

UAV drop physic

Generated by Doxygen 1.9.1



<b>1 Namespace Index</b>	<b>1</b>
1.1 Namespace List	1
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 File Index</b>	<b>9</b>
4.1 File List	9
<b>5 Namespace Documentation</b>	<b>13</b>
5.1 controllers Namespace Reference	13
5.2 def Namespace Reference	13
5.2.1 Detailed Description	13
5.2.2 Variable Documentation	13
5.2.2.1 INFO_PERIOD	13
<b>6 Class Documentation</b>	<b>15</b>
6.1 Accelerometer Class Reference	15
6.1.1 Detailed Description	15
6.1.2 Constructor & Destructor Documentation	15
6.1.2.1 Accelerometer()	16
6.1.3 Member Function Documentation	16
6.1.3.1 update()	16
6.1.4 Member Data Documentation	16
6.1.4.1 g	16
6.2 AeroCoefficients Struct Reference	16
6.2.1 Detailed Description	17
6.2.2 Member Data Documentation	17
6.2.2.1 C0	17
6.2.2.2 Cab	17
6.2.2.3 Cpqr	17
6.2.2.4 d	17
6.2.2.5 eAR	17
6.2.2.6 S	17
6.2.2.7 stallLimit	18
6.3 AHRS Class Reference	18
6.3.1 Detailed Description	18
6.3.2 Constructor & Destructor Documentation	18
6.3.2.1 AHRS()	18
6.3.2.2 ~AHRS()	19
6.3.3 Member Function Documentation	19

6.3.3.1 getGyroBias()	19
6.3.3.2 getOri()	19
6.3.3.3 rot_bw()	20
6.3.3.4 update()	20
6.3.4 Member Data Documentation	20
6.3.4.1 env	20
6.3.4.2 logger	20
6.3.4.3 mtxOri	20
6.3.4.4 ori_est	21
6.4 AHRS_complementary Class Reference	21
6.4.1 Detailed Description	21
6.4.2 Constructor & Destructor Documentation	21
6.4.2.1 AHRS_complementary()	21
6.4.2.2 ~AHRS_complementary()	22
6.4.3 Member Function Documentation	22
6.4.3.1 rot_bw()	22
6.4.3.2 update()	22
6.4.4 Member Data Documentation	22
6.4.4.1 alpha	22
6.5 AHRS_EKF Class Reference	23
6.5.1 Detailed Description	23
6.5.2 Constructor & Destructor Documentation	23
6.5.2.1 AHRS_EKF()	23
6.5.2.2 ~AHRS_EKF()	24
6.5.3 Member Function Documentation	24
6.5.3.1 getGyroBias()	24
6.5.3.2 q()	24
6.5.3.3 quaterionToRPY()	24
6.5.3.4 rot_bw()	24
6.5.3.5 RPYToQuaterion()	25
6.5.3.6 update()	25
6.5.4 Member Data Documentation	25
6.5.4.1 P	25
6.5.4.2 Q	25
6.5.4.3 R	25
6.5.4.4 x	25
6.6 AHRSParms Struct Reference	26
6.6.1 Detailed Description	26
6.6.2 Member Data Documentation	26
6.6.2.1 alpha	26
6.6.2.2 Q	26
6.6.2.3 R	26

6.6.2.4 type	26
6.7 Ammo Class Reference	27
6.7.1 Constructor & Destructor Documentation	27
6.7.1.1 Ammo() [1/2]	27
6.7.1.2 Ammo() [2/2]	27
6.7.2 Member Function Documentation	27
6.7.2.1 getV0()	28
6.7.2.2 operator=()	28
6.7.3 Member Data Documentation	28
6.7.3.1 _V0	28
6.8 controllers::BangBang Class Reference	28
6.8.1 Constructor & Destructor Documentation	29
6.8.1.1 BangBang() [1/2]	29
6.8.1.2 BangBang() [2/2]	29
6.8.2 Member Function Documentation	29
6.8.2.1 calc()	29
6.8.2.2 clear()	30
6.8.2.3 clone()	30
6.9 Barometer Class Reference	30
6.9.1 Detailed Description	31
6.9.2 Constructor & Destructor Documentation	31
6.9.2.1 Barometer()	31
6.9.3 Member Function Documentation	31
6.9.3.1 update()	31
6.10 Cargo Class Reference	31
6.10.1 Constructor & Destructor Documentation	32
6.10.1.1 Cargo() [1/2]	32
6.10.1.2 Cargo() [2/2]	32
6.11 Control Class Reference	32
6.11.1 Detailed Description	33
6.11.2 Constructor & Destructor Documentation	33
6.11.2.1 Control()	33
6.11.2.2 ~Control()	34
6.11.3 Member Function Documentation	34
6.11.3.1 handleMsg()	34
6.11.3.2 prepare()	34
6.11.3.3 recv()	34
6.11.3.4 sendHinge()	35
6.11.3.5 sendSpeed()	36
6.11.3.6 sendSurface()	36
6.11.3.7 setMode()	36
6.11.3.8 start()	36

6.11.3.9 startJet()	37
6.11.3.10 stop()	37
6.12 Controller Class Reference	37
6.12.1 Constructor & Destructor Documentation	38
6.12.1.1 Controller()	38
6.12.1.2 ~Controller()	38
6.12.2 Member Function Documentation	38
6.12.2.1 calc() [1/2]	38
6.12.2.2 calc() [2/2]	39
6.12.2.3 clear()	39
6.12.2.4 clone()	39
6.12.2.5 ControllerFactory()	39
6.12.2.6 set_dt()	40
6.12.3 Member Data Documentation	40
6.12.3.1 _dt	40
6.13 ControllerLoop Class Reference	40
6.13.1 Detailed Description	41
6.13.2 Constructor & Destructor Documentation	41
6.13.2.1 ControllerLoop()	41
6.13.2.2 ~ControllerLoop()	42
6.13.3 Member Function Documentation	42
6.13.3.1 checkJoystickLength()	42
6.13.3.2 ControllerLoopFactory()	42
6.13.3.3 demandInfo()	43
6.13.3.4 getMode()	43
6.13.3.5 handleJoystick()	43
6.13.3.6 job()	43
6.13.3.7 overridePositionAndSpeed()	44
6.13.3.8 requiredcontrollers()	44
6.13.4 Member Data Documentation	44
6.13.4.1 _mode	45
6.13.4.2 required_controllers	45
6.14 ControllerLoopFACRO Class Reference	45
6.14.1 Constructor & Destructor Documentation	45
6.14.1.1 ControllerLoopFACRO()	45
6.14.2 Member Function Documentation	45
6.14.2.1 demandInfo()	46
6.14.2.2 handleJoystick()	46
6.14.2.3 job()	46
6.15 ControllerLoopFANGLE Class Reference	46
6.15.1 Constructor & Destructor Documentation	47
6.15.1.1 ControllerLoopFANGLE()	47

6.15.2 Member Function Documentation	47
6.15.2.1 demandInfo()	47
6.15.2.2 handleJoystick()	47
6.15.2.3 job()	47
6.15.2.4 overridePositionAndSpeed()	47
6.15.3 Member Data Documentation	48
6.15.3.1 angleLimit	48
6.16 ControllerLoopFMANUAL Class Reference	48
6.16.1 Constructor & Destructor Documentation	48
6.16.1.1 ControllerLoopFMANUAL()	48
6.16.2 Member Function Documentation	49
6.16.2.1 handleJoystick()	49
6.16.2.2 job()	49
6.17 ControllerLoopNONE Class Reference	49
6.17.1 Constructor & Destructor Documentation	49
6.17.1.1 ControllerLoopNONE()	50
6.18 ControllerLoopQACRO Class Reference	50
6.18.1 Constructor & Destructor Documentation	50
6.18.1.1 ControllerLoopQACRO()	50
6.18.2 Member Function Documentation	50
6.18.2.1 demandInfo()	51
6.18.2.2 handleJoystick()	51
6.18.2.3 job()	51
6.19 ControllerLoopQANGLE Class Reference	51
6.19.1 Constructor & Destructor Documentation	52
6.19.1.1 ControllerLoopQANGLE()	52
6.19.2 Member Function Documentation	52
6.19.2.1 demandInfo()	52
6.19.2.2 handleJoystick()	52
6.19.2.3 job()	52
6.19.2.4 overridePositionAndSpeed()	52
6.20 ControllerLoopQPOS Class Reference	53
6.20.1 Constructor & Destructor Documentation	53
6.20.1.1 ControllerLoopQPOS()	53
6.20.2 Member Function Documentation	53
6.20.2.1 demandInfo()	54
6.20.2.2 handleJoystick()	54
6.20.2.3 job()	54
6.20.2.4 overridePositionAndSpeed()	54
6.21 ControllerLoopRANGLE Class Reference	55
6.21.1 Constructor & Destructor Documentation	55
6.21.1.1 ControllerLoopRANGLE()	55

6.21.2 Member Function Documentation	55
6.21.2.1 demandInfo()	56
6.21.2.2 handleJoystick()	56
6.21.2.3 job()	56
6.21.2.4 overridePositionAndSpeed()	56
6.21.3 Member Data Documentation	57
6.21.3.1 angleLimit	57
6.21.3.2 demandedPsi	57
6.21.3.3 demandedTheta	57
6.22 ControllerLoopRAUTOLAUNCH Class Reference	57
6.22.1 Constructor & Destructor Documentation	57
6.22.1.1 ControllerLoopRAUTOLAUNCH()	58
6.22.2 Member Function Documentation	58
6.22.2.1 job()	58
6.23 ControllerLoopRGUIDED Class Reference	58
6.23.1 Constructor & Destructor Documentation	59
6.23.1.1 ControllerLoopRGUIDED()	59
6.23.2 Member Function Documentation	59
6.23.2.1 demandInfo()	59
6.23.2.2 job()	59
6.23.3 Member Data Documentation	59
6.23.3.1 detection_limit	59
6.23.3.2 target	60
6.24 ControllerLoopRMANUAL Class Reference	60
6.24.1 Constructor & Destructor Documentation	60
6.24.1.1 ControllerLoopRMANUAL()	60
6.24.2 Member Function Documentation	60
6.24.2.1 demandInfo()	61
6.24.2.2 handleJoystick()	61
6.24.2.3 job()	61
6.24.3 Member Data Documentation	61
6.24.3.1 demanded_H	61
6.24.3.2 demanded_V	61
6.25 ControllerTest Class Reference	62
6.25.1 Member Function Documentation	62
6.25.1.1 SetUp()	62
6.25.1.2 TearDown()	62
6.26 ControlSurfaces Class Reference	62
6.26.1 Detailed Description	63
6.26.2 Constructor & Destructor Documentation	63
6.26.2.1 ControlSurfaces() [1/2]	63
6.26.2.2 ControlSurfaces() [2/2]	63



6.26.3 Member Function Documentation	63
6.26.3.1 getCoefficients()	63
6.26.3.2 getNoOfSurface()	64
6.26.3.3 getValues()	64
6.26.3.4 restoreTrim()	64
6.26.3.5 setValues()	64
6.27 ControlSystem Class Reference	64
6.27.1 Detailed Description	65
6.27.2 Constructor & Destructor Documentation	65
6.27.2.1 ControlSystem()	65
6.27.2.2 ~ControlSystem()	65
6.27.3 Member Function Documentation	65
6.27.3.1 exitController()	65
6.27.3.2 run()	66
6.27.3.3 setMode()	66
6.27.4 Friends And Related Function Documentation	66
6.27.4.1 Control	66
6.28 controllers::DoubleSetpoint Class Reference	66
6.28.1 Constructor & Destructor Documentation	67
6.28.1.1 DoubleSetpoint() [1/2]	67
6.28.1.2 DoubleSetpoint() [2/2]	67
6.28.2 Member Function Documentation	68
6.28.2.1 calc()	68
6.28.2.2 clear()	68
6.28.2.3 clone()	68
6.29 Drive Struct Reference	69
6.29.1 Detailed Description	69
6.29.2 Member Data Documentation	69
6.29.2.1 axis	69
6.29.2.2 hinges	69
6.29.2.3 noOfHinges	69
6.29.2.4 position	70
6.30 EKF Class Reference	70
6.30.1 Detailed Description	70
6.30.2 Constructor & Destructor Documentation	70
6.30.2.1 EKF()	70
6.30.3 Member Function Documentation	71
6.30.3.1 getPos()	71
6.30.3.2 getVel()	71
6.30.3.3 log()	71
6.30.3.4 predict()	72
6.30.3.5 updateBaro()	72

6.30.3.6 updateGPS()	72
6.30.3.7 updateGPSVel()	72
6.31 EKFPParams Struct Reference	73
6.31.1 Detailed Description	73
6.31.2 Member Data Documentation	73
6.31.2.1 P0	73
6.31.2.2 Q	73
6.31.2.3 RBaro	74
6.31.2.4 RGPSPos	74
6.31.2.5 RGPSVel	74
6.32 EKFScalers Struct Reference	74
6.32.1 Detailed Description	74
6.32.2 Member Data Documentation	74
6.32.2.1 baroScaler	75
6.32.2.2 predictScaler	75
6.32.2.3 updateScaler	75
6.32.2.4 zScaler	75
6.33 Environment Class Reference	75
6.33.1 Constructor & Destructor Documentation	76
6.33.1.1 Environment()	76
6.33.1.2 ~Environment()	76
6.33.2 Member Function Documentation	76
6.33.2.1 getAngularAcceleraton()	77
6.33.2.2 getAngularVelocity()	77
6.33.2.3 getLinearAcceleration()	77
6.33.2.4 getLinearVelocity()	77
6.33.2.5 getOrientation()	78
6.33.2.6 getPosition()	78
6.33.2.7 getRnb()	78
6.33.2.8 getTime()	78
6.33.2.9 getWorldAngularVelocity()	79
6.33.2.10 getWorldLinearVelocity()	79
6.33.2.11 updateSensors()	79
6.33.3 Member Data Documentation	79
6.33.3.1 sensors	79
6.33.3.2 sensorsVec3d	80
6.34 GPS Class Reference	80
6.34.1 Detailed Description	80
6.34.2 Constructor & Destructor Documentation	80
6.34.2.1 GPS()	80
6.34.3 Member Function Documentation	81
6.34.3.1 update()	81

6.35 GPSVel Class Reference	81
6.35.1 Detailed Description	81
6.35.2 Constructor & Destructor Documentation	81
6.35.2.1 GPSVel()	82
6.35.3 Member Function Documentation	82
6.35.3.1 update()	82
6.36 Gyroscope Class Reference	82
6.36.1 Detailed Description	82
6.36.2 Constructor & Destructor Documentation	83
6.36.2.1 Gyroscope()	83
6.36.3 Member Function Documentation	83
6.36.3.1 update()	83
6.37 Hinge Class Reference	83
6.37.1 Detailed Description	84
6.37.2 Constructor & Destructor Documentation	84
6.37.2.1 Hinge() [1/3]	84
6.37.2.2 Hinge() [2/3]	84
6.37.2.3 Hinge() [3/3]	84
6.37.3 Member Function Documentation	84
6.37.3.1 getRot()	84
6.37.3.2 operator=()	85
6.37.3.3 updateValue()	85
6.38 Jet Class Reference	86
6.38.1 Detailed Description	86
6.38.2 Member Function Documentation	86
6.38.2.1 getLastThrust()	87
6.38.2.2 getThrust()	87
6.38.2.3 start()	87
6.38.3 Member Data Documentation	87
6.38.3.1 phases	88
6.38.3.2 thrust	88
6.38.3.3 time	88
6.39 Load Class Reference	88
6.39.1 Detailed Description	89
6.39.2 Constructor & Destructor Documentation	89
6.39.2.1 Load() [1/2]	89
6.39.2.2 Load() [2/2]	89
6.39.3 Member Function Documentation	89
6.39.3.1 getAmmount()	89
6.39.3.2 getMass()	89
6.39.3.3 getOffset()	90
6.39.3.4 operator=()	90

6.39.3.5 release()	90
6.40 Logger Class Reference	90
6.40.1 Detailed Description	91
6.40.2 Constructor & Destructor Documentation	91
6.40.2.1 Logger()	91
6.40.2.2 ~Logger()	91
6.40.3 Member Function Documentation	92
6.40.3.1 log() [1/2]	92
6.40.3.2 log() [2/2]	92
6.40.3.3 setFmt()	92
6.40.3.4 setLogDirectory()	93
6.41 Magnetometer Class Reference	93
6.41.1 Detailed Description	93
6.41.2 Constructor & Destructor Documentation	93
6.41.2.1 Magnetometer()	94
6.41.3 Member Function Documentation	94
6.41.3.1 update()	94
6.41.4 Member Data Documentation	94
6.41.4.1 mag	94
6.42 NS Class Reference	94
6.42.1 Detailed Description	95
6.42.2 Constructor & Destructor Documentation	95
6.42.2.1 NS()	95
6.42.2.2 ~NS()	95
6.42.3 Member Function Documentation	95
6.42.3.1 getAngularVelocity()	96
6.42.3.2 getLinearVelocity()	96
6.42.3.3 getOrientation()	96
6.42.3.4 getPosition()	96
6.42.3.5 getRotationMatrixBodyToWorld()	97
6.43 ODE Class Reference	97
6.43.1 Detailed Description	98
6.43.2 Member Enumeration Documentation	98
6.43.2.1 ODEMethod	98
6.43.3 Constructor & Destructor Documentation	98
6.43.3.1 ODE()	98
6.43.3.2 ~ODE()	98
6.43.4 Member Function Documentation	98
6.43.4.1 factory()	99
6.43.4.2 fromString()	100
6.43.4.3 getMicrosteps() [1/2]	100
6.43.4.4 getMicrosteps() [2/2]	100

6.43.4.5 step()	101
6.44 ODE_Euler Class Reference	101
6.44.1 Detailed Description	102
6.44.2 Constructor & Destructor Documentation	102
6.44.2.1 ODE_Euler()	102
6.44.3 Member Function Documentation	102
6.44.3.1 step()	102
6.45 ODE_Heun Class Reference	103
6.45.1 Detailed Description	103
6.45.2 Constructor & Destructor Documentation	103
6.45.2.1 ODE_Heun()	103
6.45.3 Member Function Documentation	103
6.45.3.1 step()	103
6.46 ODE_PC2 Class Reference	104
6.46.1 Detailed Description	104
6.46.2 Constructor & Destructor Documentation	104
6.46.2.1 ODE_PC2()	105
6.46.3 Member Function Documentation	105
6.46.3.1 step()	105
6.47 ODE_PC4 Class Reference	105
6.47.1 Detailed Description	106
6.47.2 Constructor & Destructor Documentation	106
6.47.2.1 ODE_PC4()	106
6.47.3 Member Function Documentation	106
6.47.3.1 step()	106
6.48 ODE_RK4 Class Reference	107
6.48.1 Detailed Description	107
6.48.2 Constructor & Destructor Documentation	107
6.48.2.1 ODE_RK4()	107
6.48.3 Member Function Documentation	107
6.48.3.1 step()	107
6.49 ODETest Class Reference	108
6.49.1 Member Function Documentation	108
6.49.1.1 SetUp()	108
6.49.1.2 TearDown()	108
6.50 Params Class Reference	109
6.50.1 Detailed Description	109
6.50.2 Constructor & Destructor Documentation	109
6.50.2.1 Params() [1/3]	109
6.50.2.2 Params() [2/3]	110
6.50.2.3 Params() [3/3]	110
6.50.2.4 ~Params()	110

6.50.3 Member Function Documentation	110
6.50.3.1 getSingleton()	110
6.50.3.2 operator=()	110
6.50.4 Member Data Documentation	110
6.50.4.1 STEP_TIME	111
6.51 controllers::PID Class Reference	111
6.51.1 Member Enumeration Documentation	111
6.51.1.1 AntiWindUpMode	111
6.51.2 Constructor & Destructor Documentation	112
6.51.2.1 PID() [1/2]	112
6.51.2.2 PID() [2/2]	112
6.51.3 Member Function Documentation	113
6.51.3.1 calc()	113
6.51.3.2 clear()	113
6.51.3.3 clone()	113
6.52 controllers::PID_Discrete Class Reference	114
6.52.1 Constructor & Destructor Documentation	114
6.52.1.1 PID_Discrete() [1/2]	114
6.52.1.2 PID_Discrete() [2/2]	115
6.52.2 Member Function Documentation	115
6.52.2.1 calc()	115
6.52.2.2 clear()	115
6.52.2.3 clone()	116
6.52.2.4 set_dt()	116
6.53 Rotor Struct Reference	116
6.53.1 Detailed Description	117
6.53.2 Member Data Documentation	117
6.53.2.1 direction	117
6.53.2.2 forceCoff	117
6.53.2.3 hoverSpeed	117
6.53.2.4 maxSpeed	117
6.53.2.5 timeConstant	117
6.53.2.6 torqueCoff	117
6.54 Sensor< T > Class Template Reference	118
6.54.1 Detailed Description	118
6.54.2 Constructor & Destructor Documentation	119
6.54.2.1 Sensor()	119
6.54.3 Member Function Documentation	119
6.54.3.1 error()	119
6.54.3.2 getReading()	119
6.54.3.3 getSd()	120
6.54.3.4 isReady()	120

6.54.3.5 shouldUpdate()	120
6.54.3.6 update()	120
6.54.4 Member Data Documentation	121
6.54.4.1 bias	121
6.54.4.2 dist	121
6.54.4.3 env	121
6.54.4.4 gen	121
6.54.4.5 lastUpdate	121
6.54.4.6 logger	121
6.54.4.7 ready	122
6.54.4.8 refreshTime	122
6.54.4.9 value	122
6.55 SensorParams Struct Reference	122
6.55.1 Detailed Description	122
6.55.2 Member Data Documentation	122
6.55.2.1 bias	123
6.55.2.2 name	123
6.55.2.3 refreshTime	123
6.55.2.4 sd	123
6.56 TimedLoop Class Reference	123
6.56.1 Detailed Description	123
6.56.2 Constructor & Destructor Documentation	124
6.56.2.1 TimedLoop()	124
6.56.3 Member Function Documentation	125
6.56.3.1 go() [1/2]	125
6.56.3.2 go() [2/2]	125
6.57 UAVparams Struct Reference	125
6.57.1 Detailed Description	126
6.57.2 Constructor & Destructor Documentation	127
6.57.2.1 UAVparams()	127
6.57.2.2 ~UAVparams()	127
6.57.3 Member Function Documentation	127
6.57.3.1 getRotorHoverSpeeds()	127
6.57.3.2 getRotorMaxSpeeds()	127
6.57.3.3 getRotorTimeContants()	127
6.57.3.4 getSingleton()	127
6.57.3.5 loadConfig()	128
6.57.4 Member Data Documentation	128
6.57.4.1 aero_coffs	128
6.57.4.2 ahrs	128
6.57.4.3 ammo	128
6.57.4.4 cargo	128

6.57.4.5 controllers	128
6.57.4.6 ekf	128
6.57.4.7 initialMode	129
6.57.4.8 initialOrientation	129
6.57.4.9 initialPosition	129
6.57.4.10 initialVelocity	129
6.57.4.11 instantRun	129
6.57.4.12 lx	129
6.57.4.13 lxy	129
6.57.4.14 lxz	129
6.57.4.15 ly	130
6.57.4.16 lyz	130
6.57.4.17 lz	130
6.57.4.18 jets	130
6.57.4.19 m	130
6.57.4.20 name	130
6.57.4.21 noOfAmmo	130
6.57.4.22 noOfCargo	130
6.57.4.23 noOfJets	131
6.57.4.24 noOfRotors	131
6.57.4.25 rotorMixer	131
6.57.4.26 rotors	131
6.57.4.27 sensors	131
6.57.4.28 surfaceMixer	131
6.57.4.29 surfaces	131
6.57.4.30 target	132
6.58 controllers::ZTransform Class Reference	132
6.58.1 Constructor & Destructor Documentation	132
6.58.1.1 ZTransform() [1/2]	132
6.58.1.2 ZTransform() [2/2]	133
6.58.2 Member Function Documentation	133
6.58.2.1 calc()	133
6.58.2.2 clear()	133
6.58.2.3 clone()	134
6.59 controllers::ZTransformStatic< N, D > Class Template Reference	134
6.59.1 Constructor & Destructor Documentation	134
6.59.1.1 ZTransformStatic() [1/2]	134
6.59.1.2 ZTransformStatic() [2/2]	135
6.59.2 Member Function Documentation	135
6.59.2.1 calc()	135
6.59.2.2 clear()	135
6.59.2.3 clone()	136



<b>7 File Documentation</b>	<b>137</b>
7.1 build/CMakeFiles/3.22.1/CompilerIdC/CMakeCCompilerId.c File Reference	137
7.1.1 Macro Definition Documentation	137
7.1.1.1 __has_include	138
7.1.1.2 ARCHITECTURE_ID	138
7.1.1.3 C_VERSION	138
7.1.1.4 COMPILER_ID	138
7.1.1.5 DEC	138
7.1.1.6 HEX	138
7.1.1.7 PLATFORM_ID	139
7.1.1.8 STRINGIFY	139
7.1.1.9 STRINGIFY_HELPER	139
7.1.2 Function Documentation	139
7.1.2.1 main()	139
7.1.3 Variable Documentation	139
7.1.3.1 info_arch	139
7.1.3.2 info_compiler	139
7.1.3.3 info_language_extensions_default	140
7.1.3.4 info_language_standard_default	140
7.1.3.5 info_platform	140
7.2 build/CMakeFiles/3.22.1/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference	140
7.2.1 Macro Definition Documentation	141
7.2.1.1 __has_include	141
7.2.1.2 ARCHITECTURE_ID	141
7.2.1.3 COMPILER_ID	141
7.2.1.4 CXX_STD	141
7.2.1.5 DEC	141
7.2.1.6 HEX	142
7.2.1.7 PLATFORM_ID	142
7.2.1.8 STRINGIFY	142
7.2.1.9 STRINGIFY_HELPER	142
7.2.2 Function Documentation	142
7.2.2.1 main()	142
7.2.3 Variable Documentation	142
7.2.3.1 info_arch	143
7.2.3.2 info_compiler	143
7.2.3.3 info_language_extensions_default	143
7.2.3.4 info_language_standard_default	143
7.2.3.5 info_platform	143
7.3 build/CMakeFiles/controller.dir/src/communication/control.cpp.o.d File Reference	145
7.4 build/CMakeFiles/controller.dir/src/communication/control_recv.cpp.o.d File Reference	145
7.5 build/CMakeFiles/controller.dir/src/communication/control_send.cpp.o.d File Reference	145

7.6 build/CMakeFiles/controller.dir/src/controller/controller.cpp.o.d File Reference . . . . .	145
7.7 build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/controller.cpp.o.d File Reference . . .	145
7.8 build/CMakeFiles/controller.dir/src/controller/controller_loop.cpp.o.d File Reference . . . . .	145
7.9 build/CMakeFiles/controller.dir/src/controller/mixers.cpp.o.d File Reference . . . . .	145
7.10 build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_FACRO.cpp.o.d File Reference .	145
7.11 build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_FANGLE.cpp.o.d File Reference	145
7.12 build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_FMANUAL.cpp.o.d File Reference	145
7.13 build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_NONE.cpp.o.d File Reference .	145
7.14 build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_QACRO.cpp.o.d File Reference	145
7.15 build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_QANGLE.cpp.o.d File Reference	145
7.16 build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_QPOS.cpp.o.d File Reference .	145
7.17 build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_RANGLE.cpp.o.d File Reference	145
7.18 build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_RAUTOLAUNCH.cpp.o.d File Reference . . . . .	145
7.19 build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_RGUIDED.cpp.o.d File Reference	145
7.20 build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_RMANUAL.cpp.o.d File Reference	145
7.21 build/CMakeFiles/controller.dir/src/main.cpp.o.d File Reference . . . . .	145
7.22 build/CMakeFiles/controller.dir/src/navigation/AHRS.cpp.o.d File Reference . . . . .	145
7.23 build/CMakeFiles/controller.dir/src/navigation/AHRS/AHRS_complementary.cpp.o.d File Reference .	145
7.24 build/CMakeFiles/controller.dir/src/navigation/AHRS/AHRS_EKF.cpp.o.d File Reference . . . . .	145
7.25 build/CMakeFiles/controller.dir/src/navigation/EKF.cpp.o.d File Reference . . . . .	145
7.26 build/CMakeFiles/controller.dir/src/navigation/environment.cpp.o.d File Reference . . . . .	145
7.27 build/CMakeFiles/controller.dir/src/navigation/NS.cpp.o.d File Reference . . . . .	145
7.28 build/CMakeFiles/controller.dir/src/navigation/sensors.cpp.o.d File Reference . . . . .	145
7.29 build/CMakeFiles/controller.dir/src/params.cpp.o.d File Reference . . . . .	145
7.30 build/lib/UAV_common/CMakeFiles/common.dir/src/components/control_surfaces.cpp.o.d File Refer- ence . . . . .	145
7.31 build/lib/UAV_common/CMakeFiles/common.dir/src/components/drive.cpp.o.d File Reference . . .	145
7.32 build/lib/UAV_common/CMakeFiles/common.dir/src/components/hinge.cpp.o.d File Reference . . .	145
7.33 build/lib/UAV_common/CMakeFiles/common.dir/src/components/loads.cpp.o.d File Reference . . .	145
7.34 build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/bang_bang.cpp.o.d File Reference	145
7.35 build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/double_setpoint.cpp.o.d File Reference . . . . .	145
7.36 build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/PID.cpp.o.d File Reference . . .	145
7.37 build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/PID_discrete.cpp.o.d File Refer- ence . . . . .	145
7.38 build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/z_trans.cpp.o.d File Reference .	145
7.39 build/lib/UAV_common/CMakeFiles/common.dir/src/logger/logger.cpp.o.d File Reference . . . . .	145
7.40 build/lib/UAV_common/CMakeFiles/common.dir/src/ode/ode.cpp.o.d File Reference . . . . .	145
7.41 build/lib/UAV_common/CMakeFiles/common.dir/src/parser/parser.cpp.o.d File Reference . . . . .	145
7.42 build/lib/UAV_common/CMakeFiles/common.dir/src/parser/uav_params.cpp.o.d File Reference . . .	145
7.43 build/lib/UAV_common/CMakeFiles/common.dir/src/timed_loop/timed_loop.cpp.o.d File Reference .	145

7.44 build/lib/UAV_common/CMakeFiles/Controller_test.dir/src/controllers/controller_test.cpp.o.d File Reference . . . . .	145
7.45 build/lib/UAV_common/CMakeFiles/ODE_test.dir/src/ode/ode_test.cpp.o.d File Reference . . . . .	145
7.46 lib/UAV_common/header/common.hpp File Reference . . . . .	145
7.47 lib/UAV_common/scripts/controller_plots.m File Reference . . . . .	146
7.47.1 Function Documentation . . . . .	146
7.47.1.1 legend() . . . . .	146
7.47.1.2 plot() . . . . .	147
7.47.1.3 title() . . . . .	147
7.47.1.4 xlabel() . . . . .	147
7.47.1.5 ylabel() . . . . .	147
7.47.2 Variable Documentation . . . . .	147
7.47.2.1 clc . . . . .	147
7.47.2.2 csvFiles . . . . .	147
7.47.2.3 data . . . . .	148
7.47.2.4 figure . . . . .	148
7.47.2.5 folderPath . . . . .	148
7.47.2.6 i . . . . .	148
7.47.2.7 off . . . . .	148
7.47.2.8 on . . . . .	148
7.47.2.9 x . . . . .	148
7.47.2.10 y . . . . .	149
7.48 lib/UAV_common/src/components/aero_coefficients.hpp File Reference . . . . .	149
7.49 lib/UAV_common/src/components/components.hpp File Reference . . . . .	149
7.50 lib/UAV_common/src/components/control_surfaces.cpp File Reference . . . . .	149
7.51 lib/UAV_common/src/components/control_surfaces.hpp File Reference . . . . .	149
7.52 lib/UAV_common/src/components/drive.cpp File Reference . . . . .	150
7.53 lib/UAV_common/src/components/drive.hpp File Reference . . . . .	150
7.54 lib/UAV_common/src/components/hinge.cpp File Reference . . . . .	150
7.54.1 Function Documentation . . . . .	150
7.54.1.1 asSkewMatrix() . . . . .	150
7.55 lib/UAV_common/src/components/hinge.hpp File Reference . . . . .	150
7.56 lib/UAV_common/src/components/loads.cpp File Reference . . . . .	151
7.57 lib/UAV_common/src/components/loads.hpp File Reference . . . . .	151
7.58 lib/UAV_common/src/components/navi.hpp File Reference . . . . .	151
7.59 lib/UAV_common/src/controllers/controller.cpp File Reference . . . . .	151
7.60 src/controller/controller.cpp File Reference . . . . .	152
7.61 lib/UAV_common/src/controllers/controller.hpp File Reference . . . . .	152
7.62 src/controller/controller.hpp File Reference . . . . .	152
7.63 lib/UAV_common/src/controllers/controller_test.cpp File Reference . . . . .	152
7.63.1 Function Documentation . . . . .	153
7.63.1.1 getMethodsToTest() . . . . .	153

7.63.1.2	INstantiate_TEST_SUITE_P()	153
7.63.1.3	main()	153
7.63.1.4	TEST_P() [1/2]	154
7.63.1.5	TEST_P() [2/2]	154
7.63.2	Variable Documentation	154
7.63.2.1	plot	154
7.63.2.2	plot_directory_name	154
7.64	lib/UAV_common/src/controllers/impl/bang_bang.cpp File Reference	154
7.65	lib/UAV_common/src/controllers/impl/bang_bang.hpp File Reference	154
7.66	lib/UAV_common/src/controllers/impl/double_setpoint.cpp File Reference	155
7.67	lib/UAV_common/src/controllers/impl/double_setpoint.hpp File Reference	155
7.68	lib/UAV_common/src/controllers/impl/PID.cpp File Reference	155
7.69	lib/UAV_common/src/controllers/impl/PID.hpp File Reference	156
7.70	lib/UAV_common/src/controllers/impl/PID_discrete.cpp File Reference	156
7.71	lib/UAV_common/src/controllers/impl/PID_discrete.hpp File Reference	156
7.72	lib/UAV_common/src/controllers/impl/z_trans.cpp File Reference	157
7.72.1	Function Documentation	157
7.72.1.1	splitStringToDoubleVector()	157
7.73	lib/UAV_common/src/controllers/impl/z_trans.hpp File Reference	157
7.74	lib/UAV_common/src/logger/logger.cpp File Reference	158
7.74.1	Function Documentation	158
7.74.1.1	shouldLog()	158
7.75	lib/UAV_common/src/logger/logger.hpp File Reference	158
7.75.1	Macro Definition Documentation	158
7.75.1.1	LOGGER_MASK	159
7.76	lib/UAV_common/src/ode/ode.cpp File Reference	159
7.77	lib/UAV_common/src/ode/ode.hpp File Reference	159
7.78	lib/UAV_common/src/ode/ode_impl.hpp File Reference	159
7.79	lib/UAV_common/src/ode/ode_test.cpp File Reference	159
7.79.1	Function Documentation	160
7.79.1.1	getMethodsToTest()	160
7.79.1.2	INstantiate_TEST_SUITE_P()	160
7.79.1.3	main()	160
7.79.1.4	TEST_F() [1/2]	160
7.79.1.5	TEST_F() [2/2]	161
7.79.1.6	TEST_P() [1/4]	161
7.79.1.7	TEST_P() [2/4]	161
7.79.1.8	TEST_P() [3/4]	161
7.79.1.9	TEST_P() [4/4]	161
7.80	lib/UAV_common/src/parser/parser.cpp File Reference	161
7.80.1	Function Documentation	162
7.80.1.1	parseMatrixXd()	162

7.80.1.2 parseVectorXd()	162
7.81 lib/UAV_common/src/parser/parser.hpp File Reference	163
7.81.1 Function Documentation	163
7.81.1.1 parseMatrixXd()	163
7.81.1.2 parseVectorXd()	163
7.82 lib/UAV_common/src/parser/uav_params.cpp File Reference	164
7.82.1 Function Documentation	164
7.82.1.1 parseHinge()	164
7.83 lib/UAV_common/src/parser/uav_params.hpp File Reference	164
7.84 lib/UAV_common/src/timed_loop/status.hpp File Reference	165
7.84.1 Enumeration Type Documentation	165
7.84.1.1 Status	165
7.85 lib/UAV_common/src/timed_loop/timed_loop.cpp File Reference	165
7.86 lib/UAV_common/src/timed_loop/timed_loop.hpp File Reference	165
7.87 src/communication/control.cpp File Reference	166
7.87.1 Function Documentation	166
7.87.1.1 orderServerJob()	166
7.88 src/communication/control.hpp File Reference	166
7.89 src/communication/control_recv.cpp File Reference	167
7.90 src/communication/control_send.cpp File Reference	167
7.91 src/controller/controller_loop.cpp File Reference	167
7.92 src/controller/controller_loop.hpp File Reference	167
7.93 src/controller/controller_mode.hpp File Reference	168
7.93.1 Enumeration Type Documentation	168
7.93.1.1 ControllerMode	168
7.93.2 Function Documentation	169
7.93.2.1 ControllerModeFromString()	169
7.93.2.2 ControllerModeToString()	169
7.94 src/controller/mixers.cpp File Reference	169
7.94.1 Function Documentation	170
7.94.1.1 applyMixerRotors()	170
7.94.1.2 applyMixerRotorsHover()	170
7.94.1.3 applyMixerSurfaces()	171
7.95 src/controller/mixers.hpp File Reference	171
7.95.1 Function Documentation	172
7.95.1.1 applyMixerRotors()	172
7.95.1.2 applyMixerRotorsHover()	172
7.95.1.3 applyMixerSurfaces()	173
7.96 src/controller/modes/controller_loop_FACRO.cpp File Reference	173
7.97 src/controller/modes/controller_loop_FACRO.hpp File Reference	173
7.98 src/controller/modes/controller_loop_FANGLE.cpp File Reference	173
7.99 src/controller/modes/controller_loop_FANGLE.hpp File Reference	174

7.100 src/controller/modes/controller_loop_FMANUAL.cpp File Reference . . . . .	174
7.101 src/controller/modes/controller_loop_FMANUAL.hpp File Reference . . . . .	174
7.102 src/controller/modes/controller_loop_NONE.cpp File Reference . . . . .	174
7.103 src/controller/modes/controller_loop_NONE.hpp File Reference . . . . .	174
7.104 src/controller/modes/controller_loop_QACRO.cpp File Reference . . . . .	175
7.105 src/controller/modes/controller_loop_QACRO.hpp File Reference . . . . .	175
7.106 src/controller/modes/controller_loop_QANGLE.cpp File Reference . . . . .	175
7.107 src/controller/modes/controller_loop_QANGLE.hpp File Reference . . . . .	175
7.108 src/controller/modes/controller_loop_QPOS.cpp File Reference . . . . .	175
7.109 src/controller/modes/controller_loop_QPOS.hpp File Reference . . . . .	175
7.110 src/controller/modes/controller_loop_RANGLE.cpp File Reference . . . . .	176
7.111 src/controller/modes/controller_loop_RANGLE.hpp File Reference . . . . .	176
7.112 src/controller/modes/controller_loop_RAUTOLAUNCH.cpp File Reference . . . . .	176
7.113 src/controller/modes/controller_loop_RAUTOLAUNCH.hpp File Reference . . . . .	176
7.114 src/controller/modes/controller_loop_RGUIDED.cpp File Reference . . . . .	176
7.115 src/controller/modes/controller_loop_RGUIDED.hpp File Reference . . . . .	177
7.116 src/controller/modes/controller_loop_RMANUAL.cpp File Reference . . . . .	177
7.117 src/controller/modes/controller_loop_RMANUAL.hpp File Reference . . . . .	177
7.118 src/defines.hpp File Reference . . . . .	177
7.118.1 Macro Definition Documentation . . . . .	178
7.118.1.1 USE_QUATERIONS . . . . .	178
7.119 src/main.cpp File Reference . . . . .	178
7.119.1 Macro Definition Documentation . . . . .	178
7.119.1.1 LOGGER_MASK . . . . .	179
7.119.2 Function Documentation . . . . .	179
7.119.2.1 main() . . . . .	179
7.119.2.2 parseArgs() . . . . .	179
7.119.3 Variable Documentation . . . . .	179
7.119.3.1 log_path . . . . .	179
7.120 src/navigation/AHRS.cpp File Reference . . . . .	180
7.121 src/navigation/AHRS.hpp File Reference . . . . .	180
7.122 src/navigation/AHRS/AHRS_complementary.cpp File Reference . . . . .	180
7.122.1 Function Documentation . . . . .	180
7.122.1.1 calcRbn() . . . . .	180
7.122.1.2 calcRnb() . . . . .	181
7.122.1.3 calcTom() . . . . .	181
7.122.1.4 clampOrientation() . . . . .	181
7.123 src/navigation/AHRS/AHRS_complementary.hpp File Reference . . . . .	181
7.124 src/navigation/AHRS/AHRS_EKF.cpp File Reference . . . . .	181
7.124.1 Function Documentation . . . . .	182
7.124.1.1 C() . . . . .	182
7.124.1.2 S() . . . . .	182

7.125 src/navigation/AHRS/AHRS_EKF.hpp File Reference . . . . .	182
7.126 src/navigation/EKF.cpp File Reference . . . . .	182
7.127 src/navigation/EKF.hpp File Reference . . . . .	182
7.128 src/navigation/environment.cpp File Reference . . . . .	183
7.128.1 Function Documentation . . . . .	183
7.128.1.1 connectConflateSocket() . . . . .	183
7.128.1.2 r_nb() [1/2] . . . . .	184
7.128.1.3 r_nb() [2/2] . . . . .	184
7.128.1.4 recvVectors() . . . . .	184
7.129 src/navigation/environment.hpp File Reference . . . . .	184
7.130 src/navigation/NS.cpp File Reference . . . . .	184
7.131 src/navigation/NS.hpp File Reference . . . . .	185
7.132 src/navigation/sensors.cpp File Reference . . . . .	185
7.133 src/navigation/sensors.hpp File Reference . . . . .	185
7.134 src/params.cpp File Reference . . . . .	186
7.135 src/params.hpp File Reference . . . . .	186
7.136 src/utils.hpp File Reference . . . . .	186
7.136.1 Function Documentation . . . . .	186
7.136.1.1 circularError() . . . . .	186
7.136.1.2 clampAngle() . . . . .	187
7.136.1.3 safeGet() . . . . .	187
7.136.1.4 safeSet() . . . . .	188
<b>Index</b>	<b>189</b>





# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">controllers</a> . . . . .	<a href="#">13</a>
<a href="#">def</a>	
<a href="#">Controller</a> constants . . . . .	<a href="#">13</a>



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

AeroCoefficients . . . . .	16
AHRS . . . . .	18
AHRS_EKF . . . . .	23
AHRS_complementary . . . . .	21
AHRSParams . . . . .	26
Control . . . . .	32
Controller . . . . .	37
controllers::BangBang . . . . .	28
controllers::DoubleSetpoint . . . . .	66
controllers::PID . . . . .	111
controllers::PID_Discrete . . . . .	114
controllers::ZTransform . . . . .	132
controllers::ZTransformStatic< N, D > . . . . .	134
ControllerLoop . . . . .	40
ControllerLoopFACRO . . . . .	45
ControllerLoopFANGLE . . . . .	46
ControllerLoopFMANUAL . . . . .	48
ControllerLoopNONE . . . . .	49
ControllerLoopQACRO . . . . .	50
ControllerLoopQANGLE . . . . .	51
ControllerLoopQPOS . . . . .	53
ControllerLoopRANGLE . . . . .	55
ControllerLoopRAUTOLAUNCH . . . . .	57
ControllerLoopRGUIDED . . . . .	58
ControllerLoopRMANUAL . . . . .	60
ControlSurfaces . . . . .	62
ControlSystem . . . . .	64
Drive . . . . .	69
Jet . . . . .	86
Rotor . . . . .	116
EKF . . . . .	70
EKFParams . . . . .	73
EKFScalers . . . . .	74
Environment . . . . .	75

Hinge . . . . .	83
Load . . . . .	88
Ammo . . . . .	27
Cargo . . . . .	31
Logger . . . . .	90
NS . . . . .	94
ODE . . . . .	97
ODE_Euler . . . . .	101
ODE_Heun . . . . .	103
ODE_PC2 . . . . .	104
ODE_PC4 . . . . .	105
ODE_RK4 . . . . .	107
Params . . . . .	109
Sensor< T > . . . . .	118
Sensor< double > . . . . .	118
Barometer . . . . .	30
Sensor< Eigen::Vector3d > . . . . .	118
Accelerometer . . . . .	15
GPS . . . . .	80
GPSVel . . . . .	81
Gyroscope . . . . .	82
Magnetometer . . . . .	93
SensorParams . . . . .	122
testing::TestWithParam	
ControllerTest . . . . .	62
ODETest . . . . .	108
TimedLoop . . . . .	123
UAVparams . . . . .	125

## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">Accelerometer</a>	
Representation of accelerometer . . . . .	15
<a href="#">AeroCoefficients</a>	
Aerodynamic coefficient . . . . .	16
<a href="#">AHRS</a>	
Attitude and heading reference system . . . . .	18
<a href="#">AHRS_complementary</a>	
Implementation of <a href="#">AHRS</a> based on Complementary Filter . . . . .	21
<a href="#">AHRS_EKF</a>	
Implementation of <a href="#">AHRS</a> based on Extended Kalman Filter . . . . .	23
<a href="#">AHRSParams</a>	
<a href="#">AHRS</a> parameters . . . . .	26
<a href="#">Ammo</a> . . . . .	27
<a href="#">controllers::BangBang</a> . . . . .	28
<a href="#">Barometer</a>	
Representation of barometer . . . . .	30
<a href="#">Cargo</a> . . . . .	31
<a href="#">Control</a>	
<a href="#">Control</a> command listener & sender . . . . .	32
<a href="#">Controller</a> . . . . .	37
<a href="#">ControllerLoop</a>	
This class is interface of controller modes. All modes should keep this strucure and implements all true virtual methods . . . . .	40
<a href="#">ControllerLoopFACRO</a> . . . . .	45
<a href="#">ControllerLoopFANGLE</a> . . . . .	46
<a href="#">ControllerLoopFMANUAL</a> . . . . .	48
<a href="#">ControllerLoopNONE</a> . . . . .	49
<a href="#">ControllerLoopQACRO</a> . . . . .	50
<a href="#">ControllerLoopQANGLE</a> . . . . .	51
<a href="#">ControllerLoopQPOS</a> . . . . .	53
<a href="#">ControllerLoopRANGLE</a> . . . . .	55
<a href="#">ControllerLoopRAUTOLAUNCH</a> . . . . .	57
<a href="#">ControllerLoopRGUIDED</a> . . . . .	58
<a href="#">ControllerLoopRMANUAL</a> . . . . .	60
<a href="#">ControllerTest</a> . . . . .	62

<a href="#">ControlSurfaces</a>	
Aircraft's control surfaces	62
<a href="#">ControlSystem</a>	
Central controller class	64
<a href="#">controllers::DoubleSetpoint</a>	66
<a href="#">Drive</a>	
Drive propelling aircraft	69
<a href="#">EKF</a>	
Extended Kalman Filter	70
<a href="#">EKFPParams</a>	
EK filer parameters	73
<a href="#">EKFScalers</a>	
Scalers for <a href="#">EKF</a>	74
<a href="#">Environment</a>	75
<a href="#">GPS</a>	
Representation of <a href="#">GPS</a> position measure	80
<a href="#">GPSVel</a>	
Representation of <a href="#">GPS</a> velocity measure	81
<a href="#">Gyroscope</a>	
Representation of gyroscope	82
<a href="#">Hinge</a>	
Hinge connecting aircraft with drives	83
<a href="#">Jet</a>	
Jet rocket engine	86
<a href="#">Load</a>	
Load of aircraft that can be dropped or launched	88
<a href="#">Logger</a>	
Log vector data with timestamp in file	90
<a href="#">Magnetometer</a>	
Representation of magnetometer	93
<a href="#">NS</a>	
Navigation system	94
<a href="#">ODE</a>	
Ordinal differencial equation solver	97
<a href="#">ODE_Euler</a>	
Explicit Euler algorithm	101
<a href="#">ODE_Heun</a>	
Second order explicit Heun algorithm	103
<a href="#">ODE_PC2</a>	
Second order predictor-corrector method Second order Adams-bashforth and Adams-moulton	104
<a href="#">ODE_PC4</a>	
Fourth order predictor-corrector method Fourth order Adams-bashforth and Adams-moulton	105
<a href="#">ODE_RK4</a>	
Fourth order Runge Kutta algorithm	107
<a href="#">ODETest</a>	108
<a href="#">Params</a>	
Simulation parameters	109
<a href="#">controllers::PID</a>	111
<a href="#">controllers::PID_Discrete</a>	114
<a href="#">Rotor</a>	
Rotor engine with controlled speed	116
<a href="#">Sensor&lt; T &gt;</a>	
Sensors base class	118
<a href="#">SensorParams</a>	
Base parameters of a sensor	122
<a href="#">TimedLoop</a>	
Simulation of real-time synchronized loop	123

---

UAVparams	
Parsed UAV configuration from XML	125
controllers::ZTransform	132
controllers::ZTransformStatic< N, D >	134





## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

build/CMakeFiles/3.22.1/CompilerIdC/CMakeCCompilerId.c	137
build/CMakeFiles/3.22.1/CompilerIdCXX/CMakeCXXCompilerId.cpp	140
build/CMakeFiles/controller.dir/src/main.cpp.o.d	145
build/CMakeFiles/controller.dir/src/params.cpp.o.d	145
build/CMakeFiles/controller.dir/src/communication/control.cpp.o.d	145
build/CMakeFiles/controller.dir/src/communication/control_recv.cpp.o.d	145
build/CMakeFiles/controller.dir/src/communication/control_send.cpp.o.d	145
build/CMakeFiles/controller.dir/src/controller/controller.cpp.o.d	145
build/CMakeFiles/controller.dir/src/controller/controller_loop.cpp.o.d	145
build/CMakeFiles/controller.dir/src/controller/mixers.cpp.o.d	145
build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_FACRO.cpp.o.d	145
build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_FANGLE.cpp.o.d	145
build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_FMANUAL.cpp.o.d	145
build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_NONE.cpp.o.d	145
build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_QACRO.cpp.o.d	145
build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_QANGLE.cpp.o.d	145
build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_QPOS.cpp.o.d	145
build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_RANGLE.cpp.o.d	145
build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_RAUTOLAUNCH.cpp.o.d	145
build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_RGUIDED.cpp.o.d	145
build/CMakeFiles/controller.dir/src/controller/modes/controller_loop_RMANUAL.cpp.o.d	145
build/CMakeFiles/controller.dir/src/navigation/AHRS.cpp.o.d	145
build/CMakeFiles/controller.dir/src/navigation/EKF.cpp.o.d	145
build/CMakeFiles/controller.dir/src/navigation/environment.cpp.o.d	145
build/CMakeFiles/controller.dir/src/navigation/NS.cpp.o.d	145
build/CMakeFiles/controller.dir/src/navigation/sensors.cpp.o.d	145
build/CMakeFiles/controller.dir/src/navigation/AHRS/AHRS_complementary.cpp.o.d	145
build/CMakeFiles/controller.dir/src/navigation/AHRS/AHRS_EKF.cpp.o.d	145
build/lib/UAV_common/CMakeFiles/common.dir/src/components/control_surfaces.cpp.o.d	145
build/lib/UAV_common/CMakeFiles/common.dir/src/components/drive.cpp.o.d	145
build/lib/UAV_common/CMakeFiles/common.dir/src/components/hinge.cpp.o.d	145
build/lib/UAV_common/CMakeFiles/common.dir/src/components/loads.cpp.o.d	145
build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/controller.cpp.o.d	145
build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/bang_bang.cpp.o.d	145
build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/double_setpoint.cpp.o.d	145

build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/PID.cpp.o.d	145
build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/PID_discrete.cpp.o.d	145
build/lib/UAV_common/CMakeFiles/common.dir/src/controllers/impl/z_trans.cpp.o.d	145
build/lib/UAV_common/CMakeFiles/common.dir/src/logger/logger.cpp.o.d	145
build/lib/UAV_common/CMakeFiles/common.dir/src/ode/ode.cpp.o.d	145
build/lib/UAV_common/CMakeFiles/common.dir/src/parser/parser.cpp.o.d	145
build/lib/UAV_common/CMakeFiles/common.dir/src/parser/uav_params.cpp.o.d	145
build/lib/UAV_common/CMakeFiles/common.dir/src/timed_loop/timed_loop.cpp.o.d	145
build/lib/UAV_common/CMakeFiles/Controller_test.dir/src/controllers/controller_test.cpp.o.d	145
build/lib/UAV_common/CMakeFiles/ODE_test.dir/src/ode/ode_test.cpp.o.d	145
lib/UAV_common/header/common.hpp	145
lib/UAV_common/scripts/controller_plots.m	146
lib/UAV_common/src/components/aero_coefficients.hpp	149
lib/UAV_common/src/components/components.hpp	149
lib/UAV_common/src/components/control_surfaces.cpp	149
lib/UAV_common/src/components/control_surfaces.hpp	149
lib/UAV_common/src/components/drive.cpp	150
lib/UAV_common/src/components/drive.hpp	150
lib/UAV_common/src/components/hinge.cpp	150
lib/UAV_common/src/components/hinge.hpp	150
lib/UAV_common/src/components/loads.cpp	151
lib/UAV_common/src/components/loads.hpp	151
lib/UAV_common/src/components/navi.hpp	151
lib/UAV_common/src/controllers/controller.cpp	151
lib/UAV_common/src/controllers/controller.hpp	152
lib/UAV_common/src/controllers/controller_test.cpp	152
lib/UAV_common/src/controllers/impl/bang_bang.cpp	154
lib/UAV_common/src/controllers/impl/bang_bang.hpp	154
lib/UAV_common/src/controllers/impl/double_setpoint.cpp	155
lib/UAV_common/src/controllers/impl/double_setpoint.hpp	155
lib/UAV_common/src/controllers/impl/PID.cpp	155
lib/UAV_common/src/controllers/impl/PID.hpp	156
lib/UAV_common/src/controllers/impl/PID_discrete.cpp	156
lib/UAV_common/src/controllers/impl/PID_discrete.hpp	156
lib/UAV_common/src/controllers/impl/z_trans.cpp	157
lib/UAV_common/src/controllers/impl/z_trans.hpp	157
lib/UAV_common/src/logger/logger.cpp	158
lib/UAV_common/src/logger/logger.hpp	158
lib/UAV_common/src/ode/ode.cpp	159
lib/UAV_common/src/ode/ode.hpp	159
lib/UAV_common/src/ode/ode_impl.hpp	159
lib/UAV_common/src/ode/ode_test.cpp	159
lib/UAV_common/src/parser/parser.cpp	161
lib/UAV_common/src/parser/parser.hpp	163
lib/UAV_common/src/parser/uav_params.cpp	164
lib/UAV_common/src/parser/uav_params.hpp	164
lib/UAV_common/src/timed_loop/status.hpp	165
lib/UAV_common/src/timed_loop/timed_loop.cpp	165
lib/UAV_common/src/timed_loop/timed_loop.hpp	165
src/defines.hpp	177
src/main.cpp	178
src/params.cpp	186
src/params.hpp	186
src/utlis.hpp	186
src/communication/control.cpp	166
src/communication/control.hpp	166
src/communication/control_recv.cpp	167
src/communication/control_send.cpp	167

src/controller/controller.cpp	152
src/controller/controller.hpp	152
src/controller/controller_loop.cpp	167
src/controller/controller_loop.hpp	167
src/controller/controller_mode.hpp	168
src/controller/mixers.cpp	169
src/controller/mixers.hpp	171
src/controller/modes/controller_loop_FACRO.cpp	173
src/controller/modes/controller_loop_FACRO.hpp	173
src/controller/modes/controller_loop_FANGLE.cpp	173
src/controller/modes/controller_loop_FANGLE.hpp	174
src/controller/modes/controller_loop_FMANUAL.cpp	174
src/controller/modes/controller_loop_FMANUAL.hpp	174
src/controller/modes/controller_loop_NONE.cpp	174
src/controller/modes/controller_loop_NONE.hpp	174
src/controller/modes/controller_loop_QACRO.cpp	175
src/controller/modes/controller_loop_QACRO.hpp	175
src/controller/modes/controller_loop_QANGLE.cpp	175
src/controller/modes/controller_loop_QANGLE.hpp	175
src/controller/modes/controller_loop_QPOS.cpp	175
src/controller/modes/controller_loop_QPOS.hpp	175
src/controller/modes/controller_loop_RANGLE.cpp	176
src/controller/modes/controller_loop_RANGLE.hpp	176
src/controller/modes/controller_loop_RAUTOLAUNCH.cpp	176
src/controller/modes/controller_loop_RAUTOLAUNCH.hpp	176
src/controller/modes/controller_loop_RGUIDED.cpp	176
src/controller/modes/controller_loop_RGUIDED.hpp	177
src/controller/modes/controller_loop_RMANUAL.cpp	177
src/controller/modes/controller_loop_RMANUAL.hpp	177
src/navigation/AHRS.cpp	180
src/navigation/AHRS.hpp	180
src/navigation/EKF.cpp	182
src/navigation/EKF.hpp	182
src/navigation/environment.cpp	183
src/navigation/environment.hpp	184
src/navigation/NS.cpp	184
src/navigation/NS.hpp	185
src/navigation/sensors.cpp	185
src/navigation/sensors.hpp	185
src/navigation/AHRS/AHRS_complementary.cpp	180
src/navigation/AHRS/AHRS_complementary.hpp	181
src/navigation/AHRS/AHRS_EKF.cpp	181
src/navigation/AHRS/AHRS_EKF.hpp	182



## Chapter 5

# Namespace Documentation

### 5.1 controllers Namespace Reference

#### Classes

- class [BangBang](#)
- class [DoubleSetpoint](#)
- class [PID](#)
- class [PID\\_Discrete](#)
- class [ZTransformStatic](#)
- class [ZTransform](#)

### 5.2 def Namespace Reference

[Controller](#) constants.

#### Variables

- const int [INFO\\_PERIOD](#) = 2  
*How often send demands in response to stick command.*

#### 5.2.1 Detailed Description

[Controller](#) constants.

#### 5.2.2 Variable Documentation

##### 5.2.2.1 INFO\_PERIOD

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

How often send demands in response to stick command.



## Chapter 6

# Class Documentation

### 6.1 Accelerometer Class Reference

Representation of accelerometer.

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

Inheritance diagram for Accelerometer:

Collaboration diagram for Accelerometer:

#### Public Member Functions

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

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

#### Static Public Attributes

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

#### Additional Inherited Members

#### 6.1.1 Detailed Description

Representation of accelerometer.

#### 6.1.2 Constructor & Destructor Documentation

### 6.1.2.1 Accelerometer()

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

## 6.1.3 Member Function Documentation

### 6.1.3.1 update()

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

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

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

## 6.1.4 Member Data Documentation

### 6.1.4.1 g

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

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

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

## 6.2 AeroCoefficients Struct Reference

Aerodynamic coefficient.

```
#include <aero_coefficients.hpp>
```

### Public Attributes

- double [S](#)
- double [d](#)
- double [eAR](#)
- Eigen::Vector< double, 6 > [C0](#)
- Eigen::Matrix< double, 6, 3 > [Cpqr](#)
- Eigen::Matrix< double, 6, 4 > [Cab](#)
- double [stallLimit](#)



## 6.2.1 Detailed Description

Aerodynamic coefficient.

## 6.2.2 Member Data Documentation

### 6.2.2.1 C0

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

### 6.2.2.2 Cab

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

### 6.2.2.3 Cpqr

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

### 6.2.2.4 d

```
double AeroCoefficients::d
```

### 6.2.2.5 eAR

```
double AeroCoefficients::eAR
```

### 6.2.2.6 S

```
double AeroCoefficients::S
```

### 6.2.2.7 stallLimit

```
double AeroCoefficients::stallLimit
```

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

- [lib/UAV\\_common/src/components/aero\\_coefficients.hpp](#)

## 6.3 AHRS Class Reference

Attitude and heading reference system.

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

Inheritance diagram for AHRS:

Collaboration diagram for AHRS:

### Public Member Functions

- [AHRS](#) ([Environment](#) &*env*)  
*Constructor.*
- [~AHRS](#) ()  
*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 <a href="#">AHRS</a> 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 controllers::BangBang Class Reference

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

Inheritance diagram for controllers::BangBang:

Collaboration diagram for controllers::BangBang:

### Public Member Functions

- [BangBang](#) (double high, double low, double delta=0.0)  
*Constructor with all Bang-bang controller parameters.*
- [BangBang](#) (rapidxml::xml\_node<> \*controller\_node)  
*Construct controller with parameters from xml.*
- double [calc](#) (double desired, double actual, [[maybe\_unused]] double dt) override  
*calc output of controller with specific time step*
- void [clear](#) () override  
*clear internal state*
- std::unique\_ptr< [Controller](#) > [clone](#) () const override  
*virtual clone method*

## Additional Inherited Members

### 6.8.1 Constructor & Destructor Documentation

#### 6.8.1.1 BangBang() [1/2]

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

Constructor with all Bang-bang controller parameters.

##### Parameters

<i>high</i>	output when error is positive
<i>low</i>	output when error is negative
<i>delta</i>	hysteresis symetrical to zero

#### 6.8.1.2 BangBang() [2/2]

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

Construct controller with parameters from xml.

##### Parameters

<i>controller_node</i>	xml node with controller params
------------------------	---------------------------------

### 6.8.2 Member Function Documentation

#### 6.8.2.1 calc()

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

calc output of controller with specific time step

**Parameters**

<i>desired</i>	input of controller, desired value
<i>actual</i>	measured actual value
<i>dt</i>	time step

**Returns**

output of controller

**6.8.2.2 clear()**

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

clear internal state

Implements [Controller](#).

**6.8.2.3 clone()**

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

virtual clone method

Implements [Controller](#).

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

- [lib/UAV\\_common/src/controllers/impl/bang\\_bang.hpp](#)
- [lib/UAV\\_common/src/controllers/impl/bang\\_bang.cpp](#)

**6.9 Barometer Class Reference**

Representation of barometer.

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

Inheritance diagram for Barometer:

Collaboration diagram for Barometer:

## Public Member Functions

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

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

## Additional Inherited Members

### 6.9.1 Detailed Description

Representation of barometer.

### 6.9.2 Constructor & Destructor Documentation

#### 6.9.2.1 Barometer()

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

### 6.9.3 Member Function Documentation

#### 6.9.3.1 update()

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

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

Implements [Sensor< double >](#).

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

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

## 6.10 Cargo Class Reference

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

Inheritance diagram for Cargo:

Collaboration diagram for Cargo:

## Public Member Functions

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

## Additional Inherited Members

### 6.10.1 Constructor & Destructor Documentation

#### 6.10.1.1 Cargo() [1/2]

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

#### 6.10.1.2 Cargo() [2/2]

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

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

- [lib/UAV\\_common/src/components/loads.hpp](#)
- [lib/UAV\\_common/src/components/loads.cpp](#)

## 6.11 Control Class Reference

[Control](#) command listener & sender.

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



## Public Member Functions

- [Control](#) (zmq::context\_t \*ctx, std::string uav\_address, [ControlSystem](#) \*controller)  
*Constructor.*
- [~Control](#) ()  
*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.*
- void [setMode](#) ([ControllerMode](#) mode)

### 6.11.1 Detailed Description

[Control](#) command listener & sender.

### 6.11.2 Constructor & Destructor Documentation

#### 6.11.2.1 Control()

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

Constructor.

#### Parameters

<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

### 6.11.2.2 ~Control()

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

Deconstructor.

## 6.11.3 Member Function Documentation

### 6.11.3.1 handleMsg()

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

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

#### Parameters

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

#### Returns

reply to message

### 6.11.3.2 prepare()

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

Sends ping command.

### 6.11.3.3 recv()

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

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

#### 6.11.3.4 sendHinge()

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

Sends command to control hinge 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

**6.11.3.5 sendSpeed()**

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

Sends new demanded rotors speed.

**Parameters**

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

**6.11.3.6 sendSurface()**

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

Sends new demanded surface deflections.

**Parameters**

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

**6.11.3.7 setMode()**

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

**6.11.3.8 start()**

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

Sends start command.

### 6.11.3.9 startJet()

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

Sends command to start jet engine of given index.

#### Parameters

<i>index</i>	jet engine index
--------------	------------------

### 6.11.3.10 stop()

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

Sends stop command.

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

- [src/communication/control.hpp](#)
- [src/communication/control.cpp](#)
- [src/communication/control\\_recv.cpp](#)
- [src/communication/control\\_send.cpp](#)

## 6.12 Controller Class Reference

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

Inheritance diagram for Controller:

### Public Member Functions

- [Controller](#) ()  
*Default constructor.*
- [~Controller](#) ()  
*Empty destructor for derived classes.*
- virtual void [set\\_dt](#) (double dt)  
*Set new time step.*
- double [calc](#) (double desired, double actual)  
*calc output of controller*
- virtual double [calc](#) (double desired, double actual, double dt)=0  
*calc output of controller with specific time step*
- virtual void [clear](#) ()=0  
*clear internal state*
- virtual std::unique\_ptr< [Controller](#) > [clone](#) () const =0  
*virtual clone method*

## Static Public Member Functions

- static std::unique\_ptr< [Controller](#) > [ControllerFactory](#) (rapidxml::xml\_node<> \*controller\_node)  
*construct controller from given node. If xml is not valid return nullptr.*

## Protected Attributes

- double [\\_dt](#)

## 6.12.1 Constructor & Destructor Documentation

### 6.12.1.1 Controller()

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

Default constructor.

### 6.12.1.2 ~Controller()

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

Empty deconstructor for derived classes.

## 6.12.2 Member Function Documentation

### 6.12.2.1 calc() [1/2]

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

calc output of controller

#### Parameters

<i>desired</i>	input of controller, desired value
<i>actual</i>	measured actual value

**Returns**

output of controller

**6.12.2.2 calc() [2/2]**

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

calc output of controller with specific time step

**Parameters**

<i>desired</i>	input of controller, desired value
<i>actual</i>	measured actual value
<i>dt</i>	time step

**Returns**

output of controller

Implemented in [controllers::PID\\_Discrete](#), [controllers::PID](#), and [controllers::DoubleSetpoint](#).

**6.12.2.3 clear()**

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

clear internal state

Implemented in [controllers::ZTransform](#), [controllers::ZTransformStatic< N, D >](#), [controllers::PID\\_Discrete](#), [controllers::PID](#), [controllers::DoubleSetpoint](#), and [controllers::BangBang](#).

**6.12.2.4 clone()**

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

virtual clone method

Implemented in [controllers::ZTransform](#), [controllers::ZTransformStatic< N, D >](#), [controllers::PID\\_Discrete](#), [controllers::PID](#), [controllers::DoubleSetpoint](#), and [controllers::BangBang](#).

**6.12.2.5 ControllerFactory()**

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

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

## Parameters

<i>controller_node</i>	xml node with controller config
------------------------	---------------------------------

**6.12.2.6 set\_dt()**

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

Set new time step.

## Parameters

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

Reimplemented in [controllers::PID\\_Discrete](#).

**6.12.3 Member Data Documentation****6.12.3.1 \_dt**

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

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

- lib/UAV\_common/src/controllers/[controller.hpp](#)
- lib/UAV\_common/src/controllers/[controller.cpp](#)

**6.13 ControllerLoop Class Reference**

This class is interface of controller modes. All modes should keep this 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, std::unique\_ptr< [Controller](#) >> &controllers, [[maybe\_unused]] [Control](#) &control, [[maybe\_unused]] [NS](#) &navisys)  
*[Controller](#) job that will be called in control loop.*
- virtual void [handleJoystick](#) ([[maybe\_unused]] Eigen::VectorXd joystick)  
*Handle incomming joystick deflection.*
- virtual std::string [demandInfo](#) ()  
*Prepare info about state and demands.*
- virtual const std::vector< std::string > & [requiredcontrollers](#) ()  
*Defines controllers controller required by mode.*
- virtual void [overridePositionAndSpeed](#) ([[maybe\_unused]] Eigen::Vector3d position, [[maybe\_unused]] Eigen::Vector3d orientation, [[maybe\_unused]] Eigen::Vector3d velocity)  
*Overrides demands to apply to given postion, orientation and speed.*
- [ControllerMode](#) [getMode](#) ()  
*Returns assigned mode enum value.*

## Static Public Member Functions

- static [ControllerLoop](#) \* [ControllerLoopFactory](#) ([ControllerMode](#) mode)  
*[ControllerLoop](#) factor. Returns instace of [ControllerLoop](#) that implements specified mode.*

## Protected Member Functions

- bool [checkJoystickLength](#) (const Eigen::VectorXd &joystick, const int minimalSize)  
*Check if joystick input vector is correct.*

## Protected Attributes

- const [ControllerMode](#) [\\_mode](#)
- std::vector< std::string > [required\\_controllers](#)

### 6.13.1 Detailed Description

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

### 6.13.2 Constructor & Destructor Documentation

#### 6.13.2.1 ControllerLoop()

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

Base class constructor.

## Parameters

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

**6.13.2.2 ~ControllerLoop()**

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

Virtual destructor for defined behavior.

**6.13.3 Member Function Documentation****6.13.3.1 checkJoystickLength()**

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

Check if joystick input vector is correct.

## Parameters

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

## Returns

return true if joystick input vector is long enough

**6.13.3.2 ControllerLoopFactory()**

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

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

## Parameters

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

**Returns**

Pointer to dynamically allocated [ControllerLoop](#)

**6.13.3.3 demandInfo()**

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

Prepare info about state and demands.

**Returns**

information about mode and actually set demands

Reimplemented in [ControllerLoopRMANUAL](#), [ControllerLoopRGUIDED](#), [ControllerLoopRANGLE](#), [ControllerLoopQPOS](#), [ControllerLoopQANGLE](#), [ControllerLoopQACRO](#), [ControllerLoopFANGLE](#), and [ControllerLoopFACRO](#).

**6.13.3.4 getMode()**

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

Returns assigned mode enum value.

**Returns**

mode enum value

**6.13.3.5 handleJoystick()**

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

Handle incoming joystick deflection.

**Parameters**

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

**6.13.3.6 job()**

```
void ControllerLoop::job (
```

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

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

#### Parameters

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

#### 6.13.3.7 overridePositionAndSpeed()

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

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

#### Parameters

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

Reimplemented in [ControllerLoopRANGLE](#), [ControllerLoopQPOS](#), [ControllerLoopQANGLE](#), and [ControllerLoopFANGLE](#).

#### 6.13.3.8 requiredcontrollers()

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

Defines controllers controller required by mode.

#### Returns

vector of names of required controllers

### 6.13.4 Member Data Documentation

#### 6.13.4.1 `_mode`

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

#### 6.13.4.2 `required_controllers`

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

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

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

## 6.14 ControllerLoopFACRO Class Reference

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

Inheritance diagram for ControllerLoopFACRO:

Collaboration diagram for ControllerLoopFACRO:

### Public Member Functions

- [ControllerLoopFACRO](#) ()
  - void [job](#) (std::map< std::string, std::unique\_ptr< [Controller](#) >> &controllers, [Control](#) &control, [NS](#) &navisys) override
  - void [handleJoystick](#) (Eigen::VectorXd joystick) override
  - std::string [demandInfo](#) () override
- Prepare info about state and demands.*

### Additional Inherited Members

#### 6.14.1 Constructor & Destructor Documentation

##### 6.14.1.1 ControllerLoopFACRO()

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

#### 6.14.2 Member Function Documentation

### 6.14.2.1 demandInfo()

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

Prepare info about state and demands.

#### Returns

information about mode and actually set demands

Reimplemented from [ControllerLoop](#).

### 6.14.2.2 handleJoystick()

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

### 6.14.2.3 job()

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

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

- src/controller/modes/[controller\\_loop\\_FACRO.hpp](#)
- src/controller/modes/[controller\\_loop\\_FACRO.cpp](#)

## 6.15 ControllerLoopFANGLE Class Reference

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

Inheritance diagram for ControllerLoopFANGLE:

Collaboration diagram for ControllerLoopFANGLE:

### Public Member Functions

- [ControllerLoopFANGLE](#) ()
- void [job](#) (std::map< std::string, std::unique\_ptr< [Controller](#) >> &controllers, [Control](#) &control, [NS](#) &navisys) override
- void [handleJoystick](#) (Eigen::VectorXd joystick) override
- std::string [demandInfo](#) () override
 

*Prepare info about state and demands.*
- void [overridePositionAndSpeed](#) ([[maybe\_unused]] Eigen::Vector3d position, [[maybe\_unused]] Eigen::↔ Vector3d orientation, [[maybe\_unused]] Eigen::Vector3d velocity) override
 

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

## Static Public Attributes

- static constexpr double [angleLimit](#) = std::numbers::pi/2.0

## Additional Inherited Members

### 6.15.1 Constructor & Destructor Documentation

#### 6.15.1.1 ControllerLoopFANGLE()

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

### 6.15.2 Member Function Documentation

#### 6.15.2.1 demandInfo()

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

Prepare info about state and demands.

##### Returns

information about mode and actually set demands

Reimplemented from [ControllerLoop](#).

#### 6.15.2.2 handleJoystick()

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

#### 6.15.2.3 job()

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

#### 6.15.2.4 overridePositionAndSpeed()

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

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

## Parameters

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

Reimplemented from [ControllerLoop](#).

### 6.15.3 Member Data Documentation

#### 6.15.3.1 angleLimit

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

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

- src/controller/modes/[controller\\_loop\\_FANGLE.hpp](#)
- src/controller/modes/[controller\\_loop\\_FANGLE.cpp](#)

## 6.16 ControllerLoopFMANUAL Class Reference

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

Inheritance diagram for ControllerLoopFMANUAL:

Collaboration diagram for ControllerLoopFMANUAL:

### Public Member Functions

- [ControllerLoopFMANUAL](#) ()
- void [job](#) ([[maybe\_unused]] std::map< std::string, std::unique\_ptr< [Controller](#) >> &controllers, [Control](#) &control, [[maybe\_unused]] [NS](#) &navisys) override
- void [handleJoystick](#) (Eigen::VectorXd joystick) override

### Additional Inherited Members

#### 6.16.1 Constructor & Destructor Documentation

##### 6.16.1.1 ControllerLoopFMANUAL()

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



## 6.16.2 Member Function Documentation

### 6.16.2.1 handleJoystick()

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

### 6.16.2.2 job()

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

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

- [src/controller/modes/controller\\_loop\\_FMANUAL.hpp](#)
- [src/controller/modes/controller\\_loop\\_FMANUAL.cpp](#)

## 6.17 ControllerLoopNONE Class Reference

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

Inheritance diagram for ControllerLoopNONE:

Collaboration diagram for ControllerLoopNONE:

### Public Member Functions

- [ControllerLoopNONE](#) ()

### Additional Inherited Members

### 6.17.1 Constructor & Destructor Documentation

### 6.17.1.1 ControllerLoopNONE()

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

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

- src/controller/modes/[controller\\_loop\\_NONE.hpp](#)
- src/controller/modes/[controller\\_loop\\_NONE.cpp](#)

## 6.18 ControllerLoopQACRO Class Reference

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

Inheritance diagram for ControllerLoopQACRO:

Collaboration diagram for ControllerLoopQACRO:

### Public Member Functions

- [ControllerLoopQACRO](#) ()
- void [job](#) (std::map< std::string, std::unique\_ptr< [Controller](#) >> &controllers, [Control](#) &control, [NS](#) &navisys) override
- void [handleJoystick](#) (Eigen::VectorXd joystick) override
- std::string [demandInfo](#) () override  
*Prepare info about state and demands.*

### Additional Inherited Members

## 6.18.1 Constructor & Destructor Documentation

### 6.18.1.1 ControllerLoopQACRO()

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

## 6.18.2 Member Function Documentation

### 6.18.2.1 demandInfo()

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

Prepare info about state and demands.

#### Returns

information about mode and actually set demands

Reimplemented from [ControllerLoop](#).

### 6.18.2.2 handleJoystick()

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

### 6.18.2.3 job()

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

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

- [src/controller/modes/controller\\_loop\\_QACRO.hpp](#)
- [src/controller/modes/controller\\_loop\\_QACRO.cpp](#)

## 6.19 ControllerLoopQANGLE Class Reference

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

Inheritance diagram for ControllerLoopQANGLE:

Collaboration diagram for ControllerLoopQANGLE:

### Public Member Functions

- [ControllerLoopQANGLE](#) ()
- void [job](#) (std::map< std::string, std::unique\_ptr< [Controller](#) >> &controllers, [Control](#) &control, [NS](#) &navisys) override
- void [handleJoystick](#) (Eigen::VectorXd joystick) override
- std::string [demandInfo](#) () override
 

*Prepare info about state and demands.*
- void [overridePositionAndSpeed](#) ([[maybe\_unused]] Eigen::Vector3d position, [[maybe\_unused]] Eigen::↔ Vector3d orientation, [[maybe\_unused]] Eigen::Vector3d velocity) override
 

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

## Additional Inherited Members

### 6.19.1 Constructor & Destructor Documentation

#### 6.19.1.1 ControllerLoopQANGLE()

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

### 6.19.2 Member Function Documentation

#### 6.19.2.1 demandInfo()

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

Prepare info about state and demands.

##### Returns

information about mode and actually set demands

Reimplemented from [ControllerLoop](#).

#### 6.19.2.2 handleJoystick()

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

#### 6.19.2.3 job()

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

#### 6.19.2.4 overridePositionAndSpeed()

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

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

## Parameters

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

Reimplemented from [ControllerLoop](#).

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

- src/controller/modes/[controller\\_loop\\_QANGLE.hpp](#)
- src/controller/modes/[controller\\_loop\\_QANGLE.cpp](#)

## 6.20 ControllerLoopQPOS Class Reference

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

Inheritance diagram for ControllerLoopQPOS:

Collaboration diagram for ControllerLoopQPOS:

### Public Member Functions

- [ControllerLoopQPOS](#) ()
- void [job](#) (std::map< std::string, std::unique\_ptr< [Controller](#) >> &controllers, [Control](#) &control, [NS](#) &navisys) override
- void [handleJoystick](#) (Eigen::VectorXd joystick) override
- std::string [demandInfo](#) () override
 

*Prepare info about state and demands.*
- void [overridePositionAndSpeed](#) ([[maybe\_unused]] Eigen::Vector3d position, [[maybe\_unused]] Eigen::Vector3d orientation, [[maybe\_unused]] Eigen::Vector3d velocity) override
 

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

### Additional Inherited Members

#### 6.20.1 Constructor & Destructor Documentation

##### 6.20.1.1 ControllerLoopQPOS()

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

#### 6.20.2 Member Function Documentation

### 6.20.2.1 demandInfo()

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

Prepare info about state and demands.

#### Returns

information about mode and actually set demands

Reimplemented from [ControllerLoop](#).

### 6.20.2.2 handleJoystick()

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

### 6.20.2.3 job()

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

### 6.20.2.4 overridePositionAndSpeed()

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

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

#### Parameters

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

Reimplemented from [ControllerLoop](#).

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

- [src/controller/modes/controller\\_loop\\_QPOS.hpp](#)
- [src/controller/modes/controller\\_loop\\_QPOS.cpp](#)

## 6.21 ControllerLoopRANGLE Class Reference

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

Inheritance diagram for ControllerLoopRANGLE:

Collaboration diagram for ControllerLoopRANGLE:

### Public Member Functions

- [ControllerLoopRANGLE](#) ()
- void [job](#) ([[maybe\_unused]] std::map< std::string, std::unique\_ptr< [Controller](#) >> &controllers, [Control](#) &control, [[maybe\_unused]] [NS](#) &navisys) override
- void [handleJoystick](#) (Eigen::VectorXd joystick) override
- std::string [demandInfo](#) () override  
*Prepare info about state and demands.*
- void [overridePositionAndSpeed](#) ([[maybe\_unused]] Eigen::Vector3d position, [[maybe\_unused]] Eigen::Vector3d orientation, [[maybe\_unused]] Eigen::Vector3d velocity) override  
*Overrides demands to apply to given position, orientation and speed.*

### Protected Attributes

- std::atomic< double > [demandedTheta](#) = 0.0
- std::atomic< double > [demandedPsi](#) = 0.0

### Static Protected Attributes

- static constexpr double [angleLimit](#) = std::numbers::pi/2.0

### Additional Inherited Members

#### 6.21.1 Constructor & Destructor Documentation

##### 6.21.1.1 ControllerLoopRANGLE()

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

#### 6.21.2 Member Function Documentation

### 6.21.2.1 demandInfo()

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

Prepare info about state and demands.

#### Returns

information about mode and actually set demands

Reimplemented from [ControllerLoop](#).

### 6.21.2.2 handleJoystick()

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

### 6.21.2.3 job()

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

### 6.21.2.4 overridePositionAndSpeed()

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

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

#### Parameters

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

Reimplemented from [ControllerLoop](#).



## 6.21.3 Member Data Documentation

### 6.21.3.1 angleLimit

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

### 6.21.3.2 demandedPsi

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

### 6.21.3.3 demandedTheta

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

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

- src/controller/modes/[controller\\_loop\\_RANGLE.hpp](#)
- src/controller/modes/[controller\\_loop\\_RANGLE.cpp](#)

## 6.22 ControllerLoopRAUTOLAUNCH Class Reference

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

Inheritance diagram for ControllerLoopRAUTOLAUNCH:

Collaboration diagram for ControllerLoopRAUTOLAUNCH:

### Public Member Functions

- [ControllerLoopRAUTOLAUNCH](#) ()
- void [job](#) ([[maybe\_unused]] std::map< std::string, std::unique\_ptr< [Controller](#) >> &controllers, [Control](#) &control, [[maybe\_unused]] [NS](#) &navisys) override

### Additional Inherited Members

#### 6.22.1 Constructor & Destructor Documentation

### 6.22.1.1 ControllerLoopRAUTOLAUNCH()

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

## 6.22.2 Member Function Documentation

### 6.22.2.1 job()

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

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

- src/controller/modes/[controller\\_loop\\_RAUTOLAUNCH.hpp](#)
- src/controller/modes/[controller\\_loop\\_RAUTOLAUNCH.cpp](#)

## 6.23 ControllerLoopRGUIDED Class Reference

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

Inheritance diagram for ControllerLoopRGUIDED:

Collaboration diagram for ControllerLoopRGUIDED:

### Public Member Functions

- [ControllerLoopRGUIDED](#) ()
- void [job](#) ([[maybe\_unused]] std::map< std::string, std::unique\_ptr< [Controller](#) >> &controllers, [Control](#) &control, [[maybe\_unused]] [NS](#) &navisys) override
- std::string [demandInfo](#) () override  
*Prepare info about state and demands.*

### Protected Attributes

- const Eigen::Vector3d [target](#)

### Static Protected Attributes

- static constexpr double [detection\\_limit](#) = std::numbers::pi/3.0

## Additional Inherited Members

### 6.23.1 Constructor & Destructor Documentation

#### 6.23.1.1 ControllerLoopRGUIDED()

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

### 6.23.2 Member Function Documentation

#### 6.23.2.1 demandInfo()

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

Prepare info about state and demands.

##### Returns

information about mode and actually set demands

Reimplemented from [ControllerLoop](#).

#### 6.23.2.2 job()

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

### 6.23.3 Member Data Documentation

#### 6.23.3.1 detection\_limit

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

### 6.23.3.2 target

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

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

- src/controller/modes/[controller\\_loop\\_RGUIDED.hpp](#)
- src/controller/modes/[controller\\_loop\\_RGUIDED.cpp](#)

## 6.24 ControllerLoopRMANUAL Class Reference

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

Inheritance diagram for ControllerLoopRMANUAL:

Collaboration diagram for ControllerLoopRMANUAL:

### Public Member Functions

- [ControllerLoopRMANUAL](#) ()
  - void [job](#) ([[maybe\_unused]] std::map< std::string, std::unique\_ptr< [Controller](#) >> &controllers, [Control](#) &control, [[maybe\_unused]] [NS](#) &navisys) override
  - void [handleJoystick](#) (Eigen::VectorXd joystick) override
  - std::string [demandInfo](#) () override
- Prepare info about state and demands.*

### Protected Attributes

- std::atomic< double > [demanded\\_H](#) = 0.0
- std::atomic< double > [demanded\\_V](#) = 0.0

### Additional Inherited Members

## 6.24.1 Constructor & Destructor Documentation

### 6.24.1.1 ControllerLoopRMANUAL()

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

## 6.24.2 Member Function Documentation

### 6.24.2.1 demandInfo()

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

Prepare info about state and demands.

#### Returns

information about mode and actually set demands

Reimplemented from [ControllerLoop](#).

### 6.24.2.2 handleJoystick()

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

### 6.24.2.3 job()

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

## 6.24.3 Member Data Documentation

### 6.24.3.1 demanded\_H

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

### 6.24.3.2 demanded\_V

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

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

- [src/controller/modes/controller\\_loop\\_RMANUAL.hpp](#)
- [src/controller/modes/controller\\_loop\\_RMANUAL.cpp](#)

## 6.25 ControllerTest Class Reference

Inheritance diagram for ControllerTest:

Collaboration diagram for ControllerTest:

### Protected Member Functions

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

### 6.25.1 Member Function Documentation

#### 6.25.1.1 SetUp()

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

#### 6.25.1.2 TearDown()

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

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

- lib/UAV\_common/src/controllers/[controller\\_test.cpp](#)

## 6.26 ControlSurfaces Class Reference

Aircraft's control surfaces.

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

### Public Member Functions

- [ControlSurfaces](#) ()
- [ControlSurfaces](#) (int noOfSurfaces, Eigen::Matrix< double, 6,-1 > matrix, Eigen::VectorXd min, Eigen::VectorXd max, Eigen::VectorXd trim)
- *Constructor.*
- Eigen::Vector< double, 6 > [getCoefficients](#) () const
- bool [setValues](#) (Eigen::VectorXd new\_values)
- void [restoreTrim](#) ()
- int [getNoOfSurface](#) () const
- Eigen::VectorXd [getValues](#) () const

## 6.26.1 Detailed Description

Aircraft's control surfaces.

## 6.26.2 Constructor & Destructor Documentation

### 6.26.2.1 ControlSurfaces() [1/2]

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

### 6.26.2.2 ControlSurfaces() [2/2]

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

Constructor.

#### Parameters

<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

## 6.26.3 Member Function Documentation

### 6.26.3.1 getCoefficients()

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

### 6.26.3.2 getNoOfSurface()

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

### 6.26.3.3 getValues()

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

### 6.26.3.4 restoreTrim()

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

### 6.26.3.5 setValues()

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

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

- [lib/UAV\\_common/src/components/control\\_surfaces.hpp](#)
- [lib/UAV\\_common/src/components/control\\_surfaces.cpp](#)

## 6.27 ControlSystem Class Reference

Central controller class.

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

### Public Member Functions

- [ControlSystem](#) (zmq::context\_t \*ctx, std::string uav\_address)  
*Constructor.*
- [~ControlSystem](#) ()
- void [run](#) ()  
*Run controller.*
- void [setMode](#) ([ControllerMode](#) new\_mode)  
*Change controller mode.*
- void [exitController](#) ()  
*Stop controller loop.*



## Friends

- class [Control](#)

### 6.27.1 Detailed Description

Central controller class.

### 6.27.2 Constructor & Destructor Documentation

#### 6.27.2.1 ControlSystem()

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

Constructor.

Parameters

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

#### 6.27.2.2 ~ControlSystem()

```
ControlSystem::~~ControlSystem ( )
```

### 6.27.3 Member Function Documentation

#### 6.27.3.1 exitController()

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

Stop controller loop.

### 6.27.3.2 run()

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

Run controller.

### 6.27.3.3 setMode()

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

Change controller mode.

#### Parameters

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

## 6.27.4 Friends And Related Function Documentation

### 6.27.4.1 Control

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

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

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

## 6.28 controllers::DoubleSetpoint Class Reference

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

Inheritance diagram for controllers::DoubleSetpoint:

Collaboration diagram for controllers::DoubleSetpoint:

## Public Member Functions

- [DoubleSetpoint](#) (double high, double mid, double low, double mid\_range, double delta=0.0)  
*Constructor with all Bang-bang controller parameters.*
- [DoubleSetpoint](#) (rapidxml::xml\_node<> \*controller\_node)  
*Construct controller with parameters from xml.*
- double [calc](#) (double desired, double actual, double dt) override  
*calc output of controller with specific time step*
- void [clear](#) () override  
*clear internal state*
- std::unique\_ptr< [Controller](#) > [clone](#) () const override  
*virtual clone method*

## Additional Inherited Members

### 6.28.1 Constructor & Destructor Documentation

#### 6.28.1.1 DoubleSetpoint() [1/2]

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

Constructor with all Bang-bang controller parameters.

##### Parameters

<i>high</i>	output when error is in positive range
<i>mid</i>	output when error is in center range
<i>low</i>	output when error is in negative range
<i>mid_range</i>	size of center field from zero
<i>delta</i>	hysteresis symmetrical to zero

#### 6.28.1.2 DoubleSetpoint() [2/2]

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

Construct controller with parameters from xml.

## Parameters

<i>controller_node</i>	xml node with controller params
------------------------	---------------------------------

## 6.28.2 Member Function Documentation

### 6.28.2.1 calc()

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

calc output of controller with specific time step

## Parameters

<i>desired</i>	input of controller, desired value
<i>actual</i>	measured actual value
<i>dt</i>	time step

## Returns

output of controller

Implements [Controller](#).

### 6.28.2.2 clear()

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

clear internal state

Implements [Controller](#).

### 6.28.2.3 clone()

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

virtual clone method

Implements [Controller](#).

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

- [lib/UAV\\_common/src/controllers/impl/double\\_setpoint.hpp](#)
- [lib/UAV\\_common/src/controllers/impl/double\\_setpoint.cpp](#)

## 6.29 Drive Struct Reference

[Drive](#) propelling aircraft.

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

Inheritance diagram for Drive:

Collaboration diagram for Drive:

### Public Attributes

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

### 6.29.1 Detailed Description

[Drive](#) propelling aircraft.

### 6.29.2 Member Data Documentation

#### 6.29.2.1 axis

[Eigen::Vector3d](#) [Drive::axis](#)

#### 6.29.2.2 hinges

[Hinge](#) [Drive::hinges](#)[2]

#### 6.29.2.3 noOfHinges

int [Drive::noOfHinges](#)

### 6.29.2.4 position

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

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

- [lib/UAV\\_common/src/components/drive.hpp](#)

## 6.30 EKF Class Reference

Extended Kalman Filter.

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

### Public Member Functions

- [EKF](#) ([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.*

### 6.30.1 Detailed Description

Extended Kalman Filter.

### 6.30.2 Constructor & Destructor Documentation

#### 6.30.2.1 EKF()

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

Constructor.

## Parameters

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

### 6.30.3 Member Function Documentation

#### 6.30.3.1 getPos()

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

Returns estimated position vector.

## Returns

position vector in world frame

#### 6.30.3.2 getVel()

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

Returns estimated velocity vector.

## Returns

velocity vector in world frame

#### 6.30.3.3 log()

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

Log filter state.

## Parameters

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

#### 6.30.3.4 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

#### 6.30.3.5 updateBaro()

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

Update phase. Height correction.

##### Parameters

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

#### 6.30.3.6 updateGPS()

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

Update phase. Position correction.

##### Parameters

<i>time</i>	simulation time
<i>baro</i>	<a href="#">GPS</a> location measure

#### 6.30.3.7 updateGPSVel()

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



Update phase. Velocity correction.

#### Parameters

<i>time</i>	simulation time
<i>baro</i>	<a href="#">GPS</a> velocity measure

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

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

## 6.31 EKFPARAMS Struct Reference

EK 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](#)

### 6.31.1 Detailed Description

EK filter parameters.

### 6.31.2 Member Data Documentation

#### 6.31.2.1 P0

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

#### 6.31.2.2 Q

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

### 6.31.2.3 RBaro

```
double EKFPParams::RBaro
```

### 6.31.2.4 RGPSPos

```
Eigen::Matrix3d EKFPParams::RGPSPos
```

### 6.31.2.5 RGPSPVel

```
Eigen::Matrix3d EKFPParams::RGPSPVel
```

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

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

## 6.32 EKFScalers Struct Reference

Scalers for [EKF](#).

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

### Public Attributes

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

### 6.32.1 Detailed Description

Scalers for [EKF](#).

### 6.32.2 Member Data Documentation

**6.32.2.1 baroScaler**

```
double EKFScalers::baroScaler
```

**6.32.2.2 predictScaler**

```
double EKFScalers::predictScaler
```

**6.32.2.3 updateScaler**

```
double EKFScalers::updateScaler
```

**6.32.2.4 zScaler**

```
double EKFScalers::zScaler
```

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

- [lib/UAV\\_common/src/components/navi.hpp](#)

**6.33 Environment Class Reference**

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

**Public Member Functions**

- [Environment](#) (zmq::context\_t \*ctx, std::string uav\_address)  
*Constructor.*
- [~Environment](#) ()  
*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*

## 6.33.1 Constructor & Destructor Documentation

### 6.33.1.1 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

### 6.33.1.2 ~Environment()

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

Deconstructor.

## 6.33.2 Member Function Documentation

### 6.33.2.1 getAngularAcceleraton()

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

Returns exact angular acceleration vector.

#### Returns

angular acceleration vector in body frame

### 6.33.2.2 getAngularVelocity()

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

Returns exact angular velocity vector.

#### Returns

angular velocities vector in body frame

### 6.33.2.3 getLinearAcceleration()

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

Returns exact linear acceleration vector.

#### Returns

linear acceleration vector in body frame

### 6.33.2.4 getLinearVelocity()

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

Returns exact linear velocity vector.

#### Returns

linear velocity vector in body frame

#### 6.33.2.5 getOrientation()

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

Returns exact orientation vector.

##### Returns

orientation vector in world frame

#### 6.33.2.6 getPosition()

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

Returns exact position vector.

##### Returns

position vector in world frame

#### 6.33.2.7 getRnb()

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

Get rotation matrix from world to body frame.

##### Returns

rotation matrix

#### 6.33.2.8 getTime()

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

Returns time of simulation.

##### Returns

simulation time

#### 6.33.2.9 `getWorldAngularVelocity()`

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

Returns exact angular velocity vector.

##### Returns

linear angular vector in world frame

#### 6.33.2.10 `getWorldLinearVelocity()`

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

Returns exact linear velocity vector.

##### Returns

linear velocity vector in world frame

#### 6.33.2.11 `updateSensors()`

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

update all sensors

### 6.33.3 Member Data Documentation

#### 6.33.3.1 `sensors`

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

map of sensors that measure single value

### 6.33.3.2 sensorsVec3d

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

map of sensors that measure values which is 3 element vector

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

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

## 6.34 GPS Class Reference

Representation of [GPS](#) position measure.

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

Inheritance diagram for GPS:

Collaboration diagram for GPS:

### Public Member Functions

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

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

### Additional Inherited Members

#### 6.34.1 Detailed Description

Representation of [GPS](#) position measure.

#### 6.34.2 Constructor & Destructor Documentation

##### 6.34.2.1 GPS()

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



### 6.34.3 Member Function Documentation

#### 6.34.3.1 update()

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

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

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

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

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

## 6.35 GPSVel Class Reference

Representation of [GPS](#) velocity measure.

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

Inheritance diagram for GPSVel:

Collaboration diagram for GPSVel:

### Public Member Functions

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

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

### Additional Inherited Members

#### 6.35.1 Detailed Description

Representation of [GPS](#) velocity measure.

#### 6.35.2 Constructor & Destructor Documentation

### 6.35.2.1 GPSVel()

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

## 6.35.3 Member Function Documentation

### 6.35.3.1 update()

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

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

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

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

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

## 6.36 Gyroscope Class Reference

Representation of gyroscope.

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

Inheritance diagram for Gyroscope:

Collaboration diagram for Gyroscope:

### Public Member Functions

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

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

### Additional Inherited Members

### 6.36.1 Detailed Description

Representation of gyroscope.

## 6.36.2 Constructor & Destructor Documentation

### 6.36.2.1 Gyroscope()

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

## 6.36.3 Member Function Documentation

### 6.36.3.1 update()

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

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

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

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

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

## 6.37 Hinge Class Reference

[Hinge](#) connecting aircraft with drives.

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

### Public Member Functions

- [Hinge](#) ()=default
- [Hinge](#) (Eigen::Vector3d axis, double max, double min, double trim)
- [Hinge](#) (const [Hinge](#) &old)
- [Hinge](#) & [operator=](#) (const [Hinge](#) &old)
- void [updateValue](#) (double newValue)  
*set new angle on hinge*
- const Eigen::Matrix3d [getRot](#) ()  
*Get rotation matrix of orientation change due to hinge.*

### 6.37.1 Detailed Description

[Hinge](#) connecting aircraft with drives.

### 6.37.2 Constructor & Destructor Documentation

#### 6.37.2.1 Hinge() [1/3]

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

#### 6.37.2.2 Hinge() [2/3]

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

#### 6.37.2.3 Hinge() [3/3]

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

### 6.37.3 Member Function Documentation

#### 6.37.3.1 getRot()

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

Get rotation matrix of orientation change due to hinge.

##### Returns

rotation matrix

### 6.37.3.2 operator=()

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

### 6.37.3.3 updateValue()

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

set new angle on hinge

#### Parameters

<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](#)

## 6.38 Jet Class Reference

[Jet](#) rocket engine.

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

Inheritance diagram for Jet:

Collaboration diagram for Jet:

### Public Member Functions

- bool [start](#) (double [time](#))  
*start jet engine*
- double [getThrust](#) (double [time](#))  
*get thrust in specific time*
- double [getLastThrust](#) ()  
*get last calculated thrust*

### Public Attributes

- int [phases](#)
- Eigen::VectorXd [thrust](#)
- Eigen::VectorXd [time](#)

#### 6.38.1 Detailed Description

[Jet](#) rocket engine.

#### 6.38.2 Member Function Documentation

### 6.38.2.1 getLastThrust()

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

get last calculated thrust

#### Returns

last calculated thrust

### 6.38.2.2 getThrust()

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

get thrust in specific time

#### Parameters

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

#### Returns

thrust value in Newtons

### 6.38.2.3 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

## 6.38.3 Member Data Documentation

### 6.38.3.1 phases

```
int Jet::phases
```

### 6.38.3.2 thrust

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

### 6.38.3.3 time

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

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

- [lib/UAV\\_common/src/components/drive.hpp](#)
- [lib/UAV\\_common/src/components/drive.cpp](#)

## 6.39 Load Class Reference

[Load](#) of aircraft that can be 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 [getAmmount](#) ()  
*get ammount of load*
- int [release](#) (double time)  
*Try to release load.*

### Protected Member Functions

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



### 6.39.1 Detailed Description

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

### 6.39.2 Constructor & Destructor Documentation

#### 6.39.2.1 Load() [1/2]

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

#### 6.39.2.2 Load() [2/2]

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

### 6.39.3 Member Function Documentation

#### 6.39.3.1 getAmmount()

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

get ammount of load

##### Returns

ammount

#### 6.39.3.2 getMass()

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

get mass of load

##### Returns

mass

### 6.39.3.3 getOffset()

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

get offset of load

#### Returns

offset vector

### 6.39.3.4 operator=()

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

### 6.39.3.5 release()

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

Try to release load.

#### Parameters

<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](#)

## 6.40 Logger Class Reference

Log vector data with timestamp in file.

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

## Public Member Functions

- [Logger](#) (std::string path, std::string fmt="", uint8\_t group=0)  
*Constructor.*
- [~Logger](#) ()  
*Destructor.*
- void [setFmt](#) (std::string fmt)  
*Set new format if was not known in constructor.*
- void [log](#) (double time, std::initializer\_list< Eigen::VectorXd > args)  
*Log one row.*
- void [log](#) (double time, std::initializer\_list< double > args)  
*Log one row.*

## Static Public Member Functions

- static void [setLogDirectory](#) (std::string subdirectory)  
*Set global path that log should be created at. Path will be added to relative path of specific log instance.*

### 6.40.1 Detailed Description

Log vector data with timestamp in file.

### 6.40.2 Constructor & Destructor Documentation

#### 6.40.2.1 Logger()

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

Constructor.

##### Parameters

<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>

#### 6.40.2.2 ~Logger()

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

Destructor.

### 6.40.3 Member Function Documentation

#### 6.40.3.1 `log()` [1/2]

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

Log one row.

##### Parameters

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

#### 6.40.3.2 `log()` [2/2]

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

Log one row.

##### Parameters

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

#### 6.40.3.3 `setFmt()`

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

Set new format if was not known in constructor.

##### Parameters

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

#### 6.40.3.4 setLogDirectory()

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

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

##### Parameters

<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](#)

## 6.41 Magnetometer Class Reference

Representation of magnetometer.

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

Inheritance diagram for Magnetometer:

Collaboration diagram for Magnetometer:

### Public Member Functions

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

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

### Static Public Attributes

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

### Additional Inherited Members

#### 6.41.1 Detailed Description

Representation of magnetometer.

#### 6.41.2 Constructor & Destructor Documentation

### 6.41.2.1 Magnetometer()

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

## 6.41.3 Member Function Documentation

### 6.41.3.1 update()

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

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

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

## 6.41.4 Member Data Documentation

### 6.41.4.1 mag

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

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

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

## 6.42 NS Class Reference

Navigation system.

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

## Public Member Functions

- [NS](#) ([Environment](#) &env)  
*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](#).*
- Eigen::Matrix3d [getRotationMatrixBodyToWorld](#) ()  
*Returns rotation matrix from body to world frame.*

### 6.42.1 Detailed Description

Navigation system.

### 6.42.2 Constructor & Destructor Documentation

#### 6.42.2.1 NS()

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

Constructor.

Parameters

<i>env</i>	reference to environment, that <a href="#">NS</a> navigate through
------------	--

#### 6.42.2.2 ~NS()

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

Destructor.

### 6.42.3 Member Function Documentation

#### 6.42.3.1 getAngularVelocity()

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

Returns rates estimated by [NS](#).

##### Returns

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

#### 6.42.3.2 getLinearVelocity()

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

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

##### Returns

linear velocity vector in world frame

#### 6.42.3.3 getOrientation()

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

Returns orientation estimated by [NS](#).

##### Returns

orientation vector (RPY) in world frame

#### 6.42.3.4 getPosition()

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

Returns position estimated by [NS](#).

##### Returns

position vector in world frame



### 6.42.3.5 getRotationMatrixBodyToWorld()

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

Returns rotation matrix from body to world frame.

#### Returns

rotation matrix

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

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

## 6.43 ODE Class Reference

Ordinal differential equation solver.

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

Inheritance diagram for ODE:

### Public Types

- enum [ODEMethod](#) {  
[Euler](#) , [Heun](#) , [RK4](#) , [PC2](#) ,  
[PC4](#) , [NONE](#) }  
*Supported solving method.*

### Public Member Functions

- [ODE](#) (int micro\_steps)  
*Constructor.*
- virtual [~ODE](#) ()  
*Virtual 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.*

### 6.43.1 Detailed Description

Ordinal differencial equation solver.

### 6.43.2 Member Enumeration Documentation

#### 6.43.2.1 ODEMethod

```
enum ODE::ODEMethod
```

Supported solving method.

Enumerator

Euler	
Heun	
RK4	
PC2	
PC4	
NONE	

### 6.43.3 Constructor & Destructor Documentation

#### 6.43.3.1 ODE()

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

Constructor.

#### 6.43.3.2 ~ODE()

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

Virtual deconstructor.

### 6.43.4 Member Function Documentation

#### 6.43.4.1 factory()

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

Factory constructing ODE solvers.

**Parameters**

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

**Returns**

instance of [ODE](#) solver

**6.43.4.2 fromString()**

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

Parse solving method from string.

**Parameters**

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

**Returns**

solving method if parsed, NONE if unknown

**6.43.4.3 getMicrosteps() [1/2]**

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

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

**Returns**

microsteps

**6.43.4.4 getMicrosteps() [2/2]**

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

Get microsteps of given method.

## Parameters

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

## Returns

number of microstep in one algorithm step

**6.43.4.5 step()**

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

One step of explicit solving algorithm.

## Parameters

<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\\_PC4](#), [ODE\\_PC2](#), [ODE\\_RK4](#), [ODE\\_Heun](#), and [ODE\\_Euler](#).

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

- [lib/UAV\\_common/src/ode/ode.hpp](#)
- [lib/UAV\\_common/src/ode/ode.cpp](#)

**6.44 ODE\_Euler Class Reference**

Explicit Euler algorithm.

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

Inheritance diagram for ODE\_Euler:

Collaboration diagram for ODE\_Euler:

## Public Member Functions

- [ODE\\_Euler](#) ()
- `Eigen::VectorXd` [step](#) (double t, `Eigen::VectorXd` y0, `std::function< Eigen::VectorXd(double, Eigen::VectorXd)>` rhs\_fun, double h) override  
*One step of explicit solving algorithm.*

## Additional Inherited Members

### 6.44.1 Detailed Description

Explicit Euler algorithm.

### 6.44.2 Constructor & Destructor Documentation

#### 6.44.2.1 ODE\_Euler()

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

### 6.44.3 Member Function Documentation

#### 6.44.3.1 step()

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

One step of explicit solving algorithm.

#### Parameters

<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`

## 6.45 ODE\_Heun Class Reference

Second order explicit Heun algorithm.

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

Inheritance diagram for ODE\_Heun:

Collaboration diagram for ODE\_Heun:

### Public Member Functions

- [ODE\\_Heun](#) ()
- `Eigen::VectorXd step (double t, Eigen::VectorXd y0, std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun, double h) override`  
*One step of explicit solving algorithm.*

### Additional Inherited Members

#### 6.45.1 Detailed Description

Second order explicit Heun algorithm.

#### 6.45.2 Constructor & Destructor Documentation

##### 6.45.2.1 ODE\_Heun()

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

#### 6.45.3 Member Function Documentation

##### 6.45.3.1 step()

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

One step of explicit solving algorithm.

## Parameters

$t$	start time
$y0$	start variable
$rhs\_fun$	right-hand-side function, calculation of derivative
$h$	time step

## Returns

Implements [ODE](#).

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

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

## 6.46 ODE\_PC2 Class Reference

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

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

Inheritance diagram for ODE\_PC2:

Collaboration diagram for ODE\_PC2:

### Public Member Functions

- [ODE\\_PC2](#) ()
- `Eigen::VectorXd step (double t, Eigen::VectorXd y0, std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun, double h) override`  
*One step of explicit solving algorithm.*

### Additional Inherited Members

#### 6.46.1 Detailed Description

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

#### 6.46.2 Constructor & Destructor Documentation



### 6.46.2.1 ODE\_PC2()

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

## 6.46.3 Member Function Documentation

### 6.46.3.1 step()

```
Eigen::VectorXd ODE_PC2::step (
    double t,
    Eigen::VectorXd y0,
    std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun,
    double h ) [inline], [override], [virtual]
```

One step of explicit solving algorithm.

#### Parameters

<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](#)

## 6.47 ODE\_PC4 Class Reference

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

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

Inheritance diagram for ODE\_PC4:

Collaboration diagram for ODE\_PC4:

## Public Member Functions

- [ODE\\_PC4](#) ()
- Eigen::VectorXd [step](#) (double t, Eigen::VectorXd y0, std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs\_fun, double h) override

*One step of explicit solving algorithm.*

## Additional Inherited Members

### 6.47.1 Detailed Description

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

### 6.47.2 Constructor & Destructor Documentation

#### 6.47.2.1 ODE\_PC4()

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

### 6.47.3 Member Function Documentation

#### 6.47.3.1 step()

```
Eigen::VectorXd ODE_PC4::step (
    double t,
    Eigen::VectorXd y0,
    std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun,
    double h ) [inline], [override], [virtual]
```

One step of explicit solving algorithm.

#### Parameters

<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`

## 6.48 ODE\_RK4 Class Reference

Fourth order Runge Kutta algorithm.

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

Inheritance diagram for ODE\_RK4:

Collaboration diagram for ODE\_RK4:

### Public Member Functions

- [ODE\\_RK4](#) ()
- `Eigen::VectorXd step (double t, Eigen::VectorXd y0, std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun, double h) override`  
*One step of explicit solving algorithm.*

### Additional Inherited Members

#### 6.48.1 Detailed Description

Fourth order Runge Kutta algorithm.

#### 6.48.2 Constructor & Destructor Documentation

##### 6.48.2.1 ODE\_RK4()

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

#### 6.48.3 Member Function Documentation

##### 6.48.3.1 step()

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

One step of explicit solving algorithm.

**Parameters**

<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](#)

## 6.49 ODETest Class Reference

Inheritance diagram for ODETest:

Collaboration diagram for ODETest:

### Protected Member Functions

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

#### 6.49.1 Member Function Documentation

##### 6.49.1.1 SetUp()

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

##### 6.49.1.2 TearDown()

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

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

- [lib/UAV\\_common/src/ode/ode\\_test.cpp](#)

## 6.50 Params Class Reference

Simulation parameters.

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

### Public Member Functions

- [Params](#) ()  
*Constructor.*
- [Params](#) (const [Params](#) &)=delete
- [Params](#) & [operator=](#) (const [Params](#) &)=delete
- [Params](#) ([Params](#) &&)=delete
- [~Params](#) ()  
*Destructor.*

### Static Public Member Functions

- static const [Params](#) \* [getSingleton](#) ()  
*Get singleton of [Params](#).*

### Public Attributes

- double [STEP\\_TIME](#)  
*Step time of simulation. Step of [ODE](#) solving methods.*

#### 6.50.1 Detailed Description

Simulation parameters.

#### 6.50.2 Constructor & Destructor Documentation

##### 6.50.2.1 Params() [1/3]

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

Constructor.

### 6.50.2.2 Params() [2/3]

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

### 6.50.2.3 Params() [3/3]

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

### 6.50.2.4 ~Params()

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

Deconstructor.

## 6.50.3 Member Function Documentation

### 6.50.3.1 getSingleton()

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

Get singleton of [Params](#).

#### Returns

const pointer to [Params](#) instance. Return nullptr if not initialized

### 6.50.3.2 operator=()

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

## 6.50.4 Member Data Documentation

### 6.50.4.1 STEP\_TIME

```
double Params::STEP_TIME
```

Step time of simulation. Step of [ODE](#) solving methods.

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

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

## 6.51 controllers::PID Class Reference

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

Inheritance diagram for controllers::PID:

Collaboration diagram for controllers::PID:

### Public Types

- enum class [AntiWindUpMode](#) { [NONE](#) , [CLAMPING](#) }  
*Methods of handling windup in controller.*

### Public Member Functions

- [PID](#) (double Kp, double Ki, double Kd, double Kff=0.0, double min=-std::numeric\_limits< double >::max(), double max=std::numeric\_limits< double >::max(), [AntiWindUpMode](#) antiWindUp=[AntiWindUpMode::CLAMPING](#))  
*Constructor with all [PID](#) controller parameters.*
- [PID](#) (rapidxml::xml\_node<> \*controller\_node)  
*Construct controller with parameters from xml.*
- double [calc](#) (double desired, double actual, double dt) override  
*calc output of controller with specific time step*
- void [clear](#) () override  
*clear internal state*
- std::unique\_ptr< [Controller](#) > [clone](#) () const override  
*virtual clone method*

### Additional Inherited Members

#### 6.51.1 Member Enumeration Documentation

##### 6.51.1.1 AntiWindUpMode

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

Methods of handling windup in controller.

## Enumerator

NONE	
CLAMPING	

## 6.51.2 Constructor & Destructor Documentation

### 6.51.2.1 PID() [1/2]

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

Constructor with all [PID](#) controller parameters.

## Parameters

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

### 6.51.2.2 PID() [2/2]

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

Construct controller with parameters from xml.

## Parameters

<i>controller_node</i>	xml node with controller params
------------------------	---------------------------------



### 6.51.3 Member Function Documentation

#### 6.51.3.1 calc()

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

calc output of controller with specific time step

##### Parameters

<i>desired</i>	input of controller, desired value
<i>actual</i>	measured actual value
<i>dt</i>	time step

##### Returns

output of controller

Implements [Controller](#).

#### 6.51.3.2 clear()

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

clear internal state

Implements [Controller](#).

#### 6.51.3.3 clone()

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

virtual clone method

Implements [Controller](#).

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

- lib/UAV\_common/src/controllers/impl/[PID.hpp](#)
- lib/UAV\_common/src/controllers/impl/[PID.cpp](#)

## 6.52 controllers::PID\_Discrete Class Reference

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

Inheritance diagram for controllers::PID\_Discrete:

Collaboration diagram for controllers::PID\_Discrete:

### Public Member Functions

- [PID\\_Discrete](#) (double Kp, double Ki, double Kd, double Kff=0.0, double N=100.0, double min=-std::numeric\_limits< double >::max(), double max=std::numeric\_limits< double >::max())  
*Constructor with all [PID](#) controller parameters.*
- [PID\\_Discrete](#) (rapidxml::xml\_node<> \*controller\_node)  
*Construct controller with parameters from xml.*
- double [calc](#) (double desired, double actual, double dt) override  
*calc output of controller with specific time step*
- void [set\\_dt](#) (double dt) override  
*Set new time step.*
- void [clear](#) () override  
*clear internal state*
- std::unique\_ptr< [Controller](#) > [clone](#) () const override  
*virtual clone method*

### Additional Inherited Members

#### 6.52.1 Constructor & Destructor Documentation

##### 6.52.1.1 PID\_Discrete() [1/2]

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

Constructor with all [PID](#) controller parameters.

#### Parameters

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

### 6.52.1.2 PID\_Discrete() [2/2]

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

Construct controller with parameters from xml.

#### Parameters

<i>controller_node</i>	xml node with controller params
------------------------	---------------------------------

## 6.52.2 Member Function Documentation

### 6.52.2.1 calc()

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

calc output of controller with specific time step

#### Parameters

<i>desired</i>	input of controller, desired value
<i>actual</i>	measured actual value
<i>dt</i>	time step

#### Returns

output of controller

Implements [Controller](#).

### 6.52.2.2 clear()

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

clear internal state

Implements [Controller](#).

### 6.52.2.3 clone()

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

virtual clone method

Implements [Controller](#).

### 6.52.2.4 set\_dt()

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

Set new time step.

Parameters

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

Reimplemented from [Controller](#).

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

- [lib/UAV\\_common/src/controllers/impl/PID\\_discrete.hpp](#)
- [lib/UAV\\_common/src/controllers/impl/PID\\_discrete.cpp](#)

## 6.53 Rotor Struct Reference

[Rotor](#) engine with controlled speed.

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

Inheritance diagram for Rotor:

Collaboration diagram for Rotor:

### Public Attributes

- double [forceCoff](#)
- double [torqueCoff](#)
- int [direction](#)
- double [timeConstant](#)
- double [maxSpeed](#)
- double [hoverSpeed](#)

### 6.53.1 Detailed Description

[Rotor](#) engine with controlled speed.

### 6.53.2 Member Data Documentation

#### 6.53.2.1 direction

```
int Rotor::direction
```

#### 6.53.2.2 forceCoff

```
double Rotor::forceCoff
```

#### 6.53.2.3 hoverSpeed

```
double Rotor::hoverSpeed
```

#### 6.53.2.4 maxSpeed

```
double Rotor::maxSpeed
```

#### 6.53.2.5 timeConstant

```
double Rotor::timeConstant
```

#### 6.53.2.6 torqueCoff

```
double Rotor::torqueCoff
```

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

- [lib/UAV\\_common/src/components/drive.hpp](#)

## 6.54 Sensor< T > Class Template Reference

Sensors base class.

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

Collaboration diagram for Sensor< T >:

### Public Member Functions

- [Sensor](#) ([Environment](#) &[env](#), double [sd](#), T [bias](#), std::string [path](#), std::string [fmt](#), double [refreshTime](#))  
*Constructor.*
- virtual void [update](#) ()=0  
*Update sensor state. Measured value is updated if sensor is ready for next read.*
- T [getReading](#) ()  
*Returns recent measure.*
- double [getSd](#) ()  
*Returns standard deviation.*
- bool [isReady](#) ()  
*Checks if sensor is ready.*

### Protected Member Functions

- bool [shouldUpdate](#) ()  
*Checks if sensor should measure next value.*
- double [error](#) ()

### Protected Attributes

- [Environment](#) & [env](#)
- T [value](#)
- double [refreshTime](#)
- double [lastUpdate](#)
- std::atomic\_bool [ready](#)
- std::normal\_distribution< double > [dist](#)
- T [bias](#)
- [Logger](#) [logger](#)

### Static Protected Attributes

- static std::mt19937 [gen](#) = std::mt19937(std::random\_device())()

#### 6.54.1 Detailed Description

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

Sensors base class.

## Template Parameters

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

## 6.54.2 Constructor &amp; Destructor Documentation

## 6.54.2.1 Sensor()

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

Constructor.

## Parameters

<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

## 6.54.3 Member Function Documentation

## 6.54.3.1 error()

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

## 6.54.3.2 getReading()

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

Returns recent measure.

**Returns**

sensor measure

**6.54.3.3 getSd()**

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

Returns standard deviation.

**Returns**

standard deviation

**6.54.3.4 isReady()**

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

Checks if sensor is ready.

**Returns**

true if sensor is ready

**6.54.3.5 shouldUpdate()**

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

Checks if sensor should measure next value.

**Returns**

true if sensor is ready for next measure

**6.54.3.6 update()**

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

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

Implemented in [GPSVel](#), [GPS](#), [Barometer](#), [Magnetometer](#), [Gyroscope](#), and [Accelerometer](#).



## 6.54.4 Member Data Documentation

### 6.54.4.1 bias

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

### 6.54.4.2 dist

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

### 6.54.4.3 env

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

### 6.54.4.4 gen

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

### 6.54.4.5 lastUpdate

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

### 6.54.4.6 logger

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

#### 6.54.4.7 ready

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

#### 6.54.4.8 refreshTime

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

#### 6.54.4.9 value

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

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

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

## 6.55 SensorParams Struct Reference

Base parameters of a sensor.

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

### Public Attributes

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

#### 6.55.1 Detailed Description

Base parameters of a sensor.

#### 6.55.2 Member Data Documentation

### 6.55.2.1 bias

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

### 6.55.2.2 name

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

### 6.55.2.3 refreshTime

```
double SensorParams::refreshTime
```

### 6.55.2.4 sd

```
double SensorParams::sd
```

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

- [lib/UAV\\_common/src/components/navi.hpp](#)

## 6.56 TimedLoop Class Reference

Simulation of real-time synchronized loop.

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

### Public Member Functions

- [TimedLoop](#) (int periodInMs, std::function< void(void)> func, [Status](#) &status)  
*Constructor.*
- void [go](#) ()  
*start infinite loop*
- void [go](#) (uint32\_t loops)  
*start loop for specific cycle numbers*

### 6.56.1 Detailed Description

Simulation of real-time synchronized loop.

## 6.56.2 Constructor & Destructor Documentation

### 6.56.2.1 TimedLoop()

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

Constructor.

## Parameters

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

### 6.56.3 Member Function Documentation

#### 6.56.3.1 go() [1/2]

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

start infinite loop

#### 6.56.3.2 go() [2/2]

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

start loop for specific cycle numbers

## Parameters

<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](#)

## 6.57 UAVparams Struct Reference

Parsed UAV configuration from XML.

```
#include <uav_params.hpp>
```

Collaboration diagram for UAVparams:

## Public Member Functions

- [UAVparams](#) ()  
*Initialize default data.*
- [~UAVparams](#) ()
- void [loadConfig](#) (std::string configFile)
- Eigen::VectorXd [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](#)
- Eigen::Vector3d [target](#)
- 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, std::unique\_ptr< [Controller](#) > > [controllers](#)
- 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](#)

### 6.57.1 Detailed Description

Parsed UAV configuration from XML.

## 6.57.2 Constructor & Destructor Documentation

### 6.57.2.1 UAVparams()

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

Initialize default data.

### 6.57.2.2 ~UAVparams()

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

## 6.57.3 Member Function Documentation

### 6.57.3.1 getRotorHoverSpeeds()

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

### 6.57.3.2 getRotorMaxSpeeds()

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

### 6.57.3.3 getRotorTimeContants()

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

### 6.57.3.4 getSingleton()

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

### 6.57.3.5 loadConfig()

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

## 6.57.4 Member Data Documentation

### 6.57.4.1 aero\_coffs

[AeroCoefficients](#) UAVparams::aero\_coffs

### 6.57.4.2 ahrs

[AHRSParams](#) UAVparams::ahrs

### 6.57.4.3 ammo

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

### 6.57.4.4 cargo

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

### 6.57.4.5 controllers

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

### 6.57.4.6 ekf

[EKFScalers](#) UAVparams::ekf



#### 6.57.4.7 initialMode

`std::string UAVparams::initialMode`

#### 6.57.4.8 initialOrientation

`Eigen::Vector3d UAVparams::initialOrientation`

#### 6.57.4.9 initialPosition

`Eigen::Vector3d UAVparams::initialPosition`

#### 6.57.4.10 initialVelocity

`Eigen::Vector3d UAVparams::initialVelocity`

#### 6.57.4.11 instantRun

`bool UAVparams::instantRun`

#### 6.57.4.12 Ix

`double UAVparams::Ix`

#### 6.57.4.13 Ixy

`double UAVparams::Ixy`

#### 6.57.4.14 Ixz

`double UAVparams::Ixz`

**6.57.4.15 ly**

```
double UAVparams::Iy
```

**6.57.4.16 Iyz**

```
double UAVparams::Iyz
```

**6.57.4.17 Iz**

```
double UAVparams::Iz
```

**6.57.4.18 jets**

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

**6.57.4.19 m**

```
double UAVparams::m
```

**6.57.4.20 name**

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

**6.57.4.21 noOfAmmo**

```
int UAVparams::noOfAmmo
```

**6.57.4.22 noOfCargo**

```
int UAVparams::noOfCargo
```

#### 6.57.4.23 noOfJets

```
int UAVparams::noOfJets
```

#### 6.57.4.24 noOfRotors

```
int UAVparams::noOfRotors
```

#### 6.57.4.25 rotorMixer

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

#### 6.57.4.26 rotors

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

#### 6.57.4.27 sensors

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

#### 6.57.4.28 surfaceMixer

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

#### 6.57.4.29 surfaces

```
ControlSurfaces UAVparams::surfaces
```

### 6.57.4.30 target

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

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

- [lib/UAV\\_common/src/parser/uav\\_params.hpp](#)
- [lib/UAV\\_common/src/parser/uav\\_params.cpp](#)

## 6.58 controllers::ZTransform Class Reference

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

Inheritance diagram for controllers::ZTransform:

Collaboration diagram for controllers::ZTransform:

### Public Member Functions

- [ZTransform](#) (const std::vector< double > &num, const std::vector< double > &den, double min=-std::numeric\_limits< double >::max(), double max=std::numeric\_limits< double >::max())  
*Constructor of Z-Transform controller.*
- [ZTransform](#) (rapidxml::xml\_node<> \*controller\_node)  
*Construct controller with parameters from xml.*
- double [calc](#) (double desired, double actual, [[maybe\_unused]] double dt) override  
*calc output of controller*
- void [clear](#) () override  
*clear internal state*
- std::unique\_ptr< [Controller](#) > [clone](#) () const override  
*virtual clone method*

### Additional Inherited Members

#### 6.58.1 Constructor & Destructor Documentation

##### 6.58.1.1 ZTransform() [1/2]

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

Constructor of Z-Transform controller.

## Parameters

<i>min</i>	saturation - lower range limit
<i>max</i>	saturation - upper range limit

**6.58.1.2 ZTransform()** [2/2]

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

Construct controller with parameters from xml.

## Parameters

<i>controller_node</i>	xml node with controller params
------------------------	---------------------------------

**6.58.2 Member Function Documentation****6.58.2.1 calc()**

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

calc output of controller

## Parameters

<i>desired</i>	input of controller, desired value
<i>actual</i>	measured actual value

## Returns

output of controller

**6.58.2.2 clear()**

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

clear internal state

Implements [Controller](#).

### 6.58.2.3 clone()

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

virtual clone method

Implements [Controller](#).

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

- [lib/UAV\\_common/src/controllers/impl/z\\_trans.hpp](#)
- [lib/UAV\\_common/src/controllers/impl/z\\_trans.cpp](#)

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

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

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

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

### Public Member Functions

- [ZTransformStatic](#) (const std::array< double, N > &num, const std::array< double, D > &den, double min=-std::numeric\_limits< double >::max(), double max=std::numeric\_limits< double >::max())  
*Constructor of Z-Transform controller.*
- [ZTransformStatic](#) (rapidxml::xml\_node<> \*controller\_node)=delete  
*Construct controller with parameters from xml.*
- double [calc](#) (double desired, double actual, [[maybe\_unused]] double dt) override  
*calc output of controller*
- void [clear](#) () override  
*clear internal state*
- std::unique\_ptr< [Controller](#) > [clone](#) () const override  
*virtual clone method*

### Additional Inherited Members

#### 6.59.1 Constructor & Destructor Documentation

##### 6.59.1.1 ZTransformStatic() [1/2]

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

Constructor of Z-Transform controller.

## Parameters

<i>min</i>	saturation - lower range limit
<i>max</i>	saturation - upper range limit

## 6.59.1.2 ZTransformStatic() [2/2]

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

Construct controller with parameters from xml.

## Parameters

<i>controller_node</i>	xml node with controller params
------------------------	---------------------------------

## 6.59.2 Member Function Documentation

## 6.59.2.1 calc()

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

calc output of controller

## Parameters

<i>desired</i>	input of controller, desired value
<i>actual</i>	measured actual value

## Returns

output of controller

## 6.59.2.2 clear()

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

clear internal state

Implements [Controller](#).

### 6.59.2.3 clone()

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

virtual clone method

Implements [Controller](#).

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

- [lib/UAV\\_common/src/controllers/impl/z\\_trans.hpp](#)



## Chapter 7

# File Documentation

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

#### Macros

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

#### Functions

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

#### Variables

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

#### 7.1.1 Macro Definition Documentation

#### 7.1.1.1 \_\_has\_include

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

#### 7.1.1.2 ARCHITECTURE\_ID

```
#define ARCHITECTURE_ID
```

#### 7.1.1.3 C\_VERSION

```
#define C_VERSION
```

#### 7.1.1.4 COMPILER\_ID

```
#define COMPILER_ID ""
```

#### 7.1.1.5 DEC

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

##### Value:

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

#### 7.1.1.6 HEX

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

##### Value:

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

### 7.1.1.7 PLATFORM\_ID

```
#define PLATFORM_ID
```

### 7.1.1.8 STRINGIFY

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

### 7.1.1.9 STRINGIFY\_HELPER

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

## 7.1.2 Function Documentation

### 7.1.2.1 main()

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

## 7.1.3 Variable Documentation

### 7.1.3.1 info\_arch

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

### 7.1.3.2 info\_compiler

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

### 7.1.3.3 info\_language\_extensions\_default

```
const char* info_language_extensions_default
```

**Initial value:**

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

### 7.1.3.4 info\_language\_standard\_default

```
const char* info_language_standard_default
```

**Initial value:**

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

### 7.1.3.5 info\_platform

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

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

### Macros

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

### Functions

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

### Variables

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

## 7.2.1 Macro Definition Documentation

### 7.2.1.1 `__has_include`

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

### 7.2.1.2 `ARCHITECTURE_ID`

```
#define ARCHITECTURE_ID
```

### 7.2.1.3 `COMPILER_ID`

```
#define COMPILER_ID ""
```

### 7.2.1.4 `CXX_STD`

```
#define CXX_STD __cplusplus
```

### 7.2.1.5 `DEC`

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

#### Value:

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

### 7.2.1.6 HEX

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

**Value:**

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

### 7.2.1.7 PLATFORM\_ID

```
#define PLATFORM_ID
```

### 7.2.1.8 STRINGIFY

```
#define STRINGIFY(  
    X ) STRINGIFY\_HELPER(X)
```

### 7.2.1.9 STRINGIFY\_HELPER

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

## 7.2.2 Function Documentation

### 7.2.2.1 main()

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

## 7.2.3 Variable Documentation

### 7.2.3.1 info\_arch

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

### 7.2.3.2 info\_compiler

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

### 7.2.3.3 info\_language\_extensions\_default

```
const char* info_language_extensions_default
```

**Initial value:**

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

### 7.2.3.4 info\_language\_standard\_default

```
const char* info_language_standard_default
```

**Initial value:**

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

### 7.2.3.5 info\_platform

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





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

### 7.4 build/CMakeFiles/controller.dir/src/communication/control\_↔recv.cpp.o.d File Reference

### 7.5 build/CMakeFiles/controller.dir/src/communication/control\_↔send.cpp.o.d File Reference

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

### 7.7 build/lib/UAV\_common/CMake↔Files/common.dir/src/controllers/controller.cpp.o.d File Reference

### 7.8 build/CMakeFiles/controller.dir/src/controller/controller\_loop.cpp.o.d File Reference

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

### 7.10 build/CMakeFiles/controller.dir/src/controller/modes/controller\_↔loop\_FACRO.cpp.o.d File Reference

### 7.11 build/CMakeFiles/controller.dir/src/controller/modes/controller\_↔loop\_FANGLE.cpp.o.d File Reference

### 7.12 build/CMakeFiles/controller.dir/src/controller/modes/controller\_↔loop\_FMANUAL.cpp.o.d File Reference

### 7.13 build/CMakeFiles/controller.dir/src/controller/modes/controller\_↔loop\_NONE.cpp.o.d File Reference

### 7.14 build/CMakeFiles/controller.dir/src/controller/modes/controller\_↔loop\_QACRO.cpp.o.d File Reference

### 7.15 build/CMakeFiles/controller.dir/src/controller/modes/controller\_↔loop\_QANGLE.cpp.o.d File Reference

### 7.16 build/CMakeFiles/controller.dir/src/controller/modes/controller\_↔loop\_QPOS.cpp.o.d File Reference

### 7.17 build/CMakeFiles/controller.dir/src/controller/modes/controller\_↔loop\_RANGLE.cpp.o.d File Reference

### 7.18 build/CMakeFiles/controller.dir/src/controller/modes/controller\_↔loop\_RPOS.cpp.o.d File Reference

```
#include "../src/ode/ode.hpp"
#include "../src/controllers/controller.hpp"
#include "../src/timed_loop/timed_loop.hpp"
#include "../src/timed_loop/status.hpp"
#include "../src/parser/parser.hpp"
#include "../src/parser/uav_params.hpp"
#include "../src/components/components.hpp"
```

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

## 7.47 lib/UAV\_common/scripts/controller\_plots.m File Reference

### Functions

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

### Variables

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

### 7.47.1 Function Documentation

#### 7.47.1.1 `legend()`

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

### 7.47.1.2 plot()

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

### 7.47.1.3 title()

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

### 7.47.1.4 xlabel()

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

### 7.47.1.5 ylabel()

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

## 7.47.2 Variable Documentation

### 7.47.2.1 clc

```
clc
```

### 7.47.2.2 csvFiles

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

#### 7.47.2.3 data

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

#### 7.47.2.4 figure

```
figure
```

#### 7.47.2.5 folderPath

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

#### 7.47.2.6 i

```
for i
```

##### Initial value:

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

#### 7.47.2.7 off

```
hold off
```

#### 7.47.2.8 on

```
hold on
```

#### 7.47.2.9 x

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

#### 7.47.2.10 y

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

## 7.48 lib/UAV\_common/src/components/aero\_coefficients.hpp File Reference

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

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

### Classes

- struct [AeroCoefficients](#)  
*Aerodynamic coefficient.*

## 7.49 lib/UAV\_common/src/components/components.hpp File Reference

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

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

## 7.50 lib/UAV\_common/src/components/control\_surfaces.cpp File Reference

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

Include dependency graph for control\_surfaces.cpp:

## 7.51 lib/UAV\_common/src/components/control\_surfaces.hpp File Reference

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

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

### Classes

- class [ControlSurfaces](#)  
*Aircraft's control surfaces.*

## 7.52 lib/UAU\_common/src/components/drive.cpp File Reference

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

Include dependency graph for drive.cpp:

## 7.53 lib/UAU\_common/src/components/drive.hpp File Reference

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

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

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

### Classes

- struct [Drive](#)  
*Drive* propelling aircraft.
- struct [Rotor](#)  
*Rotor* engine with controlled speed.
- class [Jet](#)  
*Jet* rocket engine.

## 7.54 lib/UAU\_common/src/components/hinge.cpp File Reference

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

Include dependency graph for hinge.cpp:

### Functions

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

### 7.54.1 Function Documentation

#### 7.54.1.1 asSkewMatrix()

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

## 7.55 lib/UAU\_common/src/components/hinge.hpp File Reference

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

```
#include <mutex>
```

```
#include <memory>
```

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

## Classes

- class [Hinge](#)  
*Hinge connecting aircraft with drives.*

## 7.56 lib/UAV\_common/src/components/loads.cpp File Reference

```
#include "loads.hpp"
#include <limits>
Include dependency graph for loads.cpp:
```

## 7.57 lib/UAV\_common/src/components/loads.hpp File Reference

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

## Classes

- class [Load](#)  
*Load of aircraft that can be dropped or launched.*
- class [Ammo](#)
- class [Cargo](#)

## 7.58 lib/UAV\_common/src/components/navi.hpp File Reference

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

## Classes

- struct [SensorParams](#)  
*Base parameters of a sensor.*
- struct [AHRSParams](#)  
*AHRS parameters.*
- struct [EKFScalers](#)  
*Scalers for EKF.*

## 7.59 lib/UAV\_common/src/controllers/controller.cpp File Reference

```
#include "controller.hpp"
#include "impl/PID.hpp"
#include "impl/PID_discrete.hpp"
#include "impl/bang_bang.hpp"
#include "impl/double_setpoint.hpp"
#include "impl/z_trans.hpp"
#include <cstring>
#include <stdexcept>
Include dependency graph for controller.cpp:
```

## 7.60 src/controller/controller.cpp File Reference

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

## 7.61 lib/UAV\_common/src/controllers/controller.hpp File Reference

```
#include <memory>
#include "rapidxml/rapidxml.hpp"
Include dependency graph for controller.hpp: This graph shows which files directly or indirectly include this file:
```

### Classes

- class [Controller](#)

## 7.62 src/controller/controller.hpp File Reference

```
#include <map>
#include <string>
#include <Eigen/Dense>
#include <functional>
#include <optional>
#include "../navigation/NS.hpp"
#include "../navigation/environment.hpp"
#include "mixers.hpp"
#include "controller_mode.hpp"
#include "controller_loop.hpp"
#include "common.hpp"
#include "../communication/control.hpp"
Include dependency graph for controller.hpp: This graph shows which files directly or indirectly include this file:
```

### Classes

- class [ControlSystem](#)  
*Central controller class.*

## 7.63 lib/UAV\_common/src/controllers/controller\_test.cpp File Reference

```
#include "impl/PID.hpp"
#include "impl/PID_discrete.hpp"
#include "impl/bang_bang.hpp"
#include "impl/double_setpoint.hpp"
#include "impl/z_trans.hpp"
#include <gtest/gtest.h>
#include <memory>
#include <filesystem>
#include <fstream>
Include dependency graph for controller_test.cpp:
```



## Classes

- class [ControllerTest](#)

## Functions

- `std::vector< std::shared_ptr< Controller > > getMethodsToTest ()`
- `TEST\_P (ControllerTest, TestConstFunction)`
- `TEST\_P (ControllerTest, SimpleObjectControl)`
- `INSTANTIATE\_TEST\_SUITE\_P (TestDerivedClasses, ControllerTest, testing::ValuesIn(getMethodsToTest()))`
- `int main (int argc, char **argv)`

## Variables

- `constexpr bool plot = true`
- `constexpr auto plot\_directory\_name = "controller_plots"`

### 7.63.1 Function Documentation

#### 7.63.1.1 [getMethodsToTest\(\)](#)

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

#### 7.63.1.2 [INSTANTIATE\\_TEST\\_SUITE\\_P\(\)](#)

```
INSTANTIATE\_TEST\_SUITE\_P (  
    TestDerivedClasses ,  
    ControllerTest ,  
    testing::ValuesIn(getMethodsToTest()) )
```

#### 7.63.1.3 [main\(\)](#)

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

**7.63.1.4 TEST\_P() [1/2]**

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

**7.63.1.5 TEST\_P() [2/2]**

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

**7.63.2 Variable Documentation****7.63.2.1 plot**

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

**7.63.2.2 plot\_directory\_name**

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

**7.64 lib/UAV\_common/src/controllers/impl/bang\_bang.cpp File Reference**

```
#include "bang_bang.hpp"
#include <cstring>
#include <string>
Include dependency graph for bang_bang.cpp:
```

**7.65 lib/UAV\_common/src/controllers/impl/bang\_bang.hpp File Reference**

```
#include <memory>
#include "rapidxml/rapidxml.hpp"
#include "../controller.hpp"
Include dependency graph for bang_bang.hpp: This graph shows which files directly or indirectly include this file:
```

## Classes

- class [controllers::BangBang](#)

## Namespaces

- [controllers](#)

## 7.66 lib/UAV\_common/src/controllers/impl/double\_setpoint.cpp File Reference

```
#include "double_setpoint.hpp"
#include <cstring>
#include <string>
Include dependency graph for double_setpoint.cpp:
```

## 7.67 lib/UAV\_common/src/controllers/impl/double\_setpoint.hpp File Reference

```
#include <memory>
#include "rapidxml/rapidxml.hpp"
#include "../controller.hpp"
Include dependency graph for double_setpoint.hpp: This graph shows which files directly or indirectly include this file:
```

## Classes

- class [controllers::DoubleSetpoint](#)

## Namespaces

- [controllers](#)

## 7.68 lib/UAV\_common/src/controllers/impl/PID.cpp File Reference

```
#include "PID.hpp"
#include <algorithm>
#include <cstring>
#include <string>
#include <stdexcept>
Include dependency graph for PID.cpp:
```

## 7.69 lib/UAV\_common/src/controllers/impl/PID.hpp File Reference

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

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

### Classes

- class [controllers::PID](#)

### Namespaces

- [controllers](#)

## 7.70 lib/UAV\_common/src/controllers/impl/PID\_discrete.cpp File Reference

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

Include dependency graph for PID\_discrete.cpp:

## 7.71 lib/UAV\_common/src/controllers/impl/PID\_discrete.hpp File Reference

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

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

### Classes

- class [controllers::PID\\_Discrete](#)

### Namespaces

- [controllers](#)

## 7.72 lib/UAV\_common/src/controllers/impl/z\_trans.cpp File Reference

```
#include "z_trans.hpp"
#include <sstream>
#include <iterator>
#include <string>
#include <cstring>
Include dependency graph for z_trans.cpp:
```

### Functions

- `std::vector< double >` [splitStringToDoubleVector](#) (const std::string &input)

### 7.72.1 Function Documentation

#### 7.72.1.1 [splitStringToDoubleVector\(\)](#)

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

## 7.73 lib/UAV\_common/src/controllers/impl/z\_trans.hpp File Reference

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

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

### Classes

- class [controllers::ZTransformStatic< N, D >](#)
- class [controllers::ZTransform](#)

### Namespaces

- [controllers](#)

## 7.74 lib/UAV\_common/src/logger/logger.cpp File Reference

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

### Functions

- bool [shouldLog](#) (uint8\_t group)

### 7.74.1 Function Documentation

#### 7.74.1.1 shouldLog()

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

## 7.75 lib/UAV\_common/src/logger/logger.hpp File Reference

```
#include <Eigen/Dense>
#include <iostream>
#include <fstream>
#include <initializer_list>
#include <string>
#include <filesystem>
```

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

### Classes

- class [Logger](#)  
*Log vector data with timestamp in file.*

### Macros

- `#define` [LOGGER\\_MASK](#) -1

### 7.75.1 Macro Definition Documentation

### 7.75.1.1 `LOGGER_MASK`

```
#define LOGGER_MASK -1
```

## 7.76 lib/UAV\_common/src/ode/ode.cpp File Reference

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

## 7.77 lib/UAV\_common/src/ode/ode.hpp File Reference

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

### Classes

- class [ODE](#)  
*Ordinal differential equation solver.*

## 7.78 lib/UAV\_common/src/ode/ode\_impl.hpp File Reference

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

### Classes

- class [ODE\\_Euler](#)  
*Explicit Euler algorithm.*
- class [ODE\\_Heun](#)  
*Second order explicit Heun algorithm.*
- class [ODE\\_RK4](#)  
*Fourth order Runge Kutta algorithm.*
- class [ODE\\_PC2](#)  
*Second order predictor-corrector method Second order Adams-bashforth and Adams-moulton.*
- class [ODE\\_PC4](#)  
*Fourth order predictor-corrector method Fourth order Adams-bashforth and Adams-moulton.*

## 7.79 lib/UAV\_common/src/ode/ode\_test.cpp File Reference

```
#include "ode.hpp"  
#include <gtest/gtest.h>  
#include <numbers>  
Include dependency graph for ode_test.cpp:
```

## Classes

- class [ODETest](#)

## Functions

- `std::vector< ODE::ODEMethod > getMethodsToTest ()`
- `TEST\_F (ODETest, FromStringTest)`
- `TEST\_F (ODETest, FactoryTest)`
- `TEST\_P (ODETest, TestConstFunction)`
- `TEST\_P (ODETest, TestFirstOrder)`
- `TEST\_P (ODETest, TestRHSCalls)`
- `TEST\_P (ODETest, TestHarmonicOscillator)`
- `INSTANTIATE\_TEST\_SUITE\_P (TestDerivedClasses, ODETest, testing::ValuesIn(getMethodsToTest()))`
- `int main (int argc, char **argv)`

## 7.79.1 Function Documentation

### 7.79.1.1 [getMethodsToTest\(\)](#)

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

### 7.79.1.2 [INSTANTIATE\\_TEST\\_SUITE\\_P\(\)](#)

```
INSTANTIATE\_TEST\_SUITE\_P (
    TestDerivedClasses ,
    ODETest ,
    testing::ValuesIn(getMethodsToTest()) )
```

### 7.79.1.3 [main\(\)](#)

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

### 7.79.1.4 [TEST\\_F\(\)](#) [1/2]

```
TEST\_F (
    ODETest ,
    FactoryTest )
```



#### 7.79.1.5 TEST\_F() [2/2]

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

#### 7.79.1.6 TEST\_P() [1/4]

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

#### 7.79.1.7 TEST\_P() [2/4]

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

#### 7.79.1.8 TEST\_P() [3/4]

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

#### 7.79.1.9 TEST\_P() [4/4]

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

## 7.80 lib/UAV\_common/src/parser/parser.cpp File Reference

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

## Functions

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

### 7.80.1 Function Documentation

#### 7.80.1.1 parseMatrixXd()

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

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

##### Parameters

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

##### Returns

parsed matrix

#### 7.80.1.2 parseVectorXd()

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

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

##### Parameters

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

**Returns**

parsed vector

## 7.81 lib/UAU\_common/src/parser/parser.hpp File Reference

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

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

**Functions**

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

### 7.81.1 Function Documentation

#### 7.81.1.1 parseMatrixXd()

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

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

**Parameters**

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

**Returns**

parsed matrix

#### 7.81.1.2 parseVectorXd()

```
Eigen::VectorXd parseVectorXd (
    std::string str,
```

```
int noOfElem,
char delimiter = ' ' )
```

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

#### Parameters

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

#### Returns

parsed vector

## 7.82 lib/UAV\_common/src/parser/uav\_params.cpp File Reference

```
#include <Eigen/Dense>
#include "uav_params.hpp"
#include <iostream>
#include <fstream>
#include <filesystem>
#include <mutex>
#include "rapidxml/rapidxml.hpp"
#include "parser.hpp"
Include dependency graph for uav_params.cpp:
```

### Functions

- void [parseHinge](#) (rapidxml::xml\_node<> \*hingeNode, [Hinge](#) \*hinge)

### 7.82.1 Function Documentation

#### 7.82.1.1 parseHinge()

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

## 7.83 lib/UAV\_common/src/parser/uav\_params.hpp File Reference

```
#include <Eigen/Dense>
#include <mutex>
#include <memory>
#include <map>
#include "rapidxml/rapidxml.hpp"
#include "../components/components.hpp"
#include "../controllers/controller.hpp"
Include dependency graph for uav_params.hpp: This graph shows which files directly or indirectly include this file:
```

## Classes

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

## 7.84 lib/UAV\_common/src/timed\_loop/status.hpp File Reference

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

## Enumerations

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

### 7.84.1 Enumeration Type Documentation

#### 7.84.1.1 Status

enum [Status](#)

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

#### Enumerator

<a href="#">idle</a>	loop is ready to run
<a href="#">running</a>	loop is running
<a href="#">exiting</a>	loop will be break in next occasion.
<a href="#">reload</a>	loop job should be reloaded

## 7.85 lib/UAV\_common/src/timed\_loop/timed\_loop.cpp File Reference

```
#include "timed_loop.hpp"
#include <stdint.h>
#include <chrono>
#include <thread>
#include "status.hpp"
#include <iostream>
Include dependency graph for timed_loop.cpp:
```

## 7.86 lib/UAV\_common/src/timed\_loop/timed\_loop.hpp File Reference

```
#include <stdint.h>
```

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

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

## Classes

- class [TimedLoop](#)  
*Simulation of real-time synchronized loop.*

## 7.87 src/communication/control.cpp File Reference

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

Include dependency graph for `control.cpp`:

## Functions

- void [orderServerJob](#) (zmq::context\_t \*ctx, std::string uav\_address, std::function< std::string(std::string)> handleMsg, bool &run)

### 7.87.1 Function Documentation

#### 7.87.1.1 orderServerJob()

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

## 7.88 src/communication/control.hpp File Reference

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

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

## Classes

- class [Control](#)  
*Control command listener & sender.*

## 7.89 src/communication/control\_recv.cpp File Reference

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

## 7.90 src/communication/control\_send.cpp File Reference

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

## 7.91 src/controller/controller\_loop.cpp File Reference

```
#include "controller_loop.hpp"
#include "modes/controller_loop_NONE.hpp"
#include "modes/controller_loop_QACRO.hpp"
#include "modes/controller_loop_QANGLE.hpp"
#include "modes/controller_loop_QPOS.hpp"
#include "modes/controller_loop_FMANUAL.hpp"
#include "modes/controller_loop_FACRO.hpp"
#include "modes/controller_loop_FANGLE.hpp"
#include "modes/controller_loop_RMANUAL.hpp"
#include "modes/controller_loop_RAUTOLAUNCH.hpp"
#include "modes/controller_loop_RANGLE.hpp"
#include "modes/controller_loop_RGUIDED.hpp"
Include dependency graph for controller_loop.cpp:
```

## 7.92 src/controller/controller\_loop.hpp File Reference

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

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

### Classes

- class [ControllerLoop](#)

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

## 7.93 src/controller/controller\_mode.hpp File Reference

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

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

### Enumerations

- enum [ControllerMode](#) {  
[NONE](#) = 0 , [QPOS](#) = 1 , [QANGLE](#) = 2 , [QACRO](#) = 3 ,  
[FMANUAL](#) = 4 , [FACRO](#) = 5 , [FANGLE](#) = 6 , [RMANUAL](#) = 7 ,  
[RAUTOLAUNCH](#) = 8 , [RANGLE](#) = 9 , [RGUIDED](#) = 10 }  
[Controller](#) modes.

### Functions

- constexpr const char \* [ControllerModeToString](#) ([ControllerMode](#) mode) throw ()  
Serializes controller mode to string.
- constexpr [ControllerMode](#) [ControllerModeFromString](#) (const char \*mode) throw ()  
Parse string to controller mode.

## 7.93.1 Enumeration Type Documentation

### 7.93.1.1 ControllerMode

```
enum ControllerMode
```

[Controller](#) modes.

#### Enumerator

NONE	
QPOS	
QANGLE	
QACRO	
FMANUAL	
FACRO	
FANGLE	
RMANUAL	
RAUTOLAUNCH	
RANGLE	
RGUIDED	



## 7.93.2 Function Documentation

### 7.93.2.1 ControllerModeFromString()

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

Parse string to controller mode.

#### Parameters

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

#### Returns

parsing result, NONE if parse failed

### 7.93.2.2 ControllerModeToString()

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

Serializes controller mode to string.

#### Parameters

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

#### Returns

serialized mode

## 7.94 src/controller/mixers.cpp File Reference

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

## Functions

- Eigen::VectorXd [applyMixerRotors](#) (double climb\_rate, double roll\_rate, double pitch\_rate, double yaw\_rate)

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

- Eigen::VectorXd [applyMixerRotorsHover](#) (double throttle, double roll\_rate, double pitch\_rate, double yaw\_rate)

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

- Eigen::VectorXd [applyMixerSurfaces](#) (double throttle, double roll\_rate, double pitch\_rate, double yaw\_rate)

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

## 7.94.1 Function Documentation

### 7.94.1.1 [applyMixerRotors\(\)](#)

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

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

#### Parameters

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

#### Returns

Rotors demanded speed

### 7.94.1.2 [applyMixerRotorsHover\(\)](#)

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

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

## Parameters

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

## Returns

Rotors demanded speed

## 7.94.1.3 applyMixerSurfaces()

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

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

## Parameters

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

## Returns

demanded surfaces deflection

## 7.95 src/controller/mixers.hpp File Reference

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

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

## Functions

- Eigen::VectorXd [applyMixerRotors](#) (double climb\_rate, double roll\_rate, double pitch\_rate, double yaw\_rate)  
*Calculates rotor demanded speed as result of multiplication mixer matrix and rates. Average speed is proportional to climb rate.*
- Eigen::VectorXd [applyMixerRotorsHover](#) (double throttle, double roll\_rate, double pitch\_rate, double yaw\_rate)  
*Calculates rotor demanded speed as result of multiplication mixer matrix and rates. Average speed is proportional to throttle. It's scaled to achieve hover at centered throttle.*
- Eigen::VectorXd [applyMixerSurfaces](#) (double throttle, double roll\_rate, double pitch\_rate, double yaw\_rate)  
*Calculated demanded surfaces deflection result of multiplication mixer matrix and rates.*

## 7.95.1 Function Documentation

### 7.95.1.1 `applyMixerRotors()`

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

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

#### Parameters

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

#### Returns

Rotors demanded speed

### 7.95.1.2 `applyMixerRotorsHover()`

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

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

#### Parameters

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

#### Returns

Rotors demanded speed

### 7.95.1.3 applyMixerSurfaces()

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

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

#### Parameters

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

#### Returns

demanded surfaces deflection

## 7.96 src/controller/modes/controller\_loop\_FACRO.cpp File Reference

```
#include "controller_loop_FACRO.hpp"
#include "../utils.hpp"
Include dependency graph for controller_loop_FACRO.cpp:
```

## 7.97 src/controller/modes/controller\_loop\_FACRO.hpp File Reference

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

#### Classes

- class [ControllerLoopFACRO](#)

## 7.98 src/controller/modes/controller\_loop\_FANGLE.cpp File Reference

```
#include "controller_loop_FANGLE.hpp"
#include "../utils.hpp"
Include dependency graph for controller_loop_FANGLE.cpp:
```

## 7.99 src/controller/modes/controller\_loop\_FANGLE.hpp File Reference

```
#include "../controller_loop.hpp"
```

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

### Classes

- class [ControllerLoopFANGLE](#)

## 7.100 src/controller/modes/controller\_loop\_FMANUAL.cpp File Reference

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

Include dependency graph for controller\_loop\_FMANUAL.cpp:

## 7.101 src/controller/modes/controller\_loop\_FMANUAL.hpp File Reference

```
#include "../controller_loop.hpp"
```

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

### Classes

- class [ControllerLoopFMANUAL](#)

## 7.102 src/controller/modes/controller\_loop\_NONE.cpp File Reference

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

Include dependency graph for controller\_loop\_NONE.cpp:

## 7.103 src/controller/modes/controller\_loop\_NONE.hpp File Reference

```
#include "../controller_loop.hpp"
```

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

### Classes

- class [ControllerLoopNONE](#)

## 7.104 src/controller/modes/controller\_loop\_QACRO.cpp File Reference

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

Include dependency graph for controller\_loop\_QACRO.cpp:

## 7.105 src/controller/modes/controller\_loop\_QACRO.hpp File Reference

```
#include "../controller_loop.hpp"
```

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

### Classes

- class [ControllerLoopQACRO](#)

## 7.106 src/controller/modes/controller\_loop\_QANGLE.cpp File Reference

```
#include "controller_loop_QANGLE.hpp"
```

```
#include "../../utils.hpp"
```

Include dependency graph for controller\_loop\_QANGLE.cpp:

## 7.107 src/controller/modes/controller\_loop\_QANGLE.hpp File Reference

```
#include "../controller_loop.hpp"
```

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

### Classes

- class [ControllerLoopQANGLE](#)

## 7.108 src/controller/modes/controller\_loop\_QPOS.cpp File Reference

```
#include "controller_loop_QPOS.hpp"
```

```
#include "../../utils.hpp"
```

Include dependency graph for controller\_loop\_QPOS.cpp:

## 7.109 src/controller/modes/controller\_loop\_QPOS.hpp File Reference

```
#include "../controller_loop.hpp"
```

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

## Classes

- class [ControllerLoopQPOS](#)

### 7.110 src/controller/modes/controller\_loop\_RANGLE.cpp File Reference

```
#include "controller_loop_RANGLE.hpp"
#include "../utils.hpp"
Include dependency graph for controller_loop_RANGLE.cpp:
```

### 7.111 src/controller/modes/controller\_loop\_RANGLE.hpp File Reference

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

## Classes

- class [ControllerLoopRANGLE](#)

### 7.112 src/controller/modes/controller\_loop\_RAUTOLAUNCH.cpp File Reference

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

### 7.113 src/controller/modes/controller\_loop\_RAUTOLAUNCH.hpp File Reference

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

## Classes

- class [ControllerLoopRAUTOLAUNCH](#)

### 7.114 src/controller/modes/controller\_loop\_RGUIDED.cpp File Reference

```
#include "controller_loop_RGUIDED.hpp"
#include "../utils.hpp"
#include "common.hpp"
Include dependency graph for controller_loop_RGUIDED.cpp:
```



## 7.115 src/controller/modes/controller\_loop\_RGUIDED.hpp File Reference

```
#include "../controller_loop.hpp"
```

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

### Classes

- class [ControllerLoopRGUIDED](#)

## 7.116 src/controller/modes/controller\_loop\_RMANUAL.cpp File Reference

```
#include "controller_loop_RMANUAL.hpp"
```

Include dependency graph for controller\_loop\_RMANUAL.cpp:

## 7.117 src/controller/modes/controller\_loop\_RMANUAL.hpp File Reference

```
#include "../controller_loop.hpp"
```

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

### Classes

- class [ControllerLoopRMANUAL](#)

## 7.118 src/defines.hpp File Reference

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

### Namespaces

- [def](#)  
*Controller constants.*

### Macros

- `#define` [USE\\_QUATERIONS](#) 1

## Variables

- const int `def::INFO_PERIOD` = 2  
*How often send demands in response to stick command.*

### 7.118.1 Macro Definition Documentation

#### 7.118.1.1 USE\_QUATERIONS

```
#define USE_QUATERIONS 1
```

## 7.119 src/main.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <cxxopts.hpp>
#include <thread>
#include <chrono>
#include <filesystem>
#include "zmq.hpp"
#include "controller/controller.hpp"
#include "common.hpp"
#include "params.hpp"
```

Include dependency graph for main.cpp:

## Macros

- #define `LOGGER_MASK` 5

## Functions

- void `parseArgs` (int argc, char \*\*argv, `UAVparams` \*params, `Params` &p)  
*Parse CL arguments.*
- int `main` (int argc, char \*\*argv)

## Variables

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

### 7.119.1 Macro Definition Documentation

### 7.119.1.1 `LOGGER_MASK`

```
#define LOGGER_MASK 5
```

## 7.119.2 Function Documentation

### 7.119.2.1 `main()`

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

### 7.119.2.2 `parseArgs()`

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

Parse CL arguments.

#### Parameters

<i>argc</i>	number of argument
<i>argv</i>	argument array
<i>params</i>	pointer to <a href="#">UAVparams</a> instant that should be filled
<i>p</i>	internal params reference

## 7.119.3 Variable Documentation

### 7.119.3.1 `log_path`

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

## 7.120 src/navigation/AHRS.cpp File Reference

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

## 7.121 src/navigation/AHRS.hpp File Reference

```
#include <Eigen/Dense>
#include <random>
#include <optional>
#include "environment.hpp"
#include "sensors.hpp"
Include dependency graph for AHRS.hpp: This graph shows which files directly or indirectly include this file:
```

### Classes

- class [AHRS](#)  
*Attitude and heading reference system.*

## 7.122 src/navigation/AHRS/AHRS\_complementary.cpp File Reference

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

### Functions

- Eigen::Matrix3d [calcRnb](#) (Eigen::Vector3d ori)
- Eigen::Matrix3d [calcRbn](#) (Eigen::Vector3d ori)
- Eigen::Matrix3d [calcTom](#) (Eigen::Vector3d ori)
- void [clampOrientation](#) (Eigen::Vector3d &vec)

### 7.122.1 Function Documentation

#### 7.122.1.1 calcRbn()

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

### 7.122.1.2 calcRnb()

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

### 7.122.1.3 calcTom()

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

### 7.122.1.4 clampOrientation()

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

## 7.123 src/navigation/AHRS/AHRS\_complementary.hpp File Reference

```
#include <Eigen/Dense>
#include <random>
#include "../environment.hpp"
#include "../sensors.hpp"
#include "common.hpp"
#include "../AHRS.hpp"
```

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

### Classes

- class [AHRS\\_complementary](#)  
*Implementation of [AHRS](#) based on Complementary Filter.*

## 7.124 src/navigation/AHRS/AHRS\_EKF.cpp File Reference

```
#include "AHRS_EKF.hpp"
#include <Eigen/Dense>
#include <random>
#include <iostream>
```

Include dependency graph for AHRS\_EKF.cpp:

### Functions

- Eigen::Matrix< double, 4, 3 > [S](#) (Eigen::Vector4d q)
- Eigen::Matrix< double, 6, 7 > [C](#) (Eigen::Vector4d q)

### 7.124.1 Function Documentation

#### 7.124.1.1 C()

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

#### 7.124.1.2 S()

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

## 7.125 src/navigation/AHRS/AHRS\_EKF.hpp File Reference

```
#include <Eigen/Dense>
#include "../environment.hpp"
#include "../sensors.hpp"
#include "common.hpp"
#include "../AHRS.hpp"
```

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

### Classes

- class [AHRS\\_EKF](#)

*Implementation of [AHRS](#) based on Extended Kalman Filter.*

## 7.126 src/navigation/EKF.cpp File Reference

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

Include dependency graph for EKF.cpp:

## 7.127 src/navigation/EKF.hpp File Reference

```
#include <Eigen/Dense>
#include "environment.hpp"
#include "sensors.hpp"
```

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

## Classes

- struct [EKFPParams](#)  
*EK filter parameters.*
- class [EKF](#)  
*Extended Kalman Filter.*

## 7.128 src/navigation/environment.cpp File Reference

```
#include "environment.hpp"
#include <zmq.hpp>
#include <thread>
#include <Eigen/Dense>
#include <mutex>
#include <vector>
#include <memory>
#include <iostream>
#include <initializer_list>
#include "../utils.hpp"
#include "common.hpp"
#include "sensors.hpp"
#include "../defines.hpp"
Include dependency graph for environment.cpp:
```

## Functions

- void [connectConflateSocket](#) (zmq::socket\_t &sock, std::string address, std::string topic)
- template<int Size1, int Size2>  
bool [recvVectors](#) (zmq::socket\_t &sock, int skip, Eigen::Vector< double, Size1 > &vec1, Eigen::Vector< double, Size2 > &vec2)
- Eigen::Matrix< double, 3, 3 > [r\\_nb](#) (const Eigen::Vector3d &RPY)
- Eigen::Matrix< double, 3, 3 > [r\\_nb](#) (const Eigen::Vector4d &e)

### 7.128.1 Function Documentation

#### 7.128.1.1 connectConflateSocket()

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

**7.128.1.2 r\_nb() [1/2]**

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

**7.128.1.3 r\_nb() [2/2]**

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

**7.128.1.4 recvVectors()**

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

**7.129 src/navigation/environment.hpp File Reference**

```
#include <zmq.hpp>
#include <thread>
#include <Eigen/Dense>
#include <mutex>
#include <vector>
#include <memory>
#include <atomic>
#include <map>
#include "sensors.hpp"
#include "common.hpp"
#include "../defines.hpp"
```

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

**Classes**

- class [Environment](#)

**7.130 src/navigation/NS.cpp File Reference**

```
#include "NS.hpp"
#include <Eigen/Dense>
#include <iostream>
#include "AHRS/AHRS_EKF.hpp"
#include "AHRS/AHRS_complementary.hpp"
#include "../defines.hpp"
#include "../params.hpp"
```

Include dependency graph for NS.cpp:



## 7.131 src/navigation/NS.hpp File Reference

```
#include <Eigen/Dense>
#include "environment.hpp"
#include "sensors.hpp"
#include "AHRS.hpp"
#include "EKF.hpp"
```

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

### Classes

- class [NS](#)  
*Navigation system.*

## 7.132 src/navigation/sensors.cpp File Reference

```
#include "sensors.hpp"
#include <Eigen/Dense>
#include <random>
#include <limits>
#include "environment.hpp"
#include "common.hpp"
```

Include dependency graph for sensors.cpp:

## 7.133 src/navigation/sensors.hpp File Reference

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

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

### Classes

- class [Sensor< T >](#)  
*Sensors base class.*
- class [Accelerometer](#)  
*Representation of accelerometer.*
- class [Gyroscope](#)  
*Representation of gyroscope.*
- class [Magnetometer](#)  
*Representation of magnetometer.*
- class [Barometer](#)  
*Representation of barometer.*
- class [GPS](#)  
*Representation of [GPS](#) position measure.*
- class [GPSVel](#)  
*Representation of [GPS](#) velocity measure.*

## 7.134 src/params.cpp File Reference

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

## 7.135 src/params.hpp File Reference

```
#include <string>
Include dependency graph for params.hpp: This graph shows which files directly or indirectly include this file:
```

### Classes

- class [Params](#)  
*Simulation parameters.*

## 7.136 src/utils.hpp File Reference

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

### Functions

- template<typename T >  
void [safeSet](#) (T &vec, T &new\_val, std::mutex &mtx)  
*Safe setter for T type value protected by mutex.*
- template<typename T >  
T [safeGet](#) (T &vec, std::mutex &mtx)  
*Safe getter for T type value protected by mutex.*
- double [circularError](#) (double demanded, double val)  
*Calculates error between demanded and actual angle. Finds shorter path. For example if actual value is -0.9pi and demanded is 0.9pi error is equal -0.2pi.*
- double [clampAngle](#) (double angle)  
*Clamps angle given in radians to range <-pi,pi>*

### 7.136.1 Function Documentation

#### 7.136.1.1 circularError()

```
double circularError (
    double demanded,
    double val ) [inline]
```

Calculates error between demanded and actual angle. Finds shorter path. For example if actual value is -0.9pi and demanded is 0.9pi error is equal -0.2pi.

## Parameters

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

## Returns

angle error

**7.136.1.2 clampAngle()**

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

Clamps angle given in radians to range  $\langle -\pi, \pi \rangle$

## Parameters

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

## Returns

angle converted to range  $\langle -\pi, \pi \rangle$

**7.136.1.3 safeGet()**

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

Safe getter for T type value protected by mutex.

## Template Parameters

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

## Parameters

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

**Returns**

value of *vec*

**7.136.1.4 safeSet()**

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

Safe setter for T type value protected by mutex.

**Template Parameters**

<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, 28](#)
  - [\\_\\_has\\_include](#)
    - [CMakeCCompilerId.c, 137](#)
    - [CMakeCXXCompilerId.cpp, 141](#)
  - [\\_dt](#)
    - [Controller, 40](#)
  - [\\_mode](#)
    - [ControllerLoop, 44](#)
  - [~AHRS](#)
    - [AHRS, 19](#)
  - [~AHRS\\_EKF](#)
    - [AHRS\\_EKF, 23](#)
  - [~AHRS\\_complementary](#)
    - [AHRS\\_complementary, 21](#)
  - [~Control](#)
    - [Control, 34](#)
  - [~ControlSystem](#)
    - [ControlSystem, 65](#)
  - [~Controller](#)
    - [Controller, 38](#)
  - [~ControllerLoop](#)
    - [ControllerLoop, 42](#)
  - [~Environment](#)
    - [Environment, 76](#)
  - [~Logger](#)
    - [Logger, 91](#)
  - [~NS](#)
    - [NS, 95](#)
  - [~ODE](#)
    - [ODE, 98](#)
  - [~Params](#)
    - [Params, 110](#)
  - [~UAVparams](#)
    - [UAVparams, 127](#)
- [Accelerometer, 15](#)
  - [Accelerometer, 15](#)
  - [g, 16](#)
  - [update, 16](#)
- [aero\\_coffs](#)
  - [UAVparams, 128](#)
- [AeroCoefficients, 16](#)
  - [C0, 17](#)
  - [Cab, 17](#)
  - [Cpqr, 17](#)
  - [d, 17](#)
  - [eAR, 17](#)
  - [S, 17](#)
  - [stallLimit, 17](#)
- [AHRS, 18](#)
  - [~AHRS, 19](#)
  - [AHRS, 18](#)
  - [env, 20](#)
  - [getGyroBias, 19](#)
  - [getOri, 19](#)
  - [logger, 20](#)
  - [mtxOri, 20](#)
  - [ori\\_est, 20](#)
  - [rot\\_bw, 19](#)
  - [update, 20](#)
- [ahrs](#)
  - [UAVparams, 128](#)
- [AHRS\\_complementary, 21](#)
  - [~AHRS\\_complementary, 21](#)
  - [AHRS\\_complementary, 21](#)
  - [alpha, 22](#)
  - [rot\\_bw, 22](#)
  - [update, 22](#)
- [AHRS\\_complementary.cpp](#)
  - [calcRbn, 180](#)
  - [calcRnb, 180](#)
  - [calcTom, 181](#)
  - [clampOrientation, 181](#)
- [AHRS\\_EKF, 23](#)
  - [~AHRS\\_EKF, 23](#)
  - [AHRS\\_EKF, 23](#)
  - [getGyroBias, 24](#)
  - [P, 25](#)
  - [Q, 25](#)
  - [q, 24](#)
  - [quaternionToRPY, 24](#)
  - [R, 25](#)
  - [rot\\_bw, 24](#)
  - [RPYToQuaternion, 24](#)
  - [update, 25](#)
  - [x, 25](#)
- [AHRS\\_EKF.cpp](#)
  - [C, 182](#)
  - [S, 182](#)
- [AHRSParams, 26](#)
  - [alpha, 26](#)
  - [Q, 26](#)
  - [R, 26](#)
  - [type, 26](#)
- [alpha](#)
  - [AHRS\\_complementary, 22](#)
  - [AHRSParams, 26](#)
- [Ammo, 27](#)

- [\\_V0](#), [28](#)
- [Ammo](#), [27](#)
- [getV0](#), [27](#)
- [operator=](#), [28](#)
- [ammo](#)
  - [UAVparams](#), [128](#)
- [angleLimit](#)
  - [ControllerLoopFANGLE](#), [48](#)
  - [ControllerLoopRANGLE](#), [57](#)
- [AntiWindUpMode](#)
  - [controllers::PID](#), [111](#)
- [applyMixerRotors](#)
  - [mixers.cpp](#), [170](#)
  - [mixers.hpp](#), [172](#)
- [applyMixerRotorsHover](#)
  - [mixers.cpp](#), [170](#)
  - [mixers.hpp](#), [172](#)
- [applyMixerSurfaces](#)
  - [mixers.cpp](#), [171](#)
  - [mixers.hpp](#), [172](#)
- [ARCHITECTURE\\_ID](#)
  - [CMakeCCompilerId.c](#), [138](#)
  - [CMakeCXXCompilerId.cpp](#), [141](#)
- [asSkewMatrix](#)
  - [hinge.cpp](#), [150](#)
- [axis](#)
  - [Drive](#), [69](#)
- [BangBang](#)
  - [controllers::BangBang](#), [29](#)
- [Barometer](#), [30](#)
  - [Barometer](#), [31](#)
  - [update](#), [31](#)
- [baroScaler](#)
  - [EKFScalers](#), [74](#)
- [bias](#)
  - [Sensor< T >](#), [121](#)
  - [SensorParams](#), [122](#)
- [build/CMakeFiles/3.22.1/CompilerIdC/CMakeCCompilerId.c](#), [137](#)
- [build/CMakeFiles/3.22.1/CompilerIdCXX/CMakeCXXCompilerId.cpp](#), [140](#)
- [build/CMakeFiles/controller.dir/src/communication/control.cpp.o.d](#), [145](#)
- [build/CMakeFiles/controller.dir/src/communication/control\\_receive.cpp.o.d](#), [145](#)
- [build/CMakeFiles/controller.dir/src/communication/control\\_send.cpp.o.d](#), [145](#)
- [build/CMakeFiles/controller.dir/src/controller/controller.cpp.o.d](#), [145](#)
- [build/CMakeFiles/controller.dir/src/controller/controller\\_loop.cpp.o.d](#), [145](#)
- [build/CMakeFiles/controller.dir/src/controller/mixers.cpp.o.d](#), [145](#)
- [build/CMakeFiles/controller.dir/src/controller/modes/controller\\_loop\\_FANGLE.cpp.o.d](#), [145](#)
- [build/CMakeFiles/controller.dir/src/controller/modes/controller\\_loop\\_NONE.cpp.o.d](#), [145](#)
- [build/CMakeFiles/controller.dir/src/controller/modes/controller\\_loop\\_QACF.cpp.o.d](#), [145](#)
- [build/CMakeFiles/controller.dir/src/controller/modes/controller\\_loop\\_QANG.cpp.o.d](#), [145](#)
- [build/CMakeFiles/controller.dir/src/controller/modes/controller\\_loop\\_QPOS.cpp.o.d](#), [145](#)
- [build/CMakeFiles/controller.dir/src/controller/modes/controller\\_loop\\_RANG.cpp.o.d](#), [145](#)
- [build/CMakeFiles/controller.dir/src/controller/modes/controller\\_loop\\_RAUT.cpp.o.d](#), [145](#)
- [build/CMakeFiles/controller.dir/src/controller/modes/controller\\_loop\\_RGUI.cpp.o.d](#), [145](#)
- [build/CMakeFiles/controller.dir/src/controller/modes/controller\\_loop\\_RMAN.cpp.o.d](#), [145](#)
- [build/CMakeFiles/controller.dir/src/main.cpp.o.d](#), [145](#)
- [build/CMakeFiles/controller.dir/src/navigation/AHRS.cpp.o.d](#), [145](#)
- [build/CMakeFiles/controller.dir/src/navigation/AHRS/AHRS\\_complementar.cpp.o.d](#), [145](#)
- [build/CMakeFiles/controller.dir/src/navigation/AHRS/AHRS\\_EKF.cpp.o.d](#), [145](#)
- [build/CMakeFiles/controller.dir/src/navigation/EKF.cpp.o.d](#), [145](#)
- [build/CMakeFiles/controller.dir/src/navigation/environment.cpp.o.d](#), [145](#)
- [build/CMakeFiles/controller.dir/src/navigation/NS.cpp.o.d](#), [145](#)
- [build/CMakeFiles/controller.dir/src/navigation/sensors.cpp.o.d](#), [145](#)
- [build/CMakeFiles/controller.dir/src/params.cpp.o.d](#), [145](#)
- [build/lib/UAV\\_common/CMakeFiles/common.dir/src/components/control\\_s.cpp.o.d](#), [145](#)
- [build/lib/UAV\\_common/CMakeFiles/common.dir/src/components/drive.cpp.o.d](#), [145](#)
- [build/lib/UAV\\_common/CMakeFiles/common.dir/src/components/hinge.cpp.o.d](#), [145](#)
- [build/lib/UAV\\_common/CMakeFiles/common.dir/src/components/loads.cpp.o.d](#), [145](#)
- [build/lib/UAV\\_common/CMakeFiles/common.dir/src/controllers/controller.c](#), [145](#)
- [build/lib/UAV\\_common/CMakeFiles/common.dir/src/controllers/impl/bang\\_](#), [145](#)
- [build/lib/UAV\\_common/CMakeFiles/common.dir/src/controllers/impl/double](#), [145](#)
- [build/lib/UAV\\_common/CMakeFiles/common.dir/src/controllers/impl/PID.cp](#), [145](#)
- [build/lib/UAV\\_common/CMakeFiles/common.dir/src/controllers/impl/PID\\_d](#), [145](#)
- [build/lib/UAV\\_common/CMakeFiles/common.dir/src/controllers/impl/z\\_tran](#), [145](#)
- [build/lib/UAV\\_common/CMakeFiles/common.dir/src/logger/logger.cpp.o.d](#), [145](#)
- [build/lib/UAV\\_common/CMakeFiles/common.dir/src/ode/ode.cpp.o.d](#), [145](#)

build/lib/UAV\_common/CMakeFiles/common.dir/src/parser/parser.cpp, 145  
 build/lib/UAV\_common/CMakeFiles/common.dir/src/parser/uav\_params.cpp, 145  
 build/lib/UAV\_common/CMakeFiles/common.dir/src/timed\_loop/timed\_loop.cpp, 145  
 build/lib/UAV\_common/CMakeFiles/Controller\_test.dir/src/Controller\_test.cpp.o.d, 145  
 build/lib/UAV\_common/CMakeFiles/ODE\_test.dir/src/ode/ode\_test.cpp.o.d, 145  
  
**C**  
     AHRS\_EKF.cpp, 182  
**C0**  
     AeroCoefficients, 17  
**C\_VERSION**  
     CMakeCCompilerId.c, 138  
**Cab**  
     AeroCoefficients, 17  
**calc**  
     Controller, 38, 39  
     controllers::BangBang, 29  
     controllers::DoubleSetpoint, 68  
     controllers::PID, 113  
     controllers::PID\_Discrete, 115  
     controllers::ZTransform, 133  
     controllers::ZTransformStatic< N, D >, 135  
**calcRbn**  
     AHRS\_complementary.cpp, 180  
**calcRnb**  
     AHRS\_complementary.cpp, 180  
**calcTom**  
     AHRS\_complementary.cpp, 181  
**Cargo**, 31  
     Cargo, 32  
**cargo**  
     UAVparams, 128  
**checkJoystickLength**  
     ControllerLoop, 42  
**circularError**  
     utils.hpp, 186  
**clampAngle**  
     utils.hpp, 187  
**CLAMPING**  
     controllers::PID, 112  
**clampOrientation**  
     AHRS\_complementary.cpp, 181  
**clc**  
     controller\_plots.m, 147  
**clear**  
     Controller, 39  
     controllers::BangBang, 30  
     controllers::DoubleSetpoint, 68  
     controllers::PID, 113  
     controllers::PID\_Discrete, 115  
     controllers::ZTransform, 133  
     controllers::ZTransformStatic< N, D >, 135  
**clone**  
     Controller, 39  
     controllers::BangBang, 30  
     controllers::DoubleSetpoint, 68  
     controllers::PID, 113  
     controllers::PID\_Discrete, 115  
     controllers::ZTransform, 134  
     controllers::ZTransformStatic< N, D >, 136  
     \_\_has\_include, 137  
     ARCHITECTURE\_ID, 138  
     C\_VERSION, 138  
     COMPILER\_ID, 138  
     DEC, 138  
     HEX, 138  
     info\_arch, 139  
     info\_compiler, 139  
     info\_language\_extensions\_default, 139  
     info\_language\_standard\_default, 140  
     info\_platform, 140  
     main, 139  
     PLATFORM\_ID, 138  
     STRINGIFY, 139  
     STRINGIFY\_HELPER, 139  
**CMakeCXXCompilerId.cpp**  
     \_\_has\_include, 141  
     ARCHITECTURE\_ID, 141  
     COMPILER\_ID, 141  
     CXX\_STD, 141  
     DEC, 141  
     HEX, 141  
     info\_arch, 142  
     info\_compiler, 143  
     info\_language\_extensions\_default, 143  
     info\_language\_standard\_default, 143  
     info\_platform, 143  
     main, 142  
     PLATFORM\_ID, 142  
     STRINGIFY, 142  
     STRINGIFY\_HELPER, 142  
**COMPILER\_ID**  
     CMakeCCompilerId.c, 138  
     CMakeCXXCompilerId.cpp, 141  
**connectConflateSocket**  
     environment.cpp, 183  
**Control**, 32  
     ~Control, 34  
     Control, 33  
     ControlSystem, 66  
     handleMsg, 34  
     prepare, 34  
     recv, 34  
     sendHinge, 34  
     sendSpeed, 36  
     sendSurface, 36  
     setMode, 36  
     start, 36  
     startJet, 36  
     stop, 37  
**control.cpp**

- orderServerJob, 166
- Controller, 37
  - \_dt, 40
  - ~Controller, 38
  - calc, 38, 39
  - clear, 39
  - clone, 39
  - Controller, 38
  - ControllerFactory, 39
  - set\_dt, 40
- controller\_mode.hpp
  - ControllerMode, 168
  - ControllerModeFromString, 169
  - ControllerModeToString, 169
  - FACRO, 168
  - FANGLE, 168
  - FMANUAL, 168
  - NONE, 168
  - QACRO, 168
  - QANGLE, 168
  - QPOS, 168
  - RANGLE, 168
  - RAUTOLAUNCH, 168
  - RGUIDED, 168
  - RMANUAL, 168
- controller\_plots.m
  - clc, 147
  - csvFiles, 147
  - data, 147
  - figure, 148
  - folderPath, 148
  - i, 148
  - legend, 146
  - off, 148
  - on, 148
  - plot, 146
  - title, 147
  - x, 148
  - xlabel, 147
  - y, 148
  - ylabel, 147
- controller\_test.cpp
  - getMethodsToTest, 153
  - INstantiate\_Test\_Suite\_P, 153
  - main, 153
  - plot, 154
  - plot\_directory\_name, 154
  - TEST\_P, 153, 154
- ControllerFactory
  - Controller, 39
- ControllerLoop, 40
  - \_mode, 44
  - ~ControllerLoop, 42
  - checkJoystickLength, 42
  - ControllerLoop, 41
  - ControllerLoopFactory, 42
  - demandInfo, 43
  - getMode, 43
  - handleJoystick, 43
  - job, 43
  - overridePositionAndSpeed, 44
  - required\_controllers, 45
  - requiredcontrollers, 44
- ControllerLoopFACRO, 45
  - ControllerLoopFACRO, 45
  - demandInfo, 45
  - handleJoystick, 46
  - job, 46
- ControllerLoopFactory
  - ControllerLoop, 42
- ControllerLoopFANGLE, 46
  - angleLimit, 48
  - ControllerLoopFANGLE, 47
  - demandInfo, 47
  - handleJoystick, 47
  - job, 47
  - overridePositionAndSpeed, 47
- ControllerLoopFMANUAL, 48
  - ControllerLoopFMANUAL, 48
  - handleJoystick, 49
  - job, 49
- ControllerLoopNONE, 49
  - ControllerLoopNONE, 49
- ControllerLoopQACRO, 50
  - ControllerLoopQACRO, 50
  - demandInfo, 50
  - handleJoystick, 51
  - job, 51
- ControllerLoopQANGLE, 51
  - ControllerLoopQANGLE, 52
  - demandInfo, 52
  - handleJoystick, 52
  - job, 52
  - overridePositionAndSpeed, 52
- ControllerLoopQPOS, 53
  - ControllerLoopQPOS, 53
  - demandInfo, 53
  - handleJoystick, 54
  - job, 54
  - overridePositionAndSpeed, 54
- ControllerLoopRANGLE, 55
  - angleLimit, 57
  - ControllerLoopRANGLE, 55
  - demandPsi, 57
  - demandTheta, 57
  - demandInfo, 55
  - handleJoystick, 56
  - job, 56
  - overridePositionAndSpeed, 56
- ControllerLoopRAUTOLAUNCH, 57
  - ControllerLoopRAUTOLAUNCH, 57
  - job, 58
- ControllerLoopRGUIDED, 58
  - ControllerLoopRGUIDED, 59
  - demandInfo, 59
  - detection\_limit, 59



- job, [59](#)
  - target, [59](#)
- ControllerLoopRMANUAL, [60](#)
  - ControllerLoopRMANUAL, [60](#)
  - demanded\_H, [61](#)
  - demanded\_V, [61](#)
  - demandInfo, [60](#)
  - handleJoystick, [61](#)
  - job, [61](#)
- ControllerMode
  - controller\_mode.hpp, [168](#)
- ControllerModeFromString
  - controller\_mode.hpp, [169](#)
- ControllerModeToString
  - controller\_mode.hpp, [169](#)
- controllers, [13](#)
  - UAVparams, [128](#)
- controllers::BangBang, [28](#)
  - BangBang, [29](#)
  - calc, [29](#)
  - clear, [30](#)
  - clone, [30](#)
- controllers::DoubleSetpoint, [66](#)
  - calc, [68](#)
  - clear, [68](#)
  - clone, [68](#)
  - DoubleSetpoint, [67](#)
- controllers::PID, [111](#)
  - AntiWindUpMode, [111](#)
  - calc, [113](#)
  - CLAMPING, [112](#)
  - clear, [113](#)
  - clone, [113](#)
  - NONE, [112](#)
  - PID, [112](#)
- controllers::PID\_Discrete, [114](#)
  - calc, [115](#)
  - clear, [115](#)
  - clone, [115](#)
  - PID\_Discrete, [114](#), [115](#)
  - set\_dt, [116](#)
- controllers::ZTransform, [132](#)
  - calc, [133](#)
  - clear, [133](#)
  - clone, [134](#)
  - ZTransform, [132](#), [133](#)
- controllers::ZTransformStatic< N, D >, [134](#)
  - calc, [135](#)
  - clear, [135](#)
  - clone, [136](#)
  - ZTransformStatic, [134](#), [135](#)
- ControllerTest, [62](#)
  - SetUp, [62](#)
  - TearDown, [62](#)
- ControlSurfaces, [62](#)
  - ControlSurfaces, [63](#)
  - getCoefficients, [63](#)
  - getNoOfSurface, [63](#)
- getValues, [64](#)
  - restoreTrim, [64](#)
  - setValues, [64](#)
- ControlSystem, [64](#)
  - ~ControlSystem, [65](#)
  - Control, [66](#)
  - ControlSystem, [65](#)
  - exitController, [65](#)
  - run, [65](#)
  - setMode, [66](#)
- Cpqr
  - AeroCoefficients, [17](#)
- csvFiles
  - controller\_plots.m, [147](#)
- CXX\_STD
  - CMakeCXXCompilerId.cpp, [141](#)
- d
  - AeroCoefficients, [17](#)
- data
  - controller\_plots.m, [147](#)
- DEC
  - CMakeCCompilerId.c, [138](#)
  - CMakeCXXCompilerId.cpp, [141](#)
- def, [13](#)
  - INFO\_PERIOD, [13](#)
- defines.hpp
  - USE\_QUATERIONS, [178](#)
- demanded\_H
  - ControllerLoopRMANUAL, [61](#)
- demanded\_V
  - ControllerLoopRMANUAL, [61](#)
- demandedPsi
  - ControllerLoopRANGLE, [57](#)
- demandedTheta
  - ControllerLoopRANGLE, [57](#)
- demandInfo
  - ControllerLoop, [43](#)
  - ControllerLoopFACRO, [45](#)
  - ControllerLoopFANGLE, [47](#)
  - ControllerLoopQACRO, [50](#)
  - ControllerLoopQANGLE, [52](#)
  - ControllerLoopQPOS, [53](#)
  - ControllerLoopRANGLE, [55](#)
  - ControllerLoopRGUIDED, [59](#)
  - ControllerLoopRMANUAL, [60](#)
- detection\_limit
  - ControllerLoopRGUIDED, [59](#)
- direction
  - Rotor, [117](#)
- dist
  - Sensor< T >, [121](#)
- DoubleSetpoint
  - controllers::DoubleSetpoint, [67](#)
- Drive, [69](#)
  - axis, [69](#)
  - hinges, [69](#)
  - noOfHinges, [69](#)
  - position, [69](#)

- eAR
  - AeroCoefficients, 17
- EKF, 70
  - EKF, 70
  - getPos, 71
  - getVel, 71
  - log, 71
  - predict, 71
  - updateBaro, 72
  - updateGPS, 72
  - updateGPSVel, 72
- ekf
  - UAVparams, 128
- EKFParams, 73
  - P0, 73
  - Q, 73
  - RBaro, 73
  - RGPSPos, 74
  - RGPSVel, 74
- EKFScalers, 74
  - baroScaler, 74
  - predictScaler, 75
  - updateScaler, 75
  - zScaler, 75
- env
  - AHRS, 20
  - Sensor< T >, 121
- Environment, 75
  - ~Environment, 76
  - Environment, 76
  - getAngularAcceleraton, 76
  - getAngularVelocity, 77
  - getLinearAcceleration, 77
  - getLinearVelocity, 77
  - getOrientation, 77
  - getPosition, 78
  - getRnb, 78
  - getTime, 78
  - getWorldAngularVelocity, 78
  - getWorldLinearVelocity, 79
  - sensors, 79
  - sensorsVec3d, 79
  - updateSensors, 79
- environment.cpp
  - connectConflateSocket, 183
  - r\_nb, 183, 184
  - recvVectors, 184
- error
  - Sensor< T >, 119
- Euler
  - ODE, 98
- exitController
  - ControlSystem, 65
- exiting
  - status.hpp, 165
- FACRO
  - controller\_mode.hpp, 168
- factory
  - ODE, 98
- FANGLE
  - controller\_mode.hpp, 168
- figure
  - controller\_plots.m, 148
- FMANUAL
  - controller\_mode.hpp, 168
- folderPath
  - controller\_plots.m, 148
- forceCoff
  - Rotor, 117
- fromString
  - ODE, 100
- g
  - Accelerometer, 16
- gen
  - Sensor< T >, 121
- getAmmount
  - Load, 89
- getAngularAcceleraton
  - Environment, 76
- getAngularVelocity
  - Environment, 77
  - NS, 95
- getCoefficients
  - ControlSurfaces, 63
- getGyroBias
  - AHRS, 19
  - AHRS\_EKF, 24
- getLastThrust
  - Jet, 86
- getLinearAcceleration
  - Environment, 77
- getLinearVelocity
  - Environment, 77
  - NS, 96
- getMass
  - Load, 89
- getMethodsToTest
  - controller\_test.cpp, 153
  - ode\_test.cpp, 160
- getMicrosteps
  - ODE, 100
- getMode
  - ControllerLoop, 43
- getNoOfSurface
  - ControlSurfaces, 63
- getOffset
  - Load, 89
- getOri
  - AHRS, 19
- getOrientation
  - Environment, 77
  - NS, 96
- getPos
  - EKF, 71
- getPosition
  - Environment, 78

- NS, 96
- getReading
  - Sensor< T >, 119
- getRnb
  - Environment, 78
- getRot
  - Hinge, 84
- getRotationMatrixBodyToWorld
  - NS, 96
- getRotorHoverSpeeds
  - UAVparams, 127
- getRotorMaxSpeeds
  - UAVparams, 127
- getRotorTimeConstants
  - UAVparams, 127
- getSd
  - Sensor< T >, 120
- getSingleton
  - Params, 110
  - UAVparams, 127
- getThrust
  - Jet, 87
- getTime
  - Environment, 78
- getV0
  - Ammo, 27
- getValues
  - ControlSurfaces, 64
- getVel
  - EKF, 71
- getWorldAngularVelocity
  - Environment, 78
- getWorldLinearVelocity
  - Environment, 79
- go
  - TimedLoop, 125
- GPS, 80
  - GPS, 80
  - update, 81
- GPSVel, 81
  - GPSVel, 81
  - update, 82
- Gyroscope, 82
  - Gyroscope, 83
  - update, 83
- handleJoystick
  - ControllerLoop, 43
  - ControllerLoopFACRO, 46
  - ControllerLoopFANGLE, 47
  - ControllerLoopFMANUAL, 49
  - ControllerLoopQACRO, 51
  - ControllerLoopQANGLE, 52
  - ControllerLoopQPOS, 54
  - ControllerLoopRANGLE, 56
  - ControllerLoopRMANUAL, 61
- handleMsg
  - Control, 34
- Heun
  - ODE, 98
- HEX
  - CMakeCCompilerId.c, 138
  - CMakeCXXCompilerId.cpp, 141
- Hinge, 83
  - getRot, 84
  - Hinge, 84
  - operator=, 84
  - updateValue, 85
- hinge.cpp
  - asSkewMatrix, 150
- hinges
  - Drive, 69
- hoverSpeed
  - Rotor, 117
- i
  - controller\_plots.m, 148
- idle
  - status.hpp, 165
- info\_arch
  - CMakeCCompilerId.c, 139
  - CMakeCXXCompilerId.cpp, 142
- info\_compiler
  - CMakeCCompilerId.c, 139
  - CMakeCXXCompilerId.cpp, 143
- info\_language\_extensions\_default
  - CMakeCCompilerId.c, 139
  - CMakeCXXCompilerId.cpp, 143
- info\_language\_standard\_default
  - CMakeCCompilerId.c, 140
  - CMakeCXXCompilerId.cpp, 143
- INFO\_PERIOD
  - def, 13
- info\_platform
  - CMakeCCompilerId.c, 140
  - CMakeCXXCompilerId.cpp, 143
- initialMode
  - UAVparams, 128
- initialOrientation
  - UAVparams, 129
- initialPosition
  - UAVparams, 129
- initialVelocity
  - UAVparams, 129
- INstantiate\_TEST\_SUITE\_P
  - controller\_test.cpp, 153
  - ode\_test.cpp, 160
- instantRun
  - UAVparams, 129
- isReady
  - Sensor< T >, 120
- lx
  - UAVparams, 129
- lxy
  - UAVparams, 129
- lxz
  - UAVparams, 129
- ly

- UAVparams, 129
- lyz
  - UAVparams, 130
- lz
  - UAVparams, 130
- Jet, 86
  - getLastThrust, 86
  - getThrust, 87
  - phases, 87
  - start, 87
  - thrust, 88
  - time, 88
- jets
  - UAVparams, 130
- job
  - ControllerLoop, 43
  - ControllerLoopFACRO, 46
  - ControllerLoopFANGLE, 47
  - ControllerLoopFMANUAL, 49
  - ControllerLoopQACRO, 51
  - ControllerLoopQANGLE, 52
  - ControllerLoopQPOS, 54
  - ControllerLoopRANGLE, 56
  - ControllerLoopRAUTOLAUNCH, 58
  - ControllerLoopRGUIDED, 59
  - ControllerLoopRMANUAL, 61
- lastUpdate
  - Sensor< T >, 121
- legend
  - controller\_plots.m, 146
- lib/UAV\_common/header/common.hpp, 145
- lib/UAV\_common/scripts/controller\_plots.m, 146
- lib/UAV\_common/src/components/aero\_coefficients.hpp, 149
- lib/UAV\_common/src/components/components.hpp, 149
- lib/UAV\_common/src/components/control\_surfaces.cpp, 149
- lib/UAV\_common/src/components/control\_surfaces.hpp, 149
- lib/UAV\_common/src/components/drive.cpp, 150
- lib/UAV\_common/src/components/drive.hpp, 150
- lib/UAV\_common/src/components/hinge.cpp, 150
- lib/UAV\_common/src/components/hinge.hpp, 150
- lib/UAV\_common/src/components/loads.cpp, 151
- lib/UAV\_common/src/components/loads.hpp, 151
- lib/UAV\_common/src/components/navi.hpp, 151
- lib/UAV\_common/src/controllers/controller.cpp, 151
- lib/UAV\_common/src/controllers/controller.hpp, 152
- lib/UAV\_common/src/controllers/controller\_test.cpp, 152
- lib/UAV\_common/src/controllers/impl/bang\_bang.cpp, 154
- lib/UAV\_common/src/controllers/impl/bang\_bang.hpp, 154
- lib/UAV\_common/src/controllers/impl/double\_setpoint.cpp, 155
- lib/UAV\_common/src/controllers/impl/double\_setpoint.hpp, 155
- lib/UAV\_common/src/controllers/impl/PID.cpp, 156
- lib/UAV\_common/src/controllers/impl/PID.hpp, 156
- lib/UAV\_common/src/controllers/impl/PID\_discrete.cpp, 156
- lib/UAV\_common/src/controllers/impl/PID\_discrete.hpp, 156
- lib/UAV\_common/src/controllers/impl/z\_trans.cpp, 157
- lib/UAV\_common/src/controllers/impl/z\_trans.hpp, 157
- lib/UAV\_common/src/logger/logger.cpp, 158
- lib/UAV\_common/src/logger/logger.hpp, 158
- lib/UAV\_common/src/ode/ode.cpp, 159
- lib/UAV\_common/src/ode/ode.hpp, 159
- lib/UAV\_common/src/ode/ode\_impl.hpp, 159
- lib/UAV\_common/src/ode/ode\_test.cpp, 159
- lib/UAV\_common/src/parser/parser.cpp, 161
- lib/UAV\_common/src/parser/parser.hpp, 163
- lib/UAV\_common/src/parser/uav\_params.cpp, 164
- lib/UAV\_common/src/parser/uav\_params.hpp, 164
- lib/UAV\_common/src/timed\_loop/status.hpp, 165
- lib/UAV\_common/src/timed\_loop/timed\_loop.cpp, 165
- lib/UAV\_common/src/timed\_loop/timed\_loop.hpp, 165
- Load, 88
  - getAmmount, 89
  - getMass, 89
  - getOffset, 89
  - Load, 89
  - operator=, 90
  - release, 90
- loadConfig
  - UAVparams, 127
- log
  - EKF, 71
  - Logger, 92
- log\_path
  - main.cpp, 179
- Logger, 90
  - ~Logger, 91
  - log, 92
  - Logger, 91
  - setFmt, 92
  - setLogDirectory, 92
- logger
  - AHRS, 20
  - Sensor< T >, 121
- logger.cpp
  - shouldLog, 158
- logger.hpp
  - LOGGER\_MASK, 158
- LOGGER\_MASK
  - logger.hpp, 158
  - main.cpp, 178
- m
  - UAVparams, 130
- Magnetometer, 94
- Magnetometer, 93

- mag, [94](#)
- Magnetometer, [93](#)
- update, [94](#)
- main
  - CMakeCCompilerId.c, [139](#)
  - CMakeCXXCompilerId.cpp, [142](#)
  - controller\_test.cpp, [153](#)
  - main.cpp, [179](#)
  - ode\_test.cpp, [160](#)
- main.cpp
  - log\_path, [179](#)
  - LOGGER\_MASK, [178](#)
  - main, [179](#)
  - parseArgs, [179](#)
- maxSpeed
  - Rotor, [117](#)
- mixers.cpp
  - applyMixerRotors, [170](#)
  - applyMixerRotorsHover, [170](#)
  - applyMixerSurfaces, [171](#)
- mixers.hpp
  - applyMixerRotors, [172](#)
  - applyMixerRotorsHover, [172](#)
  - applyMixerSurfaces, [172](#)
- mtxOri
  - AHRS, [20](#)
- name
  - SensorParams, [123](#)
  - UAVparams, [130](#)
- NONE
  - controller\_mode.hpp, [168](#)
  - controllers::PID, [112](#)
  - ODE, [98](#)
- noOfAmmo
  - UAVparams, [130](#)
- noOfCargo
  - UAVparams, [130](#)
- noOfHinges
  - Drive, [69](#)
- noOfJets
  - UAVparams, [130](#)
- noOfRotors
  - UAVparams, [131](#)
- NS, [94](#)
  - ~NS, [95](#)
  - getAngularVelocity, [95](#)
  - getLinearVelocity, [96](#)
  - getOrientation, [96](#)
  - getPosition, [96](#)
  - getRotationMatrixBodyToWorld, [96](#)
  - NS, [95](#)
- ODE, [97](#)
  - ~ODE, [98](#)
  - Euler, [98](#)
  - factory, [98](#)
  - fromString, [100](#)
  - getMicrosteps, [100](#)
  - Heun, [98](#)
  - NONE, [98](#)
  - ODE, [98](#)
  - ODEMethod, [98](#)
  - PC2, [98](#)
  - PC4, [98](#)
  - RK4, [98](#)
  - step, [101](#)
  - ODE\_Euler, [101](#)
    - ODE\_Euler, [102](#)
    - step, [102](#)
  - ODE\_Heun, [103](#)
    - ODE\_Heun, [103](#)
    - step, [103](#)
  - ODE\_PC2, [104](#)
    - ODE\_PC2, [104](#)
    - step, [105](#)
  - ODE\_PC4, [105](#)
    - ODE\_PC4, [106](#)
    - step, [106](#)
  - ODE\_RK4, [107](#)
    - ODE\_RK4, [107](#)
    - step, [107](#)
  - ode\_test.cpp
    - getMethodsToTest, [160](#)
    - INstantiate\_Test\_Suite\_P, [160](#)
    - main, [160](#)
    - TEST\_F, [160](#)
    - TEST\_P, [161](#)
  - ODEMethod
    - ODE, [98](#)
  - ODETest, [108](#)
    - SetUp, [108](#)
    - TearDown, [108](#)
- off
  - controller\_plots.m, [148](#)
- on
  - controller\_plots.m, [148](#)
- operator=
  - Ammo, [28](#)
  - Hinge, [84](#)
  - Load, [90](#)
  - Params, [110](#)
- orderServerJob
  - control.cpp, [166](#)
- ori\_est
  - AHRS, [20](#)
- overridePositionAndSpeed
  - ControllerLoop, [44](#)
  - ControllerLoopFANGLE, [47](#)
  - ControllerLoopQANGLE, [52](#)
  - ControllerLoopQPOS, [54](#)
  - ControllerLoopRANGLE, [56](#)
- P
  - AHRS\_EKF, [25](#)
- P0
  - EKFParams, [73](#)
- Params, [109](#)

- ~Params, 110
  - getSingleton, 110
  - operator=, 110
  - Params, 109, 110
  - STEP\_TIME, 110
- parseArgs
  - main.cpp, 179
- parseHinge
  - uav\_params.cpp, 164
- parseMatrixXd
  - parser.cpp, 162
  - parser.hpp, 163
- parser.cpp
  - parseMatrixXd, 162
  - parseVectorXd, 162
- parser.hpp
  - parseMatrixXd, 163
  - parseVectorXd, 163
- parseVectorXd
  - parser.cpp, 162
  - parser.hpp, 163
- PC2
  - ODE, 98
- PC4
  - ODE, 98
- phases
  - Jet, 87
- PID
  - controllers::PID, 112
- PID\_Discrete
  - controllers::PID\_Discrete, 114, 115
- PLATFORM\_ID
  - CMakeCCompilerId.c, 138
  - CMakeCXXCompilerId.cpp, 142
- plot
  - controller\_plots.m, 146
  - controller\_test.cpp, 154
- plot\_directory\_name
  - controller\_test.cpp, 154
- position
  - Drive, 69
- predict
  - EKF, 71
- predictScaler
  - EKFScalers, 75
- prepare
  - Control, 34
- Q
  - AHRS\_EKF, 25
  - AHRSParams, 26
  - EKFParams, 73
- q
  - AHRS\_EKF, 24
- QACRO
  - controller\_mode.hpp, 168
- QANGLE
  - controller\_mode.hpp, 168
- QPOS
  - controller\_mode.hpp, 168
- quaternionToRPY
  - AHRS\_EKF, 24
- R
  - AHRS\_EKF, 25
  - AHRSParams, 26
- r\_nb
  - environment.cpp, 183, 184
- RANGLE
  - controller\_mode.hpp, 168
- RAUTOLAUNCH
  - controller\_mode.hpp, 168
- RBaro
  - EKFParams, 73
- ready
  - Sensor< T >, 121
- recv
  - Control, 34
- recvVectors
  - environment.cpp, 184
- refreshTime
  - Sensor< T >, 122
  - SensorParams, 123
- release
  - Load, 90
- reload
  - status.hpp, 165
- required\_controllers
  - ControllerLoop, 45
- requiredcontrollers
  - ControllerLoop, 44
- restoreTrim
  - ControlSurfaces, 64
- RGPSPos
  - EKFParams, 74
- RGPSVel
  - EKFParams, 74
- RGUIDED
  - controller\_mode.hpp, 168
- RK4
  - ODE, 98
- RMANUAL
  - controller\_mode.hpp, 168
- rot\_bw
  - AHRS, 19
  - AHRS\_complementary, 22
  - AHRS\_EKF, 24
- Rotor, 116
  - direction, 117
  - forceCoff, 117
  - hoverSpeed, 117
  - maxSpeed, 117
  - timeConstant, 117
  - torqueCoff, 117
- rotorMixer
  - UAVparams, 131
- rotors
  - UAVparams, 131

- RPYToQuaterion
  - AHRS\_EKF, [24](#)
- run
  - ControlSystem, [65](#)
- running
  - status.hpp, [165](#)
- S
  - AeroCoefficients, [17](#)
  - AHRS\_EKF.cpp, [182](#)
- safeGet
  - utils.hpp, [187](#)
- safeSet
  - utils.hpp, [188](#)
- sd
  - SensorParams, [123](#)
- sendHinge
  - Control, [34](#)
- sendSpeed
  - Control, [36](#)
- sendSurface
  - Control, [36](#)
- Sensor
  - Sensor< T >, [119](#)
- Sensor< T >, [118](#)
  - bias, [121](#)
  - dist, [121](#)
  - env, [121](#)
  - error, [119](#)
  - gen, [121](#)
  - getReading, [119](#)
  - getSd, [120](#)
  - isReady, [120](#)
  - lastUpdate, [121](#)
  - logger, [121](#)
  - ready, [121](#)
  - refreshTime, [122](#)
  - Sensor, [119](#)
  - shouldUpdate, [120](#)
  - update, [120](#)
  - value, [122](#)
- SensorParams, [122](#)
  - bias, [122](#)
  - name, [123](#)
  - refreshTime, [123](#)
  - sd, [123](#)
- sensors
  - Environment, [79](#)
  - UAVparams, [131](#)
- sensorsVec3d
  - Environment, [79](#)
- set\_dt
  - Controller, [40](#)
  - controllers::PID\_Discrete, [116](#)
- setFmt
  - Logger, [92](#)
- setLogDirectory
  - Logger, [92](#)
- setMode
  - Control, [36](#)
  - ControlSystem, [66](#)
- SetUp
  - ControllerTest, [62](#)
  - ODETest, [108](#)
- setValues
  - ControlSurfaces, [64](#)
- shouldLog
  - logger.cpp, [158](#)
- shouldUpdate
  - Sensor< T >, [120](#)
- splitStringToDoubleVector
  - z\_trans.cpp, [157](#)
- src/communication/control.cpp, [166](#)
- src/communication/control.hpp, [166](#)
- src/communication/control\_recv.cpp, [167](#)
- src/communication/control\_send.cpp, [167](#)
- src/controller/controller.cpp, [152](#)
- src/controller/controller.hpp, [152](#)
- src/controller/controller\_loop.cpp, [167](#)
- src/controller/controller\_loop.hpp, [167](#)
- src/controller/controller\_mode.hpp, [168](#)
- src/controller/mixers.cpp, [169](#)
- src/controller/mixers.hpp, [171](#)
- src/controller/modes/controller\_loop\_FACRO.cpp, [173](#)
- src/controller/modes/controller\_loop\_FACRO.hpp, [173](#)
- src/controller/modes/controller\_loop\_FANGLE.cpp, [173](#)
- src/controller/modes/controller\_loop\_FANGLE.hpp, [174](#)
- src/controller/modes/controller\_loop\_FMANUAL.cpp, [174](#)
- src/controller/modes/controller\_loop\_FMANUAL.hpp, [174](#)
- src/controller/modes/controller\_loop\_NONE.cpp, [174](#)
- src/controller/modes/controller\_loop\_NONE.hpp, [174](#)
- src/controller/modes/controller\_loop\_QACRO.cpp, [175](#)
- src/controller/modes/controller\_loop\_QACRO.hpp, [175](#)
- src/controller/modes/controller\_loop\_QANGLE.cpp, [175](#)
- src/controller/modes/controller\_loop\_QANGLE.hpp, [175](#)
- src/controller/modes/controller\_loop\_QPOS.cpp, [175](#)
- src/controller/modes/controller\_loop\_QPOS.hpp, [175](#)
- src/controller/modes/controller\_loop\_RANGLE.cpp, [176](#)
- src/controller/modes/controller\_loop\_RANGLE.hpp, [176](#)
- src/controller/modes/controller\_loop\_RAUTOLAUNCH.cpp, [176](#)
- src/controller/modes/controller\_loop\_RAUTOLAUNCH.hpp, [176](#)
- src/controller/modes/controller\_loop\_RGUIDED.cpp, [176](#)
- src/controller/modes/controller\_loop\_RGUIDED.hpp, [177](#)
- src/controller/modes/controller\_loop\_RMANUAL.cpp, [177](#)
- src/controller/modes/controller\_loop\_RMANUAL.hpp, [177](#)
- src/defines.hpp, [177](#)
- src/main.cpp, [178](#)
- src/navigation/AHRS.cpp, [180](#)
- src/navigation/AHRS.hpp, [180](#)

- src/navigation/AHRS/AHRS\_complementary.cpp, 180
- src/navigation/AHRS/AHRS\_complementary.hpp, 181
- src/navigation/AHRS/AHRS\_EKF.cpp, 181
- src/navigation/AHRS/AHRS\_EKF.hpp, 182
- src/navigation/EKF.cpp, 182
- src/navigation/EKF.hpp, 182
- src/navigation/environment.cpp, 183
- src/navigation/environment.hpp, 184
- src/navigation/NS.cpp, 184
- src/navigation/NS.hpp, 185
- src/navigation/sensors.cpp, 185
- src/navigation/sensors.hpp, 185
- src/params.cpp, 186
- src/params.hpp, 186
- src/utlis.hpp, 186
- stallLimit
  - AeroCoefficients, 17
- start
  - Control, 36
  - Jet, 87
- startJet
  - Control, 36
- Status
  - status.hpp, 165
- status.hpp
  - exiting, 165
  - idle, 165
  - reload, 165
  - running, 165
  - Status, 165
- step
  - ODE, 101
  - ODE\_Euler, 102
  - ODE\_Heun, 103
  - ODE\_PC2, 105
  - ODE\_PC4, 106
  - ODE\_RK4, 107
- STEP\_TIME
  - Params, 110
- stop
  - Control, 37
- STRINGIFY
  - CMakeCCompilerId.c, 139
  - CMakeCXXCompilerId.cpp, 142
- STRINGIFY\_HELPER
  - CMakeCCompilerId.c, 139
  - CMakeCXXCompilerId.cpp, 142
- surfaceMixer
  - UAVparams, 131
- surfaces
  - UAVparams, 131
- target
  - ControllerLoopRGUIDED, 59
  - UAVparams, 131
- TearDown
  - ControllerTest, 62
  - ODETest, 108
- TEST\_F
  - ode\_test.cpp, 160
- TEST\_P
  - controller\_test.cpp, 153, 154
  - ode\_test.cpp, 161
- thrust
  - Jet, 88
- time
  - Jet, 88
- timeConstant
  - Rotor, 117
- TimedLoop, 123
  - go, 125
  - TimedLoop, 124
- title
  - controller\_plots.m, 147
- torqueCoff
  - Rotor, 117
- type
  - AHRSPParams, 26
- uav\_params.cpp
  - parseHinge, 164
- UAVparams, 125
  - ~UAVparams, 127
  - aero\_coffs, 128
  - ahrs, 128
  - ammo, 128
  - cargo, 128
  - controllers, 128
  - ekf, 128
  - getRotorHoverSpeeds, 127
  - getRotorMaxSpeeds, 127
  - getRotorTimeContants, 127
  - getSingleton, 127
  - initialMode, 128
  - initialOrientation, 129
  - initialPosition, 129
  - initialVelocity, 129
  - instantRun, 129
  - lx, 129
  - lxy, 129
  - lxz, 129
  - ly, 129
  - lyz, 130
  - lz, 130
  - jets, 130
  - loadConfig, 127
  - m, 130
  - name, 130
  - noOfAmmo, 130
  - noOfCargo, 130
  - noOfJets, 130
  - noOfRotors, 131
  - rotorMixer, 131
  - rotors, 131
  - sensors, 131
  - surfaceMixer, 131
  - surfaces, 131
  - target, 131



- UAVparams, [127](#)
- update
  - Accelerometer, [16](#)
  - AHRS, [20](#)
  - AHRS\_complementary, [22](#)
  - AHRS\_EKF, [25](#)
  - Barometer, [31](#)
  - GPS, [81](#)
  - GPSVel, [82](#)
  - Gyroscope, [83](#)
  - Magnetometer, [94](#)
  - Sensor< T >, [120](#)
- updateBaro
  - EKF, [72](#)
- updateGPS
  - EKF, [72](#)
- updateGPSVel
  - EKF, [72](#)
- updateScaler
  - EKFScalers, [75](#)
- updateSensors
  - Environment, [79](#)
- updateValue
  - Hinge, [85](#)
- USE\_QUATERIONS
  - defines.hpp, [178](#)
- utils.hpp
  - circularError, [186](#)
  - clampAngle, [187](#)
  - safeGet, [187](#)
  - safeSet, [188](#)
- value
  - Sensor< T >, [122](#)
- x
  - AHRS\_EKF, [25](#)
  - controller\_plots.m, [148](#)
- xlabel
  - controller\_plots.m, [147](#)
- y
  - controller\_plots.m, [148](#)
- ylabel
  - controller\_plots.m, [147](#)
- z\_trans.cpp
  - splitStringToDoubleVector, [157](#)
- zScaler
  - EKFScalers, [75](#)
- ZTransform
  - controllers::ZTransform, [132](#), [133](#)
- ZTransformStatic
  - controllers::ZTransformStatic< N, D >, [134](#), [135](#)