Bebop Controller Package

Generated by Doxygen 1.9.3

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 *rosrun* 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:

Chapter 7

Hierarchical Index

7.1 Class Hierarchy

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

Chapter 8

Class Index

8.1 Class List

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

Chapter 9

File Index

9.1 File List

Here is a list of all files with brief descriptions:

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

value	Value to limit.
min	Minimum value.
max	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

value	Value to limit.
lim	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

val1	Value 1 to compare.
val2	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

value	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.

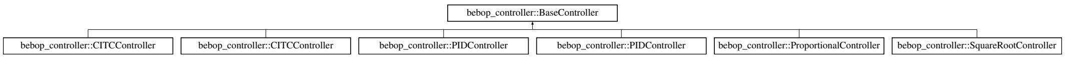
Chapter 11

Class Documentation

11.1 bebop_controller::BaseController Class Reference

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

Inheritance diagram for bebop_controller::BaseController:


bebop_controller::BaseController
bebop_controller::CITCController
bebop_controller::CITCController
bebop_controller::PIDController
bebop_controller::PIDController
bebop_controller::ProportionalController
bebop_controller::SquareRootController

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.

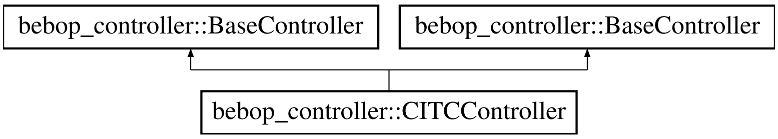
The documentation for this class was generated from the following files:

- 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:


bebop_controller::BaseController
bebop_controller::BaseController
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 citc_controller_angles.h.

11.6.2.2 horizontal

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

Definition at line 45 of file citc_controller_twist.h.

11.6.2.3 rotation

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

Definition at line 47 of file citc_controller_angles.h.

11.6.2.4 vertical

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

Definition at line 46 of file citc_controller_angles.h.

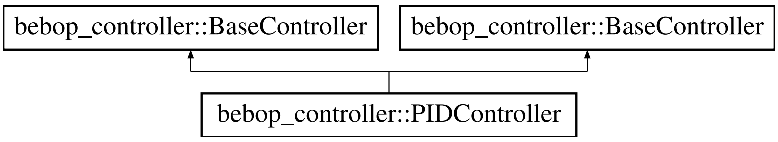
The documentation for this struct was generated from the following files:

- C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/src/nodes/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/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:


bebop_controller::BaseController
bebop_controller::BaseController
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.

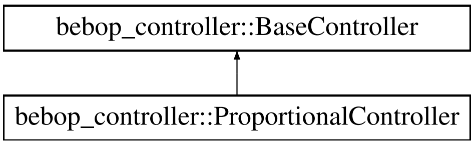
The documentation for this class was generated from the following file:

- 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:


bebop_controller::BaseController
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.

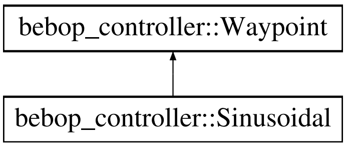
The documentation for this class was generated from the following files:

- 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.

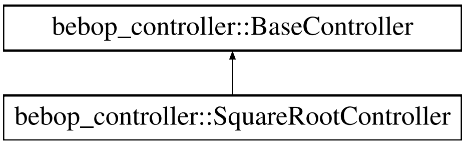
The documentation for this class was generated from the following file:

- 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:


bebop_controller::BaseController
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.

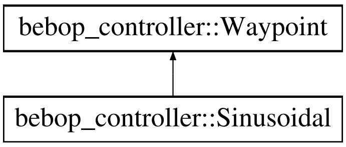
The documentation for this struct was generated from the following file:

- 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_;
```

```

            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;

            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);
    };

}
```

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]
```

Funtion 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.

```


#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>

static const float DEG_2_RAD = M_PI / 180.0;
static const float RAD_2_DEG = 180.0 / M_PI;

namespace bebop_controller {

    double bound(double value, double min, double max){
        if (value < min){
            return min;
        }
        else if(value > max){
            return max;
        }
        else {
            return value;
        }
    }

    double clamp(double value, double lim) {
        return bound(value, -std::abs(lim), std::abs(lim));
    }

    double sgn(double value){
        if (value > 0){
            return 1;
        }
        else if (value < 0){
            return -1;
        }
        else {
            return 0;
        }
    }

    double max(double val1, double val2){
        if (val1 > val2) {
            return val1;
        }
        else {
            return val2;
        }
    }

    struct Vector3 {
        double x;
        double y;
        double z;
    };

    struct Vector4 {
        double x;
        double y;
        double z;
        double yaw;
    };

    struct State {
        Vector3 position;
        Vector3 orientation;
        Vector3 velocity;
        Vector3 angular_velocity;
        Vector3 acceleration;
        Vector3 angular_acceleration;
    };

    struct Command_Velocities {
        double x;
        double y;
        double z;
        double yaw;
    };

    template<typename T> inline void GetRosParameter(const ros::NodeHandle& nh,
                                                    const std::string& key,
                                                    const T& default_value,
                                                    T* value) {
        ROS_ASSERT(value != nullptr);
        bool have_parameter = nh.getParam(key, *value);
        if (!have_parameter) {
            ROS_WARN_STREAM("[rosparam]: could not find parameter " « nh.getNamespace()
                        « "/" « key « ", setting to default: " « default_value);
            *value = default_value;
        }
    }

    inline double yawFromQuaternion(const Eigen::Quaterniond& q) {
        return atan2(2.0 * (q.w() * q.z() + q.x() * q.y()),
                    1.0 - 2.0 * (q.y() * q.y() + q.z() * q.z()));
    }

    Eigen::Vector3d Quat2RPY(Eigen::Vector4d& Quaternion) {
        double roll, pitch, yaw;
        tf::Quaternion q(Quaternion.x(), Quaternion.y(), Quaternion.z(), Quaternion.w());
        tf::Matrix3x3 m(q);
        m.getRPY(roll, pitch, yaw);
        return Eigen::Vector3d(roll, pitch, yaw);
    }

    Eigen::Vector4d RPY2Quat(Eigen::Vector3d& RPY) {
        tf::Quaternion q;
        tf::Matrix3x3 m;
        m.setEulerYPR(RPY.z(), RPY.y(), RPY.x());
        m.getRotation(q);
        return Eigen::Vector4d(q.x(), q.y(), q.z(), q.w());
    }

}

inline Eigen::Vector4d vector4FromQuaternionMsg(const geometry_msgs::Quaternion& msg) {
    return Eigen::Vector4d(msg.x, msg.y, msg.z, msg.w);
}

inline double secsFromHeaderMsg(const std_msgs::Header& msg) {
    return (double) (msg.stamp.sec) + (double) (msg.stamp.nsec)/1.0e9;
}
```

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

```
    double TrajectoryTime;
    double DiffTime;
    double MarginTime;
    std::string topic_command_trajectory;
    std::string topic_csv_begin;
    std::string topic_csv_end;
};

enum Status {
    BeforeTrajectory,
    Trajectory,
    AfterTrajectory,
};

namespace bebop_controller {

    class Waypoint{
        public:
            Waypoint(WaypointParameters param);
            ~Waypoint();

        protected:
            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_;

            void Start_CB(const ros::TimerEvent& event);
            void Stop_CB(const ros::TimerEvent& event);
            virtual void Trajectory_CB(const ros::TimerEvent& event);

            enum Status status;
    };

}
```

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.

```
<?xml version="1.0"?>

<launch>
    <arg name="Dir" default="$(env HOME)/CSV/"/>
    <param name="Subfolder" type="str" command="date +'%d-%m-%Y_%Ih%Mm%Ss'"/>
    <arg name="GazeboRealTime" default="0.5"/>
    <arg name="YAML" default="citc_controller_angles.yaml"/>

    <node name="gazebo" pkg="bebop_controller" type="gazebo.py" output="screen">
      <param name="Topic" type="str" value="/bebop/pose"/>
      <param name="GazeboRealTime" type="double" value="$(arg GazeboRealTime)"/>
    </node>

    <node name="citc_controller_angles" pkg="bebop_controller" type="citc_controller_angles" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/citc_controller_angles.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/normalize_angles.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/safe_zone.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/max_speed.yaml" />
    </node>

    <node name="waypoint" pkg="bebop_controller" type="sinusoidal" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/trajectory.yaml" />
    </node>

    <node name="data_to_csv" pkg="bebop_controller" type="data_to_csv" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
      <param name="Dir" type="str" value="$(arg Dir)"/>
    </node>

    <node name="plot" pkg="bebop_controller" type="plot.py" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <param name="Dir" type="str" value="$(arg Dir)"/>
      <param name="YAML" type="str" value="$(arg YAML)"/>
    </node>

</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.

```
<?xml version="1.0"?>

<launch>
    <arg name="Dir" default="$(env HOME)/CSV/"/>
    <param name="Subfolder" type="str" command="date +’%d-%m-%Y_%Ih%Mm%Ss’"/>
    <arg name="GazeboRealTime" default="0.5"/>
    <arg name="YAML" default="citc_controller_twist.yaml"/>

    <node name="gazebo" pkg="bebop_controller" type="gazebo.py" output="screen">
      <param name="Topic" type="str" value="/bebop/pose"/>
      <param name="GazeboRealTime" type="double" value="$(arg GazeboRealTime)"/>
    </node>

    <node name="citc_controller_twist" pkg="bebop_controller" type="citc_controller_twist" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/citc_controller_twist.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/normalize_angles.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/safe_zone.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/max_speed.yaml" />
    </node>

    <node name="waypoint" pkg="bebop_controller" type="sinusoidal" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/trajectory.yaml" />
    </node>

    <node name="data_to_csv" pkg="bebop_controller" type="data_to_csv" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
      <param name="Dir" type="str" value="$(arg Dir)"/>
    </node>

    <node name="plot" pkg="bebop_controller" type="plot.py" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <param name="Dir" type="str" value="$(arg Dir)"/>
      <param name="YAML" type="str" value="$(arg YAML)"/>
    </node>

</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.

```
<?xml version="1.0"?>

<launch>
    <node name="gazebo" pkg="bebop_controller" type="gazebo.py" output="screen">
        <param name="Topic" type="str" value="/bebop/pose"/>
        <param name="GazeboRealTime" type="double" value="0.5"/>
    </node>
</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.

```
<?xml version="1.0"?>

<launch>
    <arg name="Dir" default="$(env HOME)/CSV/"/>
    <param name="Subfolder" type="str" command="date +’%d-%m-%Y_%Ih%Mm%Ss’"/>
    <arg name="GazeboRealTime" default="0.5"/>
    <arg name="YAML" default="pid_controller_angles.yaml"/>

    <node name="gazebo" pkg="bebop_controller" type="gazebo.py" output="screen">
      <param name="Topic" type="str" value="/bebop/pose"/>
      <param name="GazeboRealTime" type="double" value="$(arg GazeboRealTime)"/>
    </node>

    <node name="pid_controller_angles" pkg="bebop_controller" type="pid_controller_angles" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/pid_controller_angles.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/normalize_angles.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/safe_zone.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/max_speed.yaml" />
    </node>

    <node name="waypoint" pkg="bebop_controller" type="sinusoidal" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/trajectory.yaml" />
    </node>

    <node name="data_to_csv" pkg="bebop_controller" type="data_to_csv" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
      <param name="Dir" type="str" value="$(arg Dir)"/>
    </node>

    <node name="plot" pkg="bebop_controller" type="plot.py" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <param name="Dir" type="str" value="$(arg Dir)"/>
      <param name="YAML" type="str" value="$(arg YAML)"/>
    </node>

</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.

```
<?xml version="1.0"?>

<launch>
    <arg name="Dir" default="$(env HOME)/CSV/"/>
    <param name="Subfolder" type="str" command="date +’%d-%m-%Y_%Ih%Mm%Ss’"/>
    <arg name="GazeboRealTime" default="0.5"/>
    <arg name="YAML" default="pid_controller_twist.yaml"/>

    <node name="gazebo" pkg="bebop_controller" type="gazebo.py" output="screen">
      <param name="Topic" type="str" value="/bebop/pose"/>
      <param name="GazeboRealTime" type="double" value="$(arg GazeboRealTime)"/>
    </node>

    <node name="pid_controller_twist" pkg="bebop_controller" type="pid_controller_twist" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/pid_controller_twist.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/normalize_twist.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/safe_zone.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/max_speed.yaml" />
    </node>

    <node name="waypoint" pkg="bebop_controller" type="sinusoidal" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/trajectory.yaml" />
    </node>

    <node name="data_to_csv" pkg="bebop_controller" type="data_to_csv" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
      <param name="Dir" type="str" value="$(arg Dir)"/>
    </node>

    <node name="plot" pkg="bebop_controller" type="plot.py" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <param name="Dir" type="str" value="$(arg Dir)"/>
      <param name="YAML" type="str" value="$(arg YAML)"/>
    </node>

</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.

```
<?xml version="1.0"?>

<launch>
    <arg name="Dir" default="$(env HOME)/CSV/"/>
    <param name="Subfolder" type="str" command="date +’%d-%m-%Y_%Ih%Mm%Ss’"/>
    <arg name="GazeboRealTime" default="0.5"/>
    <arg name="YAML" default="proportional_controller.yaml"/>

    <node name="gazebo" pkg="bebop_controller" type="gazebo.py" output="screen">
      <param name="Topic" type="str" value="/bebop/pose"/>
      <param name="GazeboRealTime" type="double" value="$(arg GazeboRealTime)"/>
    </node>

    <node name="proportional_controller" pkg="bebop_controller" type="proportional_controller" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/proportional_controller.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/normalize_twist.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/safe_zone.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/max_speed.yaml" />
    </node>

    <node name="waypoint" pkg="bebop_controller" type="sinusoidal" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/trajectory.yaml" />
    </node>

    <node name="data_to_csv" pkg="bebop_controller" type="data_to_csv" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
      <param name="Dir" type="str" value="$(arg Dir)"/>
    </node>

    <node name="plot" pkg="bebop_controller" type="plot.py" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <param name="Dir" type="str" value="$(arg Dir)"/>
      <param name="YAML" type="str" value="$(arg YAML)"/>
    </node>

</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.

```
<?xml version="1.0"?>

<launch>
    <arg name="Dir" default="$(env HOME)/CSV/"/>
    <param name="Subfolder" type="str" command="date +'%d-%m-%Y_%Ih%Mm%Ss'"/>
    <arg name="GazeboRealTime" default="0.5"/>
    <arg name="YAML" default="square_root_controller.yaml"/>

    <node name="gazebo" pkg="bebop_controller" type="gazebo.py" output="screen">
      <param name="Topic" type="str" value="/bebop/pose"/>
      <param name="GazeboRealTime" type="double" value="$(arg GazeboRealTime)"/>
    </node>

    <node name="square_root_controller" pkg="bebop_controller" type="square_root_controller" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/square_root_controller.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/normalize_twist.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/safe_zone.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/max_speed.yaml" />
    </node>

    <node name="waypoint" pkg="bebop_controller" type="sinusoidal" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/trajectory.yaml" />
    </node>

    <node name="data_to_csv" pkg="bebop_controller" type="data_to_csv" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <rosparam command="load" file="$(find bebop_controller)/resource/waypoint.yaml" />
      <param name="Dir" type="str" value="$(arg Dir)"/>
    </node>

    <node name="plot" pkg="bebop_controller" type="plot.py" output="screen">
      <rosparam command="load" file="$(find bebop_controller)/resource/topics.yaml" />
      <param name="Dir" type="str" value="$(arg Dir)"/>
      <param name="YAML" type="str" value="$(arg YAML)"/>
    </node>

</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.

```
<?xml version="1.0"?>

<launch>
    <arg name="server" default="192.168.42.16"/>
    <node pkg="vrpn_client_ros" type="vrpn_client_node" name="vrpn_client_node" output="screen">
        <rosparam subst_value="true">
            server: $(arg server)
            port: 3883
            frame_id: world
            broadcast_tf: true
            refresh_tracker_frequency: 120.0
        </rosparam>
    </node>
</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

Gains/K1x	Ganancia K_1

Definition in file citc_controller_angles.yaml.

12.25 citc_controller_angles.yaml

Go to the documentation of this file.

```
# CITC controller parameters
Gains: {K1x: 0.9,
        K1y: 0.9,
        K1z: 5.0,
        K1yaw: 0.3,
        K2x: 1.2,
        K2y: 1.2,
        K2z: 3.0,
        K2yaw: 2.4,
        K3x: 0,
        K3y: 0,
        K3z: 0,
        K3yaw: 0,
        K4x: 0,
        K4y: 0,
        K4z: 0,
        K4yaw: 0}
Reference_Gains: {X: 0,
                  Y: 0,
                  Z: 0,
                  Yaw: 0}
Lambda: {X: 1.3151,
         Y: 1.3151,
         Z: 4.6697}
Mass: 0.5
Sigma: 0.8
Disable_Commands: False
```

12.26 C:/Users/franc/Desktop/Ubuntu/bebop_controller/src/bebop_controller/resource/citc_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 citc_controller_twist.yaml.

12.27 citc_controller_twist.yaml

Go to the documentation of this file.

```
# CITC controller parameters
Gains: {K1x: 2.05,
        K1y: 2.05,
        K1z: 0.01,
        K1yaw: 0.009172,
        K2x: 5.45,
        K2y: 5.45,
        K2z: 0.1,
        K2yaw: 0.036114,
        K3x: 0,
        K3y: 0,
        K3z: 0,
        K3yaw: 0,
        K4x: 0,
        K4y: 0,
        K4z: 0,
        K4yaw: 0}
Reference_Gains: {X: 0,
                  Y: 0,
                  Z: 0,
                  Yaw: 0}
Lambda: {X: 1.3151,
         Y: 1.3151,
         Z: 4.6697}
Mass: 0.5
Sigma: 0.8
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.

```
Max_Speed: {X: 0.3,
            Y: 0.3,
            Z: 0.3,
            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.

```
Normalize: {Max_Tilt_Angle: 20.0,
            Max_Vertical_Speed: 1.0,
            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.

```
Normalize: {Max_Horizontal_Speed: 9.5,
            Max_Vertical_Speed: 1.0,
            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.

```
# Proportional integral derivative controller parameters
Gains: {Px: 1.8,
        Py: 1.8,
        Pz: 5.0,
        Pyaw: 4.8,
        Dx: 1.4,
        Dy: 1.4,
        Dz: 7.2,
        Dyaw: 0.6,
        Ix: 0.3,
        Iy: 0.3,
        Iz: 1.0,
        Iyaw: 1.0}
Limits: {Ix: 3.0,
         Iy: 3.0,
         Iz: 1.0,
         Iyaw: 10.0}
Reference_Gains: {X: 1.0,
                  Y: 1.0,
                  Z: 1.0,
                  Yaw: 1.0}
Lambda: {X: 1.3151,
         Y: 1.3151,
         Z: 4.6697}
Mass: 0.5
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.

```
# Proportional integral derivative controller parameters
Gains: {Px: 0.9,
        Py: 0.9,
        Pz: 12.0,
        Pyaw: 0.0,
        Dx: 0.85,
        Dy: 0.85,
        Dz: 10.0,
        Dyaw: 0.0,
        Ix: 0.2,
        Iy: 0.2,
        Iz: 8.0,
        Iyaw: 0.0}
Limits: {Ix: 3.0,
         Iy: 3.0,
         Iz: 1.0,
         Iyaw: 10.0}
Reference_Gains: {X: 1.0,
                  Y: 1.0,
                  Z: 1.0,
                  Yaw: 1.0}
Lambda: {X: 1.3151,
         Y: 1.3151,
         Z: 4.6697}
Mass: 0.5
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.

```
# Proportional controller parameters
Gains: {Px: 1.2,
        Py: 1.2,
        Pz: 1.0,
        Pyaw: 1.0}
Reference_Gains: {X: 1.0,
                  Y: 1.0,
                  Z: 1.0,
                  Yaw: 1.0}
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.

```
Safe_Zone: {X: 1.6,
            Y: 0.77,
            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.

```
# Square root controller parameters
Gains: {Px: 1000.0,
        Py: 1000.0,
        Pz: 5.0,
        Pyaw: 3.0}
Reference_Gains: {X: 1.0,
                  Y: 1.0,
                  Z: 1.0,
                  Yaw: 1.0}
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.

```
# Topics
Topics: { Command_Trajectory: "/bebop/command/trajectory",
          Pose: "/vrpn_client_node/RigidBody1/pose",
          CMD_Vel: "/bebop/cmd_vel",
          TakeOff: "/bebop/takeoff",
          Land: "/bebop/land",
          CSV_Begin: "/csv/begin",
          CSV_End: "/csv/end",
          Velocities: "/bebop/odom"}
# /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.

```
Trajectory: { X_Initial: 0,
              Y_Initial: 0,
              Z_Initial: 1.4,
              X_Distance: 1.2,
              Y_Distance: 0.6,
              Z_Distance: 0.3}
Yaw_Enabled: False
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.

```
Waypoint: { TimeBeforeTrajectory: 20,
            TrajectoryTime: 30,
            MarginTime: 10,
            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

∼*Topic*	Topic to publish the position data.
∼*GazeboRealTime*	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.

```
#!/usr/bin/env python



from subprocess import PIPE, Popen
from threading  import Thread
import sys
import numpy as np
import re
from queue import Queue, Empty
import rospy
from geometry_msgs.msg import PoseStamped
from tf.transformations import quaternion_from_euler


class SphinxPublisher:
    def __init__(self):
        self.q = Queue()
        self.pos = np.zeros(3)
        self.att = np.zeros(3)
        self.cont = 1

        # Run the command
        ON_POSIX = ’posix’ in sys.builtin_module_names
        command = "tlm-data-logger -r 0 inet:127.0.0.1:9060"
        p = Popen(command, stdout=PIPE, bufsize=1, close_fds=ON_POSIX, shell=True)

        # Create a thread which dies with main program
        t = Thread(target=self.process_output, args=(p.stdout, self.q))
        t.daemon = True
        t.start()
        Topic = rospy.get_param(’~Topic’)
        GazeboRealTime = rospy.get_param(’~GazeboRealTime’)
        self.odom_pub = rospy.Publisher(Topic, PoseStamped, queue_size=10)
        Period = 1.0/(120.0*GazeboRealTime)
        rospy.Timer(rospy.Duration(Period), self.publish)
        rospy.spin()


    def process_output(self, out, queue):
        for line in iter(out.readline, b”):
            line = str(line)
            if ".worldPosition" in line:
                number = re.findall(r"[-+]?\d*\.\d+", line)[0]
                if ".x" in line:
                    self.pos[0] = float(number)
                if ".y" in line:
                    self.pos[1] = float(number)
                if ".z" in line:
                    self.pos[2] = float(number)


            if ".worldAttitude" in line:
                number = re.findall(r"[-+]?\d*\.\d+", line)[0]
                if ".x" in line:
                    self.att[0] = float(number)
                if ".y" in line:
                    self.att[1] = float(number)
                if ".z" in line:
                    self.att[2] = float(number)

            queue.put(line)
        out.close()


    def publish(self,event):
        try:
            line = self.q.get_nowait()
        except Empty:
            self.q.queue.clear()
        qx, qy, qz, qw = quaternion_from_euler(self.att[0], self.att[1], self.att[2], 'sxyz')
        pose = PoseStamped()
        pose.header.seq = self.cont
        pose.header.stamp = rospy.Time.now()
        pose.header.frame_id = "bebop2"
        pose.pose.position.x = self.pos[1]
        pose.pose.position.y = self.pos[2]
        pose.pose.position.z = self.pos[0]
        pose.pose.orientation.x = qy
        pose.pose.orientation.y = qz
        pose.pose.orientation.z = qx
        pose.pose.orientation.w = qw
        self.cont += 1
        self.odom_pub.publish(pose)

if __name__ == '__main__':
    rospy.init_node('SphinxPublisher')
    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

∼Dir	The directory where the CSV files are saved.
/Subfolder	The subfolder where the CSV file is located.
∼Topics/CSV_End	Topic used to communicate when the CSV file is complete and graphics can be generated.
∼YAML	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.

```
#!/usr/bin/env python
import rospy
from std_msgs.msg import Empty
import os
import csv
import time
import pandas as pd
import matplotlib
matplotlib.use(’Agg’)
#import matplotlib.pyplot as plt
#import sys



class Plots:
    def __init__(self, path, topic, yaml):
        self.path = path
        self.yaml = yaml
        rospy.Subscriber(topic, Empty, self.Plot)
        rospy.spin()


    def Plot(self, data):
        data = pd.read_csv(os.path.join(self.path,’data.csv’))
        print(os.path.join(self.path,’data.csv’))
        col={’x_ref’: r’$x_r$’, ’y_ref’: r’$y_r$’, ’z_ref’: r’$z_r$’, ’yaw_ref’: r’$\psi_r$’,
                ’x’: r’$x_d$’, ’y’: r’$y_d$’, ’z’: r’$z_d$’, ’yaw’: r’$\psi_d$’,
                ’v_x’: r’$v_x$’, ’v_y’: r’$v_y$’, ’v_z’: r’$v_z$’, ’v_yaw’: r’$v_\psi$’}
        data.rename(columns=col, inplace = True)
        drone_df = data.loc[:, [’Time’,r’$x_d$’,r’$y_d$’,r’$z_d$’,r’$\psi_d$’]]
        reference_df = data.loc[:, [’Time’,r’$x_r$’,r’$y_r$’,r’$z_r$’,r’$\psi_r$’]]
        cmd_vel_df = data.loc[:, [’Time’,r’$v_x$’,r’$v_y$’,r’$v_z$’,r’$v_\psi$’]]

        minV = drone_df[’Time’].min()
        maxV = drone_df[’Time’].max()

        ax = drone_df.plot(x=0,grid=True,title=’Drone’)
        ax.set_xlim(minV,maxV)
        ax.get_figure().savefig(os.path.join(self.path,’drone.png’))
        ax = reference_df.plot(x=0,grid=True,title=’Reference’)
        ax.set_xlim(minV,maxV)
        ax.get_figure().savefig(os.path.join(self.path,’reference.png’))
        ax = cmd_vel_df.plot(x=0,grid=True,title=’CMD_Vel’)
        ax.set_xlim(minV,maxV)
        ax.get_figure().savefig(os.path.join(self.path,’cmd_vel.png’))

        for ax in [r’x’,r’y’,r’z’,r’\psi’]:
            ax_orig = r’$’ + ax + r’$’
            ax_drone = r’$’ + ax + r’_d$’
            ax_ref = r’$’ + ax + r’_r$’
            df = data.loc[:,['Time',ax_drone, ax_ref]]
            name = ax + '.png'
            fn = os.path.join(self.path,name)
            ax = df.plot(x=0,grid=True,title=ax_orig)
            ax.set_xlim(minV,maxV)
            ax.get_figure().savefig(fn)

        os.system("cp $(rospack find bebop_controller)/resource/{} {}".format(self.yaml, self.path))
        os.system("cp $(rospack find bebop_controller)/resource/max_speed.yaml {}".format(self.path))
        os.system("cp $(rospack find bebop_controller)/resource/safe_zone.yaml {}".format(self.path))

        rospy.signal_shutdown("")

if __name__ == '__main__':
    rospy.init_node('plot')
    path = rospy.get_param('~Dir')
    sub = rospy.get_param('/Subfolder')
    topic = rospy.get_param('~Topics/CSV_End')
    yaml = rospy.get_param('~YAML')
    sub = sub[:-1]
    plt = Plots(os.path.join(path,sub),topic,yaml)


```

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.

```


#include "bebop_controller/base_controller.h"

namespace bebop_controller {

    BaseController::BaseController():
    waypointHasBeenPublished_(false),
    takeoff(false),
    controller_active_(false),
    disable_commands(true),
    stop(false) {
        ROS_INFO_ONCE("Started position controller");
        ros::NodeHandle nh("~");

        std::string bebop_command_trajectory, bebop_pose, bebop_cmd_vel, bebop_takeoff, bebop_land, csv_end;
        GetRosParameter(nh, std::string("Topics/Command_Trajectory"),
                        std::string(BEBOP_COMMAND_TRAJECTORY), &bebop_command_trajectory);
        GetRosParameter(nh, std::string("Topics/Pose"),
                        std::string(BEBOP_POSE), &bebop_pose);
        GetRosParameter(nh, std::string("Topics/CMD_Vel"),
                        std::string(BEBOP_CMD_VEL), &bebop_cmd_vel);
        GetRosParameter(nh, std::string("Topics/TakeOff"),
                        std::string(BEBOP_TAKEOFF), &bebop_takeoff);
        GetRosParameter(nh, std::string("Topics/Land"),
                        std::string(BEBOP_LAND), &bebop_land);
        GetRosParameter(nh, std::string("Topics/CSV_End"),
                        std::string(CSV_END), &csv_end);
        GetRosParameter(nh, std::string("Disable_Commands"),
                        DISABLE_COMMANDS, &disable_commands);
        GetRosParameter(nh, std::string("Safe_Zone/X"),
                        SAFE_ZONE_X, &safe_zone.x);
        GetRosParameter(nh, std::string("Safe_Zone/Y"),
                        SAFE_ZONE_Y, &safe_zone.y);
        GetRosParameter(nh, std::string("Safe_Zone/Z"),
                        SAFE_ZONE_Z, &safe_zone.z);
        GetRosParameter(nh, std::string("Max_Speed/X"),
                        LIMIT_X, &max_speed.x);
        GetRosParameter(nh, std::string("Max_Speed/Y"),
                        LIMIT_Y, &max_speed.y);
        GetRosParameter(nh, std::string("Max_Speed/Z"),
                        LIMIT_Z, &max_speed.z);
        GetRosParameter(nh, std::string("Max_Speed/Yaw"),
                        LIMIT_YAW, &max_speed.yaw);

        cmd_multi_dof_joint_trajectory_sub_ = nh.subscribe(bebop_command_trajectory, 1,
                                                        &BaseController::MultiDOFJointTrajectory_CB, this);
        odom_sub_ = nh.subscribe(bebop_pose, 30, &BaseController::Odometry_CB, this);
        end_sub_ = nh.subscribe(csv_end, 1, &BaseController::Stop_CB, this);
        motor_velocity_reference_pub_ = nh.advertise<geometry_msgs::Twist>(bebop_cmd_vel, 1);
        takeoff_pub_ = nh.advertise<std_msgs::Empty>(bebop_takeoff, 1);
        land_pub_ = nh.advertise<std_msgs::Empty>(bebop_land, 1);
        timeOut = nh.createTimer(ros::Duration(1.0), &BaseController::TimeOut_CB, this);

        last_state.position.x = 0;
        last_state.position.y = 0;
        last_state.position.z = 0;
        last_state.orientation.x = 0;
        last_state.orientation.y = 0;
        last_state.orientation.z = 0;
    }

    BaseController::~BaseController() {
        Stop(true);
    }

    void BaseController::MultiDOFJointTrajectory_CB(const trajectory_msgs::MultiDOFJointTrajectoryConstPtr& msg) {
        const size_t n_commands = msg->points.size();
        if (n_commands < 1){
            ROS_WARN_STREAM("Got MultiDOFJointTrajectory message, but message has no points.");
            return;
        }
        mav_msgs::eigenTrajectoryPointFromMsg(msg->points.front(), &command_trajectory_);
        waypointHasBeenPublished_ = true;
        ROS_INFO_ONCE("Got first MultiDOFJointTrajectory message.");
    }

    void BaseController::Stop_CB(const std_msgs::Empty::ConstPtr& empty_msg) {
        Stop(false);
    }

    void BaseController::Odometry_CB(const geometry_msgs::PoseStamped& pose_msg) {
        ROS_INFO_ONCE("Controller got first pose message.");
        timeOut.stop();
        timeOut.start();
        if (waypointHasBeenPublished_) {
            if (!takeoff) {
                TakeOff();
            }
            if (!controller_active_) {
                lastTime = ros::Time::now();
            }
            const Eigen::Vector3d position = Eigen::Vector3d(pose_msg.pose.position.x,
                                                            pose_msg.pose.position.y,
                                                            pose_msg.pose.position.z);
            const Eigen::Quaterniond orientation = Eigen::Quaterniond(pose_msg.pose.orientation.w,
                                                                    pose_msg.pose.orientation.x,
                                                                    pose_msg.pose.orientation.y,
                                                                    pose_msg.pose.orientation.z);
            const Eigen::Vector3d zeros = Eigen::Vector3d(0,0,0);
            mav_msgs::EigenOdometry odometry =
mav_msgs::EigenOdometry(position,orientation,zeros,zeros);
            ros::Time now = ros::Time::now();
            diff = now.toSec() - lastTime.toSec();
            lastTime = now;
            SetOdometry(odometry);
            EstimateVelocity();
            EstimateAcceleration();
            if (!CheckSafeZone()) {
                Stop(true);
            }
            geometry_msgs::Twist ref_command_signals;
            CalculateCommandVelocities(ref_command_signals);
            if (!disable_commands && !stop) {
                motor_velocity_reference_pub_.publish(ref_command_signals);
            }
        }
    }

    void BaseController::TimeOut_CB(const ros::TimerEvent& event) {
        if (takeoff){
            ROS_INFO("Pose messages have not been received for one second. Landing the drone.");
            Stop(true);
        }
    }

    void BaseController::TakeOff() {
        takeoff = true;
        if (disable_commands || stop) {
            return;
        }
        std_msgs::Empty empty_msg;
        takeoff_pub_.publish(empty_msg);
    }

    void BaseController::Land() {
        takeoff = false;
        controller_active_= false;
        if (disable_commands) {
            return;
        }
        std_msgs::Empty empty_msg;
        land_pub_.publish(empty_msg);
    }

    void BaseController::SetOdometry(mav_msgs::EigenOdometry& odometry) {
        odometry_ = odometry;
        controller_active_= true;
        state.position.x = odometry_.position_W[2];
        state.position.y = odometry_.position_W[0];
        state.position.z = odometry_.position_W[1];
        Quaternion2Euler(state.orientation.x, state.orientation.y, state.orientation.z);
    }

    void BaseController::GetErrors(Vector4& e) {
        e.x = state.position.x - command_trajectory_.position_W[0];
        e.y = state.position.y - command_trajectory_.position_W[1];
        e.z = state.position.z - command_trajectory_.position_W[2];
        e.yaw = state.orientation.z - yawFromQuaternion(command_trajectory_.orientation_W_B);
        while (std::abs(e.yaw) > M_PI) {
            if (e.yaw > 0) {
                e.yaw -= 2*M_PI;
            }
            else {
                e.yaw += 2*M_PI;
            }
        }
    }

    void BaseController::GetVelocityErrors(Vector4& dot_e) {
        dot_e.x = state.velocity.x - command_trajectory_.velocity_W[0];
        dot_e.y = state.velocity.y - command_trajectory_.velocity_W[1];
        dot_e.z = state.velocity.z - command_trajectory_.velocity_W[2];
        dot_e.yaw = state.angular_velocity.z - command_trajectory_.angular_velocity_W[2];
    }

    void BaseController::Quaternion2Euler(double& roll, double& pitch, double& yaw) const {
        tf::Quaternion q(odometry_.orientation_W_B.z(), odometry_.orientation_W_B.x(),
                         odometry_.orientation_W_B.y(), odometry_.orientation_W_B.w());
        tf::Matrix3x3 m(q);
        m.getRPY(roll, pitch, yaw);
    }

    void BaseController::CalculateCommandVelocities(geometry_msgs::Twist& ref_command_signals) {
        ref_command_signals.linear.x = 0.0;
        ref_command_signals.linear.y = 0.0;
        ref_command_signals.linear.z = 0.0;
        ref_command_signals.angular.z = 0.0;
    }

    void BaseController::EstimateVelocity() {
        state.velocity.x = (state.position.x - last_state.position.x)/diff;
        state.velocity.y = (state.position.y - last_state.position.y)/diff;
        state.velocity.z = (state.position.z - last_state.position.z)/diff;
        state.angular_velocity.x = (state.orientation.x - last_state.orientation.x)/diff;
        state.angular_velocity.y = (state.orientation.y - last_state.orientation.y)/diff;
        state.angular_velocity.z = (state.orientation.z - last_state.orientation.z)/diff;
        last_state.position = state.position;
        last_state.orientation = state.orientation;
    }

    void BaseController::EstimateAcceleration() {
        state.acceleration.x = (state.velocity.x - last_state.velocity.x)/diff;
        state.acceleration.y = (state.velocity.y - last_state.velocity.y)/diff;
        state.acceleration.z = (state.velocity.z - last_state.velocity.z)/diff;
        state.angular_acceleration.x = (state.angular_velocity.x - last_state.angular_velocity.x)/diff;
        state.angular_acceleration.y = (state.angular_velocity.y - last_state.angular_velocity.y)/diff;
        state.angular_acceleration.z = (state.angular_velocity.z - last_state.angular_velocity.z)/diff;
        last_state.velocity = state.velocity;
        last_state.angular_velocity = state.angular_velocity;
    }

    void BaseController::Stop(bool failsafe) {
        if (failsafe) {
            ROS_INFO_ONCE("Failsafe mode");
        }
        stop = true;
        geometry_msgs::Twist ref_command_signals;
        ref_command_signals.linear.x = 0.0;
        ref_command_signals.linear.y = 0.0;
        ref_command_signals.linear.z = 0.0;
        ref_command_signals.angular.z = 0.0;
        motor_velocity_reference_pub_.publish(ref_command_signals);
        Land();
    }

    bool BaseController::CheckSafeZone() {
        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);
    }

    void BaseController::CalculateLeashLength(Vector4& e, Vector4& P) {
        Vector3 Ppos, vel, accel;
        Ppos.x = P.x*sqrt(std::abs(e.x));
        Ppos.y = P.y*sqrt(std::abs(e.y));
        vel.x = std::abs(max(state.velocity.x,MIN_VEL));
        vel.y = std::abs(max(state.velocity.y,MIN_VEL));
        accel.x = std::abs(max(state.acceleration.x,MIN_ACCEL));
        accel.y = std::abs(max(state.acceleration.y,MIN_ACCEL));
        leash_length.x = accel.x/(2*pow(Ppos.x,2)) + pow(vel.x,2)/(2*accel.x);
        leash_length.y = accel.y/(2*pow(Ppos.y,2)) + pow(vel.y,2)/(2*accel.y);
    }

    void BaseController::LimitPositionErrors(Vector4& e) {
        e.x = clamp(e.x,leash_length.x);
        e.y = clamp(e.y,leash_length.y);
    }

}
```

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.

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

namespace bebop_controller {

    Waypoint::Waypoint(WaypointParameters param):
    wp_params_(param),
    status(BeforeTrajectory) {
        setpoint_pub_ = nh.advertise<trajectory_msgs::MultiDOFJointTrajectory>(wp_params_.topic_command_trajectory, 1);
        Begin_ = nh.advertise<std_msgs::Empty>(wp_params_.topic_csv_begin,1);
        End_ = nh.advertise<std_msgs::Empty>(wp_params_.topic_csv_end,1);

        double TotalTime = wp_params_.TimeBeforeTrajectory + wp_params_.TrajectoryTime + wp_params_.MarginTime;
        _timer1 = _n1.createTimer(ros::Duration(wp_params_.DiffTime), &Waypoint::Trajectory_CB, this, false, true);
        _timer2 = _n2.createTimer(ros::Duration(wp_params_.TimeBeforeTrajectory), &Waypoint::Start_CB, this, true, true);
        _timer3 = _n3.createTimer(ros::Duration(TotalTime), &Waypoint::Stop_CB, this, false, true);
    }

    Waypoint::~Waypoint() {}

    void Waypoint::Start_CB(const ros::TimerEvent& event) {
        status = Trajectory;
        std_msgs::Empty empty_;
        for (int i = 0; i < 50; i++){
            Begin_.publish(empty_);
            std::this_thread::sleep_for(std::chrono::microseconds(100));
        }
        Initial_Time = ros::Time::now();
    }

    void Waypoint::Stop_CB(const ros::TimerEvent& event) {
        std_msgs::Empty empty_;
        for (int i = 0; i < 50; i++){
            End_.publish(empty_);
            std::this_thread::sleep_for(std::chrono::microseconds(100));
        }
        ros::shutdown();
    }

    void Waypoint::Trajectory_CB(const ros::TimerEvent& event) {}

}
```

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 citc_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 citc_controller_angles.cpp.

12.59 citc_controller_angles.cpp

Go to the documentation of this file.

```
00001
00003
00004 #include "citc_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);
```

```
        GetRosParameter(pnh, "Reference_Gains/X", RGxDefaultValue, &RG.x);
        GetRosParameter(pnh, "Reference_Gains/Y", RGyDefaultValue, &RG.y);
        GetRosParameter(pnh, "Reference_Gains/Z", RGzDefaultValue, &RG.z);
        GetRosParameter(pnh, "Reference_Gains/Yaw", RGyawDefaultValue, &RG.yaw);
        GetRosParameter(pnh, "Lambda/X", LambdaX, &lambda.x);
        GetRosParameter(pnh, "Lambda/Y", LambdaY, &lambda.y);
        GetRosParameter(pnh, "Lambda/Z", LambdaZ, &lambda.z);
        GetRosParameter(pnh, "Sigma", Sigma, &sigma);
        GetRosParameter(pnh, std::string("Normalize/Max_Tilt_Angle"),
                        MAX_TILT_ANGLE, &norm.angle);
        GetRosParameter(pnh, std::string("Normalize/Max_Vertical_Speed"),
                        MAX_VERTICAL_SPEED, &norm.vertical);
        GetRosParameter(pnh, std::string("Normalize/Max_Rotation_Speed"),
                        MAX_ROTATION_SPEED, &norm.rotation);
        GetRosParameter(pnh, "Mass", MASS, &mass);
    }

    void CITCController::CalculateCommandVelocities(geometry_msgs::Twist& ref_command_signals){
        if (!controller_active_){
            ref_command_signals.linear.x = 0.0;
            ref_command_signals.linear.y = 0.0;
            ref_command_signals.linear.z = 0.0;
            ref_command_signals.angular.z = 0.0;
            return;
        }
        Vector4 a, v_nom, v_stc, e, dot_e, ddot_e;
        GetErrors(e);
        GetVelocityErrors(dot_e);
        GetAccelerationErrors(ddot_e, dot_e);

        v_nom.x = -K1.x*pow(std::abs(dot_e.x),sigma)*sgn(dot_e.x) - K2.x*pow(std::abs(e.x),((sigma)/(2-sigma)))*sgn(e.x);
        v_nom.y = -K1.y*pow(std::abs(dot_e.y),sigma)*sgn(dot_e.y) - K2.y*pow(std::abs(e.y),((sigma)/(2-sigma)))*sgn(e.y);
        v_nom.z = -K1.z*pow(std::abs(dot_e.z),sigma)*sgn(dot_e.z) - K2.z*pow(std::abs(e.z),((sigma)/(2-sigma)))*sgn(e.z);
        v_nom.yaw = -K1.yaw*pow(std::abs(dot_e.yaw),sigma)*sgn(dot_e.yaw) - K2.yaw*pow(std::abs(e.yaw),((sigma)/(2-sigma)))*sgn(e.yaw);

        IntegrateZetaAndEta(v_nom, ddot_e);

        v_stc.x = -K3.x*std::abs(int_zeta.x)*sgn(int_zeta.x) + int_eta.x;
        v_stc.y = -K3.y*std::abs(int_zeta.y)*sgn(int_zeta.y) + int_eta.y;
        v_stc.z = -K3.z *std::abs(int_zeta.z)*sgn(int_zeta.z) + int_eta.z;
        v_stc.yaw = -K3.yaw *std::abs(int_zeta.yaw)*sgn(int_zeta.yaw) + int_eta.yaw;

        a.x = clamp(v_nom.x + v_stc.x + RG.x*command_trajectory_.acceleration_W[0],
                    mass*(lambda.x-std::abs(command_trajectory_.acceleration_W[0])));
        a.y = clamp(v_nom.y + v_stc.y + RG.y*command_trajectory_.acceleration_W[1],
                    mass*(lambda.y-std::abs(command_trajectory_.acceleration_W[1])));
        a.z = clamp(v_nom.z + v_stc.z + RG.z*command_trajectory_.acceleration_W[2],
                    mass*(lambda.z-std::abs(command_trajectory_.acceleration_W[2])));
        a.yaw = v_nom.yaw + v_stc.yaw;

        double u_Terr, u_T, phi_r, theta_r, psi;
        u_Terr = a.z + mass*GRAVITY;
        u_T = sqrt(pow(a.x,2) + pow(a.y,2) + pow(u_Terr,2));
        psi = state.orientation.z;
        theta_r = atan(((a.x * cos(psi)) + (a.y * sin(psi)))/u_Terr);
        phi_r = atan(cos(theta_r)*(((a.x * sin(psi)) - (a.y * cos(psi)))/(u_Terr)));

        u_z = clamp(u_z + diff*a.z,max_speed.z*norm.vertical);

        ref_command_signals.linear.x = clamp(theta_r*RAD_2_DEG/norm.angle,max_speed.x);
        ref_command_signals.linear.y = clamp(-phi_r*RAD_2_DEG/norm.angle,max_speed.y);
        ref_command_signals.linear.z = clamp(u_z/norm.vertical,max_speed.z);
        ref_command_signals.angular.z = clamp(a.yaw*RAD_2_DEG/norm.rotation,max_speed.yaw);
    }

    void CITCController::GetAccelerationErrors(Vector4& ddot_e, Vector4& dot_e) {
        ddot_e.x = (dot_e.x - last_dot_e.x)/diff;
        ddot_e.y = (dot_e.y - last_dot_e.y)/diff;
        ddot_e.z = (dot_e.z - last_dot_e.z)/diff;
        ddot_e.yaw = (dot_e.yaw - last_dot_e.yaw)/diff;
        last_dot_e.x = dot_e.x;
        last_dot_e.y = dot_e.y;
        last_dot_e.z = dot_e.z;
        last_dot_e.yaw = dot_e.yaw;
    }

    void CITCController::IntegrateZetaAndEta(Vector4& v_nom, Vector4& ddot_e) {
        Vector4 zeta, eta;
        zeta.x = ddot_e.x - v_nom.x;
        zeta.y = ddot_e.y - v_nom.y;
        zeta.z = ddot_e.z - v_nom.z;
        zeta.yaw = ddot_e.yaw - v_nom.yaw;
        int_zeta.x += zeta.x*diff;
        int_zeta.y += zeta.y*diff;
        int_zeta.z += zeta.z*diff;
        int_zeta.yaw += zeta.yaw*diff;
        eta.x = -K4.x*sgn(int_zeta.x);
        eta.y = -K4.y*sgn(int_zeta.y);
        eta.z = -K4.z*sgn(int_zeta.z);
        eta.yaw = -K4.yaw*sgn(int_zeta.yaw);
        int_eta.x += eta.x*diff;
        int_eta.y += eta.y*diff;
        int_eta.z += eta.z*diff;
        int_eta.yaw += eta.yaw*diff;
    }

}

int main(int argc, char** argv){
    ros::init(argc, argv, "citc_controller_node");
    ros::NodeHandle nh;
    bebop_controller::CITCController citc_controller;
    ros::spin();
    return 0;
}
```

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 citc_controller_angles.h.

12.60.2.11 K3yDefaultValue

```
#define K3yDefaultValue 1.0
```

Definition at line 17 of file citc_controller_angles.h.

12.60.2.12 K3zDefaultValue

```
#define K3zDefaultValue 1.0
```

Definition at line 18 of file citc_controller_angles.h.

12.60.2.13 K4xDefaultValue

```
#define K4xDefaultValue 1.0
```

Definition at line 21 of file citc_controller_angles.h.

12.60.2.14 K4yawDefaultValue

```
#define K4yawDefaultValue 1.0
```

Definition at line 24 of file citc_controller_angles.h.

12.60.2.15 K4yDefaultValue

```
#define K4yDefaultValue 1.0
```

Definition at line 22 of file citc_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 citc_controller_angles.h.

12.60.2.23 MAX_VERTICAL_SPEED

```
#define MAX_VERTICAL_SPEED 1.0
```

Definition at line 37 of file citc_controller_angles.h.

12.60.2.24 RGxDefaultValue

```
#define RGxDefaultValue 0.0
```

Definition at line 26 of file citc_controller_angles.h.

12.60.2.25 RGyawDefaultValue

```
#define RGyawDefaultValue 0.0
```

Definition at line 29 of file citc_controller_angles.h.

12.60.2.26 RGyDefaultValue

```
#define RGyDefaultValue 0.0
```

Definition at line 27 of file citc_controller_angles.h.

12.60.2.27 RGzDefaultValue

```
#define RGzDefaultValue 0.0
```

Definition at line 28 of file citc_controller_angles.h.

12.60.2.28 Sigma

```
#define Sigma 0.8
```

Definition at line 34 of file citc_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.

```


#include "citc_controller_twist.h"

namespace bebop_controller {

    CITCController::CITCController() {
        ros::NodeHandle pnh("~");
        int_zeta.x = int_zeta.y = int_zeta.z = int_zeta.yaw = 0;
        int_eta.x = int_eta.y = int_eta.z = int_eta.yaw = 0;
        last_dot_e.x = last_dot_e.y = last_dot_e.z = last_dot_e.yaw = 0;
        u.x = u.y = u.z = u.yaw = 0;
        GetRosParameter(pnh, "Gains/K1x", K1xDefaultValue, &K1.x);
        GetRosParameter(pnh, "Gains/K1y", K1yDefaultValue, &K1.y);
        GetRosParameter(pnh, "Gains/K1z", K1zDefaultValue, &K1.z);
        GetRosParameter(pnh, "Gains/K1yaw", K1yawDefaultValue, &K1.yaw);
        GetRosParameter(pnh, "Gains/K2x", K2xDefaultValue, &K2.x);
        GetRosParameter(pnh, "Gains/K2y", K2yDefaultValue, &K2.y);
        GetRosParameter(pnh, "Gains/K2z", K2zDefaultValue, &K2.z);
        GetRosParameter(pnh, "Gains/K2yaw", K2yawDefaultValue, &K2.yaw);
        GetRosParameter(pnh, "Gains/K3x", K3xDefaultValue, &K3.x);
        GetRosParameter(pnh, "Gains/K3y", K3yDefaultValue, &K3.y);
        GetRosParameter(pnh, "Gains/K3z", K3zDefaultValue, &K3.z);
        GetRosParameter(pnh, "Gains/K3yaw", K3yawDefaultValue, &K3.yaw);
        GetRosParameter(pnh, "Gains/K4x", K4xDefaultValue, &K4.x);
        GetRosParameter(pnh, "Gains/K4y", K4yDefaultValue, &K4.y);
        GetRosParameter(pnh, "Gains/K4z", K4zDefaultValue, &K4.z);
        GetRosParameter(pnh, "Gains/K4yaw", K4yawDefaultValue, &K4.yaw);
        GetRosParameter(pnh, "Reference_Gains/X", RGxDefaultValue, &RG.x);
        GetRosParameter(pnh, "Reference_Gains/Y", RGyDefaultValue, &RG.y);
        GetRosParameter(pnh, "Reference_Gains/Z", RGzDefaultValue, &RG.z);
        GetRosParameter(pnh, "Reference_Gains/Yaw", RGyawDefaultValue, &RG.yaw);
        GetRosParameter(pnh, "Lambda/X", LambdaX, &lambda.x);
        GetRosParameter(pnh, "Lambda/Y", LambdaY, &lambda.y);
        GetRosParameter(pnh, "Lambda/Z", LambdaZ, &lambda.z);
        GetRosParameter(pnh, "Sigma", Sigma, &sigma);
        GetRosParameter(pnh, std::string("Normalize/Max_Horizontal_Speed"),
                        MAX_HORIZONTAL_SPEED, &norm.horizontal);
        GetRosParameter(pnh, std::string("Normalize/Max_Vertical_Speed"),
                        MAX_VERTICAL_SPEED, &norm.vertical);
        GetRosParameter(pnh, std::string("Normalize/Max_Rotation_Speed"),
                        MAX_ROTATION_SPEED, &norm.rotation);
        GetRosParameter(pnh, "Mass", MASS, &mass);
    }

    void CITCController::CalculateCommandVelocities(geometry_msgs::Twist& ref_command_signals){
        if (!controller_active_){
            ref_command_signals.linear.x = 0.0;
            ref_command_signals.linear.y = 0.0;
            ref_command_signals.linear.z = 0.0;
            ref_command_signals.angular.z = 0.0;
            return;
        }
        Vector4 a, v_nom, v_stc, e, dot_e, ddot_e;
        GetErrors(e);
        GetVelocityErrors(dot_e);
        GetAccelerationErrors(ddot_e, dot_e);

        v_nom.x = -K1.x*pow(std::abs(dot_e.x),sigma)*sgn(dot_e.x) -
 K2.x*pow(std::abs(e.x),((sigma)/(2-sigma)))*sgn(e.x);
        v_nom.y = -K1.y*pow(std::abs(dot_e.y),sigma)*sgn(dot_e.y) -
 K2.y*pow(std::abs(e.y),((sigma)/(2-sigma)))*sgn(e.y);
        v_nom.z = -K1.z*pow(std::abs(dot_e.z),sigma)*sgn(dot_e.z) - K2.z
 *pow(std::abs(e.z),((sigma)/(2-sigma)))*sgn(e.z);
        v_nom.yaw = -K1.yaw*pow(std::abs(dot_e.yaw),sigma)*sgn(dot_e.yaw) - K2.yaw
 *pow(std::abs(e.yaw),((sigma)/(2-sigma)))*sgn(e.yaw);

        IntegrateZetaAndEta(v_nom, ddot_e);

        v_stc.x = -K3.x*std::abs(int_zeta.x)*sgn(int_zeta.x) + int_eta.x;
        v_stc.y = -K3.y*std::abs(int_zeta.y)*sgn(int_zeta.y) + int_eta.y;
        v_stc.z = -K3.z *std::abs(int_zeta.z)*sgn(int_zeta.z) + int_eta.z;
        v_stc.yaw = -K3.yaw *std::abs(int_zeta.yaw)*sgn(int_zeta.yaw) + int_eta.yaw;

        a.x = clamp(v_nom.x + v_stc.x + RG.x*command_trajectory_.acceleration_W[0],
                    mass*(lambda.x-std::abs(command_trajectory_.acceleration_W[0])));
        a.y = clamp(v_nom.y + v_stc.y + RG.y*command_trajectory_.acceleration_W[1],
                    mass*(lambda.y-std::abs(command_trajectory_.acceleration_W[1])));
        a.z = clamp(v_nom.z + v_stc.z + RG.z*command_trajectory_.acceleration_W[2],
                    mass*(lambda.z-std::abs(command_trajectory_.acceleration_W[2])));
        a.yaw = v_nom.yaw + v_stc.yaw;

        u.x = clamp(u.x + (diff*a.x)/norm.horizontal,max_speed.x);
        u.y = clamp(u.y + (diff*a.y)/norm.horizontal,max_speed.y);
        u.z = clamp(u.z + (diff*a.z)/norm.vertical,max_speed.z);
        u.yaw = clamp(u.yaw + (diff*a.yaw)*RAD_2_DEG/norm.rotation,max_speed.yaw);

        ref_command_signals.linear.x = clamp(cos(state.orientation.z)*u.x - sin(state.orientation.z)*u.y,max_speed.x);
        ref_command_signals.linear.y = clamp(sin(state.orientation.z)*u.x + cos(state.orientation.z)*u.y,max_speed.y);
        ref_command_signals.linear.z = clamp(u.z,max_speed.z);
        ref_command_signals.angular.z = clamp(u.yaw,max_speed.yaw);
    }

    void CITCController::GetAccelerationErrors(Vector4& ddot_e, Vector4& dot_e) {
        ddot_e.x = (dot_e.x - last_dot_e.x)/diff;
        ddot_e.y = (dot_e.y - last_dot_e.y)/diff;
        ddot_e.z = (dot_e.z - last_dot_e.z)/diff;
        ddot_e.yaw = (dot_e.yaw - last_dot_e.yaw)/diff;
        last_dot_e.x = dot_e.x;
        last_dot_e.y = dot_e.y;
        last_dot_e.z = dot_e.z;
        last_dot_e.yaw = dot_e.yaw;
    }

    void CITCController::IntegrateZetaAndEta(Vector4& v_nom, Vector4& ddot_e) {
        Vector4 zeta, eta;
        zeta.x = ddot_e.x - v_nom.x;
        zeta.y = ddot_e.y - v_nom.y;
        zeta.z = ddot_e.z - v_nom.z;
        zeta.yaw = ddot_e.yaw - v_nom.yaw;
        int_zeta.x += zeta.x*diff;
        int_zeta.y += zeta.y*diff;
        int_zeta.z += zeta.z*diff;
        int_zeta.yaw += zeta.yaw*diff;
        eta.x = -K4.x*sgn(int_zeta.x);
        eta.y = -K4.y*sgn(int_zeta.y);
        eta.z = -K4.z*sgn(int_zeta.z);
        eta.yaw = -K4.yaw*sgn(int_zeta.yaw);
        int_eta.x += eta.x*diff;
        int_eta.y += eta.y*diff;
        int_eta.z += eta.z*diff;
        int_eta.yaw += eta.yaw*diff;
    }

}

int main(int argc, char** argv){
    ros::init(argc, argv, "citc_controller_node");
    ros::NodeHandle nh;
    bebop_controller::CITCController citc_controller;
    ros::spin();
    return 0;
}
```

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 citc_controller_twist.h.

12.64.2.27 RGzDefaultValue

```
#define RGzDefaultValue 0.0
```

Definition at line 28 of file citc_controller_twist.h.

12.64.2.28 Sigma

```
#define Sigma 0.8
```

Definition at line 34 of file citc_controller_twist.h.

12.65 citc_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
```

```
namespace bebop_controller {

    struct Normalize {
        double horizontal;
        double vertical;
        double rotation;
    };

    class CITCController : public BaseController {
        public:
            CITCController();

        private:
            Vector4 K1, K2, K3, K4, RG, u;
            Vector4 int_zeta, int_eta, last_dot_e;
            Vector3 lambda;
            Normalize norm;
            double mass, sigma;
            void CalculateCommandVelocities(geometry_msgs::Twist& ref_command_signals) override;
            void GetAccelerationErrors(Vector4& ddot_e, Vector4& dot_e);
            void IntegrateZetaAndEta(Vector4& v_nom, Vector4& ddot_e);
    };

}
```

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.

```


#include "data_to_csv.h"

namespace bebop_controller {

    DataToCSV::DataToCSV(DataToCSVParameters params):
    parameters(params) {
        drone_pose.position.x = drone_pose.position.y = drone_pose.position.z = 0;
        drone_pose.orientation.x = drone_pose.orientation.y = drone_pose.orientation.z = 0;
        reference_pose.position.x = reference_pose.position.y = reference_pose.position.z = 0;
        reference_pose.orientation.x = reference_pose.orientation.y = reference_pose.orientation.z = 0;
        odometry.position.x = odometry.position.y = odometry.position.z = 0;
        odometry.velocity.x = odometry.velocity.y = odometry.velocity.z = 0;
        CSV_File.open(parameters.File.c_str());
        CSV_File « "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";
        drone_pose_sub = nh.subscribe(parameters.Topic_Drone_Pose, 1,
                                                      &DataToCSV::Odometry_CB, this);
        reference_pose_sub = nh.subscribe(parameters.Topic_Reference_Pose, 1,
                                                      &DataToCSV::Reference_CB, this);
        cmd_vel_sub = nh.subscribe(parameters.Topic_CMD_Vel, 1,
                                                      &DataToCSV::Command_Velocities_CB, this);
        csv_begin = nh.subscribe(parameters.Topic_CSV_Begin, 1, &DataToCSV::Begin_CB, this);
        csv_end = nh.subscribe(parameters.Topic_CSV_End, 1, &DataToCSV::Stop_CB, this);
        timer = nh.createTimer(ros::Duration(0.01), &DataToCSV::Timer_CB, this);
        velocities_sub = nh.subscribe(parameters.Topic_Velocities, 1,
                                                      &DataToCSV::Drone_CB, this);
    }

    DataToCSV::~DataToCSV() {}

    void DataToCSV::Begin_CB(const std_msgs::Empty::ConstPtr& empty_msg) {
        ROS_INFO_ONCE("Data is being stored in CSV.");
        Initial_Time = ros::Time::now();
        hasBegun = true;
    }

    void DataToCSV::Stop_CB(const std_msgs::Empty::ConstPtr& empty_msg) {
        hasBegun = false;
        CSV_File.close();
        ROS_INFO_ONCE("Data has finished saving in CSV");
        ros::shutdown();
    }

    void DataToCSV::Odometry_CB(const geometry_msgs::PoseStamped& pose_msg) {
        Eigen::Vector4d q = Eigen::Vector4d(pose_msg.pose.orientation.z, pose_msg.pose.orientation.x,
                                            pose_msg.pose.orientation.y, pose_msg.pose.orientation.w);
        //Eigen::Vector4d q = vector4FromQuaternionMsg(pose_msg.pose.orientation);
        Eigen::Vector3d rpy = Quat2RPY(q);
        drone_pose.position.x = pose_msg.pose.position.z;
        drone_pose.position.y = pose_msg.pose.position.x;
        drone_pose.position.z = pose_msg.pose.position.y;
        drone_pose.orientation.z = rpy.z();
    }

    void DataToCSV::Reference_CB(const trajectory_msgs::MultiDOFJointTrajectoryConstPtr& reference_msg) {
        const size_t n_commands = reference_msg->points.size();
        if (n_commands < 1){
            return;
        }
        mav_msgs::EigenTrajectoryPoint reference;
        mav_msgs::eigenTrajectoryPointFromMsg(reference_msg->points.front(), &reference);
        reference_pose.position.x = reference.position_W[0];
        reference_pose.position.y = reference.position_W[1];
        reference_pose.position.z = reference.position_W[2];
        reference_pose.orientation.z = yawFromQuaternion(reference.orientation_W_B);
    }

    void DataToCSV::Timer_CB(const ros::TimerEvent& event) {
        if (hasBegun) {
            ros::Time now = ros::Time::now();
            double t = now.toSec() - Initial_Time.toSec();
            if (t < 0.005) {
                return;
            }
            //ROS_INFO("CSV - Time: %f; Initial_Time: %f; now: %f", t, Initial_Time.toSec(), now.toSec());
            CSV_File    « t « ","
                        « reference_pose.position.x « ","
                        « reference_pose.position.y « ","
                        « reference_pose.position.z « ","
                        « reference_pose.orientation.z « ","
                        « drone_pose.position.x « ","
                        « drone_pose.position.y « ","
                        « drone_pose.position.z « ","
                        « drone_pose.orientation.z « ","
                        « cmd_vel.x « ","
                        « cmd_vel.y « ","
                        « cmd_vel.z « ","
                        « cmd_vel.yaw « ","
                        « odometry.position.x « ","
                        « odometry.position.y « ","
                        « odometry.position.z « ","
                        « odometry.velocity.x « ","
                        « odometry.velocity.y « ","
                        « odometry.velocity.z « "\n";
        }
    }

    void DataToCSV::Command_Velocities_CB(const geometry_msgs::Twist cmd_vel_msg) {
        cmd_vel.x = cmd_vel_msg.linear.x;
        cmd_vel.y = cmd_vel_msg.linear.y;
        cmd_vel.z = cmd_vel_msg.linear.z;
        cmd_vel.yaw = cmd_vel_msg.angular.z;
    }

    void DataToCSV::Drone_CB(const nav_msgs::OdometryConstPtr& odom_msg) {
        odometry.position.x = odom_msg->pose.pose.position.x;
        odometry.position.y = odom_msg->pose.pose.position.y;
        odometry.position.z = odom_msg->pose.pose.position.z;
        odometry.velocity.x = odom_msg->twist.twist.linear.x;
        odometry.velocity.y = odom_msg->twist.twist.linear.y;
        odometry.velocity.z = odom_msg->twist.twist.linear.z;
    }

}

int main(int argc, char** argv){
    ros::init(argc, argv, "DataToCSV", ros::init_options::AnonymousName);
    ros::NodeHandle nh;
    bebop_controller::DataToCSVParameters parameters;
    std::string Path, DateAndTime, Name, FullPath;
    ros::param::get("~Topics/Command_Trajectory", parameters.Topic_Reference_Pose);
    ros::param::get("~Topics/Pose", parameters.Topic_Drone_Pose);
    ros::param::get("~Topics/CMD_Vel", parameters.Topic_CMD_Vel);
    ros::param::get("~Topics/CSV_Begin", parameters.Topic_CSV_Begin);
    ros::param::get("~Topics/CSV_End", parameters.Topic_CSV_End);
    ros::param::get("~Waypoint/MarginTime", parameters.Margin_Time);
    ros::param::get("~Dir", Path);
    ros::param::get("/Subfolder", DateAndTime);
    ros::param::get("~Topics/Velocities", parameters.Topic_Velocities);
    DateAndTime.pop_back();
    FullPath = Path + DateAndTime;
    mkdir(FullPath.c_str(), 0777);
    parameters.File = FullPath + "/" + std::string("data.csv");
    bebop_controller::DataToCSV DataToCSV_(parameters);
    ros::spin();
    return 0;
}
```

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.

```


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

namespace bebop_controller {

    struct DataToCSVParameters {
        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;
    };

    class DataToCSV{
        public:
            DataToCSV(DataToCSVParameters params);
            ~DataToCSV();

        private:
            State drone_pose;
            State reference_pose;
            State odometry;

            Command_Velocities cmd_vel;
            DataToCSVParameters parameters;

            ros::NodeHandle nh;
            ros::Subscriber drone_pose_sub;
            ros::Subscriber reference_pose_sub;
            ros::Subscriber cmd_vel_sub;
            ros::Subscriber csv_begin;
            ros::Subscriber csv_end;
            ros::Subscriber velocities_sub;
            ros::Timer timer;
            ros::Time Initial_Time;

            bool hasBegun = false;
            std::ofstream CSV_File;

            void Begin_CB(const std_msgs::Empty::ConstPtr& empty_msg);
            void Stop_CB(const std_msgs::Empty::ConstPtr& empty_msg);
            void Odometry_CB(const geometry_msgs::PoseStamped& pose_msg);
            void Reference_CB(const trajectory_msgs::MultiDOFJointTrajectoryConstPtr& reference_msg);
            void Command_Velocities_CB(const geometry_msgs::Twist cmd_vel_msg);
            void Timer_CB(const ros::TimerEvent& event);
            void Drone_CB(const nav_msgs::OdometryConstPtr& odom_msg);
    };

}
```

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.

```
#include "pid_controller_angles.h"

namespace bebop_controller {

    PIDController::PIDController() {
        ros::NodeHandle pnh("~");
        int_e.x = int_e.y = int_e.z = int_e.yaw = u_z = 0;
        GetRosParameter(pnh, "Gains/Px", PxDefaultValue, &P.x);
        GetRosParameter(pnh, "Gains/Py", PyDefaultValue, &P.y);
        GetRosParameter(pnh, "Gains/Pz", PzDefaultValue, &P.z);
        GetRosParameter(pnh, "Gains/Pyaw", PyawDefaultValue, &P.yaw);
        GetRosParameter(pnh, "Gains/Dx", DxDefaultValue, &D.x);
        GetRosParameter(pnh, "Gains/Dy", DyDefaultValue, &D.y);
        GetRosParameter(pnh, "Gains/Dz", DzDefaultValue, &D.z);
        GetRosParameter(pnh, "Gains/Dyaw", DyawDefaultValue, &D.yaw);
        GetRosParameter(pnh, "Gains/Ix", IxDefaultValue, &I.x);
        GetRosParameter(pnh, "Gains/Iy", IyDefaultValue, &I.y);
        GetRosParameter(pnh, "Gains/Iz", IzDefaultValue, &I.z);
        GetRosParameter(pnh, "Gains/Iyaw", IzDefaultValue, &I.yaw);
        GetRosParameter(pnh, "Limits/Ix", LIxDefaultValue, &LI.x);
        GetRosParameter(pnh, "Limits/Iy", LIyDefaultValue, &LI.y);
        GetRosParameter(pnh, "Limits/Iz", LIzDefaultValue, &LI.z);
        GetRosParameter(pnh, "Limits/Iyaw", LIyawDefaultValue, &LI.yaw);
        GetRosParameter(pnh, "Reference_Gains/X", RGxDefaultValue, &RG.x);
        GetRosParameter(pnh, "Reference_Gains/Y", RGyDefaultValue, &RG.y);
        GetRosParameter(pnh, "Reference_Gains/Z", RGzDefaultValue, &RG.z);
        GetRosParameter(pnh, "Reference_Gains/Yaw", RGyawDefaultValue, &RG.yaw);
        GetRosParameter(pnh, "Lambda/X", LambdaX, &lambda.x);
        GetRosParameter(pnh, "Lambda/Y", LambdaY, &lambda.y);
        GetRosParameter(pnh, "Lambda/Z", LambdaZ, &lambda.z);
        GetRosParameter(pnh, std::string("Normalize/Max_Tilt_Angle"),
                        MAX_TILT_ANGLE, &norm.angle);
        GetRosParameter(pnh, std::string("Normalize/Max_Vertical_Speed"),
                        MAX_VERTICAL_SPEED, &norm.vertical);
        GetRosParameter(pnh, std::string("Normalize/Max_Rotation_Speed"),
                        MAX_ROTATION_SPEED, &norm.rotation);
        GetRosParameter(pnh, "Mass", MASS, &mass);
    }

    void PIDController::CalculateCommandVelocities(geometry_msgs::Twist& ref_command_signals){
        if (!controller_active_){
            ref_command_signals.linear.x = 0.0;
            ref_command_signals.linear.y = 0.0;
            ref_command_signals.linear.z = 0.0;
            ref_command_signals.angular.z = 0.0;
            return;
        }
        Vector4 a, e, dot_e;
        GetErrors(e);
        CalculateLeashLength(e, P);
        LimitPositionErrors(e);
        IntegrateErrors(e);
        LimitIntegralPart();
        GetVelocityErrors(dot_e);

        a.x = clamp(-P.x*e.x - D.x*dot_e.x - I.x*int_e.x + RG.x*command_trajectory_.acceleration_W[0],
                    mass*(lambda.x-std::abs(command_trajectory_.acceleration_W[0])));
        a.y = clamp(-P.y*e.y - D.y*dot_e.y - I.y*int_e.y + RG.y*command_trajectory_.acceleration_W[1],
                    mass*(lambda.y-std::abs(command_trajectory_.acceleration_W[1])));
        a.z = clamp(-P.z*e.z - D.z*dot_e.z - I.z*int_e.z + RG.z*command_trajectory_.acceleration_W[2],
                    mass*(lambda.z-std::abs(command_trajectory_.acceleration_W[2])));
        a.yaw = -P.yaw*e.yaw - D.yaw*dot_e.yaw - I.yaw*int_e.yaw;

        double u_Terr, u_T, phi_r, theta_r, psi;
        u_Terr = a.z + mass*GRAVITY;
        u_T = sqrt(pow(a.x,2) + pow(a.y,2) + pow(u_Terr,2));
        psi = state.orientation.z;
        theta_r = atan(((a.x * cos(psi)) + (a.y * sin(psi)))/u_Terr);
        phi_r = atan(cos(theta_r)*(((a.x * sin(psi)) - (a.y * cos(psi)))/(u_Terr)));

        u_z = clamp(u_z + diff*a.z,max_speed.z*norm.vertical);

        ref_command_signals.linear.x = clamp(theta_r*RAD_2_DEG/norm.angle,max_speed.x);
        ref_command_signals.linear.y = clamp(-phi_r*RAD_2_DEG/norm.angle,max_speed.y);
        ref_command_signals.linear.z = clamp(u_z/norm.vertical,max_speed.z);
        ref_command_signals.angular.z = clamp(a.yaw*RAD_2_DEG/norm.rotation,max_speed.yaw);
    }

    void PIDController::IntegrateErrors(Vector4& e) {
        int_e.x += e.x*diff;
        int_e.y += e.y*diff;
        int_e.z += e.z*diff;
        int_e.yaw += e.yaw*diff;
    }

    void PIDController::LimitIntegralPart() {
        if (LI.x > 0) int_e.x = clamp(int_e.x, LI.x/I.x);
        if (LI.y > 0) int_e.y = clamp(int_e.y, LI.y/I.y);
        if (LI.z > 0) int_e.z = clamp(int_e.z, LI.z/I.z);
        if (LI.yaw > 0) int_e.yaw = clamp(int_e.yaw, LI.yaw/I.yaw);
    }

}

int main(int argc, char** argv){
    ros::init(argc, argv, "pid_controller_node");
    ros::NodeHandle nh;
    bebop_controller::PIDController pid_controller;
    ros::spin();
    return 0;
}
```

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 LIxDefaultValue 1.0
- #define LIyDefaultValue 1.0
- #define LIzDefaultValue 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 LIxDefaultValue

```
#define LIxDefaultValue 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.

```


#include "bebop_controller/base_controller.h"

#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 LIxDefaultValue 1.0
#define LIyDefaultValue 1.0
#define LIzDefaultValue 1.0
#define LIyawDefaultValue 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

namespace bebop_controller {

    struct Normalize {
        double angle;
        double vertical;
        double rotation;
    };

    class PIDController : public BaseController {
        public:
            PIDController();

        private:
            Vector4 P, D, I, LI, RG, int_e;
            Vector3 lambda;
            Normalize norm;
            double mass, u_z;
            void CalculateCommandVelocities(geometry_msgs::Twist& ref_command_signals) override;
            void IntegrateErrors(Vector4& e);
            void LimitIntegralPart();
    };

}
```

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.

```


#include "pid_controller_twist.h"

namespace bebop_controller {

    PIDController::PIDController() {
        u.x = u.y = u.z = u.yaw = 0;
        int_e.x = int_e.y = int_e.z = int_e.yaw = 0;
        ros::NodeHandle pnh("~");
        GetRosParameter(pnh, "Gains/Px", PxDefaultValue, &P.x);
        GetRosParameter(pnh, "Gains/Py", PyDefaultValue, &P.y);
        GetRosParameter(pnh, "Gains/Pz", PzDefaultValue, &P.z);
        GetRosParameter(pnh, "Gains/Pyaw", PyawDefaultValue, &P.yaw);
        GetRosParameter(pnh, "Gains/Dx", DxDefaultValue, &D.x);
        GetRosParameter(pnh, "Gains/Dy", DyDefaultValue, &D.y);
        GetRosParameter(pnh, "Gains/Dz", DzDefaultValue, &D.z);
        GetRosParameter(pnh, "Gains/Dyaw", DyawDefaultValue, &D.yaw);
        GetRosParameter(pnh, "Gains/Ix", IxDefaultValue, &I.x);
        GetRosParameter(pnh, "Gains/Iy", IyDefaultValue, &I.y);
        GetRosParameter(pnh, "Gains/Iz", IzDefaultValue, &I.z);
        GetRosParameter(pnh, "Gains/Iyaw", IzDefaultValue, &I.yaw);
        GetRosParameter(pnh, "Limits/Ix", LIxDefaultValue, &LI.x);
        GetRosParameter(pnh, "Limits/Iy", LIyDefaultValue, &LI.y);
        GetRosParameter(pnh, "Limits/Iz", LIzDefaultValue, &LI.z);
        GetRosParameter(pnh, "Limits/Iyaw", LIyawDefaultValue, &LI.yaw);
        GetRosParameter(pnh, "Reference_Gains/X", RGxDefaultValue, &RG.x);
        GetRosParameter(pnh, "Reference_Gains/Y", RGyDefaultValue, &RG.y);
        GetRosParameter(pnh, "Reference_Gains/Z", RGzDefaultValue, &RG.z);
        GetRosParameter(pnh, "Reference_Gains/Yaw", RGyawDefaultValue, &RG.yaw);
        GetRosParameter(pnh, "Lambda/X", LambdaX, &lambda.x);
        GetRosParameter(pnh, "Lambda/Y", LambdaY, &lambda.y);
        GetRosParameter(pnh, "Lambda/Z", LambdaZ, &lambda.z);
        GetRosParameter(pnh, std::string("Normalize/Max_Horizontal_Speed"),
                        MAX_HORIZONTAL_SPEED, &norm.horizontal);
        GetRosParameter(pnh, std::string("Normalize/Max_Vertical_Speed"),
                        MAX_VERTICAL_SPEED, &norm.vertical);
        GetRosParameter(pnh, std::string("Normalize/Max_Rotation_Speed"),
                        MAX_ROTATION_SPEED, &norm.rotation);
        GetRosParameter(pnh, "Mass", MASS, &mass);
    }

    void PIDController::CalculateCommandVelocities(geometry_msgs::Twist& ref_command_signals){
        if (!controller_active_){
            ref_command_signals.linear.x = 0.0;
            ref_command_signals.linear.y = 0.0;
            ref_command_signals.linear.z = 0.0;
            ref_command_signals.angular.z = 0.0;
            return;
        }
        Vector4 a, e, dot_e;
        GetErrors(e);
        CalculateLeashLength(e, P);
        LimitPositionErrors(e);
        IntegrateErrors(e);
        LimitIntegralPart();
        GetVelocityErrors(dot_e);

        a.x = -P.x*e.x - D.x*dot_e.x - I.x*int_e.x + RG.z*command_trajectory_.acceleration_W[0];
        a.y = -P.y*e.y - D.y*dot_e.y - I.y*int_e.y + RG.z*command_trajectory_.acceleration_W[1];
        a.z = -P.z*e.z - D.z*dot_e.z - I.z*int_e.z + RG.z*command_trajectory_.acceleration_W[2];
        a.yaw = -P.yaw*e.yaw - D.yaw*dot_e.yaw - I.yaw*int_e.yaw;

        u.x = clamp(u.x + (diff*a.x)/norm.horizontal,max_speed.x);
        u.y = clamp(u.y + (diff*a.y)/norm.horizontal,max_speed.y);
        u.z = clamp(u.z + (diff*a.z)/norm.vertical,max_speed.z);
        u.yaw = clamp(u.yaw + (diff*a.yaw)*RAD_2_DEG/norm.rotation,max_speed.yaw);

        ref_command_signals.linear.x = clamp(cos(state.orientation.z)*u.x - sin(state.orientation.z)*u.y,max_speed.x);
        ref_command_signals.linear.y = clamp(sin(state.orientation.z)*u.x + cos(state.orientation.z)*u.y,max_speed.y);
        ref_command_signals.linear.z = clamp(u.z,max_speed.z);
        ref_command_signals.angular.z = clamp(u.yaw,max_speed.yaw);
    }

    void PIDController::IntegrateErrors(Vector4& e) {
        int_e.x += e.x*diff;
        int_e.y += e.y*diff;
        int_e.z += e.z*diff;
        int_e.yaw += e.yaw*diff;
    }

    void PIDController::LimitIntegralPart() {
        if (LI.x > 0) int_e.x = clamp(int_e.x, LI.x/I.x);
        if (LI.y > 0) int_e.y = clamp(int_e.y, LI.y/I.y);
        if (LI.z > 0) int_e.z = clamp(int_e.z, LI.z/I.z);
        if (LI.yaw > 0) int_e.yaw = clamp(int_e.yaw, LI.yaw/I.yaw);
    }

}

int main(int argc, char** argv){
    ros::init(argc, argv, "pid_controller_node");
    ros::NodeHandle nh;
    bebop_controller::PIDController pid_controller;
    ros::spin();
    return 0;
}
```

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 IxDefaultValue 1.0
- #define IyDefaultValue 1.0
- #define IzDefaultValue 1.0
- #define IyawDefaultValue 1.0
- #define LIxDefaultValue 1.0
- #define LIyDefaultValue 1.0
- #define LIzDefaultValue 1.0
- #define LIyawDefaultValue 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 LIxDefaultValue

```
#define LIxDefaultValue 1.0
```

Definition at line 21 of file pid_controller_twist.h.

12.76.2.13 LIyawDefaultValue

```
#define LIyawDefaultValue 1.0
```

Definition at line 24 of file pid_controller_twist.h.

12.76.2.14 LIyDefaultValue

```
#define LIyDefaultValue 1.0
```

Definition at line 22 of file pid_controller_twist.h.

12.76.2.15 LIzDefaultValue

```
#define LIzDefaultValue 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 {

    struct Normalize {
        double horizontal;
        double vertical;
        double rotation;
    };

    class PIDController : public BaseController {
        public:
            PIDController();

        private:
            Vector4 P, D, I, LI, u, RG, int_e;
            Vector3 lambda;
            Normalize norm;
            double mass;
            void CalculateCommandVelocities(geometry_msgs::Twist& ref_command_signals) override;
            void IntegrateErrors(Vector4& e);
            void LimitIntegralPart();
    };

}
```

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.

```


#include "proportional_controller.h"

namespace bebop_controller {

    ProportionalController::ProportionalController(){
        ros::NodeHandle pnh("~");
        GetRosParameter(pnh, "Gains/Px", PxDefaultValue, &P.x);
        GetRosParameter(pnh, "Gains/Py", PyDefaultValue, &P.y);
        GetRosParameter(pnh, "Gains/Pz", PzDefaultValue, &P.z);
        GetRosParameter(pnh, "Gains/Pyaw", PyawDefaultValue, &P.yaw);
        GetRosParameter(pnh, "Reference_Gains/X", RGxDefaultValue, &RG.x);
        GetRosParameter(pnh, "Reference_Gains/Y", RGyDefaultValue, &RG.y);
        GetRosParameter(pnh, "Reference_Gains/Z", RGzDefaultValue, &RG.z);
        GetRosParameter(pnh, "Reference_Gains/Yaw", RGyawDefaultValue, &RG.yaw);
        GetRosParameter(pnh, std::string("Normalize/Max_Horizontal_Speed"),
                        MAX_HORIZONTAL_SPEED, &norm.horizontal);
        GetRosParameter(pnh, std::string("Normalize/Max_Vertical_Speed"),
                        MAX_VERTICAL_SPEED, &norm.vertical);
        GetRosParameter(pnh, std::string("Normalize/Max_Rotation_Speed"),
                        MAX_ROTATION_SPEED, &norm.rotation);
    }

    void ProportionalController::CalculateCommandVelocities(geometry_msgs::Twist&
 ref_command_signals){
        if (!controller_active_){
            ref_command_signals.linear.x = 0.0;
            ref_command_signals.linear.y = 0.0;
            ref_command_signals.linear.z = 0.0;
            ref_command_signals.angular.z = 0.0;
            return;
        }
        Vector4 u, e;
        GetErrors(e);
        CalculateLeashLength(e, P);
        LimitPositionErrors(e);

        u.x = (-P.x*e.x + RG.x*command_trajectory_.velocity_W[0])/norm.horizontal;
        u.y = (-P.y*e.y + RG.y*command_trajectory_.velocity_W[1])/norm.horizontal;
        u.z = (-P.z*e.z + RG.z*command_trajectory_.velocity_W[2])/norm.vertical;
        u.yaw = (-P.yaw*e.yaw +
 RG.yaw*command_trajectory_.angular_velocity_W[2])*RAD_2_DEG/norm.rotation;

        ref_command_signals.linear.x = clamp(cos(state.orientation.z)*u.x -
 sin(state.orientation.z)*u.y,max_speed.x);
        ref_command_signals.linear.y = clamp(sin(state.orientation.z)*u.x +
 cos(state.orientation.z)*u.y,max_speed.y);
        ref_command_signals.linear.z = clamp(u.z,max_speed.z);
        ref_command_signals.angular.z = clamp(u.yaw,max_speed.yaw);
    }

}

int main(int argc, char** argv){
    ros::init(argc, argv, "proportional_controller_node");
    ros::NodeHandle nh;
    bebop_controller::ProportionalController proportional_controller;
    ros::spin();
    return 0;
}
```

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 RGyawDefaultValue

```
#define RGyawDefaultValue 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
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 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.

```


#include "sinusoidal.h"

namespace bebop_controller {

    Sinusoidal::Sinusoidal(WaypointParameters wp_params, TrajectoryParameters t_params):
    Waypoint(wp_params),
    t_params_(t_params) {}

    void Sinusoidal::Trajectory_CB(const ros::TimerEvent& event) {
        ros::Time now = ros::Time::now();
        double x, y, z, yaw;
        double dot_x, dot_y, dot_z, dot_yaw;
        double ddot_x, ddot_y, ddot_z;
        double t = now.toSec() - Initial_Time.toSec() + wp_params_.DiffTime;
        double W1, W2, W3;
        W1 = W3 = 2*M_PI/wp_params_.TrajectoryTime;
        W2 = 4*M_PI/wp_params_.TrajectoryTime;

        x = t_params_.TrajectoryDistance.x()*sin(W1*t) + t_params_.PositionBeforeTrajectory.x();
        y = t_params_.TrajectoryDistance.y()*sin(W2*t) + t_params_.PositionBeforeTrajectory.y();
        z = t_params_.TrajectoryDistance.z()*sin(W3*t) + t_params_.PositionBeforeTrajectory.z();

        dot_x = t_params_.TrajectoryDistance.x()*W1*cos(W1*t);
        dot_y = t_params_.TrajectoryDistance.y()*W2*cos(W2*t);
        dot_z = t_params_.TrajectoryDistance.z()*W3*cos(W3*t);

        ddot_x = -t_params_.TrajectoryDistance.x()*pow(W1,2)*sin(W1*t);
        ddot_y = -t_params_.TrajectoryDistance.y()*pow(W2,2)*sin(W2*t);
        ddot_z = -t_params_.TrajectoryDistance.z()*pow(W3,2)*sin(W3*t);

        switch (status)
        {
        case BeforeTrajectory:
        case AfterTrajectory:
            x = t_params_.PositionBeforeTrajectory.x();
            y = t_params_.PositionBeforeTrajectory.y();
            z = t_params_.PositionBeforeTrajectory.z();
            if (t_params_.Yaw_Enabled){
                yaw = atan2(2*t_params_.TrajectoryDistance.y(),t_params_.TrajectoryDistance.x()) +
 t_params_.Yaw_Offset;
            }
            else {
                yaw = 0;
            }
            dot_x = dot_y = dot_z = ddot_x = ddot_y = ddot_z = dot_yaw = 0;
            break;

        case Trajectory:
            if (t > wp_params_.TrajectoryTime) {
                status = AfterTrajectory;
                return;
            }
            if (t_params_.Yaw_Enabled){
                yaw = atan2(dot_y,dot_x) + t_params_.Yaw_Offset;
                dot_yaw = ((dot_x*ddot_y - dot_y*ddot_x)/pow(dot_x,2))/(1+pow(dot_y/dot_x,2));
            }
            else {
                yaw = dot_yaw = 0;
            }
            break;
        }

        Eigen::Vector3d desired_position(x, y, z);
        Eigen::Vector3d velocities(dot_x,dot_y,dot_z);
        Eigen::Vector3d accelerations(ddot_x,ddot_y,ddot_z);

        mav_msgs::EigenTrajectoryPoint point;
        point.position_W = desired_position;
        point.setFromYaw(yaw);
        point.velocity_W = velocities;
        point.setFromYawRate(dot_yaw);
        point.acceleration_W = accelerations;

        mav_msgs::msgMultiDofJointTrajectoryFromEigen(point, &position_target_);

        setpoint_pub_.publish(position_target_);
    }

}

int main(int argc, char** argv){
    ros::init(argc, argv, "sinusoidal");
    ros::NodeHandle nh;

    double X_I, Y_I, Z_I, DIST_X, DIST_Y, DIST_Z;
    WaypointParameters wp_params;
    TrajectoryParameters t_params;

    ros::param::get("~Waypoint/TimeBeforeTrajectory", wp_params.TimeBeforeTrajectory);
    ros::param::get("~Waypoint/TrajectoryTime", wp_params.TrajectoryTime);
    ros::param::get("~Waypoint/DiffTime", wp_params.DiffTime);
    ros::param::get("~Waypoint/MarginTime", wp_params.MarginTime);
    ros::param::get("~Trajectory/X_Initial", X_I);
    ros::param::get("~Trajectory/Y_Initial", Y_I);
    ros::param::get("~Trajectory/Z_Initial", Z_I);
    ros::param::get("~Trajectory/X_Distance", DIST_X);
    ros::param::get("~Trajectory/Y_Distance", DIST_Y);
    ros::param::get("~Trajectory/Z_Distance", DIST_Z);
    ros::param::get("~Yaw_Enabled", t_params.Yaw_Enabled);
    ros::param::get("~Yaw_Offset", t_params.Yaw_Offset);
    ros::param::get("~Topics/Command_Trajectory", wp_params.topic_command_trajectory);
    ros::param::get("~Topics/CSV_Begin", wp_params.topic_csv_begin);
    ros::param::get("~Topics/CSV_End", wp_params.topic_csv_end);

    t_params.PositionBeforeTrajectory = Eigen::Vector3d(X_I,Y_I,Z_I);
    t_params.TrajectoryDistance = Eigen::Vector3d(DIST_X,DIST_Y,DIST_Z);

    bebop_controller::Sinusoidal waypoint(wp_params, t_params);
    ros::spin();
    return 0;
}
```

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.

```


#include "bebop_controller/waypoint.h"

struct TrajectoryParameters {
    Eigen::Vector3d PositionBeforeTrajectory;
    Eigen::Vector3d TrajectoryDistance;
    bool Yaw_Enabled;
    double Yaw_Offset;
};

namespace bebop_controller {

    class Sinusoidal : public Waypoint {
        public:
            Sinusoidal(WaypointParameters wp_params, TrajectoryParameters t_params);

        private:
            TrajectoryParameters t_params_;
            void Trajectory_CB(const ros::TimerEvent& event) override;
    };

}
```

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
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));
```

```
        Ppos.z = P.z*sqrt(std::abs(e.z));
        Ppos.yaw = P.yaw*sqrt(std::abs(e.yaw));

        CalculateLeashLength(e, P);
        LimitPositionErrors(e);

        u.x = (-Ppos.x*e.x + RG.x*command_trajectory_.velocity_W[0])/norm.horizontal;
        u.y = (-Ppos.y*e.y + RG.y*command_trajectory_.velocity_W[1])/norm.horizontal;
        u.z = (-Ppos.z*e.z + RG.z*command_trajectory_.velocity_W[2])/norm.vertical;
        u.yaw = (-Ppos.yaw*e.yaw +
  RG.yaw*command_trajectory_.angular_velocity_W[2])*RAD_2_DEG/norm.rotation;

        ref_command_signals.linear.x = clamp(cos(state.orientation.z)*u.x -
  sin(state.orientation.z)*u.y,max_speed.x);
        ref_command_signals.linear.y = clamp(sin(state.orientation.z)*u.x +
  cos(state.orientation.z)*u.y,max_speed.y);
        ref_command_signals.linear.z = clamp(u.z,max_speed.z);
        ref_command_signals.angular.z = clamp(u.yaw,max_speed.yaw);
    }

}

int main(int argc, char** argv){
    ros::init(argc, argv, "square_root_controller_node");
    ros::NodeHandle nh;
    bebop_controller::SquareRootController square_root_controller;
    ros::spin();
    return 0;
}
```

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.

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

#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

namespace bebop_controller {

    struct Normalize {
        double horizontal;
        double vertical;
        double rotation;
    };

    class SquareRootController : public BaseController {
        public:
            SquareRootController();

        private:
            Vector4 P, RG;
            Normalize norm;
            void CalculateCommandVelocities(geometry_msgs::Twist& ref_command_signals) override;
    };

}
```

Index