

UAV drop physic

Generated by Doxygen 1.9.1

1 Namespace Index	1
1.1 Namespace List	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Namespace Documentation	9
5.1 def Namespace Reference	9
5.1.1 Detailed Description	9
5.1.2 Variable Documentation	9
5.1.2.1 INFO_PERIOD	9
5.1.2.2 STEP_TIME	9
6 Class Documentation	11
6.1 Accelerometer Class Reference	11
6.1.1 Detailed Description	11
6.1.2 Constructor & Destructor Documentation	11
6.1.2.1 Accelerometer()	12
6.1.3 Member Function Documentation	12
6.1.3.1 update()	12
6.1.4 Member Data Documentation	12
6.1.4.1 g	12
6.2 AeroCoefficients Struct Reference	12
6.2.1 Detailed Description	13
6.2.2 Member Data Documentation	13
6.2.2.1 C0	13
6.2.2.2 Cab	13
6.2.2.3 Cpqr	13
6.2.2.4 d	13
6.2.2.5 eAR	13
6.2.2.6 S	13
6.2.2.7 stallLimit	14
6.3 AHRS Class Reference	14
6.3.1 Detailed Description	14
6.3.2 Constructor & Destructor Documentation	14
6.3.2.1 AHRS()	14
6.3.2.2 ~AHRS()	15
6.3.3 Member Function Documentation	15

6.3.3.1 getGyroBias()	15
6.3.3.2 getOri()	15
6.3.3.3 rot_bw()	16
6.3.3.4 update()	16
6.3.4 Member Data Documentation	16
6.3.4.1 env	16
6.3.4.2 logger	16
6.3.4.3 mtxOri	16
6.3.4.4 ori_est	17
6.4 AHRS_complementary Class Reference	17
6.4.1 Detailed Description	17
6.4.2 Constructor & Destructor Documentation	17
6.4.2.1 AHRS_complementary()	17
6.4.2.2 ~AHRS_complementary()	18
6.4.3 Member Function Documentation	18
6.4.3.1 rot_bw()	18
6.4.3.2 update()	18
6.4.4 Member Data Documentation	18
6.4.4.1 alpha	18
6.5 AHRS_EKF Class Reference	19
6.5.1 Detailed Description	19
6.5.2 Constructor & Destructor Documentation	19
6.5.2.1 AHRS_EKF()	19
6.5.2.2 ~AHRS_EKF()	20
6.5.3 Member Function Documentation	20
6.5.3.1 getGyroBias()	20
6.5.3.2 q()	20
6.5.3.3 quaterionToRPY()	20
6.5.3.4 rot_bw()	20
6.5.3.5 RPYToQuaterion()	21
6.5.3.6 update()	21
6.5.4 Member Data Documentation	21
6.5.4.1 P	21
6.5.4.2 Q	21
6.5.4.3 R	21
6.5.4.4 x	21
6.6 AHRSParams Struct Reference	22
6.6.1 Detailed Description	22
6.6.2 Member Data Documentation	22
6.6.2.1 alpha	22
6.6.2.2 Q	22
6.6.2.3 R	22

6.6.2.4 type	22
6.7 Ammo Class Reference	23
6.7.1 Constructor & Destructor Documentation	23
6.7.1.1 Ammo() [1/2]	23
6.7.1.2 Ammo() [2/2]	23
6.7.2 Member Function Documentation	23
6.7.2.1 getV0()	24
6.7.2.2 operator=()	24
6.7.3 Member Data Documentation	24
6.7.3.1 _V0	24
6.8 Barometer Class Reference	24
6.8.1 Detailed Description	25
6.8.2 Constructor & Destructor Documentation	25
6.8.2.1 Barometer()	25
6.8.3 Member Function Documentation	25
6.8.3.1 update()	25
6.9 Cargo Class Reference	25
6.9.1 Constructor & Destructor Documentation	26
6.9.1.1 Cargo() [1/2]	26
6.9.1.2 Cargo() [2/2]	26
6.10 Control Class Reference	26
6.10.1 Detailed Description	27
6.10.2 Constructor & Destructor Documentation	27
6.10.2.1 Control()	27
6.10.2.2 ~Control()	28
6.10.3 Member Function Documentation	28
6.10.3.1 handleMsg()	28
6.10.3.2 prepare()	28
6.10.3.3 recv()	28
6.10.3.4 sendHinge()	29
6.10.3.5 sendSpeed()	30
6.10.3.6 sendSurface()	30
6.10.3.7 start()	30
6.10.3.8 startJet()	30
6.10.3.9 stop()	31
6.11 Controller Class Reference	31
6.11.1 Detailed Description	31
6.11.2 Constructor & Destructor Documentation	32
6.11.2.1 Controller()	32
6.11.2.2 ~Controller()	33
6.11.3 Member Function Documentation	33
6.11.3.1 exitController()	33

6.11.3.2 run()	33
6.11.3.3 setMode()	33
6.11.4 Friends And Related Function Documentation	34
6.11.4.1 Control	34
6.12 ControllerLoop Class Reference	34
6.12.1 Detailed Description	35
6.12.2 Constructor & Destructor Documentation	35
6.12.2.1 ControllerLoop()	35
6.12.2.2 ~ControllerLoop()	35
6.12.3 Member Function Documentation	35
6.12.3.1 checkJoystickLength()	36
6.12.3.2 ControllerLoopFactory()	37
6.12.3.3 demandInfo()	37
6.12.3.4 getMode()	37
6.12.3.5 handleJoystick()	38
6.12.3.6 job()	38
6.12.3.7 overridePosition()	38
6.12.3.8 requiredPIDs()	38
6.12.4 Member Data Documentation	39
6.12.4.1 _mode	39
6.12.4.2 required_pids	39
6.13 ControllerLoopFMANUAL Class Reference	39
6.13.1 Constructor & Destructor Documentation	39
6.13.1.1 ControllerLoopFMANUAL()	40
6.13.2 Member Function Documentation	40
6.13.2.1 handleJoystick()	40
6.13.2.2 job()	40
6.14 ControllerLoopNONE Class Reference	40
6.14.1 Constructor & Destructor Documentation	40
6.14.1.1 ControllerLoopNONE()	41
6.15 ControllerLoopQACRO Class Reference	41
6.15.1 Constructor & Destructor Documentation	41
6.15.1.1 ControllerLoopQACRO()	41
6.15.2 Member Function Documentation	41
6.15.2.1 demandInfo()	42
6.15.2.2 handleJoystick()	42
6.15.2.3 job()	42
6.16 ControllerLoopQANGLE Class Reference	42
6.16.1 Constructor & Destructor Documentation	43
6.16.1.1 ControllerLoopQANGLE()	43
6.16.2 Member Function Documentation	43
6.16.2.1 demandInfo()	43

6.16.2.2 handleJoystick()	43
6.16.2.3 job()	43
6.16.2.4 overridePosition()	44
6.17 ControllerLoopQPOS Class Reference	44
6.17.1 Constructor & Destructor Documentation	44
6.17.1.1 ControllerLoopQPOS()	44
6.17.2 Member Function Documentation	44
6.17.2.1 demandInfo()	45
6.17.2.2 handleJoystick()	45
6.17.2.3 job()	45
6.17.2.4 overridePosition()	45
6.18 ControllerLoopRMANUAL Class Reference	45
6.18.1 Constructor & Destructor Documentation	46
6.18.1.1 ControllerLoopRMANUAL()	46
6.18.2 Member Function Documentation	46
6.18.2.1 demandInfo()	46
6.18.2.2 handleJoystick()	46
6.18.2.3 job()	47
6.18.3 Member Data Documentation	47
6.18.3.1 demandedX	47
6.18.3.2 demandedY	47
6.19 ControlSurfaces Class Reference	47
6.19.1 Detailed Description	48
6.19.2 Constructor & Destructor Documentation	48
6.19.2.1 ControlSurfaces() [1/2]	48
6.19.2.2 ControlSurfaces() [2/2]	48
6.19.3 Member Function Documentation	48
6.19.3.1 getCoefficients()	48
6.19.3.2 getNoOfSurface()	49
6.19.3.3 getValues()	49
6.19.3.4 restoreTrim()	49
6.19.3.5 setValues()	49
6.20 Drive Struct Reference	49
6.20.1 Detailed Description	50
6.20.2 Member Data Documentation	50
6.20.2.1 axis	50
6.20.2.2 hinges	50
6.20.2.3 noOfHinges	50
6.20.2.4 position	50
6.21 EKF Class Reference	50
6.21.1 Detailed Description	51
6.21.2 Constructor & Destructor Documentation	51

6.21.2.1 EKF()	51
6.21.3 Member Function Documentation	51
6.21.3.1 getPos()	51
6.21.3.2 getVel()	52
6.21.3.3 log()	52
6.21.3.4 predict()	52
6.21.3.5 updateBaro()	52
6.21.3.6 updateGPS()	53
6.21.3.7 updateGPSVel()	53
6.22 EKFPARAMS Struct Reference	53
6.22.1 Detailed Description	54
6.22.2 Member Data Documentation	54
6.22.2.1 P0	54
6.22.2.2 Q	54
6.22.2.3 RBaro	54
6.22.2.4 RGPSPos	54
6.22.2.5 RGPSPVel	54
6.23 EKFSCLERS Struct Reference	55
6.23.1 Detailed Description	55
6.23.2 Member Data Documentation	55
6.23.2.1 baroScaler	55
6.23.2.2 predictScaler	55
6.23.2.3 updateScaler	55
6.23.2.4 zScaler	55
6.24 Environment Class Reference	56
6.24.1 Constructor & Destructor Documentation	56
6.24.1.1 Environment()	56
6.24.1.2 ~Environment()	57
6.24.2 Member Function Documentation	57
6.24.2.1 getAngularAcceleraton()	57
6.24.2.2 getAngularVelocity()	57
6.24.2.3 getLinearAcceleration()	58
6.24.2.4 getLinearVelocity()	58
6.24.2.5 getOrientation()	58
6.24.2.6 getPosition()	58
6.24.2.7 getRnb()	59
6.24.2.8 getTime()	59
6.24.2.9 getWorldAngularVelocity()	59
6.24.2.10 getWorldLinearVelocity()	59
6.24.2.11 updateSensors()	60
6.24.3 Member Data Documentation	60
6.24.3.1 sensors	60

6.24.3.2 sensorsVec3d	60
6.25 GPS Class Reference	60
6.25.1 Detailed Description	61
6.25.2 Constructor & Destructor Documentation	61
6.25.2.1 GPS()	61
6.25.3 Member Function Documentation	61
6.25.3.1 update()	61
6.26 GPSVel Class Reference	61
6.26.1 Detailed Description	62
6.26.2 Constructor & Destructor Documentation	62
6.26.2.1 GPSVel()	62
6.26.3 Member Function Documentation	62
6.26.3.1 update()	62
6.27 Gyroscope Class Reference	63
6.27.1 Detailed Description	63
6.27.2 Constructor & Destructor Documentation	63
6.27.2.1 Gyroscope()	63
6.27.3 Member Function Documentation	63
6.27.3.1 update()	63
6.28 Hinge Class Reference	64
6.28.1 Detailed Description	64
6.28.2 Constructor & Destructor Documentation	64
6.28.2.1 Hinge() [1/3]	64
6.28.2.2 Hinge() [2/3]	64
6.28.2.3 Hinge() [3/3]	64
6.28.3 Member Function Documentation	65
6.28.3.1 getRot()	65
6.28.3.2 operator=()	65
6.28.3.3 updateValue()	65
6.29 Jet Class Reference	65
6.29.1 Detailed Description	66
6.29.2 Member Function Documentation	66
6.29.2.1 getLastThrust()	66
6.29.2.2 getThrust()	66
6.29.2.3 start()	67
6.29.3 Member Data Documentation	67
6.29.3.1 phases	67
6.29.3.2 thrust	67
6.29.3.3 time	67
6.30 Load Class Reference	68
6.30.1 Detailed Description	68
6.30.2 Constructor & Destructor Documentation	68

6.30.2.1 Load() [1/2]	68
6.30.2.2 Load() [2/2]	68
6.30.3 Member Function Documentation	69
6.30.3.1 getMass()	69
6.30.3.2 getOffset()	69
6.30.3.3 operator=()	69
6.30.3.4 release()	69
6.31 Logger Class Reference	70
6.31.1 Detailed Description	70
6.31.2 Constructor & Destructor Documentation	70
6.31.2.1 Logger()	70
6.31.2.2 ~Logger()	71
6.31.3 Member Function Documentation	71
6.31.3.1 log() [1/2]	71
6.31.3.2 log() [2/2]	71
6.31.3.3 setFmt()	72
6.31.3.4 setLogDirectory()	72
6.32 Magnetometer Class Reference	72
6.32.1 Detailed Description	73
6.32.2 Constructor & Destructor Documentation	73
6.32.2.1 Magnetometer()	73
6.32.3 Member Function Documentation	73
6.32.3.1 update()	73
6.32.4 Member Data Documentation	73
6.32.4.1 mag	73
6.33 NS Class Reference	74
6.33.1 Detailed Description	74
6.33.2 Constructor & Destructor Documentation	74
6.33.2.1 NS()	74
6.33.2.2 ~NS()	74
6.33.3 Member Function Documentation	75
6.33.3.1 getAngularVelocity()	75
6.33.3.2 getLinearVelocity()	75
6.33.3.3 getOrientation()	75
6.33.3.4 getPosition()	76
6.34 ODE Class Reference	76
6.34.1 Detailed Description	77
6.34.2 Member Enumeration Documentation	77
6.34.2.1 ODEMethod	77
6.34.3 Constructor & Destructor Documentation	77
6.34.3.1 ODE()	77
6.34.3.2 ~ODE()	77

6.34.4 Member Function Documentation	77
6.34.4.1 factory()	77
6.34.4.2 fromString()	78
6.34.4.3 getMicrosteps() [1/2]	78
6.34.4.4 getMicrosteps() [2/2]	78
6.34.4.5 step()	79
6.35 ODE_Euler Class Reference	79
6.35.1 Detailed Description	80
6.35.2 Constructor & Destructor Documentation	80
6.35.2.1 ODE_Euler()	80
6.35.3 Member Function Documentation	80
6.35.3.1 step()	80
6.36 ODE_Heun Class Reference	81
6.36.1 Detailed Description	81
6.36.2 Constructor & Destructor Documentation	81
6.36.2.1 ODE_Heun()	81
6.36.3 Member Function Documentation	81
6.36.3.1 step()	81
6.37 ODE_RK4 Class Reference	82
6.37.1 Detailed Description	82
6.37.2 Constructor & Destructor Documentation	82
6.37.2.1 ODE_RK4()	83
6.37.3 Member Function Documentation	83
6.37.3.1 step()	83
6.38 ODETest Class Reference	83
6.38.1 Member Function Documentation	84
6.38.1.1 SetUp()	84
6.38.1.2 TearDown()	84
6.39 PID Class Reference	84
6.39.1 Detailed Description	84
6.39.2 Constructor & Destructor Documentation	84
6.39.2.1 PID()	85
6.39.2.2 ~PID()	85
6.39.3 Member Function Documentation	85
6.39.3.1 calc() [1/2]	85
6.39.3.2 calc() [2/2]	85
6.39.3.3 clear()	86
6.39.3.4 set_dt()	86
6.40 Rotor Struct Reference	86
6.40.1 Detailed Description	87
6.40.2 Member Data Documentation	87
6.40.2.1 direction	87

6.40.2.2 forceCoff	87
6.40.2.3 hoverSpeed	87
6.40.2.4 maxSpeed	87
6.40.2.5 timeConstant	87
6.40.2.6 torqueCoff	87
6.41 Sensor< T > Class Template Reference	88
6.41.1 Detailed Description	88
6.41.2 Constructor & Destructor Documentation	89
6.41.2.1 Sensor()	89
6.41.3 Member Function Documentation	89
6.41.3.1 error()	89
6.41.3.2 getReading()	89
6.41.3.3 getSd()	90
6.41.3.4 isReady()	90
6.41.3.5 shouldUpdate()	90
6.41.3.6 update()	90
6.41.4 Member Data Documentation	91
6.41.4.1 bias	91
6.41.4.2 dist	91
6.41.4.3 env	91
6.41.4.4 gen	91
6.41.4.5 lastUpdate	91
6.41.4.6 logger	91
6.41.4.7 ready	92
6.41.4.8 refreshTime	92
6.41.4.9 value	92
6.42 SensorParams Struct Reference	92
6.42.1 Detailed Description	92
6.42.2 Member Data Documentation	92
6.42.2.1 bias	93
6.42.2.2 name	93
6.42.2.3 refreshTime	93
6.42.2.4 sd	93
6.43 TimedLoop Class Reference	93
6.43.1 Detailed Description	93
6.43.2 Constructor & Destructor Documentation	94
6.43.2.1 TimedLoop()	94
6.43.3 Member Function Documentation	95
6.43.3.1 go() [1/2]	95
6.43.3.2 go() [2/2]	95
6.44 UAVparams Struct Reference	95
6.44.1 Detailed Description	96

6.44.2 Constructor & Destructor Documentation	97
6.44.2.1 UAVparams()	97
6.44.2.2 ~UAVparams()	97
6.44.3 Member Function Documentation	97
6.44.3.1 getRotorHoverSpeeds()	97
6.44.3.2 getRotorMaxSpeeds()	97
6.44.3.3 getRotorTimeContants()	97
6.44.3.4 getSingleton()	97
6.44.3.5 loadConfig()	98
6.44.4 Member Data Documentation	98
6.44.4.1 aero_coffs	98
6.44.4.2 ahrs	98
6.44.4.3 ammo	98
6.44.4.4 cargo	98
6.44.4.5 ekf	98
6.44.4.6 initialMode	98
6.44.4.7 initialOrientation	99
6.44.4.8 initialPosition	99
6.44.4.9 initialVelocity	99
6.44.4.10 instantRun	99
6.44.4.11 lx	99
6.44.4.12 lxy	99
6.44.4.13 lxz	99
6.44.4.14 ly	99
6.44.4.15 lyz	100
6.44.4.16 lz	100
6.44.4.17 jets	100
6.44.4.18 m	100
6.44.4.19 name	100
6.44.4.20 noOfAmmo	100
6.44.4.21 noOfCargo	100
6.44.4.22 noOfJets	100
6.44.4.23 noOfRotors	101
6.44.4.24 pids	101
6.44.4.25 rotorMixer	101
6.44.4.26 rotors	101
6.44.4.27 sensors	101
6.44.4.28 surfaceMixer	101
6.44.4.29 surfaces	101
7 File Documentation	103
7.1 lib/UAV_common/header/common.hpp File Reference	103

7.2 lib/UAV_common/src/components/aero_coefficients.hpp File Reference	103
7.3 lib/UAV_common/src/components/components.hpp File Reference	103
7.4 lib/UAV_common/src/components/control_surfaces.cpp File Reference	104
7.5 lib/UAV_common/src/components/control_surfaces.hpp File Reference	104
7.6 lib/UAV_common/src/components/drive.cpp File Reference	104
7.7 lib/UAV_common/src/components/drive.hpp File Reference	104
7.8 lib/UAV_common/src/components/hinge.cpp File Reference	104
7.8.1 Function Documentation	105
7.8.1.1 asSkewMatrix()	105
7.9 lib/UAV_common/src/components/hinge.hpp File Reference	105
7.10 lib/UAV_common/src/components/loads.cpp File Reference	105
7.11 lib/UAV_common/src/components/loads.hpp File Reference	105
7.12 lib/UAV_common/src/components/navi.hpp File Reference	106
7.13 lib/UAV_common/src/logger/logger.cpp File Reference	106
7.13.1 Function Documentation	106
7.13.1.1 shouldLog()	106
7.14 lib/UAV_common/src/logger/logger.hpp File Reference	106
7.14.1 Macro Definition Documentation	107
7.14.1.1 LOGGER_MASK	107
7.15 lib/UAV_common/src/ode/ode.cpp File Reference	107
7.16 lib/UAV_common/src/ode/ode.hpp File Reference	107
7.17 lib/UAV_common/src/ode/ode_impl.hpp File Reference	107
7.18 lib/UAV_common/src/ode/ode_test.cpp File Reference	108
7.18.1 Function Documentation	108
7.18.1.1 getMethodsToTest()	108
7.18.1.2 INSTITUTE_TEST_SUITE_P()	108
7.18.1.3 main()	109
7.18.1.4 TEST_F() [1/2]	109
7.18.1.5 TEST_F() [2/2]	109
7.18.1.6 TEST_P() [1/3]	109
7.18.1.7 TEST_P() [2/3]	109
7.18.1.8 TEST_P() [3/3]	109
7.19 lib/UAV_common/src/parser/parser.cpp File Reference	110
7.19.1 Function Documentation	110
7.19.1.1 parseMatrixXd()	110
7.19.1.2 parseVectorXd()	110
7.20 lib/UAV_common/src/parser/parser.hpp File Reference	111
7.20.1 Function Documentation	111
7.20.1.1 parseMatrixXd()	111
7.20.1.2 parseVectorXd()	112
7.21 lib/UAV_common/src/parser/uav_params.cpp File Reference	112
7.21.1 Function Documentation	112

7.21.1.1 parseHinge()	112
7.21.1.2 parsePID()	113
7.22 lib/UAV_common/src/parser/uav_params.hpp File Reference	113
7.23 lib/UAV_common/src/PID/PID.cpp File Reference	113
7.24 lib/UAV_common/src/PID/PID.hpp File Reference	113
7.24.1 Enumeration Type Documentation	113
7.24.1.1 AntiWindUpMode	113
7.25 lib/UAV_common/src/timed_loop/status.hpp File Reference	114
7.25.1 Enumeration Type Documentation	114
7.25.1.1 Status	114
7.26 lib/UAV_common/src/timed_loop/timed_loop.cpp File Reference	114
7.27 lib/UAV_common/src/timed_loop/timed_loop.hpp File Reference	115
7.28 src/communication/control.cpp File Reference	115
7.28.1 Function Documentation	115
7.28.1.1 orderServerJob()	115
7.29 src/communication/control.hpp File Reference	115
7.30 src/communication/control_recv.cpp File Reference	116
7.31 src/communication/control_send.cpp File Reference	116
7.32 src/controller/controller.cpp File Reference	116
7.33 src/controller/controller.hpp File Reference	116
7.34 src/controller/controller_loop.cpp File Reference	117
7.35 src/controller/controller_loop.hpp File Reference	117
7.36 src/controller/controller_mode.hpp File Reference	117
7.36.1 Enumeration Type Documentation	118
7.36.1.1 ControllerMode	118
7.36.2 Function Documentation	118
7.36.2.1 ControllerModeFromString()	118
7.36.2.2 ControllerModeToString()	118
7.37 src/controller/mixers.cpp File Reference	119
7.37.1 Function Documentation	119
7.37.1.1 applyMixerRotors()	119
7.37.1.2 applyMixerRotorsHover()	120
7.37.1.3 applyMixerSurfaces()	120
7.38 src/controller/mixers.hpp File Reference	121
7.38.1 Function Documentation	121
7.38.1.1 applyMixerRotors()	121
7.38.1.2 applyMixerRotorsHover()	121
7.38.1.3 applyMixerSurfaces()	122
7.39 src/controller/modes/controller_loop_FMANUAL.cpp File Reference	122
7.40 src/controller/modes/controller_loop_FMANUAL.hpp File Reference	123
7.41 src/controller/modes/controller_loop_NONE.cpp File Reference	123
7.42 src/controller/modes/controller_loop_NONE.hpp File Reference	123

7.43 src/controller/modes/controller_loop_QACRO.cpp File Reference	123
7.44 src/controller/modes/controller_loop_QACRO.hpp File Reference	123
7.45 src/controller/modes/controller_loop_QANGLE.cpp File Reference	124
7.46 src/controller/modes/controller_loop_QANGLE.hpp File Reference	124
7.47 src/controller/modes/controller_loop_QPOS.cpp File Reference	124
7.48 src/controller/modes/controller_loop_QPOS.hpp File Reference	124
7.49 src/controller/modes/controller_loop_RMANUAL.cpp File Reference	124
7.50 src/controller/modes/controller_loop_RMANUAL.hpp File Reference	124
7.51 src/defines.hpp File Reference	125
7.51.1 Macro Definition Documentation	125
7.51.1.1 USE_QUATERIONS	125
7.52 src/main.cpp File Reference	125
7.52.1 Macro Definition Documentation	126
7.52.1.1 LOGGER_MASK	126
7.52.2 Function Documentation	126
7.52.2.1 main()	126
7.52.2.2 parseArgs()	126
7.52.3 Variable Documentation	127
7.52.3.1 log_path	127
7.53 src/navigation/AHRS.cpp File Reference	127
7.54 src/navigation/AHRS.hpp File Reference	127
7.55 src/navigation/AHRS/AHRS_complementary.cpp File Reference	127
7.55.1 Function Documentation	128
7.55.1.1 calcRbn()	128
7.55.1.2 calcRnb()	128
7.55.1.3 calcTom()	128
7.55.1.4 clampOrientation()	128
7.56 src/navigation/AHRS/AHRS_complementary.hpp File Reference	128
7.57 src/navigation/AHRS/AHRS_EKF.cpp File Reference	129
7.57.1 Function Documentation	129
7.57.1.1 C()	129
7.57.1.2 S()	129
7.58 src/navigation/AHRS/AHRS_EKF.hpp File Reference	129
7.59 src/navigation/EKF.cpp File Reference	130
7.60 src/navigation/EKF.hpp File Reference	130
7.61 src/navigation/environment.cpp File Reference	130
7.61.1 Function Documentation	131
7.61.1.1 connectConflateSocket()	131
7.61.1.2 r_nb() [1/2]	131
7.61.1.3 r_nb() [2/2]	131
7.61.1.4 recvVectors()	131
7.62 src/navigation/environment.hpp File Reference	132

7.63 src/navigation/NS.cpp File Reference	132
7.64 src/navigation/NS.hpp File Reference	132
7.65 src/navigation/sensors.cpp File Reference	133
7.66 src/navigation/sensors.hpp File Reference	133
7.67 src/utlis.hpp File Reference	133
7.67.1 Function Documentation	134
7.67.1.1 circularError()	134
7.67.1.2 clampAngle()	134
7.67.1.3 safeGet()	135
7.67.1.4 safeSet()	135
Index	137

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

def	Controller constants	9
---------------------	--	-------------------

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

AeroCoefficients	12
AHRS	14
AHRS_EKF	19
AHRS_complementary	17
AHRSParams	22
Control	26
Controller	31
ControllerLoop	34
ControllerLoopFMANUAL	39
ControllerLoopNONE	40
ControllerLoopQACRO	41
ControllerLoopQANGLE	42
ControllerLoopQPOS	44
ControllerLoopRMANUAL	45
ControlSurfaces	47
Drive	49
Jet	65
Rotor	86
EKF	50
EKFParams	53
EKFScalers	55
Environment	56
Hinge	64
Load	68
Ammo	23
Cargo	25
Logger	70
NS	74
ODE	76
ODE_Euler	79
ODE_Heun	81
ODE_RK4	82
PID	84
Sensor< T >	88

Sensor< double >	88
Barometer	24
Sensor< Eigen::Vector3d >	88
Accelerometer	11
GPS	60
GPSVel	61
Gyroscope	63
Magnetometer	72
SensorParams	92
testing::TestWithParam	
ODETest	83
TimedLoop	93
UAVparams	95

Chapter 3

Class Index

3.1 Class List

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

Accelerometer	
Representation of accelerometer	11
AeroCoefficients	
Aerodynamic coefficient	12
AHRS	
Attitude and heading reference system	14
AHRS_complementary	
Implementation of AHRS based on Complementary Filter	17
AHRS_EKF	
Implementation of AHRS based on Extended Kalman Filter	19
AHRSParams	
AHRS parameters	22
Ammo	23
Barometer	
Representation of barometer	24
Cargo	25
Control	
Control command listener & sender	26
Controller	
Central controller class	31
ControllerLoop	
This class is interface of controller modes. All modes should keep this strucure and implements all true virtual methods	34
ControllerLoopFMANUAL	39
ControllerLoopNONE	40
ControllerLoopQACRO	41
ControllerLoopQANGLE	42
ControllerLoopQPOS	44
ControllerLoopRMANUAL	45
ControlSurfaces	
Aircraft's control surfaces	47
Drive	
Drive propelling aircraft	49
EKF	
Extended Kalman Filter	50

EKFPParams		
	EK filter parameters	53
EKFScalars		
	Scalars for EKF	55
Environment		56
GPS		
	Representation of GPS position measure	60
GPSVel		
	Representation of GPS velocity measure	61
Gyroscope		
	Representation of gyroscope	63
Hinge		
	Hinge connecting aircraft with drives	64
Jet		
	Jet rocket engine	65
Load		
	Load of aircraft that can be dropped or launched	68
Logger		
	Log vector data with timestamp in file	70
Magnetometer		
	Representation of magnetometer	72
NS		
	Navigation system	74
ODE		
	Ordinal differential equation solver	76
ODE_Euler		
	Explicit Euler algorithm	79
ODE_Heun		
	Second order explicit Heun algorithm	81
ODE_RK4		
	Fourth order Runge Kutta algorithm	82
ODETest		83
PID		
	PID discrete controller	84
Rotor		
	Rotor engine with controlled speed	86
Sensor< T >		
	Sensors base class	88
SensorParams		
	Base parameters of a sensor	92
TimedLoop		
	Simulation of real-time synchronized loop	93
UAVparams		
	Parsed UAV configuration from XML	95

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

lib/UAV_common/header/ common.hpp	103
lib/UAV_common/src/components/ aero_coefficients.hpp	103
lib/UAV_common/src/components/ components.hpp	103
lib/UAV_common/src/components/ control_surfaces.cpp	104
lib/UAV_common/src/components/ control_surfaces.hpp	104
lib/UAV_common/src/components/ drive.cpp	104
lib/UAV_common/src/components/ drive.hpp	104
lib/UAV_common/src/components/ hinge.cpp	104
lib/UAV_common/src/components/ hinge.hpp	105
lib/UAV_common/src/components/ loads.cpp	105
lib/UAV_common/src/components/ loads.hpp	105
lib/UAV_common/src/components/ navi.hpp	106
lib/UAV_common/src/logger/ logger.cpp	106
lib/UAV_common/src/logger/ logger.hpp	106
lib/UAV_common/src/ode/ ode.cpp	107
lib/UAV_common/src/ode/ ode.hpp	107
lib/UAV_common/src/ode/ ode_impl.hpp	107
lib/UAV_common/src/ode/ ode_test.cpp	108
lib/UAV_common/src/parser/ parser.cpp	110
lib/UAV_common/src/parser/ parser.hpp	111
lib/UAV_common/src/parser/ uav_params.cpp	112
lib/UAV_common/src/parser/ uav_params.hpp	113
lib/UAV_common/src/PID/ PID.cpp	113
lib/UAV_common/src/PID/ PID.hpp	113
lib/UAV_common/src/timed_loop/ status.hpp	114
lib/UAV_common/src/timed_loop/ timed_loop.cpp	114
lib/UAV_common/src/timed_loop/ timed_loop.hpp	115
src/ defines.hpp	125
src/ main.cpp	125
src/ utils.hpp	133
src/communication/ control.cpp	115
src/communication/ control.hpp	115
src/communication/ control_recv.cpp	116
src/communication/ control_send.cpp	116
src/controller/ controller.cpp	116

src/controller/controller.hpp	116
src/controller/controller_loop.cpp	117
src/controller/controller_loop.hpp	117
src/controller/controller_mode.hpp	117
src/controller/mixers.cpp	119
src/controller/mixers.hpp	121
src/controller/modes/controller_loop_FMANUAL.cpp	122
src/controller/modes/controller_loop_FMANUAL.hpp	123
src/controller/modes/controller_loop_NONE.cpp	123
src/controller/modes/controller_loop_NONE.hpp	123
src/controller/modes/controller_loop_QACRO.cpp	123
src/controller/modes/controller_loop_QACRO.hpp	123
src/controller/modes/controller_loop_QANGLE.cpp	124
src/controller/modes/controller_loop_QANGLE.hpp	124
src/controller/modes/controller_loop_QPOS.cpp	124
src/controller/modes/controller_loop_QPOS.hpp	124
src/controller/modes/controller_loop_RMANUAL.cpp	124
src/controller/modes/controller_loop_RMANUAL.hpp	124
src/navigation/AHRS.cpp	127
src/navigation/AHRS.hpp	127
src/navigation/EKF.cpp	130
src/navigation/EKF.hpp	130
src/navigation/environment.cpp	130
src/navigation/environment.hpp	132
src/navigation/NS.cpp	132
src/navigation/NS.hpp	132
src/navigation/sensors.cpp	133
src/navigation/sensors.hpp	133
src/navigation/AHRS/AHRS_complementary.cpp	127
src/navigation/AHRS/AHRS_complementary.hpp	128
src/navigation/AHRS/AHRS_EKF.cpp	129
src/navigation/AHRS/AHRS_EKF.hpp	129

Chapter 5

Namespace Documentation

5.1 def Namespace Reference

[Controller](#) constants.

Variables

- const double [STEP_TIME](#) = 0.003
Step time of controller. Step of [PID](#) and [EKF](#) calculations.
- const int [INFO_PERIOD](#) = 2
How often send demands in response to stick command.

5.1.1 Detailed Description

[Controller](#) constants.

5.1.2 Variable Documentation

5.1.2.1 INFO_PERIOD

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

How often send demands in response to stick command.

5.1.2.2 STEP_TIME

```
const double def::STEP_TIME = 0.003
```

Step time of controller. Step of [PID](#) and [EKF](#) calculations.

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](#) ()
Destructor.
- [Eigen::Vector3d](#) [getOri](#) ()
Returns estimated orientation vector (roll, pitch, yaw)
- virtual [Eigen::Vector3d](#) [getGyroBias](#) ()
Returns estimated 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

<i>env</i>	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 estimated 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](#) [quaternionToRPY](#) ([Eigen::Vector4d](#) *q*)
- [Eigen::Vector4d](#) [RPYToQuaternion](#) ([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 estimated gyroscope bias.

Returns

gyroscope bias

Reimplemented from [AHRS](#).

6.5.3.2 `q()`

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

6.5.3.3 `quaternionToRPY()`

```
Eigen::Vector3d AHRS_EKF::quaternionToRPY (
    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 AHRSPParams 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 AHRSPParams::alpha
```

6.6.2.2 Q

```
double AHRSPParams::Q
```

6.6.2.3 R

```
double AHRSPParams::R
```

6.6.2.4 type

```
std::string AHRSPParams::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 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

6.8.1 Detailed Description

Representation of barometer.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 Barometer()

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

6.8.3 Member Function Documentation

6.8.3.1 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](#)

6.9 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

6.9.1 Constructor & Destructor Documentation

6.9.1.1 Cargo() [1/2]

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

6.9.1.2 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](#)

6.10 Control Class Reference

[Control](#) command listener & sender.

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

Public Member Functions

- [Control](#) (zmq::context_t *ctx, std::string uav_address, [Controller](#) *controller)
Constructor.
- [~Control](#) ()
Destructor.
- 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 deflections.
- 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 deflection.
- std::string [handleMsg](#) (std::string msg)
Handle incomming control message - message that instruct controller what to do.

6.10.1 Detailed Description

[Control](#) command listener & sender.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 Control()

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

Constructor.

Parameters

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

6.10.2.2 ~Control()

```
Control::~~Control ( )
```

Deconstructor.

6.10.3 Member Function Documentation

6.10.3.1 handleMsg()

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

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

Parameters

<i>msg</i>	message content
------------	-----------------

Returns

reply to message

6.10.3.2 prepare()

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

Sends ping command.

6.10.3.3 recv()

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

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

6.10.3.4 sendHinge()

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

Sends command to control hinge deflection.

Parameters

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

6.10.3.5 sendSpeed()

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

Sends new demanded rotors speed.

Parameters

<i>speeds</i>	vector of demanded speeds
---------------	---------------------------

6.10.3.6 sendSurface()

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

Sends new demanded surface deflections.

Parameters

<i>speeds</i>	vector of surface deflections
---------------	-------------------------------

6.10.3.7 start()

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

Sends start command.

6.10.3.8 startJet()

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

Sends command to start jet engine of given index.

Parameters

index	jet engine index
-----------------------	------------------

6.10.3.9 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](#)

6.11 Controller Class Reference

Central controller class.

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

Public Member Functions

- [Controller](#) (zmq::context_t *ctx, std::string uav_address)
Constructor.
- [~Controller](#) ()
- void [run](#) ()
Run controller.
- void [setMode](#) ([ControllerMode](#) new_mode)
Change controller mode.
- void [exitController](#) ()
Stop controller loop.

Friends

- class [Control](#)

6.11.1 Detailed Description

Central controller class.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 Controller()

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

Constructor.

Parameters

<i>ctx</i>	zero mq context
<i>uav_address</i>	address of simulation sockets

6.11.2.2 ~Controller()

```
Controller::~~Controller ( )
```

6.11.3 Member Function Documentation**6.11.3.1 exitController()**

```
void Controller::exitController ( )
```

Stop controller loop.

6.11.3.2 run()

```
void Controller::run ( )
```

Run controller.

6.11.3.3 setMode()

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

Change controller mode.

Parameters

<i>new_mode</i>	new controller mode
-----------------	---------------------

6.11.4 Friends And Related Function Documentation

6.11.4.1 Control

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

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

- [src/controller/controller.hpp](#)
- [src/controller/controller.cpp](#)

6.12 ControllerLoop Class Reference

This class is interface of controller modes. All modes should keep this structure 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 destructor for defined behavior.
- virtual void [job](#) ([[maybe_unused]] std::map< std::string, [PID](#) > &pids, [[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 incoming joystick deflection.
- virtual std::string [demandInfo](#) ()
Prepare info about state and demands.
- virtual const std::vector< std::string > & [requiredPIDs](#) ()
Defines pids controller required by mode.
- virtual void [overridePosition](#) ([[maybe_unused]] Eigen::Vector3d position, [[maybe_unused]] Eigen::Vector3d orientation)
Overrides demands to apply to given position and orientation.
- [ControllerMode](#) [getMode](#) ()
Returns assigned mode enum value.

Static Public Member Functions

- static [ControllerLoop](#) * [ControllerLoopFactory](#) ([ControllerMode](#) mode)
[ControllerLoop](#) factor. Returns instance 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_pids](#)

6.12.1 Detailed Description

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

6.12.2 Constructor & Destructor Documentation

6.12.2.1 ControllerLoop()

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

Base class constructor.

Parameters

<i>mode</i>	mode enum value
-------------	-----------------

6.12.2.2 ~ControllerLoop()

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

Virtual destructor for defined behavior.

6.12.3 Member Function Documentation

6.12.3.1 checkJoystickLength()

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

Check if joystick input vector is correct.

Parameters

<i>joystick</i>	joystick axes deflection
<i>minimalSize</i>	minimal length of deflation vector that can be interpreted

Returns

return true if joystick input vector is long enough

6.12.3.2 ControllerLoopFactory()

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

[ControllerLoop](#) factor. Returns instace of [ControllerLoop](#) that implements specified mode.

Parameters

<i>mode</i>	demanded mode
-------------	---------------

Returns

Pointer to dynamically allocated [ControllerLoop](#)

6.12.3.3 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](#), [ControllerLoopQPOS](#), [ControllerLoopQANGLE](#), and [ControllerLoopQACRO](#).

6.12.3.4 getMode()

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

Returns assigned mode enum value.

Returns

mode enum value

6.12.3.5 handleJoystick()

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

Handle incoming joystick deflection.

Parameters

<i>joystick</i>	joystick axes deflection
-----------------	--------------------------

6.12.3.6 job()

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

[Controller](#) job that will be called in control loop.

Parameters

<i>pids</i>	map of available pid controllers
<i>control</i>	reference to control instance that is used to send control commands
<i>navisys</i>	navigation system reference

6.12.3.7 overridePosition()

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

Overrides demands to apply to given position and orientation.

Parameters

<i>position</i>	position vector in world frame
<i>orientation</i>	orientation vector in world frame

6.12.3.8 requiredPIDs()

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

Defines pids controller required by mode.

Returns

vector of names of required pids

6.12.4 Member Data Documentation**6.12.4.1 `_mode`**

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

6.12.4.2 `required_pids`

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

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

- [src/controller/controller_loop.hpp](#)
- [src/controller/controller_loop.cpp](#)

6.13 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, [PID](#) > &pids, [Control](#) &control, [[maybe_unused]] [NS](#) &navsys) override
- void [handleJoystick](#) (Eigen::VectorXd joystick) override

Additional Inherited Members**6.13.1 Constructor & Destructor Documentation**

6.13.1.1 ControllerLoopFMANUAL()

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

6.13.2 Member Function Documentation

6.13.2.1 handleJoystick()

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

6.13.2.2 job()

```
void ControllerLoopFMANUAL::job (
    [[maybe_unused]] std::map< std::string, PID > & pids,
    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](#)

6.14 ControllerLoopNONE Class Reference

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

Inheritance diagram for ControllerLoopNONE:

Collaboration diagram for ControllerLoopNONE:

Public Member Functions

- [ControllerLoopNONE](#) ()

Additional Inherited Members

6.14.1 Constructor & Destructor Documentation

6.14.1.1 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](#)

6.15 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, [PID](#) > &pids, [Control](#) &control, [NS](#) &navisys) override
- void [handleJoystick](#) (Eigen::VectorXd joystick) override
- std::string [demandInfo](#) () override

Prepare info about state and demands.

Additional Inherited Members

6.15.1 Constructor & Destructor Documentation

6.15.1.1 ControllerLoopQACRO()

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

6.15.2 Member Function Documentation

6.15.2.1 demandInfo()

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

Prepare info about state and demands.

Returns

information about mode and actually set demands

Reimplemented from [ControllerLoop](#).

6.15.2.2 handleJoystick()

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

6.15.2.3 job()

```
void ControllerLoopQACRO::job (
    std::map< std::string, PID > & pids,
    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](#)

6.16 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, PID > &pids, [Control](#) &control, [NS](#) &navisys) override
- void [handleJoystick](#) (Eigen::VectorXd joystick) override
- std::string [demandInfo](#) () override

Prepare info about state and demands.
- void [overridePosition](#) (Eigen::Vector3d position, Eigen::Vector3d orientation) override

Additional Inherited Members

6.16.1 Constructor & Destructor Documentation

6.16.1.1 ControllerLoopQANGLE()

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

6.16.2 Member Function Documentation

6.16.2.1 demandInfo()

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

Prepare info about state and demands.

Returns

information about mode and actually set demands

Reimplemented from [ControllerLoop](#).

6.16.2.2 handleJoystick()

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

6.16.2.3 job()

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

6.16.2.4 overridePosition()

```
void ControllerLoopQANGLE::overridePosition (
    Eigen::Vector3d position,
    Eigen::Vector3d orientation ) [override]
```

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](#)

6.17 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, [PID](#) > &pids, [Control](#) &control, [NS](#) &navisys) override
- void [handleJoystick](#) (Eigen::VectorXd joystick) override
- std::string [demandInfo](#) () override
 - Prepare info about state and demands.*
- void [overridePosition](#) (Eigen::Vector3d position, Eigen::Vector3d orientation) override

Additional Inherited Members

6.17.1 Constructor & Destructor Documentation

6.17.1.1 ControllerLoopQPOS()

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

6.17.2 Member Function Documentation

6.17.2.1 demandInfo()

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

Prepare info about state and demands.

Returns

information about mode and actually set demands

Reimplemented from [ControllerLoop](#).

6.17.2.2 handleJoystick()

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

6.17.2.3 job()

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

6.17.2.4 overridePosition()

```
void ControllerLoopQPOS::overridePosition (
    Eigen::Vector3d position,
    Eigen::Vector3d orientation ) [override]
```

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](#)

6.18 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, [PID](#) > &pids, [Control](#) &control, [[maybe_unused]] [NS](#) &navsys) override
- void [handleJoystick](#) (Eigen::VectorXd joystick) override
- std::string [demandInfo](#) () override

Prepare info about state and demands.

Protected Attributes

- double [demandedX](#)
- double [demandedY](#)

Additional Inherited Members

6.18.1 Constructor & Destructor Documentation

6.18.1.1 ControllerLoopRMANUAL()

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

6.18.2 Member Function Documentation

6.18.2.1 demandInfo()

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

Prepare info about state and demands.

Returns

information about mode and actually set demands

Reimplemented from [ControllerLoop](#).

6.18.2.2 handleJoystick()

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


6.18.2.3 job()

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

6.18.3 Member Data Documentation

6.18.3.1 demandedX

```
double ControllerLoopRMANUAL::demandedX [protected]
```

6.18.3.2 demandedY

```
double ControllerLoopRMANUAL::demandedY [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

6.19 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

6.19.1 Detailed Description

Aircraft's control surfaces.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 ControlSurfaces() [1/2]

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

6.19.2.2 ControlSurfaces() [2/2]

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

Constructor.

Parameters

<i>noOfSurfaces</i>	number of independent surfaces
<i>matrix</i>	coefficients matrix
<i>min</i>	vector of min angles
<i>max</i>	vector of max angles
<i>trim</i>	vector of trim angles

6.19.3 Member Function Documentation

6.19.3.1 getCoefficients()

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

6.19.3.2 getNoOfSurface()

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

6.19.3.3 getValues()

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

6.19.3.4 restoreTrim()

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

6.19.3.5 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](#)

6.20 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]

6.20.1 Detailed Description

[Drive](#) propelling aircraft.

6.20.2 Member Data Documentation

6.20.2.1 axis

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

6.20.2.2 hinges

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

6.20.2.3 noOfHinges

```
int Drive::noOfHinges
```

6.20.2.4 position

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

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

- [lib/UAV_common/src/components/drive.hpp](#)

6.21 EKF Class Reference

Extended Kalman Filter.

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

Public Member Functions

- [EKF](#) ([EKFPParams](#) 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.

6.21.1 Detailed Description

Extended Kalman Filter.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 EKF()

```
EKF::EKF (
    EKFPParams params )
```

Constructor.

Parameters

<i>params</i>	filter parameters
---------------	-------------------

6.21.3 Member Function Documentation

6.21.3.1 getPos()

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

Returns estimated position vector.

Returns

position vector in world frame

6.21.3.2 getVel()

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

Returns estimated velocity vector.

Returns

velocity vector in world frame

6.21.3.3 log()

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

Log filter state.

Parameters

<i>time</i>	simulation time
-------------	-----------------

6.21.3.4 predict()

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

Predict phase. Integration of accelerometer measures.

Parameters

<i>time</i>	simulation time
<i>acc</i>	accelerometer measure

6.21.3.5 updateBaro()

```
void EKF::updateBaro (
```

```
double time,
double baro )
```

Update phase. Height correction.

Parameters

<i>time</i>	simulation time
<i>baro</i>	barometer measure

6.21.3.6 updateGPS()

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

Update phase. Position correction.

Parameters

<i>time</i>	simulation time
<i>baro</i>	GPS location measure

6.21.3.7 updateGPSVel()

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

Update phase. Velocity correction.

Parameters

<i>time</i>	simulation time
<i>baro</i>	GPS velocity measure

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

- [src/navigation/EKF.hpp](#)
- [src/navigation/EKF.cpp](#)

6.22 EKFPARAMS Struct Reference

EK filter 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](#)

6.22.1 Detailed Description

EK filer parameters.

6.22.2 Member Data Documentation

6.22.2.1 P0

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

6.22.2.2 Q

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

6.22.2.3 RBaro

`double EKFPARAMS::RBaro`

6.22.2.4 RGPSPos

`Eigen::Matrix3d EKFPARAMS::RGPSPos`

6.22.2.5 RGPSVel

`Eigen::Matrix3d EKFPARAMS::RGPSVel`

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

- `src/navigation/EKF.hpp`

6.23 EKFScalers Struct Reference

Scalers for [EKF](#).

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

Public Attributes

- double [predictScaler](#)
- double [updateScaler](#)
- double [baroScaler](#)
- double [zScaler](#)

6.23.1 Detailed Description

Scalers for [EKF](#).

6.23.2 Member Data Documentation

6.23.2.1 [baroScaler](#)

```
double EKFScalers::baroScaler
```

6.23.2.2 [predictScaler](#)

```
double EKFScalers::predictScaler
```

6.23.2.3 [updateScaler](#)

```
double EKFScalers::updateScaler
```

6.23.2.4 [zScaler](#)

```
double EKFScalers::zScaler
```

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

- [lib/UAV_common/src/components/navi.hpp](#)

6.24 Environment Class Reference

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

Public Member Functions

- [Environment](#) (zmq::context_t *ctx, std::string uav_address)
Constructor.
- [~Environment](#) ()
Destructor.
- double [getTime](#) ()
Returns time of simulation.
- Eigen::Vector3d [getPosition](#) ()
Returns exact position 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

6.24.1 Constructor & Destructor Documentation

6.24.1.1 Environment()

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

Constructor.

Parameters

<i>ctx</i>	zero mq context
<i>uav_address</i>	address to state PUB socket that enviroment should listen

6.24.1.2 `~Environment()`

```
Environment::~~Environment ( )
```

Deconstructor.

6.24.2 Member Function Documentation**6.24.2.1 `getAngularAcceleraton()`**

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

Returns exact angular acceleration vector.

Returns

angular acceleration vector in body frame

6.24.2.2 `getAngularVelocity()`

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

Returns exact angular velocity vector.

Returns

angular velocities vector in body frame

6.24.2.3 getLinearAcceleration()

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

Returns exact linear acceleration vector.

Returns

linear acceleration vector in body frame

6.24.2.4 getLinearVelocity()

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

Returns exact linear velocity vector.

Returns

linear velocity vector in body frame

6.24.2.5 getOrientation()

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

Returns exact orientation vector.

Returns

orientation vector in world frame

6.24.2.6 getPosition()

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

Returns exact position vector.

Returns

position vector in world frame

6.24.2.7 getRnb()

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

Get rotation matrix from world to body frame.

Returns

rotation matrix

6.24.2.8 getTime()

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

Returns time of simulation.

Returns

simulation time

6.24.2.9 getWorldAngularVelocity()

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

Returns exact angular velocity vector.

Returns

linear angular vector in world frame

6.24.2.10 getWorldLinearVelocity()

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

Returns exact linear velocity vector.

Returns

linear velocity vector in world frame

6.24.2.11 updateSensors()

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

update all sensors

6.24.3 Member Data Documentation

6.24.3.1 sensors

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

map of sensors that measure single value

6.24.3.2 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](#)

6.25 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

6.25.1 Detailed Description

Representation of [GPS](#) position measure.

6.25.2 Constructor & Destructor Documentation

6.25.2.1 GPS()

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

6.25.3 Member Function Documentation

6.25.3.1 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](#)

6.26 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

6.26.1 Detailed Description

Representation of [GPS](#) velocity measure.

6.26.2 Constructor & Destructor Documentation

6.26.2.1 GPSVel()

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

6.26.3 Member Function Documentation

6.26.3.1 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](#)

6.27 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

6.27.1 Detailed Description

Representation of gyroscope.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 Gyroscope()

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

6.27.3 Member Function Documentation

6.27.3.1 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](#)

6.28 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 rotation matrix of orientation change due to hinge.

6.28.1 Detailed Description

[Hinge](#) connecting aircraft with drives.

6.28.2 Constructor & Destructor Documentation

6.28.2.1 Hinge() [1/3]

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

6.28.2.2 Hinge() [2/3]

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

6.28.2.3 Hinge() [3/3]

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

6.28.3 Member Function Documentation

6.28.3.1 getRot()

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

Get rotation matrix of orientation change due to hinge.

Returns

rotation matrix

6.28.3.2 operator=()

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

6.28.3.3 updateValue()

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

set new angle on hinge

Parameters

<i>newValue</i>	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](#)

6.29 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](#)

6.29.1 Detailed Description

[Jet](#) rocket engine.

6.29.2 Member Function Documentation

6.29.2.1 [getLastThrust\(\)](#)

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

get last calculated thrust

Returns

last calculated thrust

6.29.2.2 [getThrust\(\)](#)

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

get thrust in specific time

Parameters

<i>time</i>	timestamp
-------------	-----------

Returns

thrust value in Newtons

6.29.2.3 start()

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

start jet engine

Parameters

<i>time</i>	timestamp of start
-------------	--------------------

Returns

true if start succesful, false if already started

6.29.3 Member Data Documentation**6.29.3.1 phases**

```
int Jet::phases
```

6.29.3.2 thrust

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

6.29.3.3 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](#)

6.30 Load Class Reference

[Load](#) of aircraft that can be dropped or launched.

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

Inheritance diagram for Load:

Public Member Functions

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

Protected Member Functions

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

6.30.1 Detailed Description

[Load](#) of aircraft that can be dropped or launched.

6.30.2 Constructor & Destructor Documentation

6.30.2.1 Load() [1/2]

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

6.30.2.2 Load() [2/2]

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

6.30.3 Member Function Documentation

6.30.3.1 getMass()

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

get mass of load

Returns

mass

6.30.3.2 getOffset()

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

get offset of load

Returns

offset vector

6.30.3.3 operator=()

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

6.30.3.4 release()

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

Try to release load.

Parameters

<i>time</i>	
-------------	--

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](#)

6.31 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.

6.31.1 Detailed Description

Log vector data with timestamp in file.

6.31.2 Constructor & Destructor Documentation

6.31.2.1 Logger()

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

Constructor.

Parameters

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

6.31.2.2 ~Logger()

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

Deconstructor.

6.31.3 Member Function Documentation**6.31.3.1 log() [1/2]**

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

Log one row.

Parameters

<i>time</i>	timestamp
<i>args</i>	list of doubles

6.31.3.2 log() [2/2]

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

Log one row.

Parameters

<i>time</i>	timestamp
<i>args</i>	list of double vectors

6.31.3.3 setFmt()

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

Set new format if was not known in constructor.

Parameters

<i>fmt</i>	new format
------------	------------

6.31.3.4 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

<i>subdirectory</i>	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](#)

6.32 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

6.32.1 Detailed Description

Representation of magnetometer.

6.32.2 Constructor & Destructor Documentation

6.32.2.1 Magnetometer()

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

6.32.3 Member Function Documentation

6.32.3.1 update()

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

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

Implements [Sensor< Eigen::Vector3d >](#).

6.32.4 Member Data Documentation

6.32.4.1 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](#)

6.33 NS Class Reference

Navigation system.

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

Public Member Functions

- [NS](#) ([Environment](#) &env)
Constructor.
- [~NS](#) ()
Destructor.
- 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](#).

6.33.1 Detailed Description

Navigation system.

6.33.2 Constructor & Destructor Documentation

6.33.2.1 NS()

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

Constructor.

Parameters

<i>env</i>	reference to environment, that NS navigate through
------------	--

6.33.2.2 ~NS()

```
NS::~NS ( )
```

Deconstructor.

6.33.3 Member Function Documentation

6.33.3.1 getAngularVelocity()

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

Returns rates estimated by [NS](#).

Returns

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

6.33.3.2 getLinearVelocity()

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

Returns linear velocity estimated by [NS](#).

Returns

linear velocity vector in world frame

6.33.3.3 getOrientation()

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

Returns orientation estimated by [NS](#).

Returns

orientation vector (RPY) in world frame

6.33.3.4 getPosition()

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

Returns position estimated by [NS](#).

Returns

position vector in world frame

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

- src/navigation/[NS.hpp](#)
- src/navigation/[NS.cpp](#)

6.34 ODE Class Reference

Ordinal differential equation solver.

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

Inheritance diagram for ODE:

Public Types

- enum [ODEMethod](#) { [Euler](#) , [Heun](#) , [RK4](#) , [NONE](#) }
- Supported solving method.*

Public Member Functions

- [ODE](#) (int micro_steps)
Constructor.
- virtual [~ODE](#) ()
Virtual destructor.
- 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.

6.34.1 Detailed Description

Ordinal differential equation solver.

6.34.2 Member Enumeration Documentation

6.34.2.1 ODEMethod

```
enum ODE::ODEMethod
```

Supported solving method.

Enumerator

Euler	
Heun	
RK4	
NONE	

6.34.3 Constructor & Destructor Documentation

6.34.3.1 ODE()

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

Constructor.

6.34.3.2 ~ODE()

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

Virtual destructor.

6.34.4 Member Function Documentation

6.34.4.1 factory()

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

Factory constructing ODE solvers.

Parameters

<i>method</i>	type of desired method
---------------	------------------------

Returns

instance of [ODE](#) solver

6.34.4.2 fromString()

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

Parse solving method from string.

Parameters

<i>str</i>	input string
------------	--------------

Returns

solving method if parsed, NONE if unknown

6.34.4.3 getMicrosteps() [1/2]

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

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

Returns

microsteps

6.34.4.4 getMicrosteps() [2/2]

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

Get microsteps of given method.

Parameters

<i>method</i>	method type
---------------	-------------

Returns

number of microstep in one algorithm step

6.34.4.5 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

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

Returns

Implemented in [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](#)

6.35 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

6.35.1 Detailed Description

Explicit Euler algorithm.

6.35.2 Constructor & Destructor Documentation

6.35.2.1 ODE_Euler()

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

6.35.3 Member Function Documentation

6.35.3.1 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

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

Returns

Implements [ODE](#).

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

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

6.36 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

6.36.1 Detailed Description

Second order explicit Heun algorithm.

6.36.2 Constructor & Destructor Documentation

6.36.2.1 ODE_Heun()

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

6.36.3 Member Function Documentation

6.36.3.1 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

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

Returns

Implements [ODE](#).

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

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

6.37 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

6.37.1 Detailed Description

Fourth order Runge Kutta algorithm.

6.37.2 Constructor & Destructor Documentation

6.37.2.1 ODE_RK4()

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

6.37.3 Member Function Documentation

6.37.3.1 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

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

Returns

Implements [ODE](#).

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

- [lib/UAV_common/src/ode/ode_impl.hpp](#)

6.38 ODETest Class Reference

Inheritance diagram for ODETest:

Collaboration diagram for ODETest:

Protected Member Functions

- void [SetUp](#) () override
- void [TearDown](#) () override

6.38.1 Member Function Documentation

6.38.1.1 Setup()

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

6.38.1.2 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](#)

6.39 PID Class Reference

[PID](#) discrete controller.

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

Public Member Functions

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

6.39.1 Detailed Description

[PID](#) discrete controller.

6.39.2 Constructor & Destructor Documentation

6.39.2.1 PID()

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

6.39.2.2 ~PID()

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

6.39.3 Member Function Documentation

6.39.3.1 calc() [1/2]

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

calc output of controller

Parameters

<i>error</i>	input of controller
--------------	---------------------

Returns

output of controller

6.39.3.2 calc() [2/2]

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

calc output of controller with specific time step

Parameters

<i>error</i>	input of controller
<i>dt</i>	time step

Returns

output of controller

6.39.3.3 clear()

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

clear internal state

6.39.3.4 set_dt()

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

Set new time step.

Parameters

<i>dt</i>	new time step
-----------	---------------

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

- [lib/UAV_common/src/PID/PID.hpp](#)
- [lib/UAV_common/src/PID/PID.cpp](#)

6.40 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](#)

6.40.1 Detailed Description

[Rotor](#) engine with controlled speed.

6.40.2 Member Data Documentation

6.40.2.1 direction

```
int Rotor::direction
```

6.40.2.2 forceCoff

```
double Rotor::forceCoff
```

6.40.2.3 hoverSpeed

```
double Rotor::hoverSpeed
```

6.40.2.4 maxSpeed

```
double Rotor::maxSpeed
```

6.40.2.5 timeConstant

```
double Rotor::timeConstant
```

6.40.2.6 torqueCoff

```
double Rotor::torqueCoff
```

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

- [lib/UAV_common/src/components/drive.hpp](#)

6.41 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())()

6.41.1 Detailed Description

```
template<class T>
class Sensor< T >
```

Sensors base class.

Template Parameters

<i>T</i>	type of data read by sensor
----------	-----------------------------

6.41.2 Constructor & Destructor Documentation

6.41.2.1 Sensor()

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

Constructor.

Parameters

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

6.41.3 Member Function Documentation

6.41.3.1 error()

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

6.41.3.2 getReading()

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

Returns recent measure.

Returns

sensor measure

6.41.3.3 getSd()

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

Returns standard deviation.

Returns

standard deviation

6.41.3.4 isReady()

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

Checks if sensor is ready.

Returns

true if sensor is ready

6.41.3.5 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

6.41.3.6 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](#).

6.41.4 Member Data Documentation

6.41.4.1 bias

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

6.41.4.2 dist

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

6.41.4.3 env

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

6.41.4.4 gen

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

6.41.4.5 lastUpdate

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

6.41.4.6 logger

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

6.41.4.7 ready

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

6.41.4.8 refreshTime

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

6.41.4.9 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](#)

6.42 SensorParams Struct Reference

Base parameters of a sensor.

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

Public Attributes

- std::string [name](#)
- double [sd](#)
- Eigen::Vector3d [bias](#)
- double [refreshTime](#)

6.42.1 Detailed Description

Base parameters of a sensor.

6.42.2 Member Data Documentation

6.42.2.1 bias

`Eigen::Vector3d SensorParams::bias`

6.42.2.2 name

`std::string SensorParams::name`

6.42.2.3 refreshTime

`double SensorParams::refreshTime`

6.42.2.4 sd

`double SensorParams::sd`

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

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

6.43 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

6.43.1 Detailed Description

Simulation of real-time synchronized loop.

6.43.2 Constructor & Destructor Documentation

6.43.2.1 TimedLoop()

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

Constructor.

Parameters

<i>periodInMs</i>	loop period in milliseconds
<i>func</i>	function that should be called in loop
<i>status</i>	reference to controlling status

6.43.3 Member Function Documentation

6.43.3.1 go() [1/2]

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

start infinite loop

6.43.3.2 go() [2/2]

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

start loop for specific cycle numbers

Parameters

<i>loops</i>	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](#)

6.44 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 [getRotorTimeConstants](#) () const
- Eigen::VectorXd [getRotorMaxSpeeds](#) () const
- Eigen::VectorXd [getRotorHoverSpeeds](#) () const

Static Public Member Functions

- static const [UAVparams](#) * [getSingleton](#) ()

Public Attributes

- std::string [name](#)
- bool [instantRun](#)
- std::string [initialMode](#)
- Eigen::Vector3d [initialPosition](#)
- Eigen::Vector3d [initialOrientation](#)
- Eigen::Vector3d [initialVelocity](#)
- double [m](#)
- double [lx](#)
- double [ly](#)
- double [lz](#)
- double [lxy](#)
- double [lxz](#)
- double [lyz](#)
- int [noOfRotors](#)
- std::unique_ptr< [Rotor](#)[]> [rotors](#)
- int [noOfJets](#)
- std::unique_ptr< [Jet](#)[]> [jets](#)
- [ControlSurfaces](#) [surfaces](#)
- [AeroCoefficients](#) [aero_coffs](#)
- std::map< std::string, [PID](#) > [pids](#)
- std::vector< [SensorParams](#) > [sensors](#)
- [AHRSParams](#) [ahrs](#)
- [EKFSalers](#) [ekf](#)
- Eigen::MatrixX4d [rotorMixer](#)
- Eigen::MatrixX4d [surfaceMixer](#)
- int [noOfAmmo](#)
- std::unique_ptr< [Ammo](#)[]> [ammo](#)
- int [noOfCargo](#)
- std::unique_ptr< [Cargo](#)[]> [cargo](#)

6.44.1 Detailed Description

Parsed UAV configuration from XML.

6.44.2 Constructor & Destructor Documentation

6.44.2.1 UAVparams()

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

Initialize default data.

6.44.2.2 ~UAVparams()

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

6.44.3 Member Function Documentation

6.44.3.1 getRotorHoverSpeeds()

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

6.44.3.2 getRotorMaxSpeeds()

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

6.44.3.3 getRotorTimeContants()

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

6.44.3.4 getSingleton()

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

6.44.3.5 loadConfig()

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

6.44.4 Member Data Documentation

6.44.4.1 aero_coffs

[AeroCoefficients](#) UAVparams::aero_coffs

6.44.4.2 ahrs

[AHRSParams](#) UAVparams::ahrs

6.44.4.3 ammo

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

6.44.4.4 cargo

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

6.44.4.5 ekf

[EKFScalers](#) UAVparams::ekf

6.44.4.6 initialMode

`std::string` UAVparams::initialMode

6.44.4.7 initialOrientation

`Eigen::Vector3d UAVparams::initialOrientation`

6.44.4.8 initialPosition

`Eigen::Vector3d UAVparams::initialPosition`

6.44.4.9 initialVelocity

`Eigen::Vector3d UAVparams::initialVelocity`

6.44.4.10 instantRun

`bool UAVparams::instantRun`

6.44.4.11 Ix

`double UAVparams::Ix`

6.44.4.12 Ixy

`double UAVparams::Ixy`

6.44.4.13 Ixz

`double UAVparams::Ixz`

6.44.4.14 Iy

`double UAVparams::Iy`

6.44.4.15 Iyz

```
double UAVparams::Iyz
```

6.44.4.16 Iz

```
double UAVparams::Iz
```

6.44.4.17 jets

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

6.44.4.18 m

```
double UAVparams::m
```

6.44.4.19 name

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

6.44.4.20 noOfAmmo

```
int UAVparams::noOfAmmo
```

6.44.4.21 noOfCargo

```
int UAVparams::noOfCargo
```

6.44.4.22 noOfJets

```
int UAVparams::noOfJets
```

6.44.4.23 noOfRotors

```
int UAVparams::noOfRotors
```

6.44.4.24 pids

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

6.44.4.25 rotorMixer

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

6.44.4.26 rotors

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

6.44.4.27 sensors

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

6.44.4.28 surfaceMixer

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

6.44.4.29 surfaces

```
ControlSurfaces UAVparams::surfaces
```

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](#)

Chapter 7

File Documentation

7.1 lib/UAV_common/header/common.hpp File Reference

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

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

7.2 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.

7.3 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:

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

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

Include dependency graph for control_surfaces.cpp:

7.5 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.

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

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

Include dependency graph for drive.cpp:

7.7 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.

7.8 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)

7.8.1 Function Documentation

7.8.1.1 asSkewMatrix()

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

7.9 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.

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

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

Include dependency graph for loads.cpp:

7.11 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 dropped or launched.
- class [Ammo](#)
- class [Cargo](#)

7.12 lib/UAU_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.

7.13 lib/UAU_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)

7.13.1 Function Documentation

7.13.1.1 shouldLog()

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

7.14 lib/UAU_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

7.14.1 Macro Definition Documentation

7.14.1.1 `LOGGER_MASK`

```
#define LOGGER_MASK -1
```

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

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

7.16 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.

7.17 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.

7.18 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)`
- `INSTANTIATE_TEST_SUITE_P (TestDerivedClasses, ODETest, testing::ValuesIn(getMethodsToTest()))`
- `int main (int argc, char **argv)`

7.18.1 Function Documentation

7.18.1.1 [getMethodsToTest\(\)](#)

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

7.18.1.2 [INSTANTIATE_TEST_SUITE_P\(\)](#)

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

7.18.1.3 main()

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

7.18.1.4 TEST_F() [1/2]

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

7.18.1.5 TEST_F() [2/2]

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

7.18.1.6 TEST_P() [1/3]

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

7.18.1.7 TEST_P() [2/3]

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

7.18.1.8 TEST_P() [3/3]

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

7.19 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.

7.19.1 Function Documentation

7.19.1.1 parseMatrixXd()

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

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

Parameters

<i>input</i>	input string
<i>R</i>	number of rows
<i>C</i>	number of columns
<i>delimiter</i>	delimiter

Returns

parsed matrix

7.19.1.2 parseVectorXd()

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

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

Parameters

<i>str</i>	input string
<i>noOfElem</i>	length of vector
<i>delimiter</i>	delimiter

Returns

parsed vector

7.20 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.

7.20.1 Function Documentation

7.20.1.1 parseMatrixXd()

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

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

Parameters

<i>input</i>	input string
<i>R</i>	number of rows
<i>C</i>	number of columns
<i>delimiter</i>	delimiter

Returns

parsed matrix

7.20.1.2 parseVectorXd()

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

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

Parameters

<i>str</i>	input string
<i>noOfElem</i>	length of vector
<i>delimiter</i>	delimiter

Returns

parsed vector

7.21 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)
- [PID](#) [parsePID](#) (rapidxml::xml_node<> *PIDNode)

7.21.1 Function Documentation

7.21.1.1 parseHinge()

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

7.21.1.2 parsePID()

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

7.22 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 "../PID/PID.hpp"
```

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

Classes

- struct [UAVparams](#)
Parsed UAV configuration from XML.

7.23 lib/UAV_common/src/PID/PID.cpp File Reference

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

Include dependency graph for PID.cpp:

7.24 lib/UAV_common/src/PID/PID.hpp File Reference

```
#include <limits>
```

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

Classes

- class [PID](#)
PID discrete controller.

Enumerations

- enum [AntiWindUpMode](#) { [None](#) , [Clamping](#) }
Methods of handling windup in controller.

7.24.1 Enumeration Type Documentation

7.24.1.1 AntiWindUpMode

```
enum AntiWindUpMode
```

Methods of handling windup in controller.

Enumerator

None	
Clamping	

7.25 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*

7.25.1 Enumeration Type Documentation

7.25.1.1 Status

enum [Status](#)

status of timed loop. [Control](#) it's job

Enumerator

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

7.26 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:

7.27 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.

7.28 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)

7.28.1 Function Documentation

7.28.1.1 orderServerJob()

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

7.29 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.

7.30 src/communication/control_recv.cpp File Reference

```
#include "control.hpp"
#include <iostream>
#include "../defines.hpp"
Include dependency graph for control_recv.cpp:
```

7.31 src/communication/control_send.cpp File Reference

```
#include "control.hpp"
#include <iostream>
Include dependency graph for control_send.cpp:
```

7.32 src/controller/controller.cpp File Reference

```
#include "controller.hpp"
#include <iostream>
#include "../defines.hpp"
Include dependency graph for controller.cpp:
```

7.33 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 [Controller](#)

Central controller class.

7.34 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_RMANUAL.hpp"
#include "modes/controller_loop_FMANUAL.hpp"
Include dependency graph for controller_loop.cpp:
```

7.35 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 structure and implements all true virtual methods.

7.36 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 }
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.

7.36.1 Enumeration Type Documentation

7.36.1.1 ControllerMode

enum `ControllerMode`

`Controller` modes.

Enumerator

NONE	
QPOS	
QANGLE	
QACRO	
FMANUAL	
FACRO	
FANGLE	
RMANUAL	

7.36.2 Function Documentation

7.36.2.1 ControllerModeFromString()

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

Parse string to controller mode.

Parameters

<i>mode</i>	string to parse
-------------	-----------------

Returns

parsing result, NONE if parse failed

7.36.2.2 ControllerModeToString()

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

Serializes controller mode to string.

Parameters

<i>mode</i>	controller mode
-------------	-----------------

Returns

serialized mode

7.37 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.

7.37.1 Function Documentation

7.37.1.1 [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

<i>climb_rate</i>	
<i>roll_rate</i>	
<i>pitch_rate</i>	
<i>yaw_rate</i>	

Returns

Rotors demanded speed

7.37.1.2 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

<i>throttle</i>	
<i>roll_rate</i>	
<i>pitch_rate</i>	
<i>yaw_rate</i>	

Returns

Rotors demanded speed

7.37.1.3 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

<i>throttle</i>	
<i>roll_rate</i>	
<i>pitch_rate</i>	
<i>yaw_rate</i>	

Returns

demanded surfaces deflection

7.38 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.

7.38.1 Function Documentation

7.38.1.1 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

<i>climb_rate</i>	
<i>roll_rate</i>	
<i>pitch_rate</i>	
<i>yaw_rate</i>	

Returns

Rotors demanded speed

7.38.1.2 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

<i>throttle</i>	
<i>roll_rate</i>	
<i>pitch_rate</i>	
<i>yaw_rate</i>	

Returns

Rotors demanded speed

7.38.1.3 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

<i>throttle</i>	
<i>roll_rate</i>	
<i>pitch_rate</i>	
<i>yaw_rate</i>	

Returns

demanded surfaces deflection

7.39 src/controller/modes/controller_loop_FMANUAL.cpp File Reference

```
#include "controller_loop_FMANUAL.hpp"
```

Include dependency graph for controller_loop_FMANUAL.cpp:

7.40 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](#)

7.41 src/controller/modes/controller_loop_NONE.cpp File Reference

```
#include "controller_loop_NONE.hpp"
```

Include dependency graph for controller_loop_NONE.cpp:

7.42 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](#)

7.43 src/controller/modes/controller_loop_QACRO.cpp File Reference

```
#include "controller_loop_QACRO.hpp"
```

Include dependency graph for controller_loop_QACRO.cpp:

7.44 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](#)

7.45 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:
```

7.46 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](#)

7.47 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:
```

7.48 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](#)

7.49 src/controller/modes/controller_loop_RMANUAL.cpp File Reference

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

7.50 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](#)

7.51 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 double` [def::STEP_TIME](#) = 0.003
Step time of controller. Step of [PID](#) and [EKF](#) calculations.
- `const int` [def::INFO_PERIOD](#) = 2
How often send demands in response to stick command.

7.51.1 Macro Definition Documentation

7.51.1.1 USE_QUATERIONS

```
#define USE_QUATERIONS 1
```

7.52 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 dependency graph for main.cpp:
```

Macros

- `#define` [LOGGER_MASK](#) 5

Functions

- void [parseArgs](#) (int argc, char **argv, [UAVparams](#) *params)
Parse CL arguments.
- int [main](#) (int argc, char **argv)

Variables

- std::string [log_path](#) = "logs/"

7.52.1 Macro Definition Documentation

7.52.1.1 [LOGGER_MASK](#)

```
#define LOGGER\_MASK 5
```

7.52.2 Function Documentation

7.52.2.1 [main\(\)](#)

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

7.52.2.2 [parseArgs\(\)](#)

```
void parseArgs (  
    int argc,  
    char ** argv,  
    UAVparams * params )
```

Parse CL arguments.

Parameters

<i>argc</i>	number of argument
<i>argv</i>	argument array
<i>params</i>	pointer to UAVparams instant that should be filled

7.52.3 Variable Documentation

7.52.3.1 log_path

```
std::string log_path = "logs/"
```

7.53 src/navigation/AHRS.cpp File Reference

```
#include "AHRS.hpp"
#include <Eigen/Dense>
#include <random>
#include "common.hpp"
Include dependency graph for AHRS.cpp:
```

7.54 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.

7.55 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)

7.55.1 Function Documentation

7.55.1.1 [calcRbn\(\)](#)

```
Eigen::Matrix3d calcRbn (
    Eigen::Vector3d ori )
```

7.55.1.2 [calcRnb\(\)](#)

```
Eigen::Matrix3d calcRnb (
    Eigen::Vector3d ori )
```

7.55.1.3 [calcTom\(\)](#)

```
Eigen::Matrix3d calcTom (
    Eigen::Vector3d ori )
```

7.55.1.4 [clampOrientation\(\)](#)

```
void clampOrientation (
    Eigen::Vector3d & vec )
```

7.56 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.

7.57 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`)

7.57.1 Function Documentation

7.57.1.1 C()

```
Eigen::Matrix<double,6,7> C (
    Eigen::Vector4d q )
```

7.57.1.2 S()

```
Eigen::Matrix<double,4,3> S (
    Eigen::Vector4d q )
```

7.58 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.

7.59 src/navigation/EKF.cpp File Reference

```
#include "EKF.hpp"
#include <Eigen/Dense>
#include <iostream>
#include "common.hpp"
Include dependency graph for EKF.cpp:
```

7.60 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 [EKFPParams](#)

EK filter parameters.

- class [EKF](#)

Extended Kalman Filter.

7.61 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)

7.61.1 Function Documentation

7.61.1.1 connectConflateSocket()

```
void connectConflateSocket (
    zmq::socket_t & sock,
    std::string address,
    std::string topic )
```

7.61.1.2 r_nb() [1/2]

```
Eigen::Matrix<double, 3, 3> r_nb (
    const Eigen::Vector3d & RPY )
```

7.61.1.3 r_nb() [2/2]

```
Eigen::Matrix<double, 3, 3> r_nb (
    const Eigen::Vector4d & e )
```

7.61.1.4 recvVectors()

```
template<int Size1, int Size2>
bool recvVectors (
    zmq::socket_t & sock,
    int skip,
    Eigen::Vector< double, Size1 > & vec1,
    Eigen::Vector< double, Size2 > & vec2 )
```

7.62 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](#)

7.63 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 dependency graph for NS.cpp:

7.64 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.

7.65 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:
```

7.66 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.

7.67 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>

7.67.1 Function Documentation

7.67.1.1 [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

<i>demanded</i>	demanded angle in radian
<i>val</i>	actual angle in radian

Returns

angle error

7.67.1.2 [clampAngle\(\)](#)

```
double clampAngle (
    double angle ) [inline]
```

Clamps angle given in radians to range <-pi,pi>

Parameters

<i>angle</i>	angle in radian
--------------	-----------------

Returns

angle converted to range $<-\pi, \pi>$

7.67.1.3 safeGet()

```
template<typename T >
T safeGet (
    T & vec,
    std::mutex & mtx ) [inline]
```

Safe getter for T type value protected by mutex.

Template Parameters

<i>T</i>	Type of variable
----------	------------------

Parameters

<i>vec</i>	value to be get
<i>mtx</i>	mutex

Returns

value of *vec*

7.67.1.4 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

<i>T</i>	Type of variable
----------	------------------

Parameters

<i>vec</i>	value to be set
<i>new_val</i>	new value
<i>mtx</i>	mutex

Index

- [_V0](#)
 - [Ammo](#), [24](#)
- [_mode](#)
 - [ControllerLoop](#), [39](#)
- [~AHRS](#)
 - [AHRS](#), [15](#)
- [~AHRS_EKF](#)
 - [AHRS_EKF](#), [19](#)
- [~AHRS_complementary](#)
 - [AHRS_complementary](#), [17](#)
- [~Control](#)
 - [Control](#), [28](#)
- [~Controller](#)
 - [Controller](#), [33](#)
- [~ControllerLoop](#)
 - [ControllerLoop](#), [35](#)
- [~Environment](#)
 - [Environment](#), [57](#)
- [~Logger](#)
 - [Logger](#), [71](#)
- [~NS](#)
 - [NS](#), [74](#)
- [~ODE](#)
 - [ODE](#), [77](#)
- [~PID](#)
 - [PID](#), [85](#)
- [~UAVparams](#)
 - [UAVparams](#), [97](#)

[Accelerometer](#), [11](#)

- [Accelerometer](#), [11](#)
- [g](#), [12](#)
- [update](#), [12](#)

[aero_coffs](#)

- [UAVparams](#), [98](#)

[AeroCoefficients](#), [12](#)

- [C0](#), [13](#)
- [Cab](#), [13](#)
- [Cpqr](#), [13](#)
- [d](#), [13](#)
- [eAR](#), [13](#)
- [S](#), [13](#)
- [stallLimit](#), [13](#)

[AHRS](#), [14](#)

- [~AHRS](#), [15](#)
- [AHRS](#), [14](#)
- [env](#), [16](#)
- [getGyroBias](#), [15](#)
- [getOri](#), [15](#)
- [logger](#), [16](#)
- [mtxOri](#), [16](#)
- [ori_est](#), [16](#)
- [rot_bw](#), [15](#)
- [update](#), [16](#)

[ahrs](#)

- [UAVparams](#), [98](#)

[AHRS_complementary](#), [17](#)

- [~AHRS_complementary](#), [17](#)
- [AHRS_complementary](#), [17](#)
- [alpha](#), [18](#)
- [rot_bw](#), [18](#)
- [update](#), [18](#)

[AHRS_complementary.cpp](#)

- [calcRbn](#), [128](#)
- [calcRnb](#), [128](#)
- [calcTom](#), [128](#)
- [clampOrientation](#), [128](#)

[AHRS_EKF](#), [19](#)

- [~AHRS_EKF](#), [19](#)
- [AHRS_EKF](#), [19](#)
- [getGyroBias](#), [20](#)
- [P](#), [21](#)
- [Q](#), [21](#)
- [q](#), [20](#)
- [quaternionToRPY](#), [20](#)
- [R](#), [21](#)
- [rot_bw](#), [20](#)
- [RPYToQuaternion](#), [20](#)
- [update](#), [21](#)
- [x](#), [21](#)

[AHRS_EKF.cpp](#)

- [C](#), [129](#)
- [S](#), [129](#)

[AHRSParams](#), [22](#)

- [alpha](#), [22](#)
- [Q](#), [22](#)
- [R](#), [22](#)
- [type](#), [22](#)

[alpha](#)

- [AHRS_complementary](#), [18](#)
- [AHRSParams](#), [22](#)

[Ammo](#), [23](#)

- [_V0](#), [24](#)
- [Ammo](#), [23](#)
- [getV0](#), [23](#)
- [operator=](#), [24](#)

[ammo](#)

- [UAVparams](#), [98](#)

[AntiWindUpMode](#)

- PID.hpp, 113
- applyMixerRotors
 - mixers.cpp, 119
 - mixers.hpp, 121
- applyMixerRotorsHover
 - mixers.cpp, 120
 - mixers.hpp, 121
- applyMixerSurfaces
 - mixers.cpp, 120
 - mixers.hpp, 122
- asSkewMatrix
 - hinge.cpp, 105
- axis
 - Drive, 50
- Barometer, 24
 - Barometer, 25
 - update, 25
- baroScaler
 - EKFScalers, 55
- bias
 - Sensor< T >, 91
 - SensorParams, 92
- C
 - AHRS_EKF.cpp, 129
- C0
 - AeroCoefficients, 13
- Cab
 - AeroCoefficients, 13
- calc
 - PID, 85
- calcRbn
 - AHRS_complementary.cpp, 128
- calcRnb
 - AHRS_complementary.cpp, 128
- calcTom
 - AHRS_complementary.cpp, 128
- Cargo, 25
 - Cargo, 26
- cargo
 - UAVparams, 98
- checkJoystickLength
 - ControllerLoop, 35
- circularError
 - utils.hpp, 134
- clampAngle
 - utils.hpp, 134
- Clamping
 - PID.hpp, 114
- clampOrientation
 - AHRS_complementary.cpp, 128
- clear
 - PID, 86
- connectConflateSocket
 - environment.cpp, 131
- Control, 26
 - ~Control, 28
 - Control, 27
- Controller, 34
 - handleMsg, 28
 - prepare, 28
 - recv, 28
 - sendHinge, 28
 - sendSpeed, 30
 - sendSurface, 30
 - start, 30
 - startJet, 30
 - stop, 31
- control.cpp
 - orderServerJob, 115
- Controller, 31
 - ~Controller, 33
 - Control, 34
 - Controller, 32
 - exitController, 33
 - run, 33
 - setMode, 33
- controller_mode.hpp
 - ControllerMode, 118
 - ControllerModeFromString, 118
 - ControllerModeToString, 118
 - FACRO, 118
 - FANGLE, 118
 - FMANUAL, 118
 - NONE, 118
 - QACRO, 118
 - QANGLE, 118
 - QPOS, 118
 - RMANUAL, 118
- ControllerLoop, 34
 - _mode, 39
 - ~ControllerLoop, 35
 - checkJoystickLength, 35
 - ControllerLoop, 35
 - ControllerLoopFactory, 37
 - demandInfo, 37
 - getMode, 37
 - handleJoystick, 37
 - job, 38
 - overridePosition, 38
 - required_pids, 39
 - requiredPIDs, 38
- ControllerLoopFactory
 - ControllerLoop, 37
- ControllerLoopFMANUAL, 39
 - ControllerLoopFMANUAL, 39
 - handleJoystick, 40
 - job, 40
- ControllerLoopNONE, 40
 - ControllerLoopNONE, 40
- ControllerLoopQACRO, 41
 - ControllerLoopQACRO, 41
 - demandInfo, 41
 - handleJoystick, 42
 - job, 42
- ControllerLoopQANGLE, 42

- ControllerLoopQANGLE, 43
 - demandInfo, 43
 - handleJoystick, 43
 - job, 43
 - overridePosition, 43
- ControllerLoopQPOS, 44
 - ControllerLoopQPOS, 44
 - demandInfo, 44
 - handleJoystick, 45
 - job, 45
 - overridePosition, 45
- ControllerLoopRMANUAL, 45
 - ControllerLoopRMANUAL, 46
 - demandX, 47
 - demandY, 47
 - demandInfo, 46
 - handleJoystick, 46
 - job, 46
- ControllerMode
 - controller_mode.hpp, 118
- ControllerModeFromString
 - controller_mode.hpp, 118
- ControllerModeToString
 - controller_mode.hpp, 118
- ControlSurfaces, 47
 - ControlSurfaces, 48
 - getCoefficients, 48
 - getNoOfSurface, 48
 - getValues, 49
 - restoreTrim, 49
 - setValues, 49
- Cpqr
 - AeroCoefficients, 13
- d
 - AeroCoefficients, 13
- def, 9
 - INFO_PERIOD, 9
 - STEP_TIME, 9
- defines.hpp
 - USE_QUATERIONS, 125
- demandX
 - ControllerLoopRMANUAL, 47
- demandY
 - ControllerLoopRMANUAL, 47
- demandInfo
 - ControllerLoop, 37
 - ControllerLoopQACRO, 41
 - ControllerLoopQANGLE, 43
 - ControllerLoopQPOS, 44
 - ControllerLoopRMANUAL, 46
- direction
 - Rotor, 87
- dist
 - Sensor< T >, 91
- Drive, 49
 - axis, 50
 - hinges, 50
 - noOfHinges, 50
 - position, 50
- eAR
 - AeroCoefficients, 13
- EKF, 50
 - EKF, 51
 - getPos, 51
 - getVel, 52
 - log, 52
 - predict, 52
 - updateBaro, 52
 - updateGPS, 53
 - updateGPSVel, 53
- ekf
 - UAVparams, 98
- EKFParams, 53
 - P0, 54
 - Q, 54
 - RBaro, 54
 - RGPSPos, 54
 - RGPSVel, 54
- EKFScalers, 55
 - baroScaler, 55
 - predictScaler, 55
 - updateScaler, 55
 - zScaler, 55
- env
 - AHRS, 16
 - Sensor< T >, 91
- Environment, 56
 - ~Environment, 57
 - Environment, 56
 - getAngularAcceleraton, 57
 - getAngularVelocity, 57
 - getLinearAcceleration, 57
 - getLinearVelocity, 58
 - getOrientation, 58
 - getPosition, 58
 - getRnb, 58
 - getTime, 59
 - getWorldAngularVelocity, 59
 - getWorldLinearVelocity, 59
 - sensors, 60
 - sensorsVec3d, 60
 - updateSensors, 59
- environment.cpp
 - connectConflateSocket, 131
 - r_nb, 131
 - recvVectors, 131
- error
 - Sensor< T >, 89
- Euler
 - ODE, 77
- exitController
 - Controller, 33
- exiting
 - status.hpp, 114
- FACRO

- controller_mode.hpp, 118
- factory
 - ODE, 77
- FANGLE
 - controller_mode.hpp, 118
- FMANUAL
 - controller_mode.hpp, 118
- forceCoff
 - Rotor, 87
- fromString
 - ODE, 78
- g
 - Accelerometer, 12
- gen
 - Sensor< T >, 91
- getAngularAcceleraton
 - Environment, 57
- getAngularVelocity
 - Environment, 57
 - NS, 75
- getCoefficients
 - ControlSurfaces, 48
- getGyroBias
 - AHRS, 15
 - AHRS_EKF, 20
- getLastThrust
 - Jet, 66
- getLinearAcceleration
 - Environment, 57
- getLinearVelocity
 - Environment, 58
 - NS, 75
- getMass
 - Load, 69
- getMethodsToTest
 - ode_test.cpp, 108
- getMicrosteps
 - ODE, 78
- getMode
 - ControllerLoop, 37
- getNoOfSurface
 - ControlSurfaces, 48
- getOffset
 - Load, 69
- getOri
 - AHRS, 15
- getOrientation
 - Environment, 58
 - NS, 75
- getPos
 - EKF, 51
- getPosition
 - Environment, 58
 - NS, 75
- getReading
 - Sensor< T >, 89
- getRnb
 - Environment, 58
- getRot
 - Hinge, 65
- getRotorHoverSpeeds
 - UAVparams, 97
- getRotorMaxSpeeds
 - UAVparams, 97
- getRotorTimeContants
 - UAVparams, 97
- getSd
 - Sensor< T >, 90
- getSingleton
 - UAVparams, 97
- getThrust
 - Jet, 66
- getTime
 - Environment, 59
- getV0
 - Ammo, 23
- getValues
 - ControlSurfaces, 49
- getVel
 - EKF, 52
- getWorldAngularVelocity
 - Environment, 59
- getWorldLinearVelocity
 - Environment, 59
- go
 - TimedLoop, 95
- GPS, 60
 - GPS, 61
 - update, 61
- GPSVel, 61
 - GPSVel, 62
 - update, 62
- Gyroscope, 63
 - Gyroscope, 63
 - update, 63
- handleJoystick
 - ControllerLoop, 37
 - ControllerLoopFMANUAL, 40
 - ControllerLoopQACRO, 42
 - ControllerLoopQANGLE, 43
 - ControllerLoopQPOS, 45
 - ControllerLoopRMANUAL, 46
- handleMsg
 - Control, 28
- Heun
 - ODE, 77
- Hinge, 64
 - getRot, 65
 - Hinge, 64
 - operator=, 65
 - updateValue, 65
- hinge.cpp
 - asSkewMatrix, 105
- hinges
 - Drive, 50
- hoverSpeed

- Rotor, [87](#)
- idle
 - status.hpp, [114](#)
- INFO_PERIOD
 - def, [9](#)
- initialMode
 - UAVparams, [98](#)
- initialOrientation
 - UAVparams, [98](#)
- initialPosition
 - UAVparams, [99](#)
- initialVelocity
 - UAVparams, [99](#)
- INstantiate_TEST_SUITE_P
 - ode_test.cpp, [108](#)
- instantRun
 - UAVparams, [99](#)
- isReady
 - Sensor< T >, [90](#)
- lx
 - UAVparams, [99](#)
- lxy
 - UAVparams, [99](#)
- lxz
 - UAVparams, [99](#)
- ly
 - UAVparams, [99](#)
- lyz
 - UAVparams, [99](#)
- lz
 - UAVparams, [100](#)
- Jet, [65](#)
 - getLastThrust, [66](#)
 - getThrust, [66](#)
 - phases, [67](#)
 - start, [67](#)
 - thrust, [67](#)
 - time, [67](#)
- jets
 - UAVparams, [100](#)
- job
 - ControllerLoop, [38](#)
 - ControllerLoopFMANUAL, [40](#)
 - ControllerLoopQACRO, [42](#)
 - ControllerLoopQANGLE, [43](#)
 - ControllerLoopQPOS, [45](#)
 - ControllerLoopRMANUAL, [46](#)
- lastUpdate
 - Sensor< T >, [91](#)
- lib/UAV_common/header/common.hpp, [103](#)
- lib/UAV_common/src/components/aero_coefficients.hpp, [103](#)
- lib/UAV_common/src/components/components.hpp, [103](#)
- lib/UAV_common/src/components/control_surfaces.cpp, [104](#)
- lib/UAV_common/src/components/control_surfaces.hpp, [104](#)
- lib/UAV_common/src/components/drive.cpp, [104](#)
- lib/UAV_common/src/components/drive.hpp, [104](#)
- lib/UAV_common/src/components/hinge.cpp, [104](#)
- lib/UAV_common/src/components/hinge.hpp, [105](#)
- lib/UAV_common/src/components/loads.cpp, [105](#)
- lib/UAV_common/src/components/loads.hpp, [105](#)
- lib/UAV_common/src/components/navi.hpp, [106](#)
- lib/UAV_common/src/logger/logger.cpp, [106](#)
- lib/UAV_common/src/logger/logger.hpp, [106](#)
- lib/UAV_common/src/ode/ode.cpp, [107](#)
- lib/UAV_common/src/ode/ode.hpp, [107](#)
- lib/UAV_common/src/ode/ode_impl.hpp, [107](#)
- lib/UAV_common/src/ode/ode_test.cpp, [108](#)
- lib/UAV_common/src/parser/parser.cpp, [110](#)
- lib/UAV_common/src/parser/parser.hpp, [111](#)
- lib/UAV_common/src/parser/uav_params.cpp, [112](#)
- lib/UAV_common/src/parser/uav_params.hpp, [113](#)
- lib/UAV_common/src/PID/PID.cpp, [113](#)
- lib/UAV_common/src/PID/PID.hpp, [113](#)
- lib/UAV_common/src/timed_loop/status.hpp, [114](#)
- lib/UAV_common/src/timed_loop/timed_loop.cpp, [114](#)
- lib/UAV_common/src/timed_loop/timed_loop.hpp, [115](#)
- Load, [68](#)
 - getMass, [69](#)
 - getOffset, [69](#)
 - Load, [68](#)
 - operator=, [69](#)
 - release, [69](#)
- loadConfig
 - UAVparams, [97](#)
- log
 - EKF, [52](#)
 - Logger, [71](#)
- log_path
 - main.cpp, [127](#)
- Logger, [70](#)
 - ~Logger, [71](#)
 - log, [71](#)
 - Logger, [70](#)
 - setFmt, [72](#)
 - setLogDirectory, [72](#)
- logger
 - AHRS, [16](#)
 - Sensor< T >, [91](#)
- logger.cpp
 - shouldLog, [106](#)
- logger.hpp
 - LOGGER_MASK, [107](#)
- LOGGER_MASK
 - logger.hpp, [107](#)
 - main.cpp, [126](#)
- m
 - UAVparams, [100](#)
- mag
 - Magnetometer, [73](#)
- Magnetometer, [72](#)

- mag, 73
- Magnetometer, 73
- update, 73
- main
 - main.cpp, 126
 - ode_test.cpp, 108
- main.cpp
 - log_path, 127
 - LOGGER_MASK, 126
 - main, 126
 - parseArgs, 126
- maxSpeed
 - Rotor, 87
- mixers.cpp
 - applyMixerRotors, 119
 - applyMixerRotorsHover, 120
 - applyMixerSurfaces, 120
- mixers.hpp
 - applyMixerRotors, 121
 - applyMixerRotorsHover, 121
 - applyMixerSurfaces, 122
- mtxOri
 - AHRS, 16
- name
 - SensorParams, 93
 - UAVparams, 100
- NONE
 - controller_mode.hpp, 118
 - ODE, 77
- None
 - PID.hpp, 114
- noOfAmmo
 - UAVparams, 100
- noOfCargo
 - UAVparams, 100
- noOfHinges
 - Drive, 50
- noOfJets
 - UAVparams, 100
- noOfRotors
 - UAVparams, 100
- NS, 74
 - ~NS, 74
 - getAngularVelocity, 75
 - getLinearVelocity, 75
 - getOrientation, 75
 - getPosition, 75
 - NS, 74
- ODE, 76
 - ~ODE, 77
 - Euler, 77
 - factory, 77
 - fromString, 78
 - getMicrosteps, 78
 - Heun, 77
 - NONE, 77
 - ODE, 77
 - ODEMethod, 77
 - RK4, 77
 - step, 79
- ODE_Euler, 79
 - ODE_Euler, 80
 - step, 80
- ODE_Heun, 81
 - ODE_Heun, 81
 - step, 81
- ODE_RK4, 82
 - ODE_RK4, 82
 - step, 83
- ode_test.cpp
 - getMethodsToTest, 108
 - INstantiate_Test_Suite_P, 108
 - main, 108
 - TEST_F, 109
 - TEST_P, 109
- ODEMethod
 - ODE, 77
- ODETest, 83
 - SetUp, 84
 - TearDown, 84
- operator=
 - Ammo, 24
 - Hinge, 65
 - Load, 69
- orderServerJob
 - control.cpp, 115
- ori_est
 - AHRS, 16
- overridePosition
 - ControllerLoop, 38
 - ControllerLoopQANGLE, 43
 - ControllerLoopQPOS, 45
- P
 - AHRS_EKF, 21
- P0
 - EKFParams, 54
- parseArgs
 - main.cpp, 126
- parseHinge
 - uav_params.cpp, 112
- parseMatrixXd
 - parser.cpp, 110
 - parser.hpp, 111
- parsePID
 - uav_params.cpp, 112
- parser.cpp
 - parseMatrixXd, 110
 - parseVectorXd, 110
- parser.hpp
 - parseMatrixXd, 111
 - parseVectorXd, 112
- parseVectorXd
 - parser.cpp, 110
 - parser.hpp, 112
- phases

- Jet, [67](#)
- PID, [84](#)
 - ~PID, [85](#)
 - calc, [85](#)
 - clear, [86](#)
 - PID, [84](#)
 - set_dt, [86](#)
- PID.hpp
 - AntiWindUpMode, [113](#)
 - Clamping, [114](#)
 - None, [114](#)
- pids
 - UAVparams, [101](#)
- position
 - Drive, [50](#)
- predict
 - EKF, [52](#)
- predictScaler
 - EKFScalers, [55](#)
- prepare
 - Control, [28](#)
- Q
 - AHRS_EKF, [21](#)
 - AHRSParams, [22](#)
 - EKFParams, [54](#)
- q
 - AHRS_EKF, [20](#)
- QACRO
 - controller_mode.hpp, [118](#)
- QANGLE
 - controller_mode.hpp, [118](#)
- QPOS
 - controller_mode.hpp, [118](#)
- quaternionToRPY
 - AHRS_EKF, [20](#)
- R
 - AHRS_EKF, [21](#)
 - AHRSParams, [22](#)
- r_nb
 - environment.cpp, [131](#)
- RBaro
 - EKFParams, [54](#)
- ready
 - Sensor< T >, [91](#)
- recv
 - Control, [28](#)
- recvVectors
 - environment.cpp, [131](#)
- refreshTime
 - Sensor< T >, [92](#)
 - SensorParams, [93](#)
- release
 - Load, [69](#)
- reload
 - status.hpp, [114](#)
- required_pids
 - ControllerLoop, [39](#)
- requiredPIDs
 - ControllerLoop, [38](#)
- restoreTrim
 - ControlSurfaces, [49](#)
- RGPSPos
 - EKFParams, [54](#)
- RGPSVel
 - EKFParams, [54](#)
- RK4
 - ODE, [77](#)
- RMANUAL
 - controller_mode.hpp, [118](#)
- rot_bw
 - AHRS, [15](#)
 - AHRS_complementary, [18](#)
 - AHRS_EKF, [20](#)
- Rotor, [86](#)
 - direction, [87](#)
 - forceCoff, [87](#)
 - hoverSpeed, [87](#)
 - maxSpeed, [87](#)
 - timeConstant, [87](#)
 - torqueCoff, [87](#)
- rotorMixer
 - UAVparams, [101](#)
- rotors
 - UAVparams, [101](#)
- RPYToQuaternion
 - AHRS_EKF, [20](#)
- run
 - Controller, [33](#)
- running
 - status.hpp, [114](#)
- S
 - AeroCoefficients, [13](#)
 - AHRS_EKF.cpp, [129](#)
- safeGet
 - utils.hpp, [135](#)
- safeSet
 - utils.hpp, [135](#)
- sd
 - SensorParams, [93](#)
- sendHinge
 - Control, [28](#)
- sendSpeed
 - Control, [30](#)
- sendSurface
 - Control, [30](#)
- Sensor
 - Sensor< T >, [89](#)
- Sensor< T >, [88](#)
 - bias, [91](#)
 - dist, [91](#)
 - env, [91](#)
 - error, [89](#)
 - gen, [91](#)
 - getReading, [89](#)
 - getSd, [90](#)

- isReady, 90
- lastUpdate, 91
- logger, 91
- ready, 91
- refreshTime, 92
- Sensor, 89
- shouldUpdate, 90
- update, 90
- value, 92
- SensorParams, 92
 - bias, 92
 - name, 93
 - refreshTime, 93
 - sd, 93
- sensors
 - Environment, 60
 - UAVparams, 101
- sensorsVec3d
 - Environment, 60
- set_dt
 - PID, 86
- setFmt
 - Logger, 72
- setLogDirectory
 - Logger, 72
- setMode
 - Controller, 33
- SetUp
 - ODETest, 84
- setValues
 - ControlSurfaces, 49
- shouldLog
 - logger.cpp, 106
- shouldUpdate
 - Sensor< T >, 90
- src/communication/control.cpp, 115
- src/communication/control.hpp, 115
- src/communication/control_recv.cpp, 116
- src/communication/control_send.cpp, 116
- src/controller/controller.cpp, 116
- src/controller/controller.hpp, 116
- src/controller/controller_loop.cpp, 117
- src/controller/controller_loop.hpp, 117
- src/controller/controller_mode.hpp, 117
- src/controller/mixers.cpp, 119
- src/controller/mixers.hpp, 121
- src/controller/modes/controller_loop_FMANUAL.cpp, 122
- src/controller/modes/controller_loop_FMANUAL.hpp, 123
- src/controller/modes/controller_loop_NONE.cpp, 123
- src/controller/modes/controller_loop_NONE.hpp, 123
- src/controller/modes/controller_loop_QACRO.cpp, 123
- src/controller/modes/controller_loop_QACRO.hpp, 123
- src/controller/modes/controller_loop_QANGLE.cpp, 124
- src/controller/modes/controller_loop_QANGLE.hpp, 124
- src/controller/modes/controller_loop_QPOS.cpp, 124
- src/controller/modes/controller_loop_QPOS.hpp, 124
- src/controller/modes/controller_loop_RMANUAL.cpp, 124
- src/controller/modes/controller_loop_RMANUAL.hpp, 124
- src/defines.hpp, 125
- src/main.cpp, 125
- src/navigation/AHRS.cpp, 127
- src/navigation/AHRS.hpp, 127
- src/navigation/AHRS/AHRS_complementary.cpp, 127
- src/navigation/AHRS/AHRS_complementary.hpp, 128
- src/navigation/AHRS/AHRS_EKF.cpp, 129
- src/navigation/AHRS/AHRS_EKF.hpp, 129
- src/navigation/EKF.cpp, 130
- src/navigation/EKF.hpp, 130
- src/navigation/environment.cpp, 130
- src/navigation/environment.hpp, 132
- src/navigation/NS.cpp, 132
- src/navigation/NS.hpp, 132
- src/navigation/sensors.cpp, 133
- src/navigation/sensors.hpp, 133
- src/utils.hpp, 133
- stallLimit
 - AeroCoefficients, 13
- start
 - Control, 30
 - Jet, 67
- startJet
 - Control, 30
- Status
 - status.hpp, 114
- status.hpp
 - exiting, 114
 - idle, 114
 - reload, 114
 - running, 114
 - Status, 114
- step
 - ODE, 79
 - ODE_Euler, 80
 - ODE_Heun, 81
 - ODE_RK4, 83
- STEP_TIME
 - def, 9
- stop
 - Control, 31
- surfaceMixer
 - UAVparams, 101
- surfaces
 - UAVparams, 101
- TearDown
 - ODETest, 84
- TEST_F
 - ode_test.cpp, 109
- TEST_P
 - ode_test.cpp, 109
- thrust
 - Jet, 67
- time

- Jet, [67](#)
- timeConstant
 - Rotor, [87](#)
- TimedLoop, [93](#)
 - go, [95](#)
 - TimedLoop, [94](#)
- torqueCoff
 - Rotor, [87](#)
- type
 - AHRSPParams, [22](#)
- uav_params.cpp
 - parseHinge, [112](#)
 - parsePID, [112](#)
- UAVparams, [95](#)
 - ~UAVparams, [97](#)
 - aero_coffs, [98](#)
 - ahrs, [98](#)
 - ammo, [98](#)
 - cargo, [98](#)
 - ekf, [98](#)
 - getRotorHoverSpeeds, [97](#)
 - getRotorMaxSpeeds, [97](#)
 - getRotorTimeContants, [97](#)
 - getSingleton, [97](#)
 - initialMode, [98](#)
 - initialOrientation, [98](#)
 - initialPosition, [99](#)
 - initialVelocity, [99](#)
 - instantRun, [99](#)
 - lx, [99](#)
 - lxy, [99](#)
 - lxz, [99](#)
 - ly, [99](#)
 - lyz, [99](#)
 - lz, [100](#)
 - jets, [100](#)
 - loadConfig, [97](#)
 - m, [100](#)
 - name, [100](#)
 - noOfAmmo, [100](#)
 - noOfCargo, [100](#)
 - noOfJets, [100](#)
 - noOfRotors, [100](#)
 - pids, [101](#)
 - rotorMixer, [101](#)
 - rotors, [101](#)
 - sensors, [101](#)
 - surfaceMixer, [101](#)
 - surfaces, [101](#)
 - UAVparams, [97](#)
- update
 - Accelerometer, [12](#)
 - AHRS, [16](#)
 - AHRS_complementary, [18](#)
 - AHRS_EKF, [21](#)
 - Barometer, [25](#)
 - GPS, [61](#)
 - GPSVel, [62](#)
 - Gyroscope, [63](#)
 - Magnetometer, [73](#)
 - Sensor< T >, [90](#)
 - updateBaro
 - EKF, [52](#)
 - updateGPS
 - EKF, [53](#)
 - updateGPSVel
 - EKF, [53](#)
 - updateScaler
 - EKFScalers, [55](#)
 - updateSensors
 - Environment, [59](#)
 - updateValue
 - Hinge, [65](#)
 - USE_QUATERIONS
 - defines.hpp, [125](#)
 - utils.hpp
 - circularError, [134](#)
 - clampAngle, [134](#)
 - safeGet, [135](#)
 - safeSet, [135](#)
 - value
 - Sensor< T >, [92](#)
 - x
 - AHRS_EKF, [21](#)
 - zScaler
 - EKFScalers, [55](#)