DuoArm

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Class Documentation

3.1 hardware_center.motor_control.motorControl Class Reference

Public Member Functions

- def __init__ (self)
- def init_communication (self)
- def load_and_set_boundaries (self)
- def setup_servos_and_queues (self)
- def init_servo_threads (self)
- def servo_control_loop (self, servo_key)
- def control_servo_speed (self, servo_key, target_angle)
- def follow_path_callback (self, msg)
- def follow_path (self, angles, num_cycles)
- def limp_and_set_origin (self, msg)
- def publish_joint_angles (self, servo_key, angle)
- def calc_position (self, angle)
- def calc_angle (self, position)
- def read_angles_callback (self)
- def manualy_callback (self)
- def start_read_callback (self, msg)
- def check_state_and_stop (self)
- def state_callback (self, msg)

Public Attributes

- **real_time_qos**
- **boundaries**
- **threads**
- **lock**
- **current_target**
- **state**
- **total_movements**
- **completed_movements**
- **current_movement_nr**
- **actual_joint_angles_publisher**
- **joint_angles_publisher**

- **start_read_sub**
- **manual_readings_pub**
- **limp_and_reset_origin_sub**
- **read_angels_sub**
- **joint_angles_subscription**
- **path_done_pub**
- **state_subscription**
- **servos**
- **angles_queues**

3.1.1 Detailed Description

```
@class motorControl
@brief Used to controll and read values form LSS servo motor during mapping and path execution

The motorControl node is designed to interface with LSS servos and manage their operations
through various topics. This node handles real-time control, state management,
and boundary safety for motor operations within this ROS system.
```

3.1.2 Constructor & Destructor Documentation

3.1.2.1 __init__()

```
def hardware_center.motor_control.motorControl.__init__ (
              self )
```

```
Initialize the motor control node, setting up publishers, subscribers, and servo control threads.
```

3.1.3 Member Function Documentation

3.1.3.1 calc_angle()

```
def hardware_center.motor_control.motorControl.calc_angle (
              self,
              position )
```

```
Calculate the joint angle based on the servo position reading. position is LSS servomotors way of describing a
Converts raw position data from the servo to a human-readable angle format.
```

3.1.3.2 calc_position()

```
def hardware_center.motor_control.motorControl.calc_position (
              self,
              angle )
```

Calculate the joint position. position is LSS servomotors way of describing angle, position = angle*10
this is used when moving the servos

3.1.3.3 check_state_and_stop()

```
def hardware_center.motor_control.motorControl.check_state_and_stop (
              self )
```

Check the system's state and stop all servo movements if necessary.
sadly we couldnt get the node to recive messages whils operating the servos, and therefor this funciton is use

3.1.3.4 control_servo_speed()

```
def hardware_center.motor_control.motorControl.control_servo_speed (
              self,
              servo_key,
              target_angle )
```

Control the speed of a servo to reach a specified target angle.
Implements PID control to reach the target angle by comparing current and target angle.

3.1.3.5 follow_path()

```
def hardware_center.motor_control.motorControl.follow_path (
              self,
              angles,
              num_cycles )
```

Reads the target angles and puts them in the queue for independent servo threads to read
Keeps track of movement iteration and correctly ending a path execution

3.1.3.6 follow_path_callback()

```
def hardware_center.motor_control.motorControl.follow_path_callback (
              self,
              msg )
```

Callback for moving servos according to a set of predefined angles.
Handles the reception of an array of angles, setting the path for the servos to follow, and number of times to

3.1.3.7 init_communication()

```
def hardware_center.motor_control.motorControl.init_communication (
              self )
```

Initialize communication for the node.
Setup all necessary publishers and subscribers for sending and receiving commands
and data to and from other components of ROS system.

3.1.3.8 init_servo_threads()

```
def hardware_center.motor_control.motorControl.init_servo_threads (
              self )
```

Initialize threads for controlling servos independently.

3.1.3.9 limp_and_set_origin()

```
def hardware_center.motor_control.motorControl.limp_and_set_origin (
              self,
              msg )
```

Makes the servos go limp and sets a new origin after a delay.
Used at the start of mapping process

3.1.3.10 load_and_set_boundaries()

```
def hardware_center.motor_control.motorControl.load_and_set_boundaries (
              self )
```

Load and set boundaries for servo movements from a JSON configuration file.
Ensures that servos operate within safe operational limits to avoid mechanical damage.

3.1.3.11 manualy_callback()

```
def hardware_center.motor_control.motorControl.manualy_callback (
              self )
```

Read the servos when during testing and debugging

3.1.3.12 publish_joint_angles()

```
def hardware_center.motor_control.motorControl.publish_joint_angles (
              self,
              servo_key,
              angle )
```

Publish the current joint angles to display node for visualization, sadly didnt get it working.

3.1.3.13 read_angles_callback()

```
def hardware_center.motor_control.motorControl.read_angles_callback (
              self )
```

Read the servos when during mapping, is called when arm state='map' and map button is pressed

3.1.3.14 servo_control_loop()

```
def hardware_center.motor_control.motorControl.servo_control_loop (
              self,
              servo_key )
```

Control loop for servos.
Handles target angles for a single servo from its angles queue.

3.1.3.15 setup_servos_and_queues()

```
def hardware_center.motor_control.motorControl.setup_servos_and_queues (
              self )
```

Initializes servo objects and angles queues for asynchronous operation.

3.1.3.16 start_read_callback()

```
def hardware_center.motor_control.motorControl.start_read_callback (
              self,
              msg )
```

callback for starting test reading

3.1.3.17 state_callback()

```
def hardware_center.motor_control.motorControl.state_callback (
              self,
              msg )
```

Callback for updating the system's operational state.
Receives state updates from the action controller.

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

- src/hardware_center/hardware_center/motor_control.py