member Functions

- Barometer (Environment &env, double sd, Eigen::Vector3d bias, double refreshTime)
- void update () override

  Update sensor state. Measured value is updated if sensor is ready for next read.

Additional Inherited Members

6.9.1 Detailed Description

Representation of barometer.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Barometer()

```
Barometer::Barometer (
            Environment & env,
            double sd,
            Eigen::Vector3d bias,
            double refreshTime )
```

6.9.3 Member Function Documentation

6.9.3.1 update()

```
void Barometer::update ( )  [override], [virtual]
```

Update sensor state. Measured value is updated if sensor is ready for next read.

Implements Sensor< double >.

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

- src/navigation/sensors.hpp
- src/navigation/sensors.cpp

6.10 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

6.10.1 Constructor & Destructor Documentation

6.10.1.1 Cargo() [1/2]

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

6.10.1.2 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:

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

6.11 Control Class Reference

Control command listener & sender.

```
#include <control.hpp>
```

Public Member Functions

- Control (zmq::context_t ∗ctx, std::string uav_address, ControlSystem ∗controller)

  Constructor.
- ∼Control ()

  Deconstructor.
- void prepare ()

  Sends ping command.
- void start ()

  Sends start command.
- void stop ()

  Sends stop command.
- void recv ()

  Recivers reply and check if it contains "ok" phrase.
- void sendSpeed (Eigen::VectorXd speeds)

  Sends new demanded rotors speed.
- void sendSurface (Eigen::VectorXd angels)

  Sends new demanded surface deflactions.
- void startJet (int index)

  Sends command to start jet engine of given index.
- void sendHinge (char type, int index, int hinge_index, double value)

  Sends command to control hinge deflaction.
- std::string handleMsg (std::string msg)

  Handle incomming control message - message that instruct controller what to do.
- void setMode (ControllerMode mode)

6.11.1 Detailed Description

Control command listener & sender.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 Control()

```
Control::Control (
                zmq::context_t * ctx,
                std::string uav_address,
                ControlSystem * controller )
```

Constructor.

Parameters

ctx	zero mq context
uav_address	address to REP socket in simulation of controller uav
controller	pointer to controller instance

6.11.2.2 ∼Control()

```
Control::∼Control ( )
```

Deconstructor.

6.11.3 Member Function Documentation

6.11.3.1 handleMsg()

```
std::string Control::handleMsg (
             std::string msg )
```

Handle incomming control message - message that instruct controller what to do.

Parameters

msg	message content

Returns

reply to message

6.11.3.2 prepare()

```
void Control::prepare ( )
```

Sends ping command.

6.11.3.3 recv()

```
void Control::recv ( )
```

Recivers reply and check if it contains "ok" phrase.

6.11.3.4 sendHinge()

```
void Control::sendHinge (
             char type,
             int index,
             int hinge_index,
             double value )
```

Sends command to control hinge deflaction.

Parameters

type	hinge type: 'r' - rotor, 'j' - jet
index	drive index
hinge_index	hinge index
value	new deflection

6.11.3.5 sendSpeed()

```
void Control::sendSpeed (
            Eigen::VectorXd speeds )
```

Sends new demanded rotors speed.

Parameters

speeds	vector of demanded speeds

6.11.3.6 sendSurface()

```
void Control::sendSurface (
            Eigen::VectorXd angels )
```

Sends new demanded surface deflactions.

Parameters

speeds	vector of surface deflactions

6.11.3.7 setMode()

```
void Control::setMode (
            ControllerMode mode )
```

6.11.3.8 start()

```
void Control::start ( )
```

Sends start command.

6.11.3.9 startJet()

```
void Control::startJet (
             int index )
```

Sends command to start jet engine of given index.

Parameters

index	jet engine index

6.11.3.10 stop()

```
void Control::stop ( )
```

Sends stop command.

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

- src/communication/control.hpp
- src/communication/control.cpp
- src/communication/control_recv.cpp
- src/communication/control_send.cpp

6.12 Controller Class Reference

```
#include <controller.hpp>
```

Inheritance diagram for Controller:

Public Member Functions

- Controller ()

  Default constructor.
- ∼Controller ()

  Empty deconstructor for derived classes.
- virtual void set_dt (double dt)

  Set new time step.
- double calc (double desired, double actual)

  calc output of controller
- virtual double calc (double desired, double actual, double dt)=0

  calc output of controller with specific time step
- virtual void clear ()=0

  clear internal state
- virtual std::unique_ptr< Controller > clone () const =0

  virtual clone method

Static Public Member Functions

- static std::unique_ptr< Controller > ControllerFactory (rapidxml::xml_node<> *controller_node)

  construct controller from given node. If xml is not valid return nullptr.

Protected Attributes

- double _dt

6.12.1 Constructor & Destructor Documentation

6.12.1.1 Controller()

```
Controller::Controller ( )  [inline]
```

Default constructor.

6.12.1.2 ∼Controller()

```
Controller::∼Controller ( )  [inline]
```

Empty deconstructor for derived classes.

6.12.2 Member Function Documentation

6.12.2.1 calc() [1/2]

```
double Controller::calc (
            double desired,
            double actual )  [inline]
```

calc output of controller

Parameters

desired	input of controller, desired value
actual	measured actual value

Returns

output of controller

6.12.2.2 calc() [2/2]

```
virtual double Controller::calc (
             double desired,
             double actual,
             double dt ) [pure virtual]
```

calc output of controller with specific time step

Parameters

desired	input of controller, desired value
actual	measured actual value
dt	time step

Returns

output of controller

Implemented in controllers::PID_Discrete, controllers::PID, and controllers::DoubleSetpoint.

6.12.2.3 clear()

```
virtual void Controller::clear ( ) [pure virtual]
```

clear internal state

Implemented in controllers::ZTransform, controllers::ZTransformStatic< N, D >, controllers::PID_Discrete, controllers::PID, controllers::DoubleSetpoint, and controllers::BangBang.

6.12.2.4 clone()

```
virtual std::unique_ptr<Controller> Controller::clone ( ) const [pure virtual]
```

virtual clone method

Implemented in controllers::ZTransform, controllers::ZTransformStatic< N, D >, controllers::PID_Discrete, controllers::PID, controllers::DoubleSetpoint, and controllers::BangBang.

6.12.2.5 ControllerFactory()

```
std::unique_ptr< Controller > Controller::ControllerFactory (
             rapidxml::xml_node<> * controller_node ) [static]
```

construct controller from given node. If xml is not valid return nullptr.

Parameters

controller_node	xml node with controller config

6.12.2.6 set_dt()

```
virtual void Controller::set_dt (
             double dt )  [inline], [virtual]
```

Set new time step.

Parameters

dt	new time step

Reimplemented in controllers::PID_Discrete.

6.12.3 Member Data Documentation

6.12.3.1 _dt

```
double Controller::_dt  [protected]
```

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

- lib/UAV_common/src/controllers/controller.hpp
- lib/UAV_common/src/controllers/controller.cpp

6.13 ControllerLoop Class Reference

This class is interface of controller modes. All modes should keep this strucure and implements all true virtual methods.

```
#include <controller_loop.hpp>
```

Inheritance diagram for ControllerLoop:

Public Member Functions

- ControllerLoop (ControllerMode mode)

  Base class constructor.
- virtual ∼ControllerLoop ()

  Virtual deconstructor for defined behavior.
- virtual void job ([[maybe_unused]] std::map< std::string, std::unique_ptr< Controller >> &controllers, [[maybe_unused]] Control &control, [[maybe_unused]] NS &navisys)

  Controller job that will be called in control loop.
- virtual void handleJoystick ([[maybe_unused]] Eigen::VectorXd joystick)

  Handle incomming joystick deflaction.
- virtual std::string demandInfo ()

  Prepare info about state and demands.
- virtual const std::vector< std::string > & requiredcontrollers ()

  Defines controllers controller required by mode.
- virtual void overridePositionAndSpeed ([[maybe_unused]] Eigen::Vector3d position, [[maybe_unused]] Eigen::Vector3d orientation, [[maybe_unused]] Eigen::Vector3d velocity)

  Overrides demands to apply to given postion, orientation and speed.
- ControllerMode getMode ()

  Returns assigned mode enum value.

Static Public Member Functions

- static ControllerLoop ∗ ControllerLoopFactory (ControllerMode mode)

  ControllerLoop factor. Returns instace of ControllerLoop that implements specified mode.

Protected Member Functions

- bool checkJoystickLength (const Eigen::VectorXd &joystick, const int minimalSize)

  Check if joystick input vector is correct.

Protected Attributes

- const ControllerMode _mode
- std::vector< std::string > required_controllers

6.13.1 Detailed Description

This class is interface of controller modes. All modes should keep this strucure and implements all true virtual methods.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 ControllerLoop()

```
ControllerLoop::ControllerLoop (
            ControllerMode mode )
```

Base class constructor.

Parameters

mode	mode enum value

6.13.2.2 ∼ControllerLoop()

```
virtual ControllerLoop::∼ControllerLoop ( ) [inline], [virtual]
```

Virtual deconstructor for defined behavior.

6.13.3 Member Function Documentation

6.13.3.1 checkJoystickLength()

```
bool ControllerLoop::checkJoystickLength (
            const Eigen::VectorXd & joystick,
            const int minimalSize ) [protected]
```

Check if joystick input vector is correct.

Parameters

joystick	joystick axes deflaction
minimalSize	minimal length of deflation vector that can be interpreted

Returns

return true if joystick input vector is long enough

6.13.3.2 ControllerLoopFactory()

```
ControllerLoop * ControllerLoop::ControllerLoopFactory (
            ControllerMode mode ) [static]
```

ControllerLoop factor. Returns instace of ControllerLoop that implements specified mode.

Parameters

mode	demanded mode

Returns

Pointer to dynamically alocated ControllerLoop

6.13.3.3 demandInfo()

```
virtual std::string ControllerLoop::demandInfo ( )  [inline], [virtual]
```

Prepare info about state and demands.

Returns

information about mode and actually set demands

Reimplemented in ControllerLoopRMANUAL, ControllerLoopRGUIDED, ControllerLoopRANGLE, ControllerLoopQPOS, ControllerLoopQANGLE, ControllerLoopQACRO, ControllerLoopFANGLE, and ControllerLoopFACRO.

6.13.3.4 getMode()

```
ControllerMode ControllerLoop::getMode ( )  [inline]
```

Returns assigned mode enum value.

Returns

mode enum value

6.13.3.5 handleJoystick()

```
virtual void ControllerLoop::handleJoystick (
             [[maybe_unused] ] Eigen::VectorXd joystick )  [inline], [virtual]
```

Handle incomming joystick deflaction.

Parameters

joystick	joystick axes deflaction

6.13.3.6 job()

```
void ControllerLoop::job (
             [[maybe_unused] ] std::map< std::string, std::unique_ptr< Controller >> & controllers,
             [[maybe_unused] ] Control & control,
             [[maybe_unused] ] NS & navisys )  [virtual]
```

Controller job that will be called in control loop.

Parameters

controllers	map of aviliable controllers
control	reference to control instatce that is used to send control commands
navisys	navigation system reference

6.13.3.7 overridePositionAndSpeed()

```
virtual void ControllerLoop::overridePositionAndSpeed (
             [[maybe_unused] ] Eigen::Vector3d position,
             [[maybe_unused] ] Eigen::Vector3d orientation,
             [[maybe_unused] ] Eigen::Vector3d velocity )  [inline], [virtual]
```

Overrides demands to apply to given postion, orientation and speed.

Parameters

position	position vector in world frame
orientation	orientation vector in world frame
orientation	linear velocity vector in world frame

Reimplemented in ControllerLoopRANGLE, ControllerLoopQPOS, ControllerLoopQANGLE, and ControllerLoopFANGLE.

6.13.3.8 requiredcontrollers()

```
virtual const std::vector<std::string>& ControllerLoop::requiredcontrollers ( )  [inline],
[virtual]
```

Defines controllers controller required by mode.

Returns

vector of names of required controllers

6.13.4 Member Data Documentation

6.13.4.1 _mode

```
const ControllerMode ControllerLoop::_mode  [protected]
```

6.13.4.2 required_controllers

```
std::vector<std::string> ControllerLoop::required_controllers  [protected]
```

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

- src/controller/controller_loop.hpp
- src/controller/controller_loop.cpp

6.14 ControllerLoopFACRO Class Reference

```
#include <controller_loop_FACRO.hpp>
```

Inheritance diagram for ControllerLoopFACRO:

Collaboration diagram for ControllerLoopFACRO:

Public Member Functions

- ControllerLoopFACRO ()
- void job (std::map< std::string, std::unique_ptr< Controller >> &controllers, Control &control, NS &navisys) override
- void handleJoystick (Eigen::VectorXd joystick) override
- std::string demandInfo () override

    Prepare info about state and demands.

Additional Inherited Members

6.14.1 Constructor & Destructor Documentation

6.14.1.1 ControllerLoopFACRO()

```
ControllerLoopFACRO::ControllerLoopFACRO ( )
```

6.14.2 Member Function Documentation

6.14.2.1 demandInfo()

```
std::string ControllerLoopFACRO::demandInfo ( )  [override], [virtual]
```

Prepare info about state and demands.

Returns

information about mode and actually set demands

Reimplemented from ControllerLoop.

6.14.2.2 handleJoystick()

```
void ControllerLoopFACRO::handleJoystick (
            Eigen::VectorXd joystick )  [override]
```

6.14.2.3 job()

```
void ControllerLoopFACRO::job (
            std::map< std::string, std::unique_ptr< Controller >> & controllers,
            Control & control,
            NS & navisys )  [override]
```

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

- src/controller/modes/controller_loop_FACRO.hpp
- src/controller/modes/controller_loop_FACRO.cpp

6.15 ControllerLoopFANGLE Class Reference

```
#include <controller_loop_FANGLE.hpp>
```

Inheritance diagram for ControllerLoopFANGLE:

Collaboration diagram for ControllerLoopFANGLE:

Public Member Functions

- ControllerLoopFANGLE ()
- void job (std::map< std::string, std::unique_ptr< Controller >> &controllers, Control &control, NS &navisys) override
- void handleJoystick (Eigen::VectorXd joystick) override
- std::string demandInfo () override

  Prepare info about state and demands.
- void overridePositionAndSpeed ([[maybe_unused]] Eigen::Vector3d position, [[maybe_unused]] Eigen::↩ Vector3d orientation, [[maybe_unused]] Eigen::Vector3d velocity) override

  Overrides demands to apply to given postion, orientation and speed.

Static Public Attributes

- static constexpr double angleLimit = std::numbers::pi/2.0

Additional Inherited Members

6.15.1 Constructor & Destructor Documentation

6.15.1.1 ControllerLoopFANGLE()

```
ControllerLoopFANGLE::ControllerLoopFANGLE ( )
```

6.15.2 Member Function Documentation

6.15.2.1 demandInfo()

```
std::string ControllerLoopFANGLE::demandInfo ( )  [override], [virtual]
```

Prepare info about state and demands.

Returns

information about mode and actually set demands

Reimplemented from ControllerLoop.

6.15.2.2 handleJoystick()

```
void ControllerLoopFANGLE::handleJoystick (
             Eigen::VectorXd joystick )  [override]
```

6.15.2.3 job()

```
void ControllerLoopFANGLE::job (
             std::map< std::string, std::unique_ptr< Controller >> & controllers,
             Control & control,
             NS & navisys )  [override]
```

6.15.2.4 overridePositionAndSpeed()

```
void ControllerLoopFANGLE::overridePositionAndSpeed (
             [[maybe_unused] ] Eigen::Vector3d position,
             [[maybe_unused] ] Eigen::Vector3d orientation,
             [[maybe_unused] ] Eigen::Vector3d velocity )  [override], [virtual]
```

Overrides demands to apply to given postion, orientation and speed.

Parameters

position	position vector in world frame
orientation	orientation vector in world frame
orientation	linear velocity vector in world frame

Reimplemented from ControllerLoop.

6.15.3 Member Data Documentation

6.15.3.1 angleLimit

```
constexpr double ControllerLoopFANGLE::angleLimit = std::numbers::pi/2.0  [static], [constexpr]
```

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

- src/controller/modes/controller_loop_FANGLE.hpp
- src/controller/modes/controller_loop_FANGLE.cpp

6.16 ControllerLoopFMANUAL Class Reference

```
#include <controller_loop_FMANUAL.hpp>
```

Inheritance diagram for ControllerLoopFMANUAL:

Collaboration diagram for ControllerLoopFMANUAL:

Public Member Functions

- ControllerLoopFMANUAL ()
- void job ([[maybe_unused]] std::map< std::string, std::unique_ptr< Controller >> &controllers, Control &control, [[maybe_unused]] NS &navisys) override
- void handleJoystick (Eigen::VectorXd joystick) override

Additional Inherited Members

6.16.1 Constructor & Destructor Documentation

6.16.1.1 ControllerLoopFMANUAL()

```
ControllerLoopFMANUAL::ControllerLoopFMANUAL ( )
```

6.16.2 Member Function Documentation

6.16.2.1 handleJoystick()

```
void ControllerLoopFMANUAL::handleJoystick (
            Eigen::VectorXd joystick )  [override]
```

6.16.2.2 job()

```
void ControllerLoopFMANUAL::job (
            [[maybe_unused] ] std::map< std::string, std::unique_ptr< Controller >> & controllers,
            Control & control,
            [[maybe_unused] ] NS & navisys )  [override]
```

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

- src/controller/modes/controller_loop_FMANUAL.hpp
- src/controller/modes/controller_loop_FMANUAL.cpp

6.17 ControllerLoopNONE Class Reference

```
#include <controller_loop_NONE.hpp>
```

Inheritance diagram for ControllerLoopNONE:

Collaboration diagram for ControllerLoopNONE:

Public Member Functions

- ControllerLoopNONE ()

Additional Inherited Members

6.17.1 Constructor & Destructor Documentation

6.17.1.1 ControllerLoopNONE()

```
ControllerLoopNONE::ControllerLoopNONE ( )
```

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

- src/controller/modes/controller_loop_NONE.hpp
- src/controller/modes/controller_loop_NONE.cpp

6.18 ControllerLoopQACRO Class Reference

```
#include <controller_loop_QACRO.hpp>
```

Inheritance diagram for ControllerLoopQACRO:

Collaboration diagram for ControllerLoopQACRO:

Public Member Functions

- ControllerLoopQACRO ()
- void job (std::map< std::string, std::unique_ptr< Controller >> &controllers, Control &control, NS &navisys) override
- void handleJoystick (Eigen::VectorXd joystick) override
- std::string demandInfo () override

  Prepare info about state and demands.

Additional Inherited Members

6.18.1 Constructor & Destructor Documentation

6.18.1.1 ControllerLoopQACRO()

```
ControllerLoopQACRO::ControllerLoopQACRO ( )
```

6.18.2 Member Function Documentation

6.18.2.1 demandInfo()

```
std::string ControllerLoopQACRO::demandInfo ( )  [override], [virtual]
```

Prepare info about state and demands.

Returns

information about mode and actually set demands

Reimplemented from ControllerLoop.

6.18.2.2 handleJoystick()

```
void ControllerLoopQACRO::handleJoystick (
            Eigen::VectorXd joystick )  [override]
```

6.18.2.3 job()

```
void ControllerLoopQACRO::job (
            std::map< std::string, std::unique_ptr< Controller >> & controllers,
            Control & control,
            NS & navisys )  [override]
```

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

- src/controller/modes/controller_loop_QACRO.hpp
- src/controller/modes/controller_loop_QACRO.cpp

6.19 ControllerLoopQANGLE Class Reference

```
#include <controller_loop_QANGLE.hpp>
```

Inheritance diagram for ControllerLoopQANGLE:

Collaboration diagram for ControllerLoopQANGLE:

Public Member Functions

- ControllerLoopQANGLE ()
- void job (std::map< std::string, std::unique_ptr< Controller >> &controllers, Control &control, NS &navisys) override
- void handleJoystick (Eigen::VectorXd joystick) override
- std::string demandInfo () override

  Prepare info about state and demands.
- void overridePositionAndSpeed ([[maybe_unused]] Eigen::Vector3d position, [[maybe_unused]] Eigen::↩Vector3d orientation, [[maybe_unused]] Eigen::Vector3d velocity) override

  Overrides demands to apply to given postion, orientation and speed.

Additional Inherited Members

6.19.1 Constructor & Destructor Documentation

6.19.1.1 ControllerLoopQANGLE()

```
ControllerLoopQANGLE::ControllerLoopQANGLE ( )
```

6.19.2 Member Function Documentation

6.19.2.1 demandInfo()

```
std::string ControllerLoopQANGLE::demandInfo ( ) [override], [virtual]
```

Prepare info about state and demands.

Returns

information about mode and actually set demands

Reimplemented from ControllerLoop.

6.19.2.2 handleJoystick()

```
void ControllerLoopQANGLE::handleJoystick (
             Eigen::VectorXd joystick ) [override]
```

6.19.2.3 job()

```
void ControllerLoopQANGLE::job (
             std::map< std::string, std::unique_ptr< Controller >> & controllers,
             Control & control,
             NS & navisys ) [override]
```

6.19.2.4 overridePositionAndSpeed()

```
void ControllerLoopQANGLE::overridePositionAndSpeed (
             [[maybe_unused] ] Eigen::Vector3d position,
             [[maybe_unused] ] Eigen::Vector3d orientation,
             [[maybe_unused] ] Eigen::Vector3d velocity ) [override], [virtual]
```

Overrides demands to apply to given postion, orientation and speed.

Parameters

position	position vector in world frame
orientation	orientation vector in world frame
orientation	linear velocity vector in world frame

Reimplemented from ControllerLoop.

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

- src/controller/modes/controller_loop_QANGLE.hpp
- src/controller/modes/controller_loop_QANGLE.cpp

6.20 ControllerLoopQPOS Class Reference

```
#include <controller_loop_QPOS.hpp>
```

Inheritance diagram for ControllerLoopQPOS:

Collaboration diagram for ControllerLoopQPOS:

Public Member Functions

- ControllerLoopQPOS ()
- void job (std::map< std::string, std::unique_ptr< Controller >> &controllers, Control &control, NS &navisys) override
- void handleJoystick (Eigen::VectorXd joystick) override
- std::string demandInfo () override

  Prepare info about state and demands.
- void overridePositionAndSpeed ([[maybe_unused]] Eigen::Vector3d position, [[maybe_unused]] Eigen::↩ Vector3d orientation, [[maybe_unused]] Eigen::Vector3d velocity) override

  Overrides demands to apply to given postion, orientation and speed.

Additional Inherited Members

6.20.1 Constructor & Destructor Documentation

6.20.1.1 ControllerLoopQPOS()

```
ControllerLoopQPOS::ControllerLoopQPOS ( )
```

6.20.2 Member Function Documentation

6.20.2.1 demandInfo()

```
std::string ControllerLoopQPOS::demandInfo ( ) [override], [virtual]
```

Prepare info about state and demands.

Returns

information about mode and actually set demands

Reimplemented from ControllerLoop.

6.20.2.2 handleJoystick()

```
void ControllerLoopQPOS::handleJoystick (
            Eigen::VectorXd joystick ) [override]
```

6.20.2.3 job()

```
void ControllerLoopQPOS::job (
            std::map< std::string, std::unique_ptr< Controller >> & controllers,
            Control & control,
            NS & navisys ) [override]
```

6.20.2.4 overridePositionAndSpeed()

```
void ControllerLoopQPOS::overridePositionAndSpeed (
            [[maybe_unused] ] Eigen::Vector3d position,
            [[maybe_unused] ] Eigen::Vector3d orientation,
            [[maybe_unused] ] Eigen::Vector3d velocity ) [override], [virtual]
```

Overrides demands to apply to given postion, orientation and speed.

Parameters

position	position vector in world frame
orientation	orientation vector in world frame
orientation	linear velocity vector in world frame

Reimplemented from ControllerLoop.

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

- src/controller/modes/controller_loop_QPOS.hpp
- src/controller/modes/controller_loop_QPOS.cpp

6.21 ControllerLoopRANGLE Class Reference

```
#include <controller_loop_RANGLE.hpp>
```

Inheritance diagram for ControllerLoopRANGLE:

Collaboration diagram for ControllerLoopRANGLE:

Public Member Functions

- ControllerLoopRANGLE ()
- void job ([[maybe_unused]] std::map< std::string, std::unique_ptr< Controller >> &controllers, Control &control, [[maybe_unused]] NS &navisys) override
- void handleJoystick (Eigen::VectorXd joystick) override
- std::string demandInfo () override

  Prepare info about state and demands.
- void overridePositionAndSpeed ([[maybe_unused]] Eigen::Vector3d position, [[maybe_unused]] Eigen::↩ Vector3d orientation, [[maybe_unused]] Eigen::Vector3d velocity) override

  Overrides demands to apply to given postion, orientation and speed.

Protected Attributes

- std::atomic< double > demandedTheta = 0.0
- std::atomic< double > demandedPsi = 0.0

Static Protected Attributes

- static constexpr double angleLimit = std::numbers::pi/2.0

Additional Inherited Members

6.21.1 Constructor & Destructor Documentation

6.21.1.1 ControllerLoopRANGLE()

```
ControllerLoopRANGLE::ControllerLoopRANGLE ( )
```

6.21.2 Member Function Documentation

6.21.2.1 demandInfo()

```
std::string ControllerLoopRANGLE::demandInfo ( )  [override], [virtual]
```

Prepare info about state and demands.

Returns

information about mode and actually set demands

Reimplemented from ControllerLoop.

6.21.2.2 handleJoystick()

```
void ControllerLoopRANGLE::handleJoystick (
             Eigen::VectorXd joystick )  [override]
```

6.21.2.3 job()

```
void ControllerLoopRANGLE::job (
             [[maybe_unused] ] std::map< std::string, std::unique_ptr< Controller >> & controllers,
             Control & control,
             [[maybe_unused] ] NS & navisys )  [override]
```

6.21.2.4 overridePositionAndSpeed()

```
void ControllerLoopRANGLE::overridePositionAndSpeed (
             [[maybe_unused] ] Eigen::Vector3d position,
             [[maybe_unused] ] Eigen::Vector3d orientation,
             [[maybe_unused] ] Eigen::Vector3d velocity )  [override], [virtual]
```

Overrides demands to apply to given postion, orientation and speed.

Parameters

position	position vector in world frame
orientation	orientation vector in world frame
orientation	linear velocity vector in world frame

Reimplemented from ControllerLoop.

6.21.3 Member Data Documentation

6.21.3.1 angleLimit

```
constexpr double ControllerLoopRANGLE::angleLimit = std::numbers::pi/2.0  [static], [constexpr],
[protected]
```

6.21.3.2 demandedPsi

```
std::atomic<double> ControllerLoopRANGLE::demandedPsi = 0.0  [protected]
```

6.21.3.3 demandedTheta

```
std::atomic<double> ControllerLoopRANGLE::demandedTheta = 0.0  [protected]
```

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

- src/controller/modes/controller_loop_RANGLE.hpp
- src/controller/modes/controller_loop_RANGLE.cpp

6.22 ControllerLoopRAUTOLAUNCH Class Reference

```
#include <controller_loop_RAUTOLAUNCH.hpp>
```

Inheritance diagram for ControllerLoopRAUTOLAUNCH:

Collaboration diagram for ControllerLoopRAUTOLAUNCH:

Public Member Functions

- ControllerLoopRAUTOLAUNCH ()
- void job ([[maybe_unused]] std::map< std::string, std::unique_ptr< Controller >> &controllers, Control &control, [[maybe_unused]] NS &navisys) override

Additional Inherited Members

6.22.1 Constructor & Destructor Documentation

6.22.1.1 ControllerLoopRAUTOLAUNCH()

```
ControllerLoopRAUTOLAUNCH::ControllerLoopRAUTOLAUNCH ( )
```

6.22.2 Member Function Documentation

6.22.2.1 job()

```
void ControllerLoopRAUTOLAUNCH::job (
            [[maybe_unused] ] std::map< std::string, std::unique_ptr< Controller >> & controllers,
            Control & control,
            [[maybe_unused] ] NS & navisys )  [override]
```

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

- src/controller/modes/controller_loop_RAUTOLAUNCH.hpp
- src/controller/modes/controller_loop_RAUTOLAUNCH.cpp

6.23 ControllerLoopRGUIDED Class Reference

```
#include <controller_loop_RGUIDED.hpp>
```

Inheritance diagram for ControllerLoopRGUIDED:

Collaboration diagram for ControllerLoopRGUIDED:

Public Member Functions

- ControllerLoopRGUIDED ()
- void job ([[maybe_unused]] std::map< std::string, std::unique_ptr< Controller >> &controllers, Control &control, [[maybe_unused]] NS &navisys) override
- std::string demandInfo () override

    Prepare info about state and demands.

Protected Attributes

- const Eigen::Vector3d target

Static Protected Attributes

- static constexpr double detection_limit = std::numbers::pi/3.0

Additional Inherited Members

6.23.1 Constructor & Destructor Documentation

6.23.1.1 ControllerLoopRGUIDED()

```
ControllerLoopRGUIDED::ControllerLoopRGUIDED ( )
```

6.23.2 Member Function Documentation

6.23.2.1 demandInfo()

```
std::string ControllerLoopRGUIDED::demandInfo ( )  [override], [virtual]
```

Prepare info about state and demands.

Returns

information about mode and actually set demands

Reimplemented from ControllerLoop.

6.23.2.2 job()

```
void ControllerLoopRGUIDED::job (
             [[maybe_unused] ] std::map< std::string, std::unique_ptr< Controller >> & controllers,
             Control & control,
             [[maybe_unused] ] NS & navisys )  [override]
```

6.23.3 Member Data Documentation

6.23.3.1 detection_limit

```
constexpr double ControllerLoopRGUIDED::detection_limit = std::numbers::pi/3.0  [static],
[constexpr], [protected]
```

6.23.3.2 target

```
const Eigen::Vector3d ControllerLoopRGUIDED::target  [protected]
```

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

- src/controller/modes/controller_loop_RGUIDED.hpp
- src/controller/modes/controller_loop_RGUIDED.cpp

6.24 ControllerLoopRMANUAL Class Reference

```
#include <controller_loop_RMANUAL.hpp>
```

Inheritance diagram for ControllerLoopRMANUAL:

Collaboration diagram for ControllerLoopRMANUAL:

Public Member Functions

- ControllerLoopRMANUAL ()
- void job ([[maybe_unused]] std::map< std::string, std::unique_ptr< Controller >> &controllers, Control &control, [[maybe_unused]] NS &navisys) override
- void handleJoystick (Eigen::VectorXd joystick) override
- std::string demandInfo () override

  Prepare info about state and demands.

Protected Attributes

- std::atomic< double > demanded_H = 0.0
- std::atomic< double > demanded_V = 0.0

Additional Inherited Members

6.24.1 Constructor & Destructor Documentation

6.24.1.1 ControllerLoopRMANUAL()

```
ControllerLoopRMANUAL::ControllerLoopRMANUAL ( )
```

6.24.2 Member Function Documentation

6.24.2.1 demandInfo()

```
std::string ControllerLoopRMANUAL::demandInfo ( )  [override], [virtual]
```

Prepare info about state and demands.

Returns

information about mode and actually set demands

Reimplemented from ControllerLoop.

6.24.2.2 handleJoystick()

```
void ControllerLoopRMANUAL::handleJoystick (
            Eigen::VectorXd joystick )  [override]
```

6.24.2.3 job()

```
void ControllerLoopRMANUAL::job (
            [[maybe_unused] ] std::map< std::string, std::unique_ptr< Controller >> & controllers,
            Control & control,
            [[maybe_unused] ] NS & navisys )  [override]
```

6.24.3 Member Data Documentation

6.24.3.1 demanded_H

```
std::atomic<double> ControllerLoopRMANUAL::demanded_H = 0.0  [protected]
```

6.24.3.2 demanded_V

```
std::atomic<double> ControllerLoopRMANUAL::demanded_V = 0.0  [protected]
```

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

- src/controller/modes/controller_loop_RMANUAL.hpp
- src/controller/modes/controller_loop_RMANUAL.cpp

6.25 ControllerTest Class Reference

Inheritance diagram for ControllerTest:

Collaboration diagram for ControllerTest:

Protected Member Functions

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

6.25.1 Member Function Documentation

6.25.1.1 SetUp()

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

6.25.1.2 TearDown()

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

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

- lib/UAV_common/src/controllers/controller_test.cpp

6.26 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

6.26.1 Detailed Description

Aircraft's control surfaces.

6.26.2 Constructor & Destructor Documentation

6.26.2.1 ControlSurfaces() [1/2]

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

6.26.2.2 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

6.26.3 Member Function Documentation

6.26.3.1 getCoefficients()

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

6.26.3.2 getNoOfSurface()

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

6.26.3.3 getValues()

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

6.26.3.4 restoreTrim()

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

6.26.3.5 setValues()

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

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

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

6.27 ControlSystem Class Reference

Central controller class.

```
#include <controller.hpp>
```

Public Member Functions

- ControlSystem (zmq::context_t ∗ctx, std::string uav_address)

  Constructor.
- ∼ControlSystem ()
- void run ()

  Run controller.
- void setMode (ControllerMode new_mode)

  Change controller mode.
- void exitController ()

  Stop controller loop.

Friends

- class Control

6.27.1 Detailed Description

Central controller class.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 ControlSystem()

```
ControlSystem::ControlSystem (
              zmq::context_t * ctx,
              std::string uav_address )
```

Constructor.

Parameters

ctx	zero mq context
uav_address	address of simulation sockets

6.27.2.2 ∼ControlSystem()

```
ControlSystem::∼ControlSystem ( )
```

6.27.3 Member Function Documentation

6.27.3.1 exitController()

```
void ControlSystem::exitController ( )
```

Stop controller loop.

6.27.3.2 run()

```
void ControlSystem::run ( )
```

Run controller.

6.27.3.3 setMode()

```
void ControlSystem::setMode (
            ControllerMode new_mode )
```

Change controller mode.

Parameters

new_mode	new contoller mode

6.27.4 Friends And Related Function Documentation

6.27.4.1 Control

```
friend class Control  [friend]
```

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

- src/controller/controller.hpp
- src/controller/controller.cpp

6.28 controllers::DoubleSetpoint Class Reference

```
#include <double_setpoint.hpp>
```

Inheritance diagram for controllers::DoubleSetpoint:

Collaboration diagram for controllers::DoubleSetpoint:

Public Member Functions

- DoubleSetpoint (double high, double mid, double low, double mid_range, double delta=0.0)

  Constructor with all Bang-bang controller parameters.
- DoubleSetpoint (rapidxml::xml_node<> *controller_node)

  Construct controller with parameters from xml.
- double calc (double desired, double actual, double dt) override

  calc output of controller with specific time step
- void clear () override

  clear internal state
- std::unique_ptr< Controller > clone () const override

  virtual clone method

Additional Inherited Members

6.28.1 Constructor & Destructor Documentation

6.28.1.1 DoubleSetpoint() [1/2]

```
controllers::DoubleSetpoint::DoubleSetpoint (
             double high,
             double mid,
             double low,
             double mid_range,
             double delta = 0.0 )
```

Constructor with all Bang-bang controller parameters.

Parameters

high	output when error is in positive range
mid	output when error is in center range
low	output when error is in negative range
mid_range	size of center field from zero
delta	histeresis symetrical to zero

6.28.1.2 DoubleSetpoint() [2/2]

```
controllers::DoubleSetpoint::DoubleSetpoint (
             rapidxml::xml_node<> * controller_node )
```

Construct controller with parameters from xml.

Parameters

controller_node	xml node with controller params

6.28.2 Member Function Documentation

6.28.2.1 calc()

```
double controllers::DoubleSetpoint::calc (
             double desired,
             double actual,
             double dt )  [override], [virtual]
```

calc output of controller with specific time step

Parameters

desired	input of controller, desired value
actual	measured actual value
dt	time step

Returns

output of controller

Implements Controller.

6.28.2.2 clear()

```
void controllers::DoubleSetpoint::clear ( )  [override], [virtual]
```

clear internal state

Implements Controller.

6.28.2.3 clone()

```
std::unique_ptr< Controller > controllers::DoubleSetpoint::clone ( ) const  [override], [virtual]
```

virtual clone method

Implements Controller.

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

- lib/UAV_common/src/controllers/impl/double_setpoint.hpp
- lib/UAV_common/src/controllers/impl/double_setpoint.cpp

6.29 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]

6.29.1 Detailed Description

Drive propelling aircraft.

6.29.2 Member Data Documentation

6.29.2.1 axis

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

6.29.2.2 hinges

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

6.29.2.3 noOfHinges

```
int Drive::noOfHinges
```

6.29.2.4 position

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

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

- lib/UAV_common/src/components/drive.hpp

6.30 EKF Class Reference

Extended Kalman Filter.

```
#include <EKF.hpp>
```

Public Member Functions

- EKF (EKFParams params)

  Constructor.
- Eigen::Vector3d getPos ()

  Returns estimated position vector.
- Eigen::Vector3d getVel ()

  Returns estimated velocity vector.
- void predict (double time, Eigen::Vector3d acc)

  Predict phase. Integration of accelerometer measures.
- void updateBaro (double time, double baro)

  Update phase. Height correction.
- void updateGPS (double time, Eigen::Vector3d pos)

  Update phase. Position correction.
- void updateGPSVel (double time, Eigen::Vector3d vel)

  Update phase. Velocity correction.
- void log (double time)

  Log filter state.

6.30.1 Detailed Description

Extended Kalman Filter.

6.30.2 Constructor & Destructor Documentation

6.30.2.1 EKF()

```
EKF::EKF (
                EKFParams params )
```

Constructor.

Parameters

params	filter parameters

6.30.3 Member Function Documentation

6.30.3.1 getPos()

```
Eigen::Vector3d EKF::getPos ( )
```

Returns estimated position vector.

Returns

position vector in world frame

6.30.3.2 getVel()

```
Eigen::Vector3d EKF::getVel ( )
```

Returns estimated velocity vector.

Returns

velocity vector in world frame

6.30.3.3 log()

```
void EKF::log (
             double time )
```

Log filter state.

Parameters

time	simulation time

6.30.3.4 predict()

```
void EKF::predict (
            double time,
            Eigen::Vector3d acc )
```

Predict phase. Integration of accelerometer measures.

Parameters

time	simulation time
acc	accelerometer measure

6.30.3.5 updateBaro()

```
void EKF::updateBaro (
            double time,
            double baro )
```

Update phase. Height correction.

Parameters

time	simulation time
baro	barometer measure

6.30.3.6 updateGPS()

```
void EKF::updateGPS (
            double time,
            Eigen::Vector3d pos )
```

Update phase. Position correction.

Parameters

time	simulation time
baro	GPS location measure

6.30.3.7 updateGPSVel()

```
void EKF::updateGPSVel (
            double time,
            Eigen::Vector3d vel )
```

Update phase. Velocity correction.

Parameters

time	simulation time
baro	GPS velocity measure

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

- src/navigation/EKF.hpp
- src/navigation/EKF.cpp

6.31 EKFParams Struct Reference

EK filer parameters.

```
#include <EKF.hpp>
```

Public Attributes

- Eigen::Matrix< double, 6, 6 > P0
- Eigen::Matrix< double, 6, 6 > Q
- double RBaro
- Eigen::Matrix3d RGPSPos
- Eigen::Matrix3d RGPSVel

6.31.1 Detailed Description

EK filer parameters.

6.31.2 Member Data Documentation

6.31.2.1 P0

```
Eigen::Matrix<double,6,6> EKFParams::P0
```

6.31.2.2 Q

```
Eigen::Matrix<double,6,6> EKFParams::Q
```

6.31.2.3 RBaro

```
double EKFParams::RBaro
```

6.31.2.4 RGPSPos

```
Eigen::Matrix3d EKFParams::RGPSPos
```

6.31.2.5 RGPSVel

```
Eigen::Matrix3d EKFParams::RGPSVel
```

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

- src/navigation/EKF.hpp

6.32 EKFScalers Struct Reference

Scalers for EKF.

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

Public Attributes

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

6.32.1 Detailed Description

Scalers for EKF.

6.32.2 Member Data Documentation

6.32.2.1 baroScaler

```
double EKFScalers::baroScaler
```

6.32.2.2 predictScaler

```
double EKFScalers::predictScaler
```

6.32.2.3 updateScaler

```
double EKFScalers::updateScaler
```

6.32.2.4 zScaler

```
double EKFScalers::zScaler
```

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

- lib/UAV_common/src/components/navi.hpp

6.33 Environment Class Reference

```
#include <environment.hpp>
```

Public Member Functions

- Environment (zmq::context_t ∗ctx, std::string uav_address)

  Constructor.
- ∼Environment ()

  Deconstructor.
- double getTime ()

  Returns time of simulation.
- Eigen::Vector3d getPosition ()

  Returns exact postion vector.
- Eigen::Vector4d getOrientation ()

  Returns exact orientation vector.
- Eigen::Vector3d getWorldLinearVelocity ()

  Returns exact linear velocity vector.
- Eigen::Vector3d getWorldAngularVelocity ()

  Returns exact angular velocity vector.

- Eigen::Vector3d getLinearVelocity ()

  Returns exact linear velocity vector.
- Eigen::Vector3d getAngularVelocity ()

  Returns exact angular velocity vector.
- Eigen::Vector3d getLinearAcceleration ()

  Returns exact linear acceleration vector.
- Eigen::Vector3d getAngularAcceleraton ()

  Returns exact angular acceleration vector.
- Eigen::Matrix3d getRnb ()

  Get rotation matrix from world to body frame.
- void updateSensors ()

  update all sensors

Public Attributes

- std::map< std::string, std::unique_ptr< Sensor< Eigen::Vector3d > > > sensorsVec3d

  map of sensors that measure values which is 3 element vector
- std::map< std::string, std::unique_ptr< Sensor< double > > > sensors

  map of sensors that measure single value

6.33.1 Constructor & Destructor Documentation

6.33.1.1 Environment()

```
Environment::Environment (
            zmq::context_t * ctx,
            std::string uav_address )
```

Constructor.

Parameters

ctx	zero mq context
uav_address	address to state PUB socket that enviroment should listen

6.33.1.2 ∼Environment()

```
Environment::∼Environment ( )
```

Deconstructor.

6.33.2 Member Function Documentation

6.33.2.1 getAngularAcceleraton()

```
Eigen::Vector3d Environment::getAngularAcceleraton ( )
```

Returns exact angular acceleration vector.

Returns

angular acceleration vector in body frame

6.33.2.2 getAngularVelocity()

```
Eigen::Vector3d Environment::getAngularVelocity ( )
```

Returns exact angular velocity vector.

Returns

angular velocities vector in body frame

6.33.2.3 getLinearAcceleration()

```
Eigen::Vector3d Environment::getLinearAcceleration ( )
```

Returns exact linear acceleration vector.

Returns

linear acceleration vector in body frame

6.33.2.4 getLinearVelocity()

```
Eigen::Vector3d Environment::getLinearVelocity ( )
```

Returns exact linear velocity vector.

Returns

linear velocity vector in body frame

6.33.2.5 getOrientation()

```
Eigen::Vector4d Environment::getOrientation ( )
```

Returns exact orientation vector.

Returns

orientation vector in world frame

6.33.2.6 getPosition()

```
Eigen::Vector3d Environment::getPosition ( )
```

Returns exact postion vector.

Returns

position vector in world frame

6.33.2.7 getRnb()

```
Eigen::Matrix3d Environment::getRnb ( )
```

Get rotation matrix from world to body frame.

Returns

rotation matrix

6.33.2.8 getTime()

```
double Environment::getTime ( )
```

Returns time of simulation.

Returns

simulation time

6.33.2.9 getWorldAngularVelocity()

```
Eigen::Vector3d Environment::getWorldAngularVelocity ( )
```

Returns exact angular velocity vector.

Returns

linear angular vector in world frame

6.33.2.10 getWorldLinearVelocity()

```
Eigen::Vector3d Environment::getWorldLinearVelocity ( )
```

Returns exact linear velocity vector.

Returns

linear velocity vector in world frame

6.33.2.11 updateSensors()

```
void Environment::updateSensors ( )
```

update all sensors

6.33.3 Member Data Documentation

6.33.3.1 sensors

```
std::map<std::string,std::unique_ptr<Sensor<double> > > Environment::sensors
```

map of sensors that measure single value

6.33.3.2 sensorsVec3d

```
std::map<std::string,std::unique_ptr<Sensor<Eigen::Vector3d> > > Environment::sensorsVec3d
```

map of sensors that measure values which is 3 element vector

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

- src/navigation/environment.hpp
- src/navigation/environment.cpp

6.34 GPS Class Reference

Representation of GPS position measure.

```
#include <sensors.hpp>
```

Inheritance diagram for GPS:

Collaboration diagram for GPS:

Public Member Functions

- GPS (Environment &env, double sd, Eigen::Vector3d bias, double refreshTime)
- void update () override

  Update sensor state. Measured value is updated if sensor is ready for next read.

Additional Inherited Members

6.34.1 Detailed Description

Representation of GPS position measure.

6.34.2 Constructor & Destructor Documentation

6.34.2.1 GPS()

```
GPS::GPS (
            Environment & env,
            double sd,
            Eigen::Vector3d bias,
            double refreshTime )
```

6.34.3 Member Function Documentation

6.34.3.1 update()

```
void GPS::update ( )  [override], [virtual]
```

Update sensor state. Measured value is updated if sensor is ready for next read.

Implements Sensor< Eigen::Vector3d >.

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

- src/navigation/sensors.hpp
- src/navigation/sensors.cpp

6.35 GPSVel Class Reference

Representation of GPS velocity measure.

```
#include <sensors.hpp>
```

Inheritance diagram for GPSVel:

Collaboration diagram for GPSVel:

Public Member Functions

- GPSVel (Environment &env, double sd, Eigen::Vector3d bias, double refreshTime)
- void update () override

  Update sensor state. Measured value is updated if sensor is ready for next read.

Additional Inherited Members

6.35.1 Detailed Description

Representation of GPS velocity measure.

6.35.2 Constructor & Destructor Documentation

6.35.2.1 GPSVel()

```
GPSVel::GPSVel (
            Environment & env,
            double sd,
            Eigen::Vector3d bias,
            double refreshTime )
```

6.35.3 Member Function Documentation

6.35.3.1 update()

```
void GPSVel::update ( )  [override], [virtual]
```

Update sensor state. Measured value is updated if sensor is ready for next read.

Implements Sensor< Eigen::Vector3d >.

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

- src/navigation/sensors.hpp
- src/navigation/sensors.cpp

6.36 Gyroscope Class Reference

Representation of gyroscope.

`#include <sensors.hpp>`

Inheritance diagram for Gyroscope:

Collaboration diagram for Gyroscope:

Public Member Functions

- Gyroscope (Environment &env, double sd, Eigen::Vector3d bias, double refreshTime)
- void update () override

  Update sensor state. Measured value is updated if sensor is ready for next read.

Additional Inherited Members

6.36.1 Detailed Description

Representation of gyroscope.

6.36.2 Constructor & Destructor Documentation

6.36.2.1 Gyroscope()

```
Gyroscope::Gyroscope (
            Environment & env,
            double sd,
            Eigen::Vector3d bias,
            double refreshTime )
```

6.36.3 Member Function Documentation

6.36.3.1 update()

```
void Gyroscope::update ( )  [override], [virtual]
```

Update sensor state. Measured value is updated if sensor is ready for next read.

Implements Sensor< Eigen::Vector3d >.

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

- src/navigation/sensors.hpp
- src/navigation/sensors.cpp

6.37 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.

6.37.1 Detailed Description

Hinge connecting aircraft with drives.

6.37.2 Constructor & Destructor Documentation

6.37.2.1 Hinge() [1/3]

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

6.37.2.2 Hinge() [2/3]

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

6.37.2.3 Hinge() [3/3]

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

6.37.3 Member Function Documentation

6.37.3.1 getRot()

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

Get rotattion matrix of orientation change due to hinge.

Returns

rotation matrix

6.37.3.2 operator=()

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

6.37.3.3 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:

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

6.38 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

6.38.1 Detailed Description

Jet rocket engine.

6.38.2 Member Function Documentation

6.38.2.1 getLastThrust()

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

get last calculated thrust

Returns

last calculated thrust

6.38.2.2 getThrust()

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

get thrust in specific time

Parameters

time	timestamp

Returns

thrust value in Newtons

6.38.2.3 start()

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

start jet engine

Parameters

time	timestamp of start

Returns

true if start succesful, false if already started

6.38.3 Member Data Documentation

6.38.3.1 phases

```
int Jet::phases
```

6.38.3.2 thrust

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

6.38.3.3 time

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

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

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

6.39 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 getAmmount ()

  get ammount 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)

6.39.1 Detailed Description

Load of aircraft that can be droped or launched.

6.39.2 Constructor & Destructor Documentation

6.39.2.1 Load() [1/2]

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

6.39.2.2 Load() [2/2]

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

6.39.3 Member Function Documentation

6.39.3.1 getAmmount()

```
int Load::getAmmount ( )  [inline]
```

get ammount of load

Returns

ammount

6.39.3.2 getMass()

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

get mass of load

Returns

mass

6.39.3.3 getOffset()

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

get offset of load

Returns

offset vector

6.39.3.4 operator=()

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

6.39.3.5 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:

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

6.40 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.

6.40.1 Detailed Description

Log vector data with timestamp in file.

6.40.2 Constructor & Destructor Documentation

6.40.2.1 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

6.40.2.2 ∼Logger()

```
Logger::∼Logger ( )
```

Deconstructor.

6.40.3 Member Function Documentation

6.40.3.1 log() [1/2]

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

Log one row.

Parameters

time	timestamp
args	list of doubles

6.40.3.2 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

6.40.3.3 setFmt()

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

Set new format if was not known in constructor.

Parameters

fmt	new format

6.40.3.4 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:

- lib/UAV_common/src/logger/logger.hpp
- lib/UAV_common/src/logger/logger.cpp

6.41 Magnetometer Class Reference

Representation of magnetometer.

```
#include <sensors.hpp>
```

Inheritance diagram for Magnetometer:

Collaboration diagram for Magnetometer:

Public Member Functions

- Magnetometer (Environment &env, double sd, Eigen::Vector3d bias, double refreshTime)
- void update () override

  Update sensor state. Measured value is updated if sensor is ready for next read.

Static Public Attributes

- static const Eigen::Vector3d mag = Eigen::Vector3d(60.0,0.0,0.0)

Additional Inherited Members

6.41.1 Detailed Description

Representation of magnetometer.

6.41.2 Constructor & Destructor Documentation

6.41.2.1 Magnetometer()

```
Magnetometer::Magnetometer (
             Environment & env,
             double sd,
             Eigen::Vector3d bias,
             double refreshTime )
```

6.41.3 Member Function Documentation

6.41.3.1 update()

```
void Magnetometer::update ( )  [override], [virtual]
```

Update sensor state. Measured value is updated if sensor is ready for next read.

Implements Sensor< Eigen::Vector3d >.

6.41.4 Member Data Documentation

6.41.4.1 mag

```
const Eigen::Vector3d Magnetometer::mag = Eigen::Vector3d(60.0,0.0,0.0)  [static]
```

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

- src/navigation/sensors.hpp
- src/navigation/sensors.cpp

6.42 NS Class Reference

Navigation system.

```
#include <NS.hpp>
```

Public Member Functions

- NS (Environment &env)

  Consturctor.
- ∼NS ()

  Deconstructor.
- Eigen::Vector3d getPosition ()

  Returns position estimated by NS.
- Eigen::Vector3d getLinearVelocity ()

  Returns linear velocity estimated by NS.
- Eigen::Vector3d getOrientation ()

  Returns orientation estimated by NS.
- Eigen::Vector3d getAngularVelocity ()

  Returns rates estimated by NS.
- Eigen::Matrix3d getRotationMatrixBodyToWorld ()

  Returns rotation matrix from body to world frame.

6.42.1 Detailed Description

Navigation system.

6.42.2 Constructor & Destructor Documentation

6.42.2.1 NS()

```
NS::NS (
            Environment & env )
```

Consturctor.

Parameters

env	reference to environment, that NS navigate through

6.42.2.2 ∼NS()

```
NS::∼NS ( )
```

Deconstructor.

6.42.3 Member Function Documentation

6.42.3.1 getAngularVelocity()

```
Eigen::Vector3d NS::getAngularVelocity ( )
```

Returns rates estimated by NS.

Returns

angular velocity vector (roll rate, pitch rate, yaw rate) in body frame

6.42.3.2 getLinearVelocity()

```
Eigen::Vector3d NS::getLinearVelocity ( )
```

Returns linear velocity estimated by NS.

Returns

linear velocity vector in world frame

6.42.3.3 getOrientation()

```
Eigen::Vector3d NS::getOrientation ( )
```

Returns orientation estimated by NS.

Returns

orientation vector (RPY) in world frame

6.42.3.4 getPosition()

```
Eigen::Vector3d NS::getPosition ( )
```

Returns position estimated by NS.

Returns

position vector in world frame

6.42.3.5 getRotationMatrixBodyToWorld()

```
Eigen::Matrix3d NS::getRotationMatrixBodyToWorld ( )
```

Returns rotation matrix from body to world frame.

Returns

rotation matrix

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

- src/navigation/NS.hpp
- src/navigation/NS.cpp

6.43 ODE Class Reference

Ordinal differencial equation solver.

```
#include <ode.hpp>
```

Inheritance diagram for ODE:

Public Types

- enum ODEMethod {
  Euler , Heun , RK4 , PC2 ,
  PC4 , 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.

6.43.1 Detailed Description

Ordinal differencial equation solver.

6.43.2 Member Enumeration Documentation

6.43.2.1 ODEMethod

```
enum ODE::ODEMethod
```

Supported solving method.

Enumerator

Euler	
Heun	
RK4	
PC2	
PC4	
NONE	

6.43.3 Constructor & Destructor Documentation

6.43.3.1 ODE()

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

Constructor.

6.43.3.2 ∼ODE()

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

Virtual deconstructor.

6.43.4 Member Function Documentation

6.43.4.1 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

6.43.4.2 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

6.43.4.3 getMicrosteps() [1/2]

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

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

Returns

microsteps

6.43.4.4 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

6.43.4.5 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_PC4, ODE_PC2, ODE_RK4, ODE_Heun, and ODE_Euler.

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

- lib/UAV_common/src/ode/ode.hpp
- lib/UAV_common/src/ode/ode.cpp

6.44 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::VectorXd)> rhs_fun, double h) override

  One step of explicit solving algorithm.

Additional Inherited Members

6.44.1 Detailed Description

Explicit Euler algorithm.

6.44.2 Constructor & Destructor Documentation

6.44.2.1 ODE_Euler()

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

6.44.3 Member Function Documentation

6.44.3.1 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:

- lib/UAV_common/src/ode/ode_impl.hpp

6.45 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::VectorXd)> rhs_fun, double h) override

  One step of explicit solving algorithm.

Additional Inherited Members

6.45.1 Detailed Description

Second order explicit Heun algorithm.

6.45.2 Constructor & Destructor Documentation

6.45.2.1 ODE_Heun()

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

6.45.3 Member Function Documentation

6.45.3.1 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:

- lib/UAV_common/src/ode/ode_impl.hpp

6.46 ODE_PC2 Class Reference

Second order predictor-corrector method Second order Adams-bashforth and Adams-moulton.

```
#include <ode_impl.hpp>
```

Inheritance diagram for ODE_PC2:

Collaboration diagram for ODE_PC2:

Public Member Functions

- ODE_PC2 ()
- Eigen::VectorXd step (double t, Eigen::VectorXd y0, std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun, double h) override

  One step of explicit solving algorithm.

Additional Inherited Members

6.46.1 Detailed Description

Second order predictor-corrector method Second order Adams-bashforth and Adams-moulton.

6.46.2 Constructor & Destructor Documentation

6.46.2.1 ODE_PC2()

```
ODE_PC2::ODE_PC2 ( )  [inline]
```

6.46.3 Member Function Documentation

6.46.3.1 step()

```
Eigen::VectorXd ODE_PC2::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:

- lib/UAV_common/src/ode/ode_impl.hpp

6.47 ODE_PC4 Class Reference

Fourth order predictor-corrector method Fourth order Adams-bashforth and Adams-moulton.

```
#include <ode_impl.hpp>
```

Inheritance diagram for ODE_PC4:

Collaboration diagram for ODE_PC4:

Public Member Functions

- ODE_PC4 ()
- Eigen::VectorXd step (double t, Eigen::VectorXd y0, std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun, double h) override

  One step of explicit solving algorithm.

Additional Inherited Members

6.47.1 Detailed Description

Fourth order predictor-corrector method Fourth order Adams-bashforth and Adams-moulton.

6.47.2 Constructor & Destructor Documentation

6.47.2.1 ODE_PC4()

```
ODE_PC4::ODE_PC4 ( )  [inline]
```

6.47.3 Member Function Documentation

6.47.3.1 step()

```
Eigen::VectorXd ODE_PC4::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:

- lib/UAV_common/src/ode/ode_impl.hpp

6.48 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::VectorXd)> rhs_fun, double h) override

    One step of explicit solving algorithm.

Additional Inherited Members

6.48.1 Detailed Description

Fourth order Runge Kutta algorithm.

6.48.2 Constructor & Destructor Documentation

6.48.2.1 ODE_RK4()

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

6.48.3 Member Function Documentation

6.48.3.1 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:

- lib/UAV_common/src/ode/ode_impl.hpp

6.49 ODETest Class Reference

Inheritance diagram for ODETest:

Collaboration diagram for ODETest:

Protected Member Functions

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

6.49.1 Member Function Documentation

6.49.1.1 SetUp()

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

6.49.1.2 TearDown()

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

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

- lib/UAV_common/src/ode/ode_test.cpp

6.50 Params Class Reference

Simulation parameters.

```
#include <params.hpp>
```

Public Member Functions

- Params ()

  Constructor.
- Params (const Params &)=delete
- Params & operator= (const Params &)=delete
- Params (Params &&)=delete
- ∼Params ()

  Deconstructor.

Static Public Member Functions

- static const Params ∗ getSingleton ()

  Get singleton of Params.

Public Attributes

- double STEP_TIME

  Step time of simulation. Step of ODE solving methods.

6.50.1 Detailed Description

Simulation parameters.

6.50.2 Constructor & Destructor Documentation

6.50.2.1 Params() [1/3]

```
Params::Params ( )
```

Constructor.

6.50.2.2 Params() [2/3]

```
Params::Params (
            const Params & )  [delete]
```

6.50.2.3 Params() [3/3]

```
Params::Params (
            Params && )  [delete]
```

6.50.2.4 ∼Params()

```
Params::∼Params ( )
```

Deconstructor.

6.50.3 Member Function Documentation

6.50.3.1 getSingleton()

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

Get singleton of Params.

Returns

const pointer to Params instance. Return nullptr if not initialized

6.50.3.2 operator=()

```
Params& Params::operator= (
            const Params & )  [delete]
```

6.50.4 Member Data Documentation

6.50.4.1 STEP_TIME

```
double Params::STEP_TIME
```

Step time of simulation. Step of ODE solving methods.

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

- src/params.hpp
- src/params.cpp

6.51 controllers::PID Class Reference

```
#include <PID.hpp>
```

Inheritance diagram for controllers::PID:

Collaboration diagram for controllers::PID:

Public Types

- enum class AntiWindUpMode { NONE , CLAMPING }

  Methods of handling windup in controller.

Public Member Functions

- PID (double Kp, double Ki, double Kd, double Kff=0.0, double min=-std::numeric_limits< double >::max(), double max=std::numeric_limits< double >::max(), AntiWindUpMode antiWindUp=AntiWindUpMode::CLAMPING)

  Constructor with all PID controller parameters.
- PID (rapidxml::xml_node<> ∗controller_node)

  Construct controller with parameters from xml.
- double calc (double desired, double actual, double dt) override

  calc output of controller with specific time step
- void clear () override

  clear internal state
- std::unique_ptr< Controller > clone () const override

  virtual clone method

Additional Inherited Members

6.51.1 Member Enumeration Documentation

6.51.1.1 AntiWindUpMode

```
enum controllers::PID::AntiWindUpMode  [strong]
```

Methods of handling windup in controller.

Enumerator

NONE	
CLAMPING	

6.51.2 Constructor & Destructor Documentation

6.51.2.1 PID() [1/2]

```
PID::PID (
            double Kp,
            double Ki,
            double Kd,
            double Kff = 0.0,
            double min = -std::numeric_limits<double>::max(),
            double max = std::numeric_limits<double>::max(),
            AntiWindUpMode antiWindUp = AntiWindUpMode::CLAMPING )
```

Constructor with all PID controller parameters.

Parameters

Kp	P term
Ki	I term
Kd	D term
Kff	FF term
min	saturation - lower range limit
max	saturation - upper range limit
antiWindUp	antiwindup method

6.51.2.2 PID() [2/2]

```
PID::PID (
            rapidxml::xml_node<> * controller_node )
```

Construct controller with parameters from xml.

Parameters

controller_node	xml node with controller params

6.51.3 Member Function Documentation

6.51.3.1 calc()

```
double controllers::PID::calc (
             double desired,
             double actual,
             double dt )  [override], [virtual]
```

calc output of controller with specific time step

Parameters

desired	input of controller, desired value
actual	measured actual value
dt	time step

Returns

output of controller

Implements Controller.

6.51.3.2 clear()

```
void PID::clear ( )  [override], [virtual]
```

clear internal state

Implements Controller.

6.51.3.3 clone()

```
std::unique_ptr< Controller > PID::clone ( ) const  [override], [virtual]
```

virtual clone method

Implements Controller.

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

- lib/UAV_common/src/controllers/impl/PID.hpp
- lib/UAV_common/src/controllers/impl/PID.cpp

6.52 controllers::PID_Discrete Class Reference

```
#include <PID_discrete.hpp>
```

Inheritance diagram for controllers::PID_Discrete:

Collaboration diagram for controllers::PID_Discrete:

Public Member Functions

- PID_Discrete (double Kp, double Ki, double Kd, double Kff=0.0, double N=100.0, double min=-std::numeric_limits< double >::max(), double max=std::numeric_limits< double >::max())

  Constructor with all PID controller parameters.
- PID_Discrete (rapidxml::xml_node<> *controller_node)

  Construct controller with parameters from xml.
- double calc (double desired, double actual, double dt) override

  calc output of controller with specific time step
- void set_dt (double dt) override

  Set new time step.
- void clear () override

  clear internal state
- std::unique_ptr< Controller > clone () const override

  virtual clone method

Additional Inherited Members

6.52.1 Constructor & Destructor Documentation

6.52.1.1 PID_Discrete() [1/2]

```
controllers::PID_Discrete::PID_Discrete (
                double Kp,
                double Ki,
                double Kd,
                double Kff = 0.0,
                double N = 100.0,
                double min = -std::numeric_limits<double>::max(),
                double max = std::numeric_limits<double>::max() )
```

Constructor with all PID controller parameters.

Parameters

Kp	P term
Ki	I term
Kd	D term
Kff	FF term
min	saturation - lower range limit
max	saturation - upper range limit
antiWindUp	antiwindup method

6.52.1.2 PID_Discrete() [2/2]

```
controllers::PID_Discrete::PID_Discrete (
             rapidxml::xml_node<> * controller_node )
```

Construct controller with parameters from xml.

Parameters

controller_node	xml node with controller params

6.52.2 Member Function Documentation

6.52.2.1 calc()

```
double controllers::PID_Discrete::calc (
             double desired,
             double actual,
             double dt )  [override], [virtual]
```

calc output of controller with specific time step

Parameters

desired	input of controller, desired value
actual	measured actual value
dt	time step

Returns

output of controller

Implements Controller.

6.52.2.2 clear()

```
void controllers::PID_Discrete::clear ( )  [override], [virtual]
```

clear internal state

Implements Controller.

6.52.2.3 clone()

```
std::unique_ptr< Controller > controllers::PID_Discrete::clone ( ) const  [override], [virtual]
```

virtual clone method

Implements Controller.

6.52.2.4 set_dt()

```
void controllers::PID_Discrete::set_dt (
            double dt )  [override], [virtual]
```

Set new time step.

Parameters

dt	new time step

Reimplemented from Controller.

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

- lib/UAV_common/src/controllers/impl/PID_discrete.hpp
- lib/UAV_common/src/controllers/impl/PID_discrete.cpp

6.53 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

6.53.1 Detailed Description

Rotor engine with controlled speed.

6.53.2 Member Data Documentation

6.53.2.1 direction

```
int Rotor::direction
```

6.53.2.2 forceCoff

```
double Rotor::forceCoff
```

6.53.2.3 hoverSpeed

```
double Rotor::hoverSpeed
```

6.53.2.4 maxSpeed

```
double Rotor::maxSpeed
```

6.53.2.5 timeConstant

```
double Rotor::timeConstant
```

6.53.2.6 torqueCoff

```
double Rotor::torqueCoff
```

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

- lib/UAV_common/src/components/drive.hpp

6.54 Sensor< T > Class Template Reference

Sensors base class.

```
#include <sensors.hpp>
```

Collaboration diagram for Sensor< T >:

Public Member Functions

- Sensor (Environment &env, double sd, T bias, std::string path, std::string fmt, double refreshTime)

  Constructor.
- virtual void update ()=0

  Update sensor state. Measured value is updated if sensor is ready for next read.
- T getReading ()

  Returns recent measure.
- double getSd ()

  Returns standard deviation.
- bool isReady ()

  Checks if sensor is ready.

Protected Member Functions

- bool shouldUpdate ()

  Checks if sensor should measure next value.
- double error ()

Protected Attributes

- Environment & env
- T value
- double refreshTime
- double lastUpdate
- std::atomic_bool ready
- std::normal_distribution< double > dist
- T bias
- Logger logger

Static Protected Attributes

- static std::mt19937 gen = std::mt19937(std::random_device()())

6.54.1 Detailed Description

template<class T>
class Sensor< T >

Sensors base class.

Template Parameters

T	type of data read by sensor

6.54.2 Constructor & Destructor Documentation

6.54.2.1 Sensor()

```
template<class T >
Sensor< T >::Sensor (
            Environment & env,
            double sd,
            T bias,
            std::string path,
            std::string fmt,
            double refreshTime )
```

Constructor.

Parameters

env	reference to environment sensor measures
sd	standard deviation of reading
bias	reading bias
path	path where sensor logs are saved
fmt	header of log file
refreshTime	sample period

6.54.3 Member Function Documentation

6.54.3.1 error()

```
template<class T >
double Sensor< T >::error  [protected]
```

6.54.3.2 getReading()

```
template<class T >
T Sensor< T >::getReading ( )  [inline]
```

Returns recent measure.

Returns

sensor measure

6.54.3.3 getSd()

```
template<class T >
double Sensor< T >::getSd ( )  [inline]
```

Returns standard deviation.

Returns

standard deviation

6.54.3.4 isReady()

```
template<class T >
bool Sensor< T >::isReady ( )  [inline]
```

Checks if sensor is ready.

Returns

true if sensor is ready

6.54.3.5 shouldUpdate()

```
template<class T >
bool Sensor< T >::shouldUpdate  [protected]
```

Checks if sensor should measure next value.

Returns

true if sensor is ready for next measure

6.54.3.6 update()

```
template<class T >
virtual void Sensor< T >::update ( )  [pure virtual]
```

Update sensor state. Measured value is updated if sensor is ready for next read.

Implemented in GPSVel, GPS, Barometer, Magnetometer, Gyroscope, and Accelerometer.

6.54.4 Member Data Documentation

6.54.4.1 bias

```
template<class T >
T Sensor< T >::bias  [protected]
```

6.54.4.2 dist

```
template<class T >
std::normal_distribution<double> Sensor< T >::dist  [protected]
```

6.54.4.3 env

```
template<class T >
Environment& Sensor< T >::env  [protected]
```

6.54.4.4 gen

```
template<class T >
std::mt19937 Sensor< T >::gen = std::mt19937(std::random_device()())  [static], [protected]
```

6.54.4.5 lastUpdate

```
template<class T >
double Sensor< T >::lastUpdate  [protected]
```

6.54.4.6 logger

```
template<class T >
Logger Sensor< T >::logger  [protected]
```

6.54.4.7 ready

```
template<class T >
std::atomic_bool Sensor< T >::ready  [protected]
```

6.54.4.8 refreshTime

```
template<class T >
double Sensor< T >::refreshTime  [protected]
```

6.54.4.9 value

```
template<class T >
T Sensor< T >::value  [protected]
```

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

- src/navigation/sensors.hpp
- src/navigation/sensors.cpp

6.55 SensorParams Struct Reference

Base parameters of a sensor.

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

Public Attributes

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

6.55.1 Detailed Description

Base parameters of a sensor.

6.55.2 Member Data Documentation

6.55.2.1 bias

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

6.55.2.2 name

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

6.55.2.3 refreshTime

```
double SensorParams::refreshTime
```

6.55.2.4 sd

```
double SensorParams::sd
```

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

- lib/UAV_common/src/components/navi.hpp

6.56 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

6.56.1 Detailed Description

Simulation of real-time synchronized loop.

6.56.2 Constructor & Destructor Documentation

6.56.2.1 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

6.56.3 Member Function Documentation

6.56.3.1 go() [1/2]

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

start infinite loop

6.56.3.2 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:

- lib/UAV_common/src/timed_loop/timed_loop.hpp
- lib/UAV_common/src/timed_loop/timed_loop.cpp

6.57 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
- Eigen::Vector3d target
- 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, std::unique_ptr< Controller > > controllers
- 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

6.57.1 Detailed Description

Parsed UAV configuration from XML.

6.57.2 Constructor & Destructor Documentation

6.57.2.1 UAVparams()

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

Initialize default data.

6.57.2.2 ∼UAVparams()

```
UAVparams::∼UAVparams ( )
```

6.57.3 Member Function Documentation

6.57.3.1 getRotorHoverSpeeds()

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

6.57.3.2 getRotorMaxSpeeds()

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

6.57.3.3 getRotorTimeContants()

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

6.57.3.4 getSingleton()

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

6.57.3.5 loadConfig()

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

6.57.4 Member Data Documentation

6.57.4.1 aero_coffs

```
AeroCoefficients UAVparams::aero_coffs
```

6.57.4.2 ahrs

```
AHRSParams UAVparams::ahrs
```

6.57.4.3 ammo

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

6.57.4.4 cargo

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

6.57.4.5 controllers

```
std::map<std::string,std::unique_ptr<Controller> > UAVparams::controllers
```

6.57.4.6 ekf

```
EKFScalers UAVparams::ekf
```

6.57.4.7 initialMode

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

6.57.4.8 initialOrientation

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

6.57.4.9 initialPosition

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

6.57.4.10 initialVelocity

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

6.57.4.11 instantRun

```
bool UAVparams::instantRun
```

6.57.4.12 Ix

```
double UAVparams::Ix
```

6.57.4.13 Ixy

```
double UAVparams::Ixy
```

6.57.4.14 Ixz

```
double UAVparams::Ixz
```

6.57.4.15 Iy

```
double UAVparams::Iy
```

6.57.4.16 Iyz

```
double UAVparams::Iyz
```

6.57.4.17 Iz

```
double UAVparams::Iz
```

6.57.4.18 jets

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

6.57.4.19 m

```
double UAVparams::m
```

6.57.4.20 name

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

6.57.4.21 noOfAmmo

```
int UAVparams::noOfAmmo
```

6.57.4.22 noOfCargo

```
int UAVparams::noOfCargo
```

6.57.4.23 noOfJets

```
int UAVparams::noOfJets
```

6.57.4.24 noOfRotors

```
int UAVparams::noOfRotors
```

6.57.4.25 rotorMixer

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

6.57.4.26 rotors

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

6.57.4.27 sensors

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

6.57.4.28 surfaceMixer

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

6.57.4.29 surfaces

```
ControlSurfaces UAVparams::surfaces
```

6.57.4.30 target

```
Eigen::Vector3d UAVparams::target
```

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

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

6.58 controllers::ZTransform Class Reference

```
#include <z_trans.hpp>
```

Inheritance diagram for controllers::ZTransform:

Collaboration diagram for controllers::ZTransform:

Public Member Functions

- ZTransform (const std::vector< double > &num, const std::vector< double > &den, double min=-std::numeric_limits< double >::max(), double max=std::numeric_limits< double >::max())

  Constructorof Z-Transform controller.
- ZTransform (rapidxml::xml_node<> ∗controller_node)

  Construct controller with parameters from xml.
- double calc (double desired, double actual, [[maybe_unused]] double dt) override

  calc output of controller
- void clear () override

  clear internal state
- std::unique_ptr< Controller > clone () const override

  virtual clone method

Additional Inherited Members

6.58.1 Constructor & Destructor Documentation

6.58.1.1 ZTransform() [1/2]

```
ZTransform::ZTransform (
             const std::vector< double > & num,
             const std::vector< double > & den,
             double min = -std::numeric_limits<double>::max(),
             double max = std::numeric_limits<double>::max() )
```

Constructorof Z-Transform controller.

Parameters

min	saturation - lower range limit
max	saturation - upper range limit

6.58.1.2 ZTransform() [2/2]

```
controllers::ZTransform::ZTransform (
             rapidxml::xml_node<> * controller_node )
```

Construct controller with parameters from xml.

Parameters

controller_node	xml node with controller params

6.58.2 Member Function Documentation

6.58.2.1 calc()

```
double ZTransform::calc (
             double desired,
             double actual,
             [[maybe_unused] ] double dt )  [override]
```

calc output of controller

Parameters

desired	input of controller, desired value
actual	measured actual value

Returns

output of controller

6.58.2.2 clear()

```
void ZTransform::clear ( )  [override], [virtual]
```

clear internal state

Implements Controller.

6.58.2.3 clone()

```
std::unique_ptr< Controller > ZTransform::clone ( ) const  [override], [virtual]
```

virtual clone method

Implements Controller.

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

- lib/UAV_common/src/controllers/impl/z_trans.hpp
- lib/UAV_common/src/controllers/impl/z_trans.cpp

6.59 controllers::ZTransformStatic< N, D > Class Template Reference

```
#include <z_trans.hpp>
```

Inheritance diagram for controllers::ZTransformStatic< N, D >:

Collaboration diagram for controllers::ZTransformStatic< N, D >:

Public Member Functions

- ZTransformStatic (const std::array< double, N > &num, const std::array< double, D > &den, double min=-std::numeric_limits< double >::max(), double max=std::numeric_limits< double >::max())

  Constructorof Z-Transform controller.
- ZTransformStatic (rapidxml::xml_node<> ∗controller_node)=delete

  Construct controller with parameters from xml.
- double calc (double desired, double actual, [[maybe_unused]] double dt) override

  calc output of controller
- void clear () override

  clear internal state
- std::unique_ptr< Controller > clone () const override

  virtual clone method

Additional Inherited Members

6.59.1 Constructor & Destructor Documentation

6.59.1.1 ZTransformStatic() [1/2]

```
template<unsigned int N, unsigned int D>
controllers::ZTransformStatic< N, D >::ZTransformStatic (
             const std::array< double, N > & num,
             const std::array< double, D > & den,
             double min = -std::numeric_limits<double>::max(),
             double max = std::numeric_limits<double>::max() )
```

Constructorof Z-Transform controller.

Parameters

min	saturation - lower range limit
max	saturation - upper range limit

6.59.1.2 ZTransformStatic() [2/2]

```
template<unsigned int N, unsigned int D>
controllers::ZTransformStatic< N, D >::ZTransformStatic (
            rapidxml::xml_node<> * controller_node )  [delete]
```

Construct controller with parameters from xml.

Parameters

controller_node	xml node with controller params

6.59.2 Member Function Documentation

6.59.2.1 calc()

```
template<unsigned int N, unsigned int D>
double controllers::ZTransformStatic< N, D >::calc (
            double desired,
            double actual,
            [[maybe_unused] ] double dt )  [override]
```

calc output of controller

Parameters

desired	input of controller, desired value
actual	measured actual value

Returns

output of controller

6.59.2.2 clear()

```
template<unsigned int N, unsigned int D>
void controllers::ZTransformStatic< N, D >::clear  [override], [virtual]
```

clear internal state

Implements Controller.

6.59.2.3 clone()

```
template<unsigned int N, unsigned int D>
std::unique_ptr< Controller > controllers::ZTransformStatic< N, D >::clone  [override], [virtual]
```

virtual clone method

Implements Controller.

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

- lib/UAV_common/src/controllers/impl/z_trans.hpp

Chapter 7

File Documentation

7.1 build/CMakeFiles/3.22.1/CompilerIdC/CMakeCCompilerId.c File Reference

Macros

- #define __has_include(x) 0
- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X) #X
- #define STRINGIFY(X) STRINGIFY_HELPER(X)
- #define PLATFORM_ID
- #define ARCHITECTURE_ID
- #define DEC(n)
- #define HEX(n)
- #define C_VERSION

Functions

- int main (int argc, char ∗argv[ ])

Variables

- char const ∗ info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const ∗ info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const ∗ info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char ∗ info_language_standard_default
- const char ∗ info_language_extensions_default

7.1.1 Macro Definition Documentation

7.1.1.1 __has_include

```
#define __has_include(
              x ) 0
```

7.1.1.2 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

7.1.1.3 C_VERSION

```
#define C_VERSION
```

7.1.1.4 COMPILER_ID

```
#define COMPILER_ID ""
```

7.1.1.5 DEC

```
#define DEC(
              n )
```

Value:

```
  (’0’ + (((n) / 10000000)%10)), \
  (’0’ + (((n) / 1000000)%10)),  \
  (’0’ + (((n) / 100000)%10)),   \
  (’0’ + (((n) / 10000)%10)),    \
  (’0’ + (((n) / 1000)%10)),     \
  (’0’ + (((n) / 100)%10)),      \
  (’0’ + (((n) / 10)%10)),       \
  (’0’ +  ((n) % 10))
```

7.1.1.6 HEX

```
#define HEX(
              n )
```

Value:

```
  (’0’ + ((n)»28 & 0xF)), \
  (’0’ + ((n)»24 & 0xF)), \
  (’0’ + ((n)»20 & 0xF)), \
  (’0’ + ((n)»16 & 0xF)), \
  (’0’ + ((n)»12 & 0xF)), \
  (’0’ + ((n)»8  & 0xF)), \
  (’0’ + ((n)»4  & 0xF)), \
  (’0’ + ((n)     & 0xF))
```

7.1.1.7 PLATFORM_ID

```
#define PLATFORM_ID
```

7.1.1.8 STRINGIFY

```
#define STRINGIFY(
              X ) STRINGIFY_HELPER(X)
```

7.1.1.9 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
              X ) #X
```

7.1.2 Function Documentation

7.1.2.1 main()

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

7.1.3 Variable Documentation

7.1.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

7.1.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

7.1.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

Initial value:

```
= "INFO" ":" "extensions_default["
  "OFF"
"]"
```

7.1.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

Initial value:

```
=
  "INFO" ":" "standard_default[" C_VERSION "]"
```

7.1.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

7.2 build/CMakeFiles/3.22.1/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference

Macros

- #define __has_include(x) 0
- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X) #X
- #define STRINGIFY(X) STRINGIFY_HELPER(X)
- #define PLATFORM_ID
- #define ARCHITECTURE_ID
- #define DEC(n)
- #define HEX(n)
- #define CXX_STD __cplusplus

Functions

- int main (int argc, char ∗argv[ ])

Variables

- char const ∗ info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const ∗ info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const ∗ info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char ∗ info_language_standard_default
- const char ∗ info_language_extensions_default

7.2.1 Macro Definition Documentation

7.2.1.1 __has_include

```
#define __has_include(
              x ) 0
```

7.2.1.2 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

7.2.1.3 COMPILER_ID

```
#define COMPILER_ID ""
```

7.2.1.4 CXX_STD

```
#define CXX_STD __cplusplus
```

7.2.1.5 DEC

```
#define DEC(
              n )
```

Value:

```
  (’0’ + (((n) / 10000000)%10)), \
  (’0’ + (((n) / 1000000)%10)),  \
  (’0’ + (((n) / 100000)%10)),   \
  (’0’ + (((n) / 10000)%10)),    \
  (’0’ + (((n) / 1000)%10)),     \
  (’0’ + (((n) / 100)%10)),      \
  (’0’ + (((n) / 10)%10)),       \
  (’0’ +  ((n) % 10))
```

7.2.1.6 HEX

```
#define HEX(
             n )
```

Value:

```
('0' + ((n)»28 & 0xF)), \
('0' + ((n)»24 & 0xF)), \
('0' + ((n)»20 & 0xF)), \
('0' + ((n)»16 & 0xF)), \
('0' + ((n)»12 & 0xF)), \
('0' + ((n)»8  & 0xF)), \
('0' + ((n)»4  & 0xF)), \
('0' + ((n)     & 0xF))
```

7.2.1.7 PLATFORM_ID

```
#define PLATFORM_ID
```

7.2.1.8 STRINGIFY

```
#define STRINGIFY(
             X ) STRINGIFY_HELPER(X)
```

7.2.1.9 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
             X ) #X
```

7.2.2 Function Documentation

7.2.2.1 main()

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

7.2.3 Variable Documentation

7.2.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

7.2.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

7.2.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

Initial value:

```
= "INFO" ":" "extensions_default["
  "OFF"
"]"
```

7.2.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

Initial value:

```
= "INFO" ":" "standard_default["
  "98"
"]"
```

7.2.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

7.3 build/CMakeFiles/controller.dir/src/communication/control.cpp.o.d File Reference

7.4 build/CMakeFiles/controller.dir/src/communication/control_recv.cpp.o.d File Reference

7.5 build/CMakeFiles/controller.dir/src/communication/control_send.cpp.o.d File Reference

7.6 build/CMakeFiles/controller.dir/src/controller/controller.cpp.o.d File Reference

7.7 build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/controller.cpp.o.d File Reference

7.8 build/CMakeFiles/controller.dir/src/controller/controller_loop.cpp.o.d File Reference

7.9 build/CMakeFiles/controller.dir/src/controller/mixers.cpp.o.d File Reference

7.10 build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_FACRO.cpp.o.d File Reference

7.11 build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_FANGLE.cpp.o.d File Reference

7.12 build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_FMANUAL.cpp.o.d File Reference

7.13 build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_NONE.cpp.o.d File Reference

7.14 build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_QACRO.cpp.o.d File Reference

7.15 build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_QANGLE.cpp.o.d File Reference

7.16 build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_QPOS.cpp.o.d File Reference

7.17 build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_RANGLE.cpp.o.d File Reference

7.18 build/CMakeFiles/controller.dir/src/controller/modes/controller_

```
#include "../src/ode/ode.hpp"
#include "../src/controllers/controller.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: This graph shows which files directly or indirectly include this file:

7.47 lib/UAV_common/scripts/controller_plots.m File Reference

Functions

- plot (x, y, 'DisplayName', csvFiles(i).name)
- end xlabel ('Czas')
- ylabel ('Wartość regulowana')
- title ('Test regulatorów')
- legend ('Location', 'Best')

Variables

- clc
- clear folderPath = '../build/controller_plots/'
- csvFiles = dir(fullfile(folderPath, '∗.csv'))
- figure
- hold on
- for i
- data = readmatrix(filePath)
- x = data(:, 1)
- y = data(:, 2)
- hold off

7.47.1 Function Documentation

7.47.1.1 legend()

```
legend (
            'Location' ,
            'Best'  )
```

7.47.1.2 plot()

```
plot (
             x ,
             y ,
             'DisplayName' ,
             csvFiles(i).  name )
```

7.47.1.3 title()

```
title (
             'Test regulatorów'  )
```

7.47.1.4 xlabel()

```
end xlabel (
             'Czas'  )
```

7.47.1.5 ylabel()

```
ylabel (
             'Wartość regulowana'  )
```

7.47.2 Variable Documentation

7.47.2.1 clc

```
clc
```

7.47.2.2 csvFiles

```
csvFiles = dir(fullfile(folderPath, '*.csv'))
```

7.47.2.3 data

```
data = readmatrix(filePath)
```

7.47.2.4 figure

```
figure
```

7.47.2.5 folderPath

```
clear folderPath = '../build/controller_plots/'
```

7.47.2.6 i

```
for i
```

Initial value:

```
= 1:length(csvFiles)
    filePath = fullfile(folderPath, csvFiles(i).name)
```

7.47.2.7 off

```
hold off
```

7.47.2.8 on

```
hold on
```

7.47.2.9 x

```
x = data(:, 1)
```

7.47.2.10 y

```
y = data(:, 2)
```

7.48 lib/UAV_common/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.

7.49 lib/UAV_common/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:

7.50 lib/UAV_common/src/components/control_surfaces.cpp File Reference

```
#include "control_surfaces.hpp"
```

Include dependency graph for control_surfaces.cpp:

7.51 lib/UAV_common/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.

7.52 lib/UAV_common/src/components/drive.cpp File Reference

```
#include "drive.hpp"
```

Include dependency graph for drive.cpp:

7.53 lib/UAV_common/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.

7.54 lib/UAV_common/src/components/hinge.cpp File Reference

```
#include "hinge.hpp"
```

Include dependency graph for hinge.cpp:

Functions

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

7.54.1 Function Documentation

7.54.1.1 asSkewMatrix()

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

7.55 lib/UAV_common/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.

7.56 lib/UAV_common/src/components/loads.cpp File Reference

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

Include dependency graph for loads.cpp:

7.57 lib/UAV_common/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

7.58 lib/UAV_common/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.

7.59 lib/UAV_common/src/controllers/controller.cpp File Reference

```
#include "controller.hpp"
#include "impl/PID.hpp"
#include "impl/PID_discrete.hpp"
#include "impl/bang_bang.hpp"
#include "impl/double_setpoint.hpp"
#include "impl/z_trans.hpp"
#include <cstring>
#include <stdexcept>
```

Include dependency graph for controller.cpp:

7.60 src/controller/controller.cpp File Reference

```
#include "controller.hpp"
#include <iostream>
#include "../defines.hpp"
#include "../params.hpp"
```

Include dependency graph for controller.cpp:

7.61 lib/UAV_common/src/controllers/controller.hpp File Reference

```
#include <memory>
#include "rapidxml/rapidxml.hpp"
```

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

Classes

- class Controller

7.62 src/controller/controller.hpp File Reference

```
#include <map>
#include <string>
#include <Eigen/Dense>
#include <functional>
#include <optional>
#include "../navigation/NS.hpp"
#include "../navigation/environment.hpp"
#include "mixers.hpp"
#include "controller_mode.hpp"
#include "controller_loop.hpp"
#include "common.hpp"
#include "../communication/control.hpp"
```

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

Classes

- class ControlSystem

  Central controller class.

7.63 lib/UAV_common/src/controllers/controller_test.cpp File Reference

```
#include "impl/PID.hpp"
#include "impl/PID_discrete.hpp"
#include "impl/bang_bang.hpp"
#include "impl/double_setpoint.hpp"
#include "impl/z_trans.hpp"
#include <gtest/gtest.h>
#include <memory>
#include <filesystem>
#include <fstream>
```

Include dependency graph for controller_test.cpp:

Classes

- class ControllerTest

Functions

- std::vector< std::shared_ptr< Controller > > getMethodsToTest ()
- TEST_P (ControllerTest, TestConstFunction)
- TEST_P (ControllerTest, SimpleObjectControl)
- INSTANTIATE_TEST_SUITE_P (TestDerivedClasses, ControllerTest, testing::ValuesIn(getMethodsToTest()))
- int main (int argc, char **argv)

Variables

- constexpr bool plot = true
- constexpr auto plot_directory_name = "controller_plots"

7.63.1 Function Documentation

7.63.1.1 getMethodsToTest()

```
std::vector<std::shared_ptr<Controller> > getMethodsToTest ( )
```

7.63.1.2 INSTANTIATE_TEST_SUITE_P()

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

7.63.1.3 main()

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

7.63.1.4 TEST_P() [1/2]

```
TEST_P (
            ControllerTest ,
            SimpleObjectControl  )
```

7.63.1.5 TEST_P() [2/2]

```
TEST_P (
            ControllerTest ,
            TestConstFunction  )
```

7.63.2 Variable Documentation

7.63.2.1 plot

```
constexpr bool plot = true  [constexpr]
```

7.63.2.2 plot_directory_name

```
constexpr auto plot_directory_name = "controller_plots"  [constexpr]
```

7.64 lib/UAV_common/src/controllers/impl/bang_bang.cpp File Reference

```
#include "bang_bang.hpp"
#include <cstring>
#include <string>
```

Include dependency graph for bang_bang.cpp:

7.65 lib/UAV_common/src/controllers/impl/bang_bang.hpp File Reference

```
#include <memory>
#include "rapidxml/rapidxml.hpp"
#include "../controller.hpp"
```

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

Classes

- class controllers::BangBang

Namespaces

- controllers

7.66 lib/UAV_common/src/controllers/impl/double_setpoint.cpp File Reference

```
#include "double_setpoint.hpp"
#include <cstring>
#include <string>
```

Include dependency graph for double_setpoint.cpp:

7.67 lib/UAV_common/src/controllers/impl/double_setpoint.hpp File Reference

```
#include <memory>
#include "rapidxml/rapidxml.hpp"
#include "../controller.hpp"
```

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

Classes

- class controllers::DoubleSetpoint

Namespaces

- controllers

7.68 lib/UAV_common/src/controllers/impl/PID.cpp File Reference

```
#include "PID.hpp"
#include <algorithm>
#include <cstring>
#include <string>
#include <stdexcept>
```

Include dependency graph for PID.cpp:

7.69 lib/UAV_common/src/controllers/impl/PID.hpp File Reference

```
#include <memory>
#include <limits>
#include "rapidxml/rapidxml.hpp"
#include "../controller.hpp"
#include "z_trans.hpp"
```

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

Classes

- class controllers::PID

Namespaces

- controllers

7.70 lib/UAV_common/src/controllers/impl/PID_discrete.cpp File Reference

```
#include "PID_discrete.hpp"
#include <iostream>
#include <string>
#include <cstring>
```

Include dependency graph for PID_discrete.cpp:

7.71 lib/UAV_common/src/controllers/impl/PID_discrete.hpp File Reference

```
#include <memory>
#include <limits>
#include "rapidxml/rapidxml.hpp"
#include "../controller.hpp"
#include "z_trans.hpp"
```

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

Classes

- class controllers::PID_Discrete

Namespaces

- controllers

7.72 lib/UAV_common/src/controllers/impl/z_trans.cpp File Reference

```
#include "z_trans.hpp"
#include <sstream>
#include <iterator>
#include <string>
#include <cstring>
```

Include dependency graph for z_trans.cpp:

Functions

- std::vector< double > splitStringToDoubleVector (const std::string &input)

7.72.1 Function Documentation

7.72.1.1 splitStringToDoubleVector()

```
std::vector<double> splitStringToDoubleVector (
          const std::string & input )
```

7.73 lib/UAV_common/src/controllers/impl/z_trans.hpp File Reference

```
#include <memory>
#include <limits>
#include <array>
#include <vector>
#include <algorithm>
#include <numeric>
#include <stdexcept>
#include <cassert>
#include "rapidxml/rapidxml.hpp"
#include "../controller.hpp"
```

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

Classes

- class controllers::ZTransformStatic< N, D >
- class controllers::ZTransform

Namespaces

- controllers

7.74 lib/UAV_common/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)

7.74.1 Function Documentation

7.74.1.1 shouldLog()

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

7.75 lib/UAV_common/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

7.75.1 Macro Definition Documentation

7.75.1.1 LOGGER_MASK

```
#define LOGGER_MASK -1
```

7.76 lib/UAV_common/src/ode/ode.cpp File Reference

```
#include "ode.hpp"
#include "ode_impl.hpp"
```

Include dependency graph for ode.cpp:

7.77 lib/UAV_common/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.

7.78 lib/UAV_common/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.
- class ODE_PC2

    Second order predictor-corrector method Second order Adams-bashforth and Adams-moulton.
- class ODE_PC4

    Fourth order predictor-corrector method Fourth order Adams-bashforth and Adams-moulton.

7.79 lib/UAV_common/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)
- TEST_P (ODETest, TestHarmonicOscillator)
- INSTANTIATE_TEST_SUITE_P (TestDerivedClasses, ODETest, testing::ValuesIn(getMethodsToTest()))
- int main (int argc, char **argv)

7.79.1 Function Documentation

7.79.1.1 getMethodsToTest()

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

7.79.1.2 INSTANTIATE_TEST_SUITE_P()

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

7.79.1.3 main()

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

7.79.1.4 TEST_F() [1/2]

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

7.79.1.5 TEST_F() [2/2]

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

7.79.1.6 TEST_P() [1/4]

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

7.79.1.7 TEST_P() [2/4]

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

7.79.1.8 TEST_P() [3/4]

```
TEST_P (
          ODETest ,
          TestHarmonicOscillator )
```

7.79.1.9 TEST_P() [4/4]

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

7.80 lib/UAV_common/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.

7.80.1 Function Documentation

7.80.1.1 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

7.80.1.2 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

7.81 lib/UAV_common/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.

7.81.1 Function Documentation

7.81.1.1 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

7.81.1.2 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

7.82 lib/UAV_common/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)

7.82.1 Function Documentation

7.82.1.1 parseHinge()

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

7.83 lib/UAV_common/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 "../controllers/controller.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.

7.84 lib/UAV_common/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

7.84.1 Enumeration Type Documentation

7.84.1.1 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

7.85 lib/UAV_common/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:

7.86 lib/UAV_common/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.

7.87 src/communication/control.cpp File Reference

```
#include "control.hpp"
#include <iostream>
```

Include dependency graph for control.cpp:

Functions

- void orderServerJob (zmq::context_t ∗ctx, std::string uav_address, std::function< std::string(std::string)> handleMsg, bool &run)

7.87.1 Function Documentation

7.87.1.1 orderServerJob()

```
void orderServerJob (
            zmq::context_t * ctx,
            std::string uav_address,
            std::function< std::string(std::string)> handleMsg,
            bool & run )
```

7.88 src/communication/control.hpp File Reference

```
#include <zmq.hpp>
#include <Eigen/Dense>
#include <atomic>
#include <thread>
#include <functional>
#include "../controller/controller.hpp"
```

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

Classes

- class Control

  Control command listener & sender.

7.89 src/communication/control_recv.cpp File Reference

```
#include "control.hpp"
#include <iostream>
#include "../defines.hpp"
```

Include dependency graph for control_recv.cpp:

7.90 src/communication/control_send.cpp File Reference

```
#include "control.hpp"
#include <iostream>
```

Include dependency graph for control_send.cpp:

7.91 src/controller/controller_loop.cpp File Reference

```
#include "controller_loop.hpp"
#include "modes/controller_loop_NONE.hpp"
#include "modes/controller_loop_QACRO.hpp"
#include "modes/controller_loop_QANGLE.hpp"
#include "modes/controller_loop_QPOS.hpp"
#include "modes/controller_loop_FMANUAL.hpp"
#include "modes/controller_loop_FACRO.hpp"
#include "modes/controller_loop_FANGLE.hpp"
#include "modes/controller_loop_RMANUAL.hpp"
#include "modes/controller_loop_RAUTOLAUNCH.hpp"
#include "modes/controller_loop_RANGLE.hpp"
#include "modes/controller_loop_RGUIDED.hpp"
```

Include dependency graph for controller_loop.cpp:

7.92 src/controller/controller_loop.hpp File Reference

```
#include <Eigen/Dense>
#include <map>
#include "controller_mode.hpp"
#include "common.hpp"
#include "mixers.hpp"
#include "../communication/control.hpp"
#include "../navigation/NS.hpp"
```

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

Classes

- class ControllerLoop

  This class is interface of controller modes. All modes should keep this strucure and implements all true virtual methods.

7.93 src/controller/controller_mode.hpp File Reference

```
#include <string_view>
#include <iostream>
```

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

Enumerations

- enum ControllerMode {
  NONE = 0 , QPOS = 1 , QANGLE = 2 , QACRO = 3 ,
  FMANUAL = 4 , FACRO = 5 , FANGLE = 6 , RMANUAL = 7 ,
  RAUTOLAUNCH = 8 , RANGLE = 9 , RGUIDED = 10 }

  Controller modes.

Functions

- constexpr const char ∗ ControllerModeToString (ControllerMode mode) throw ()

  Serializes controller mode to string.
- constexpr ControllerMode ControllerModeFromString (const char ∗mode) throw ()

  Parse string to controller mode.

7.93.1 Enumeration Type Documentation

7.93.1.1 ControllerMode

```
enum ControllerMode
```

Controller modes.

Enumerator

NONE	
QPOS	
QANGLE	
QACRO	
FMANUAL	
FACRO	
FANGLE	
RMANUAL	
RAUTOLAUNCH	
RANGLE	
RGUIDED	

7.93.2 Function Documentation

7.93.2.1 ControllerModeFromString()

```
constexpr ControllerMode ControllerModeFromString (
             const char * mode ) throw ( )   [constexpr]
```

Parse string to controller mode.

Parameters

mode	string to parse

Returns

parsing result, NONE if parse failed

7.93.2.2 ControllerModeToString()

```
constexpr const char* ControllerModeToString (
             ControllerMode mode ) throw ( )   [constexpr]
```

Serializes controller mode to string.

Parameters

mode	controller mode

Returns

serialized mode

7.94 src/controller/mixers.cpp File Reference

```
#include "mixers.hpp"
#include <Eigen/Dense>
#include "common.hpp"
```

Include dependency graph for mixers.cpp:

Functions

- Eigen::VectorXd applyMixerRotors (double climb_rate, double roll_rate, double pitch_rate, double yaw_rate)

Calculates rotor demanded speed as result of multiplication mixer matrix and rates. Average speed is proportional to climb rate.

- Eigen::VectorXd applyMixerRotorsHover (double throttle, double roll_rate, double pitch_rate, double yaw_↩ rate)

  Calculates rotor demanded speed as result of multiplication mixer matrix and rates. Average speed is proportional to throttle. It's scaled to achieve hover at centered throttle.
- Eigen::VectorXd applyMixerSurfaces (double throttle, double roll_rate, double pitch_rate, double yaw_rate)

  Calculated demanded surfaces deflection result of multiplication mixer matrix and rates.

7.94.1 Function Documentation

7.94.1.1 applyMixerRotors()

```
Eigen::VectorXd applyMixerRotors (
             double climb_rate,
             double roll_rate,
             double pitch_rate,
             double yaw_rate )
```

Calculates rotor demanded speed as result of multiplication mixer matrix and rates. Average speed is proportional to climb rate.

Parameters

climb_rate	
roll_rate	
pitch_rate	
yaw_rate	

Returns

Rotors demanded speed

7.94.1.2 applyMixerRotorsHover()

```
Eigen::VectorXd applyMixerRotorsHover (
             double throttle,
             double roll_rate,
             double pitch_rate,
             double yaw_rate )
```

Calculates rotor demanded speed as result of multiplication mixer matrix and rates. Average speed is proportional to throttle. It's scaled to achieve hover at centered throttle.

Parameters

throttle	
roll_rate	
pitch_rate	
yaw_rate	

Returns

Rotors demanded speed

7.94.1.3 applyMixerSurfaces()

```
Eigen::VectorXd applyMixerSurfaces (
            double throttle,
            double roll_rate,
            double pitch_rate,
            double yaw_rate )
```

Calculated demanded surfaces deflection result of multiplication mixer matrix and rates.

Parameters

throttle	
roll_rate	
pitch_rate	
yaw_rate	

Returns

demanded surfaces deflection

7.95 src/controller/mixers.hpp File Reference

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

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

Functions

- Eigen::VectorXd applyMixerRotors (double climb_rate, double roll_rate, double pitch_rate, double yaw_rate)

  Calculates rotor demanded speed as result of multiplication mixer matrix and rates. Average speed is proportional to climb rate.
- Eigen::VectorXd applyMixerRotorsHover (double throttle, double roll_rate, double pitch_rate, double yaw_↩ rate)

  Calculates rotor demanded speed as result of multiplication mixer matrix and rates. Average speed is proportional to throttle. It's scaled to achieve hover at centered throttle.
- Eigen::VectorXd applyMixerSurfaces (double throttle, double roll_rate, double pitch_rate, double yaw_rate)

  Calculated demanded surfaces deflection result of multiplication mixer matrix and rates.

7.95.1 Function Documentation

7.95.1.1 applyMixerRotors()

```
Eigen::VectorXd applyMixerRotors (
            double climb_rate,
            double roll_rate,
            double pitch_rate,
            double yaw_rate )
```

Calculates rotor demanded speed as result of multiplication mixer matrix and rates. Average speed is proportional to climb rate.

Parameters

climb_rate	
roll_rate	
pitch_rate	
yaw_rate	

Returns

Rotors demanded speed

7.95.1.2 applyMixerRotorsHover()

```
Eigen::VectorXd applyMixerRotorsHover (
            double throttle,
            double roll_rate,
            double pitch_rate,
            double yaw_rate )
```

Calculates rotor demanded speed as result of multiplication mixer matrix and rates. Average speed is proportional to throttle. It's scaled to achieve hover at centered throttle.

Parameters

throttle	
roll_rate	
pitch_rate	
yaw_rate	

Returns

Rotors demanded speed

7.95.1.3 applyMixerSurfaces()

```
Eigen::VectorXd applyMixerSurfaces (
            double throttle,
            double roll_rate,
            double pitch_rate,
            double yaw_rate )
```

Calculated demanded surfaces deflection result of multiplication mixer matrix and rates.

Parameters

throttle	
roll_rate	
pitch_rate	
yaw_rate	

Returns

demanded surfaces deflection

7.96 src/controller/modes/controller_loop_FACRO.cpp File Reference

```
#include "controller_loop_FACRO.hpp"
#include "../../utils.hpp"
```

Include dependency graph for controller_loop_FACRO.cpp:

7.97 src/controller/modes/controller_loop_FACRO.hpp File Reference

```
#include "../controller_loop.hpp"
```

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

Classes

- class ControllerLoopFACRO

7.98 src/controller/modes/controller_loop_FANGLE.cpp File Reference

```
#include "controller_loop_FANGLE.hpp"
#include "../../utils.hpp"
```

Include dependency graph for controller_loop_FANGLE.cpp:

7.99 src/controller/modes/controller_loop_FANGLE.hpp File Reference

```
#include "../controller_loop.hpp"
```

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

Classes

- class ControllerLoopFANGLE

7.100 src/controller/modes/controller_loop_FMANUAL.cpp File Reference

```
#include "controller_loop_FMANUAL.hpp"
```

Include dependency graph for controller_loop_FMANUAL.cpp:

7.101 src/controller/modes/controller_loop_FMANUAL.hpp File Reference

```
#include "../controller_loop.hpp"
```

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

Classes

- class ControllerLoopFMANUAL

7.102 src/controller/modes/controller_loop_NONE.cpp File Reference

```
#include "controller_loop_NONE.hpp"
```

Include dependency graph for controller_loop_NONE.cpp:

7.103 src/controller/modes/controller_loop_NONE.hpp File Reference

```
#include "../controller_loop.hpp"
```

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

Classes

- class ControllerLoopNONE

7.104 src/controller/modes/controller_loop_QACRO.cpp File Reference

```
#include "controller_loop_QACRO.hpp"
```

Include dependency graph for controller_loop_QACRO.cpp:

7.105 src/controller/modes/controller_loop_QACRO.hpp File Reference

```
#include "../controller_loop.hpp"
```

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

Classes

- class ControllerLoopQACRO

7.106 src/controller/modes/controller_loop_QANGLE.cpp File Reference

```
#include "controller_loop_QANGLE.hpp"
#include "../../utils.hpp"
```

Include dependency graph for controller_loop_QANGLE.cpp:

7.107 src/controller/modes/controller_loop_QANGLE.hpp File Reference

```
#include "../controller_loop.hpp"
```

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

Classes

- class ControllerLoopQANGLE

7.108 src/controller/modes/controller_loop_QPOS.cpp File Reference

```
#include "controller_loop_QPOS.hpp"
#include "../../utils.hpp"
```

Include dependency graph for controller_loop_QPOS.cpp:

7.109 src/controller/modes/controller_loop_QPOS.hpp File Reference

```
#include "../controller_loop.hpp"
```

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

Classes

- class ControllerLoopQPOS

7.110 src/controller/modes/controller_loop_RANGLE.cpp File Reference

```
#include "controller_loop_RANGLE.hpp"
#include "../../utils.hpp"
```

Include dependency graph for controller_loop_RANGLE.cpp:

7.111 src/controller/modes/controller_loop_RANGLE.hpp File Reference

```
#include "../controller_loop.hpp"
```

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

Classes

- class ControllerLoopRANGLE

7.112 src/controller/modes/controller_loop_RAUTOLAUNCH.cpp File Reference

```
#include "controller_loop_RAUTOLAUNCH.hpp"
```

Include dependency graph for controller_loop_RAUTOLAUNCH.cpp:

7.113 src/controller/modes/controller_loop_RAUTOLAUNCH.hpp File Reference

```
#include "../controller_loop.hpp"
```

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

Classes

- class ControllerLoopRAUTOLAUNCH

7.114 src/controller/modes/controller_loop_RGUIDED.cpp File Reference

```
#include "controller_loop_RGUIDED.hpp"
#include "../../utils.hpp"
#include "common.hpp"
```

Include dependency graph for controller_loop_RGUIDED.cpp:

7.115 src/controller/modes/controller_loop_RGUIDED.hpp File Reference

```
#include "../controller_loop.hpp"
```

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

Classes

- class ControllerLoopRGUIDED

7.116 src/controller/modes/controller_loop_RMANUAL.cpp File Reference

```
#include "controller_loop_RMANUAL.hpp"
```

Include dependency graph for controller_loop_RMANUAL.cpp:

7.117 src/controller/modes/controller_loop_RMANUAL.hpp File Reference

```
#include "../controller_loop.hpp"
```

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

Classes

- class ControllerLoopRMANUAL

7.118 src/defines.hpp File Reference

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

Namespaces

- def

  Controller constants.

Macros

- #define USE_QUATERIONS 1

Variables

- const int def::INFO_PERIOD = 2

  How often send demands in response to stick command.

7.118.1 Macro Definition Documentation

7.118.1.1 USE_QUATERIONS

```
#define USE_QUATERIONS 1
```

7.119 src/main.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <cxxopts.hpp>
#include <thread>
#include <chrono>
#include <filesystem>
#include "zmq.hpp"
#include "controller/controller.hpp"
#include "common.hpp"
#include "params.hpp"
```

Include dependency graph for main.cpp:

Macros

- #define LOGGER_MASK 5

Functions

- void parseArgs (int argc, char **argv, UAVparams *params, Params &p)

  Parse CL arguments.
- int main (int argc, char **argv)

Variables

- std::string log_path = "logs/"

7.119.1 Macro Definition Documentation

7.119.1.1 LOGGER_MASK

```
#define LOGGER_MASK 5
```

7.119.2 Function Documentation

7.119.2.1 main()

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

7.119.2.2 parseArgs()

```
void parseArgs (
            int argc,
            char ** argv,
            UAVparams * params,
            Params & p )
```

Parse CL arguments.

Parameters

argc	number of argument
argv	argument array
params	pointer to UAVparams instant that should be filled
p	internal params reference

7.119.3 Variable Documentation

7.119.3.1 log_path

```
std::string log_path = "logs/"
```

7.120 src/navigation/AHRS.cpp File Reference

```
#include "AHRS.hpp"
#include <Eigen/Dense>
#include <random>
#include "common.hpp"
```

Include dependency graph for AHRS.cpp:

7.121 src/navigation/AHRS.hpp File Reference

```
#include <Eigen/Dense>
#include <random>
#include <optional>
#include "environment.hpp"
#include "sensors.hpp"
```

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

Classes

- class AHRS

  Attitude and heading reference system.

7.122 src/navigation/AHRS/AHRS_complementary.cpp File Reference

```
#include "AHRS_complementary.hpp"
#include <Eigen/Dense>
#include <random>
#include <iostream>
#include "common.hpp"
```

Include dependency graph for AHRS_complementary.cpp:

Functions

- Eigen::Matrix3d calcRnb (Eigen::Vector3d ori)
- Eigen::Matrix3d calcRbn (Eigen::Vector3d ori)
- Eigen::Matrix3d calcTom (Eigen::Vector3d ori)
- void clampOrientation (Eigen::Vector3d &vec)

7.122.1 Function Documentation

7.122.1.1 calcRbn()

```
Eigen::Matrix3d calcRbn (
            Eigen::Vector3d ori )
```

7.122.1.2 calcRnb()

```
Eigen::Matrix3d calcRnb (
            Eigen::Vector3d ori )
```

7.122.1.3 calcTom()

```
Eigen::Matrix3d calcTom (
            Eigen::Vector3d ori )
```

7.122.1.4 clampOrientation()

```
void clampOrientation (
            Eigen::Vector3d & vec )
```

7.123 src/navigation/AHRS/AHRS_complementary.hpp File Reference

```
#include <Eigen/Dense>
#include <random>
#include "../environment.hpp"
#include "../sensors.hpp"
#include "common.hpp"
#include "../AHRS.hpp"
```

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

Classes

- class AHRS_complementary

  Implementation of AHRS based on Complementary Filter.

7.124 src/navigation/AHRS/AHRS_EKF.cpp File Reference

```
#include "AHRS_EKF.hpp"
#include <Eigen/Dense>
#include <random>
#include <iostream>
```

Include dependency graph for AHRS_EKF.cpp:

Functions

- Eigen::Matrix< double, 4, 3 > S (Eigen::Vector4d q)
- Eigen::Matrix< double, 6, 7 > C (Eigen::Vector4d q)

7.124.1 Function Documentation

7.124.1.1 C()

```
Eigen::Matrix<double,6,7> C (
            Eigen::Vector4d q )
```

7.124.1.2 S()

```
Eigen::Matrix<double,4,3> S (
            Eigen::Vector4d q )
```

7.125 src/navigation/AHRS/AHRS_EKF.hpp File Reference

```
#include <Eigen/Dense>
#include "../environment.hpp"
#include "../sensors.hpp"
#include "common.hpp"
#include "../AHRS.hpp"
```

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

Classes

- class AHRS_EKF

  Implementation of AHRS based on Extended Kalman Filter.

7.126 src/navigation/EKF.cpp File Reference

```
#include "EKF.hpp"
#include <Eigen/Dense>
#include <iostream>
#include "common.hpp"
```

Include dependency graph for EKF.cpp:

7.127 src/navigation/EKF.hpp File Reference

```
#include <Eigen/Dense>
#include "environment.hpp"
#include "sensors.hpp"
```

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

Classes

- struct EKFParams

  EK filer parameters.
- class EKF

  Extended Kalman Filter.

7.128 src/navigation/environment.cpp File Reference

```
#include "environment.hpp"
#include <zmq.hpp>
#include <thread>
#include <Eigen/Dense>
#include <mutex>
#include <vector>
#include <memory>
#include <iostream>
#include <initializer_list>
#include "../utils.hpp"
#include "common.hpp"
#include "sensors.hpp"
#include "../defines.hpp"
```

Include dependency graph for environment.cpp:

Functions

- void connectConflateSocket (zmq::socket_t &sock, std::string address, std::string topic)
- template<int Size1, int Size2>
  bool recvVectors (zmq::socket_t &sock, int skip, Eigen::Vector< double, Size1 > &vec1, Eigen::Vector< double, Size2 > &vec2)
- Eigen::Matrix< double, 3, 3 > r_nb (const Eigen::Vector3d &RPY)
- Eigen::Matrix< double, 3, 3 > r_nb (const Eigen::Vector4d &e)

7.128.1 Function Documentation

7.128.1.1 connectConflateSocket()

```
void connectConflateSocket (
            zmq::socket_t & sock,
            std::string address,
            std::string topic )
```

7.128.1.2 r_nb() [1/2]

```
Eigen::Matrix<double, 3, 3> r_nb (
            const Eigen::Vector3d & RPY )
```

7.128.1.3 r_nb() [2/2]

```
Eigen::Matrix<double, 3, 3> r_nb (
            const Eigen::Vector4d & e )
```

7.128.1.4 recvVectors()

```
template<int Size1, int Size2>
bool recvVectors (
            zmq::socket_t & sock,
            int skip,
            Eigen::Vector< double, Size1 > & vec1,
            Eigen::Vector< double, Size2 > & vec2 )
```

7.129 src/navigation/environment.hpp File Reference

```
#include <zmq.hpp>
#include <thread>
#include <Eigen/Dense>
#include <mutex>
#include <vector>
#include <memory>
#include <atomic>
#include <map>
#include "sensors.hpp"
#include "common.hpp"
#include "../defines.hpp"
```

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

Classes

- class Environment

7.130 src/navigation/NS.cpp File Reference

```
#include "NS.hpp"
#include <Eigen/Dense>
#include <iostream>
#include "AHRS/AHRS_EKF.hpp"
#include "AHRS/AHRS_complementary.hpp"
#include "../defines.hpp"
#include "../params.hpp"
```

Include dependency graph for NS.cpp:

7.131 src/navigation/NS.hpp File Reference

```
#include <Eigen/Dense>
#include "environment.hpp"
#include "sensors.hpp"
#include "AHRS.hpp"
#include "EKF.hpp"
```

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

Classes

- class NS

  Navigation system.

7.132 src/navigation/sensors.cpp File Reference

```
#include "sensors.hpp"
#include <Eigen/Dense>
#include <random>
#include <limits>
#include "environment.hpp"
#include "common.hpp"
```

Include dependency graph for sensors.cpp:

7.133 src/navigation/sensors.hpp File Reference

```
#include <Eigen/Dense>
#include <random>
#include <atomic>
#include "common.hpp"
```

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

Classes

- class Sensor< T >

  Sensors base class.
- class Accelerometer

  Representation of accelerometer.
- class Gyroscope

  Representation of gyroscope.
- class Magnetometer

  Representation of magnetometer.
- class Barometer

  Representation of barometer.
- class GPS

  Representation of GPS position measure.
- class GPSVel

  Representation of GPS velocity measure.

7.134 src/params.cpp File Reference

```
#include "params.hpp"
#include <iostream>
```

Include dependency graph for params.cpp:

7.135 src/params.hpp File Reference

```
#include <string>
```

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

Classes

- class Params

  Simulation parameters.

7.136 src/utils.hpp File Reference

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

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

Functions

- template<typename T >
  void safeSet (T &vec, T &new_val, std::mutex &mtx)

  Safe setter for T type value protected by mutex.
- template<typename T >
  T safeGet (T &vec, std::mutex &mtx)

  Safe getter for T type value protected by mutex.
- double circularError (double demanded, double val)

  Calculates error between demanded and actual angle. Finds shorter path. For example if actual value is -0.9pi and demanded is 0.9pi error is equal -0.2pi.
- double clampAngle (double angle)

  Clamps angle given in radians to range <-pi,pi>

7.136.1 Function Documentation

7.136.1.1 circularError()

```
double circularError (
            double demanded,
            double val )  [inline]
```

Calculates error between demanded and actual angle. Finds shorter path. For example if actual value is -0.9pi and demanded is 0.9pi error is equal -0.2pi.

Parameters

demanded	demanded angle in radian
val	actual angle in radian

Returns

angle error

7.136.1.2 clampAngle()

```
double clampAngle (
             double angle )  [inline]
```

Clamps angle given in radians to range <-pi,pi>

Parameters

angle	angle in radian

Returns

angle converted to range <-pi,pi>

7.136.1.3 safeGet()

```
template<typename T >
T safeGet (
             T & vec,
             std::mutex & mtx )  [inline]
```

Safe getter for T type value protected by mutex.

Template Parameters

T	Type of variable

Parameters

vec	value to be get
mtx	mutex

Returns

value of vec

7.136.1.4 safeSet()

```
template<typename T >
void safeSet (
            T & vec,
            T & new_val,
            std::mutex & mtx )  [inline]
```

Safe setter for T type value protected by mutex.

Template Parameters

T	Type of variable

Parameters

vec	value to be set
new_val	new value
mtx	mutex

Index