ros_diff_drive

Generated by Doxygen 1.8.17

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

Module Documentation

5.1 Pre initial values of PID regulators

Values of PID parameters before the update from the dynamic reconfigure module. Advice is to keep current values.

Variables

- float config.KP_ROT = 0.0

  Rotation KP constant.
- float config.TI_ROT = 0.000001

  Rotation TI constant.
- float config.TD_ROT = 0.0

  Rotation TD constant.
- float config.INT_LIMIT_ROT = 0.0

  Limit of the control value used as protection from the wind-up - used for rotation.
- float config.P_ANG_DST = 0.0

  Gaol angle filter constant.
- float config.P_ANG_THT = 0.0

  Current angle filter constant.
- float config.KP_FWD = 0.0

  Move forward KP constant.
- float config.TI_FWD = 0.000001

  Move forward TI constant.
- float config.TD_FWD = 0.0

  Move forward TD constant.
- float config.P_FWD_DST = 0.0

  Gaol distance filter constant.
- float config.P_FWD_CUR = 0.0

  Current distance filter constant.
- float config.INT_LIMIT_FWD = 0.0

  Limit of the control value used as protection from the wind-up - used for moving forward.

5.1.1 Detailed Description

Values of PID parameters before the update from the dynamic reconfigure module. Advice is to keep current values.

5.1.2 Variable Documentation

5.1.2.1 INT_LIMIT_FWD

```
float config.INT_LIMIT_FWD = 0.0
```

Limit of the control value used as protection from the wind-up - used for moving forward.

Overwritten later by dynamic reconfigure module.

5.1.2.2 INT_LIMIT_ROT

```
float config.INT_LIMIT_ROT = 0.0
```

Limit of the control value used as protection from the wind-up - used for rotation.

Overwritten later by dynamic reconfigure module.

5.1.2.3 KP_FWD

```
float config.KP_FWD = 0.0
```

Move forward KP constant.

Overwritten later by dynamic reconfigure module.

5.1.2.4 KP_ROT

```
float config.KP_ROT = 0.0
```

Rotation KP constant.

Overwritten later by dynamic reconfigure module.

5.1.2.5 P_ANG_DST

```
float config.P_ANG_DST = 0.0
```

Gaol angle filter constant.

Overwritten later by dynamic reconfigure module.

5.1.2.6 P_ANG_THT

```
float config.P_ANG_THT = 0.0
```

Current angle filter constant.

Overwritten later by dynamic reconfigure module.

5.1.2.7 P_FWD_CUR

```
float config.P_FWD_CUR = 0.0
```

Current distance filter constant.

Overwritten later by dynamic reconfigure module.

5.1.2.8 P_FWD_DST

```
float config.P_FWD_DST = 0.0
```

Gaol distance filter constant.

Overwritten later by dynamic reconfigure module.

5.1.2.9 TD_FWD

```
float config.TD_FWD = 0.0
```

Move forward TD constant.

Overwritten later by dynamic reconfigure module.

5.1.2.10 TD_ROT

```
float config.TD_ROT = 0.0
```

Rotation TD constant.

Overwritten later by dynamic reconfigure module.

5.1.2.11 TI_FWD

```
float config.TI_FWD = 0.000001
```

Move forward TI constant.

Overwritten later by dynamic reconfigure module.

5.1.2.12 TI_ROT

```
float config.TI_ROT = 0.000001
```

Rotation TI constant.

Overwritten later by dynamic reconfigure module.

5.2 Values updated during the callback of the odometry

Variables

- float move_to_point.xInitial = 0.0

  Initial x coordinate from the start of moving forward.
- float move_to_point.yInitial = 0.0

  Initial y coordinate from the start of moving forward.

5.2.1 Detailed Description

5.3 Values updated during the callback of the odometry

These values are updated during position_callback.

Variables

- move_to_point.cur_pos = Point()

  Contains X and Y coordinates of the current position.
- move_to_point.x

  X coordinate of the current position - initialization.
- move_to_point.y

  Y coordinate of the current position - initialization.
- float move_to_point.theta = 0.0

  Represents current angle of the robot.

5.3.1 Detailed Description

These values are updated during position_callback.

5.4 Variables used for debugging purposes

If debugging only rotation/forward, they receive the value from the dynamic reconfigure.

Variables

- float move_to_point.GOAL_THETA = 0.0

  Goal angle.
- float move_to_point.GOAL_DIST = 0.0

  Goal distance.

5.4.1 Detailed Description

If debugging only rotation/forward, they receive the value from the dynamic reconfigure.

Chapter 6

Namespace Documentation

6.1 config Namespace Reference

Variables

- float controller_freq = 50.0

  Controller frequency in Hertz.
- float rot_speed_limit = 2.0

  Rotation speed limit.
- float fwd_speed_limit = 0.7

  Speed limit of moving forward.
- float angle_err_tolerance = 0.04

  Angle error tolerance in degrees.
- float dist_err_tolerance = 0.005

  Distance error tolerance.
- float KP_ROT = 0.0

  Rotation KP constant.
- float TI_ROT = 0.000001

  Rotation TI constant.
- float TD_ROT = 0.0

  Rotation TD constant.
- float INT_LIMIT_ROT = 0.0

  Limit of the control value used as protection from the wind-up - used for rotation.
- float P_ANG_DST = 0.0

  Gaol angle filter constant.
- float P_ANG_THT = 0.0

  Current angle filter constant.
- float KP_FWD = 0.0

  Move forward KP constant.
- float TI_FWD = 0.000001

  Move forward TI constant.
- float TD_FWD = 0.0

  Move forward TD constant.
- float P_FWD_DST = 0.0

  Gaol distance filter constant.
- float P_FWD_CUR = 0.0

  Current distance filter constant.
- float INT_LIMIT_FWD = 0.0

  Limit of the control value used as protection from the wind-up - used for moving forward.

6.1.1 Detailed Description

```
Static configuration of the move_to_point module
```

6.2 debug Namespace Reference

Variables

- pub_dbg_angle_err = rospy.Publisher("debug/angle_err", Float32, queue_size = 5)

  Angle error values publisher.
- pub_dbg_theta = rospy.Publisher("debug/theta", Float32, queue_size=5)

  Current angle publisher.
- pub_dbg_theta_filtr = rospy.Publisher("debug/theta_filtr", Float32, queue_size=5)

  Publisher for filtered value of the current value.
- pub_dbg_ang_to_goal = rospy.Publisher("debug/angle_to_goal", Float32, queue_size=5)

  Publisher for the goal angle.
- pub_dbg_ang_to_goal_filtr = rospy.Publisher("debug/angle_to_goal_filtr", Float32, queue_size=5)

  Publisher for the filtered value of the goal angle.
- pub_dbg_rot = rospy.Publisher("debug/rot_vel", Float32, queue_size=5)

  Publisher for the desired rotation velocity.
- pub_dbg_distance = rospy.Publisher("/debug/distance", Float32, queue_size = 5)

  Current distance publisher.
- pub_dbg_distance_filtr = rospy.Publisher("/debug/dist_filtr", Float32, queue_size = 5)

  Filtered distance publisher.
- pub_dbg_dist_to_goal = rospy.Publisher("/debug/dist_to_goal", Float32, queue_size = 5)

  Goal distance publisher.
- pub_dbg_dist_to_goal_filtr = rospy.Publisher("/debug/dist_to_goal_filtr", Float32, queue_size = 5)

  Filtered goal distance publisher.
- pub_dbg_fwd = rospy.Publisher("/debug/dist_velocity", Float32, queue_size = 5)

  Distance velocity publisher.
- pub_dbg_fwd_rot = rospy.Publisher("/debug/fwd_rot", Float32, queue_size = 5)

  Angle error during moving forward - publisher.
- pub_dbg_fwd_rot_vel = rospy.Publisher("/debug/fwd_rot_vel", Float32, queue_size = 5)

  Rotation velocity command during moving forward - publisher.

6.2.1 Detailed Description

```
Contains debug variables and publisher/subscriber definitions
```

6.3 fsm Namespace Reference

Classes

- class FsmRobot

  Finite State Machine class Provides data structure for FSM as well as main methods for normal functioning and events logging.
- class FsmState

  FSM state class.
- class FsmStates

  Enumeration containing state machine states definitions.

6.3.1 Detailed Description

```
Finite State Machine library
```

6.4 move_to_point Namespace Reference

Functions

- def angle_error_calc (target_angle, current_angle)

  Calculate rotation direction
  Positive rotation direction - clockwise
  Negative rotation direction - counter clockwise.
- def dyn_reconf_callback (config, level)

  Dynamic reconfigure callback function.
- def position_callback (msg)

  Odometry subscriber callback function.
- def goal_position_callback (msg)

  Callback function for processing goal values.
- def idle ()

  Idle function of the robot state machine.
- def rotate ()

  Rotation function of the robot state machine.
- def forward ()

  State machine functionality for moving forward.

Variables

- float T = 1.0 / controller_freq

  Controller's period in seconds.
- float xInitial = 0.0

  Initial x coordinate from the start of moving forward.
- float yInitial = 0.0

  Initial y coordinate from the start of moving forward.
- cur_pos = Point()

  Contains X and Y coordinates of the current position.
- x

  X coordinate of the current position - initialization.
- y

  Y coordinate of the current position - initialization.
- float theta = 0.0

  Represents current angle of the robot.
- goal = Point(0, 0, 0)

  Goal input coordinates Updated in goal_position_callback.
- float GOAL_THETA = 0.0

  Goal angle.
- float GOAL_DIST = 0.0

  Goal distance.
- active_goal = Point(0, 0, 0)

  Goal coordinates that are under processing.

- float angle_to_goal_filt = 0.0

  Filtered value of the goal angle.
- float theta_filt = 0.0

  Filtered value of the current angle.
- bool POSITIVE = True

  Constant for indicating positive angle.
- bool NEGATIVE = False

  Constant for indicating negative angle.
- bool theta_sign = POSITIVE

  Variable indicating signess of the current iteration angle: POSITIVE or NEGATIVE.
- bool theta_sign_prev = POSITIVE

  Variable indicating signess of the angle from the previous iteration: POSITIVE or NEGATIVE.
- float goal_distance = 0.0

  Calculated goal distance.
- float goal_distance_filt = 0.0

  Filtered value of the goal distance.
- float dist_filt = 0.0

  Filtered value of the desired distance.
- bool move_fwd_started = False

  Indicates if moving forward started or not.
- float angle_to_goal_fwd = 0.0

  Used for correcting angle error which accumulates while moving forward.
- sub_goal_position = rospy.Subscriber("/target_position/position", Pose2D, goal_position_callback)

  Goal destination subscriber.
- sub_odom = rospy.Subscriber("/m2xr_diff_drive_controller/odom", Odometry, position_callback)

  Odometry (current position) subscriber.
- pub_cmd_vel = rospy.Publisher("/m2xr_diff_drive_controller/cmd_vel", Twist, queue_size=1)

  cmd_vel publisher Used to publish desired velocity to the next layer of the control.
- r = rospy.Rate(controller_freq)

  Initialization of the speed_controller node.
- StateIdle = FsmState(FsmStates.Idle, idle)

  FSM idle state definition.
- StateRotation = FsmState(FsmStates.Rotating, rotate)

  FSM rotation state definition.
- StateForward = FsmState(FsmStates.Forward, forward)

  FSM moving forward state definition.
- list StatesList = [StateIdle, StateRotation, StateForward]

  List of permitted states.
- robot_fsm = FsmRobot("M2XR", StatesList, StatesList[0])

  FSM Initialization.
- srv__dyn_reconf = Server(DynRecPIDConfig, dyn_reconf_callback)

  Dynamic reconfigure server initialization.
- rot_pid = Regulator(KP_ROT, TI_ROT, TD_ROT, T, rot_speed_limit, INT_LIMIT_ROT)

  Normal rotation PID regulator initialization.
- fwd_pid = Regulator(KP_FWD, TI_FWD, TD_FWD, T, fwd_speed_limit, INT_LIMIT_FWD)

  Moving forward PID regulator initialization.
- fwd_pid_rot = Regulator(KP_ROT, TI_ROT, TD_ROT, T, rot_speed_limit, INT_LIMIT_ROT)

  Rotation while moving forward PID regulator initialization.

6.4.1 Detailed Description

```
Implements differential drive robot control
```

6.4.2 Function Documentation

6.4.2.1 angle_error_calc()

```
def move_to_point.angle_error_calc (
              target_angle,
              current_angle )
```

Calculate rotation direction
Positive rotation direction - clockwise
Negative rotation direction - counter clockwise.

Parameters

target_angle	Angle to be reached [degrees]
current_angle	Current angle of the robot [degrees]

Returns

Error including direction [degrees]

6.4.2.2 dyn_reconf_callback()

```
def move_to_point.dyn_reconf_callback (
              config,
              level )
```

Dynamic reconfigure callback function.

Parameters

config	Contains all dynamic parameters
level	Not used

Returns

config

6.4.2.3 forward()

```
def move_to_point.forward ( )
```

State machine functionality for moving forward.

This state controls robot when moving forward. It filters current distance and desired distance values, calculates an error, and generates desired linear speed calculated in PID routine. Desired rotation speed is then published to the velocity publisher pub_cmd_vel.

6.4.2.4 goal_position_callback()

```
def move_to_point.goal_position_callback (
              msg )
```

Callback function for processing goal values.

Parameters

msg	Message to be processed - contains desired goal position

6.4.2.5 idle()

```
def move_to_point.idle ( )
```

Idle function of the robot state machine.

In this state robot is waiting for the new command to arrive.

6.4.2.6 position_callback()

```
def move_to_point.position_callback (
              msg )
```

Odometry subscriber callback function.

Parameters

msg	The message base on Odometry message type

6.4.2.7 rotate()

```
def move_to_point.rotate ( )
```

Rotation function of the robot state machine.

This state controls robot rotation. It filters current angle and desired angle values, calculates an error in degrees and generates desired rotation speed calculated in PID routine. Desired rotation speed is then published to the velocity publisher pub_cmd_vel.

6.5 regulator Namespace Reference

Classes

- class Regulator

  Class of the regulator which contains parameters and methods which implement different control algorithms.

6.5.1 Detailed Description

```
Implements PID regulation algorithms
```

Chapter 7

Class Documentation

7.1 fsm.FsmRobot Class Reference

Finite State Machine class Provides data structure for FSM as well as main methods for normal functioning and events logging.

Public Member Functions

- def __init__ (self, name, states_list, state)

  Default constructor for FSM name: Name of the FSM.
- def switch_state (self, new_state)

  Method used for switching between states of the FSM.
- def validate_state (self, state)

  Method used for state validation.
- def default (self)

  Default method of the FSM.
- def execute (self)

  Method used to execute current_state of the FSM.

Public Attributes

- name

  Name of the FSM.
- states_list

  List of states.
- current_state

  State which is currently under the execution.
- previous_state

  Previous state.

7.1.1 Detailed Description

Finite State Machine class Provides data structure for FSM as well as main methods for normal functioning and events logging.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 __init__()

```
def fsm.FsmRobot.__init__ (
              self,
              name,
              states_list,
              state )
```

Default constructor for FSM name: Name of the FSM.

Parameters

name	Desired name of the FSM
states_list	List of permitted states
state	Desired to be current (initial) state

7.1.3 Member Function Documentation

7.1.3.1 default()

```
def fsm.FsmRobot.default (
              self )
```

Default method of the FSM.

If FSM is initialized properly, this state must not be executed!

7.1.3.2 switch_state()

```
def fsm.FsmRobot.switch_state (
              self,
              new_state )
```

Method used for switching between states of the FSM.

Parameters

new_state	State which FSM will switch to

7.1.3.3 validate_state()

```
def fsm.FsmRobot.validate_state (
              self,
              state )
```

Method used for state validation.

It basically checks if state is in the list states_list of predefined states.

Parameters

state	State to be validated

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

- /home/djordje/catkin_ws/src/ros_diff_drive/scripts/fsm.py

7.2 fsm.FsmState Class Reference

FSM state class.

Public Member Functions

- def __init__ (self, state, method)

  Constructor which contains desired state enumerator and method.

Public Attributes

- state

  state which stores enumeration value from FsmStates
- method

  method which will be executed once this state is ongoing

7.2.1 Detailed Description

FSM state class.

Contains state enumerator defined FsmStates, as well as the method which shall be executed with this state.

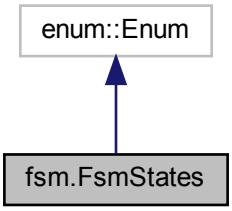
The documentation for this class was generated from the following file:

- /home/djordje/catkin_ws/src/ros_diff_drive/scripts/fsm.py

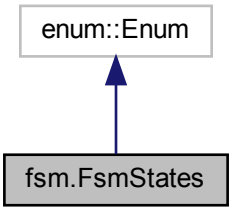
7.3 fsm.FsmStates Class Reference

Enumeration containing state machine states definitions.

Inheritance diagram for fsm.FsmStates:



Collaboration diagram for fsm.FsmStates:



Static Public Attributes

- int Default = 0

  Default state and should not be used by the user.
- int Idle = 1

  Waiting for a new command.
- int Rotating = 2

  State for the robot rotation.
- int Forward = 3

  State for moving forward.

7.3.1 Detailed Description

Enumeration containing state machine states definitions.

```
FSM states enumeration
```

7.3.2 Member Data Documentation

7.3.2.1 Default

```
int fsm.FsmStates.Default = 0  [static]
```

Default state and should not be used by the user.

Used only as initial value of previous_state.

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

- /home/djordje/catkin_ws/src/ros_diff_drive/scripts/fsm.py

7.4 regulator.Regulator Class Reference

Class of the regulator which contains parameters and methods which implement different control algorithms.

Public Member Functions

- def __init__ (self, KP, TI, TD, T, u_limit, ui_limit)
  Constructor of the regulator.
- def update_params (self, KP, TI, TD, ui_limit)
  Method for updating PID parameters.
- def pid_positional (self, error)
  Positional PID algorithm method.
- def pid_incremental (self, error)
  Incremental PID algorithm method.

Public Attributes

- KP
  KP Gain of the PID.
- KI
  KI Gain of the PID.
- KD
  KD Gain of the PID.
- u_limit
  Limit of the overall control output (only for incremental PID)
- ui_limit
  Limit of the integral controll output (only for positional PID)
- err_prev
  Error from the previous interation.
- err_p_prev
  Error from the iteration before previous one.

- u
  Calculated control output.
- T
  Period between two iteration.
- KDT
  KD ∗ T part of the PID calculation.
- KIT
  KI ∗ T part of the PID calculation.
- Ui
  Backup integral output control value.

7.4.1 Detailed Description

Class of the regulator which contains parameters and methods which implement different control algorithms.

7.4.2 Member Function Documentation

7.4.2.1 update_params()

```
def regulator.Regulator.update_params (
              self,
              KP,
              TI,
              TD,
              ui_limit )
```

Method for updating PID parameters.

Note: Should be used during debugging and PID setup Not intended to be used during normal operation because of the runtime consumption.

7.4.3 Member Data Documentation

7.4.3.1 err_p_prev

```
regulator.Regulator.err_p_prev
```

Error from the iteration before previous one.

Backup.

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

- /home/djordje/catkin_ws/src/ros_diff_drive/scripts/regulator.py

Index