

Bebop Controller Package

Generated by Doxygen 1.9.3

1 Bebop Controller Package	1
2 Installation Instructions	3
3 ROS Nodes	5
4 Launch Files	7
5 YAML Files	9
6 Namespace Index	11
6.1 Namespace List	11
7 Hierarchical Index	13
7.1 Class Hierarchy	13
8 Class Index	15
8.1 Class List	15
9 File Index	17
9.1 File List	17
10 Namespace Documentation	19
10.1 bebop_controller Namespace Reference	19
10.1.1 Detailed Description	20
10.1.2 Function Documentation	20
10.1.2.1 bound()	20
10.1.2.2 clamp()	20
10.1.2.3 GetRosParameter()	21
10.1.2.4 max()	21
10.1.2.5 Quat2RPY()	21
10.1.2.6 RPY2Quat()	22
10.1.2.7 sgn()	22
10.1.2.8 yawFromQuaternion()	22
10.2 gazebo Namespace Reference	23
10.2.1 Detailed Description	23
10.3 plot Namespace Reference	23
10.3.1 Detailed Description	23
10.3.2 Variable Documentation	23
10.3.2.1 path	23
10.3.2.2 plt	24
10.3.2.3 sub	24
10.3.2.4 topic	24
10.3.2.5 yaml	24
11 Class Documentation	25

11.1 bebop_controller::BaseController Class Reference	25
11.1.1 Detailed Description	26
11.1.2 Constructor & Destructor Documentation	26
11.1.2.1 BaseController()	26
11.1.2.2 ~BaseController()	26
11.1.3 Member Function Documentation	27
11.1.3.1 CalculateCommandVelocities()	27
11.1.3.2 CalculateLeashLength()	27
11.1.3.3 CheckSafeZone()	27
11.1.3.4 EstimateAcceleration()	27
11.1.3.5 EstimateVelocity()	27
11.1.3.6 GetErrors()	28
11.1.3.7 GetVelocityErrors()	28
11.1.3.8 Land()	28
11.1.3.9 LimitPositionErrors()	28
11.1.3.10 MultiDOFJointTrajectory_CB()	28
11.1.3.11 Odometry_CB()	28
11.1.3.12 Quaternion2Euler()	29
11.1.3.13 SetOdometry()	29
11.1.3.14 SetTrajectoryPoint()	29
11.1.3.15 Stop()	29
11.1.3.16 Stop_CB()	29
11.1.3.17 TakeOff()	29
11.1.3.18 TimeOut_CB()	30
11.1.4 Member Data Documentation	30
11.1.4.1 cmd_multi_dof_joint_trajectory_sub_	30
11.1.4.2 command_trajectory_	30
11.1.4.3 controller_active_	30
11.1.4.4 diff	30
11.1.4.5 disable_commands	30
11.1.4.6 end_sub_	31
11.1.4.7 land_pub_	31
11.1.4.8 last_state	31
11.1.4.9 lastTime	31
11.1.4.10 leash_length	31
11.1.4.11 max_speed	31
11.1.4.12 motor_velocity_reference_pub_	32
11.1.4.13 odom_sub_	32
11.1.4.14 odometry_	32
11.1.4.15 odometry_filtered_pub_	32
11.1.4.16 reference_angles_pub_	32
11.1.4.17 safe_zone	32

11.1.4.18 smoothed_reference_pub_	33
11.1.4.19 state	33
11.1.4.20 stop	33
11.1.4.21 takeoff	33
11.1.4.22 takeoff_pub_	33
11.1.4.23 timeOut	33
11.1.4.24 waypointHasBeenPublished_	34
11.2 bebop_controller::CITCController Class Reference	34
11.2.1 Detailed Description	34
11.2.2 Constructor & Destructor Documentation	34
11.2.2.1 CITCController() [1/2]	34
11.2.2.2 CITCController() [2/2]	35
11.3 bebop_controller::Command_Velocities Struct Reference	35
11.3.1 Detailed Description	35
11.3.2 Member Data Documentation	35
11.3.2.1 x	35
11.3.2.2 y	36
11.3.2.3 yaw	36
11.3.2.4 z	36
11.4 bebop_controller::DataToCSV Class Reference	36
11.4.1 Detailed Description	36
11.4.2 Constructor & Destructor Documentation	36
11.4.2.1 DataToCSV()	37
11.4.2.2 ~DataToCSV()	37
11.5 bebop_controller::DataToCSVParameters Struct Reference	37
11.5.1 Detailed Description	37
11.5.2 Member Data Documentation	37
11.5.2.1 File	38
11.5.2.2 Initial_Time	38
11.5.2.3 Margin_Time	38
11.5.2.4 Topic_CMD_Vel	38
11.5.2.5 Topic_CSV_Begin	38
11.5.2.6 Topic_CSV_End	38
11.5.2.7 Topic_Drone_Pose	39
11.5.2.8 Topic_Reference_Pose	39
11.5.2.9 Topic_Velocities	39
11.6 bebop_controller::Normalize Struct Reference	39
11.6.1 Detailed Description	39
11.6.2 Member Data Documentation	39
11.6.2.1 angle	40
11.6.2.2 horizontal	40
11.6.2.3 rotation	40

11.6.2.4 vertical	40
11.7 bebop_controller::PIDController Class Reference	40
11.7.1 Detailed Description	41
11.7.2 Constructor & Destructor Documentation	41
11.7.2.1 PIDController() [1/2]	41
11.7.2.2 PIDController() [2/2]	41
11.8 plot.Plots Class Reference	41
11.8.1 Detailed Description	42
11.8.2 Constructor & Destructor Documentation	42
11.8.2.1 __init__()	42
11.8.3 Member Function Documentation	42
11.8.3.1 Plot()	42
11.8.4 Member Data Documentation	42
11.8.4.1 path	42
11.8.4.2 yaml	43
11.9 bebop_controller::ProportionalController Class Reference	43
11.9.1 Detailed Description	43
11.9.2 Constructor & Destructor Documentation	43
11.9.2.1 ProportionalController()	43
11.10 bebop_controller::Sinusoidal Class Reference	44
11.10.1 Detailed Description	44
11.10.2 Constructor & Destructor Documentation	44
11.10.2.1 Sinusoidal()	44
11.11 gazebo.SphinxPublisher Class Reference	44
11.11.1 Detailed Description	45
11.11.2 Constructor & Destructor Documentation	45
11.11.2.1 __init__()	45
11.11.3 Member Function Documentation	45
11.11.3.1 process_output()	45
11.11.3.2 publish()	46
11.11.4 Member Data Documentation	46
11.11.4.1 att	46
11.11.4.2 cont	46
11.11.4.3 odom_pub	46
11.11.4.4 pos	46
11.11.4.5 q	47
11.12 bebop_controller::SquareRootController Class Reference	47
11.12.1 Detailed Description	47
11.12.2 Constructor & Destructor Documentation	47
11.12.2.1 SquareRootController()	47
11.13 bebop_controller::State Struct Reference	48
11.13.1 Detailed Description	48

11.13.2 Member Data Documentation	48
11.13.2.1 acceleration	48
11.13.2.2 angular_acceleration	48
11.13.2.3 angular_velocity	48
11.13.2.4 orientation	49
11.13.2.5 position	49
11.13.2.6 velocity	49
11.14 TrajectoryParameters Struct Reference	49
11.14.1 Detailed Description	49
11.14.2 Member Data Documentation	49
11.14.2.1 PositionBeforeTrajectory	50
11.14.2.2 TrajectoryDistance	50
11.14.2.3 Yaw_Enabled	50
11.14.2.4 Yaw_Offset	50
11.15 bebop_controller::Vector3 Struct Reference	50
11.15.1 Detailed Description	51
11.15.2 Member Data Documentation	51
11.15.2.1 x	51
11.15.2.2 y	51
11.15.2.3 z	51
11.16 bebop_controller::Vector4 Struct Reference	51
11.16.1 Detailed Description	52
11.16.2 Member Data Documentation	52
11.16.2.1 x	52
11.16.2.2 y	52
11.16.2.3 yaw	52
11.16.2.4 z	52
11.17 bebop_controller::Waypoint Class Reference	53
11.17.1 Detailed Description	53
11.17.2 Constructor & Destructor Documentation	53
11.17.2.1 Waypoint()	54
11.17.2.2 ~Waypoint()	54
11.17.3 Member Function Documentation	54
11.17.3.1 Start_CB()	54
11.17.3.2 Stop_CB()	54
11.17.3.3 Trajectory_CB()	54
11.17.4 Member Data Documentation	54
11.17.4.1 _n1	55
11.17.4.2 _n2	55
11.17.4.3 _n3	55
11.17.4.4 _timer1	55
11.17.4.5 _timer2	55

11.17.4.6 _timer3	55
11.17.4.7 Begin_	56
11.17.4.8 End_	56
11.17.4.9 Initial_Time	56
11.17.4.10 nh	56
11.17.4.11 position_target_	56
11.17.4.12 setpoint_pub_	56
11.17.4.13 status	57
11.17.4.14 wp_params_	57
11.18 WaypointParameters Struct Reference	57
11.18.1 Detailed Description	57
11.18.2 Member Data Documentation	57
11.18.2.1 DiffTime	57
11.18.2.2 MarginTime	58
11.18.2.3 TimeBeforeTrajectory	58
11.18.2.4 topic_command_trajectory	58
11.18.2.5 topic_csv_begin	58
11.18.2.6 topic_csv_end	58
11.18.2.7 TrajectoryTime	58
12 File Documentation	59
12.1 documentation.md File Reference	59
12.2 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/include/bebop_controller/base_↔ _controller.h File Reference	59
12.2.1 Detailed Description	60
12.2.2 Macro Definition Documentation	60
12.2.2.1 BEBOP_CMD_VEL	60
12.2.2.2 BEBOP_COMMAND_TRAJECTORY	60
12.2.2.3 BEBOP_LAND	61
12.2.2.4 BEBOP_POSE	61
12.2.2.5 BEBOP_TAKEOFF	61
12.2.2.6 CSV_END	61
12.2.2.7 DISABLE_COMMANDS	61
12.2.2.8 GRAVITY	61
12.2.2.9 LIMIT_X	62
12.2.2.10 LIMIT_Y	62
12.2.2.11 LIMIT_YAW	62
12.2.2.12 LIMIT_Z	62
12.2.2.13 MIN_ACCEL	62
12.2.2.14 MIN_VEL	62
12.2.2.15 SAFE_ZONE_X	63
12.2.2.16 SAFE_ZONE_Y	63
12.2.2.17 SAFE_ZONE_Z	63

12.3	base_controller.h	63
12.4	C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/include/bebop_controller/common.h	
	File Reference	64
12.4.1	Detailed Description	65
12.4.2	Function Documentation	65
12.4.2.1	secsFromHeaderMsg()	65
12.4.2.2	vector4FromQuaternionMsg()	66
12.5	common.h	66
12.6	C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/include/bebop_controller/waypoint.h	
	File Reference	67
12.6.1	Detailed Description	68
12.6.2	Enumeration Type Documentation	68
12.6.2.1	Status	68
12.7	waypoint.h	68
12.8	C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/citc_controller_↔	
	angles.launch File Reference	69
12.8.1	Detailed Description	69
12.9	citc_controller_angles.launch	69
12.10	C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/citc_controller_↔	
	twist.launch File Reference	70
12.10.1	Detailed Description	70
12.11	citc_controller_twist.launch	70
12.12	C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/gazebo.launch	
	File Reference	71
12.12.1	Detailed Description	71
12.13	gazebo.launch	71
12.14	C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/pid_controller_↔	
	angles.launch File Reference	71
12.14.1	Detailed Description	71
12.15	pid_controller_angles.launch	72
12.16	C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/pid_controller_↔	
	twist.launch File Reference	72
12.16.1	Detailed Description	72
12.17	pid_controller_twist.launch	72
12.18	C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/proportional_↔	
	controller.launch File Reference	73
12.18.1	Detailed Description	73
12.19	proportional_controller.launch	73
12.20	C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/square_root_↔	
	controller.launch File Reference	74
12.20.1	Detailed Description	74
12.21	square_root_controller.launch	74
12.22	C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/vrpn_client_↔	
	ros.launch File Reference	75
12.22.1	Detailed Description	75

12.23	vrpn_client_ros.launch	75
12.24	C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/citc_controller_↔ angles.yaml File Reference	75
12.24.1	Detailed Description	75
12.25	citc_controller_angles.yaml	76
12.26	C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/citc_controller_↔ twist.yaml File Reference	76
12.26.1	Detailed Description	76
12.27	citc_controller_twist.yaml	77
12.28	C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/max_speed.yaml File Reference	77
12.28.1	Detailed Description	77
12.29	max_speed.yaml	77
12.30	C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/normalize_↔ angles.yaml File Reference	77
12.30.1	Detailed Description	78
12.31	normalize_angles.yaml	78
12.32	C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/normalize_↔ twist.yaml File Reference	78
12.32.1	Detailed Description	78
12.33	normalize_twist.yaml	78
12.34	C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/pid_controller_↔ angles.yaml File Reference	78
12.34.1	Detailed Description	78
12.35	pid_controller_angles.yaml	79
12.36	C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/pid_controller_↔ twist.yaml File Reference	79
12.36.1	Detailed Description	79
12.37	pid_controller_twist.yaml	79
12.38	C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/proportional_↔ controller.yaml File Reference	80
12.38.1	Detailed Description	80
12.39	proportional_controller.yaml	80
12.40	C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/safe_zone.yaml File Reference	80
12.40.1	Detailed Description	80
12.41	safe_zone.yaml	80
12.42	C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/square_root_↔ controller.yaml File Reference	80
12.42.1	Detailed Description	81
12.43	square_root_controller.yaml	81
12.44	C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/topics.yaml File Reference	81
12.44.1	Detailed Description	81
12.45	topics.yaml	81

12.46 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/trajectory.yaml File Reference	81
12.46.1 Detailed Description	82
12.47 trajectory.yaml	82
12.48 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/waypoint.yaml File Reference	82
12.48.1 Detailed Description	82
12.49 waypoint.yaml	82
12.50 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/scripts/gazebo.py File Reference	82
12.50.1 Detailed Description	83
12.51 gazebo.py	83
12.52 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/scripts/plot.py File Reference	84
12.52.1 Detailed Description	85
12.53 plot.py	85
12.54 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/library/base_controller.cpp File Reference	86
12.54.1 Detailed Description	86
12.55 base_controller.cpp	86
12.56 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/library/waypoint.cpp File Reference	89
12.56.1 Detailed Description	90
12.57 waypoint.cpp	90
12.58 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/citc_controller_↔angles.cpp File Reference	90
12.58.1 Detailed Description	91
12.58.2 Function Documentation	91
12.58.2.1 main()	91
12.59 citc_controller_angles.cpp	91
12.60 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/citc_controller_↔angles.h File Reference	93
12.60.1 Detailed Description	94
12.60.2 Macro Definition Documentation	94
12.60.2.1 K1xDefaultValue	94
12.60.2.2 K1yawDefaultValue	94
12.60.2.3 K1yDefaultValue	94
12.60.2.4 K1zDefaultValue	95
12.60.2.5 K2xDefaultValue	95
12.60.2.6 K2yawDefaultValue	95
12.60.2.7 K2yDefaultValue	95
12.60.2.8 K2zDefaultValue	95
12.60.2.9 K3xDefaultValue	95
12.60.2.10 K3yawDefaultValue	96
12.60.2.11 K3yDefaultValue	96

12.60.2.12 K3zDefaultValue	96
12.60.2.13 K4xDefaultValue	96
12.60.2.14 K4yawDefaultValue	96
12.60.2.15 K4yDefaultValue	96
12.60.2.16 K4zDefaultValue	97
12.60.2.17 LambdaX	97
12.60.2.18 LambdaY	97
12.60.2.19 LambdaZ	97
12.60.2.20 MASS	97
12.60.2.21 MAX_ROTATION_SPEED	97
12.60.2.22 MAX_TILT_ANGLE	98
12.60.2.23 MAX_VERTICAL_SPEED	98
12.60.2.24 RGxDefaultValue	98
12.60.2.25 RGyawDefaultValue	98
12.60.2.26 RGyDefaultValue	98
12.60.2.27 RGzDefaultValue	98
12.60.2.28 Sigma	99
12.61 citc_controller_angles.h	99
12.62 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/citc_controller_↔ twist.cpp File Reference	100
12.62.1 Detailed Description	100
12.62.2 Function Documentation	100
12.62.2.1 main()	100
12.63 citc_controller_twist.cpp	101
12.64 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/citc_controller_↔ twist.h File Reference	102
12.64.1 Detailed Description	103
12.64.2 Macro Definition Documentation	103
12.64.2.1 K1xDefaultValue	103
12.64.2.2 K1yawDefaultValue	104
12.64.2.3 K1yDefaultValue	104
12.64.2.4 K1zDefaultValue	104
12.64.2.5 K2xDefaultValue	104
12.64.2.6 K2yawDefaultValue	104
12.64.2.7 K2yDefaultValue	104
12.64.2.8 K2zDefaultValue	105
12.64.2.9 K3xDefaultValue	105
12.64.2.10 K3yawDefaultValue	105
12.64.2.11 K3yDefaultValue	105
12.64.2.12 K3zDefaultValue	105
12.64.2.13 K4xDefaultValue	105
12.64.2.14 K4yawDefaultValue	106
12.64.2.15 K4yDefaultValue	106

12.64.2.16 KzDefaultValue	106
12.64.2.17 LambdaX	106
12.64.2.18 LambdaY	106
12.64.2.19 LambdaZ	106
12.64.2.20 MASS	107
12.64.2.21 MAX_HORIZONTAL_SPEED	107
12.64.2.22 MAX_ROTATION_SPEED	107
12.64.2.23 MAX_VERTICAL_SPEED	107
12.64.2.24 RGxDefaultValue	107
12.64.2.25 RGyawDefaultValue	107
12.64.2.26 RGyDefaultValue	108
12.64.2.27 RGzDefaultValue	108
12.64.2.28 Sigma	108
12.65 citc_controller_twist.h	108
12.66 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/data_to_csv.cpp	
File Reference	109
12.66.1 Detailed Description	109
12.66.2 Function Documentation	109
12.66.2.1 main()	110
12.67 data_to_csv.cpp	110
12.68 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/data_to_csv.h	
File Reference	111
12.68.1 Detailed Description	112
12.69 data_to_csv.h	112
12.70 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/pid_controller_↵ angles.cpp	File Reference
12.70.1 Detailed Description	113
12.70.2 Function Documentation	113
12.70.2.1 main()	113
12.71 pid_controller_angles.cpp	114
12.72 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/pid_controller_↵ angles.h	File Reference
12.72.1 Detailed Description	116
12.72.2 Macro Definition Documentation	116
12.72.2.1 DxDefaultValue	116
12.72.2.2 DyawDefaultValue	116
12.72.2.3 DyDefaultValue	116
12.72.2.4 DzDefaultValue	117
12.72.2.5 IxDefaultValue	117
12.72.2.6 IyawDefaultValue	117
12.72.2.7 IyDefaultValue	117
12.72.2.8 IzDefaultValue	117
12.72.2.9 LambdaX	117

12.72.2.10 LambdaY	118
12.72.2.11 LambdaZ	118
12.72.2.12 LlxDefaultValue	118
12.72.2.13 LlyawDefaultValue	118
12.72.2.14 LlyDefaultValue	118
12.72.2.15 LlzDefaultValue	118
12.72.2.16 MASS	119
12.72.2.17 MAX_ROTATION_SPEED	119
12.72.2.18 MAX_TILT_ANGLE	119
12.72.2.19 MAX_VERTICAL_SPEED	119
12.72.2.20 PxDefaultValue	119
12.72.2.21 PyawDefaultValue	119
12.72.2.22 PyDefaultValue	120
12.72.2.23 PzDefaultValue	120
12.72.2.24 RGxDefaultValue	120
12.72.2.25 RGyawDefaultValue	120
12.72.2.26 RGyDefaultValue	120
12.72.2.27 RGzDefaultValue	120
12.73 pid_controller_angles.h	121
12.74 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/pid_controller_↵ twist.cpp File Reference	121
12.74.1 Detailed Description	122
12.74.2 Function Documentation	122
12.74.2.1 main()	122
12.75 pid_controller_twist.cpp	122
12.76 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/pid_controller_↵ twist.h File Reference	123
12.76.1 Detailed Description	124
12.76.2 Macro Definition Documentation	124
12.76.2.1 DxDefaultValue	125
12.76.2.2 DyawDefaultValue	125
12.76.2.3 DyDefaultValue	125
12.76.2.4 DzDefaultValue	125
12.76.2.5 IxDefaultValue	125
12.76.2.6 IyawDefaultValue	125
12.76.2.7 IyDefaultValue	126
12.76.2.8 IzDefaultValue	126
12.76.2.9 LambdaX	126
12.76.2.10 LambdaY	126
12.76.2.11 LambdaZ	126
12.76.2.12 LlxDefaultValue	126
12.76.2.13 LlyawDefaultValue	127
12.76.2.14 LlyDefaultValue	127

12.76.2.15 LlzDefaultValue	127
12.76.2.16 MASS	127
12.76.2.17 MAX_HORIZONTAL_SPEED	127
12.76.2.18 MAX_ROTATION_SPEED	127
12.76.2.19 MAX_VERTICAL_SPEED	128
12.76.2.20 PxDefaultValue	128
12.76.2.21 PyawDefaultValue	128
12.76.2.22 PyDefaultValue	128
12.76.2.23 PzDefaultValue	128
12.76.2.24 RGxDefaultValue	128
12.76.2.25 RYawDefaultValue	129
12.76.2.26 RGyDefaultValue	129
12.76.2.27 RGzDefaultValue	129
12.77 pid_controller_twist.h	129
12.78 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/proportional_↵ controller.cpp File Reference	130
12.78.1 Detailed Description	130
12.78.2 Function Documentation	130
12.78.2.1 main()	130
12.79 proportional_controller.cpp	131
12.80 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/proportional_↵ controller.h File Reference	131
12.80.1 Detailed Description	132
12.80.2 Macro Definition Documentation	132
12.80.2.1 MAX_HORIZONTAL_SPEED	132
12.80.2.2 MAX_ROTATION_SPEED	132
12.80.2.3 MAX_VERTICAL_SPEED	133
12.80.2.4 PxDefaultValue	133
12.80.2.5 PyawDefaultValue	133
12.80.2.6 PyDefaultValue	133
12.80.2.7 PzDefaultValue	133
12.80.2.8 RGxDefaultValue	133
12.80.2.9 RYawDefaultValue	134
12.80.2.10 RGyDefaultValue	134
12.80.2.11 RGzDefaultValue	134
12.81 proportional_controller.h	134
12.82 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/sinusoidal.cpp File Reference	135
12.82.1 Function Documentation	135
12.82.1.1 main()	135
12.83 sinusoidal.cpp	135
12.84 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/sinusoidal.h File Reference	137

12.85 sinusoidal.h	137
12.86 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/square_root_↔ controller.cpp File Reference	137
12.86.1 Detailed Description	138
12.86.2 Function Documentation	138
12.86.2.1 main()	138
12.87 square_root_controller.cpp	138
12.88 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/square_root_↔ controller.h File Reference	139
12.88.1 Detailed Description	140
12.88.2 Macro Definition Documentation	140
12.88.2.1 MAX_HORIZONTAL_SPEED	140
12.88.2.2 MAX_ROTATION_SPEED	140
12.88.2.3 MAX_VERTICAL_SPEED	140
12.88.2.4 PxDefaultValue	140
12.88.2.5 PyawDefaultValue	140
12.88.2.6 PyDefaultValue	141
12.88.2.7 PzDefaultValue	141
12.88.2.8 RGxDefaultValue	141
12.88.2.9 RGyawDefaultValue	141
12.88.2.10 RGyDefaultValue	141
12.88.2.11 RGzDefaultValue	141
12.89 square_root_controller.h	142
Index	143

Chapter 1

Bebop Controller Package

This is the documentation for the Bebop Controller Package. You can download it from its [GitHub](#). More detailed installation instructions can be found at this [page](#).

Bebop Controller is a ROS Package that allows controlling the Parrot Bebop 2 drone in a real environment using an OptiTrack Motion Capture System to obtain the position data of the drone. It also includes a node that allows using Sphinx Simulator and obtain the position data of the drone from the simulator.

Chapter 2

Installation Instructions

The GitHub of the package can be found at this [link](#).

In a bash terminal, the following commands are run to create a workspace in which to install the package.

```
mkdir -p ~/bebop_controller/src
cd ~/bebop_controller
catkin init
```

Files are cloned from the GitHub using the following commands.

```
cd ~/bebop_controller/src
git clone https://github.com/Francisco8382/bebop_controller
git clone https://github.com/ethz-asl/mav_comm
```

Finally, they are compiled by running the following.

```
cd ~/bebop_controller
catkin build
echo source ~/bebop_controller/devel/setup.bash » ~/.bashrc
```


Chapter 3

ROS Nodes

The *src/nodes* and *scripts* folders contain the ROS nodes.

We can find the following node files in these folders.

- [citc_controller_angles.cpp](#)
- [citc_controller_twist.cpp](#)
- [data_to_csv.cpp](#)
- [gazebo.py](#)
- [pid_controller_angles.cpp](#)
- [pid_controller_twist.cpp](#)
- [plot.py](#)
- [proportional_controller.cpp](#)
- [sinusoidal.cpp](#)
- [square_root_controller.cpp](#)

To run these nodes it is recommended to create a launch file and run them with the *roslaunch* command. It can also be run using the *roslaunch* command, but it is more difficult to specify the parameters.

Some [Launch Files](#) are included in the *launch* folder.

Chapter 4

Launch Files

The *launch* folder contains files that are used to run one or more ROS nodes.

In this folder we can find the following files.

- [citc_controller_angles.launch](#)
- [citc_controller_twist.launch](#)
- [gazebo.launch](#)
- [pid_controller_angles.launch](#)
- [pid_controller_twist.launch](#)
- [proportional_controller.launch](#)
- [square_root_controller.launch](#)
- [vrpn_client_ros.launch](#)

Some parameters can be modified directly from these files, but most parameters are modified from YAML files. These parameters are explained in this [section](#).

To run these files, the following command is run in the terminal.

```
roslaunch bebop_controller <name_of_the_file>
```

For example, to run the [vrpn_client_ros.launch](#) file.

```
roslaunch bebop_controller vrpn_client_ros.launch
```


Chapter 5

YAML Files

YAML files are used to set the parameters used by ROS nodes. These files are in the *resource* folder.

The following files are located in this folder.

- [citc_controller_angles.yaml](#)
- [citc_controller_twist.yaml](#)
- [max_speed.yaml](#)
- [normalize_angles.yaml](#)
- [normalize_twist.yaml](#)
- [pid_controller_angles.yaml](#)
- [pid_controller_twist.yaml](#)
- [proportional_controller.yaml](#)
- [safe_zone.yaml](#)
- [square_root_controller.yaml](#)
- [topics.yaml](#)
- [trajectory.yaml](#)
- [waypoint.yaml](#)

Chapter 6

Namespace Index

6.1 Namespace List

Here is a list of all namespaces with brief descriptions:

bebop_controller	Namespace containing all the classes and functions of the Bebop Controller	19
gazebo	Namespace containing all the classes and functions used to get the drone position data from Gazebo	23
plot	Namespace that contains all the classes and functions used to generate graphs that display the test results	23

Chapter 7

Hierarchical Index

7.1 Class Hierarchy

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

bebop_controller::BaseController	25
bebop_controller::CITCController	34
bebop_controller::CITCController	34
bebop_controller::PIDController	40
bebop_controller::PIDController	40
bebop_controller::ProportionalController	43
bebop_controller::SquareRootController	47
bebop_controller::Command_Velocities	35
bebop_controller::DataToCSV	36
bebop_controller::DataToCSVParameters	37
bebop_controller::Normalize	39
plot.Plots	41
gazebo.SphinxPublisher	44
bebop_controller::State	48
TrajectoryParameters	49
bebop_controller::Vector3	50
bebop_controller::Vector4	51
bebop_controller::Waypoint	53
bebop_controller::Sinusoidal	44
WaypointParameters	57

Chapter 8

Class Index

8.1 Class List

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

bebop_controller::BaseController	25
bebop_controller::CITCController	34
bebop_controller::Command_Velocities Structure for storing the command velocities	35
bebop_controller::DataToCSV	36
bebop_controller::DataToCSVParameters	37
bebop_controller::Normalize	39
bebop_controller::PIDController	40
plot.Plots	41
bebop_controller::ProportionalController	43
bebop_controller::Sinusoidal	44
gazebo.SphinxPublisher Class of the publisher that gets the drone position data from the simulator	44
bebop_controller::SquareRootController	47
bebop_controller::State Structure for storing the drone state	48
TrajectoryParameters	49
bebop_controller::Vector3 Structure for storing 3-dimensional vector data	50
bebop_controller::Vector4 Structure for storing 4-dimensional vector data	51
bebop_controller::Waypoint	53
WaypointParameters	57

Chapter 9

File Index

9.1 File List

Here is a list of all files with brief descriptions:

C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/include/bebop_controller/base_controller.h	
Header file for the base class of controllers	59
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/include/bebop_controller/common.h	
File containing common functions used by the Bebop Controller	64
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/include/bebop_controller/waypoint.h	
Header file for the base class of the waypoint generator	67
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/citc_controller_angles.launch	
Launch file to run a test using the CITC controller with reference angles	69
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/citc_controller_twist.launch	
Launch file to run a test using the CITC controller with velocity commands	70
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/gazebo.launch	
Launch file example to run the publisher that gets the position data from the simulator	71
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/pid_controller_angles.launch	
Launch file to run a test using the PID controller with reference angles	71
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/pid_controller_twist.launch	
Launch file to run a test using the PID controller with velocity commands	72
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/proportional_controller.launch	
Launch file to run a test using the proportional controller	73
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/square_root_controller.launch	
Launch file to run a test using the square root controller	74
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/vrpn_client_ros.launch	
Launch file example to use vrpn client as ROS publisher	75
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/citc_controller_angles.yaml	
YAML file for CITC controller parameters, using reference angles	75
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/citc_controller_twist.yaml	
YAML file for CITC controller parameters, using velocity commands	76
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/max_speed.yaml	
YAML file to set the maximum velocity allowed for velocity commands	77
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/normalize_angles.yaml	
YAML file to configure the parameters used to normalize speed for controllers using reference angles	77
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/normalize_twist.yaml	
YAML file to configure the parameters used to normalize speed for controllers using velocity commands	78
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/pid_controller_angles.yaml	
YAML file for PID controller parameters, using reference angles	78

C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/pid_controller_twist.yaml	
YAML file for PID controller parameters, using velocity commands	79
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/proportional_controller.yaml	
YAML file for proportional controller parameters	80
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/safe_zone.yaml	
YAML file to specify safe zone margins	80
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/square_root_controller.yaml	
YAML file for square root controller parameters	80
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/topics.yaml	
YAML file to set the topics used by the nodes	81
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/trajectory.yaml	
YAML file to set the trajectory parameters	81
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/waypoint.yaml	
YAML file to configure trajectory times	82
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/scripts/gazebo.py	
Node file to get the drone position data from Gazebo	82
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/scripts/plot.py	
Node file for plotting test results	84
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/library/base_controller.cpp	
Library file for the base class of controllers	86
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/library/waypoint.cpp	
Library file for the base class of the waypoint generator	89
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/citc_controller_angles.cpp	
Node file for CITC Controller using reference angles	90
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/citc_controller_angles.h	
Header file for CITC Controller using reference angles	93
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/citc_controller_twist.cpp	
Node file for CITC Controller using velocity commands	100
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/citc_controller_twist.h	
Header file for CITC Controller using velocity commands	102
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/data_to_csv.cpp	
Node file to save the test results to a CSV file	109
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/data_to_csv.h	
Header file for the node that saves the test results to a CSV file	111
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/pid_controller_angles.cpp	
Node file for PID Controller using reference angles	113
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/pid_controller_angles.h	
Header file for PID Controller using reference angles	115
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/pid_controller_twist.cpp	
Node file for PID Controller using velocity commands	121
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/pid_controller_twist.h	
Header file for PID Controller using velocity commands	123
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/proportional_controller.cpp	
Node file for proportional controller	130
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/proportional_controller.h	
Header file for proportional controller	131
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/sinusoidal.cpp	135
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/sinusoidal.h	137
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/square_root_controller.cpp	
Node file for square root controller	137
C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/square_root_controller.h	
Header file for square root controller	139

Chapter 10

Namespace Documentation

10.1 bebop_controller Namespace Reference

Namespace containing all the classes and functions of the Bebop Controller.

Classes

- class [BaseController](#)
- class [CITCController](#)
- struct [Command_Velocities](#)
Structure for storing the command velocities.
- class [DataToCSV](#)
- struct [DataToCSVParameters](#)
- struct [Normalize](#)
- class [PIDController](#)
- class [ProportionalController](#)
- class [Sinusoidal](#)
- class [SquareRootController](#)
- struct [State](#)
Structure for storing the drone state.
- struct [Vector3](#)
Structure for storing 3-dimensional vector data.
- struct [Vector4](#)
Structure for storing 4-dimensional vector data.
- class [Waypoint](#)

Functions

- double [bound](#) (double value, double min, double [max](#))
- double [clamp](#) (double value, double lim)
- double [sgn](#) (double value)
- double [max](#) (double val1, double val2)
- template<typename T >
void [GetRosParameter](#) (const ros::NodeHandle &nh, const std::string &key, const T &default_value, T *value)
Template function to get ROS parameters.
- double [yawFromQuaternion](#) (const Eigen::Quaterniond &q)
- Eigen::Vector3d [Quat2RPY](#) (Eigen::Vector4d &Quaternion)
- Eigen::Vector4d [RPY2Quat](#) (Eigen::Vector3d &RPY)

10.1.1 Detailed Description

Namespace containing all the classes and functions of the Bebop Controller.

[sinusoidal.cpp](#) Node file for the sinusoidal waypoint generator.

10.1.2 Function Documentation

10.1.2.1 `bound()`

```
double bebop_controller::bound (
    double value,
    double min,
    double max )
```

Limit a variable to an upper and lower limit.

Parameters

<i>value</i>	Value to limit.
<i>min</i>	Minimum value.
<i>max</i>	Maximum value.

Returns

The limited value.

Definition at line 24 of file [common.h](#).

10.1.2.2 `clamp()`

```
double bebop_controller::clamp (
    double value,
    double lim )
```

Limit a variable to a maximum value in magnitude.

Parameters

<i>value</i>	Value to limit.
<i>lim</i>	Magnitude limit.

Returns

The limited value.

Definition at line 40 of file [common.h](#).

10.1.2.3 GetRosParameter()

```
template<typename T >
void bebop_controller::GetRosParameter (
    const ros::NodeHandle & nh,
    const std::string & key,
    const T & default_value,
    T * value ) [inline]
```

Template function to get ROS parameters.

Definition at line 106 of file [common.h](#).

10.1.2.4 max()

```
double bebop_controller::max (
    double val1,
    double val2 )
```

Max function.

Parameters

<i>val1</i>	Value 1 to compare.
<i>val2</i>	Value 2 to compare.

Returns

The maximum value between both values.

Definition at line 63 of file [common.h](#).

10.1.2.5 Quat2RPY()

```
Eigen::Vector3d bebop_controller::Quat2RPY (
    Eigen::Vector4d & Quaternion )
```

Function to convert a Quaternion into roll, pitch and yaw angles.

Returns

A 3-dimensional vector with the roll, pitch, and yaw values.

Definition at line 128 of file [common.h](#).

10.1.2.6 RPY2Quat()

```
Eigen::Vector4d bebop_controller::RPY2Quat (
    Eigen::Vector3d & RPY )
```

Function to convert roll, pitch and yaw angles into a Quaternion.

Returns

A 4-dimensional vector with the quaternion values.

Definition at line 138 of file [common.h](#).

10.1.2.7 sgn()

```
double bebop_controller::sgn (
    double value )
```

Sign function.

Parameters

<i>value</i>	Value to check.
--------------	-----------------

Returns

1 if the value is greater than 0, -1 if the value is less than 0 and 0 if the value is 0.

Definition at line 47 of file [common.h](#).

10.1.2.8 yawFromQuaternion()

```
double bebop_controller::yawFromQuaternion (
    const Eigen::Quaterniond & q ) [inline]
```

Function to get the yaw angle from a Eigen::Quaterniond reference.

Returns

The yaw angle value.

Definition at line 121 of file [common.h](#).

10.2 gazebo Namespace Reference

Namespace containing all the classes and functions used to get the drone position data from Gazebo.

Classes

- class [SphinxPublisher](#)

Class of the publisher that gets the drone position data from the simulator.

10.2.1 Detailed Description

Namespace containing all the classes and functions used to get the drone position data from Gazebo.

10.3 plot Namespace Reference

Namespace that contains all the classes and functions used to generate graphs that display the test results.

Classes

- class [Plots](#)

Variables

- [path](#) = `rospy.get_param('~Dir')`
- [sub](#) = `rospy.get_param('/Subfolder')`
- [topic](#) = `rospy.get_param('~Topics/CSV_End')`
- [yaml](#) = `rospy.get_param('~YAML')`
- [plt](#) = `Plots(os.path.join(path,sub),topic,yaml)`

10.3.1 Detailed Description

Namespace that contains all the classes and functions used to generate graphs that display the test results.

10.3.2 Variable Documentation

10.3.2.1 path

```
plot.path = rospy.get_param('~Dir')
```

Definition at line 73 of file [plot.py](#).

10.3.2.2 plt

```
plot.plt = Plots(os.path.join(path, sub), topic, yaml)
```

Definition at line 78 of file [plot.py](#).

10.3.2.3 sub

```
plot.sub = rospy.get_param('/Subfolder')
```

Definition at line 74 of file [plot.py](#).

10.3.2.4 topic

```
plot.topic = rospy.get_param('~Topics/CSV_End')
```

Definition at line 75 of file [plot.py](#).

10.3.2.5 yaml

```
plot.yaml = rospy.get_param('~YAML')
```

Definition at line 76 of file [plot.py](#).

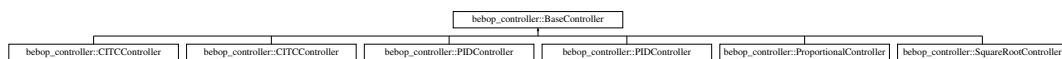
Chapter 11

Class Documentation

11.1 bebop_controller::BaseController Class Reference

```
#include <base_controller.h>
```

Inheritance diagram for bebop_controller::BaseController:



Public Member Functions

- [BaseController](#) ()
- [~BaseController](#) ()

Protected Member Functions

- void [MultiDOFJointTrajectory_CB](#) (const trajectory_msgs::MultiDOFJointTrajectoryConstPtr &trajectory_↔ reference_msg)
- void [TakeOff](#) ()
- void [Land](#) ()
- void [Odometry_CB](#) (const geometry_msgs::PoseStamped &pose_msg)
- void [Stop_CB](#) (const std_msgs::Empty::ConstPtr &empty_msg)
- void [TimeOut_CB](#) (const ros::TimerEvent &event)
- void [SetTrajectoryPoint](#) (mav_msgs::EigenTrajectoryPoint &eigen_reference)
- void [SetOdometry](#) (mav_msgs::EigenOdometry &odometry)
- void [Quaternion2Euler](#) (double &roll, double &pitch, double &yaw) const
- void [GetErrors](#) ([Vector4](#) &e)
- void [GetVelocityErrors](#) ([Vector4](#) &dot_e)
- void [EstimateVelocity](#) ()
- void [EstimateAcceleration](#) ()
- void [Stop](#) (bool failsafe)
- bool [CheckSafeZone](#) ()
- void [CalculateLeashLength](#) ([Vector4](#) &e, [Vector4](#) &P)
- void [LimitPositionErrors](#) ([Vector4](#) &e)
- virtual void [CalculateCommandVelocities](#) (geometry_msgs::Twist &ref_command_signals)

Protected Attributes

- bool [waypointHasBeenPublished_](#)
- bool [takeoff](#)
- bool [controller_active_](#)
- bool [disable_commands](#)
- bool [stop](#)
- double [diff](#)
- [ros::Subscriber](#) [cmd_multi_dof_joint_trajectory_sub_](#)
- [ros::Subscriber](#) [odom_sub_](#)
- [ros::Subscriber](#) [end_sub_](#)
- [ros::Publisher](#) [motor_velocity_reference_pub_](#)
- [ros::Publisher](#) [takeoff_pub_](#)
- [ros::Publisher](#) [land_pub_](#)
- [ros::Publisher](#) [odometry_filtered_pub_](#)
- [ros::Publisher](#) [reference_angles_pub_](#)
- [ros::Publisher](#) [smoothed_reference_pub_](#)
- [ros::Time](#) [lastTime](#)
- [ros::Timer](#) [timeOut](#)
- [mav_msgs::EigenTrajectoryPoint](#) [command_trajectory_](#)
- [mav_msgs::EigenOdometry](#) [odometry_](#)
- [State](#) [state](#)
- [State](#) [last_state](#)
- [Vector3](#) [safe_zone](#)
- [Vector4](#) [max_speed](#)
- [Vector3](#) [leash_length](#)

11.1.1 Detailed Description

Definition at line 26 of file [base_controller.h](#).

11.1.2 Constructor & Destructor Documentation

11.1.2.1 BaseController()

```
bebop_controller::BaseController::BaseController ( )
```

Definition at line 8 of file [base_controller.cpp](#).

11.1.2.2 ~BaseController()

```
bebop_controller::BaseController::~BaseController ( )
```

Definition at line 64 of file [base_controller.cpp](#).

11.1.3 Member Function Documentation

11.1.3.1 CalculateCommandVelocities()

```
void bebop_controller::BaseController::CalculateCommandVelocities (
    geometry_msgs::Twist & ref_command_signals ) [protected], [virtual]
```

Definition at line 184 of file [base_controller.cpp](#).

11.1.3.2 CalculateLeashLength()

```
void bebop_controller::BaseController::CalculateLeashLength (
    Vector4 & e,
    Vector4 & P ) [protected]
```

Definition at line 233 of file [base_controller.cpp](#).

11.1.3.3 CheckSafeZone()

```
bool bebop_controller::BaseController::CheckSafeZone ( ) [protected]
```

Definition at line 227 of file [base_controller.cpp](#).

11.1.3.4 EstimateAcceleration()

```
void bebop_controller::BaseController::EstimateAcceleration ( ) [protected]
```

Definition at line 202 of file [base_controller.cpp](#).

11.1.3.5 EstimateVelocity()

```
void bebop_controller::BaseController::EstimateVelocity ( ) [protected]
```

Definition at line 191 of file [base_controller.cpp](#).

11.1.3.6 GetErrors()

```
void bebop_controller::BaseController::GetErrors (
    Vector4 & e ) [protected]
```

Definition at line 155 of file [base_controller.cpp](#).

11.1.3.7 GetVelocityErrors()

```
void bebop_controller::BaseController::GetVelocityErrors (
    Vector4 & dot_e ) [protected]
```

Definition at line 170 of file [base_controller.cpp](#).

11.1.3.8 Land()

```
void bebop_controller::BaseController::Land ( ) [protected]
```

Definition at line 136 of file [base_controller.cpp](#).

11.1.3.9 LimitPositionErrors()

```
void bebop_controller::BaseController::LimitPositionErrors (
    Vector4 & e ) [protected]
```

Definition at line 245 of file [base_controller.cpp](#).

11.1.3.10 MultiDOFJointTrajectory_CB()

```
void bebop_controller::BaseController::MultiDOFJointTrajectory_CB (
    const trajectory_msgs::MultiDOFJointTrajectoryConstPtr & trajectory_reference_msg
) [protected]
```

Definition at line 68 of file [base_controller.cpp](#).

11.1.3.11 Odometry_CB()

```
void bebop_controller::BaseController::Odometry_CB (
    const geometry_msgs::PoseStamped & pose_msg ) [protected]
```

Definition at line 83 of file [base_controller.cpp](#).

11.1.3.12 Quaternion2Euler()

```
void bebop_controller::BaseController::Quaternion2Euler (
    double & roll,
    double & pitch,
    double & yaw ) const [protected]
```

Definition at line 177 of file [base_controller.cpp](#).

11.1.3.13 SetOdometry()

```
void bebop_controller::BaseController::SetOdometry (
    mav_msgs::EigenOdometry & odometry ) [protected]
```

Definition at line 146 of file [base_controller.cpp](#).

11.1.3.14 SetTrajectoryPoint()

```
void bebop_controller::BaseController::SetTrajectoryPoint (
    mav_msgs::EigenTrajectoryPoint & eigen_reference ) [protected]
```

11.1.3.15 Stop()

```
void bebop_controller::BaseController::Stop (
    bool failsafe ) [protected]
```

Definition at line 213 of file [base_controller.cpp](#).

11.1.3.16 Stop_CB()

```
void bebop_controller::BaseController::Stop_CB (
    const std_msgs::Empty::ConstPtr & empty_msg ) [protected]
```

Definition at line 79 of file [base_controller.cpp](#).

11.1.3.17 TakeOff()

```
void bebop_controller::BaseController::TakeOff ( ) [protected]
```

Definition at line 127 of file [base_controller.cpp](#).

11.1.3.18 TimeOut_CB()

```
void bebop_controller::BaseController::TimeOut_CB (  
    const ros::TimerEvent & event ) [protected]
```

Definition at line 120 of file [base_controller.cpp](#).

11.1.4 Member Data Documentation

11.1.4.1 cmd_multi_dof_joint_trajectory_sub_

```
ros::Subscriber bebop_controller::BaseController::cmd_multi_dof_joint_trajectory_sub_ [protected]
```

Definition at line 39 of file [base_controller.h](#).

11.1.4.2 command_trajectory_

```
mav_msgs::EigenTrajectoryPoint bebop_controller::BaseController::command_trajectory_ [protected]
```

Definition at line 53 of file [base_controller.h](#).

11.1.4.3 controller_active_

```
bool bebop_controller::BaseController::controller_active_ [protected]
```

Definition at line 34 of file [base_controller.h](#).

11.1.4.4 diff

```
double bebop_controller::BaseController::diff [protected]
```

Definition at line 37 of file [base_controller.h](#).

11.1.4.5 disable_commands

```
bool bebop_controller::BaseController::disable_commands [protected]
```

Definition at line 35 of file [base_controller.h](#).

11.1.4.6 end_sub_

`ros::Subscriber bebop_controller::BaseController::end_sub_ [protected]`

Definition at line 41 of file [base_controller.h](#).

11.1.4.7 land_pub_

`ros::Publisher bebop_controller::BaseController::land_pub_ [protected]`

Definition at line 45 of file [base_controller.h](#).

11.1.4.8 last_state

`State bebop_controller::BaseController::last_state [protected]`

Definition at line 56 of file [base_controller.h](#).

11.1.4.9 lastTime

`ros::Time bebop_controller::BaseController::lastTime [protected]`

Definition at line 50 of file [base_controller.h](#).

11.1.4.10 leash_length

`Vector3 bebop_controller::BaseController::leash_length [protected]`

Definition at line 59 of file [base_controller.h](#).

11.1.4.11 max_speed

`Vector4 bebop_controller::BaseController::max_speed [protected]`

Definition at line 58 of file [base_controller.h](#).

11.1.4.12 motor_velocity_reference_pub_

```
ros::Publisher bebop_controller::BaseController::motor_velocity_reference_pub_ [protected]
```

Definition at line 43 of file [base_controller.h](#).

11.1.4.13 odom_sub_

```
ros::Subscriber bebop_controller::BaseController::odom_sub_ [protected]
```

Definition at line 40 of file [base_controller.h](#).

11.1.4.14 odometry_

```
mav_msgs::EigenOdometry bebop_controller::BaseController::odometry_ [protected]
```

Definition at line 54 of file [base_controller.h](#).

11.1.4.15 odometry_filtered_pub_

```
ros::Publisher bebop_controller::BaseController::odometry_filtered_pub_ [protected]
```

Definition at line 46 of file [base_controller.h](#).

11.1.4.16 reference_angles_pub_

```
ros::Publisher bebop_controller::BaseController::reference_angles_pub_ [protected]
```

Definition at line 47 of file [base_controller.h](#).

11.1.4.17 safe_zone

```
Vector3 bebop_controller::BaseController::safe_zone [protected]
```

Definition at line 57 of file [base_controller.h](#).

11.1.4.18 smoothed_reference_pub_

`ros::Publisher bebop_controller::BaseController::smoothed_reference_pub_` [protected]

Definition at line 48 of file [base_controller.h](#).

11.1.4.19 state

`State bebop_controller::BaseController::state` [protected]

Definition at line 55 of file [base_controller.h](#).

11.1.4.20 stop

`bool bebop_controller::BaseController::stop` [protected]

Definition at line 36 of file [base_controller.h](#).

11.1.4.21 takeoff

`bool bebop_controller::BaseController::takeoff` [protected]

Definition at line 33 of file [base_controller.h](#).

11.1.4.22 takeoff_pub_

`ros::Publisher bebop_controller::BaseController::takeoff_pub_` [protected]

Definition at line 44 of file [base_controller.h](#).

11.1.4.23 timeOut

`ros::Timer bebop_controller::BaseController::timeOut` [protected]

Definition at line 51 of file [base_controller.h](#).

11.1.4.24 waypointHasBeenPublished_

```
bool bebop_controller::BaseController::waypointHasBeenPublished_ [protected]
```

Definition at line 32 of file [base_controller.h](#).

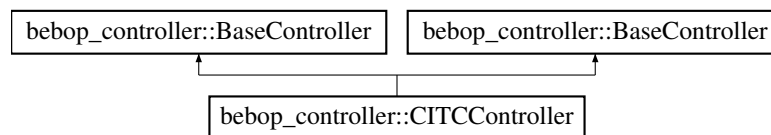
The documentation for this class was generated from the following files:

- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/include/bebop_controller/[base_controller.h](#)
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/library/[base_controller.cpp](#)

11.2 bebop_controller::CITCController Class Reference

```
#include <citc_controller_angles.h>
```

Inheritance diagram for bebop_controller::CITCController:



Public Member Functions

- [CITCController](#) ()
- [CITCController](#) ()

Additional Inherited Members

11.2.1 Detailed Description

Definition at line 50 of file [citc_controller_angles.h](#).

11.2.2 Constructor & Destructor Documentation

11.2.2.1 CITCController() [1/2]

```
bebop_controller::CITCController::CITCController ( )
```

Definition at line 8 of file [citc_controller_angles.cpp](#).

11.2.2.2 CITCController() [2/2]

```
bebop_controller::CITCController::CITCController ( )
```

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

- [C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/citc_controller_angles.h](#)
- [C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/citc_controller_twist.h](#)
- [C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/citc_controller_angles.cpp](#)
- [C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/citc_controller_twist.cpp](#)

11.3 bebop_controller::Command_Velocities Struct Reference

Structure for storing the command velocities.

```
#include <common.h>
```

Public Attributes

- double [x](#)
- double [y](#)
- double [z](#)
- double [yaw](#)

11.3.1 Detailed Description

Structure for storing the command velocities.

Definition at line 98 of file [common.h](#).

11.3.2 Member Data Documentation

11.3.2.1 x

```
double bebop_controller::Command_Velocities::x
```

Definition at line 99 of file [common.h](#).

11.3.2.2 y

```
double bebop_controller::Command_Velocities::y
```

Definition at line 100 of file [common.h](#).

11.3.2.3 yaw

```
double bebop_controller::Command_Velocities::yaw
```

Definition at line 102 of file [common.h](#).

11.3.2.4 z

```
double bebop_controller::Command_Velocities::z
```

Definition at line 101 of file [common.h](#).

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

- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/include/bebop_controller/[common.h](#)

11.4 bebop_controller::DataToCSV Class Reference

```
#include <data_to_csv.h>
```

Public Member Functions

- [DataToCSV](#) ([DataToCSVParameters](#) params)
- [~DataToCSV](#) ()

11.4.1 Detailed Description

Definition at line 24 of file [data_to_csv.h](#).

11.4.2 Constructor & Destructor Documentation

11.4.2.1 DataToCSV()

```
bebop_controller::DataToCSV::DataToCSV (
    DataToCSVParameters params )
```

Definition at line 8 of file [data_to_csv.cpp](#).

11.4.2.2 ~DataToCSV()

```
bebop_controller::DataToCSV::~DataToCSV ( )
```

Definition at line 31 of file [data_to_csv.cpp](#).

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

- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/[data_to_csv.h](#)
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/[data_to_csv.cpp](#)

11.5 bebop_controller::DataToCSVParameters Struct Reference

```
#include <data_to_csv.h>
```

Public Attributes

- std::string [Topic_Drone_Pose](#)
- std::string [Topic_Reference_Pose](#)
- std::string [Topic_CMD_Vel](#)
- std::string [Topic_CSV_Begin](#)
- std::string [Topic_CSV_End](#)
- std::string [File](#)
- double [Initial_Time](#)
- double [Margin_Time](#)
- std::string [Topic_Velocities](#)

11.5.1 Detailed Description

Definition at line 12 of file [data_to_csv.h](#).

11.5.2 Member Data Documentation

11.5.2.1 File

```
std::string bebop_controller::DataToCSVParameters::File
```

Definition at line 18 of file [data_to_csv.h](#).

11.5.2.2 Initial_Time

```
double bebop_controller::DataToCSVParameters::Initial_Time
```

Definition at line 19 of file [data_to_csv.h](#).

11.5.2.3 Margin_Time

```
double bebop_controller::DataToCSVParameters::Margin_Time
```

Definition at line 20 of file [data_to_csv.h](#).

11.5.2.4 Topic_CMD_Vel

```
std::string bebop_controller::DataToCSVParameters::Topic_CMD_Vel
```

Definition at line 15 of file [data_to_csv.h](#).

11.5.2.5 Topic_CSV_Begin

```
std::string bebop_controller::DataToCSVParameters::Topic_CSV_Begin
```

Definition at line 16 of file [data_to_csv.h](#).

11.5.2.6 Topic_CSV_End

```
std::string bebop_controller::DataToCSVParameters::Topic_CSV_End
```

Definition at line 17 of file [data_to_csv.h](#).

11.5.2.7 Topic_Drone_Pose

```
std::string bebop_controller::DataToCSVParameters::Topic_Drone_Pose
```

Definition at line 13 of file [data_to_csv.h](#).

11.5.2.8 Topic_Reference_Pose

```
std::string bebop_controller::DataToCSVParameters::Topic_Reference_Pose
```

Definition at line 14 of file [data_to_csv.h](#).

11.5.2.9 Topic_Velocities

```
std::string bebop_controller::DataToCSVParameters::Topic_Velocities
```

Definition at line 21 of file [data_to_csv.h](#).

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

- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/[data_to_csv.h](#)

11.6 bebop_controller::Normalize Struct Reference

```
#include <citc_controller_angles.h>
```

Public Attributes

- double [angle](#)
- double [vertical](#)
- double [rotation](#)
- double [horizontal](#)

11.6.1 Detailed Description

Definition at line 44 of file [citc_controller_angles.h](#).

11.6.2 Member Data Documentation

11.6.2.1 angle

```
double bebop_controller::Normalize::angle
```

Definition at line 45 of file [citic_controller_angles.h](#).

11.6.2.2 horizontal

```
double bebop_controller::Normalize::horizontal
```

Definition at line 45 of file [citic_controller_twist.h](#).

11.6.2.3 rotation

```
double bebop_controller::Normalize::rotation
```

Definition at line 47 of file [citic_controller_angles.h](#).

11.6.2.4 vertical

```
double bebop_controller::Normalize::vertical
```

Definition at line 46 of file [citic_controller_angles.h](#).

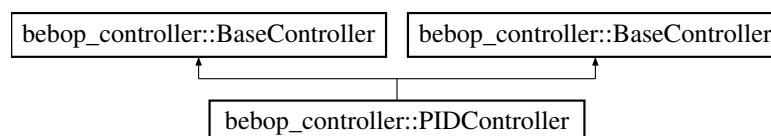
The documentation for this struct was generated from the following files:

- [C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/citic_controller_angles.h](#)
- [C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/citic_controller_twist.h](#)
- [C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/pid_controller_angles.h](#)
- [C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/pid_controller_twist.h](#)
- [C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/proportional_controller.h](#)
- [C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/square_root_controller.h](#)

11.7 bebop_controller::PIDController Class Reference

```
#include <pid_controller_angles.h>
```

Inheritance diagram for bebop_controller::PIDController:



Public Member Functions

- [PIDController](#) ()
- [PIDController](#) ()

Additional Inherited Members

11.7.1 Detailed Description

Definition at line 49 of file [pid_controller_angles.h](#).

11.7.2 Constructor & Destructor Documentation

11.7.2.1 PIDController() [1/2]

```
bebop_controller::PIDController::PIDController ( )
```

Definition at line 8 of file [pid_controller_angles.cpp](#).

11.7.2.2 PIDController() [2/2]

```
bebop_controller::PIDController::PIDController ( )
```

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

- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/[pid_controller_angles.h](#)
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/[pid_controller_twist.h](#)
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/[pid_controller_angles.cpp](#)
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/[pid_controller_twist.cpp](#)

11.8 plot.Plots Class Reference

Public Member Functions

- def [__init__](#) (self, [path](#), [topic](#), [yaml](#))
- def [Plot](#) (self, data)
Method that generates the graphs of the test results.

Public Attributes

- [path](#)
- [yaml](#)

11.8.1 Detailed Description

Definition at line 22 of file [plot.py](#).

11.8.2 Constructor & Destructor Documentation

11.8.2.1 `__init__()`

```
def plot.Plots.__init__ (
    self,
    path,
    topic,
    yaml )
```

Definition at line 23 of file [plot.py](#).

11.8.3 Member Function Documentation

11.8.3.1 `Plot()`

```
def plot.Plots.Plot (
    self,
    data )
```

Method that generates the graphs of the test results.

Definition at line 30 of file [plot.py](#).

11.8.4 Member Data Documentation

11.8.4.1 `path`

```
plot.Plots.path
```

Definition at line 24 of file [plot.py](#).

11.8.4.2 yaml

`plot.Plots.yaml`

Definition at line 25 of file [plot.py](#).

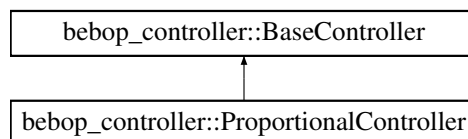
The documentation for this class was generated from the following file:

- [C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/scripts/plot.py](#)

11.9 bebop_controller::ProportionalController Class Reference

```
#include <proportional_controller.h>
```

Inheritance diagram for bebop_controller::ProportionalController:



Public Member Functions

- [ProportionalController\(\)](#)

Additional Inherited Members

11.9.1 Detailed Description

Definition at line 28 of file [proportional_controller.h](#).

11.9.2 Constructor & Destructor Documentation

11.9.2.1 ProportionalController()

```
bebop_controller::ProportionalController::ProportionalController ( )
```

Definition at line 8 of file [proportional_controller.cpp](#).

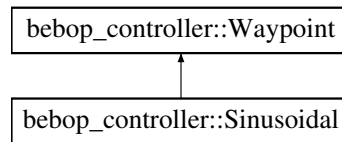
The documentation for this class was generated from the following files:

- [C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/proportional_controller.h](#)
- [C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/proportional_controller.cpp](#)

11.10 bebop_controller::Sinusoidal Class Reference

```
#include <sinusoidal.h>
```

Inheritance diagram for bebop_controller::Sinusoidal:



Public Member Functions

- [Sinusoidal](#) ([WaypointParameters](#) wp_params, [TrajectoryParameters](#) t_params)

Additional Inherited Members

11.10.1 Detailed Description

Definition at line 15 of file [sinusoidal.h](#).

11.10.2 Constructor & Destructor Documentation

11.10.2.1 Sinusoidal()

```
bebop_controller::Sinusoidal::Sinusoidal (
    WaypointParameters wp_params,
    TrajectoryParameters t_params )
```

Definition at line 8 of file [sinusoidal.cpp](#).

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

- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/[sinusoidal.h](#)
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/[sinusoidal.cpp](#)

11.11 gazebo.SphinxPublisher Class Reference

Class of the publisher that gets the drone position data from the simulator.

Public Member Functions

- `def __init__ (self)`
- `def process_output (self, out, queue)`
Method that processes the console output and stores it in class variables.
- `def publish (self, event)`
Method that publishes the drone position data in a PoseStamped message.

Public Attributes

- `q`
- `pos`
- `att`
- `cont`
- `odom_pub`

11.11.1 Detailed Description

Class of the publisher that gets the drone position data from the simulator.

Definition at line 21 of file [gazebo.py](#).

11.11.2 Constructor & Destructor Documentation

11.11.2.1 __init__()

```
def gazebo.SphinxPublisher.__init__ (  
    self )
```

Definition at line 22 of file [gazebo.py](#).

11.11.3 Member Function Documentation

11.11.3.1 process_output()

```
def gazebo.SphinxPublisher.process_output (  
    self,  
    out,  
    queue )
```

Method that processes the console output and stores it in class variables.

Definition at line 45 of file [gazebo.py](#).

11.11.3.2 `publish()`

```
def gazebo.SphinxPublisher.publish (
    self,
    event )
```

Method that publishes the drone position data in a PoseStamped message.

Definition at line 71 of file [gazebo.py](#).

11.11.4 Member Data Documentation

11.11.4.1 `att`

```
gazebo.SphinxPublisher.att
```

Definition at line 25 of file [gazebo.py](#).

11.11.4.2 `cont`

```
gazebo.SphinxPublisher.cont
```

Definition at line 26 of file [gazebo.py](#).

11.11.4.3 `odom_pub`

```
gazebo.SphinxPublisher.odom_pub
```

Definition at line 39 of file [gazebo.py](#).

11.11.4.4 `pos`

```
gazebo.SphinxPublisher.pos
```

Definition at line 24 of file [gazebo.py](#).

11.11.4.5 q

`gazebo.SphinxPublisher.q`

Definition at line 23 of file [gazebo.py](#).

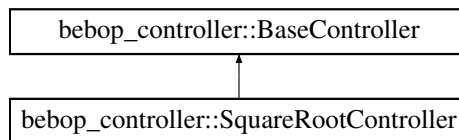
The documentation for this class was generated from the following file:

- [C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/scripts/gazebo.py](#)

11.12 bebop_controller::SquareRootController Class Reference

```
#include <square_root_controller.h>
```

Inheritance diagram for bebop_controller::SquareRootController:



Public Member Functions

- [SquareRootController\(\)](#)

Additional Inherited Members

11.12.1 Detailed Description

Definition at line 28 of file [square_root_controller.h](#).

11.12.2 Constructor & Destructor Documentation

11.12.2.1 SquareRootController()

```
bebop_controller::SquareRootController::SquareRootController ( )
```

Definition at line 8 of file [square_root_controller.cpp](#).

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

- [C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/square_root_controller.h](#)
- [C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/square_root_controller.cpp](#)

11.13 bebop_controller::State Struct Reference

Structure for storing the drone state.

```
#include <common.h>
```

Public Attributes

- [Vector3 position](#)
- [Vector3 orientation](#)
- [Vector3 velocity](#)
- [Vector3 angular_velocity](#)
- [Vector3 acceleration](#)
- [Vector3 angular_acceleration](#)

11.13.1 Detailed Description

Structure for storing the drone state.

Definition at line 88 of file [common.h](#).

11.13.2 Member Data Documentation

11.13.2.1 acceleration

[Vector3](#) bebop_controller::State::acceleration

Definition at line 93 of file [common.h](#).

11.13.2.2 angular_acceleration

[Vector3](#) bebop_controller::State::angular_acceleration

Definition at line 94 of file [common.h](#).

11.13.2.3 angular_velocity

[Vector3](#) bebop_controller::State::angular_velocity

Definition at line 92 of file [common.h](#).

11.13.2.4 orientation

`Vector3` `bebop_controller::State::orientation`

Definition at line 90 of file [common.h](#).

11.13.2.5 position

`Vector3` `bebop_controller::State::position`

Definition at line 89 of file [common.h](#).

11.13.2.6 velocity

`Vector3` `bebop_controller::State::velocity`

Definition at line 91 of file [common.h](#).

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

- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/include/bebop_controller/[common.h](#)

11.14 TrajectoryParameters Struct Reference

```
#include <sinusoidal.h>
```

Public Attributes

- `Eigen::Vector3d` [PositionBeforeTrajectory](#)
- `Eigen::Vector3d` [TrajectoryDistance](#)
- `bool` [Yaw_Enabled](#)
- `double` [Yaw_Offset](#)

11.14.1 Detailed Description

[sinusoidal.h](#) Header file for the sinusoidal waypoint generator.

Definition at line 6 of file [sinusoidal.h](#).

11.14.2 Member Data Documentation

11.14.2.1 PositionBeforeTrajectory

```
Eigen::Vector3d TrajectoryParameters::PositionBeforeTrajectory
```

Definition at line 7 of file [sinusoidal.h](#).

11.14.2.2 TrajectoryDistance

```
Eigen::Vector3d TrajectoryParameters::TrajectoryDistance
```

Definition at line 8 of file [sinusoidal.h](#).

11.14.2.3 Yaw_Enabled

```
bool TrajectoryParameters::Yaw_Enabled
```

Definition at line 9 of file [sinusoidal.h](#).

11.14.2.4 Yaw_Offset

```
double TrajectoryParameters::Yaw_Offset
```

Definition at line 10 of file [sinusoidal.h](#).

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

- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/[sinusoidal.h](#)

11.15 bebop_controller::Vector3 Struct Reference

Structure for storing 3-dimensional vector data.

```
#include <common.h>
```

Public Attributes

- double [x](#)
- double [y](#)
- double [z](#)

11.15.1 Detailed Description

Structure for storing 3-dimensional vector data.

Definition at line 73 of file [common.h](#).

11.15.2 Member Data Documentation

11.15.2.1 x

```
double bebop_controller::Vector3::x
```

Definition at line 74 of file [common.h](#).

11.15.2.2 y

```
double bebop_controller::Vector3::y
```

Definition at line 75 of file [common.h](#).

11.15.2.3 z

```
double bebop_controller::Vector3::z
```

Definition at line 76 of file [common.h](#).

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

- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/include/bebop_controller/[common.h](#)

11.16 bebop_controller::Vector4 Struct Reference

Structure for storing 4-dimensional vector data.

```
#include <common.h>
```

Public Attributes

- double [x](#)
- double [y](#)
- double [z](#)
- double [yaw](#)

11.16.1 Detailed Description

Structure for storing 4-dimensional vector data.

Definition at line 80 of file [common.h](#).

11.16.2 Member Data Documentation

11.16.2.1 x

```
double bebop_controller::Vector4::x
```

Definition at line 81 of file [common.h](#).

11.16.2.2 y

```
double bebop_controller::Vector4::y
```

Definition at line 82 of file [common.h](#).

11.16.2.3 yaw

```
double bebop_controller::Vector4::yaw
```

Definition at line 84 of file [common.h](#).

11.16.2.4 z

```
double bebop_controller::Vector4::z
```

Definition at line 83 of file [common.h](#).

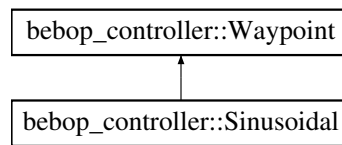
The documentation for this struct was generated from the following file:

- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/include/bebop_controller/[common.h](#)

11.17 bebop_controller::Waypoint Class Reference

```
#include <waypoint.h>
```

Inheritance diagram for bebop_controller::Waypoint:



Public Member Functions

- [Waypoint](#) ([WaypointParameters](#) param)
- [~Waypoint](#) ()

Protected Member Functions

- void [Start_CB](#) (const ros::TimerEvent &event)
- void [Stop_CB](#) (const ros::TimerEvent &event)
- virtual void [Trajectory_CB](#) (const ros::TimerEvent &event)

Protected Attributes

- [WaypointParameters](#) wp_params_
- ros::NodeHandle nh
- ros::NodeHandle _n1
- ros::NodeHandle _n2
- ros::NodeHandle _n3
- ros::Timer _timer1
- ros::Timer _timer2
- ros::Timer _timer3
- ros::Publisher [Begin_](#)
- ros::Publisher [End_](#)
- ros::Publisher [setpoint_pub_](#)
- ros::Time [Initial_Time](#)
- trajectory_msgs::MultiDOFJointTrajectory [position_target_](#)
- enum [Status](#) status

11.17.1 Detailed Description

Definition at line 26 of file [waypoint.h](#).

11.17.2 Constructor & Destructor Documentation

11.17.2.1 Waypoint()

```
bebop_controller::Waypoint::Waypoint (
    WaypointParameters param )
```

Definition at line 8 of file [waypoint.cpp](#).

11.17.2.2 ~Waypoint()

```
bebop_controller::Waypoint::~~Waypoint ( )
```

Definition at line 21 of file [waypoint.cpp](#).

11.17.3 Member Function Documentation

11.17.3.1 Start_CB()

```
void bebop_controller::Waypoint::Start_CB (
    const ros::TimerEvent & event ) [protected]
```

Definition at line 23 of file [waypoint.cpp](#).

11.17.3.2 Stop_CB()

```
void bebop_controller::Waypoint::Stop_CB (
    const ros::TimerEvent & event ) [protected]
```

Definition at line 33 of file [waypoint.cpp](#).

11.17.3.3 Trajectory_CB()

```
void bebop_controller::Waypoint::Trajectory_CB (
    const ros::TimerEvent & event ) [protected], [virtual]
```

Definition at line 42 of file [waypoint.cpp](#).

11.17.4 Member Data Documentation

11.17.4.1 _n1

`ros::NodeHandle bebop_controller::Waypoint::_n1` [protected]

Definition at line 35 of file [waypoint.h](#).

11.17.4.2 _n2

`ros::NodeHandle bebop_controller::Waypoint::_n2` [protected]

Definition at line 36 of file [waypoint.h](#).

11.17.4.3 _n3

`ros::NodeHandle bebop_controller::Waypoint::_n3` [protected]

Definition at line 37 of file [waypoint.h](#).

11.17.4.4 _timer1

`ros::Timer bebop_controller::Waypoint::_timer1` [protected]

Definition at line 38 of file [waypoint.h](#).

11.17.4.5 _timer2

`ros::Timer bebop_controller::Waypoint::_timer2` [protected]

Definition at line 39 of file [waypoint.h](#).

11.17.4.6 _timer3

`ros::Timer bebop_controller::Waypoint::_timer3` [protected]

Definition at line 40 of file [waypoint.h](#).

11.17.4.7 Begin_

```
ros::Publisher bebop_controller::Waypoint::Begin_ [protected]
```

Definition at line 42 of file [waypoint.h](#).

11.17.4.8 End_

```
ros::Publisher bebop_controller::Waypoint::End_ [protected]
```

Definition at line 43 of file [waypoint.h](#).

11.17.4.9 Initial_Time

```
ros::Time bebop_controller::Waypoint::Initial_Time [protected]
```

Definition at line 46 of file [waypoint.h](#).

11.17.4.10 nh

```
ros::NodeHandle bebop_controller::Waypoint::nh [protected]
```

Definition at line 34 of file [waypoint.h](#).

11.17.4.11 position_target_

```
trajectory_msgs::MultiDOFJointTrajectory bebop_controller::Waypoint::position_target_ [protected]
```

Definition at line 48 of file [waypoint.h](#).

11.17.4.12 setpoint_pub_

```
ros::Publisher bebop_controller::Waypoint::setpoint_pub_ [protected]
```

Definition at line 44 of file [waypoint.h](#).

11.17.4.13 status

```
enum Status bebop_controller::Waypoint::status [protected]
```

Definition at line 54 of file [waypoint.h](#).

11.17.4.14 wp_params_

```
WaypointParameters bebop_controller::Waypoint::wp_params_ [protected]
```

Definition at line 32 of file [waypoint.h](#).

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

- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/include/bebop_controller/[waypoint.h](#)
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/library/[waypoint.cpp](#)

11.18 WaypointParameters Struct Reference

```
#include <waypoint.h>
```

Public Attributes

- double [TimeBeforeTrajectory](#)
- double [TrajectoryTime](#)
- double [DiffTime](#)
- double [MarginTime](#)
- std::string [topic_command_trajectory](#)
- std::string [topic_csv_begin](#)
- std::string [topic_csv_end](#)

11.18.1 Detailed Description

Definition at line 8 of file [waypoint.h](#).

11.18.2 Member Data Documentation

11.18.2.1 DiffTime

```
double WaypointParameters::DiffTime
```

Definition at line 11 of file [waypoint.h](#).

11.18.2.2 MarginTime

```
double WaypointParameters::MarginTime
```

Definition at line 12 of file [waypoint.h](#).

11.18.2.3 TimeBeforeTrajectory

```
double WaypointParameters::TimeBeforeTrajectory
```

Definition at line 9 of file [waypoint.h](#).

11.18.2.4 topic_command_trajectory

```
std::string WaypointParameters::topic_command_trajectory
```

Definition at line 13 of file [waypoint.h](#).

11.18.2.5 topic_csv_begin

```
std::string WaypointParameters::topic_csv_begin
```

Definition at line 14 of file [waypoint.h](#).

11.18.2.6 topic_csv_end

```
std::string WaypointParameters::topic_csv_end
```

Definition at line 15 of file [waypoint.h](#).

11.18.2.7 TrajectoryTime

```
double WaypointParameters::TrajectoryTime
```

Definition at line 10 of file [waypoint.h](#).

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

- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/include/bebop_controller/[waypoint.h](#)

Chapter 12

File Documentation

12.1 documentation.md File Reference

Namespaces

- namespace [bebop_controller](#)
Namespace containing all the classes and functions of the Bebop Controller.
- namespace [gazebo](#)
Namespace containing all the classes and functions used to get the drone position data from Gazebo.
- namespace [plot](#)
Namespace that contains all the classes and functions used to generate graphs that display the test results.

12.2 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/include/bebop_controller/base_controller.h File Reference

Header file for the base class of controllers.

```
#include "bebop_controller/common.h"
```

Classes

- class [bebop_controller::BaseController](#)

Namespaces

- namespace [bebop_controller](#)
Namespace containing all the classes and functions of the Bebop Controller.

Macros

- `#define DISABLE_COMMANDS true`
- `#define BEBOP_COMMAND_TRAJECTORY "/bebop/command/trajectory"`
- `#define BEBOP_POSE "/bebop/pose"`
- `#define BEBOP_CMD_VEL "/bebop/cmd_vel"`
- `#define BEBOP_TAKEOFF "/bebop/takeoff"`
- `#define BEBOP_LAND "/bebop/land"`
- `#define CSV_END "/csv/end"`
- `#define SAFE_ZONE_X 5.0`
- `#define SAFE_ZONE_Y 5.0`
- `#define SAFE_ZONE_Z 5.0`
- `#define LIMIT_X 1.0`
- `#define LIMIT_Y 1.0`
- `#define LIMIT_Z 1.0`
- `#define LIMIT_YAW 1.0`
- `#define GRAVITY 9.80665`
- `#define MIN_VEL 0.001`
- `#define MIN_ACCEL 0.001`

12.2.1 Detailed Description

Header file for the base class of controllers.

Definition in file [base_controller.h](#).

12.2.2 Macro Definition Documentation

12.2.2.1 BEBOP_CMD_VEL

```
#define BEBOP_CMD_VEL "/bebop/cmd_vel"
```

Definition at line 9 of file [base_controller.h](#).

12.2.2.2 BEBOP_COMMAND_TRAJECTORY

```
#define BEBOP_COMMAND_TRAJECTORY "/bebop/command/trajectory"
```

Definition at line 7 of file [base_controller.h](#).

12.2.2.3 BEBOP_LAND

```
#define BEBOP_LAND "/bebop/land"
```

Definition at line 11 of file [base_controller.h](#).

12.2.2.4 BEBOP_POSE

```
#define BEBOP_POSE "/bebop/pose"
```

Definition at line 8 of file [base_controller.h](#).

12.2.2.5 BEBOP_TAKEOFF

```
#define BEBOP_TAKEOFF "/bebop/takeoff"
```

Definition at line 10 of file [base_controller.h](#).

12.2.2.6 CSV_END

```
#define CSV_END "/csv/end"
```

Definition at line 12 of file [base_controller.h](#).

12.2.2.7 DISABLE_COMMANDS

```
#define DISABLE_COMMANDS true
```

Definition at line 6 of file [base_controller.h](#).

12.2.2.8 GRAVITY

```
#define GRAVITY 9.80665
```

Definition at line 20 of file [base_controller.h](#).

12.2.2.9 LIMIT_X

```
#define LIMIT_X 1.0
```

Definition at line 16 of file [base_controller.h](#).

12.2.2.10 LIMIT_Y

```
#define LIMIT_Y 1.0
```

Definition at line 17 of file [base_controller.h](#).

12.2.2.11 LIMIT_YAW

```
#define LIMIT_YAW 1.0
```

Definition at line 19 of file [base_controller.h](#).

12.2.2.12 LIMIT_Z

```
#define LIMIT_Z 1.0
```

Definition at line 18 of file [base_controller.h](#).

12.2.2.13 MIN_ACCEL

```
#define MIN_ACCEL 0.001
```

Definition at line 22 of file [base_controller.h](#).

12.2.2.14 MIN_VEL

```
#define MIN_VEL 0.001
```

Definition at line 21 of file [base_controller.h](#).

12.2.2.15 SAFE_ZONE_X

```
#define SAFE_ZONE_X 5.0
```

Definition at line 13 of file [base_controller.h](#).

12.2.2.16 SAFE_ZONE_Y

```
#define SAFE_ZONE_Y 5.0
```

Definition at line 14 of file [base_controller.h](#).

12.2.2.17 SAFE_ZONE_Z

```
#define SAFE_ZONE_Z 5.0
```

Definition at line 15 of file [base_controller.h](#).

12.3 base_controller.h

[Go to the documentation of this file.](#)

```
00001
00003
00004 #include "bebop_controller/common.h"
00005
00006 #define DISABLE_COMMANDS true
00007 #define BEBOP_COMMAND_TRAJECTORY "/bebop/command/trajectory"
00008 #define BEBOP_POSE "/bebop/pose"
00009 #define BEBOP_CMD_VEL "/bebop/cmd_vel"
00010 #define BEBOP_TAKEOFF "/bebop/takeoff"
00011 #define BEBOP_LAND "/bebop/land"
00012 #define CSV_END "/csv/end"
00013 #define SAFE_ZONE_X 5.0
00014 #define SAFE_ZONE_Y 5.0
00015 #define SAFE_ZONE_Z 5.0
00016 #define LIMIT_X 1.0
00017 #define LIMIT_Y 1.0
00018 #define LIMIT_Z 1.0
00019 #define LIMIT_YAW 1.0
00020 #define GRAVITY 9.80665
00021 #define MIN_VEL 0.001
00022 #define MIN_ACCEL 0.001
00023
00024 namespace bebop_controller {
00025
00026     class BaseController{
00027     public:
00028         BaseController();
00029         ~BaseController();
00030
00031     protected:
00032         bool waypointHasBeenPublished_;
00033         bool takeoff;
00034         bool controller_active_;
00035         bool disable_commands;
00036         bool stop;
00037         double diff;
00038
00039         ros::Subscriber cmd_multi_dof_joint_trajectory_sub_;
00040         ros::Subscriber odom_sub_;
00041         ros::Subscriber end_sub_;
```

```

00042
00043     ros::Publisher motor_velocity_reference_pub_;
00044     ros::Publisher takeoff_pub_;
00045     ros::Publisher land_pub_;
00046     ros::Publisher odometry_filtered_pub_;
00047     ros::Publisher reference_angles_pub_;
00048     ros::Publisher smoothed_reference_pub_;
00049
00050     ros::Time lastTime;
00051     ros::Timer timeOut;
00052
00053     mav_msgs::EigenTrajectoryPoint command_trajectory_;
00054     mav_msgs::EigenOdometry odometry_;
00055     State state;
00056     State last_state;
00057     Vector3 safe_zone;
00058     Vector4 max_speed;
00059     Vector3 leash_length;
00060
00061     void MultiDOFJointTrajectory_CB(const trajectory_msgs::MultiDOFJointTrajectoryConstPtr&
trajectory_reference_msg);
00062     void TakeOff();
00063     void Land();
00064     void Odometry_CB(const geometry_msgs::PoseStamped& pose_msg);
00065     void Stop_CB(const std_msgs::Empty::ConstPtr& empty_msg);
00066     void TimeOut_CB(const ros::TimerEvent& event);
00067     void SetTrajectoryPoint(mav_msgs::EigenTrajectoryPoint& eigen_reference);
00068     void SetOdometry(mav_msgs::EigenOdometry& odometry);
00069     void Quaternion2Euler(double& roll, double& pitch, double& yaw) const;
00070     void GetErrors(Vector4& e);
00071     void GetVelocityErrors(Vector4& dot_e);
00072     void EstimateVelocity();
00073     void EstimateAcceleration();
00074     void Stop(bool failsafe);
00075     bool CheckSafeZone();
00076     void CalculateLeashLength(Vector4& e, Vector4& P);
00077     void LimitPositionErrors(Vector4& e);
00078     virtual void CalculateCommandVelocities(geometry_msgs::Twist& ref_command_signals);
00079 };
00080
00081 }

```

12.4 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_↵ controller/include/bebop_controller/common.h File Reference

File containing common functions used by the Bebop Controller.

```

#include <ros/ros.h>
#include <Eigen/Eigen>
#include <mav_msgs/eigen_mav_msgs.h>
#include <tf/tf.h>
#include <trajectory_msgs/MultiDOFJointTrajectory.h>
#include <geometry_msgs/PoseStamped.h>
#include <geometry_msgs/Twist.h>
#include <std_msgs/Empty.h>
#include <mav_msgs/conversions.h>

```

Classes

- struct [bebop_controller::Vector3](#)
Structure for storing 3-dimensional vector data.
- struct [bebop_controller::Vector4](#)
Structure for storing 4-dimensional vector data.
- struct [bebop_controller::State](#)
Structure for storing the drone state.
- struct [bebop_controller::Command_Velocities](#)
Structure for storing the command velocities.

Namespaces

- namespace [bebop_controller](#)

Namespace containing all the classes and functions of the Bebop Controller.

Functions

- double [bebop_controller::bound](#) (double value, double min, double max)
- double [bebop_controller::clamp](#) (double value, double lim)
- double [bebop_controller::sgn](#) (double value)
- double [bebop_controller::max](#) (double val1, double val2)
- template<typename T >
void [bebop_controller::GetRosParameter](#) (const ros::NodeHandle &nh, const std::string &key, const T &default_value, T *value)
Template function to get ROS parameters.
- double [bebop_controller::yawFromQuaternion](#) (const Eigen::Quaterniond &q)
- Eigen::Vector3d [bebop_controller::Quat2RPY](#) (Eigen::Vector4d &Quaternion)
- Eigen::Vector4d [bebop_controller::RPY2Quat](#) (Eigen::Vector3d &RPY)
- Eigen::Vector4d [vector4FromQuaternionMsg](#) (const geometry_msgs::Quaternion &msg)
- double [secsFromHeaderMsg](#) (const std_msgs::Header &msg)

12.4.1 Detailed Description

File containing common functions used by the Bebop Controller.

Definition in file [common.h](#).

12.4.2 Function Documentation

12.4.2.1 secsFromHeaderMsg()

```
double secsFromHeaderMsg (  
    const std_msgs::Header & msg ) [inline]
```

Function to get the seconds from a header message.

Returns

The value of seconds.

Definition at line 156 of file [common.h](#).

12.4.2.2 vector4FromQuaternionMsg()

```
Eigen::Vector4d vector4FromQuaternionMsg (
    const geometry_msgs::Quaternion & msg ) [inline]
```

Function to get a 4-dimensional vector from a Quaternion message.

Returns

A 4-dimensional vector with the quaternion values.

Definition at line 150 of file [common.h](#).

12.5 common.h

[Go to the documentation of this file.](#)

```
00001
00002
00003
00004 #include <ros/ros.h>
00005 #include <Eigen/Eigen>
00006 #include <mav_msgs/eigen_mav_msgs.h>
00007 #include <tf/tf.h>
00008 #include <trajectory_msgs/MultiDOFJointTrajectory.h>
00009 #include <geometry_msgs/PoseStamped.h>
00010 #include <geometry_msgs/Twist.h>
00011 #include <std_msgs/Empty.h>
00012 #include <mav_msgs/conversions.h>
00013
00014 static const float DEG_2_RAD = M_PI / 180.0;
00015 static const float RAD_2_DEG = 180.0 / M_PI;
00016
00017 namespace bebop_controller {
00018
00019
00024     double bound(double value, double min, double max){
00025         if (value < min){
00026             return min;
00027         }
00028         else if(value > max){
00029             return max;
00030         }
00031         else {
00032             return value;
00033         }
00034     }
00035
00040     double clamp(double value, double lim) {
00041         return bound(value, -std::abs(lim), std::abs(lim));
00042     }
00043
00047     double sgn(double value){
00048         if (value > 0){
00049             return 1;
00050         }
00051         else if (value < 0){
00052             return -1;
00053         }
00054         else {
00055             return 0;
00056         }
00057     }
00058
00063     double max(double val1, double val2){
00064         if (val1 > val2) {
00065             return val1;
00066         }
00067         else {
00068             return val2;
00069         }
00070     }
00071
00073     struct Vector3 {
00074         double x;
00075         double y;
00076         double z;
```

```
00077     };
00078
00080     struct Vector4 {
00081         double x;
00082         double y;
00083         double z;
00084         double yaw;
00085     };
00086
00088     struct State {
00089         Vector3 position;
00090         Vector3 orientation;
00091         Vector3 velocity;
00092         Vector3 angular_velocity;
00093         Vector3 acceleration;
00094         Vector3 angular_acceleration;
00095     };
00096
00098     struct Command_Velocities {
00099         double x;
00100         double y;
00101         double z;
00102         double yaw;
00103     };
00104
00106     template<typename T> inline void GetRosParameter(const ros::NodeHandle& nh,
00107                                                     const std::string& key,
00108                                                     const T& default_value,
00109                                                     T* value) {
00110         ROS_ASSERT(value != nullptr);
00111         bool have_parameter = nh.getParam(key, *value);
00112         if (!have_parameter) {
00113             ROS_WARN_STREAM("[rosparam]: could not find parameter " « nh.getNamespace()
00114                             « "/" « key « ", setting to default: " « default_value);
00115             *value = default_value;
00116         }
00117     }
00118
00121     inline double yawFromQuaternion(const Eigen::Quaterniond& q) {
00122         return atan2(2.0 * (q.w() * q.z() + q.x() * q.y()),
00123                     1.0 - 2.0 * (q.y() * q.y() + q.z() * q.z()));
00124     }
00125
00128     Eigen::Vector3d Quat2RPY(Eigen::Vector4d& Quaternion) {
00129         double roll, pitch, yaw;
00130         tf::Quaternion q(Quaternion.x(), Quaternion.y(), Quaternion.z(), Quaternion.w());
00131         tf::Matrix3x3 m(q);
00132         m.getRPY(roll, pitch, yaw);
00133         return Eigen::Vector3d(roll, pitch, yaw);
00134     }
00135
00138     Eigen::Vector4d RPY2Quat(Eigen::Vector3d& RPY) {
00139         tf::Quaternion q;
00140         tf::Matrix3x3 m;
00141         m.setEulerYPR(RPY.z(), RPY.y(), RPY.x());
00142         m.getRotation(q);
00143         return Eigen::Vector4d(q.x(), q.y(), q.z(), q.w());
00144     }
00145
00146 }
00147
00150 inline Eigen::Vector4d vector4FromQuaternionMsg(const geometry_msgs::Quaternion& msg) {
00151     return Eigen::Vector4d(msg.x, msg.y, msg.z, msg.w);
00152 }
00153
00156 inline double secsFromHeaderMsg(const std_msgs::Header& msg) {
00157     return (double) (msg.stamp.sec) + (double) (msg.stamp.nsec)/1.0e9;
00158 }
```

12.6 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_↵ controller/include/bebop_controller/waypoint.h File Reference

Header file for the base class of the waypoint generator.

```
#include "bebop_controller/common.h"
#include <thread>
#include <chrono>
```

Classes

- struct [WaypointParameters](#)
- class [bebop_controller::Waypoint](#)

Namespaces

- namespace [bebop_controller](#)
Namespace containing all the classes and functions of the Bebop Controller.

Enumerations

- enum [Status](#) { [BeforeTrajectory](#) , [Trajectory](#) , [AfterTrajectory](#) }

12.6.1 Detailed Description

Header file for the base class of the waypoint generator.

Definition in file [waypoint.h](#).

12.6.2 Enumeration Type Documentation

12.6.2.1 Status

enum [Status](#)

Enumerator

BeforeTrajectory	
Trajectory	
AfterTrajectory	

Definition at line 18 of file [waypoint.h](#).

12.7 waypoint.h

[Go to the documentation of this file.](#)

```
00001
00002
00003
00004 #include "bebop_controller/common.h"
00005 #include <thread>
00006 #include <chrono>
00007
00008 struct WaypointParameters {
00009     double TimeBeforeTrajectory;
```

```
00010     double TrajectoryTime;
00011     double DiffTime;
00012     double MarginTime;
00013     std::string topic_command_trajectory;
00014     std::string topic_csv_begin;
00015     std::string topic_csv_end;
00016 };
00017
00018 enum Status {
00019     BeforeTrajectory,
00020     Trajectory,
00021     AfterTrajectory,
00022 };
00023
00024 namespace bebop_controller {
00025
00026     class Waypoint{
00027     public:
00028         Waypoint(WaypointParameters param);
00029         ~Waypoint();
00030
00031     protected:
00032         WaypointParameters wp_params_;
00033
00034         ros::NodeHandle nh;
00035         ros::NodeHandle _n1;
00036         ros::NodeHandle _n2;
00037         ros::NodeHandle _n3;
00038         ros::Timer _timer1;
00039         ros::Timer _timer2;
00040         ros::Timer _timer3;
00041
00042         ros::Publisher Begin_;
00043         ros::Publisher End_;
00044         ros::Publisher setpoint_pub_;
00045
00046         ros::Time Initial_Time;
00047
00048         trajectory_msgs::MultiDOFJointTrajectory position_target_;
00049
00050         void Start_CB(const ros::TimerEvent& event);
00051         void Stop_CB(const ros::TimerEvent& event);
00052         virtual void Trajectory_CB(const ros::TimerEvent& event);
00053
00054         enum Status status;
00055     };
00056
00057 }
```

12.8 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_↵ controller/launch/citc_controller_angles.launch File Reference

Launch file to run a test using the CITC controller with reference angles.

12.8.1 Detailed Description

Launch file to run a test using the CITC controller with reference angles.

Definition in file [citc_controller_angles.launch](#).

12.9 citc_controller_angles.launch

[Go to the documentation of this file.](#)

```
00001 <?xml version="1.0"?>
00002
00003 <launch>
00004     <arg name="Dir" default="$(env HOME)/CSV"/>
00005     <param name="Subfolder" type="str" command="date +%d-%m-%Y_%Ih%Mm%Ss'"/>
```

```

00006     <arg name="GazeboRealTime" default="0.5"/>
00007     <arg name="YAML" default="citc_controller_angles.yaml"/>
00008
00009     <node name="gazebo" pkg="bebop_controller" type="gazebo.py" output="screen">
00010         <param name="Topic" type="str" value="/bebop/pose"/>
00011         <param name="GazeboRealTime" type="double" value="$(arg GazeboRealTime)"/>
00012     </node>
00013
00014     <node name="citc_controller_angles" pkg="bebop_controller" type="citc_controller_angles"
output="screen">
00015         <rosparam command="load" file="$(find bebop_controller)/resource/citc_controller_angles.yaml" />
00016         <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
00017         <rosparam command="load" file="$(find bebop_controller)/resource/normalize_angles.yaml" />
00018         <rosparam command="load" file="$(find bebop_controller)/resource/safe_zone.yaml" />
00019         <rosparam command="load" file="$(find bebop_controller)/resource/max_speed.yaml" />
00020     </node>
00021
00022     <node name="waypoint" pkg="bebop_controller" type="sinusoidal" output="screen">
00023         <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
00024         <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
00025         <rosparam command="load" file="$(find bebop_controller)/resource/trajectory.yaml" />
00026     </node>
00027
00028     <node name="data_to_csv" pkg="bebop_controller" type="data_to_csv" output="screen">
00029         <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
00030         <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
00031         <param name="Dir" type="str" value="$(arg Dir)"/>
00032     </node>
00033
00034     <node name="plot" pkg="bebop_controller" type="plot.py" output="screen">
00035         <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
00036         <param name="Dir" type="str" value="$(arg Dir)"/>
00037         <param name="YAML" type="str" value="$(arg YAML)"/>
00038     </node>
00039
00040 </launch>

```

12.10 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/citc_controller_twist.launch File Reference

Launch file to run a test using the CITC controller with velocity commands.

12.10.1 Detailed Description

Launch file to run a test using the CITC controller with velocity commands.

Definition in file [citc_controller_twist.launch](#).

12.11 citc_controller_twist.launch

[Go to the documentation of this file.](#)

```

00001 <?xml version="1.0"?>
00002
00003 <launch>
00004     <arg name="Dir" default="$(env HOME)/CSV"/>
00005     <param name="Subfolder" type="str" command="date +%d-%m-%Y_%Ih%Mm%Ss"/>
00006     <arg name="GazeboRealTime" default="0.5"/>
00007     <arg name="YAML" default="citc_controller_twist.yaml"/>
00008
00009     <node name="gazebo" pkg="bebop_controller" type="gazebo.py" output="screen">
00010         <param name="Topic" type="str" value="/bebop/pose"/>
00011         <param name="GazeboRealTime" type="double" value="$(arg GazeboRealTime)"/>
00012     </node>
00013
00014     <node name="citc_controller_twist" pkg="bebop_controller" type="citc_controller_twist"
output="screen">
00015         <rosparam command="load" file="$(find bebop_controller)/resource/citc_controller_twist.yaml" />
00016         <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />

```

```

00017     <rosparam command="load" file="$(find bebop_controller)/resource/normalize_angles.yaml" />
00018     <rosparam command="load" file="$(find bebop_controller)/resource/safe_zone.yaml" />
00019     <rosparam command="load" file="$(find bebop_controller)/resource/max_speed.yaml" />
00020 </node>
00021
00022 <node name="waypoint" pkg="bebop_controller" type="sinusoidal" output="screen">
00023     <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
00024     <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
00025     <rosparam command="load" file="$(find bebop_controller)/resource/trajectory.yaml" />
00026 </node>
00027
00028 <node name="data_to_csv" pkg="bebop_controller" type="data_to_csv" output="screen">
00029     <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
00030     <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
00031     <param name="Dir" type="str" value="$(arg Dir)" />
00032 </node>
00033
00034 <node name="plot" pkg="bebop_controller" type="plot.py" output="screen">
00035     <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
00036     <param name="Dir" type="str" value="$(arg Dir)" />
00037     <param name="YAML" type="str" value="$(arg YAML)" />
00038 </node>
00039
00040 </launch>

```

12.12 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/gazebo.launch File Reference

Launch file example to run the publisher that gets the position data from the simulator.

12.12.1 Detailed Description

Launch file example to run the publisher that gets the position data from the simulator.

Definition in file [gazebo.launch](#).

12.13 gazebo.launch

[Go to the documentation of this file.](#)

```

00001 <?xml version="1.0"?>
00002
00003 <launch>
00004     <node name="gazebo" pkg="bebop_controller" type="gazebo.py" output="screen">
00005         <param name="Topic" type="str" value="/bebop/pose"/>
00006         <param name="GazeboRealTime" type="double" value="0.5"/>
00007     </node>
00008 </launch>

```

12.14 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/pid_controller_angles.launch File Reference

Launch file to run a test using the PID controller with reference angles.

12.14.1 Detailed Description

Launch file to run a test using the PID controller with reference angles.

Definition in file [pid_controller_angles.launch](#).

12.15 pid_controller_angles.launch

[Go to the documentation of this file.](#)

```

00001 <?xml version="1.0"?>
00002
00003 <launch>
00004   <arg name="Dir" default="$(env HOME)/CSV"/>
00005   <param name="Subfolder" type="str" command="date +%d-%m-%Y_%Ih%Mm%Ss'"/>
00006   <arg name="GazeboRealTime" default="0.5"/>
00007   <arg name="YAML" default="pid_controller_angles.yaml"/>
00008
00009   <node name="gazebo" pkg="bebop_controller" type="gazebo.py" output="screen">
00010     <param name="Topic" type="str" value="/bebop/pose"/>
00011     <param name="GazeboRealTime" type="double" value="$(arg GazeboRealTime)"/>
00012   </node>
00013
00014   <node name="pid_controller_angles" pkg="bebop_controller" type="pid_controller_angles"
00015     output="screen">
00016     <rosparam command="load" file="$(find bebop_controller)/resource/pid_controller_angles.yaml" />
00017     <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
00018     <rosparam command="load" file="$(find bebop_controller)/resource/normalize_angles.yaml" />
00019     <rosparam command="load" file="$(find bebop_controller)/resource/safe_zone.yaml" />
00020     <rosparam command="load" file="$(find bebop_controller)/resource/max_speed.yaml" />
00021   </node>
00022   <node name="waypoint" pkg="bebop_controller" type="sinusoidal" output="screen">
00023     <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
00024     <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
00025     <rosparam command="load" file="$(find bebop_controller)/resource/trajectory.yaml" />
00026   </node>
00027   <node name="data_to_csv" pkg="bebop_controller" type="data_to_csv" output="screen">
00028     <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
00029     <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
00030     <param name="Dir" type="str" value="$(arg Dir)"/>
00031   </node>
00032   <node name="plot" pkg="bebop_controller" type="plot.py" output="screen">
00033     <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
00034     <param name="Dir" type="str" value="$(arg Dir)"/>
00035     <param name="YAML" type="str" value="$(arg YAML)"/>
00036   </node>
00037
00038 </launch>
00039
00040 </launch>

```

12.16 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/pid_controller_twist.launch File Reference

Launch file to run a test using the PID controller with velocity commands.

12.16.1 Detailed Description

Launch file to run a test using the PID controller with velocity commands.

Definition in file [pid_controller_twist.launch](#).

12.17 pid_controller_twist.launch

[Go to the documentation of this file.](#)

```

00001 <?xml version="1.0"?>
00002
00003 <launch>
00004   <arg name="Dir" default="$(env HOME)/CSV"/>
00005   <param name="Subfolder" type="str" command="date +%d-%m-%Y_%Ih%Mm%Ss'"/>
00006   <arg name="GazeboRealTime" default="0.5"/>
00007   <arg name="YAML" default="pid_controller_twist.yaml"/>

```



```
00008
00009 <node name="gazebo" pkg="bebop_controller" type="gazebo.py" output="screen">
00010   <param name="Topic" type="str" value="/bebop/pose"/>
00011   <param name="GazeboRealTime" type="double" value="$(arg GazeboRealTime)"/>
00012 </node>
00013
00014 <node name="pid_controller_twist" pkg="bebop_controller" type="pid_controller_twist"
output="screen">
00015   <rosparam command="load" file="$(find bebop_controller)/resource/pid_controller_twist.yaml" />
00016   <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
00017   <rosparam command="load" file="$(find bebop_controller)/resource/normalize_twist.yaml" />
00018   <rosparam command="load" file="$(find bebop_controller)/resource/safe_zone.yaml" />
00019   <rosparam command="load" file="$(find bebop_controller)/resource/max_speed.yaml" />
00020 </node>
00021
00022 <node name="waypoint" pkg="bebop_controller" type="sinusoidal" output="screen">
00023   <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
00024   <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
00025   <rosparam command="load" file="$(find bebop_controller)/resource/trajectory.yaml" />
00026 </node>
00027
00028 <node name="data_to_csv" pkg="bebop_controller" type="data_to_csv" output="screen">
00029   <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
00030   <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
00031   <param name="Dir" type="str" value="$(arg Dir)"/>
00032 </node>
00033
00034 <node name="plot" pkg="bebop_controller" type="plot.py" output="screen">
00035   <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
00036   <param name="Dir" type="str" value="$(arg Dir)"/>
00037   <param name="YAML" type="str" value="$(arg YAML)"/>
00038 </node>
00039
00040 </launch>
```

12.18 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_↵ controller/launch/proportional_controller.launch File Reference

Launch file to run a test using the proportional controller.

12.18.1 Detailed Description

Launch file to run a test using the proportional controller.

Definition in file [proportional_controller.launch](#).

12.19 proportional_controller.launch

[Go to the documentation of this file.](#)

```
00001 <?xml version="1.0"?>
00002
00003 <launch>
00004   <arg name="Dir" default="$(env HOME)/CSV"/>
00005   <param name="Subfolder" type="str" command="date +%d-%m-%Y_%Ih%Mm%Ss'"/>
00006   <arg name="GazeboRealTime" default="0.5"/>
00007   <arg name="YAML" default="proportional_controller.yaml"/>
00008
00009   <node name="gazebo" pkg="bebop_controller" type="gazebo.py" output="screen">
00010     <param name="Topic" type="str" value="/bebop/pose"/>
00011     <param name="GazeboRealTime" type="double" value="$(arg GazeboRealTime)"/>
00012   </node>
00013
00014   <node name="proportional_controller" pkg="bebop_controller" type="proportional_controller"
output="screen">
00015     <rosparam command="load" file="$(find bebop_controller)/resource/proportional_controller.yaml"
/>
00016     <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
00017     <rosparam command="load" file="$(find bebop_controller)/resource/normalize_twist.yaml" />
```

```

00018     <rosparam command="load" file="$(find bebop_controller)/resource/safe_zone.yaml" />
00019     <rosparam command="load" file="$(find bebop_controller)/resource/max_speed.yaml" />
00020 </node>
00021
00022 <node name="waypoint" pkg="bebop_controller" type="sinusoidal" output="screen">
00023   <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
00024   <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
00025   <rosparam command="load" file="$(find bebop_controller)/resource/trajectory.yaml" />
00026 </node>
00027
00028 <node name="data_to_csv" pkg="bebop_controller" type="data_to_csv" output="screen">
00029   <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
00030   <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
00031   <param name="Dir" type="str" value="$(arg Dir)" />
00032 </node>
00033
00034 <node name="plot" pkg="bebop_controller" type="plot.py" output="screen">
00035   <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
00036   <param name="Dir" type="str" value="$(arg Dir)" />
00037   <param name="YAML" type="str" value="$(arg YAML)" />
00038 </node>
00039
00040 </launch>

```

12.20 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/square_root_controller.launch File Reference

Launch file to run a test using the square root controller.

12.20.1 Detailed Description

Launch file to run a test using the square root controller.

Definition in file [square_root_controller.launch](#).

12.21 square_root_controller.launch

[Go to the documentation of this file.](#)

```

00001 <?xml version="1.0"?>
00002
00003 <launch>
00004   <arg name="Dir" default="$(env HOME)/CSV"/>
00005   <param name="Subfolder" type="str" command="date +%d-%m-%Y_%H%Mm%Ss"/>
00006   <arg name="GazeboRealTime" default="0.5"/>
00007   <arg name="YAML" default="square_root_controller.yaml"/>
00008
00009   <node name="gazebo" pkg="bebop_controller" type="gazebo.py" output="screen">
00010     <param name="Topic" type="str" value="/bebop/pose"/>
00011     <param name="GazeboRealTime" type="double" value="$(arg GazeboRealTime)" />
00012   </node>
00013
00014   <node name="square_root_controller" pkg="bebop_controller" type="square_root_controller"
00015     output="screen">
00016     <rosparam command="load" file="$(find bebop_controller)/resource/square_root_controller.yaml" />
00017     <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
00018     <rosparam command="load" file="$(find bebop_controller)/resource/normalize_twist.yaml" />
00019     <rosparam command="load" file="$(find bebop_controller)/resource/safe_zone.yaml" />
00020     <rosparam command="load" file="$(find bebop_controller)/resource/max_speed.yaml" />
00021   </node>
00022
00023   <node name="waypoint" pkg="bebop_controller" type="sinusoidal" output="screen">
00024     <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
00025     <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
00026     <rosparam command="load" file="$(find bebop_controller)/resource/trajectory.yaml" />
00027   </node>
00028
00029   <node name="data_to_csv" pkg="bebop_controller" type="data_to_csv" output="screen">
00030     <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />

```

12.22

C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/vrpn_client_ros.launch File Reference 75

```
00030     <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
00031     <param name="Dir" type="str" value="$(arg Dir)" />
00032 </node>
00033
00034 <node name="plot" pkg="bebop_controller" type="plot.py" output="screen">
00035   <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
00036   <param name="Dir" type="str" value="$(arg Dir)" />
00037   <param name="YAML" type="str" value="$(arg YAML)" />
00038 </node>
00039
00040 </launch>
```

12.22 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/vrpn_client_ros.launch File Reference ↩

Launch file example to use vrpn client as ROS publisher.

12.22.1 Detailed Description

Launch file example to use vrpn client as ROS publisher.

Definition in file [vrpn_client_ros.launch](#).

12.23 vrpn_client_ros.launch

[Go to the documentation of this file.](#)

```
00001 <?xml version="1.0"?>
00002
00003 <launch>
00004   <arg name="server" default="192.168.42.16"/>
00005   <node pkg="vrpn_client_ros" type="vrpn_client_node" name="vrpn_client_node" output="screen">
00006     <rosparam subst_value="true">
00007       server: $(arg server)
00008       port: 3883
00009       frame_id: world
00010       broadcast_tf: true
00011       refresh_tracker_frequency: 120.0
00012     </rosparam>
00013   </node>
00014 </launch>
```

12.24 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/citc_controller_angles.yaml File Reference ↩

YAML file for CITC controller parameters, using reference angles.

12.24.1 Detailed Description

YAML file for CITC controller parameters, using reference angles.

The following parameters can be configured in this file.

Parameters

<i>Gains/K_1x</i>	Ganancia K_1
--------------------------------	----------------

Definition in file [citic_controller_angles.yaml](#).

12.25 citic_controller_angles.yaml

[Go to the documentation of this file.](#)

```
00001 # CITC controller parameters
00002 Gains: {K1x: 0.9,
00003         K1y: 0.9,
00004         K1z: 5.0,
00005         K1yaw: 0.3,
00006         K2x: 1.2,
00007         K2y: 1.2,
00008         K2z: 3.0,
00009         K2yaw: 2.4,
00010         K3x: 0,
00011         K3y: 0,
00012         K3z: 0,
00013         K3yaw: 0,
00014         K4x: 0,
00015         K4y: 0,
00016         K4z: 0,
00017         K4yaw: 0}
00018 Reference_Gains: {X: 0,
00019                  Y: 0,
00020                  Z: 0,
00021                  Yaw: 0}
00022 Lambda: {X: 1.3151,
00023          Y: 1.3151,
00024          Z: 4.6697}
00025 Mass: 0.5
00026 Sigma: 0.8
00027 Disable_Commands: False
```

12.26 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/citic_controller_twist.yaml File Reference ↩

YAML file for CITC controller parameters, using velocity commands.

12.26.1 Detailed Description

YAML file for CITC controller parameters, using velocity commands.

Definition in file [citic_controller_twist.yaml](#).

12.27 citc_controller_twist.yaml

[Go to the documentation of this file.](#)

```
00001 # CITC controller parameters
00002 Gains: {K1x: 2.05,
00003         K1y: 2.05,
00004         K1z: 0.01,
00005         K1yaw: 0.009172,
00006         K2x: 5.45,
00007         K2y: 5.45,
00008         K2z: 0.1,
00009         K2yaw: 0.036114,
00010         K3x: 0,
00011         K3y: 0,
00012         K3z: 0,
00013         K3yaw: 0,
00014         K4x: 0,
00015         K4y: 0,
00016         K4z: 0,
00017         K4yaw: 0}
00018 Reference_Gains: {X: 0,
00019                  Y: 0,
00020                  Z: 0,
00021                  Yaw: 0}
00022 Lambda: {X: 1.3151,
00023          Y: 1.3151,
00024          Z: 4.6697}
00025 Mass: 0.5
00026 Sigma: 0.8
00027 Disable_Commands: False
```

12.28 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/max_speed.yaml File Reference

YAML file to set the maximum velocity allowed for velocity commands.

12.28.1 Detailed Description

YAML file to set the maximum velocity allowed for velocity commands.

Definition in file [max_speed.yaml](#).

12.29 max_speed.yaml

[Go to the documentation of this file.](#)

```
00001 Max_Speed: {X: 0.3,
00002             Y: 0.3,
00003             Z: 0.3,
00004             Yaw: 0.0}
```

12.30 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/normalize_angles.yaml File Reference

YAML file to configure the parameters used to normalize speed for controllers using reference angles.

12.30.1 Detailed Description

YAML file to configure the parameters used to normalize speed for controllers using reference angles.

Definition in file [normalize_angles.yaml](#).

12.31 normalize_angles.yaml

[Go to the documentation of this file.](#)

```
00001 Normalize: {Max_Tilt_Angle: 20.0,  
00002             Max_Vertical_Speed: 1.0,  
00003             Max_Rotation_Speed: 100.0}
```

12.32 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/normalize_twist.yaml File Reference

YAML file to configure the parameters used to normalize speed for controllers using velocity commands.

12.32.1 Detailed Description

YAML file to configure the parameters used to normalize speed for controllers using velocity commands.

Definition in file [normalize_twist.yaml](#).

12.33 normalize_twist.yaml

[Go to the documentation of this file.](#)

```
00001 Normalize: {Max_Horizontal_Speed: 9.5,  
00002             Max_Vertical_Speed: 1.0,  
00003             Max_Rotation_Speed: 100.0}
```

12.34 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/pid_controller_angles.yaml File Reference

YAML file for PID controller parameters, using reference angles.

12.34.1 Detailed Description

YAML file for PID controller parameters, using reference angles.

Definition in file [pid_controller_angles.yaml](#).

12.35 pid_controller_angles.yaml

[Go to the documentation of this file.](#)

```
00001 # Proportional integral derivative controller parameters
00002 Gains: {Px: 1.8,
00003          Py: 1.8,
00004          Pz: 5.0,
00005          Pyaw: 4.8,
00006          Dx: 1.4,
00007          Dy: 1.4,
00008          Dz: 7.2,
00009          Dyaw: 0.6,
00010          Ix: 0.3,
00011          Iy: 0.3,
00012          Iz: 1.0,
00013          Iyaw: 1.0}
00014 Limits: {Ix: 3.0,
00015           Iy: 3.0,
00016           Iz: 1.0,
00017           Iyaw: 10.0}
00018 Reference_Gains: {X: 1.0,
00019                  Y: 1.0,
00020                  Z: 1.0,
00021                  Yaw: 1.0}
00022 Lambda: {X: 1.3151,
00023           Y: 1.3151,
00024           Z: 4.6697}
00025 Mass: 0.5
00026 Disable_Commands: False
```

12.36 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/pid_controller_twist.yaml File Reference

YAML file for PID controller parameters, using velocity commands.

12.36.1 Detailed Description

YAML file for PID controller parameters, using velocity commands.

Definition in file [pid_controller_twist.yaml](#).

12.37 pid_controller_twist.yaml

[Go to the documentation of this file.](#)

```
00001 # Proportional integral derivative controller parameters
00002 Gains: {Px: 0.9,
00003          Py: 0.9,
00004          Pz: 12.0,
00005          Pyaw: 0.0,
00006          Dx: 0.85,
00007          Dy: 0.85,
00008          Dz: 10.0,
00009          Dyaw: 0.0,
00010          Ix: 0.2,
00011          Iy: 0.2,
00012          Iz: 8.0,
00013          Iyaw: 0.0}
00014 Limits: {Ix: 3.0,
00015           Iy: 3.0,
00016           Iz: 1.0,
00017           Iyaw: 10.0}
00018 Reference_Gains: {X: 1.0,
00019                  Y: 1.0,
00020                  Z: 1.0,
00021                  Yaw: 1.0}
00022 Lambda: {X: 1.3151,
00023           Y: 1.3151,
00024           Z: 4.6697}
00025 Mass: 0.5
00026 Disable_Commands: False
```

12.38 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/proportional_controller.yaml File Reference

YAML file for proportional controller parameters.

12.38.1 Detailed Description

YAML file for proportional controller parameters.

Definition in file [proportional_controller.yaml](#).

12.39 proportional_controller.yaml

[Go to the documentation of this file.](#)

```
00001 # Proportional controller parameters
00002 Gains: {Px: 1.2,
00003         Py: 1.2,
00004         Pz: 1.0,
00005         Pyaw: 1.0}
00006 Reference_Gains: {X: 1.0,
00007                  Y: 1.0,
00008                  Z: 1.0,
00009                  Yaw: 1.0}
00010 Disable_Commands: False
```

12.40 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/safe_zone.yaml File Reference

YAML file to specify safe zone margins.

12.40.1 Detailed Description

YAML file to specify safe zone margins.

Definition in file [safe_zone.yaml](#).

12.41 safe_zone.yaml

[Go to the documentation of this file.](#)

```
00001 Safe_Zone: {X: 1.6,
00002             Y: 0.77,
00003             Z: 1.85}
```

12.42 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/square_root_controller.yaml File Reference

YAML file for square root controller parameters.

12.42.1 Detailed Description

YAML file for square root controller parameters.

Definition in file [square_root_controller.yaml](#).

12.43 square_root_controller.yaml

[Go to the documentation of this file.](#)

```
00001 # Square root controller parameters
00002 Gains: {Px: 1000.0,
00003         Py: 1000.0,
00004         Pz: 5.0,
00005         Pyaw: 3.0}
00006 Reference_Gains: {X: 1.0,
00007                  Y: 1.0,
00008                  Z: 1.0,
00009                  Yaw: 1.0}
00010 Disable_Commands: False
```

12.44 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/topics.yaml File Reference

YAML file to set the topics used by the nodes.

12.44.1 Detailed Description

YAML file to set the topics used by the nodes.

Definition in file [topics.yaml](#).

12.45 topics.yaml

[Go to the documentation of this file.](#)

```
00001 # Topics
00002 Topics: { Command_Trajectory: "/bebop/command/trajectory",
00003         Pose: "/vrpn_client_node/RigidBody1/pose",
00004         CMD_Vel: "/bebop/cmd_vel",
00005         TakeOff: "/bebop/takeoff",
00006         Land: "/bebop/land",
00007         CSV_Begin: "/csv/begin",
00008         CSV_End: "/csv/end",
00009         Velocities: "/bebop/odom"}
00010 # /vrpn_client_node/RigidBody1/pose
```

12.46 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/trajectory.yaml File Reference

YAML file to set the trajectory parameters.

12.46.1 Detailed Description

YAML file to set the trajectory parameters.

Definition in file [trajectory.yaml](#).

12.47 trajectory.yaml

[Go to the documentation of this file.](#)

```
00001 Trajectory: { X_Initial: 0,
00002                 Y_Initial: 0,
00003                 Z_Initial: 1.4,
00004                 X_Distance: 1.2,
00005                 Y_Distance: 0.6,
00006                 Z_Distance: 0.3}
00007 Yaw_Enabled: False
00008 Yaw_Offset: 1.57
```

12.48 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/waypoint.yaml File Reference

YAML file to configure trajectory times.

12.48.1 Detailed Description

YAML file to configure trajectory times.

Definition in file [waypoint.yaml](#).

12.49 waypoint.yaml

[Go to the documentation of this file.](#)

```
00001 Waypoint: { TimeBeforeTrajectory: 20,
00002               TrajectoryTime: 30,
00003               MarginTime: 10,
00004               DiffTime: 0.01}
```

12.50 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/scripts/gazebo.py File Reference

Node file to get the drone position data from Gazebo.

Classes

- class [gazebo.SphinxPublisher](#)

Class of the publisher that gets the drone position data from the simulator.

Namespaces

- namespace [gazebo](#)

Namespace containing all the classes and functions used to get the drone position data from Gazebo.

12.50.1 Detailed Description

Node file to get the drone position data from Gazebo.

This node requires the following parameters to run.

Parameters

<code>~Topic</code>	Topic to publish the position data.
<code>~GazeboRealTime</code>	Parameter used to compensate the gazebo real time which in most cases differs from the ROS time.

Definition in file [gazebo.py](#).

12.51 gazebo.py

[Go to the documentation of this file.](#)

```

00001 #!/usr/bin/env python
00002
00003
00009
00010 from subprocess import PIPE, Popen
00011 from threading import Thread
00012 import sys
00013 import numpy as np
00014 import re
00015 from queue import Queue, Empty
00016 import rospy
00017 from geometry_msgs.msg import PoseStamped
00018 from tf.transformations import quaternion_from_euler
00019
00020
00021 class SphinxPublisher:
00022     def __init__(self):
00023         self.q = Queue()
00024         self.pos = np.zeros(3)
00025         self.att = np.zeros(3)
00026         self.cont = 1
00027
00028         # Run the command
00029         ON_POSIX = 'posix' in sys.builtin_module_names
00030         command = "tlm-data-logger -r 0 inet:127.0.0.1:9060"
00031         p = Popen(command, stdout=PIPE, bufsize=1, close_fds=ON_POSIX, shell=True)
00032
00033         # Create a thread which dies with main program
00034         t = Thread(target=self.process_output, args=(p.stdout, self.q))
00035         t.daemon = True
00036         t.start()
00037         Topic = rospy.get_param('~Topic')
00038         GazeboRealTime = rospy.get_param('~GazeboRealTime')
00039         self.odom_pub = rospy.Publisher(Topic, PoseStamped, queue_size=10)
00040         Period = 1.0/(120.0*GazeboRealTime)
00041         rospy.Timer(rospy.Duration(Period), self.publish)
00042         rospy.spin()
00043
00044
00045     def process_output(self, out, queue):
00046         for line in iter(out.readline, b''):
00047             line = str(line)
00048             if ".worldPosition" in line:
00049                 number = re.findall(r"[+]?[d*]\.[d+]", line)[0]
00050                 if "." in line:

```

```

00051         self.pos[0] = float(number)
00052     if ".y" in line:
00053         self.pos[1] = float(number)
00054     if ".z" in line:
00055         self.pos[2] = float(number)
00056
00057
00058     if ".worldAttitude" in line:
00059         number = re.findall(r"[+]?[d+]\.d+", line)[0]
00060     if ".x" in line:
00061         self.att[0] = float(number)
00062     if ".y" in line:
00063         self.att[1] = float(number)
00064     if ".z" in line:
00065         self.att[2] = float(number)
00066
00067     queue.put(line)
00068     out.close()
00069
00070
00071 def publish(self, event):
00072     try:
00073         line = self.q.get_nowait()
00074     except Empty:
00075         self.q.queue.clear()
00076     qx, qy, qz, qw = quaternion_from_euler(self.att[0], self.att[1], self.att[2], 'sxyz')
00077     pose = PoseStamped()
00078     pose.header.seq = self.cont
00079     pose.header.stamp = rospy.Time.now()
00080     pose.header.frame_id = "bebop2"
00081     pose.pose.position.x = self.pos[1]
00082     pose.pose.position.y = self.pos[2]
00083     pose.pose.position.z = self.pos[0]
00084     pose.pose.orientation.x = qy
00085     pose.pose.orientation.y = qz
00086     pose.pose.orientation.z = qx
00087     pose.pose.orientation.w = qw
00088     self.cont += 1
00089     self.odom_pub.publish(pose)
00090
00091 if __name__ == '__main__':
00092     rospy.init_node('SphinxPublisher')
00093     publisher = SphinxPublisher()

```

12.52 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_↵ controller/scripts/plot.py File Reference

Node file for plotting test results.

Classes

- class [plot.Plots](#)

Namespaces

- namespace [plot](#)

Namespace that contains all the classes and functions used to generate graphs that display the test results.

Variables

- [plot.path](#) = rospy.get_param('~Dir')
- [plot.sub](#) = rospy.get_param('/Subfolder')
- [plot.topic](#) = rospy.get_param('~Topics/CSV_End')
- [plot.yaml](#) = rospy.get_param('~YAML')
- [plot.plt](#) = Plots(os.path.join(path,sub),topic,yaml)

12.52.1 Detailed Description

Node file for plotting test results.

This node requires the following parameters to run.

Parameters

<code>~Dir</code>	The directory where the CSV files are saved.
<code>/Subfolder</code>	The subfolder where the CSV file is located.
<code>~Topics/CSV_End</code>	Topic used to communicate when the CSV file is complete and graphics can be generated.
<code>~YAML</code>	Topic used to specify which YAML file corresponds to the running controller and to store the gain values by copying this file.

Definition in file [plot.py](#).

12.53 plot.py

[Go to the documentation of this file.](#)

```

00001 #!/usr/bin/env python
00002 import rospy
00003 from std_msgs.msg import Empty
00004 import os
00005 import csv
00006 import time
00007 import pandas as pd
00008 import matplotlib
00009 matplotlib.use('Agg')
00010 #import matplotlib.pyplot as plt
00011 #import sys
00012
00013
00014
00015
00016
00017
00018
00019
00020
00021
00022 class Plots:
00023     def __init__(self, path, topic, yaml):
00024         self.path = path
00025         self.yaml = yaml
00026         rospy.Subscriber(topic, Empty, self.Plot)
00027         rospy.spin()
00028
00029
00030     def Plot(self, data):
00031         data = pd.read_csv(os.path.join(self.path, 'data.csv'))
00032         print(os.path.join(self.path, 'data.csv'))
00033         col={'x_ref': r'$x_r$', 'y_ref': r'$y_r$', 'z_ref': r'$z_r$', 'yaw_ref': r'$\psi_r$',
00034             'x': r'$x_d$', 'y': r'$y_d$', 'z': r'$z_d$', 'yaw': r'$\psi_d$',
00035             'v_x': r'$v_x$', 'v_y': r'$v_y$', 'v_z': r'$v_z$', 'v_yaw': r'$v_\psi$'}
00036         data.rename(columns=col, inplace=True)
00037         drone_df = data.loc[:, ['Time', r'$x_d$', r'$y_d$', r'$z_d$', r'$\psi_d$']]
00038         reference_df = data.loc[:, ['Time', r'$x_r$', r'$y_r$', r'$z_r$', r'$\psi_r$']]
00039         cmd_vel_df = data.loc[:, ['Time', r'$v_x$', r'$v_y$', r'$v_z$', r'$v_\psi$']]
00040
00041         minV = drone_df['Time'].min()
00042         maxV = drone_df['Time'].max()
00043
00044         ax = drone_df.plot(x=0, grid=True, title='Drone')
00045         ax.set_xlim(minV, maxV)
00046         ax.get_figure().savefig(os.path.join(self.path, 'drone.png'))
00047         ax = reference_df.plot(x=0, grid=True, title='Reference')
00048         ax.set_xlim(minV, maxV)
00049         ax.get_figure().savefig(os.path.join(self.path, 'reference.png'))
00050         ax = cmd_vel_df.plot(x=0, grid=True, title='CMD_Vel')
00051         ax.set_xlim(minV, maxV)
00052         ax.get_figure().savefig(os.path.join(self.path, 'cmd_vel.png'))
00053
00054         for ax in [r'$x', r'$y', r'$z', r'$\psi$']:
00055             ax_orig = r'$' + ax + r'$'
00056             ax_drone = r'$' + ax + r'_d$'
00057             ax_ref = r'$' + ax + r'_r$'

```

```

00058         df = data.loc[:,['Time',ax_drone, ax_ref]]
00059         name = ax + '.png'
00060         fn = os.path.join(self.path,name)
00061         ax = df.plot(x=0,grid=True,title=ax_orig)
00062         ax.set_xlim(minV,maxV)
00063         ax.get_figure().savefig(fn)
00064
00065         os.system("cp $(rospack find bebop_controller)/resource/{ } {}".format(self.yaml, self.path))
00066         os.system("cp $(rospack find bebop_controller)/resource/max_speed.yaml {}".format(self.path))
00067         os.system("cp $(rospack find bebop_controller)/resource/safe_zone.yaml {}".format(self.path))
00068
00069         rospy.signal_shutdown("")
00070
00071 if __name__ == '__main__':
00072     rospy.init_node('plot')
00073     path = rospy.get_param('~Dir')
00074     sub = rospy.get_param('/Subfolder')
00075     topic = rospy.get_param('~Topics/CSV_End')
00076     yaml = rospy.get_param('~YAML')
00077     sub = sub[:-1]
00078     plt = Plots(os.path.join(path,sub),topic,yaml)
00079
00080

```

12.54 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/library/base_controller.cpp File Reference

Library file for the base class of controllers.

```
#include "bebop_controller/base_controller.h"
```

Namespaces

- namespace `bebop_controller`

Namespace containing all the classes and functions of the Bebop Controller.

12.54.1 Detailed Description

Library file for the base class of controllers.

Definition in file `base_controller.cpp`.

12.55 base_controller.cpp

[Go to the documentation of this file.](#)

```

00001
00002
00003
00004 #include "bebop_controller/base_controller.h"
00005
00006 namespace bebop_controller {
00007
00008     BaseController::BaseController():
00009         waypointHasBeenPublished_(false),
00010         takeoff(false),
00011         controller_active_(false),
00012         disable_commands(true),
00013         stop(false) {
00014         ROS_INFO_ONCE("Started position controller");
00015         ros::NodeHandle nh("~");
00016

```

```

00017     std::string bebop_command_trajectory, bebop_pose, bebop_cmd_vel, bebop_takeoff, bebop_land,
00018     csv_end;
00019     GetRosParameter(nh, std::string("Topics/Command_Trajectory"),
00020                     std::string(BEBOP_COMMAND_TRAJECTORY), &bebop_command_trajectory);
00021     GetRosParameter(nh, std::string("Topics/Pose"),
00022                     std::string(BEBOP_POSE), &bebop_pose);
00023     GetRosParameter(nh, std::string("Topics/CMD_Vel"),
00024                     std::string(BEBOP_CMD_VEL), &bebop_cmd_vel);
00025     GetRosParameter(nh, std::string("Topics/TakeOff"),
00026                     std::string(BEBOP_TAKEOFF), &bebop_takeoff);
00027     GetRosParameter(nh, std::string("Topics/Land"),
00028                     std::string(BEBOP_LAND), &bebop_land);
00029     GetRosParameter(nh, std::string("Topics/CSV_End"),
00030                     std::string(CSV_END), &csv_end);
00031     GetRosParameter(nh, std::string("Disable_Commands"),
00032                     DISABLE_COMMANDS, &disable_commands);
00033     GetRosParameter(nh, std::string("Safe_Zone/X"),
00034                     SAFE_ZONE_X, &safe_zone.x);
00035     GetRosParameter(nh, std::string("Safe_Zone/Y"),
00036                     SAFE_ZONE_Y, &safe_zone.y);
00037     GetRosParameter(nh, std::string("Safe_Zone/Z"),
00038                     SAFE_ZONE_Z, &safe_zone.z);
00039     GetRosParameter(nh, std::string("Max_Speed/X"),
00040                     LIMIT_X, &max_speed.x);
00041     GetRosParameter(nh, std::string("Max_Speed/Y"),
00042                     LIMIT_Y, &max_speed.y);
00043     GetRosParameter(nh, std::string("Max_Speed/Z"),
00044                     LIMIT_Z, &max_speed.z);
00045     GetRosParameter(nh, std::string("Max_Speed/Yaw"),
00046                     LIMIT_YAW, &max_speed.yaw);
00047
00048     cmd_multi_dof_joint_trajectory_sub_ = nh.subscribe(bebop_command_trajectory, 1,
00049                                                         &BaseController::MultiDOFJointTrajectory_CB,
00050                                                         this);
00051     odom_sub_ = nh.subscribe(bebop_pose, 30, &BaseController::Odometry_CB, this);
00052     end_sub_ = nh.subscribe(csv_end, 1, &BaseController::Stop_CB, this);
00053     motor_velocity_reference_pub_ = nh.advertise<geometry_msgs::Twist>(bebop_cmd_vel, 1);
00054     takeoff_pub_ = nh.advertise<std_msgs::Empty>(bebop_takeoff, 1);
00055     land_pub_ = nh.advertise<std_msgs::Empty>(bebop_land, 1);
00056     timeOut = nh.createTimer(ros::Duration(1.0), &BaseController::TimeOut_CB, this);
00057
00058     last_state.position.x = 0;
00059     last_state.position.y = 0;
00060     last_state.position.z = 0;
00061     last_state.orientation.x = 0;
00062     last_state.orientation.y = 0;
00063     last_state.orientation.z = 0;
00064
00065     ~BaseController() {
00066         Stop(true);
00067     }
00068
00069     void BaseController::MultiDOFJointTrajectory_CB(const
00070     trajectory_msgs::MultiDOFJointTrajectoryConstPtr& msg) {
00071         const size_t n_commands = msg->points.size();
00072         if (n_commands < 1) {
00073             ROS_WARN_STREAM("Got MultiDOFJointTrajectory message, but message has no points.");
00074             return;
00075         }
00076         mav_msgs::eigenTrajectoryPointFromMsg(msg->points.front(), &command_trajectory_);
00077         waypointHasBeenPublished_ = true;
00078         ROS_INFO_ONCE("Got first MultiDOFJointTrajectory message.");
00079     }
00080
00081     void BaseController::Stop_CB(const std_msgs::Empty::ConstPtr& empty_msg) {
00082         Stop(false);
00083     }
00084
00085     void BaseController::Odometry_CB(const geometry_msgs::PoseStamped& pose_msg) {
00086         ROS_INFO_ONCE("Controller got first pose message.");
00087         timeOut.stop();
00088         timeOut.start();
00089         if (waypointHasBeenPublished_) {
00090             if (!takeoff) {
00091                 TakeOff();
00092             }
00093             if (!controller_active_) {
00094                 lastTime = ros::Time::now();
00095             }
00096             const Eigen::Vector3d position = Eigen::Vector3d(pose_msg.pose.position.x,
00097                                                             pose_msg.pose.position.y,
00098                                                             pose_msg.pose.position.z);
00099             const Eigen::Quaterniond orientation = Eigen::Quaterniond(pose_msg.pose.orientation.w,
00100                                                                        pose_msg.pose.orientation.x,
00101                                                                        pose_msg.pose.orientation.y,
00102                                                                        pose_msg.pose.orientation.z);

```

```

00101         const Eigen::Vector3d zeros = Eigen::Vector3d(0,0,0);
00102         mav_msgs::EigenOdometry odometry =
mav_msgs::EigenOdometry(position,orientation,zeros,zeros);
00103         ros::Time now = ros::Time::now();
00104         diff = now.toSec() - lastTime.toSec();
00105         lastTime = now;
00106         SetOdometry(odometry);
00107         EstimateVelocity();
00108         EstimateAcceleration();
00109         if (!CheckSafeZone()) {
00110             Stop(true);
00111         }
00112         geometry_msgs::Twist ref_command_signals;
00113         CalculateCommandVelocities(ref_command_signals);
00114         if (!disable_commands && !stop) {
00115             motor_velocity_reference_pub_.publish(ref_command_signals);
00116         }
00117     }
00118 }
00119
00120 void BaseController::Timeout_CB(const ros::TimerEvent& event) {
00121     if (takeoff){
00122         ROS_INFO("Pose messages have not been received for one second. Landing the drone.");
00123         Stop(true);
00124     }
00125 }
00126
00127 void BaseController::TakeOff() {
00128     takeoff = true;
00129     if (disable_commands || stop) {
00130         return;
00131     }
00132     std_msgs::Empty empty_msg;
00133     takeoff_pub_.publish(empty_msg);
00134 }
00135
00136 void BaseController::Land() {
00137     takeoff = false;
00138     controller_active_ = false;
00139     if (disable_commands) {
00140         return;
00141     }
00142     std_msgs::Empty empty_msg;
00143     land_pub_.publish(empty_msg);
00144 }
00145
00146 void BaseController::SetOdometry(mav_msgs::EigenOdometry& odometry) {
00147     odometry_ = odometry;
00148     controller_active_ = true;
00149     state.position.x = odometry_.position_W[2];
00150     state.position.y = odometry_.position_W[0];
00151     state.position.z = odometry_.position_W[1];
00152     Quaternion2Euler(state.orientation.x, state.orientation.y, state.orientation.z);
00153 }
00154
00155 void BaseController::GetErrors(Vector4& e) {
00156     e.x = state.position.x - command_trajectory_.position_W[0];
00157     e.y = state.position.y - command_trajectory_.position_W[1];
00158     e.z = state.position.z - command_trajectory_.position_W[2];
00159     e.yaw = state.orientation.z - yawFromQuaternion(command_trajectory_.orientation_W_B);
00160     while (std::abs(e.yaw) > M_PI) {
00161         if (e.yaw > 0) {
00162             e.yaw -= 2*M_PI;
00163         }
00164         else {
00165             e.yaw += 2*M_PI;
00166         }
00167     }
00168 }
00169
00170 void BaseController::GetVelocityErrors(Vector4& dot_e) {
00171     dot_e.x = state.velocity.x - command_trajectory_.velocity_W[0];
00172     dot_e.y = state.velocity.y - command_trajectory_.velocity_W[1];
00173     dot_e.z = state.velocity.z - command_trajectory_.velocity_W[2];
00174     dot_e.yaw = state.angular_velocity.z - command_trajectory_.angular_velocity_W[2];
00175 }
00176
00177 void BaseController::Quaternion2Euler(double& roll, double& pitch, double& yaw) const {
00178     tf::Quaternion q(odometry_.orientation_W_B.z(), odometry_.orientation_W_B.x(),
00179                     odometry_.orientation_W_B.y(), odometry_.orientation_W_B.w());
00180     tf::Matrix3x3 m(q);
00181     m.getRPY(roll, pitch, yaw);
00182 }
00183
00184 void BaseController::CalculateCommandVelocities(geometry_msgs::Twist& ref_command_signals) {
00185     ref_command_signals.linear.x = 0.0;
00186     ref_command_signals.linear.y = 0.0;

```



```
00187         ref_command_signals.linear.z = 0.0;
00188         ref_command_signals.angular.z = 0.0;
00189     }
00190
00191     void BaseController::EstimateVelocity() {
00192         state.velocity.x = (state.position.x - last_state.position.x)/diff;
00193         state.velocity.y = (state.position.y - last_state.position.y)/diff;
00194         state.velocity.z = (state.position.z - last_state.position.z)/diff;
00195         state.angular_velocity.x = (state.orientation.x - last_state.orientation.x)/diff;
00196         state.angular_velocity.y = (state.orientation.y - last_state.orientation.y)/diff;
00197         state.angular_velocity.z = (state.orientation.z - last_state.orientation.z)/diff;
00198         last_state.position = state.position;
00199         last_state.orientation = state.orientation;
00200     }
00201
00202     void BaseController::EstimateAcceleration() {
00203         state.acceleration.x = (state.velocity.x - last_state.velocity.x)/diff;
00204         state.acceleration.y = (state.velocity.y - last_state.velocity.y)/diff;
00205         state.acceleration.z = (state.velocity.z - last_state.velocity.z)/diff;
00206         state.angular_acceleration.x = (state.angular_velocity.x -
last_state.angular_velocity.x)/diff;
00207         state.angular_acceleration.y = (state.angular_velocity.y -
last_state.angular_velocity.y)/diff;
00208         state.angular_acceleration.z = (state.angular_velocity.z -
last_state.angular_velocity.z)/diff;
00209         last_state.velocity = state.velocity;
00210         last_state.angular_velocity = state.angular_velocity;
00211     }
00212
00213     void BaseController::Stop(bool failsafe) {
00214         if (failsafe) {
00215             ROS_INFO_ONCE("Failsafe mode");
00216         }
00217         stop = true;
00218         geometry_msgs::Twist ref_command_signals;
00219         ref_command_signals.linear.x = 0.0;
00220         ref_command_signals.linear.y = 0.0;
00221         ref_command_signals.linear.z = 0.0;
00222         ref_command_signals.angular.z = 0.0;
00223         motor_velocity_reference_pub_.publish(ref_command_signals);
00224         Land();
00225     }
00226
00227     bool BaseController::CheckSafeZone() {
00228         return (std::abs(state.position.x) < safe_zone.x)
            && (std::abs(state.position.y) < safe_zone.y)
            && (std::abs(state.position.z) < safe_zone.z);
00231     }
00232
00233     void BaseController::CalculateLeashLength(Vector4& e, Vector4& P) {
00234         Vector3 Ppos, vel, accel;
00235         Ppos.x = P.x*sqrt(std::abs(e.x));
00236         Ppos.y = P.y*sqrt(std::abs(e.y));
00237         vel.x = std::abs(max(state.velocity.x,MIN_VEL));
00238         vel.y = std::abs(max(state.velocity.y,MIN_VEL));
00239         accel.x = std::abs(max(state.acceleration.x,MIN_ACCEL));
00240         accel.y = std::abs(max(state.acceleration.y,MIN_ACCEL));
00241         leash_length.x = accel.x/(2*pow(Ppos.x,2)) + pow(vel.x,2)/(2*accel.x);
00242         leash_length.y = accel.y/(2*pow(Ppos.y,2)) + pow(vel.y,2)/(2*accel.y);
00243     }
00244
00245     void BaseController::LimitPositionErrors(Vector4& e) {
00246         e.x = clamp(e.x,leash_length.x);
00247         e.y = clamp(e.y,leash_length.y);
00248     }
00249
00250 }
```

12.56 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/library/waypoint.cpp File Reference

Library file for the base class of the waypoint generator.

```
#include "bebop_controller/waypoint.h"
```

Namespaces

- namespace [bebop_controller](#)

Namespace containing all the classes and functions of the Bebop Controller.

12.56.1 Detailed Description

Library file for the base class of the waypoint generator.

Definition in file [waypoint.cpp](#).

12.57 waypoint.cpp

[Go to the documentation of this file.](#)

```

00001
00002
00003
00004 #include "bebop_controller/waypoint.h"
00005
00006 namespace bebop_controller {
00007
00008     Waypoint::Waypoint(WaypointParameters param) :
00009         wp_params_(param),
00010         status(BeforeTrajectory) {
00011         setpoint_pub_ =
00012             nh.advertise<trajectory_msgs::MultiDOFJointTrajectory>(wp_params_.topic_command_trajectory, 1);
00013         Begin_ = nh.advertise<std_msgs::Empty>(wp_params_.topic_csv_begin, 1);
00014         End_ = nh.advertise<std_msgs::Empty>(wp_params_.topic_csv_end, 1);
00015
00016         double TotalTime = wp_params_.TimeBeforeTrajectory + wp_params_.TrajectoryTime +
00017             wp_params_.MarginTime;
00018         _timer1 = _n1.createTimer(ros::Duration(wp_params_.DiffTime), &Waypoint::Trajectory_CB, this,
00019             false, true);
00020         _timer2 = _n2.createTimer(ros::Duration(wp_params_.TimeBeforeTrajectory), &Waypoint::Start_CB,
00021             this, true, true);
00022         _timer3 = _n3.createTimer(ros::Duration(TotalTime), &Waypoint::Stop_CB, this, false, true);
00023     }
00024
00025     Waypoint::~Waypoint() {}
00026
00027     void Waypoint::Start_CB(const ros::TimerEvent& event) {
00028         status = Trajectory;
00029         std_msgs::Empty empty_;
00030         for (int i = 0; i < 50; i++){
00031             Begin_.publish(empty_);
00032             std::this_thread::sleep_for(std::chrono::microseconds(100));
00033         }
00034         Initial_Time = ros::Time::now();
00035     }
00036
00037     void Waypoint::Stop_CB(const ros::TimerEvent& event) {
00038         std_msgs::Empty empty_;
00039         for (int i = 0; i < 50; i++){
00040             End_.publish(empty_);
00041             std::this_thread::sleep_for(std::chrono::microseconds(100));
00042         }
00043         ros::shutdown();
00044     }
00045
00046     void Waypoint::Trajectory_CB(const ros::TimerEvent& event) {}
00047 }

```

12.58 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/citc_controller_angles.cpp File Reference

Node file for CITC Controller using reference angles.

```
#include "citc_controller_angles.h"
```

Namespaces

- namespace [bebop_controller](#)

Namespace containing all the classes and functions of the Bebop Controller.

Functions

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

12.58.1 Detailed Description

Node file for CITC Controller using reference angles.

Definition in file [citic_controller_angles.cpp](#).

12.58.2 Function Documentation

12.58.2.1 main()

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

Definition at line 128 of file [citic_controller_angles.cpp](#).

12.59 citc_controller_angles.cpp

[Go to the documentation of this file.](#)

```
00001
00002
00003
00004 #include "citic_controller_angles.h"
00005
00006 namespace bebop_controller {
00007
00008     CITCController::CITCController() {
00009         ros::NodeHandle pnh("~");
00010         int_zeta.x = int_zeta.y = int_zeta.z = int_zeta.yaw = 0;
00011         int_eta.x = int_eta.y = int_eta.z = int_eta.yaw = 0;
00012         last_dot_e.x = last_dot_e.y = last_dot_e.z = last_dot_e.yaw = 0;
00013         u_z = 0;
00014         GetRosParameter(pnh, "Gains/K1x", K1xDefaultValue, &K1.x);
00015         GetRosParameter(pnh, "Gains/K1y", K1yDefaultValue, &K1.y);
00016         GetRosParameter(pnh, "Gains/K1z", K1zDefaultValue, &K1.z);
00017         GetRosParameter(pnh, "Gains/K1yaw", K1yawDefaultValue, &K1.yaw);
00018         GetRosParameter(pnh, "Gains/K2x", K2xDefaultValue, &K2.x);
00019         GetRosParameter(pnh, "Gains/K2y", K2yDefaultValue, &K2.y);
00020         GetRosParameter(pnh, "Gains/K2z", K2zDefaultValue, &K2.z);
00021         GetRosParameter(pnh, "Gains/K2yaw", K2yawDefaultValue, &K2.yaw);
00022         GetRosParameter(pnh, "Gains/K3x", K3xDefaultValue, &K3.x);
00023         GetRosParameter(pnh, "Gains/K3y", K3yDefaultValue, &K3.y);
00024         GetRosParameter(pnh, "Gains/K3z", K3zDefaultValue, &K3.z);
00025         GetRosParameter(pnh, "Gains/K3yaw", K3yawDefaultValue, &K3.yaw);
00026         GetRosParameter(pnh, "Gains/K4x", K4xDefaultValue, &K4.x);
00027         GetRosParameter(pnh, "Gains/K4y", K4yDefaultValue, &K4.y);
00028         GetRosParameter(pnh, "Gains/K4z", K4zDefaultValue, &K4.z);
00029         GetRosParameter(pnh, "Gains/K4yaw", K4yawDefaultValue, &K4.yaw);
```

```

00030     GetRosParameter(pnh, "Reference_Gains/X", RGxDefaultValue, &RG.x);
00031     GetRosParameter(pnh, "Reference_Gains/Y", RGyDefaultValue, &RG.y);
00032     GetRosParameter(pnh, "Reference_Gains/Z", RGzDefaultValue, &RG.z);
00033     GetRosParameter(pnh, "Reference_Gains/Yaw", RgyawDefaultValue, &RG.yaw);
00034     GetRosParameter(pnh, "Lambda/X", LambdaX, &lambda.x);
00035     GetRosParameter(pnh, "Lambda/Y", LambdaY, &lambda.y);
00036     GetRosParameter(pnh, "Lambda/Z", LambdaZ, &lambda.z);
00037     GetRosParameter(pnh, "Sigma", Sigma, &sigma);
00038     GetRosParameter(pnh, std::string("Normalize/Max_Tilt_Angle"),
00039                     MAX_TILT_ANGLE, &norm.angle);
00040     GetRosParameter(pnh, std::string("Normalize/Max_Vertical_Speed"),
00041                     MAX_VERTICAL_SPEED, &norm.vertical);
00042     GetRosParameter(pnh, std::string("Normalize/Max_Rotation_Speed"),
00043                     MAX_ROTATION_SPEED, &norm.rotation);
00044     GetRosParameter(pnh, "Mass", MASS, &mass);
00045 }
00046
00047 void CITCController::CalculateCommandVelocities(geometry_msgs::Twist& ref_command_signals){
00048     if (!controller_active){
00049         ref_command_signals.linear.x = 0.0;
00050         ref_command_signals.linear.y = 0.0;
00051         ref_command_signals.linear.z = 0.0;
00052         ref_command_signals.angular.z = 0.0;
00053         return;
00054     }
00055     Vector4 a, v_nom, v_stc, e, dot_e, ddot_e;
00056     GetErrors(e);
00057     GetVelocityErrors(dot_e);
00058     GetAccelerationErrors(ddot_e, dot_e);
00059
00060     v_nom.x = -K1.x*pow(std::abs(dot_e.x),sigma)*sgn(dot_e.x) -
00061     K2.x*pow(std::abs(e.x),((sigma)/(2-sigma)))*sgn(e.x);
00062     v_nom.y = -K1.y*pow(std::abs(dot_e.y),sigma)*sgn(dot_e.y) -
00063     K2.y*pow(std::abs(e.y),((sigma)/(2-sigma)))*sgn(e.y);
00064     v_nom.z = -K1.z*pow(std::abs(dot_e.z),sigma)*sgn(dot_e.z) - K2.z
00065     *pow(std::abs(e.z),((sigma)/(2-sigma)))*sgn(e.z);
00066     v_nom.yaw = -K1.yaw*pow(std::abs(dot_e.yaw),sigma)*sgn(dot_e.yaw) - K2.yaw
00067     *pow(std::abs(e.yaw),((sigma)/(2-sigma)))*sgn(e.yaw);
00068
00069     IntegrateZetaAndEta(v_nom, ddot_e);
00070
00071     v_stc.x = -K3.x*std::abs(int_zeta.x)*sgn(int_zeta.x) + int_eta.x;
00072     v_stc.y = -K3.y*std::abs(int_zeta.y)*sgn(int_zeta.y) + int_eta.y;
00073     v_stc.z = -K3.z *std::abs(int_zeta.z)*sgn(int_zeta.z) + int_eta.z;
00074     v_stc.yaw = -K3.yaw *std::abs(int_zeta.yaw)*sgn(int_zeta.yaw) + int_eta.yaw;
00075
00076     a.x = clamp(v_nom.x + v_stc.x + RG.x*command_trajectory_.acceleration_W[0],
00077                 mass*(lambda.x-std::abs(command_trajectory_.acceleration_W[0])));
00078     a.y = clamp(v_nom.y + v_stc.y + RG.y*command_trajectory_.acceleration_W[1],
00079                 mass*(lambda.y-std::abs(command_trajectory_.acceleration_W[1])));
00080     a.z = clamp(v_nom.z + v_stc.z + RG.z*command_trajectory_.acceleration_W[2],
00081                 mass*(lambda.z-std::abs(command_trajectory_.acceleration_W[2])));
00082     a.yaw = v_nom.yaw + v_stc.yaw;
00083
00084     double u_Terr, u_T, phi_r, theta_r, psi;
00085     u_Terr = a.z + mass*GRAVITY;
00086     u_T = sqrt(pow(a.x,2) + pow(a.y,2) + pow(u_Terr,2));
00087     psi = state.orientation.z;
00088     theta_r = atan((a.x * cos(psi)) + (a.y * sin(psi)))/u_Terr);
00089     phi_r = atan(cos(theta_r)*((a.x * sin(psi)) - (a.y * cos(psi)))/(u_Terr));
00090
00091     u_z = clamp(u_z + diff*a.z,max_speed.z*norm.vertical);
00092
00093     ref_command_signals.linear.x = clamp(theta_r*RAD_2_DEG/norm.angle,max_speed.x);
00094     ref_command_signals.linear.y = clamp(-phi_r*RAD_2_DEG/norm.angle,max_speed.y);
00095     ref_command_signals.linear.z = clamp(u_z/norm.vertical,max_speed.z);
00096     ref_command_signals.angular.z = clamp(a.yaw*RAD_2_DEG/norm.rotation,max_speed.yaw);
00097 }
00098
00099 void CITCController::GetAccelerationErrors(Vector4& ddot_e, Vector4& dot_e) {
00100     ddot_e.x = (dot_e.x - last_dot_e.x)/diff;
00101     ddot_e.y = (dot_e.y - last_dot_e.y)/diff;
00102     ddot_e.z = (dot_e.z - last_dot_e.z)/diff;
00103     ddot_e.yaw = (dot_e.yaw - last_dot_e.yaw)/diff;
00104     last_dot_e.x = dot_e.x;
00105     last_dot_e.y = dot_e.y;
00106     last_dot_e.z = dot_e.z;
00107     last_dot_e.yaw = dot_e.yaw;
00108 }
00109
00110 void CITCController::IntegrateZetaAndEta(Vector4& v_nom, Vector4& ddot_e) {
00111     Vector4 zeta, eta;
00112     zeta.x = ddot_e.x - v_nom.x;
00113     zeta.y = ddot_e.y - v_nom.y;
00114     zeta.z = ddot_e.z - v_nom.z;
00115     zeta.yaw = ddot_e.yaw - v_nom.yaw;
00116     int_zeta.x += zeta.x*diff;

```

```

00113         int_zeta.y += zeta.y*diff;
00114         int_zeta.z += zeta.z*diff;
00115         int_zeta.yaw += zeta.yaw*diff;
00116         eta.x = -K4.x*sgn(int_zeta.x);
00117         eta.y = -K4.y*sgn(int_zeta.y);
00118         eta.z = -K4.z*sgn(int_zeta.z);
00119         eta.yaw = -K4.yaw*sgn(int_zeta.yaw);
00120         int_eta.x += eta.x*diff;
00121         int_eta.y += eta.y*diff;
00122         int_eta.z += eta.z*diff;
00123         int_eta.yaw += eta.yaw*diff;
00124     }
00125 }
00126 }
00127
00128 int main(int argc, char** argv){
00129     ros::init(argc, argv, "citc_controller_node");
00130     ros::NodeHandle nh;
00131     bebop_controller::CITCController citc_controller;
00132     ros::spin();
00133     return 0;
00134 }

```

12.60 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_↵ controller/src/nodes/citc_controller_angles.h File Reference

Header file for CITC Controller using reference angles.

```
#include "bebop_controller/base_controller.h"
```

Classes

- struct [bebop_controller::Normalize](#)
- class [bebop_controller::CITCController](#)

Namespaces

- namespace [bebop_controller](#)
Namespace containing all the classes and functions of the Bebop Controller.

Macros

- `#define` [K1xDefaultValue](#) 1.0
- `#define` [K1yDefaultValue](#) 1.0
- `#define` [K1zDefaultValue](#) 1.0
- `#define` [K1yawDefaultValue](#) 1.0
- `#define` [K2xDefaultValue](#) 1.0
- `#define` [K2yDefaultValue](#) 1.0
- `#define` [K2zDefaultValue](#) 1.0
- `#define` [K2yawDefaultValue](#) 1.0
- `#define` [K3xDefaultValue](#) 1.0
- `#define` [K3yDefaultValue](#) 1.0
- `#define` [K3zDefaultValue](#) 1.0
- `#define` [K3yawDefaultValue](#) 1.0
- `#define` [K4xDefaultValue](#) 1.0
- `#define` [K4yDefaultValue](#) 1.0

- `#define K4zDefaultValue 1.0`
- `#define K4yawDefaultValue 1.0`
- `#define RGxDefaultValue 0.0`
- `#define RGyDefaultValue 0.0`
- `#define RGzDefaultValue 0.0`
- `#define RGyawDefaultValue 0.0`
- `#define LambdaX 6.295`
- `#define LambdaY 6.295`
- `#define LambdaZ 4.6697`
- `#define Sigma 0.8`
- `#define MAX_TILT_ANGLE 20.0`
- `#define MAX_VERTICAL_SPEED 1.0`
- `#define MAX_ROTATION_SPEED 100.0`
- `#define MASS 0.5`

12.60.1 Detailed Description

Header file for CITC Controller using reference angles.

Definition in file [citc_controller_angles.h](#).

12.60.2 Macro Definition Documentation

12.60.2.1 K1xDefaultValue

```
#define K1xDefaultValue 1.0
```

Definition at line 6 of file [citc_controller_angles.h](#).

12.60.2.2 K1yawDefaultValue

```
#define K1yawDefaultValue 1.0
```

Definition at line 9 of file [citc_controller_angles.h](#).

12.60.2.3 K1yDefaultValue

```
#define K1yDefaultValue 1.0
```

Definition at line 7 of file [citc_controller_angles.h](#).

12.60.2.4 K1zDefaultValue

```
#define K1zDefaultValue 1.0
```

Definition at line 8 of file [citc_controller_angles.h](#).

12.60.2.5 K2xDefaultValue

```
#define K2xDefaultValue 1.0
```

Definition at line 11 of file [citc_controller_angles.h](#).

12.60.2.6 K2yawDefaultValue

```
#define K2yawDefaultValue 1.0
```

Definition at line 14 of file [citc_controller_angles.h](#).

12.60.2.7 K2yDefaultValue

```
#define K2yDefaultValue 1.0
```

Definition at line 12 of file [citc_controller_angles.h](#).

12.60.2.8 K2zDefaultValue

```
#define K2zDefaultValue 1.0
```

Definition at line 13 of file [citc_controller_angles.h](#).

12.60.2.9 K3xDefaultValue

```
#define K3xDefaultValue 1.0
```

Definition at line 16 of file [citc_controller_angles.h](#).

12.60.2.10 K3yawDefaultValue

```
#define K3yawDefaultValue 1.0
```

Definition at line 19 of file [citic_controller_angles.h](#).

12.60.2.11 K3yDefaultValue

```
#define K3yDefaultValue 1.0
```

Definition at line 17 of file [citic_controller_angles.h](#).

12.60.2.12 K3zDefaultValue

```
#define K3zDefaultValue 1.0
```

Definition at line 18 of file [citic_controller_angles.h](#).

12.60.2.13 K4xDefaultValue

```
#define K4xDefaultValue 1.0
```

Definition at line 21 of file [citic_controller_angles.h](#).

12.60.2.14 K4yawDefaultValue

```
#define K4yawDefaultValue 1.0
```

Definition at line 24 of file [citic_controller_angles.h](#).

12.60.2.15 K4yDefaultValue

```
#define K4yDefaultValue 1.0
```

Definition at line 22 of file [citic_controller_angles.h](#).

12.60.2.16 K4zDefaultValue

```
#define K4zDefaultValue 1.0
```

Definition at line 23 of file [citc_controller_angles.h](#).

12.60.2.17 LambdaX

```
#define LambdaX 6.295
```

Definition at line 31 of file [citc_controller_angles.h](#).

12.60.2.18 LambdaY

```
#define LambdaY 6.295
```

Definition at line 32 of file [citc_controller_angles.h](#).

12.60.2.19 LambdaZ

```
#define LambdaZ 4.6697
```

Definition at line 33 of file [citc_controller_angles.h](#).

12.60.2.20 MASS

```
#define MASS 0.5
```

Definition at line 40 of file [citc_controller_angles.h](#).

12.60.2.21 MAX_ROTATION_SPEED

```
#define MAX_ROTATION_SPEED 100.0
```

Definition at line 38 of file [citc_controller_angles.h](#).

12.60.2.22 MAX_TILT_ANGLE

```
#define MAX_TILT_ANGLE 20.0
```

Definition at line 36 of file [citic_controller_angles.h](#).

12.60.2.23 MAX_VERTICAL_SPEED

```
#define MAX_VERTICAL_SPEED 1.0
```

Definition at line 37 of file [citic_controller_angles.h](#).

12.60.2.24 RGxDefaultValue

```
#define RGxDefaultValue 0.0
```

Definition at line 26 of file [citic_controller_angles.h](#).

12.60.2.25 RGyawDefaultValue

```
#define RGyawDefaultValue 0.0
```

Definition at line 29 of file [citic_controller_angles.h](#).

12.60.2.26 RGyDefaultValue

```
#define RGyDefaultValue 0.0
```

Definition at line 27 of file [citic_controller_angles.h](#).

12.60.2.27 RGzDefaultValue

```
#define RGzDefaultValue 0.0
```

Definition at line 28 of file [citic_controller_angles.h](#).

12.60.2.28 Sigma

```
#define Sigma 0.8
```

Definition at line 34 of file [citic_controller_angles.h](#).

12.61 citc_controller_angles.h

[Go to the documentation of this file.](#)

```
00001
00003
00004 #include "bebop_controller/base_controller.h"
00005
00006 #define K1xDefaultValue 1.0
00007 #define K1yDefaultValue 1.0
00008 #define K1zDefaultValue 1.0
00009 #define K1yawDefaultValue 1.0
00010
00011 #define K2xDefaultValue 1.0
00012 #define K2yDefaultValue 1.0
00013 #define K2zDefaultValue 1.0
00014 #define K2yawDefaultValue 1.0
00015
00016 #define K3xDefaultValue 1.0
00017 #define K3yDefaultValue 1.0
00018 #define K3zDefaultValue 1.0
00019 #define K3yawDefaultValue 1.0
00020
00021 #define K4xDefaultValue 1.0
00022 #define K4yDefaultValue 1.0
00023 #define K4zDefaultValue 1.0
00024 #define K4yawDefaultValue 1.0
00025
00026 #define RGxDefaultValue 0.0
00027 #define RGyDefaultValue 0.0
00028 #define RGzDefaultValue 0.0
00029 #define RGyawDefaultValue 0.0
00030
00031 #define LambdaX 6.295
00032 #define LambdaY 6.295
00033 #define LambdaZ 4.6697
00034 #define Sigma 0.8
00035
00036 #define MAX_TILT_ANGLE 20.0
00037 #define MAX_VERTICAL_SPEED 1.0
00038 #define MAX_ROTATION_SPEED 100.0
00039
00040 #define MASS 0.5
00041
00042 namespace bebop_controller {
00043
00044     struct Normalize {
00045         double angle;
00046         double vertical;
00047         double rotation;
00048     };
00049
00050     class CITCController : public BaseController {
00051     public:
00052         CITCController();
00053
00054     private:
00055         Vector4 K1, K2, K3, K4, RG;
00056         Vector4 int_zeta, int_eta, last_dot_e;
00057         Vector3 lambda;
00058         Normalize norm;
00059         double mass, u_z, sigma;
00060         void CalculateCommandVelocities(geometry_msgs::Twist& ref_command_signals) override;
00061         void GetAccelerationErrors(Vector4& ddot_e, Vector4& dot_e);
00062         void IntegrateZetaAndEta(Vector4& v_nom, Vector4& ddot_e);
00063     };
00064
00065 }
```

12.62 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/citc_controller_twist.cpp File Reference

Node file for CITC Controller using velocity commands.

```
#include "citc_controller_twist.h"
```

Namespaces

- namespace [bebop_controller](#)
Namespace containing all the classes and functions of the Bebop Controller.

Functions

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

12.62.1 Detailed Description

Node file for CITC Controller using velocity commands.

Definition in file [citc_controller_twist.cpp](#).

12.62.2 Function Documentation

12.62.2.1 main()

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

Definition at line [124](#) of file [citc_controller_twist.cpp](#).

12.63 citc_controller_twist.cpp

[Go to the documentation of this file.](#)

```

00001
00002
00003
00004 #include "citic_controller_twist.h"
00005
00006 namespace bebop_controller {
00007
00008     CITCController::CITCController() {
00009         ros::NodeHandle pnh("~");
00010         int_zeta.x = int_zeta.y = int_zeta.z = int_zeta.yaw = 0;
00011         int_eta.x = int_eta.y = int_eta.z = int_eta.yaw = 0;
00012         last_dot_e.x = last_dot_e.y = last_dot_e.z = last_dot_e.yaw = 0;
00013         u.x = u.y = u.z = u.yaw = 0;
00014         GetRosParameter(pnh, "Gains/K1x", K1xDefaultValue, &K1.x);
00015         GetRosParameter(pnh, "Gains/K1y", K1yDefaultValue, &K1.y);
00016         GetRosParameter(pnh, "Gains/K1z", K1zDefaultValue, &K1.z);
00017         GetRosParameter(pnh, "Gains/K1yaw", K1yawDefaultValue, &K1.yaw);
00018         GetRosParameter(pnh, "Gains/K2x", K2xDefaultValue, &K2.x);
00019         GetRosParameter(pnh, "Gains/K2y", K2yDefaultValue, &K2.y);
00020         GetRosParameter(pnh, "Gains/K2z", K2zDefaultValue, &K2.z);
00021         GetRosParameter(pnh, "Gains/K2yaw", K2yawDefaultValue, &K2.yaw);
00022         GetRosParameter(pnh, "Gains/K3x", K3xDefaultValue, &K3.x);
00023         GetRosParameter(pnh, "Gains/K3y", K3yDefaultValue, &K3.y);
00024         GetRosParameter(pnh, "Gains/K3z", K3zDefaultValue, &K3.z);
00025         GetRosParameter(pnh, "Gains/K3yaw", K3yawDefaultValue, &K3.yaw);
00026         GetRosParameter(pnh, "Gains/K4x", K4xDefaultValue, &K4.x);
00027         GetRosParameter(pnh, "Gains/K4y", K4yDefaultValue, &K4.y);
00028         GetRosParameter(pnh, "Gains/K4z", K4zDefaultValue, &K4.z);
00029         GetRosParameter(pnh, "Gains/K4yaw", K4yawDefaultValue, &K4.yaw);
00030         GetRosParameter(pnh, "Reference_Gains/X", RGxDefaultValue, &RG.x);
00031         GetRosParameter(pnh, "Reference_Gains/Y", RGyDefaultValue, &RG.y);
00032         GetRosParameter(pnh, "Reference_Gains/Z", RGzDefaultValue, &RG.z);
00033         GetRosParameter(pnh, "Reference_Gains/Yaw", RGYawDefaultValue, &RG.yaw);
00034         GetRosParameter(pnh, "Lambda/X", LambdaX, &lambda.x);
00035         GetRosParameter(pnh, "Lambda/Y", LambdaY, &lambda.y);
00036         GetRosParameter(pnh, "Lambda/Z", LambdaZ, &lambda.z);
00037         GetRosParameter(pnh, "Sigma", Sigma, &sigma);
00038         GetRosParameter(pnh, std::string("Normalize/Max_Horizontal_Speed"),
00039             MAX_HORIZONTAL_SPEED, &norm.horizontal);
00040         GetRosParameter(pnh, std::string("Normalize/Max_Vertical_Speed"),
00041             MAX_VERTICAL_SPEED, &norm.vertical);
00042         GetRosParameter(pnh, std::string("Normalize/Max_Rotation_Speed"),
00043             MAX_ROTATION_SPEED, &norm.rotation);
00044         GetRosParameter(pnh, "Mass", MASS, &mass);
00045     }
00046
00047     void CITCController::CalculateCommandVelocities(geometry_msgs::Twist& ref_command_signals){
00048         if (!controller_active_){
00049             ref_command_signals.linear.x = 0.0;
00050             ref_command_signals.linear.y = 0.0;
00051             ref_command_signals.linear.z = 0.0;
00052             ref_command_signals.angular.z = 0.0;
00053             return;
00054         }
00055         Vector4 a, v_nom, v_stc, e, dot_e, ddot_e;
00056         GetErrors(e);
00057         GetVelocityErrors(dot_e);
00058         GetAccelerationErrors(ddot_e, dot_e);
00059
00060         v_nom.x = -K1.x*pow(std::abs(dot_e.x), sigma)*sgn(dot_e.x) -
00061             K2.x*pow(std::abs(e.x), ((sigma)/(2-sigma)))*sgn(e.x);
00062         v_nom.y = -K1.y*pow(std::abs(dot_e.y), sigma)*sgn(dot_e.y) -
00063             K2.y*pow(std::abs(e.y), ((sigma)/(2-sigma)))*sgn(e.y);
00064         v_nom.z = -K1.z*pow(std::abs(dot_e.z), sigma)*sgn(dot_e.z) - K2.z
00065             *pow(std::abs(e.z), ((sigma)/(2-sigma)))*sgn(e.z);
00066         v_nom.yaw = -K1.yaw*pow(std::abs(dot_e.yaw), sigma)*sgn(dot_e.yaw) - K2.yaw
00067             *pow(std::abs(e.yaw), ((sigma)/(2-sigma)))*sgn(e.yaw);
00068
00069         IntegrateZetaAndEta(v_nom, ddot_e);
00070
00071         v_stc.x = -K3.x*std::abs(int_zeta.x)*sgn(int_zeta.x) + int_eta.x;
00072         v_stc.y = -K3.y*std::abs(int_zeta.y)*sgn(int_zeta.y) + int_eta.y;
00073         v_stc.z = -K3.z*std::abs(int_zeta.z)*sgn(int_zeta.z) + int_eta.z;
00074         v_stc.yaw = -K3.yaw*std::abs(int_zeta.yaw)*sgn(int_zeta.yaw) + int_eta.yaw;
00075
00076         a.x = clamp(v_nom.x + v_stc.x + RG.x*command_trajectory_.acceleration_W[0],
00077             mass*(lambda.x-std::abs(command_trajectory_.acceleration_W[0])));
00078         a.y = clamp(v_nom.y + v_stc.y + RG.y*command_trajectory_.acceleration_W[1],
00079             mass*(lambda.y-std::abs(command_trajectory_.acceleration_W[1])));
00080         a.z = clamp(v_nom.z + v_stc.z + RG.z*command_trajectory_.acceleration_W[2],
00081             mass*(lambda.z-std::abs(command_trajectory_.acceleration_W[2])));
00082         a.yaw = v_nom.yaw + v_stc.yaw;
00083     }
00084 }

```

```

00080     u.x = clamp(u.x + (diff*a.x)/norm.horizontal,max_speed.x);
00081     u.y = clamp(u.y + (diff*a.y)/norm.horizontal,max_speed.y);
00082     u.z = clamp(u.z + (diff*a.z)/norm.vertical,max_speed.z);
00083     u.yaw = clamp(u.yaw + (diff*a.yaw)*RAD_2_DEG/norm.rotation,max_speed.yaw);
00084
00085     ref_command_signals.linear.x = clamp(cos(state.orientation.z)*u.x -
sin(state.orientation.z)*u.y,max_speed.x);
00086     ref_command_signals.linear.y = clamp(sin(state.orientation.z)*u.x +
cos(state.orientation.z)*u.y,max_speed.y);
00087     ref_command_signals.linear.z = clamp(u.z,max_speed.z);
00088     ref_command_signals.angular.z = clamp(u.yaw,max_speed.yaw);
00089 }
00090
00091 void CITCController::GetAccelerationErrors(Vector4& ddot_e, Vector4& dot_e) {
00092     ddot_e.x = (dot_e.x - last_dot_e.x)/diff;
00093     ddot_e.y = (dot_e.y - last_dot_e.y)/diff;
00094     ddot_e.z = (dot_e.z - last_dot_e.z)/diff;
00095     ddot_e.yaw = (dot_e.yaw - last_dot_e.yaw)/diff;
00096     last_dot_e.x = dot_e.x;
00097     last_dot_e.y = dot_e.y;
00098     last_dot_e.z = dot_e.z;
00099     last_dot_e.yaw = dot_e.yaw;
00100 }
00101
00102 void CITCController::IntegrateZetaAndEta(Vector4& v_nom, Vector4& ddot_e) {
00103     Vector4 zeta, eta;
00104     zeta.x = ddot_e.x - v_nom.x;
00105     zeta.y = ddot_e.y - v_nom.y;
00106     zeta.z = ddot_e.z - v_nom.z;
00107     zeta.yaw = ddot_e.yaw - v_nom.yaw;
00108     int_zeta.x += zeta.x*diff;
00109     int_zeta.y += zeta.y*diff;
00110     int_zeta.z += zeta.z*diff;
00111     int_zeta.yaw += zeta.yaw*diff;
00112     eta.x = -K4.x*sgn(int_zeta.x);
00113     eta.y = -K4.y*sgn(int_zeta.y);
00114     eta.z = -K4.z*sgn(int_zeta.z);
00115     eta.yaw = -K4.yaw*sgn(int_zeta.yaw);
00116     int_eta.x += eta.x*diff;
00117     int_eta.y += eta.y*diff;
00118     int_eta.z += eta.z*diff;
00119     int_eta.yaw += eta.yaw*diff;
00120 }
00121
00122 }
00123
00124 int main(int argc, char** argv){
00125     ros::init(argc, argv, "citc_controller_node");
00126     ros::NodeHandle nh;
00127     bebop_controller::CITCController citc_controller;
00128     ros::spin();
00129     return 0;
00130 }

```

12.64 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/citc_controller_twist.h File Reference

Header file for CITC Controller using velocity commands.

```
#include "bebop_controller/base_controller.h"
```

Classes

- struct [bebop_controller::Normalize](#)
- class [bebop_controller::CITCController](#)

Namespaces

- namespace [bebop_controller](#)

Namespace containing all the classes and functions of the Bebop Controller.

Macros

- `#define K1xDefaultValue 1.0`
- `#define K1yDefaultValue 1.0`
- `#define K1zDefaultValue 1.0`
- `#define K1yawDefaultValue 1.0`
- `#define K2xDefaultValue 1.0`
- `#define K2yDefaultValue 1.0`
- `#define K2zDefaultValue 1.0`
- `#define K2yawDefaultValue 1.0`
- `#define K3xDefaultValue 1.0`
- `#define K3yDefaultValue 1.0`
- `#define K3zDefaultValue 1.0`
- `#define K3yawDefaultValue 1.0`
- `#define K4xDefaultValue 1.0`
- `#define K4yDefaultValue 1.0`
- `#define K4zDefaultValue 1.0`
- `#define K4yawDefaultValue 1.0`
- `#define RGxDefaultValue 0.0`
- `#define RGyDefaultValue 0.0`
- `#define RGzDefaultValue 0.0`
- `#define RGyawDefaultValue 0.0`
- `#define LambdaX 6.295`
- `#define LambdaY 6.295`
- `#define LambdaZ 4.6697`
- `#define Sigma 0.8`
- `#define MAX_HORIZONTAL_SPEED 17.0`
- `#define MAX_VERTICAL_SPEED 1.0`
- `#define MAX_ROTATION_SPEED 100.0`
- `#define MASS 0.5`

12.64.1 Detailed Description

Header file for CITC Controller using velocity commands.

Definition in file [citc_controller_twist.h](#).

12.64.2 Macro Definition Documentation**12.64.2.1 K1xDefaultValue**

```
#define K1xDefaultValue 1.0
```

Definition at line 6 of file [citc_controller_twist.h](#).

12.64.2.2 K1yawDefaultValue

```
#define K1yawDefaultValue 1.0
```

Definition at line 9 of file [citc_controller_twist.h](#).

12.64.2.3 K1yDefaultValue

```
#define K1yDefaultValue 1.0
```

Definition at line 7 of file [citc_controller_twist.h](#).

12.64.2.4 K1zDefaultValue

```
#define K1zDefaultValue 1.0
```

Definition at line 8 of file [citc_controller_twist.h](#).

12.64.2.5 K2xDefaultValue

```
#define K2xDefaultValue 1.0
```

Definition at line 11 of file [citc_controller_twist.h](#).

12.64.2.6 K2yawDefaultValue

```
#define K2yawDefaultValue 1.0
```

Definition at line 14 of file [citc_controller_twist.h](#).

12.64.2.7 K2yDefaultValue

```
#define K2yDefaultValue 1.0
```

Definition at line 12 of file [citc_controller_twist.h](#).

12.64.2.8 K2zDefaultValue

```
#define K2zDefaultValue 1.0
```

Definition at line 13 of file [citc_controller_twist.h](#).

12.64.2.9 K3xDefaultValue

```
#define K3xDefaultValue 1.0
```

Definition at line 16 of file [citc_controller_twist.h](#).

12.64.2.10 K3yawDefaultValue

```
#define K3yawDefaultValue 1.0
```

Definition at line 19 of file [citc_controller_twist.h](#).

12.64.2.11 K3yDefaultValue

```
#define K3yDefaultValue 1.0
```

Definition at line 17 of file [citc_controller_twist.h](#).

12.64.2.12 K3zDefaultValue

```
#define K3zDefaultValue 1.0
```

Definition at line 18 of file [citc_controller_twist.h](#).

12.64.2.13 K4xDefaultValue

```
#define K4xDefaultValue 1.0
```

Definition at line 21 of file [citc_controller_twist.h](#).

12.64.2.14 K4yawDefaultValue

```
#define K4yawDefaultValue 1.0
```

Definition at line 24 of file [citc_controller_twist.h](#).

12.64.2.15 K4yDefaultValue

```
#define K4yDefaultValue 1.0
```

Definition at line 22 of file [citc_controller_twist.h](#).

12.64.2.16 K4zDefaultValue

```
#define K4zDefaultValue 1.0
```

Definition at line 23 of file [citc_controller_twist.h](#).

12.64.2.17 LambdaX

```
#define LambdaX 6.295
```

Definition at line 31 of file [citc_controller_twist.h](#).

12.64.2.18 LambdaY

```
#define LambdaY 6.295
```

Definition at line 32 of file [citc_controller_twist.h](#).

12.64.2.19 LambdaZ

```
#define LambdaZ 4.6697
```

Definition at line 33 of file [citc_controller_twist.h](#).

12.64.2.20 MASS

```
#define MASS 0.5
```

Definition at line 40 of file [citc_controller_twist.h](#).

12.64.2.21 MAX_HORIZONTAL_SPEED

```
#define MAX_HORIZONTAL_SPEED 17.0
```

Definition at line 36 of file [citc_controller_twist.h](#).

12.64.2.22 MAX_ROTATION_SPEED

```
#define MAX_ROTATION_SPEED 100.0
```

Definition at line 38 of file [citc_controller_twist.h](#).

12.64.2.23 MAX_VERTICAL_SPEED

```
#define MAX_VERTICAL_SPEED 1.0
```

Definition at line 37 of file [citc_controller_twist.h](#).

12.64.2.24 RGxDefaultValue

```
#define RGxDefaultValue 0.0
```

Definition at line 26 of file [citc_controller_twist.h](#).

12.64.2.25 RGyawDefaultValue

```
#define RGyawDefaultValue 0.0
```

Definition at line 29 of file [citc_controller_twist.h](#).

12.64.2.26 RGyDefaultValue

```
#define RGyDefaultValue 0.0
```

Definition at line 27 of file [citic_controller_twist.h](#).

12.64.2.27 RGzDefaultValue

```
#define RGzDefaultValue 0.0
```

Definition at line 28 of file [citic_controller_twist.h](#).

12.64.2.28 Sigma

```
#define Sigma 0.8
```

Definition at line 34 of file [citic_controller_twist.h](#).

12.65 citic_controller_twist.h

[Go to the documentation of this file.](#)

```
00001
00003
00004 #include "bebop_controller/base_controller.h"
00005
00006 #define K1xDefaultValue 1.0
00007 #define K1yDefaultValue 1.0
00008 #define K1zDefaultValue 1.0
00009 #define K1yawDefaultValue 1.0
00010
00011 #define K2xDefaultValue 1.0
00012 #define K2yDefaultValue 1.0
00013 #define K2zDefaultValue 1.0
00014 #define K2yawDefaultValue 1.0
00015
00016 #define K3xDefaultValue 1.0
00017 #define K3yDefaultValue 1.0
00018 #define K3zDefaultValue 1.0
00019 #define K3yawDefaultValue 1.0
00020
00021 #define K4xDefaultValue 1.0
00022 #define K4yDefaultValue 1.0
00023 #define K4zDefaultValue 1.0
00024 #define K4yawDefaultValue 1.0
00025
00026 #define RGxDefaultValue 0.0
00027 #define RGyDefaultValue 0.0
00028 #define RGzDefaultValue 0.0
00029 #define RgyawDefaultValue 0.0
00030
00031 #define LambdaX 6.295
00032 #define LambdaY 6.295
00033 #define LambdaZ 4.6697
00034 #define Sigma 0.8
00035
00036 #define MAX_HORIZONTAL_SPEED 17.0
00037 #define MAX_VERTICAL_SPEED 1.0
00038 #define MAX_ROTATION_SPEED 100.0
00039
00040 #define MASS 0.5
00041
```

```

00042 namespace bebop_controller {
00043
00044     struct Normalize {
00045         double horizontal;
00046         double vertical;
00047         double rotation;
00048     };
00049
00050     class CITCController : public BaseController {
00051     public:
00052         CITCController();
00053
00054     private:
00055         Vector4 K1, K2, K3, K4, RG, u;
00056         Vector4 int_zeta, int_eta, last_dot_e;
00057         Vector3 lambda;
00058         Normalize norm;
00059         double mass, sigma;
00060         void CalculateCommandVelocities(geometry_msgs::Twist& ref_command_signals) override;
00061         void GetAccelerationErrors(Vector4& ddot_e, Vector4& dot_e);
00062         void IntegrateZetaAndEta(Vector4& v_nom, Vector4& ddot_e);
00063     };
00064
00065 }

```

12.66 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/data_to_csv.cpp File Reference

Node file to save the test results to a CSV file.

```
#include "data_to_csv.h"
```

Namespaces

- namespace [bebop_controller](#)
Namespace containing all the classes and functions of the Bebop Controller.

Functions

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

12.66.1 Detailed Description

Node file to save the test results to a CSV file.

Definition in file [data_to_csv.cpp](#).

12.66.2 Function Documentation

12.66.2.1 main()

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

Definition at line 118 of file [data_to_csv.cpp](#).

12.67 data_to_csv.cpp

[Go to the documentation of this file.](#)

```
00001
00002
00003
00004 #include "data_to_csv.h"
00005
00006 namespace bebop_controller {
00007
00008     DataToCSV::DataToCSV(DataToCSVParameters params):
00009     parameters(params) {
00010         drone_pose.position.x = drone_pose.position.y = drone_pose.position.z = 0;
00011         drone_pose.orientation.x = drone_pose.orientation.y = drone_pose.orientation.z = 0;
00012         reference_pose.position.x = reference_pose.position.y = reference_pose.position.z = 0;
00013         reference_pose.orientation.x = reference_pose.orientation.y = reference_pose.orientation.z =
00014     0;
00015         odometry.position.x = odometry.position.y = odometry.position.z = 0;
00016         odometry.velocity.x = odometry.velocity.y = odometry.velocity.z = 0;
00017         CSV_File.open(parameters.File.c_str());
00018         CSV_File <<
00019         "Time,x_ref,y_ref,z_ref,yaw_ref,x,y,z,yaw,v_x,v_y,v_z,v_yaw,pos_x,pos_y,pos_z,vel_x,vel_y,vel_z\n";
00020         drone_pose_sub = nh.subscribe(parameters.Topic_Drone_Pose, 1,
00021                                     &DataToCSV::Odometry_CB, this);
00022         reference_pose_sub = nh.subscribe(parameters.Topic_Reference_Pose, 1,
00023                                         &DataToCSV::Reference_CB, this);
00024         cmd_vel_sub = nh.subscribe(parameters.Topic_CMD_Vel, 1,
00025                                   &DataToCSV::Command_Velocities_CB, this);
00026         csv_begin = nh.subscribe(parameters.Topic_CSV_Begin, 1, &DataToCSV::Begin_CB, this);
00027         csv_end = nh.subscribe(parameters.Topic_CSV_End, 1, &DataToCSV::Stop_CB, this);
00028         timer = nh.createTimer(ros::Duration(0.01), &DataToCSV::Timer_CB, this);
00029         velocities_sub = nh.subscribe(parameters.Topic_Velocities, 1,
00030                                     &DataToCSV::Drone_CB, this);
00031     }
00032
00033     DataToCSV::~DataToCSV() {}
00034
00035     void DataToCSV::Begin_CB(const std_msgs::Empty::ConstPtr& empty_msg) {
00036         ROS_INFO_ONCE("Data is being stored in CSV.");
00037         Initial_Time = ros::Time::now();
00038         hasBegun = true;
00039     }
00040
00041     void DataToCSV::Stop_CB(const std_msgs::Empty::ConstPtr& empty_msg) {
00042         hasBegun = false;
00043         CSV_File.close();
00044         ROS_INFO_ONCE("Data has finished saving in CSV");
00045         ros::shutdown();
00046     }
00047
00048     void DataToCSV::Odometry_CB(const geometry_msgs::PoseStamped& pose_msg) {
00049         Eigen::Vector4d q = Eigen::Vector4d(pose_msg.pose.orientation.z, pose_msg.pose.orientation.x,
00050                                             pose_msg.pose.orientation.y, pose_msg.pose.orientation.w);
00051         //Eigen::Vector4d q = vector4FromQuaternionMsg(pose_msg.pose.orientation);
00052         Eigen::Vector3d rpy = Quat2RPY(q);
00053         drone_pose.position.x = pose_msg.pose.position.z;
00054         drone_pose.position.y = pose_msg.pose.position.x;
00055         drone_pose.position.z = pose_msg.pose.position.y;
00056         drone_pose.orientation.z = rpy.z();
00057     }
00058
00059     void DataToCSV::Reference_CB(const trajectory_msgs::MultiDOFJointTrajectoryConstPtr&
00060 reference_msg) {
00061         const size_t n_commands = reference_msg->points.size();
00062         if (n_commands < 1){
00063             return;
00064         }
00065         mav_msgs::EigenTrajectoryPoint reference;
00066         mav_msgs::eigenTrajectoryPointFromMsg(reference_msg->points.front(), &reference);
00067         reference_pose.position.x = reference.position_W[0];
```

```

00065     reference_pose.position.y = reference.position_W[1];
00066     reference_pose.position.z = reference.position_W[2];
00067     reference_pose.orientation.z = yawFromQuaternion(reference.orientation_W_B);
00068 }
00069
00070 void DataToCSV::Timer_CB(const ros::TimerEvent& event) {
00071     if (hasBegun) {
00072         ros::Time now = ros::Time::now();
00073         double t = now.toSec() - Initial_Time.toSec();
00074         if (t < 0.005) {
00075             return;
00076         }
00077         //ROS_INFO("CSV - Time: %f; Initial_Time: %f; now: %f", t, Initial_Time.toSec(),
now.toSec());
00078         CSV_File    << t << ", "
00079                     << reference_pose.position.x << ", "
00080                     << reference_pose.position.y << ", "
00081                     << reference_pose.position.z << ", "
00082                     << reference_pose.orientation.z << ", "
00083                     << drone_pose.position.x << ", "
00084                     << drone_pose.position.y << ", "
00085                     << drone_pose.position.z << ", "
00086                     << drone_pose.orientation.z << ", "
00087                     << cmd_vel.x << ", "
00088                     << cmd_vel.y << ", "
00089                     << cmd_vel.z << ", "
00090                     << cmd_vel.yaw << ", "
00091                     << odometry.position.x << ", "
00092                     << odometry.position.y << ", "
00093                     << odometry.position.z << ", "
00094                     << odometry.velocity.x << ", "
00095                     << odometry.velocity.y << ", "
00096                     << odometry.velocity.z << "\n";
00097     }
00098 }
00099
00100 void DataToCSV::Command_Velocities_CB(const geometry_msgs::Twist cmd_vel_msg) {
00101     cmd_vel.x = cmd_vel_msg.linear.x;
00102     cmd_vel.y = cmd_vel_msg.linear.y;
00103     cmd_vel.z = cmd_vel_msg.linear.z;
00104     cmd_vel.yaw = cmd_vel_msg.angular.z;
00105 }
00106
00107 void DataToCSV::Drone_CB(const nav_msgs::OdometryConstPtr& odom_msg) {
00108     odometry.position.x = odom_msg->pose.pose.position.x;
00109     odometry.position.y = odom_msg->pose.pose.position.y;
00110     odometry.position.z = odom_msg->pose.pose.position.z;
00111     odometry.velocity.x = odom_msg->twist.twist.linear.x;
00112     odometry.velocity.y = odom_msg->twist.twist.linear.y;
00113     odometry.velocity.z = odom_msg->twist.twist.linear.z;
00114 }
00115
00116 }
00117
00118 int main(int argc, char** argv){
00119     ros::init(argc, argv, "DataToCSV", ros::init_options::AnonymousName);
00120     ros::NodeHandle nh;
00121     bebop_controller::DataToCSVParameters parameters;
00122     std::string Path, DateAndTime, Name, FullPath;
00123     ros::param::get("~/Topics/Command_Trajectory", parameters.Topic_Reference_Pose);
00124     ros::param::get("~/Topics/Pose", parameters.Topic_Drone_Pose);
00125     ros::param::get("~/Topics/CMD_Vel", parameters.Topic_CMD_Vel);
00126     ros::param::get("~/Topics/CSV_Begin", parameters.Topic_CSV_Begin);
00127     ros::param::get("~/Topics/CSV_End", parameters.Topic_CSV_End);
00128     ros::param::get("~/Waypoint/MarginTime", parameters.Margin_Time);
00129     ros::param::get("~/Dir", Path);
00130     ros::param::get("/Subfolder", DateAndTime);
00131     ros::param::get("~/Topics/Velocities", parameters.Topic_Velocities);
00132     DateAndTime.pop_back();
00133     FullPath = Path + DateAndTime;
00134     mkdir(FullPath.c_str(), 0777);
00135     parameters.File = FullPath + "/" + std::string("data.csv");
00136     bebop_controller::DataToCSV DataToCSV_(parameters);
00137     ros::spin();
00138     return 0;
00139 }
00140

```

12.68 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/data_to_csv.h File Reference

Header file for the node that saves the test results to a CSV file.

```
#include "bebop_controller/common.h"
#include <sys/stat.h>
#include <fstream>
#include <iostream>
#include <string>
```

Classes

- struct [bebop_controller::DataToCSVParameters](#)
- class [bebop_controller::DataToCSV](#)

Namespaces

- namespace [bebop_controller](#)

Namespace containing all the classes and functions of the Bebop Controller.

12.68.1 Detailed Description

Header file for the node that saves the test results to a CSV file.

Definition in file [data_to_csv.h](#).

12.69 data_to_csv.h

[Go to the documentation of this file.](#)

```
00001
00002
00003
00004 #include "bebop_controller/common.h"
00005 #include <sys/stat.h>
00006 #include <fstream>
00007 #include <iostream>
00008 #include <string>
00009
00010 namespace bebop_controller {
00011
00012     struct DataToCSVParameters {
00013         std::string Topic_Drone_Pose;
00014         std::string Topic_Reference_Pose;
00015         std::string Topic_CMD_Vel;
00016         std::string Topic_CSV_Begin;
00017         std::string Topic_CSV_End;
00018         std::string File;
00019         double Initial_Time;
00020         double Margin_Time;
00021         std::string Topic_Velocities;
00022     };
00023
00024     class DataToCSV{
00025     public:
00026         DataToCSV(DataToCSVParameters params);
00027         ~DataToCSV();
00028
00029     private:
00030         State drone_pose;
00031         State reference_pose;
00032         State odometry;
00033
00034         Command_Velocities cmd_vel;
00035         DataToCSVParameters parameters;
00036
00037         ros::NodeHandle nh;
00038         ros::Subscriber drone_pose_sub;
```



```

00039         ros::Subscriber reference_pose_sub;
00040         ros::Subscriber cmd_vel_sub;
00041         ros::Subscriber csv_begin;
00042         ros::Subscriber csv_end;
00043         ros::Subscriber velocities_sub;
00044         ros::Timer timer;
00045         ros::Time Initial_Time;
00046
00047         bool hasBegun = false;
00048         std::ofstream CSV_File;
00049
00050         void Begin_CB(const std_msgs::Empty::ConstPtr& empty_msg);
00051         void Stop_CB(const std_msgs::Empty::ConstPtr& empty_msg);
00052         void Odometry_CB(const geometry_msgs::PoseStamped& pose_msg);
00053         void Reference_CB(const trajectory_msgs::MultiDOFJointTrajectoryConstPtr& reference_msg);
00054         void Command_Velocities_CB(const geometry_msgs::Twist cmd_vel_msg);
00055         void Timer_CB(const ros::TimerEvent& event);
00056         void Drone_CB(const nav_msgs::OdometryConstPtr& odom_msg);
00057     };
00058
00059 }

```

12.70 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_↵ controller/src/nodes/pid_controller_angles.cpp File Reference

Node file for PID Controller using reference angles.

```
#include "pid_controller_angles.h"
```

Namespaces

- namespace [bebop_controller](#)
Namespace containing all the classes and functions of the Bebop Controller.

Functions

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

12.70.1 Detailed Description

Node file for PID Controller using reference angles.

Definition in file [pid_controller_angles.cpp](#).

12.70.2 Function Documentation

12.70.2.1 main()

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

Definition at line 98 of file [pid_controller_angles.cpp](#).

12.71 pid_controller_angles.cpp

[Go to the documentation of this file.](#)

```

00001
00002
00003
00004 #include "pid_controller_angles.h"
00005
00006 namespace bebop_controller {
00007
00008     PIDController::PIDController() {
00009         ros::NodeHandle pnh("~");
00010         int_e.x = int_e.y = int_e.z = int_e.yaw = u_z = 0;
00011         GetRosParameter(pnh, "Gains/Px", PxDefaultValue, &P.x);
00012         GetRosParameter(pnh, "Gains/Py", PyDefaultValue, &P.y);
00013         GetRosParameter(pnh, "Gains/Pz", PzDefaultValue, &P.z);
00014         GetRosParameter(pnh, "Gains/Pyaw", PyawDefaultValue, &P.yaw);
00015         GetRosParameter(pnh, "Gains/Dx", DxDefaultValue, &D.x);
00016         GetRosParameter(pnh, "Gains/Dy", DyDefaultValue, &D.y);
00017         GetRosParameter(pnh, "Gains/Dz", DzDefaultValue, &D.z);
00018         GetRosParameter(pnh, "Gains/Dyaw", DyawDefaultValue, &D.yaw);
00019         GetRosParameter(pnh, "Gains/Ix", IxDefaultValue, &I.x);
00020         GetRosParameter(pnh, "Gains/Iy", IyDefaultValue, &I.y);
00021         GetRosParameter(pnh, "Gains/Iz", IzDefaultValue, &I.z);
00022         GetRosParameter(pnh, "Gains/Iyaw", IzyawDefaultValue, &I.yaw);
00023         GetRosParameter(pnh, "Limits/Ix", LixDefaultValue, &LI.x);
00024         GetRosParameter(pnh, "Limits/Iy", LIyDefaultValue, &LI.y);
00025         GetRosParameter(pnh, "Limits/Iz", LIzDefaultValue, &LI.z);
00026         GetRosParameter(pnh, "Limits/Iyaw", LIyawDefaultValue, &LI.yaw);
00027         GetRosParameter(pnh, "Reference_Gains/X", RGxDefaultValue, &RG.x);
00028         GetRosParameter(pnh, "Reference_Gains/Y", RGyDefaultValue, &RG.y);
00029         GetRosParameter(pnh, "Reference_Gains/Z", RGzDefaultValue, &RG.z);
00030         GetRosParameter(pnh, "Reference_Gains/Yaw", RgyawDefaultValue, &RG.yaw);
00031         GetRosParameter(pnh, "Lambda/X", LambdaX, &lambda.x);
00032         GetRosParameter(pnh, "Lambda/Y", LambdaY, &lambda.y);
00033         GetRosParameter(pnh, "Lambda/Z", LambdaZ, &lambda.z);
00034         GetRosParameter(pnh, std::string("Normalize/Max_Tilt_Angle"),
00035             MAX_TILT_ANGLE, &norm.angle);
00036         GetRosParameter(pnh, std::string("Normalize/Max_Vertical_Speed"),
00037             MAX_VERTICAL_SPEED, &norm.vertical);
00038         GetRosParameter(pnh, std::string("Normalize/Max_Rotation_Speed"),
00039             MAX_ROTATION_SPEED, &norm.rotation);
00040         GetRosParameter(pnh, "Mass", MASS, &mass);
00041     }
00042
00043     void PIDController::CalculateCommandVelocities(geometry_msgs::Twist& ref_command_signals){
00044         if (!controller_active){
00045             ref_command_signals.linear.x = 0.0;
00046             ref_command_signals.linear.y = 0.0;
00047             ref_command_signals.linear.z = 0.0;
00048             ref_command_signals.angular.z = 0.0;
00049             return;
00050         }
00051         Vector4 a, e, dot_e;
00052         GetErrors(e);
00053         CalculateLeashLength(e, P);
00054         LimitPositionErrors(e);
00055         IntegrateErrors(e);
00056         LimitIntegralPart();
00057         GetVelocityErrors(dot_e);
00058
00059         a.x = clamp(-P.x*e.x - D.x*dot_e.x - I.x*int_e.x + RG.x*command_trajectory_.acceleration_W[0],
00060             mass*(lambda.x-std::abs(command_trajectory_.acceleration_W[0])));
00061         a.y = clamp(-P.y*e.y - D.y*dot_e.y - I.y*int_e.y + RG.y*command_trajectory_.acceleration_W[1],
00062             mass*(lambda.y-std::abs(command_trajectory_.acceleration_W[1])));
00063         a.z = clamp(-P.z*e.z - D.z*dot_e.z - I.z*int_e.z + RG.z*command_trajectory_.acceleration_W[2],
00064             mass*(lambda.z-std::abs(command_trajectory_.acceleration_W[2])));
00065         a.yaw = -P.yaw*e.yaw - D.yaw*dot_e.yaw - I.yaw*int_e.yaw;
00066
00067         double u_Terr, u_T, phi_r, theta_r, psi;
00068         u_Terr = a.z + mass*GRAVITY;
00069         u_T = sqrt(pow(a.x,2) + pow(a.y,2) + pow(u_Terr,2));
00070         psi = state.orientation.z;
00071         theta_r = atan((a.x * cos(psi)) + (a.y * sin(psi)))/u_Terr;
00072         phi_r = atan(cos(theta_r)*((a.x * sin(psi)) - (a.y * cos(psi)))/(u_Terr));
00073
00074         u_z = clamp(u_z + diff*a.z,max_speed.z*norm.vertical);
00075
00076         ref_command_signals.linear.x = clamp(theta_r*RAD_2_DEG/norm.angle,max_speed.x);
00077         ref_command_signals.linear.y = clamp(-phi_r*RAD_2_DEG/norm.angle,max_speed.y);
00078         ref_command_signals.linear.z = clamp(u_z/norm.vertical,max_speed.z);
00079         ref_command_signals.angular.z = clamp(a.yaw*RAD_2_DEG/norm.rotation,max_speed.yaw);
00080     }
00081
00082     void PIDController::IntegrateErrors(Vector4& e) {
00083         int_e.x += e.x*diff;

```

```

00084         int_e.y += e.y*diff;
00085         int_e.z += e.z*diff;
00086         int_e.yaw += e.yaw*diff;
00087     }
00088
00089     void PIDController::LimitIntegralPart() {
00090         if (LI.x > 0) int_e.x = clamp(int_e.x, LI.x/I.x);
00091         if (LI.y > 0) int_e.y = clamp(int_e.y, LI.y/I.y);
00092         if (LI.z > 0) int_e.z = clamp(int_e.z, LI.z/I.z);
00093         if (LI.yaw > 0) int_e.yaw = clamp(int_e.yaw, LI.yaw/I.yaw);
00094     }
00095
00096 }
00097
00098 int main(int argc, char** argv){
00099     ros::init(argc, argv, "pid_controller_node");
00100     ros::NodeHandle nh;
00101     bebop_controller::PIDController pid_controller;
00102     ros::spin();
00103     return 0;
00104 }

```

12.72 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/pid_controller_angles.h File Reference

Header file for PID Controller using reference angles.

```
#include "bebop_controller/base_controller.h"
```

Classes

- struct [bebop_controller::Normalize](#)
- class [bebop_controller::PIDController](#)

Namespaces

- namespace [bebop_controller](#)

Namespace containing all the classes and functions of the Bebop Controller.

Macros

- #define [PxDefaultValue](#) 1.0
- #define [PyDefaultValue](#) 1.0
- #define [PzDefaultValue](#) 1.0
- #define [PyawDefaultValue](#) 1.0
- #define [DxDefaultValue](#) 1.0
- #define [DyDefaultValue](#) 1.0
- #define [DzDefaultValue](#) 1.0
- #define [DyawDefaultValue](#) 1.0
- #define [IxDefaultValue](#) 1.0
- #define [IyDefaultValue](#) 1.0
- #define [IzDefaultValue](#) 1.0
- #define [IyawDefaultValue](#) 1.0
- #define [LxDefaultValue](#) 1.0
- #define [LyDefaultValue](#) 1.0
- #define [LzDefaultValue](#) 1.0

- `#define LlyawDefaultValue 1.0`
- `#define RGxDefaultValue 0.0`
- `#define RGyDefaultValue 0.0`
- `#define RGzDefaultValue 0.0`
- `#define RGyawDefaultValue 0.0`
- `#define LambdaX 6.295`
- `#define LambdaY 6.295`
- `#define LambdaZ 4.6697`
- `#define MAX_TILT_ANGLE 20.0`
- `#define MAX_VERTICAL_SPEED 1.0`
- `#define MAX_ROTATION_SPEED 100.0`
- `#define MASS 0.5`

12.72.1 Detailed Description

Header file for PID Controller using reference angles.

Definition in file [pid_controller_angles.h](#).

12.72.2 Macro Definition Documentation

12.72.2.1 DxDefaultValue

```
#define DxDefaultValue 1.0
```

Definition at line 11 of file [pid_controller_angles.h](#).

12.72.2.2 DyawDefaultValue

```
#define DyawDefaultValue 1.0
```

Definition at line 14 of file [pid_controller_angles.h](#).

12.72.2.3 DyDefaultValue

```
#define DyDefaultValue 1.0
```

Definition at line 12 of file [pid_controller_angles.h](#).

12.72.2.4 DzDefaultValue

```
#define DzDefaultValue 1.0
```

Definition at line 13 of file [pid_controller_angles.h](#).

12.72.2.5 IxDefaultValue

```
#define IxDefaultValue 1.0
```

Definition at line 16 of file [pid_controller_angles.h](#).

12.72.2.6 IyawDefaultValue

```
#define IyawDefaultValue 1.0
```

Definition at line 19 of file [pid_controller_angles.h](#).

12.72.2.7 IyDefaultValue

```
#define IyDefaultValue 1.0
```

Definition at line 17 of file [pid_controller_angles.h](#).

12.72.2.8 IzDefaultValue

```
#define IzDefaultValue 1.0
```

Definition at line 18 of file [pid_controller_angles.h](#).

12.72.2.9 LambdaX

```
#define LambdaX 6.295
```

Definition at line 31 of file [pid_controller_angles.h](#).

12.72.2.10 LambdaY

```
#define LambdaY 6.295
```

Definition at line 32 of file [pid_controller_angles.h](#).

12.72.2.11 LambdaZ

```
#define LambdaZ 4.6697
```

Definition at line 33 of file [pid_controller_angles.h](#).

12.72.2.12 LlxDefaultValue

```
#define LlxDefaultValue 1.0
```

Definition at line 21 of file [pid_controller_angles.h](#).

12.72.2.13 LIyawDefaultValue

```
#define LIyawDefaultValue 1.0
```

Definition at line 24 of file [pid_controller_angles.h](#).

12.72.2.14 LIyDefaultValue

```
#define LIyDefaultValue 1.0
```

Definition at line 22 of file [pid_controller_angles.h](#).

12.72.2.15 LIzDefaultValue

```
#define LIzDefaultValue 1.0
```

Definition at line 23 of file [pid_controller_angles.h](#).

12.72.2.16 MASS

```
#define MASS 0.5
```

Definition at line 39 of file [pid_controller_angles.h](#).

12.72.2.17 MAX_ROTATION_SPEED

```
#define MAX_ROTATION_SPEED 100.0
```

Definition at line 37 of file [pid_controller_angles.h](#).

12.72.2.18 MAX_TILT_ANGLE

```
#define MAX_TILT_ANGLE 20.0
```

Definition at line 35 of file [pid_controller_angles.h](#).

12.72.2.19 MAX_VERTICAL_SPEED

```
#define MAX_VERTICAL_SPEED 1.0
```

Definition at line 36 of file [pid_controller_angles.h](#).

12.72.2.20 PxDefaultValue

```
#define PxDefaultValue 1.0
```

Definition at line 6 of file [pid_controller_angles.h](#).

12.72.2.21 PyawDefaultValue

```
#define PyawDefaultValue 1.0
```

Definition at line 9 of file [pid_controller_angles.h](#).

12.72.2.22 PyDefaultValue

```
#define PyDefaultValue 1.0
```

Definition at line 7 of file [pid_controller_angles.h](#).

12.72.2.23 PzDefaultValue

```
#define PzDefaultValue 1.0
```

Definition at line 8 of file [pid_controller_angles.h](#).

12.72.2.24 RGxDefaultValue

```
#define RGxDefaultValue 0.0
```

Definition at line 26 of file [pid_controller_angles.h](#).

12.72.2.25 RGyawDefaultValue

```
#define RGyawDefaultValue 0.0
```

Definition at line 29 of file [pid_controller_angles.h](#).

12.72.2.26 RGyDefaultValue

```
#define RGyDefaultValue 0.0
```

Definition at line 27 of file [pid_controller_angles.h](#).

12.72.2.27 RGzDefaultValue

```
#define RGzDefaultValue 0.0
```

Definition at line 28 of file [pid_controller_angles.h](#).

12.73 pid_controller_angles.h

[Go to the documentation of this file.](#)

```

00001
00003
00004 #include "bebop_controller/base_controller.h"
00005
00006 #define PxDefaultValue 1.0
00007 #define PyDefaultValue 1.0
00008 #define PzDefaultValue 1.0
00009 #define PyawDefaultValue 1.0
00010
00011 #define DxDefaultValue 1.0
00012 #define DyDefaultValue 1.0
00013 #define DzDefaultValue 1.0
00014 #define DyawDefaultValue 1.0
00015
00016 #define IxDefaultValue 1.0
00017 #define IyDefaultValue 1.0
00018 #define IzDefaultValue 1.0
00019 #define IyawDefaultValue 1.0
00020
00021 #define LixDefaultValue 1.0
00022 #define LIyDefaultValue 1.0
00023 #define LIzDefaultValue 1.0
00024 #define LIyawDefaultValue 1.0
00025
00026 #define RGxDefaultValue 0.0
00027 #define RGyDefaultValue 0.0
00028 #define RGzDefaultValue 0.0
00029 #define RgyawDefaultValue 0.0
00030
00031 #define LambdaX 6.295
00032 #define LambdaY 6.295
00033 #define LambdaZ 4.6697
00034
00035 #define MAX_TILT_ANGLE 20.0
00036 #define MAX_VERTICAL_SPEED 1.0
00037 #define MAX_ROTATION_SPEED 100.0
00038
00039 #define MASS 0.5
00040
00041 namespace bebop_controller {
00042
00043     struct Normalize {
00044         double angle;
00045         double vertical;
00046         double rotation;
00047     };
00048
00049     class PIDController : public BaseController {
00050     public:
00051         PIDController();
00052
00053     private:
00054         Vector4 P, D, I, LI, RG, int_e;
00055         Vector3 lambda;
00056         Normalize norm;
00057         double mass, u_z;
00058         void CalculateCommandVelocities(geometry_msgs::Twist& ref_command_signals) override;
00059         void IntegrateErrors(Vector4& e);
00060         void LimitIntegralPart();
00061     };
00062
00063 }

```

12.74 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_↵ controller/src/nodes/pid_controller_twist.cpp File Reference

Node file for PID Controller using velocity commands.

```
#include "pid_controller_twist.h"
```

Namespaces

- namespace [bebop_controller](#)

Namespace containing all the classes and functions of the Bebop Controller.

Functions

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

12.74.1 Detailed Description

Node file for PID Controller using velocity commands.

Definition in file [pid_controller_twist.cpp](#).

12.74.2 Function Documentation

12.74.2.1 main()

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

Definition at line 92 of file [pid_controller_twist.cpp](#).

12.75 pid_controller_twist.cpp

[Go to the documentation of this file.](#)

```
00001
00002
00003
00004 #include "pid_controller_twist.h"
00005
00006 namespace bebop_controller {
00007
00008     PIDController::PIDController() {
00009         u.x = u.y = u.z = u.yaw = 0;
00010         int_e.x = int_e.y = int_e.z = int_e.yaw = 0;
00011         ros::NodeHandle pnh("~");
00012         GetRosParameter(pnh, "Gains/Px", PxDefaultValue, &P.x);
00013         GetRosParameter(pnh, "Gains/Py", PyDefaultValue, &P.y);
00014         GetRosParameter(pnh, "Gains/Pz", PzDefaultValue, &P.z);
00015         GetRosParameter(pnh, "Gains/Pyaw", PyawDefaultValue, &P.yaw);
00016         GetRosParameter(pnh, "Gains/Dx", DxDefaultValue, &D.x);
00017         GetRosParameter(pnh, "Gains/Dy", DyDefaultValue, &D.y);
00018         GetRosParameter(pnh, "Gains/Dz", DzDefaultValue, &D.z);
00019         GetRosParameter(pnh, "Gains/Dyaw", DyawDefaultValue, &D.yaw);
00020         GetRosParameter(pnh, "Gains/Ix", IxDefaultValue, &I.x);
00021         GetRosParameter(pnh, "Gains/Iy", IyDefaultValue, &I.y);
00022         GetRosParameter(pnh, "Gains/Iz", IzDefaultValue, &I.z);
00023         GetRosParameter(pnh, "Gains/Iyaw", IzyawDefaultValue, &I.yaw);
00024         GetRosParameter(pnh, "Limits/Ix", LIxDefaultValue, &LI.x);
00025         GetRosParameter(pnh, "Limits/Iy", LIyDefaultValue, &LI.y);
00026         GetRosParameter(pnh, "Limits/Iz", LIzDefaultValue, &LI.z);
00027         GetRosParameter(pnh, "Limits/Iyaw", LIyawDefaultValue, &LI.yaw);
00028         GetRosParameter(pnh, "Reference_Gains/X", RGxDefaultValue, &RG.x);
00029         GetRosParameter(pnh, "Reference_Gains/Y", RgyDefaultValue, &RG.y);
```

```

00030     GetRosParameter(pnh, "Reference_Gains/Z", RGzDefaultValue, &RG.z);
00031     GetRosParameter(pnh, "Reference_Gains/Yaw", RgyawDefaultValue, &RG.yaw);
00032     GetRosParameter(pnh, "Lambda/X", LambdaX, &lambda.x);
00033     GetRosParameter(pnh, "Lambda/Y", LambdaY, &lambda.y);
00034     GetRosParameter(pnh, "Lambda/Z", LambdaZ, &lambda.z);
00035     GetRosParameter(pnh, std::string("Normalize/Max_Horizontal_Speed"),
00036                     MAX_HORIZONTAL_SPEED, &norm.horizontal);
00037     GetRosParameter(pnh, std::string("Normalize/Max_Vertical_Speed"),
00038                     MAX_VERTICAL_SPEED, &norm.vertical);
00039     GetRosParameter(pnh, std::string("Normalize/Max_Rotation_Speed"),
00040                     MAX_ROTATION_SPEED, &norm.rotation);
00041     GetRosParameter(pnh, "Mass", MASS, &mass);
00042 }
00043
00044 void PIDController::CalculateCommandVelocities(geometry_msgs::Twist& ref_command_signals){
00045     if (!controller_active_){
00046         ref_command_signals.linear.x = 0.0;
00047         ref_command_signals.linear.y = 0.0;
00048         ref_command_signals.linear.z = 0.0;
00049         ref_command_signals.angular.z = 0.0;
00050         return;
00051     }
00052     Vector4 a, e, dot_e;
00053     GetErrors(e);
00054     CalculateLeashLength(e, P);
00055     LimitPositionErrors(e);
00056     IntegrateErrors(e);
00057     LimitIntegralPart();
00058     GetVelocityErrors(dot_e);
00059
00060     a.x = -P.x*e.x - D.x*dot_e.x - I.x*int_e.x + RG.z*command_trajectory_.acceleration_W[0];
00061     a.y = -P.y*e.y - D.y*dot_e.y - I.y*int_e.y + RG.z*command_trajectory_.acceleration_W[1];
00062     a.z = -P.z*e.z - D.z*dot_e.z - I.z*int_e.z + RG.z*command_trajectory_.acceleration_W[2];
00063     a.yaw = -P.yaw*e.yaw - D.yaw*dot_e.yaw - I.yaw*int_e.yaw;
00064
00065     u.x = clamp(u.x + (diff*a.x)/norm.horizontal,max_speed.x);
00066     u.y = clamp(u.y + (diff*a.y)/norm.horizontal,max_speed.y);
00067     u.z = clamp(u.z + (diff*a.z)/norm.vertical,max_speed.z);
00068     u.yaw = clamp(u.yaw + (diff*a.yaw)*RAD_2_DEG/norm.rotation,max_speed.yaw);
00069
00070     ref_command_signals.linear.x = clamp(cos(state.orientation.z)*u.x -
00071     sin(state.orientation.z)*u.y,max_speed.x);
00072     ref_command_signals.linear.y = clamp(sin(state.orientation.z)*u.x +
00073     cos(state.orientation.z)*u.y,max_speed.y);
00074     ref_command_signals.linear.z = clamp(u.z,max_speed.z);
00075     ref_command_signals.angular.z = clamp(u.yaw,max_speed.yaw);
00076 }
00077
00078 void PIDController::IntegrateErrors(Vector4& e) {
00079     int_e.x += e.x*diff;
00080     int_e.y += e.y*diff;
00081     int_e.z += e.z*diff;
00082     int_e.yaw += e.yaw*diff;
00083 }
00084
00085 void PIDController::LimitIntegralPart() {
00086     if (LI.x > 0) int_e.x = clamp(int_e.x, LI.x/I.x);
00087     if (LI.y > 0) int_e.y = clamp(int_e.y, LI.y/I.y);
00088     if (LI.z > 0) int_e.z = clamp(int_e.z, LI.z/I.z);
00089     if (LI.yaw > 0) int_e.yaw = clamp(int_e.yaw, LI.yaw/I.yaw);
00090 }
00091
00092 int main(int argc, char** argv){
00093     ros::init(argc, argv, "pid_controller_node");
00094     ros::NodeHandle nh;
00095     bebop_controller::PIDController pid_controller;
00096     ros::spin();
00097     return 0;
00098 }

```

12.76 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/pid_controller_twist.h File Reference

Header file for PID Controller using velocity commands.

```
#include "bebop_controller/base_controller.h"
```

Classes

- struct [bebop_controller::Normalize](#)
- class [bebop_controller::PIDController](#)

Namespaces

- namespace [bebop_controller](#)

Namespace containing all the classes and functions of the Bebop Controller.

Macros

- `#define PxDefaultValue 1.0`
- `#define PyDefaultValue 1.0`
- `#define PzDefaultValue 1.0`
- `#define PyawDefaultValue 1.0`
- `#define DxDefaultValue 1.0`
- `#define DyDefaultValue 1.0`
- `#define DzDefaultValue 1.0`
- `#define DyawDefaultValue 1.0`
- `#define LxDefaultValue 1.0`
- `#define LyDefaultValue 1.0`
- `#define LzDefaultValue 1.0`
- `#define lyawDefaultValue 1.0`
- `#define LlxDefaultValue 1.0`
- `#define LlyDefaultValue 1.0`
- `#define LlzDefaultValue 1.0`
- `#define LlyawDefaultValue 1.0`
- `#define RGxDefaultValue 0.0`
- `#define RGyDefaultValue 0.0`
- `#define RGzDefaultValue 0.0`
- `#define RGyawDefaultValue 0.0`
- `#define LambdaX 6.295`
- `#define LambdaY 6.295`
- `#define LambdaZ 4.6697`
- `#define MAX_HORIZONTAL_SPEED 17.0`
- `#define MAX_VERTICAL_SPEED 1.0`
- `#define MAX_ROTATION_SPEED 100.0`
- `#define MASS 0.5`

12.76.1 Detailed Description

Header file for PID Controller using velocity commands.

Definition in file [pid_controller_twist.h](#).

12.76.2 Macro Definition Documentation

12.76.2.1 DxDefaultValue

```
#define DxDefaultValue 1.0
```

Definition at line 11 of file [pid_controller_twist.h](#).

12.76.2.2 DyawDefaultValue

```
#define DyawDefaultValue 1.0
```

Definition at line 14 of file [pid_controller_twist.h](#).

12.76.2.3 DyDefaultValue

```
#define DyDefaultValue 1.0
```

Definition at line 12 of file [pid_controller_twist.h](#).

12.76.2.4 DzDefaultValue

```
#define DzDefaultValue 1.0
```

Definition at line 13 of file [pid_controller_twist.h](#).

12.76.2.5 IxDefaultValue

```
#define IxDefaultValue 1.0
```

Definition at line 16 of file [pid_controller_twist.h](#).

12.76.2.6 IyawDefaultValue

```
#define IyawDefaultValue 1.0
```

Definition at line 19 of file [pid_controller_twist.h](#).

12.76.2.7 IyDefaultValue

```
#define IyDefaultValue 1.0
```

Definition at line 17 of file [pid_controller_twist.h](#).

12.76.2.8 IzDefaultValue

```
#define IzDefaultValue 1.0
```

Definition at line 18 of file [pid_controller_twist.h](#).

12.76.2.9 LambdaX

```
#define LambdaX 6.295
```

Definition at line 31 of file [pid_controller_twist.h](#).

12.76.2.10 LambdaY

```
#define LambdaY 6.295
```

Definition at line 32 of file [pid_controller_twist.h](#).

12.76.2.11 LambdaZ

```
#define LambdaZ 4.6697
```

Definition at line 33 of file [pid_controller_twist.h](#).

12.76.2.12 LlxDefaultValue

```
#define LlxDefaultValue 1.0
```

Definition at line 21 of file [pid_controller_twist.h](#).

12.76.2.13 LlyawDefaultValue

```
#define LlyawDefaultValue 1.0
```

Definition at line 24 of file [pid_controller_twist.h](#).

12.76.2.14 LlyDefaultValue

```
#define LlyDefaultValue 1.0
```

Definition at line 22 of file [pid_controller_twist.h](#).

12.76.2.15 LlzDefaultValue

```
#define LlzDefaultValue 1.0
```

Definition at line 23 of file [pid_controller_twist.h](#).

12.76.2.16 MASS

```
#define MASS 0.5
```

Definition at line 39 of file [pid_controller_twist.h](#).

12.76.2.17 MAX_HORIZONTAL_SPEED

```
#define MAX_HORIZONTAL_SPEED 17.0
```

Definition at line 35 of file [pid_controller_twist.h](#).

12.76.2.18 MAX_ROTATION_SPEED

```
#define MAX_ROTATION_SPEED 100.0
```

Definition at line 37 of file [pid_controller_twist.h](#).

12.76.2.19 MAX_VERTICAL_SPEED

```
#define MAX_VERTICAL_SPEED 1.0
```

Definition at line 36 of file [pid_controller_twist.h](#).

12.76.2.20 PxDefaultValue

```
#define PxDefaultValue 1.0
```

Definition at line 6 of file [pid_controller_twist.h](#).

12.76.2.21 PyawDefaultValue

```
#define PyawDefaultValue 1.0
```

Definition at line 9 of file [pid_controller_twist.h](#).

12.76.2.22 PyDefaultValue

```
#define PyDefaultValue 1.0
```

Definition at line 7 of file [pid_controller_twist.h](#).

12.76.2.23 PzDefaultValue

```
#define PzDefaultValue 1.0
```

Definition at line 8 of file [pid_controller_twist.h](#).

12.76.2.24 RGxDefaultValue

```
#define RGxDefaultValue 0.0
```

Definition at line 26 of file [pid_controller_twist.h](#).

12.76.2.25 RGyawDefaultValue

```
#define RGyawDefaultValue 0.0
```

Definition at line 29 of file [pid_controller_twist.h](#).

12.76.2.26 RGyDefaultValue

```
#define RGyDefaultValue 0.0
```

Definition at line 27 of file [pid_controller_twist.h](#).

12.76.2.27 RGzDefaultValue

```
#define RGzDefaultValue 0.0
```

Definition at line 28 of file [pid_controller_twist.h](#).

12.77 pid_controller_twist.h

[Go to the documentation of this file.](#)

```
00001
00003
00004 #include "bebop_controller/base_controller.h"
00005
00006 #define PxDefaultValue 1.0
00007 #define PyDefaultValue 1.0
00008 #define PzDefaultValue 1.0
00009 #define PyawDefaultValue 1.0
00010
00011 #define DxDefaultValue 1.0
00012 #define DyDefaultValue 1.0
00013 #define DzDefaultValue 1.0
00014 #define DyawDefaultValue 1.0
00015
00016 #define IxDefaultValue 1.0
00017 #define IyDefaultValue 1.0
00018 #define IzDefaultValue 1.0
00019 #define IyawDefaultValue 1.0
00020
00021 #define LixDefaultValue 1.0
00022 #define LIyDefaultValue 1.0
00023 #define LIzDefaultValue 1.0
00024 #define LIyawDefaultValue 1.0
00025
00026 #define RGxDefaultValue 0.0
00027 #define RGyDefaultValue 0.0
00028 #define RGzDefaultValue 0.0
00029 #define RGyawDefaultValue 0.0
00030
00031 #define LambdaX 6.295
00032 #define LambdaY 6.295
00033 #define LambdaZ 4.6697
00034
00035 #define MAX_HORIZONTAL_SPEED 17.0
00036 #define MAX_VERTICAL_SPEED 1.0
00037 #define MAX_ROTATION_SPEED 100.0
00038
00039 #define MASS 0.5
00040
00041 namespace bebop_controller {
```

```

00042
00043     struct Normalize {
00044         double horizontal;
00045         double vertical;
00046         double rotation;
00047     };
00048
00049     class PIDController : public BaseController {
00050     public:
00051         PIDController();
00052
00053     private:
00054         Vector4 P, D, I, LI, u, RG, int_e;
00055         Vector3 lambda;
00056         Normalize norm;
00057         double mass;
00058         void CalculateCommandVelocities(geometry_msgs::Twist& ref_command_signals) override;
00059         void IntegrateErrors(Vector4& e);
00060         void LimitIntegralPart();
00061     };
00062
00063 }

```

12.78 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_↵ controller/src/nodes/proportional_controller.cpp File Reference

Node file for proportional controller.

```
#include "proportional_controller.h"
```

Namespaces

- namespace [bebop_controller](#)
Namespace containing all the classes and functions of the Bebop Controller.

Functions

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

12.78.1 Detailed Description

Node file for proportional controller.

Definition in file [proportional_controller.cpp](#).

12.78.2 Function Documentation

12.78.2.1 main()

```

int main (
    int argc,
    char ** argv )

```

Definition at line 52 of file [proportional_controller.cpp](#).

12.79 proportional_controller.cpp

[Go to the documentation of this file.](#)

```

00001
00002
00003
00004 #include "proportional_controller.h"
00005
00006 namespace bebop_controller {
00007
00008     ProportionalController::ProportionalController() {
00009         ros::NodeHandle pnh("~");
00010         GetRosParameter(pnh, "Gains/Px", PxDefaultValue, &P.x);
00011         GetRosParameter(pnh, "Gains/Py", PyDefaultValue, &P.y);
00012         GetRosParameter(pnh, "Gains/Pz", PzDefaultValue, &P.z);
00013         GetRosParameter(pnh, "Gains/Pyaw", PyawDefaultValue, &P.yaw);
00014         GetRosParameter(pnh, "Reference_Gains/X", RGxDefaultValue, &RG.x);
00015         GetRosParameter(pnh, "Reference_Gains/Y", RGyDefaultValue, &RG.y);
00016         GetRosParameter(pnh, "Reference_Gains/Z", RGzDefaultValue, &RG.z);
00017         GetRosParameter(pnh, "Reference_Gains/Yaw", RGyawDefaultValue, &RG.yaw);
00018         GetRosParameter(pnh, std::string("Normalize/Max_Horizontal_Speed"),
00019             MAX_HORIZONTAL_SPEED, &norm.horizontal);
00020         GetRosParameter(pnh, std::string("Normalize/Max_Vertical_Speed"),
00021             MAX_VERTICAL_SPEED, &norm.vertical);
00022         GetRosParameter(pnh, std::string("Normalize/Max_Rotation_Speed"),
00023             MAX_ROTATION_SPEED, &norm.rotation);
00024     }
00025
00026     void ProportionalController::CalculateCommandVelocities(geometry_msgs::Twist&
ref_command_signals) {
00027         if (!controller_active_) {
00028             ref_command_signals.linear.x = 0.0;
00029             ref_command_signals.linear.y = 0.0;
00030             ref_command_signals.linear.z = 0.0;
00031             ref_command_signals.angular.z = 0.0;
00032             return;
00033         }
00034         Vector4 u, e;
00035         GetErrors(e);
00036         CalculateLeashLength(e, P);
00037         LimitPositionErrors(e);
00038
00039         u.x = (-P.x*e.x + RG.x*command_trajectory_.velocity_W[0])/norm.horizontal;
00040         u.y = (-P.y*e.y + RG.y*command_trajectory_.velocity_W[1])/norm.horizontal;
00041         u.z = (-P.z*e.z + RG.z*command_trajectory_.velocity_W[2])/norm.vertical;
00042         u.yaw = (-P.yaw*e.yaw +
RG.yaw*command_trajectory_.angular_velocity_W[2])*RAD_2_DEG/norm.rotation;
00043
00044         ref_command_signals.linear.x = clamp(cos(state.orientation.z)*u.x -
sin(state.orientation.z)*u.y,max_speed.x);
00045         ref_command_signals.linear.y = clamp(sin(state.orientation.z)*u.x +
cos(state.orientation.z)*u.y,max_speed.y);
00046         ref_command_signals.linear.z = clamp(u.z,max_speed.z);
00047         ref_command_signals.angular.z = clamp(u.yaw,max_speed.yaw);
00048     }
00049
00050 }
00051
00052 int main(int argc, char** argv){
00053     ros::init(argc, argv, "proportional_controller_node");
00054     ros::NodeHandle nh;
00055     bebop_controller::ProportionalController proportional_controller;
00056     ros::spin();
00057     return 0;
00058 }

```

12.80 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_↵ controller/src/nodes/proportional_controller.h File Reference

Header file for proportional controller.

```
#include "bebop_controller/base_controller.h"
```

Classes

- struct [bebop_controller::Normalize](#)
- class [bebop_controller::ProportionalController](#)

Namespaces

- namespace [bebop_controller](#)
Namespace containing all the classes and functions of the Bebop Controller.

Macros

- `#define PxDefaultValue 1.0`
- `#define PyDefaultValue 1.0`
- `#define PzDefaultValue 1.0`
- `#define PyawDefaultValue 1.0`
- `#define RGxDefaultValue 0.0`
- `#define RGyDefaultValue 0.0`
- `#define RGzDefaultValue 0.0`
- `#define RGyawDefaultValue 0.0`
- `#define MAX_HORIZONTAL_SPEED 17.0`
- `#define MAX_VERTICAL_SPEED 1.0`
- `#define MAX_ROTATION_SPEED 100.0`

12.80.1 Detailed Description

Header file for proportional controller.

Definition in file [proportional_controller.h](#).

12.80.2 Macro Definition Documentation

12.80.2.1 MAX_HORIZONTAL_SPEED

```
#define MAX_HORIZONTAL_SPEED 17.0
```

Definition at line 16 of file [proportional_controller.h](#).

12.80.2.2 MAX_ROTATION_SPEED

```
#define MAX_ROTATION_SPEED 100.0
```

Definition at line 18 of file [proportional_controller.h](#).

12.80.2.3 MAX_VERTICAL_SPEED

```
#define MAX_VERTICAL_SPEED 1.0
```

Definition at line 17 of file [proportional_controller.h](#).

12.80.2.4 PxDefaultValue

```
#define PxDefaultValue 1.0
```

Definition at line 6 of file [proportional_controller.h](#).

12.80.2.5 PyawDefaultValue

```
#define PyawDefaultValue 1.0
```

Definition at line 9 of file [proportional_controller.h](#).

12.80.2.6 PyDefaultValue

```
#define PyDefaultValue 1.0
```

Definition at line 7 of file [proportional_controller.h](#).

12.80.2.7 PzDefaultValue

```
#define PzDefaultValue 1.0
```

Definition at line 8 of file [proportional_controller.h](#).

12.80.2.8 RGxDefaultValue

```
#define RGxDefaultValue 0.0
```

Definition at line 11 of file [proportional_controller.h](#).

12.80.2.9 RYawDefaultValue

```
#define RYawDefaultValue 0.0
```

Definition at line 14 of file [proportional_controller.h](#).

12.80.2.10 RGyDefaultValue

```
#define RGyDefaultValue 0.0
```

Definition at line 12 of file [proportional_controller.h](#).

12.80.2.11 RGzDefaultValue

```
#define RGzDefaultValue 0.0
```

Definition at line 13 of file [proportional_controller.h](#).

12.81 proportional_controller.h

[Go to the documentation of this file.](#)

```
00001
00002
00003
00004 #include "bebop_controller/base_controller.h"
00005
00006 #define PxDefaultValue 1.0
00007 #define PyDefaultValue 1.0
00008 #define PzDefaultValue 1.0
00009 #define PyawDefaultValue 1.0
00010
00011 #define RGxDefaultValue 0.0
00012 #define RGyDefaultValue 0.0
00013 #define RGzDefaultValue 0.0
00014 #define RYawDefaultValue 0.0
00015
00016 #define MAX_HORIZONTAL_SPEED 17.0
00017 #define MAX_VERTICAL_SPEED 1.0
00018 #define MAX_ROTATION_SPEED 100.0
00019
00020 namespace bebop_controller {
00021
00022     struct Normalize {
00023         double horizontal;
00024         double vertical;
00025         double rotation;
00026     };
00027
00028     class ProportionalController : public BaseController {
00029     public:
00030         ProportionalController();
00031
00032     private:
00033         Vector4 P, RG;
00034         Normalize norm;
00035         void CalculateCommandVelocities(geometry_msgs::Twist& ref_command_signals) override;
00036     };
00037
00038 }
```

12.82 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/sinusoidal.cpp File Reference

```
#include "sinusoidal.h"
```

Namespaces

- namespace [bebop_controller](#)

Namespace containing all the classes and functions of the Bebop Controller.

Functions

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

12.82.1 Function Documentation

12.82.1.1 main()

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

Definition at line 83 of file [sinusoidal.cpp](#).

12.83 sinusoidal.cpp

[Go to the documentation of this file.](#)

```
00001  
00003  
00004 #include "sinusoidal.h"  
00005  
00006 namespace bebop_controller {  
00007  
00008     Sinusoidal::Sinusoidal(WaypointParameters wp_params, TrajectoryParameters t_params):  
00009         Waypoint(wp_params),  
00010         t_params_(t_params) {}  
00011  
00012     void Sinusoidal::Trajectory_CB(const ros::TimerEvent& event) {  
00013         ros::Time now = ros::Time::now();  
00014         double x, y, z, yaw;  
00015         double dot_x, dot_y, dot_z, dot_yaw;  
00016         double ddot_x, ddot_y, ddot_z;  
00017         double t = now.toSec() - Initial_Time.toSec() + wp_params_.DiffTime;  
00018         double W1, W2, W3;  
00019         W1 = W3 = 2*M_PI/wp_params_.TrajectoryTime;  
00020         W2 = 4*M_PI/wp_params_.TrajectoryTime;  
00021  
00022         x = t_params_.TrajectoryDistance.x() * sin(W1*t) + t_params_.PositionBeforeTrajectory.x();  
00023         y = t_params_.TrajectoryDistance.y() * sin(W2*t) + t_params_.PositionBeforeTrajectory.y();  
00024         z = t_params_.TrajectoryDistance.z() * sin(W3*t) + t_params_.PositionBeforeTrajectory.z();  
00025  
00026         dot_x = t_params_.TrajectoryDistance.x() * W1 * cos(W1*t);  
00027         dot_y = t_params_.TrajectoryDistance.y() * W2 * cos(W2*t);  
00028         dot_z = t_params_.TrajectoryDistance.z() * W3 * cos(W3*t);
```

```

00029
00030     ddot_x = -t_params_.TrajectoryDistance.x() * pow(W1, 2) * sin(W1*t);
00031     ddot_y = -t_params_.TrajectoryDistance.y() * pow(W2, 2) * sin(W2*t);
00032     ddot_z = -t_params_.TrajectoryDistance.z() * pow(W3, 2) * sin(W3*t);
00033
00034     switch (status)
00035     {
00036     case BeforeTrajectory:
00037     case AfterTrajectory:
00038         x = t_params_.PositionBeforeTrajectory.x();
00039         y = t_params_.PositionBeforeTrajectory.y();
00040         z = t_params_.PositionBeforeTrajectory.z();
00041         if (t_params_.Yaw_Enabled) {
00042             yaw = atan2(2*t_params_.TrajectoryDistance.y(), t_params_.TrajectoryDistance.x()) +
t_params_.Yaw_Offset;
00043         }
00044         else {
00045             yaw = 0;
00046         }
00047         dot_x = dot_y = dot_z = ddot_x = ddot_y = ddot_z = dot_yaw = 0;
00048         break;
00049
00050     case Trajectory:
00051         if (t > wp_params_.TrajectoryTime) {
00052             status = AfterTrajectory;
00053             return;
00054         }
00055         if (t_params_.Yaw_Enabled) {
00056             yaw = atan2(dot_y, dot_x) + t_params_.Yaw_Offset;
00057             dot_yaw = ((dot_x*ddot_y - dot_y*ddot_x) / pow(dot_x, 2)) / (1+pow(dot_y/dot_x, 2));
00058         }
00059         else {
00060             yaw = dot_yaw = 0;
00061         }
00062         break;
00063     }
00064
00065     Eigen::Vector3d desired_position(x, y, z);
00066     Eigen::Vector3d velocities(dot_x, dot_y, dot_z);
00067     Eigen::Vector3d accelerations(ddot_x, ddot_y, ddot_z);
00068
00069     mav_msgs::EigenTrajectoryPoint point;
00070     point.position_W = desired_position;
00071     point.setFromYaw(yaw);
00072     point.velocity_W = velocities;
00073     point.setFromYawRate(dot_yaw);
00074     point.acceleration_W = accelerations;
00075
00076     mav_msgs::msgMultiDofJointTrajectoryFromEigen(point, &position_target_);
00077
00078     setpoint_pub_.publish(position_target_);
00079 }
00080 }
00081 }
00082
00083 int main(int argc, char** argv){
00084     ros::init(argc, argv, "sinusoidal");
00085     ros::NodeHandle nh;
00086
00087     double X_I, Y_I, Z_I, DIST_X, DIST_Y, DIST_Z;
00088     WaypointParameters wp_params;
00089     TrajectoryParameters t_params;
00090
00091     ros::param::get("~/Waypoint/TimeBeforeTrajectory", wp_params.TimeBeforeTrajectory);
00092     ros::param::get("~/Waypoint/TrajectoryTime", wp_params.TrajectoryTime);
00093     ros::param::get("~/Waypoint/DiffTime", wp_params.DiffTime);
00094     ros::param::get("~/Waypoint/MarginTime", wp_params.MarginTime);
00095     ros::param::get("~/Trajectory/X_Initial", X_I);
00096     ros::param::get("~/Trajectory/Y_Initial", Y_I);
00097     ros::param::get("~/Trajectory/Z_Initial", Z_I);
00098     ros::param::get("~/Trajectory/X_Distance", DIST_X);
00099     ros::param::get("~/Trajectory/Y_Distance", DIST_Y);
00100     ros::param::get("~/Trajectory/Z_Distance", DIST_Z);
00101     ros::param::get("~/Yaw_Enabled", t_params.Yaw_Enabled);
00102     ros::param::get("~/Yaw_Offset", t_params.Yaw_Offset);
00103     ros::param::get("~/Topics/Command_Trajectory", wp_params.topic_command_trajectory);
00104     ros::param::get("~/Topics/CSV_Begin", wp_params.topic_csv_begin);
00105     ros::param::get("~/Topics/CSV_End", wp_params.topic_csv_end);
00106
00107     t_params.PositionBeforeTrajectory = Eigen::Vector3d(X_I, Y_I, Z_I);
00108     t_params.TrajectoryDistance = Eigen::Vector3d(DIST_X, DIST_Y, DIST_Z);
00109
00110     bebop_controller::Sinusoidal waypoint(wp_params, t_params);
00111     ros::spin();
00112     return 0;
00113 }

```


12.84 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/sinusoidal.h File Reference

```
#include "bebop_controller/waypoint.h"
```

Classes

- struct [TrajectoryParameters](#)
- class [bebop_controller::Sinusoidal](#)

Namespaces

- namespace [bebop_controller](#)

Namespace containing all the classes and functions of the Bebop Controller.

12.85 sinusoidal.h

[Go to the documentation of this file.](#)

```
00001
00002
00003
00004 #include "bebop_controller/waypoint.h"
00005
00006 struct TrajectoryParameters {
00007     Eigen::Vector3d PositionBeforeTrajectory;
00008     Eigen::Vector3d TrajectoryDistance;
00009     bool Yaw_Enabled;
00010     double Yaw_Offset;
00011 };
00012
00013 namespace bebop_controller {
00014
00015     class Sinusoidal : public Waypoint {
00016     public:
00017         Sinusoidal(WaypointParameters wp_params, TrajectoryParameters t_params);
00018
00019     private:
00020         TrajectoryParameters t_params_;
00021         void Trajectory_CB(const ros::TimerEvent& event) override;
00022     };
00023
00024 }
```

12.86 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/square_root_controller.cpp File Reference

Node file for square root controller.

```
#include "square_root_controller.h"
```

Namespaces

- namespace [bebop_controller](#)

Namespace containing all the classes and functions of the Bebop Controller.

Functions

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

12.86.1 Detailed Description

Node file for square root controller.

Definition in file [square_root_controller.cpp](#).

12.86.2 Function Documentation

12.86.2.1 main()

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

Definition at line 59 of file [square_root_controller.cpp](#).

12.87 square_root_controller.cpp

[Go to the documentation of this file.](#)

```
00001
00002
00003
00004 #include "square_root_controller.h"
00005
00006 namespace bebop_controller {
00007
00008     SquareRootController::SquareRootController() {
00009         ros::NodeHandle pnh("~");
00010         GetRosParameter(pnh, "Gains/Px", PxDefaultValue, &P.x);
00011         GetRosParameter(pnh, "Gains/Py", PyDefaultValue, &P.y);
00012         GetRosParameter(pnh, "Gains/Pz", PzDefaultValue, &P.z);
00013         GetRosParameter(pnh, "Gains/Pyaw", PyawDefaultValue, &P.yaw);
00014         GetRosParameter(pnh, "Reference_Gains/X", RGxDefaultValue, &RG.x);
00015         GetRosParameter(pnh, "Reference_Gains/Y", RGyDefaultValue, &RG.y);
00016         GetRosParameter(pnh, "Reference_Gains/Z", RGzDefaultValue, &RG.z);
00017         GetRosParameter(pnh, "Reference_Gains/Yaw", RGyawDefaultValue, &RG.yaw);
00018         GetRosParameter(pnh, std::string("Normalize/Max_Horizontal_Speed"),
00019             MAX_HORIZONTAL_SPEED, &norm.horizontal);
00020         GetRosParameter(pnh, std::string("Normalize/Max_Vertical_Speed"),
00021             MAX_VERTICAL_SPEED, &norm.vertical);
00022         GetRosParameter(pnh, std::string("Normalize/Max_Rotation_Speed"),
00023             MAX_ROTATION_SPEED, &norm.rotation);
00024     }
00025
00026     void SquareRootController::CalculateCommandVelocities(geometry_msgs::Twist& ref_command_signals) {
00027         if (!controller_active_) {
00028             ref_command_signals.linear.x = 0.0;
00029             ref_command_signals.linear.y = 0.0;
00030             ref_command_signals.linear.z = 0.0;
00031             ref_command_signals.angular.z = 0.0;
00032             return;
00033         }
00034         Vector4 u, e;
00035         GetErrors(e);
00036
00037         Vector4 Ppos;
00038         Ppos.x = P.x*sqrt(std::abs(e.x));
00039         Ppos.y = P.y*sqrt(std::abs(e.y));
```

```

00040         Ppos.z = P.z*sqrt(std::abs(e.z));
00041         Ppos.yaw = P.yaw*sqrt(std::abs(e.yaw));
00042
00043         CalculateLeashLength(e, P);
00044         LimitPositionErrors(e);
00045
00046         u.x = (-Ppos.x*e.x + RG.x*command_trajectory_.velocity_W[0])/norm.horizontal;
00047         u.y = (-Ppos.y*e.y + RG.y*command_trajectory_.velocity_W[1])/norm.horizontal;
00048         u.z = (-Ppos.z*e.z + RG.z*command_trajectory_.velocity_W[2])/norm.vertical;
00049         u.yaw = (-Ppos.yaw*e.yaw +
RG.yaw*command_trajectory_.angular_velocity_W[2])*RAD_2_DEG/norm.rotation;
00050
00051         ref_command_signals.linear.x = clamp(cos(state.orientation.z)*u.x -
sin(state.orientation.z)*u.y,max_speed.x);
00052         ref_command_signals.linear.y = clamp(sin(state.orientation.z)*u.x +
cos(state.orientation.z)*u.y,max_speed.y);
00053         ref_command_signals.linear.z = clamp(u.z,max_speed.z);
00054         ref_command_signals.angular.z = clamp(u.yaw,max_speed.yaw);
00055     }
00056 }
00057 }
00058
00059 int main(int argc, char** argv){
00060     ros::init(argc, argv, "square_root_controller_node");
00061     ros::NodeHandle nh;
00062     bebop_controller::SquareRootController square_root_controller;
00063     ros::spin();
00064     return 0;
00065 }

```

12.88 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/square_root_controller.h File Reference

Header file for square root controller.

```
#include "bebop_controller/base_controller.h"
```

Classes

- struct [bebop_controller::Normalize](#)
- class [bebop_controller::SquareRootController](#)

Namespaces

- namespace [bebop_controller](#)
Namespace containing all the classes and functions of the Bebop Controller.

Macros

- #define [PxDefaultValue](#) 1.0
- #define [PyDefaultValue](#) 1.0
- #define [PzDefaultValue](#) 1.0
- #define [PyawDefaultValue](#) 1.0
- #define [RGxDefaultValue](#) 0.0
- #define [RGyDefaultValue](#) 0.0
- #define [RGzDefaultValue](#) 0.0
- #define [RGyawDefaultValue](#) 0.0
- #define [MAX_HORIZONTAL_SPEED](#) 17.0
- #define [MAX_VERTICAL_SPEED](#) 1.0
- #define [MAX_ROTATION_SPEED](#) 100.0

12.88.1 Detailed Description

Header file for square root controller.

Definition in file [square_root_controller.h](#).

12.88.2 Macro Definition Documentation

12.88.2.1 MAX_HORIZONTAL_SPEED

```
#define MAX_HORIZONTAL_SPEED 17.0
```

Definition at line 16 of file [square_root_controller.h](#).

12.88.2.2 MAX_ROTATION_SPEED

```
#define MAX_ROTATION_SPEED 100.0
```

Definition at line 18 of file [square_root_controller.h](#).

12.88.2.3 MAX_VERTICAL_SPEED

```
#define MAX_VERTICAL_SPEED 1.0
```

Definition at line 17 of file [square_root_controller.h](#).

12.88.2.4 PxDefaultValue

```
#define PxDefaultValue 1.0
```

Definition at line 6 of file [square_root_controller.h](#).

12.88.2.5 PyawDefaultValue

```
#define PyawDefaultValue 1.0
```

Definition at line 9 of file [square_root_controller.h](#).

12.88.2.6 PyDefaultValue

```
#define PyDefaultValue 1.0
```

Definition at line 7 of file [square_root_controller.h](#).

12.88.2.7 PzDefaultValue

```
#define PzDefaultValue 1.0
```

Definition at line 8 of file [square_root_controller.h](#).

12.88.2.8 RGxDefaultValue

```
#define RGxDefaultValue 0.0
```

Definition at line 11 of file [square_root_controller.h](#).

12.88.2.9 RGyawDefaultValue

```
#define RGyawDefaultValue 0.0
```

Definition at line 14 of file [square_root_controller.h](#).

12.88.2.10 RGyDefaultValue

```
#define RGyDefaultValue 0.0
```

Definition at line 12 of file [square_root_controller.h](#).

12.88.2.11 RGzDefaultValue

```
#define RGzDefaultValue 0.0
```

Definition at line 13 of file [square_root_controller.h](#).

12.89 square_root_controller.h

[Go to the documentation of this file.](#)

```
00001
00003
00004 #include "bebop_controller/base_controller.h"
00005
00006 #define PxDefaultValue 1.0
00007 #define PyDefaultValue 1.0
00008 #define PzDefaultValue 1.0
00009 #define PyawDefaultValue 1.0
00010
00011 #define RGxDefaultValue 0.0
00012 #define RGyDefaultValue 0.0
00013 #define RGzDefaultValue 0.0
00014 #define RgyawDefaultValue 0.0
00015
00016 #define MAX_HORIZONTAL_SPEED 17.0
00017 #define MAX_VERTICAL_SPEED 1.0
00018 #define MAX_ROTATION_SPEED 100.0
00019
00020 namespace bebop_controller {
00021
00022     struct Normalize {
00023         double horizontal;
00024         double vertical;
00025         double rotation;
00026     };
00027
00028     class SquareRootController : public BaseController {
00029     public:
00030         SquareRootController();
00031
00032     private:
00033         Vector4 P, RG;
00034         Normalize norm;
00035         void CalculateCommandVelocities(geometry_msgs::Twist& ref_command_signals) override;
00036     };
00037
00038 }
```

Index

- `__init__`
 - `gazebo.SphinxPublisher`, [45](#)
 - `plot.Plots`, [42](#)
 - `_n1`
 - `bebop_controller::Waypoint`, [54](#)
 - `_n2`
 - `bebop_controller::Waypoint`, [55](#)
 - `_n3`
 - `bebop_controller::Waypoint`, [55](#)
 - `_timer1`
 - `bebop_controller::Waypoint`, [55](#)
 - `_timer2`
 - `bebop_controller::Waypoint`, [55](#)
 - `_timer3`
 - `bebop_controller::Waypoint`, [55](#)
 - `~BaseController`
 - `bebop_controller::BaseController`, [26](#)
 - `~DataToCSV`
 - `bebop_controller::DataToCSV`, [37](#)
 - `~Waypoint`
 - `bebop_controller::Waypoint`, [54](#)
- `acceleration`
 - `bebop_controller::State`, [48](#)
- `AfterTrajectory`
 - `waypoint.h`, [68](#)
- `angle`
 - `bebop_controller::Normalize`, [39](#)
- `angular_acceleration`
 - `bebop_controller::State`, [48](#)
- `angular_velocity`
 - `bebop_controller::State`, [48](#)
- `att`
 - `gazebo.SphinxPublisher`, [46](#)
- `base_controller.h`
 - `BEBOP_CMD_VEL`, [60](#)
 - `BEBOP_COMMAND_TRAJECTORY`, [60](#)
 - `BEBOP_LAND`, [60](#)
 - `BEBOP_POSE`, [61](#)
 - `BEBOP_TAKEOFF`, [61](#)
 - `CSV_END`, [61](#)
 - `DISABLE_COMMANDS`, [61](#)
 - `GRAVITY`, [61](#)
 - `LIMIT_X`, [61](#)
 - `LIMIT_Y`, [62](#)
 - `LIMIT_YAW`, [62](#)
 - `LIMIT_Z`, [62](#)
 - `MIN_ACCEL`, [62](#)
 - `MIN_VEL`, [62](#)
 - `SAFE_ZONE_X`, [62](#)
 - `SAFE_ZONE_Y`, [63](#)
 - `SAFE_ZONE_Z`, [63](#)
- `BaseController`
 - `bebop_controller::BaseController`, [26](#)
- `BEBOP_CMD_VEL`
 - `base_controller.h`, [60](#)
- `BEBOP_COMMAND_TRAJECTORY`
 - `base_controller.h`, [60](#)
- `bebop_controller`, [19](#)
 - `bound`, [20](#)
 - `clamp`, [20](#)
 - `GetRosParameter`, [21](#)
 - `max`, [21](#)
 - `Quat2RPY`, [21](#)
 - `RPY2Quat`, [22](#)
 - `sgn`, [22](#)
 - `yawFromQuaternion`, [22](#)
- `bebop_controller::BaseController`, [25](#)
 - `~BaseController`, [26](#)
 - `BaseController`, [26](#)
 - `CalculateCommandVelocities`, [27](#)
 - `CalculateLeashLength`, [27](#)
 - `CheckSafeZone`, [27](#)
 - `cmd_multi_dof_joint_trajectory_sub_`, [30](#)
 - `command_trajectory_`, [30](#)
 - `controller_active_`, [30](#)
 - `diff`, [30](#)
 - `disable_commands`, [30](#)
 - `end_sub_`, [30](#)
 - `EstimateAcceleration`, [27](#)
 - `EstimateVelocity`, [27](#)
 - `GetErrors`, [27](#)
 - `GetVelocityErrors`, [28](#)
 - `Land`, [28](#)
 - `land_pub_`, [31](#)
 - `last_state`, [31](#)
 - `lastTime`, [31](#)
 - `leash_length`, [31](#)
 - `LimitPositionErrors`, [28](#)
 - `max_speed`, [31](#)
 - `motor_velocity_reference_pub_`, [31](#)
 - `MultiDOFJointTrajectory_CB`, [28](#)
 - `odom_sub_`, [32](#)
 - `odometry_`, [32](#)
 - `Odometry_CB`, [28](#)
 - `odometry_filtered_pub_`, [32](#)
 - `Quaternion2Euler`, [28](#)
 - `reference_angles_pub_`, [32](#)

- safe_zone, 32
- SetOdometry, 29
- SetTrajectoryPoint, 29
- smoothed_reference_pub_, 32
- state, 33
- Stop, 29
- stop, 33
- Stop_CB, 29
- TakeOff, 29
- takeoff, 33
- takeoff_pub_, 33
- timeOut, 33
- TimeOut_CB, 29
- waypointHasBeenPublished_, 33
- bebop_controller::CITCController, 34
 - CITCController, 34
- bebop_controller::Command_Velocities, 35
 - x, 35
 - y, 35
 - yaw, 36
 - z, 36
- bebop_controller::DataToCSV, 36
 - ~DataToCSV, 37
 - DataToCSV, 36
- bebop_controller::DataToCSVParameters, 37
 - File, 37
 - Initial_Time, 38
 - Margin_Time, 38
 - Topic_CMD_Vel, 38
 - Topic_CSV_Begin, 38
 - Topic_CSV_End, 38
 - Topic_Drone_Pose, 38
 - Topic_Reference_Pose, 39
 - Topic_Velocities, 39
- bebop_controller::Normalize, 39
 - angle, 39
 - horizontal, 40
 - rotation, 40
 - vertical, 40
- bebop_controller::PIDController, 40
 - PIDController, 41
- bebop_controller::ProportionalController, 43
 - ProportionalController, 43
- bebop_controller::Sinusoidal, 44
 - Sinusoidal, 44
- bebop_controller::SquareRootController, 47
 - SquareRootController, 47
- bebop_controller::State, 48
 - acceleration, 48
 - angular_acceleration, 48
 - angular_velocity, 48
 - orientation, 48
 - position, 49
 - velocity, 49
- bebop_controller::Vector3, 50
 - x, 51
 - y, 51
 - z, 51
- bebop_controller::Vector4, 51
 - x, 52
 - y, 52
 - yaw, 52
 - z, 52
- bebop_controller::Waypoint, 53
 - _n1, 54
 - _n2, 55
 - _n3, 55
 - _timer1, 55
 - _timer2, 55
 - _timer3, 55
 - ~Waypoint, 54
 - Begin_, 55
 - End_, 56
 - Initial_Time, 56
 - nh, 56
 - position_target_, 56
 - setpoint_pub_, 56
 - Start_CB, 54
 - status, 56
 - Stop_CB, 54
 - Trajectory_CB, 54
 - Waypoint, 53
 - wp_params_, 57
- BEBOP_LAND
 - base_controller.h, 60
- BEBOP_POSE
 - base_controller.h, 61
- BEBOP_TAKEOFF
 - base_controller.h, 61
- BeforeTrajectory
 - waypoint.h, 68
- Begin_
 - bebop_controller::Waypoint, 55
- bound
 - bebop_controller, 20
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/incl
 - 59, 63
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/incl
 - 64, 66
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/incl
 - 67, 68
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/laun
 - 69
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/laun
 - 70
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/laun
 - 71
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/laun
 - 71, 72
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/laun
 - 72
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/laun
 - 73
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/laun
 - 74

- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_controller.launch, 131, 134
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_angles.yaml, 135
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_wispy.yaml, 137
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_controller.yaml, 137, 138
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_initialize_twist.yaml, 139, 142
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_initialize_twist.yaml, bebop_controller::BaseController, 27
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_pid_controller_angles.yaml, bebop_controller::BaseController, 27
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_pid_controller_twist.yaml, bebop_controller::BaseController, 27
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_proportional_controller.yaml, main, 91
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_safe_zone.yaml, K1xDefaultValue, 94
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_root_controller.yaml, K1yDefaultValue, 94
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_topics.yaml, K2xDefaultValue, 95
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_trajectory.yaml, K2yDefaultValue, 95
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_waypoint.yaml, K3xDefaultValue, 95
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_waypoint.yaml, K3yDefaultValue, 96
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_waypoint.yaml, K4xDefaultValue, 96
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_waypoint.yaml, K4yDefaultValue, 96
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_waypoint.yaml, LambdaX, 97
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_waypoint.yaml, LambdaZ, 97
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_waypoint.yaml, MAX_ROTATION_SPEED, 97
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_waypoint.yaml, MAX_VERTICAL_SPEED, 98
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_waypoint.yaml, RGYawDefaultValue, 98
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_waypoint.yaml, RGzDefaultValue, 98
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_waypoint.yaml, citc_controller_twist.cpp
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_waypoint.yaml, citc_controller_twist.h
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_waypoint.yaml, K1yawDefaultValue, 103
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_waypoint.yaml, K1zDefaultValue, 104
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_waypoint.yaml, K2yawDefaultValue, 104
- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/launch/bebop_waypoint.yaml, K2zDefaultValue, 104

- citc_controller_twist.h, 103
- K1yDefaultValue
 - citc_controller_angles.h, 94
 - citc_controller_twist.h, 104
- K1zDefaultValue
 - citc_controller_angles.h, 94
 - citc_controller_twist.h, 104
- K2xDefaultValue
 - citc_controller_angles.h, 95
 - citc_controller_twist.h, 104
- K2yawDefaultValue
 - citc_controller_angles.h, 95
 - citc_controller_twist.h, 104
- K2yDefaultValue
 - citc_controller_angles.h, 95
 - citc_controller_twist.h, 104
- K2zDefaultValue
 - citc_controller_angles.h, 95
 - citc_controller_twist.h, 104
- K3xDefaultValue
 - citc_controller_angles.h, 95
 - citc_controller_twist.h, 105
- K3yawDefaultValue
 - citc_controller_angles.h, 95
 - citc_controller_twist.h, 105
- K3yDefaultValue
 - citc_controller_angles.h, 96
 - citc_controller_twist.h, 105
- K3zDefaultValue
 - citc_controller_angles.h, 96
 - citc_controller_twist.h, 105
- K4xDefaultValue
 - citc_controller_angles.h, 96
 - citc_controller_twist.h, 105
- K4yawDefaultValue
 - citc_controller_angles.h, 96
 - citc_controller_twist.h, 105
- K4yDefaultValue
 - citc_controller_angles.h, 96
 - citc_controller_twist.h, 106
- K4zDefaultValue
 - citc_controller_angles.h, 96
 - citc_controller_twist.h, 106
- LambdaX
 - citc_controller_angles.h, 97
 - citc_controller_twist.h, 106
 - pid_controller_angles.h, 117
 - pid_controller_twist.h, 126
- LambdaY
 - citc_controller_angles.h, 97
 - citc_controller_twist.h, 106
 - pid_controller_angles.h, 117
 - pid_controller_twist.h, 126
- LambdaZ
 - citc_controller_angles.h, 97
 - citc_controller_twist.h, 106
 - pid_controller_angles.h, 118
 - pid_controller_twist.h, 126
- Land
 - bebop_controller::BaseController, 28
- land_pub_
 - bebop_controller::BaseController, 31
- last_state
 - bebop_controller::BaseController, 31
- lastTime
 - bebop_controller::BaseController, 31
- leash_length
 - bebop_controller::BaseController, 31
- LIMIT_X
 - base_controller.h, 61
- LIMIT_Y
 - base_controller.h, 62
- LIMIT_YAW
 - base_controller.h, 62
- LIMIT_Z
 - base_controller.h, 62
- LimitPositionErrors
 - bebop_controller::BaseController, 28
- LlxDefaultValue
 - pid_controller_angles.h, 118
 - pid_controller_twist.h, 126
- LlyawDefaultValue
 - pid_controller_angles.h, 118
 - pid_controller_twist.h, 126
- LlyDefaultValue
 - pid_controller_angles.h, 118
 - pid_controller_twist.h, 127
- LlzDefaultValue
 - pid_controller_angles.h, 118
 - pid_controller_twist.h, 127
- main
 - citc_controller_angles.cpp, 91
 - citc_controller_twist.cpp, 100
 - data_to_csv.cpp, 109
 - pid_controller_angles.cpp, 113
 - pid_controller_twist.cpp, 122
 - proportional_controller.cpp, 130
 - sinusoidal.cpp, 135
 - square_root_controller.cpp, 138
- Margin_Time
 - bebop_controller::DataToCSVParameters, 38
- MarginTime
 - WaypointParameters, 57
- MASS
 - citc_controller_angles.h, 97
 - citc_controller_twist.h, 106
 - pid_controller_angles.h, 118
 - pid_controller_twist.h, 127
- max
 - bebop_controller, 21
- MAX_HORIZONTAL_SPEED
 - citc_controller_twist.h, 107
 - pid_controller_twist.h, 127
 - proportional_controller.h, 132
 - square_root_controller.h, 140
- MAX_ROTATION_SPEED

- citc_controller_angles.h, 97
 - citc_controller_twist.h, 107
 - pid_controller_angles.h, 119
 - pid_controller_twist.h, 127
 - proportional_controller.h, 132
 - square_root_controller.h, 140
- max_speed
 - bebop_controller::BaseController, 31
- MAX_TILT_ANGLE
 - citc_controller_angles.h, 97
 - pid_controller_angles.h, 119
- MAX_VERTICAL_SPEED
 - citc_controller_angles.h, 98
 - citc_controller_twist.h, 107
 - pid_controller_angles.h, 119
 - pid_controller_twist.h, 127
 - proportional_controller.h, 132
 - square_root_controller.h, 140
- MIN_ACCEL
 - base_controller.h, 62
- MIN_VEL
 - base_controller.h, 62
- motor_velocity_reference_pub_
 - bebop_controller::BaseController, 31
- MultiDOFJointTrajectory_CB
 - bebop_controller::BaseController, 28
- nh
 - bebop_controller::Waypoint, 56
- odom_pub
 - gazebo.SphinxPublisher, 46
- odom_sub_
 - bebop_controller::BaseController, 32
- odometry_
 - bebop_controller::BaseController, 32
- Odometry_CB
 - bebop_controller::BaseController, 28
- odometry_filtered_pub_
 - bebop_controller::BaseController, 32
- orientation
 - bebop_controller::State, 48
- path
 - plot, 23
 - plot.Plots, 42
- pid_controller_angles.cpp
 - main, 113
- pid_controller_angles.h
 - DxDefaultValue, 116
 - DyawDefaultValue, 116
 - DyDefaultValue, 116
 - DzDefaultValue, 116
 - IxDefaultValue, 117
 - IyawDefaultValue, 117
 - IyDefaultValue, 117
 - IzDefaultValue, 117
 - LambdaX, 117
 - LambdaY, 117
- LambdaZ, 118
- LlxDDefaultValue, 118
- LlyawDefaultValue, 118
- LlyDefaultValue, 118
- LlzDefaultValue, 118
- MASS, 118
- MAX_ROTATION_SPEED, 119
- MAX_TILT_ANGLE, 119
- MAX_VERTICAL_SPEED, 119
- PxDDefaultValue, 119
- PyawDefaultValue, 119
- PyDefaultValue, 119
- PzDefaultValue, 120
- RGxDefaultValue, 120
- RGyawDefaultValue, 120
- RGyDefaultValue, 120
- RGzDefaultValue, 120
- pid_controller_twist.cpp
 - main, 122
- pid_controller_twist.h
 - DxDefaultValue, 124
 - DyawDefaultValue, 125
 - DyDefaultValue, 125
 - DzDefaultValue, 125
 - IxDefaultValue, 125
 - IyawDefaultValue, 125
 - IyDefaultValue, 125
 - IzDefaultValue, 126
 - LambdaX, 126
 - LambdaY, 126
 - LambdaZ, 126
 - LlxDDefaultValue, 126
 - LlyawDefaultValue, 126
 - LlyDefaultValue, 127
 - LlzDefaultValue, 127
 - MASS, 127
 - MAX_HORIZONTAL_SPEED, 127
 - MAX_ROTATION_SPEED, 127
 - MAX_VERTICAL_SPEED, 127
 - PxDDefaultValue, 128
 - PyawDefaultValue, 128
 - PyDefaultValue, 128
 - PzDefaultValue, 128
 - RGxDefaultValue, 128
 - RGyawDefaultValue, 128
 - RGyDefaultValue, 129
 - RGzDefaultValue, 129
- PIDController
 - bebop_controller::PIDController, 41
- Plot
 - plot.Plots, 42
- plot, 23
 - path, 23
 - plt, 23
 - sub, 24
 - topic, 24
 - yaml, 24
- plot.Plots, 41

- `__init__`, 42
 - `path`, 42
 - `Plot`, 42
 - `yaml`, 42
- `plt`
- `plot`, 23
- `pos`
 - `gazebo.SphinxPublisher`, 46
- `position`
 - `bebop_controller::State`, 49
- `position_target`
 - `bebop_controller::Waypoint`, 56
- `PositionBeforeTrajectory`
 - `TrajectoryParameters`, 49
- `process_output`
 - `gazebo.SphinxPublisher`, 45
- `proportional_controller.cpp`
 - `main`, 130
- `proportional_controller.h`
 - `MAX_HORIZONTAL_SPEED`, 132
 - `MAX_ROTATION_SPEED`, 132
 - `MAX_VERTICAL_SPEED`, 132
 - `PxDefaultValue`, 133
 - `PyawDefaultValue`, 133
 - `PyDefaultValue`, 133
 - `PzDefaultValue`, 133
 - `RGxDefaultValue`, 133
 - `RGyawDefaultValue`, 133
 - `RGyDefaultValue`, 134
 - `RGzDefaultValue`, 134
- `ProportionalController`
 - `bebop_controller::ProportionalController`, 43
- `publish`
 - `gazebo.SphinxPublisher`, 45
- `PxDefaultValue`
 - `pid_controller_angles.h`, 119
 - `pid_controller_twist.h`, 128
 - `proportional_controller.h`, 133
 - `square_root_controller.h`, 140
- `PyawDefaultValue`
 - `pid_controller_angles.h`, 119
 - `pid_controller_twist.h`, 128
 - `proportional_controller.h`, 133
 - `square_root_controller.h`, 140
- `PyDefaultValue`
 - `pid_controller_angles.h`, 119
 - `pid_controller_twist.h`, 128
 - `proportional_controller.h`, 133
 - `square_root_controller.h`, 140
- `PzDefaultValue`
 - `pid_controller_angles.h`, 120
 - `pid_controller_twist.h`, 128
 - `proportional_controller.h`, 133
 - `square_root_controller.h`, 141
- `q`
 - `gazebo.SphinxPublisher`, 46
- `Quat2RPY`
 - `bebop_controller`, 21
- `Quaternion2Euler`
 - `bebop_controller::BaseController`, 28
- `reference_angles_pub`
 - `bebop_controller::BaseController`, 32
- `RGxDefaultValue`
 - `citc_controller_angles.h`, 98
 - `citc_controller_twist.h`, 107
 - `pid_controller_angles.h`, 120
 - `pid_controller_twist.h`, 128
 - `proportional_controller.h`, 133
 - `square_root_controller.h`, 141
- `RGyawDefaultValue`
 - `citc_controller_angles.h`, 98
 - `citc_controller_twist.h`, 107
 - `pid_controller_angles.h`, 120
 - `pid_controller_twist.h`, 128
 - `proportional_controller.h`, 133
 - `square_root_controller.h`, 141
- `RGyDefaultValue`
 - `citc_controller_angles.h`, 98
 - `citc_controller_twist.h`, 107
 - `pid_controller_angles.h`, 120
 - `pid_controller_twist.h`, 129
 - `proportional_controller.h`, 134
 - `square_root_controller.h`, 141
- `RGzDefaultValue`
 - `citc_controller_angles.h`, 98
 - `citc_controller_twist.h`, 108
 - `pid_controller_angles.h`, 120
 - `pid_controller_twist.h`, 129
 - `proportional_controller.h`, 134
 - `square_root_controller.h`, 141
- `rotation`
 - `bebop_controller::Normalize`, 40
- `RPY2Quat`
 - `bebop_controller`, 22
- `safe_zone`
 - `bebop_controller::BaseController`, 32
- `SAFE_ZONE_X`
 - `base_controller.h`, 62
- `SAFE_ZONE_Y`
 - `base_controller.h`, 63
- `SAFE_ZONE_Z`
 - `base_controller.h`, 63
- `secsFromHeaderMsg`
 - `common.h`, 65
- `SetOdometry`
 - `bebop_controller::BaseController`, 29
- `setpoint_pub`
 - `bebop_controller::Waypoint`, 56
- `SetTrajectoryPoint`
 - `bebop_controller::BaseController`, 29
- `sgn`
 - `bebop_controller`, 22
- `Sigma`
 - `citc_controller_angles.h`, 98
 - `citc_controller_twist.h`, 108

- Sinusoidal
 - bebop_controller::Sinusoidal, [44](#)
- sinusoidal.cpp
 - main, [135](#)
- smoothed_reference_pub_
 - bebop_controller::BaseController, [32](#)
- square_root_controller.cpp
 - main, [138](#)
- square_root_controller.h
 - MAX_HORIZONTAL_SPEED, [140](#)
 - MAX_ROTATION_SPEED, [140](#)
 - MAX_VERTICAL_SPEED, [140](#)
 - PxDefaultValue, [140](#)
 - PyDefaultValue, [140](#)
 - PzDefaultValue, [141](#)
 - RGxDefaultValue, [141](#)
 - RGyawDefaultValue, [141](#)
 - RGyDefaultValue, [141](#)
 - RGzDefaultValue, [141](#)
- SquareRootController
 - bebop_controller::SquareRootController, [47](#)
- Start_CB
 - bebop_controller::Waypoint, [54](#)
- state
 - bebop_controller::BaseController, [33](#)
- Status
 - waypoint.h, [68](#)
- status
 - bebop_controller::Waypoint, [56](#)
- Stop
 - bebop_controller::BaseController, [29](#)
- stop
 - bebop_controller::BaseController, [33](#)
- Stop_CB
 - bebop_controller::BaseController, [29](#)
 - bebop_controller::Waypoint, [54](#)
- sub
 - plot, [24](#)
- TakeOff
 - bebop_controller::BaseController, [29](#)
- takeoff
 - bebop_controller::BaseController, [33](#)
- takeoff_pub_
 - bebop_controller::BaseController, [33](#)
- TimeBeforeTrajectory
 - WaypointParameters, [58](#)
- timeOut
 - bebop_controller::BaseController, [33](#)
- TimeOut_CB
 - bebop_controller::BaseController, [29](#)
- topic
 - plot, [24](#)
- Topic_CMD_Vel
 - bebop_controller::DataToCSVParameters, [38](#)
- topic_command_trajectory
 - WaypointParameters, [58](#)
- Topic_CSV_Begin
 - bebop_controller::DataToCSVParameters, [38](#)
- topic_csv_begin
 - WaypointParameters, [58](#)
- Topic_CSV_End
 - bebop_controller::DataToCSVParameters, [38](#)
- topic_csv_end
 - WaypointParameters, [58](#)
- Topic_Drone_Pose
 - bebop_controller::DataToCSVParameters, [38](#)
- Topic_Reference_Pose
 - bebop_controller::DataToCSVParameters, [39](#)
- Topic_Velocities
 - bebop_controller::DataToCSVParameters, [39](#)
- Trajectory
 - waypoint.h, [68](#)
- Trajectory_CB
 - bebop_controller::Waypoint, [54](#)
- TrajectoryDistance
 - TrajectoryParameters, [50](#)
- TrajectoryParameters, [49](#)
 - PositionBeforeTrajectory, [49](#)
 - TrajectoryDistance, [50](#)
 - Yaw_Enabled, [50](#)
 - Yaw_Offset, [50](#)
- TrajectoryTime
 - WaypointParameters, [58](#)
- vector4FromQuaternionMsg
 - common.h, [65](#)
- velocity
 - bebop_controller::State, [49](#)
- vertical
 - bebop_controller::Normalize, [40](#)
- Waypoint
 - bebop_controller::Waypoint, [53](#)
- waypoint.h
 - AfterTrajectory, [68](#)
 - BeforeTrajectory, [68](#)
 - Status, [68](#)
 - Trajectory, [68](#)
- waypointHasBeenPublished_
 - bebop_controller::BaseController, [33](#)
- WaypointParameters, [57](#)
 - DiffTime, [57](#)
 - MarginTime, [57](#)
 - TimeBeforeTrajectory, [58](#)
 - topic_command_trajectory, [58](#)
 - topic_csv_begin, [58](#)
 - topic_csv_end, [58](#)
 - TrajectoryTime, [58](#)
- wp_params_
 - bebop_controller::Waypoint, [57](#)
- x
 - bebop_controller::Command_Velocities, [35](#)
 - bebop_controller::Vector3, [51](#)
 - bebop_controller::Vector4, [52](#)

y

- bebop_controller::Command_Velocities, [35](#)
- bebop_controller::Vector3, [51](#)
- bebop_controller::Vector4, [52](#)

yaml

- plot, [24](#)
- plot.Plots, [42](#)

yaw

- bebop_controller::Command_Velocities, [36](#)
- bebop_controller::Vector4, [52](#)

Yaw_Enabled

- TrajectoryParameters, [50](#)

Yaw_Offset

- TrajectoryParameters, [50](#)

yawFromQuaternion

- bebop_controller, [22](#)

z

- bebop_controller::Command_Velocities, [36](#)
- bebop_controller::Vector3, [51](#)
- bebop_controller::Vector4, [52](#)