Kinematic controller

Generated by Doxygen 1.9.8

1 Namespace Index **1**

1.1 Namespace List 1

2 Hierarchical Index **3**

2.1 Class Hierarchy 3

3 Data Structure Index **5**

3.1 Data Structures 5

4 File Index **7**

4.1 File List 7

5 Namespace Documentation **9**

5.1 commands Namespace Reference 9

5.1.1 Variable Documentation 9

5.1.1.1 CLOSE_GRIPPER 9

5.1.1.2 EMPTY 9

5.1.1.3 HOME_GRIPPER 9

5.1.1.4 HOME_PRESSER 10

5.1.1.5 NEW_TEST 10

5.1.1.6 PREPARE_PRESSING 10

5.1.1.7 PRESS_ALLERGEN 10

5.1.1.8 RESPONSE 10

5.2 communication Namespace Reference 10

5.2.1 Variable Documentation 11

5.2.1.1 com_test 11

5.2.1.2 SERIAL_COMMUNICATION_UNKNOWN_RESPONSE_ERROR_MSG 11

5.3 gui Namespace Reference 11

5.3.1 Variable Documentation 11

5.3.1.1 test 11

5.4 main Namespace Reference 11

5.4.1 Function Documentation 12

5.4.1.1 communication_multi_thread() 12

5.4.2 Variable Documentation 12

5.4.2.1 ALLERGEN_AMOUNT 12

5.4.2.2 BOTH 12

5.4.2.3 END_EFFECTOR_TILT_PLACE 12

5.4.2.4 END_EFFECTOR_TILT_TAKE 13

5.4.2.5 main 13

5.4.2.6 PATCH_TEST_CORNER_COORDINATES 13

5.4.2.7 performing_command 13

5.4.2.8 SIMULATING 13

5.4.2.9 SPEED 13

5.4.2.10 STORAGE_OFFSET_X 13

5.4.2.11 SYRINGE_COORDINATES 14
5.4.2.12 SYRINGE_MOVEMENT_Z 14
5.4.2.13 SYRINGE_OFFSET_X 14
5.4.2.14 Z_OFFSET 14
5.5 robot_controller Namespace Reference 14
5.6 robot_movement Namespace Reference 14
5.6.1 Function Documentation 15
5.6.1.1 best_end_joint() 15
5.6.1.2 get_rot_x_matrix() 16
5.6.1.3 get_rot_y_matrix() 16
5.6.1.4 get_rot_z_matrix() 17
5.6.1.5 joint_matrix() 17
5.6.1.6 orientation_degree_to_radians() 18
5.6.1.7 radians_to_degree() 18
5.6.1.8 radians_to_degree_list() 18
5.6.1.9 transform_list_into_range() 19
5.6.2 Variable Documentation 19
5.6.2.1 COORDINATE_SYNTAX_ERROR_MSG 19
5.6.2.2 INVALID_TIME_INCREASE_ERROR_MSG 19
5.6.2.3 JOINT_SYNTAX_ERROR_MSG 19
5.6.2.4 MAX_ANGLE_LINK_1_5 20
5.6.2.5 MAX_ANGLE_LINK_6 20
5.6.2.6 MIN_ANGLE_LINK_1_5 20
5.6.2.7 MIN_ANGLE_LINK_6 20
5.6.2.8 MOVE_NOT_POSSIBLE_ERROR_MSG 20
5.6.2.9 robot_cartesian_position 20
5.6.2.10 robot_joint_position 20
5.6.2.11 TAKES_ONLY_TWO_COORDINATES_ERROR_MSG 21
5.7 test Namespace Reference 21
5.7.1 Function Documentation 21
5.7.1.1 mySqrt() 21
5.7.2 Variable Documentation 22
5.7.2.1 testVal 22

6 Data Structure Documentation **23**
6.1 communication.CommunicationProtocol Class Reference 23
6.1.1 Detailed Description 23
6.1.2 Constructor & Destructor Documentation 23
6.1.2.1 __init__() 23
6.1.3 Member Function Documentation 24
6.1.3.1 send_command() 24
6.1.4 Field Documentation 24

6.1.4.1 baudrate . . . 24
6.1.4.2 com . . . 24
6.1.4.3 connection . . . 24
6.1.4.4 timeout . . . 24
6.2 communication.SerialCommunicationUnknownResponse Class Reference . . . 25
6.2.1 Detailed Description . . . 25
6.3 gui.GUI Class Reference . . . 26
6.3.1 Detailed Description . . . 26
6.3.2 Constructor & Destructor Documentation . . . 26
6.3.2.1 __init__() . . . 26
6.3.3 Member Function Documentation . . . 27
6.3.3.1 both_pressed() . . . 27
6.3.3.2 robot_pressed() . . . 27
6.3.3.3 run() . . . 27
6.3.3.4 simulation_pressed() . . . 28
6.3.3.5 start_pressed() . . . 28
6.3.3.6 stop_pressed() . . . 28
6.3.3.7 stop_pressed_init() . . . 29
6.3.4 Field Documentation . . . 29
6.3.4.1 button_SELECT_BOTH . . . 29
6.3.4.2 button_SELECT_ROBOT . . . 29
6.3.4.3 button_SELECT_SIMULATION . . . 29
6.3.4.4 button_START . . . 30
6.3.4.5 button_STOP . . . 30
6.3.4.6 light_SELECT_BOTH . . . 30
6.3.4.7 light_SELECT_ROBOT . . . 30
6.3.4.8 light_SELECT_SIMULATION . . . 30
6.3.4.9 mode . . . 30
6.3.4.10 pressed_buttons . . . 31
6.3.4.11 root . . . 31
6.3.4.12 running . . . 31
6.3.4.13 title . . . 31
6.4 gui.Light Class Reference . . . 32
6.4.1 Detailed Description . . . 32
6.4.2 Constructor & Destructor Documentation . . . 33
6.4.2.1 __init__() . . . 33
6.4.3 Member Function Documentation . . . 33
6.4.3.1 turn_off() . . . 33
6.4.3.2 turn_on() . . . 33
6.4.3.3 turn_ready() . . . 33
6.4.4 Field Documentation . . . 34
6.4.4.1 oval_id . . . 34

6.5 main.Main Class Reference 34
6.5.1 Detailed Description 34
6.5.2 Constructor & Destructor Documentation 34
6.5.2.1 __init__() 34
6.5.3 Member Function Documentation 35
6.5.3.1 create_gui() 35
6.5.3.2 dispense_allergen() 35
6.5.3.3 place_allergen() 35
6.5.3.4 run() 36
6.5.3.5 set_mode() 37
6.5.3.6 take_allergen() 38
6.5.4 Field Documentation 38
6.5.4.1 communication 38
6.5.4.2 gui 38
6.5.4.3 gui_created 38
6.5.4.4 gui_thread 39
6.5.4.5 patch_test_config 39
6.5.4.6 robot 39
6.6 robot_controller.RobotControllerSettings Class Reference 39
6.6.1 Detailed Description 39
6.6.2 Constructor & Destructor Documentation 40
6.6.2.1 __init__() 40
6.6.3 Field Documentation 40
6.6.3.1 dh_values 40
6.6.3.2 linear_frequency 40
6.6.3.3 update_frequency 40
6.7 robot_movement.CoordinateSyntaxError Class Reference 41
6.7.1 Detailed Description 41
6.8 robot_movement.InvalidTimeIncrease Class Reference 42
6.8.1 Detailed Description 42
6.8.2 Field Documentation 43
6.8.2.1 suppress 43
6.9 robot_movement.JointSyntaxError Class Reference 43
6.9.1 Detailed Description 44
6.10 robot_movement.Kinematics Class Reference 44
6.10.1 Detailed Description 45
6.10.2 Constructor & Destructor Documentation 45
6.10.2.1 __init__() 45
6.10.3 Member Function Documentation 45
6.10.3.1 get_joint_values() 45
6.10.3.2 kin_calculate_theta_1() 46
6.10.3.3 kin_calculate_theta_2_3() 47

6.10.3.4 kin_calculate_theta_4() 48
6.10.3.5 kin_calculate_theta_5() 48
6.10.3.6 kin_calculate_theta_6() 49
6.10.4 Field Documentation 49
6.10.4.1 dh_value_a 49
6.10.4.2 dh_value_alpha 49
6.10.4.3 dh_value_d 49
6.10.4.4 dh_value_theta 50
6.10.4.5 robot_controller_settings 50
6.11 robot_movement.MoveJ Class Reference 50
6.11.1 Detailed Description 51
6.11.2 Constructor & Destructor Documentation 51
6.11.2.1 __init__() 51
6.11.3 Member Function Documentation 52
6.11.3.1 convert_coordinates_to_joint_movements() 52
6.11.3.2 move_joint() 53
6.11.4 Field Documentation 53
6.11.4.1 joint_matrix 53
6.11.4.2 pos_w_time 53
6.11.4.3 robot 54
6.11.4.4 start_joint_values 54
6.12 robot_movement.MoveL Class Reference 54
6.12.1 Detailed Description 56
6.12.2 Constructor & Destructor Documentation 56
6.12.2.1 __init__() 56
6.12.3 Member Function Documentation 56
6.12.3.1 get_linear_via_points() 56
6.12.4 Field Documentation 57
6.12.4.1 move_time 57
6.12.4.2 pos_w_time 57
6.12.4.3 pos_w_time_linear 57
6.12.4.4 robot 57
6.12.4.5 start_pos_cartesian 57
6.13 robot_movement.MoveNotPossible Class Reference 58
6.13.1 Detailed Description 58
6.14 robot_movement.Robot Class Reference 58
6.14.1 Detailed Description 59
6.14.2 Constructor & Destructor Documentation 59
6.14.2.1 __init__() 59
6.14.3 Member Function Documentation 59
6.14.3.1 set_joints() 59
6.14.4 Field Documentation 60

6.14.4.1 max_joint_speed 60
6.14.4.2 prev_position 60
6.14.4.3 rdk_link 60
6.14.4.4 robot_settings 60
6.14.4.5 time_between_move 60
6.15 robot_movement.RobotMyCobot Class Reference 60
6.15.1 Detailed Description 61
6.15.2 Constructor & Destructor Documentation 61
6.15.2.1 __init__() 61
6.15.3 Member Function Documentation 61
6.15.3.1 set_joints() 61
6.15.4 Field Documentation 61
6.15.4.1 robo_link 61
6.16 robot_movement.RobotMyCobotAndSim Class Reference 62
6.16.1 Detailed Description 62
6.16.2 Constructor & Destructor Documentation 62
6.16.2.1 __init__() 62
6.16.3 Member Function Documentation 62
6.16.3.1 set_joints() 62
6.16.4 Field Documentation 63
6.16.4.1 rdk_link 63
6.16.4.2 robo_link 63
6.17 robot_movement.TakesOnlyTwoCoordinates Class Reference 63
6.17.1 Detailed Description 64

7 File Documentation **65**
7.1 commands.py File Reference 65
7.2 commands.py 65
7.3 communication.py File Reference 65
7.4 communication.py 66
7.5 gui.py File Reference 66
7.6 gui.py 67
7.7 main.py File Reference 69
7.8 main.py 70
7.9 robot_controller.py File Reference 73
7.10 robot_controller.py 73
7.11 robot_movement.py File Reference 74
7.12 robot_movement.py 75
7.13 test.py File Reference 84
7.14 test.py 84

Index **87**

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

commands 9
communication 10
gui 11
main 11
robot_controller 14
robot_movement 14
test 21

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

- Canvas
  - gui.Light 32
- communication.CommunicationProtocol 23
- Exception
  - communication.SerialCommunicationUnknownResponse 25
  - robot_movement.CoordinateSyntaxError 41
  - robot_movement.InvalidTimeIncrease 42
  - robot_movement.JointSyntaxError 43
  - robot_movement.MoveNotPossible 58
  - robot_movement.TakesOnlyTwoCoordinates 63
- gui.GUI 26
- main.Main 34
- robot_controller.RobotControllerSettings 39
- robot_movement.Kinematics 44
  - robot_movement.MoveJ 50
    - robot_movement.MoveL 54
- robot_movement.Robot 58
- robot_movement.RobotMyCobot 60
- robot_movement.RobotMyCobotAndSim 62

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

communication.CommunicationProtocol 23
communication.SerialCommunicationUnknownResponse 25
gui.GUI 26
gui.Light 32
main.Main 34
robot_controller.RobotControllerSettings 39
robot_movement.CoordinateSyntaxError 41
robot_movement.InvalidTimeIncrease 42
robot_movement.JointSyntaxError 43
robot_movement.Kinematics 44
robot_movement.MoveJ 50
robot_movement.MoveL 54
robot_movement.MoveNotPossible 58
robot_movement.Robot 58
robot_movement.RobotMyCobot 60
robot_movement.RobotMyCobotAndSim 62
robot_movement.TakesOnlyTwoCoordinates 63

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

commands.py 65
communication.py 65
gui.py 66
main.py 69
robot_controller.py 73
robot_movement.py 74
test.py 84

Chapter 5

Namespace Documentation

5.1 commands Namespace Reference

Variables

- str HOME_GRIPPER = "home -g"
- str CLOSE_GRIPPER = "close"
- str HOME_PRESSER = "home -p"
- str PREPARE_PRESSING = "prepare pressing"
- str PRESS_ALLERGEN = "press -"
- str NEW_TEST = "new test"
- str RESPONSE = "b'Done'"
- str EMPTY = "b''"

5.1.1 Variable Documentation

5.1.1.1 CLOSE_GRIPPER

```
str commands.CLOSE_GRIPPER = "close"
```

Definition at line 2 of file commands.py.

5.1.1.2 EMPTY

```
str commands.EMPTY = "b''"
```

Definition at line 11 of file commands.py.

5.1.1.3 HOME_GRIPPER

```
str commands.HOME_GRIPPER = "home -g"
```

Definition at line 1 of file commands.py.

5.1.1.4 HOME_PRESSER

```
str commands.HOME_PRESSER = "home -p"
```

Definition at line 4 of file commands.py.

5.1.1.5 NEW_TEST

```
str commands.NEW_TEST = "new test"
```

Definition at line 8 of file commands.py.

5.1.1.6 PREPARE_PRESSING

```
str commands.PREPARE_PRESSING = "prepare pressing"
```

Definition at line 5 of file commands.py.

5.1.1.7 PRESS_ALLERGEN

```
str commands.PRESS_ALLERGEN = "press -"
```

Definition at line 6 of file commands.py.

5.1.1.8 RESPONSE

```
str commands.RESPONSE = "b'Done'"
```

Definition at line 10 of file commands.py.

5.2 communication Namespace Reference

Data Structures

- class CommunicationProtocol
- class SerialCommunicationUnknownResponse

Variables

- str SERIAL_COMMUNICATION_UNKNOWN_RESPONSE_ERROR_MSG = "SerialCommunication↩
UnknownResponse: The received serial response was not valid."
- com_test = CommunicationProtocol("COM3", 9600, 0.1)

5.2.1 Variable Documentation

5.2.1.1 com_test

```
communication.com_test = CommunicationProtocol("COM3", 9600, 0.1)
```

Definition at line 40 of file communication.py.

5.2.1.2 SERIAL_COMMUNICATION_UNKNOWN_RESPONSE_ERROR_MSG

```
str communication.SERIAL_COMMUNICATION_UNKNOWN_RESPONSE_ERROR_MSG = "SerialCommunication↩
UnknownResponse:  The received serial response was not valid."
```

Definition at line 6 of file communication.py.

5.3 gui Namespace Reference

Data Structures

- class GUI
- class Light

Variables

- test = GUI()

5.3.1 Variable Documentation

5.3.1.1 test

```
gui.test = GUI()
```

Definition at line 198 of file gui.py.

5.4 main Namespace Reference

Data Structures

- class Main

Functions

- communication_multi_thread (communicator, command)

Variables

- bool BOTH = True
- bool SIMULATING = False
- int Z_OFFSET = 0
- int STORAGE_OFFSET_X = 30
- int SYRINGE_OFFSET_X = -18
- int SYRINGE_MOVEMENT_Z = 25
- int SPEED = 1/0.5
- ALLERGEN_AMOUNT = str(150)
- int END_EFFECTOR_TILT_TAKE = 85
- int END_EFFECTOR_TILT_PLACE = 89
- list PATCH_TEST_CORNER_COORDINATES = [-215, -265, 215]
- list SYRINGE_COORDINATES
- bool performing_command = False
- main = Main()

5.4.1 Function Documentation

5.4.1.1 communication_multi_thread()

```
main.communication_multi_thread (
              communicator,
              command )
```

```
Executes a command on the arduino using multi_threading
:param communicator: instance of the communication class
:param command: command to execute.
```

Definition at line 46 of file main.py.

5.4.2 Variable Documentation

5.4.2.1 ALLERGEN_AMOUNT

```
main.ALLERGEN_AMOUNT = str(150)
```

Definition at line 21 of file main.py.

5.4.2.2 BOTH

```
bool main.BOTH = True
```

Definition at line 14 of file main.py.

5.4.2.3 END_EFFECTOR_TILT_PLACE

```
int main.END_EFFECTOR_TILT_PLACE = 89
```

Definition at line 23 of file main.py.

5.4.2.4 END_EFFECTOR_TILT_TAKE

```
int main.END_EFFECTOR_TILT_TAKE = 85
```

Definition at line 22 of file main.py.

5.4.2.5 main

```
main.main = Main()
```

Definition at line 250 of file main.py.

5.4.2.6 PATCH_TEST_CORNER_COORDINATES

```
list main.PATCH_TEST_CORNER_COORDINATES = [-215, -265, 215]
```

Definition at line 26 of file main.py.

5.4.2.7 performing_command

```
bool main.performing_command = False
```

Definition at line 43 of file main.py.

5.4.2.8 SIMULATING

```
bool main.SIMULATING = False
```

Definition at line 15 of file main.py.

5.4.2.9 SPEED

```
int main.SPEED = 1/0.5
```

Definition at line 20 of file main.py.

5.4.2.10 STORAGE_OFFSET_X

```
int main.STORAGE_OFFSET_X = 30
```

Definition at line 17 of file main.py.

5.4.2.11 SYRINGE_COORDINATES

```
list main.SYRINGE_COORDINATES
```

Initial value:

```
=  [
     [305, 169.5+5, 153.5+5],
     [310, 94.5, 153.5+5],
     [310, 19.5, 153.5+5],
     [310, -65.5, 153.5+5],
     [305, -140.5, 153.5+5],
     [310, 132+5, 78],
     [310, 57, 76],
     [310, -28-5, 76],
     [310, -106-5, 76],
     [310, -184-5, 78]
]
```

Definition at line 29 of file main.py.

5.4.2.12 SYRINGE_MOVEMENT_Z

```
int main.SYRINGE_MOVEMENT_Z = 25
```

Definition at line 19 of file main.py.

5.4.2.13 SYRINGE_OFFSET_X

```
int main.SYRINGE_OFFSET_X = -18
```

Definition at line 18 of file main.py.

5.4.2.14 Z_OFFSET

```
int main.Z_OFFSET = 0
```

Definition at line 16 of file main.py.

5.5 robot_controller Namespace Reference

Data Structures

- class RobotControllerSettings

5.6 robot_movement Namespace Reference

Data Structures

- class CoordinateSyntaxError
- class InvalidTimeIncrease
- class JointSyntaxError
- class Kinematics
- class MoveJ
- class MoveL
- class MoveNotPossible
- class Robot
- class RobotMyCobot
- class RobotMyCobotAndSim
- class TakesOnlyTwoCoordinates

Functions

- joint_matrix (alpha, a, d, theta)
- get_rot_x_matrix (angle)
- get_rot_y_matrix (angle)
- get_rot_z_matrix (angle)
- orientation_degree_to_radians (orientation)
- radians_to_degree (angle)
- radians_to_degree_list (list_radians)
- transform_list_into_range (untransformed_list, min_value, max_value, adjuster)
- best_end_joint (current_joint, joints)

Variables

- int MIN_ANGLE_LINK_1_5 = -165 / 180 ∗ m.pi
- int MAX_ANGLE_LINK_1_5 = 165 / 180 ∗ m.pi
- int MIN_ANGLE_LINK_6 = -175 / 180 ∗ m.pi
- int MAX_ANGLE_LINK_6 = 175 / 180 ∗ m.pi
- str MOVE_NOT_POSSIBLE_ERROR_MSG = "MoveNotPossible: The given movement command can not be executed by the program. Please make sure that it dies not result in a singularity"
- str JOINT_SYNTAX_ERROR_MSG = "JointSyntaxError: The syntax must be [Joint1, joint2, joint3, joint4, joint5, joint6] for a joint"
- str TAKES_ONLY_TWO_COORDINATES_ERROR_MSG = "TakesOnlyTwoCoordinates: This class takes only two coordinates."
- str COORDINATE_SYNTAX_ERROR_MSG = "CoordinateSyntaxError: The syntax must be: [x, y, z, alpha, beta, gamma, time] for a coordinate."
- str INVALID_TIME_INCREASE_ERROR_MSG = "InvalidTimeIncrease: Later coordinates must have a higher time than those before or the robot will not be able to move to the given point."
- list robot_joint_position = [107.38158881112184, 46.157985403753784, 103.3563681945703, 30.↩ 485646401675925, 107.38158881112184, 135.0]
- list robot_cartesian_position = [280, 132, 74, -90, 135, -90]

5.6.1 Function Documentation

5.6.1.1 best_end_joint()

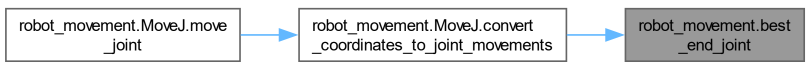
```
robot_movement.best_end_joint (
              current_joint,
              joints )
```

```
Takes the current joint position and a list containing of possible joint positions and returns the closest pos
:param current_joint: The current position of the robots joint.
:param joints: The possible joint positions for a point.
:return: The closest joint position to the current joint position
```

Definition at line 195 of file robot_movement.py.

Referenced by robot_movement.MoveJ.convert_coordinates_to_joint_movements().

Here is the caller graph for this function:


robot_movement.MoveJ.move _joint
robot_movement.MoveJ.convert _coordinates_to_joint_movements
robot_movement.best _end_joint

5.6.1.2 get_rot_x_matrix()

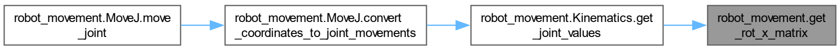
```
robot_movement.get_rot_x_matrix (
              angle )
```

```
Get rotation matrix around x axis.
:param angle: Angle the rotation matrix should rotate an object.
:return: Returns the rotation matrix.
```

Definition at line 91 of file robot_movement.py.

Referenced by robot_movement.Kinematics.get_joint_values().

Here is the caller graph for this function:


robot_movement.MoveJ.move_joint
robot_movement.MoveJ.convert_coordinates_to_joint_movements
robot_movement.Kinematics.get_joint_values
robot_movement.get_rot_x_matrix

5.6.1.3 get_rot_y_matrix()

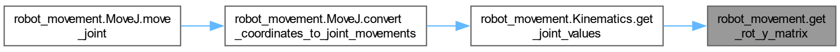
```
robot_movement.get_rot_y_matrix (
              angle )
```

```
Get rotation matrix around y axis.
:param angle: Angle the rotation matrix should rotate an object.
:return: Returns the rotation matrix.
```

Definition at line 105 of file robot_movement.py.

Referenced by robot_movement.Kinematics.get_joint_values().

Here is the caller graph for this function:


robot_movement.MoveJ.move_joint
robot_movement.MoveJ.convert_coordinates_to_joint_movements
robot_movement.Kinematics.get_joint_values
robot_movement.get_rot_y_matrix

5.6.1.4 get_rot_z_matrix()

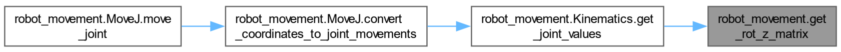
```
robot_movement.get_rot_z_matrix (
              angle )
```

```
Get rotation matrix around z axis.
:param angle: Angle the rotation matrix should rotate an object.
:return: Returns the rotation matrix.
```

Definition at line 119 of file robot_movement.py.

Referenced by robot_movement.Kinematics.get_joint_values().

Here is the caller graph for this function:


robot_movement.MoveJ.move_joint
robot_movement.MoveJ.convert_coordinates_to_joint_movements
robot_movement.Kinematics.get_joint_values
robot_movement.get_rot_z_matrix

5.6.1.5 joint_matrix()

```
robot_movement.joint_matrix (
              alpha,
              a,
              d,
              theta )
```

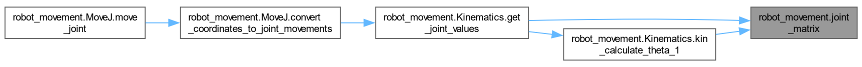
```
Function to create transformation matrix.

:param alpha: alpha value from David Hartenberg parameters.
:param a: a value from David Hartenberg parameters.
:param d: d value from David Hartenberg parameters.
:param theta: theta value from David Hartenberg parameters.
:return: returns a 4x4 matrix containing the transformation matrix for the given parameters.
```

Definition at line 73 of file robot_movement.py.

Referenced by robot_movement.Kinematics.get_joint_values(), and robot_movement.Kinematics.kin_calculate_theta_1().

Here is the caller graph for this function:


robot_movement.MoveJ.move_joint
robot_movement.MoveJ.convert_coordinates_to_joint_movements
robot_movement.Kinematics.get_joint_values
robot_movement.Kinematics.kin_calculate_theta_1
robot_movement.joint_matrix

5.6.1.6 orientation_degree_to_radians()

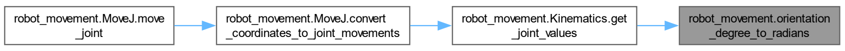
```
robot_movement.orientation_degree_to_radians (
              orientation )
```

```
Converts angle from degree to radians
:param orientation: Angle in degree.
:return: Angle in radians
```

Definition at line 133 of file robot_movement.py.

Referenced by robot_movement.Kinematics.get_joint_values().

Here is the caller graph for this function:


robot_movement.MoveJ.move_joint
robot_movement.MoveJ.convert_coordinates_to_joint_movements
robot_movement.Kinematics.get_joint_values
robot_movement.orientation_degree_to_radians

5.6.1.7 radians_to_degree()

```
robot_movement.radians_to_degree (
              angle )
```

```
Converts angle from radians to degree.
:param angle: Angle in radians.
:return: Angle in degree.
```

Definition at line 145 of file robot_movement.py.

5.6.1.8 radians_to_degree_list()

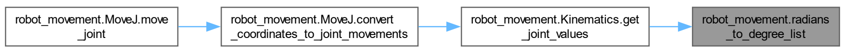
```
robot_movement.radians_to_degree_list (
              list_radians )
```

```
Converts radians in a list to degree
:param list_radians: List containing angles in radians
:return: List containing angles in degrees
```

Definition at line 156 of file robot_movement.py.

Referenced by robot_movement.Kinematics.get_joint_values().

Here is the caller graph for this function:


robot_movement.MoveJ.move_joint
robot_movement.MoveJ.convert_coordinates_to_joint_movements
robot_movement.Kinematics.get_joint_values
robot_movement.radians_to_degree_list

5.6.1.9 transform_list_into_range()

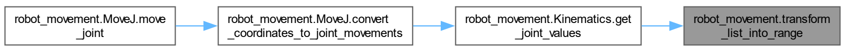
```
robot_movement.transform_list_into_range (
              untransformed_list,
              min_value,
              max_value,
              adjuster )
```

```
Takes a list containing angles and checks if they are in a given range.
:param untransformed_list: List containing angles
:param min_value: Min value in range
:param max_value: Max value in range
:param adjuster: How much the values may be adjusted
:return: Transformed list. Some of the elements may have been changed to False if the given value is not in th
```

Definition at line 170 of file robot_movement.py.

Referenced by robot_movement.Kinematics.get_joint_values().

Here is the caller graph for this function:


robot_movement.MoveJ.move_joint
robot_movement.MoveJ.convert_coordinates_to_joint_movements
robot_movement.Kinematics.get_joint_values
robot_movement.transform_list_into_range

5.6.2 Variable Documentation

5.6.2.1 COORDINATE_SYNTAX_ERROR_MSG

```
str robot_movement.COORDINATE_SYNTAX_ERROR_MSG = "CoordinateSyntaxError:  The syntax must be:
[x, y, z, alpha, beta, gamma, time] for a coordinate."
```

Definition at line 17 of file robot_movement.py.

5.6.2.2 INVALID_TIME_INCREASE_ERROR_MSG

```
str robot_movement.INVALID_TIME_INCREASE_ERROR_MSG = "InvalidTimeIncrease:  Later coordinates
must have a higher time than those before or the robot will not be able to move to the given
point."
```

Definition at line 18 of file robot_movement.py.

5.6.2.3 JOINT_SYNTAX_ERROR_MSG

```
str robot_movement.JOINT_SYNTAX_ERROR_MSG = "JointSyntaxError:  The syntax must be [Joint1,
joint2, joint3, joint4, joint5, joint6] for a joint"
```

Definition at line 15 of file robot_movement.py.

5.6.2.4 MAX_ANGLE_LINK_1_5

```
int robot_movement.MAX_ANGLE_LINK_1_5 = 165 / 180 * m.pi
```

Definition at line 10 of file robot_movement.py.

5.6.2.5 MAX_ANGLE_LINK_6

```
int robot_movement.MAX_ANGLE_LINK_6 = 175 / 180 * m.pi
```

Definition at line 12 of file robot_movement.py.

5.6.2.6 MIN_ANGLE_LINK_1_5

```
int robot_movement.MIN_ANGLE_LINK_1_5 = -165 / 180 * m.pi
```

Definition at line 9 of file robot_movement.py.

5.6.2.7 MIN_ANGLE_LINK_6

```
int robot_movement.MIN_ANGLE_LINK_6 = -175 / 180 * m.pi
```

Definition at line 11 of file robot_movement.py.

5.6.2.8 MOVE_NOT_POSSIBLE_ERROR_MSG

```
str robot_movement.MOVE_NOT_POSSIBLE_ERROR_MSG = "MoveNotPossible:  The given movement command
can not be executed by the program.  Please make sure that it dies not result in a singularity"
```

Definition at line 14 of file robot_movement.py.

5.6.2.9 robot_cartesian_position

```
list robot_movement.robot_cartesian_position = [280, 132, 74, -90, 135, -90]
```

Definition at line 22 of file robot_movement.py.

5.6.2.10 robot_joint_position

```
list robot_movement.robot_joint_position = [107.38158881112184, 46.157985403753784, 103.↩
3563681945703, 30.485646401675925, 107.38158881112184, 135.0]
```

Definition at line 21 of file robot_movement.py.

5.6.2.11 TAKES_ONLY_TWO_COORDINATES_ERROR_MSG

```
str robot_movement.TAKES_ONLY_TWO_COORDINATES_ERROR_MSG = "TakesOnlyTwoCoordinates:  This
class takes only two coordinates."
```

Definition at line 16 of file robot_movement.py.

5.7 test Namespace Reference

Functions

- mySqrt (number, guess, step, tol)

Variables

- int testVal = 9

5.7.1 Function Documentation

5.7.1.1 mySqrt()

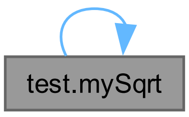
```
test.mySqrt (
              number,
              guess,
              step,
              tol )
```

Definition at line 1 of file test.py.

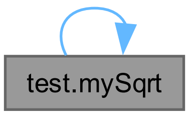
References mySqrt().

Referenced by mySqrt().

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.2 Variable Documentation

5.7.2.1 testVal

```
int test.testVal = 9
```

Definition at line 24 of file test.py.

Chapter 6

Data Structure Documentation

6.1 communication.CommunicationProtocol Class Reference

Public Member Functions

- __init__ (self, com, baudrate, timeout)
- send_command (self, command)

Data Fields

- com
- baudrate
- timeout
- connection

6.1.1 Detailed Description

Definition at line 18 of file communication.py.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 __init__()

```
communication.CommunicationProtocol.__init__ (
              self,
              com,
              baudrate,
              timeout )
```

Definition at line 19 of file communication.py.

6.1.3 Member Function Documentation

6.1.3.1 send_command()

```
communication.CommunicationProtocol.send_command (
              self,
              command )
```

Definition at line 25 of file communication.py.

References communication.CommunicationProtocol.connection.

6.1.4 Field Documentation

6.1.4.1 baudrate

```
communication.CommunicationProtocol.baudrate
```

Definition at line 21 of file communication.py.

6.1.4.2 com

```
communication.CommunicationProtocol.com
```

Definition at line 20 of file communication.py.

6.1.4.3 connection

```
communication.CommunicationProtocol.connection
```

Definition at line 23 of file communication.py.

Referenced by communication.CommunicationProtocol.send_command().

6.1.4.4 timeout

```
communication.CommunicationProtocol.timeout
```

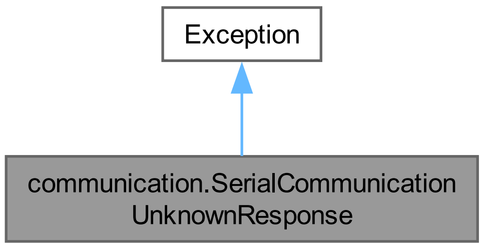
Definition at line 22 of file communication.py.

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

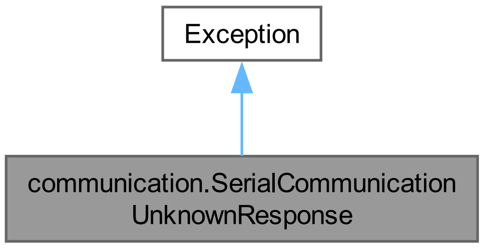
- communication.py

6.2 communication.SerialCommunicationUnknownResponse Class Reference

Inheritance diagram for communication.SerialCommunicationUnknownResponse:



Collaboration diagram for communication.SerialCommunicationUnknownResponse:



6.2.1 Detailed Description

```
Used for error handling. Also the reason for the inheritance of Exception. The following error is described as

SerialCommunicationUnknownResponse: The received serial response was not valid.
```

Definition at line 9 of file communication.py.

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

- communication.py

6.3 gui.GUI Class Reference

Public Member Functions

- __init__ (self)
- start_pressed (self)
- stop_pressed_init (self)
- stop_pressed (self)
- robot_pressed (self)
- simulation_pressed (self)
- both_pressed (self)
- run (self)

Data Fields

- root
- pressed_buttons
- mode
- running
- button_START
- button_STOP
- button_SELECT_ROBOT
- button_SELECT_SIMULATION
- button_SELECT_BOTH
- light_SELECT_ROBOT
- light_SELECT_SIMULATION
- light_SELECT_BOTH
- title

6.3.1 Detailed Description

```
Main class for the GUI
```

Definition at line 36 of file gui.py.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 __init__()

```
gui.GUI.__init__ (
                self )
```

```
Initializer for tkinter menu
```

Definition at line 40 of file gui.py.

6.3.3 Member Function Documentation

6.3.3.1 both_pressed()

```
gui.GUI.both_pressed (
              self )
```

```
Button method for when clicking on the 'Execute program on robot and in simulation'
```

Definition at line 174 of file gui.py.

References gui.GUI.button_SELECT_BOTH, gui.GUI.button_SELECT_ROBOT, gui.GUI.button_SELECT_SIMULATION, gui.GUI.button_START, gui.GUI.button_STOP, gui.GUI.light_SELECT_BOTH, gui.GUI.light_SELECT_ROBOT, gui.GUI.light_SELECT_SIMULATION, and gui.GUI.pressed_buttons.

6.3.3.2 robot_pressed()

```
gui.GUI.robot_pressed (
              self )
```

```
Button method for when clicking on the 'Execute program on robot'
```

Definition at line 142 of file gui.py.

References gui.GUI.button_SELECT_BOTH, gui.GUI.button_SELECT_ROBOT, gui.GUI.button_SELECT_SIMULATION, gui.GUI.button_START, gui.GUI.button_STOP, gui.GUI.light_SELECT_BOTH, gui.GUI.light_SELECT_ROBOT, gui.GUI.light_SELECT_SIMULATION, and gui.GUI.pressed_buttons.

6.3.3.3 run()

```
gui.GUI.run (
              self )
```

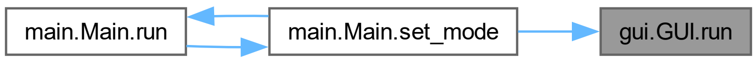
```
Runs the program
```

Definition at line 190 of file gui.py.

References gui.GUI.root.

Referenced by main.Main.set_mode().

Here is the caller graph for this function:



6.3.3.4 simulation_pressed()

```
gui.GUI.simulation_pressed (
              self )
```

```
Button method for when clicking on the 'Execute program in simulation'
```

Definition at line 158 of file gui.py.

References gui.GUI.button_SELECT_BOTH, gui.GUI.button_SELECT_ROBOT, gui.GUI.button_SELECT_SIMULATION, gui.GUI.button_START, gui.GUI.button_STOP, gui.GUI.light_SELECT_BOTH, gui.GUI.light_SELECT_ROBOT, gui.GUI.light_SELECT_SIMULATION, and gui.GUI.pressed_buttons.

6.3.3.5 start_pressed()

```
gui.GUI.start_pressed (
              self )
```

```
Button method for when clicking on the start button
```

Definition at line 79 of file gui.py.

References gui.GUI.button_SELECT_BOTH, gui.GUI.button_SELECT_ROBOT, gui.GUI.button_SELECT_SIMULATION, gui.GUI.button_START, gui.GUI.button_STOP, gui.GUI.light_SELECT_BOTH, gui.GUI.light_SELECT_ROBOT, gui.GUI.light_SELECT_SIMULATION, gui.GUI.mode, gui.GUI.pressed_buttons, and gui.GUI.running.

6.3.3.6 stop_pressed()

```
gui.GUI.stop_pressed (
              self )
```

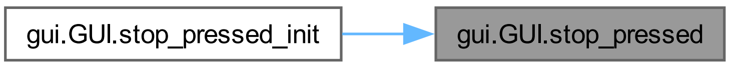
```
Button method for when the program has run the stop command and the buttons should be clickable again.
```

Definition at line 132 of file gui.py.

References gui.GUI.button_SELECT_BOTH, gui.GUI.button_SELECT_ROBOT, gui.GUI.button_SELECT_SIMULATION, gui.GUI.button_START, and gui.GUI.button_STOP.

Referenced by gui.GUI.stop_pressed_init().

Here is the caller graph for this function:



6.3.3.7 stop_pressed_init()

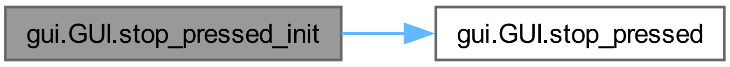
```
gui.GUI.stop_pressed_init (
              self )
```

```
Button method for when clicking on the stop button
```

Definition at line 109 of file gui.py.

References gui.GUI.button_SELECT_BOTH, gui.GUI.button_SELECT_ROBOT, gui.GUI.button_SELECT_SIMULATION, gui.GUI.button_START, gui.GUI.button_STOP, gui.GUI.light_SELECT_BOTH, gui.GUI.light_SELECT_ROBOT, gui.GUI.light_SELECT_SIMULATION, gui.GUI.mode, gui.GUI.pressed_buttons, gui.GUI.running, and gui.GUI.stop_pressed().

Here is the call graph for this function:



6.3.4 Field Documentation

6.3.4.1 button_SELECT_BOTH

```
gui.GUI.button_SELECT_BOTH
```

Definition at line 58 of file gui.py.

Referenced by gui.GUI.both_pressed(), gui.GUI.robot_pressed(), gui.GUI.simulation_pressed(), gui.GUI.start_pressed(), gui.GUI.stop_pressed(), and gui.GUI.stop_pressed_init().

6.3.4.2 button_SELECT_ROBOT

```
gui.GUI.button_SELECT_ROBOT
```

Definition at line 54 of file gui.py.

Referenced by gui.GUI.both_pressed(), gui.GUI.robot_pressed(), gui.GUI.simulation_pressed(), gui.GUI.start_pressed(), gui.GUI.stop_pressed(), and gui.GUI.stop_pressed_init().

6.3.4.3 button_SELECT_SIMULATION

```
gui.GUI.button_SELECT_SIMULATION
```

Definition at line 56 of file gui.py.

Referenced by gui.GUI.both_pressed(), gui.GUI.robot_pressed(), gui.GUI.simulation_pressed(), gui.GUI.start_pressed(), gui.GUI.stop_pressed(), and gui.GUI.stop_pressed_init().

6.3.4.4 button_START

```
gui.GUI.button_START
```

Definition at line 50 of file gui.py.

Referenced by gui.GUI.both_pressed(), gui.GUI.robot_pressed(), gui.GUI.simulation_pressed(), gui.GUI.start_pressed(), gui.GUI.stop_pressed(), and gui.GUI.stop_pressed_init().

6.3.4.5 button_STOP

```
gui.GUI.button_STOP
```

Definition at line 52 of file gui.py.

Referenced by gui.GUI.both_pressed(), gui.GUI.robot_pressed(), gui.GUI.simulation_pressed(), gui.GUI.start_pressed(), gui.GUI.stop_pressed(), and gui.GUI.stop_pressed_init().

6.3.4.6 light_SELECT_BOTH

```
gui.GUI.light_SELECT_BOTH
```

Definition at line 73 of file gui.py.

Referenced by gui.GUI.both_pressed(), gui.GUI.robot_pressed(), gui.GUI.simulation_pressed(), gui.GUI.start_pressed(), and gui.GUI.stop_pressed_init().

6.3.4.7 light_SELECT_ROBOT

```
gui.GUI.light_SELECT_ROBOT
```

Definition at line 67 of file gui.py.

Referenced by gui.GUI.both_pressed(), gui.GUI.robot_pressed(), gui.GUI.simulation_pressed(), gui.GUI.start_pressed(), and gui.GUI.stop_pressed_init().

6.3.4.8 light_SELECT_SIMULATION

```
gui.GUI.light_SELECT_SIMULATION
```

Definition at line 70 of file gui.py.

Referenced by gui.GUI.both_pressed(), gui.GUI.robot_pressed(), gui.GUI.simulation_pressed(), gui.GUI.start_pressed(), and gui.GUI.stop_pressed_init().

6.3.4.9 mode

```
gui.GUI.mode
```

Definition at line 47 of file gui.py.

Referenced by gui.GUI.start_pressed(), and gui.GUI.stop_pressed_init().

6.3.4.10 pressed_buttons

```
gui.GUI.pressed_buttons
```

Definition at line 45 of file gui.py.

Referenced by gui.GUI.both_pressed(), gui.GUI.robot_pressed(), gui.GUI.simulation_pressed(), gui.GUI.start_pressed(), and gui.GUI.stop_pressed_init().

6.3.4.11 root

```
gui.GUI.root
```

Definition at line 44 of file gui.py.

Referenced by gui.GUI.run().

6.3.4.12 running

```
gui.GUI.running
```

Definition at line 48 of file gui.py.

Referenced by gui.GUI.start_pressed(), and gui.GUI.stop_pressed_init().

6.3.4.13 title

```
gui.GUI.title
```

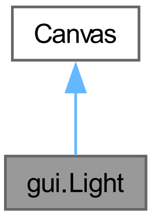
Definition at line 76 of file gui.py.

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

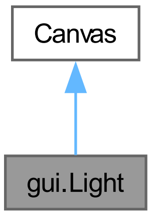
- gui.py

6.4 gui.Light Class Reference

Inheritance diagram for gui.Light:



Collaboration diagram for gui.Light:



Public Member Functions

- __init__ (self, master, ∗∗kwargs)
- turn_on (self)
- turn_off (self)
- turn_ready (self)

Data Fields

- oval_id

6.4.1 Detailed Description

```
Class to construct an oval element in the GUI
```

Definition at line 4 of file gui.py.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 __init__()

```
gui.Light.__init__ (
              self,
              master,
             ** kwargs )
```

```
initializer
:param master: the root of the tkinter menu
:param kwargs: possible arguments for the canvas.
```

Definition at line 8 of file gui.py.

6.4.3 Member Function Documentation

6.4.3.1 turn_off()

```
gui.Light.turn_off (
              self )
```

```
Changes the color of the oval to gray
```

Definition at line 23 of file gui.py.

References gui.Light.oval_id.

6.4.3.2 turn_on()

```
gui.Light.turn_on (
              self )
```

```
Changes the color of the oval to green
```

Definition at line 17 of file gui.py.

References gui.Light.oval_id.

6.4.3.3 turn_ready()

```
gui.Light.turn_ready (
              self )
```

```
Changes the color of the oval to red
```

Definition at line 29 of file gui.py.

References gui.Light.oval_id.

6.4.4 Field Documentation

6.4.4.1 oval_id

```
gui.Light.oval_id
```

Definition at line 15 of file gui.py.

Referenced by gui.Light.turn_off(), gui.Light.turn_on(), and gui.Light.turn_ready().

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

- gui.py

6.5 main.Main Class Reference

Public Member Functions

- __init__ (self)
- create_gui (self)
- set_mode (self, mode)
- run (self)
- dispense_allergen (self, index)
- take_allergen (self, index)
- place_allergen (self, index)

Data Fields

- communication
- patch_test_config
- robot
- gui_created
- gui_thread
- gui

6.5.1 Detailed Description

```
Main class containing the whole program.
```

Definition at line 59 of file main.py.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 __init__()

```
main.Main.__init__ (
              self )
```

```
Connecting to ROBODK or MyCobot.
```

Definition at line 63 of file main.py.

6.5.3 Member Function Documentation

6.5.3.1 create_gui()

```
main.Main.create_gui (
              self )
```

```
Creates the GUI
```

Definition at line 90 of file main.py.

6.5.3.2 dispense_allergen()

```
main.Main.dispense_allergen (
              self,
              index )
```

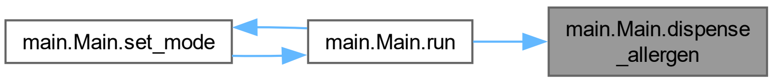
```
Runs the movements needed to dispense allergen onto a patch test
:param index: What index the position on the patch test has
:return: None
```

Definition at line 171 of file main.py.

References main.Main.communication, main.Main.robot, robot_movement.MoveJ.robot, and robot_movement.MoveL.robot.

Referenced by main.Main.run().

Here is the caller graph for this function:



6.5.3.3 place_allergen()

```
main.Main.place_allergen (
              self,
              index )
```

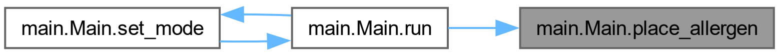
```
Places an allergen in storage
:param index: Index for storage place
:return: None
```

Definition at line 224 of file main.py.

References main.Main.communication, main.Main.robot, robot_movement.MoveJ.robot, and robot_movement.MoveL.robot.

Referenced by main.Main.run().

Here is the caller graph for this function:



6.5.3.4 run()

```
main.Main.run (
              self )
```

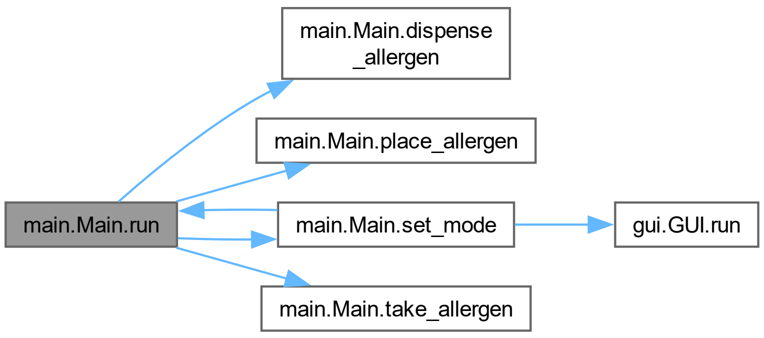
```
Runs the whole program sequence
```

Definition at line 131 of file main.py.

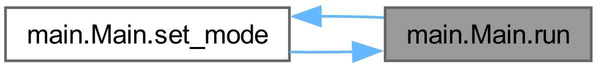
References main.Main.communication, main.Main.dispense_allergen(), main.Main.gui, main.Main.patch_test_config, main.Main.place_allergen(), main.Main.robot, robot_movement.MoveJ.robot, robot_movement.MoveL.robot, main.Main.set_mode(), and main.Main.take_allergen().

Referenced by main.Main.set_mode().

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.3.5 set_mode()

```
main.Main.set_mode (
              self,
              mode )
```

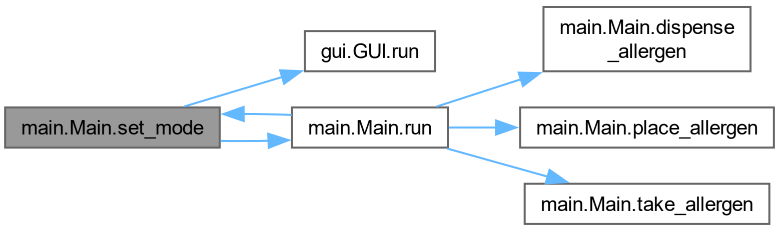
```
Sets the mode for how the robot should run
:param mode: either "both", "simulation", "robot".
```

Definition at line 100 of file main.py.

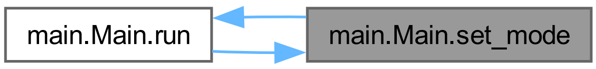
References main.Main.robot, robot_movement.MoveJ.robot, robot_movement.MoveL.robot, gui.GUI.run(), and main.Main.run().

Referenced by main.Main.run().

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.3.6 take_allergen()

```
main.Main.take_allergen (
              self,
              index )
```

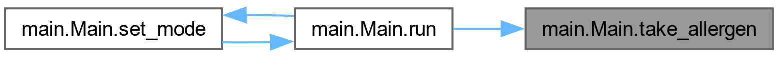
```
Takes an allergen from the storage
:param index: Index for the allergen in the storage system
:return: None
```

Definition at line 195 of file main.py.

References main.Main.communication, main.Main.robot, robot_movement.MoveJ.robot, and robot_movement.MoveL.robot.

Referenced by main.Main.run().

Here is the caller graph for this function:



6.5.4 Field Documentation

6.5.4.1 communication

`main.Main.communication`

Definition at line 67 of file main.py.

Referenced by main.Main.dispense_allergen(), main.Main.place_allergen(), main.Main.run(), and main.Main.take_allergen().

6.5.4.2 gui

`main.Main.gui`

Definition at line 94 of file main.py.

Referenced by main.Main.run().

6.5.4.3 gui_created

`main.Main.gui_created`

Definition at line 78 of file main.py.

6.5.4.4 gui_thread

```
main.Main.gui_thread
```

Definition at line 79 of file main.py.

6.5.4.5 patch_test_config

```
main.Main.patch_test_config
```

Definition at line 70 of file main.py.

Referenced by main.Main.run().

6.5.4.6 robot

```
main.Main.robot
```

Definition at line 76 of file main.py.

Referenced by main.Main.dispense_allergen(), robot_movement.MoveJ.move_joint(), main.Main.place_allergen(), main.Main.run(), main.Main.set_mode(), and main.Main.take_allergen().

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

- main.py

6.6 robot_controller.RobotControllerSettings Class Reference

Public Member Functions

- __init__ (self)

Data Fields

- update_frequency
- linear_frequency
- dh_values

6.6.1 Detailed Description

```
Robot controller settings class
```

Definition at line 4 of file robot_controller.py.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 __init__()

```
robot_controller.RobotControllerSettings.__init__ (
              self )
```

```
Settings for the robot controller
```

Definition at line 8 of file robot_controller.py.

6.6.3 Field Documentation

6.6.3.1 dh_values

```
robot_controller.RobotControllerSettings.dh_values
```

Definition at line 15 of file robot_controller.py.

6.6.3.2 linear_frequency

```
robot_controller.RobotControllerSettings.linear_frequency
```

Definition at line 13 of file robot_controller.py.

6.6.3.3 update_frequency

```
robot_controller.RobotControllerSettings.update_frequency
```

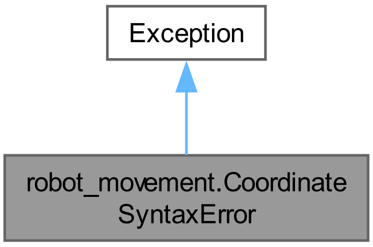
Definition at line 12 of file robot_controller.py.

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

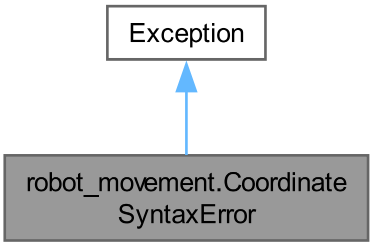
- robot_controller.py

6.7 robot_movement.CoordinateSyntaxError Class Reference

Inheritance diagram for robot_movement.CoordinateSyntaxError:



Collaboration diagram for robot_movement.CoordinateSyntaxError:



6.7.1 Detailed Description

```
Used for error handling. Also the reason for the inheritance of Exception. The following error is described as

CoordinateSyntaxError: The syntax must be: [x, y, z, alpha, beta, gamma, time] for a coordinate.
```

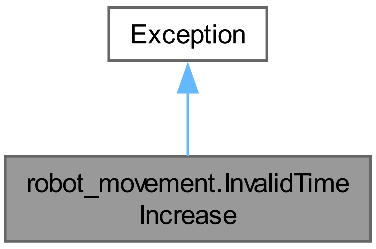
Definition at line 52 of file robot_movement.py.

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

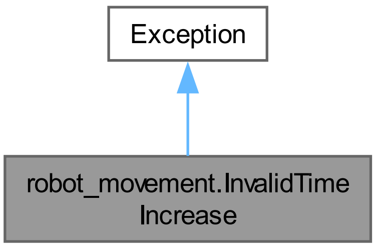
- robot_movement.py

6.8 robot_movement.InvalidTimeIncrease Class Reference

Inheritance diagram for robot_movement.InvalidTimeIncrease:



Collaboration diagram for robot_movement.InvalidTimeIncrease:



Static Public Attributes

- suppress

6.8.1 Detailed Description

```
Used for error handling. Also the reason for the inheritance of Exception. The following error is described as
InvalidTimeIncrease: Later coordinates must have a higher time than those before or the robot will not be able
```

Definition at line 61 of file robot_movement.py.

6.8.2 Field Documentation

6.8.2.1 suppress

```
robot_movement.InvalidTimeIncrease.suppress  [static]
```

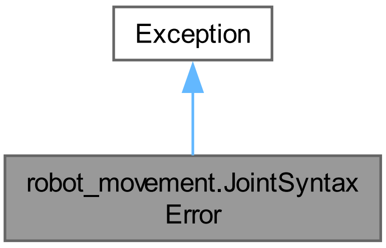
Definition at line 70 of file robot_movement.py.

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

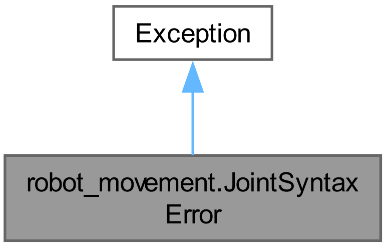
- robot_movement.py

6.9 robot_movement.JointSyntaxError Class Reference

Inheritance diagram for robot_movement.JointSyntaxError:



Collaboration diagram for robot_movement.JointSyntaxError:



6.9.1 Detailed Description

```
Used for error handling. Also the reason for the inheritance of Exception. The following error is described as

JointSyntaxError: The syntax must be [Joint1, joint2, joint3, joint4, joint5, joint6] for a joint
```

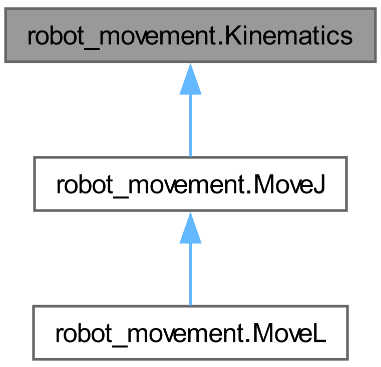
Definition at line 34 of file robot_movement.py.

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

- robot_movement.py

6.10 robot_movement.Kinematics Class Reference

Inheritance diagram for robot_movement.Kinematics:



Public Member Functions

- __init__ (self)
- get_joint_values (self, coordinate, orientation)
- kin_calculate_theta_1 (self, T_link0_link6)
- kin_calculate_theta_2_3 (self, T_link2_link3)
- kin_calculate_theta_4 (self, rot_matrix, theta_5)
- kin_calculate_theta_5 (self, rot_matrix)
- kin_calculate_theta_6 (self, rot_matrix, theta_5)

Data Fields

- robot_controller_settings
- dh_value_alpha
- dh_value_a
- dh_value_d
- dh_value_theta

6.10.1 Detailed Description

```
Class to compute the inverse kinematics.
```

Definition at line 221 of file robot_movement.py.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 __init__()

```
robot_movement.Kinematics.__init__ (
              self )
```

```
Constructor for class
```

Reimplemented in robot_movement.MoveJ, and robot_movement.MoveL.

Definition at line 225 of file robot_movement.py.

6.10.3 Member Function Documentation

6.10.3.1 get_joint_values()

```
robot_movement.Kinematics.get_joint_values (
              self,
              coordinate,
              orientation )
```

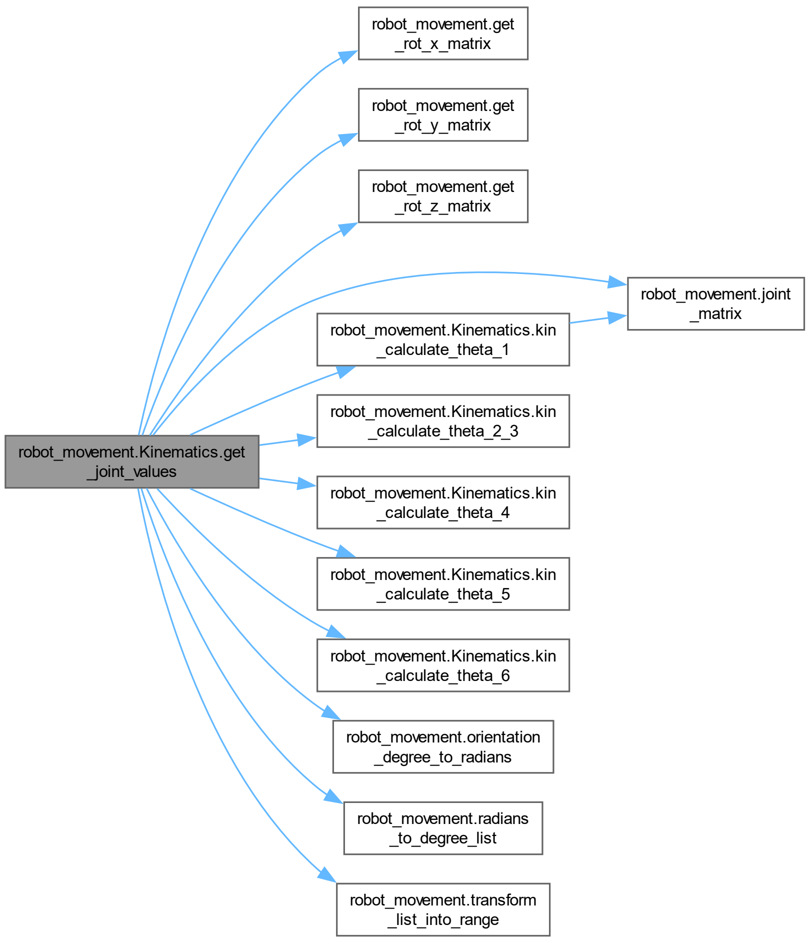
```
Method for getting the joint values from a coordinate and orientation.
:param coordinate: Coordinate in the format [x, y, z].
:param orientation: Orientation in the format [roll, pitch, yaw].
:return: Returns a list containing all possible joint configurations for the robot.
```

Definition at line 235 of file robot_movement.py.

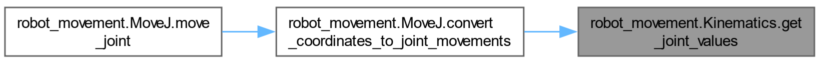
References robot_movement.Kinematics.dh_value_a, robot_movement.Kinematics.dh_value_alpha, robot_movement.Kinematics.dh_
robot_movement.Kinematics.dh_value_theta, robot_movement.get_rot_x_matrix(), robot_movement.get_rot_y_matrix(), robot_movement.get_rot_z_matrix(), robot_movement.joint_matrix(), robot_movement.Kinematics.kin_calculate_theta_1(), robot_movement.Kinematics.kin_calculate_theta_2_3(), robot_movement.Kinematics.kin_calculate_theta_4(), robot_movement.Kinematics.kin_calculate_theta_5(), robot_movement.Kinematics.kin_calculate_theta_6(), robot_movement.orientation_degree_to_radians(), robot_movement.radians_to_degree_list(), and robot_movement.transform_list_in

Referenced by robot_movement.MoveJ.convert_coordinates_to_joint_movements().

Here is the call graph for this function:


robot_movement.get_rot_x_matrix
robot_movement.get_rot_y_matrix
robot_movement.get_rot_z_matrix
robot_movement.joint_matrix
robot_movement.Kinematics.kin_calculate_theta_1
robot_movement.Kinematics.get_joint_values
robot_movement.Kinematics.kin_calculate_theta_2_3
robot_movement.Kinematics.kin_calculate_theta_4
robot_movement.Kinematics.kin_calculate_theta_5
robot_movement.Kinematics.kin_calculate_theta_6
robot_movement.orientation_degree_to_radians
robot_movement.radians_to_degree_list
robot_movement.transform_list_into_range

Here is the caller graph for this function:


robot_movement.MoveJ.move_joint
robot_movement.MoveJ.convert_coordinates_to_joint_movements
robot_movement.Kinematics.get_joint_values

6.10.3.2 kin_calculate_theta_1()

```
robot_movement.Kinematics.kin_calculate_theta_1 (
              self,
              T_link0_link6 )
```

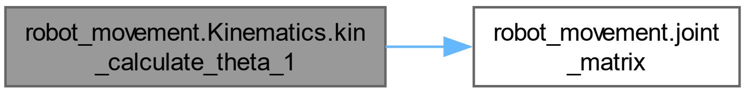
```
Calculates theta 1
:param T_link0_link6: Transformation matrix from link 1 to link 6.
:return: Returns the two possible values for theta 1
```

Definition at line 369 of file robot_movement.py.

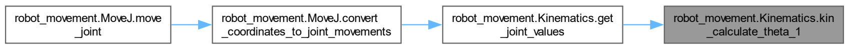
References robot_movement.Kinematics.dh_value_d, and robot_movement.joint_matrix().

Referenced by robot_movement.Kinematics.get_joint_values().

Here is the call graph for this function:



Here is the caller graph for this function:


robot_movement.MoveJ.move
_joint
robot_movement.MoveJ.convert
_coordinates_to_joint_movements
robot_movement.Kinematics.get
_joint_values
robot_movement.Kinematics.kin
_calculate_theta_1

6.10.3.3 kin_calculate_theta_2_3()

```
robot_movement.Kinematics.kin_calculate_theta_2_3 (
              self,
              T_link2_link3 )
```

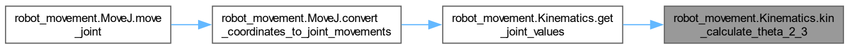
```
Calculates theta 2 and 3
:param T_link2_link3: Transformation matrix from link 2 to link 3
:return: Returns the possible configurations for theta 2 and theta 3.
```

Definition at line 387 of file robot_movement.py.

References robot_movement.Kinematics.dh_value_a.

Referenced by robot_movement.Kinematics.get_joint_values().

Here is the caller graph for this function:


robot_movement.MoveJ.move
_joint
robot_movement.MoveJ.convert
_coordinates_to_joint_movements
robot_movement.Kinematics.get
_joint_values
robot_movement.Kinematics.kin
_calculate_theta_2_3

6.10.3.4 kin_calculate_theta_4()

```
robot_movement.Kinematics.kin_calculate_theta_4 (
              self,
              rot_matrix,
              theta_5 )
```

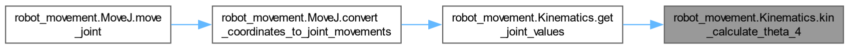
```
Calculates theta 4
:param rot_matrix: Rotation matrix describing rotation from link 4 to link 6.
:param theta_5: Value for theta 5
:return: Returns the value for theta 4
```

Definition at line 419 of file robot_movement.py.

References robot_movement.Kinematics.dh_value_theta.

Referenced by robot_movement.Kinematics.get_joint_values().

Here is the caller graph for this function:


robot_movement.MoveJ.move_joint
robot_movement.MoveJ.convert_coordinates_to_joint_movements
robot_movement.Kinematics.get_joint_values
robot_movement.Kinematics.kin_calculate_theta_4

6.10.3.5 kin_calculate_theta_5()

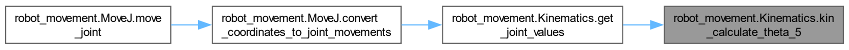
```
robot_movement.Kinematics.kin_calculate_theta_5 (
              self,
              rot_matrix )
```

```
Calculates theta 5
:param rot_matrix: Rotation matrix describing rotation from link 4 to link 6.
:return: Returns the two values for theta 5
```

Definition at line 432 of file robot_movement.py.

Referenced by robot_movement.Kinematics.get_joint_values().

Here is the caller graph for this function:


robot_movement.MoveJ.move_joint
robot_movement.MoveJ.convert_coordinates_to_joint_movements
robot_movement.Kinematics.get_joint_values
robot_movement.Kinematics.kin_calculate_theta_5

6.10.3.6 kin_calculate_theta_6()

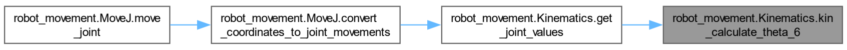
```
robot_movement.Kinematics.kin_calculate_theta_6 (
              self,
              rot_matrix,
              theta_5 )
```

```
 Calculates theta 6
 :param rot_matrix: Rotation matrix describing rotation from link 4 to link 6.
 :param theta_5: Value for theta 5
 :return: Returns the value for theta 6
```

Definition at line 444 of file robot_movement.py.

Referenced by robot_movement.Kinematics.get_joint_values().

Here is the caller graph for this function:


robot_movement.MoveJ.move_joint
robot_movement.MoveJ.convert_coordinates_to_joint_movements
robot_movement.Kinematics.get_joint_values
robot_movement.Kinematics.kin_calculate_theta_6

6.10.4 Field Documentation

6.10.4.1 dh_value_a

```
robot_movement.Kinematics.dh_value_a
```

Definition at line 231 of file robot_movement.py.

Referenced by robot_movement.Kinematics.get_joint_values(), and robot_movement.Kinematics.kin_calculate_theta_2_3().

6.10.4.2 dh_value_alpha

```
robot_movement.Kinematics.dh_value_alpha
```

Definition at line 230 of file robot_movement.py.

Referenced by robot_movement.Kinematics.get_joint_values().

6.10.4.3 dh_value_d

```
robot_movement.Kinematics.dh_value_d
```

Definition at line 232 of file robot_movement.py.

Referenced by robot_movement.Kinematics.get_joint_values(), and robot_movement.Kinematics.kin_calculate_theta_1().

6.10.4.4 dh_value_theta

```
robot_movement.Kinematics.dh_value_theta
```

Definition at line 233 of file robot_movement.py.

Referenced by robot_movement.Kinematics.get_joint_values(), and robot_movement.Kinematics.kin_calculate_theta_4().

6.10.4.5 robot_controller_settings

```
robot_movement.Kinematics.robot_controller_settings
```

Definition at line 229 of file robot_movement.py.

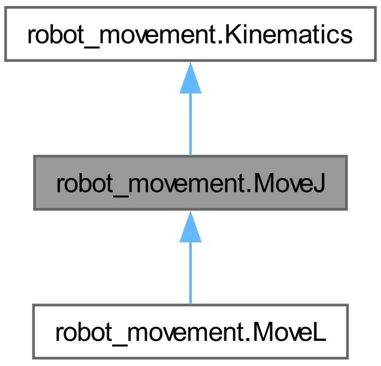
Referenced by robot_movement.MoveL.get_linear_via_points(), and robot_movement.MoveJ.move_joint().

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

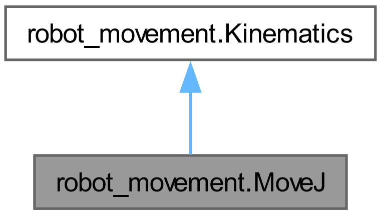
- robot_movement.py

6.11 robot_movement.MoveJ Class Reference

Inheritance diagram for robot_movement.MoveJ:



Collaboration diagram for robot_movement.MoveJ:



Public Member Functions

- __init__ (self, robot_pointer, positions_with_time)
- convert_coordinates_to_joint_movements (self)
- move_joint (self)

Public Member Functions inherited from robot_movement.Kinematics

- get_joint_values (self, coordinate, orientation)
- kin_calculate_theta_1 (self, T_link0_link6)
- kin_calculate_theta_2_3 (self, T_link2_link3)
- kin_calculate_theta_4 (self, rot_matrix, theta_5)
- kin_calculate_theta_5 (self, rot_matrix)
- kin_calculate_theta_6 (self, rot_matrix, theta_5)

Data Fields

- robot
- pos_w_time
- start_joint_values
- joint_matrix

Data Fields inherited from robot_movement.Kinematics

- robot_controller_settings
- dh_value_alpha
- dh_value_a
- dh_value_d
- dh_value_theta

6.11.1 Detailed Description

```
Class for moving in joint space
```

Definition at line 456 of file robot_movement.py.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 __init__()

```
robot_movement.MoveJ.__init__ (
              self,
              robot_pointer,
              positions_with_time )
```

```
Constructor for the MoveJ class. Takes the robot and position.
:param robot_pointer: A pointer to the robot class for the robot
:param positions_with_time: The position with time.
```

Reimplemented from robot_movement.Kinematics.

Reimplemented in robot_movement.MoveL.

Definition at line 460 of file robot_movement.py.

6.11.3 Member Function Documentation

6.11.3.1 convert_coordinates_to_joint_movements()

```
robot_movement.MoveJ.convert_coordinates_to_joint_movements (
              self )
```

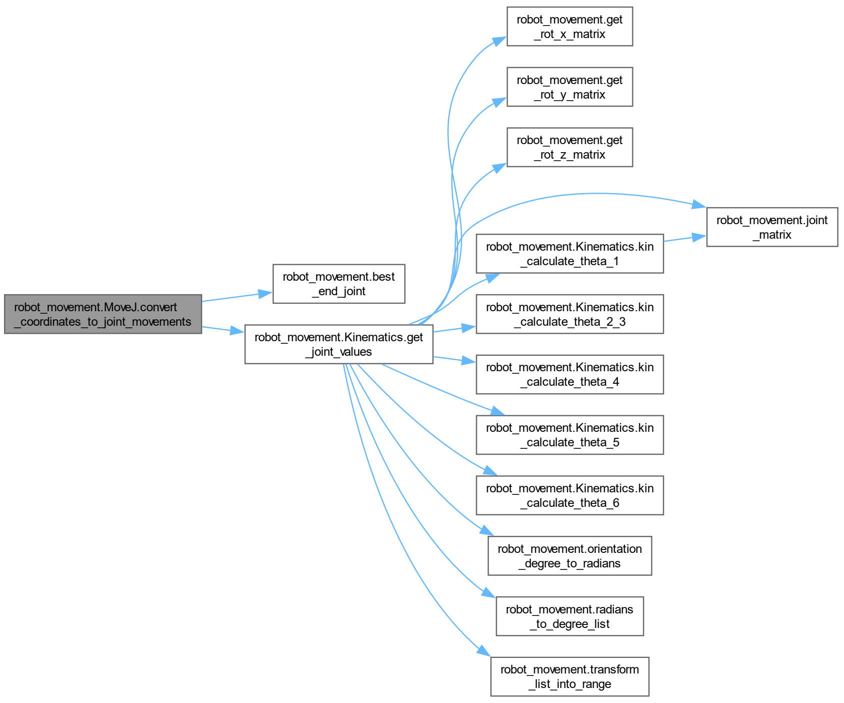
```
Method that converts the coordinates into joint degrees for each via point and end position.
:return: Returns the transformed matrix.
```

Definition at line 488 of file robot_movement.py.

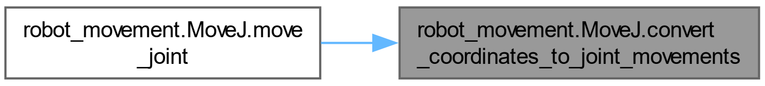
References robot_movement.best_end_joint(), robot_movement.Kinematics.get_joint_values(), robot_movement.MoveJ.pos_w_time, robot_movement.MoveL.pos_w_time, and robot_movement.MoveJ.start_joint_values.

Referenced by robot_movement.MoveJ.move_joint().

Here is the call graph for this function:



Here is the caller graph for this function:


robot_movement.MoveJ.move
_joint
robot_movement.MoveJ.convert
_coordinates_to_joint_movements

6.11.3.2 move_joint()

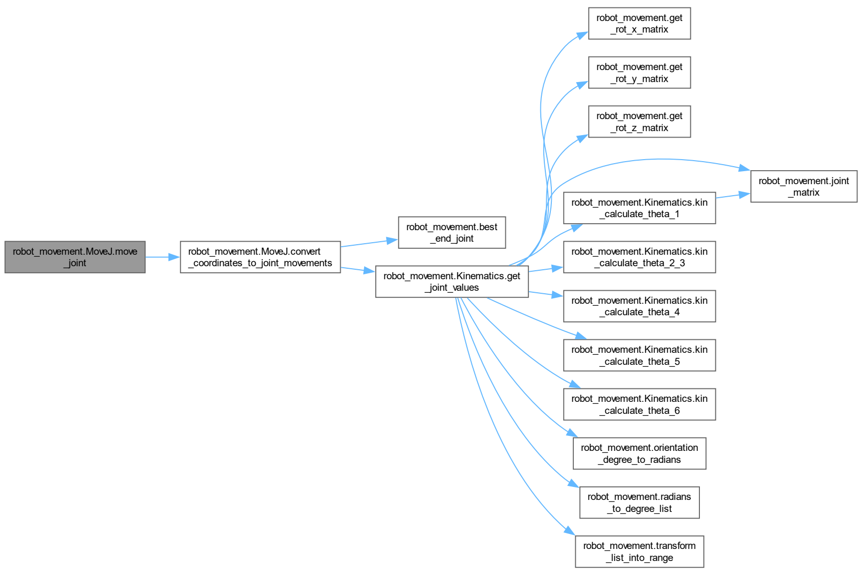
```
robot_movement.MoveJ.move_joint (
              self )
```

```
Method for moving the robot in the joint space thrugh the different via points.
```

Definition at line 517 of file robot_movement.py.

References robot_movement.MoveJ.convert_coordinates_to_joint_movements(), robot_movement.MoveJ.joint_matrix, robot_movement.MoveJ.pos_w_time, robot_movement.MoveL.pos_w_time, main.Main.robot, robot_movement.MoveJ.robot, robot_movement.MoveL.robot, and robot_movement.Kinematics.robot_controller_settings.

Here is the call graph for this function:


robot_movement.MoveJ.move _joint
robot_movement.MoveJ.convert _coordinates_to_joint_movements
robot_movement.best _end_joint
robot_movement.Kinematics.get _joint_values
robot_movement.get _rot_x_matrix
robot_movement.get _rot_y_matrix
robot_movement.get _rot_z_matrix
robot_movement.joint _matrix
robot_movement.Kinematics.kin _calculate_theta_1
robot_movement.Kinematics.kin _calculate_theta_2_3
robot_movement.Kinematics.kin _calculate_theta_4
robot_movement.Kinematics.kin _calculate_theta_5
robot_movement.Kinematics.kin _calculate_theta_6
robot_movement.orientation _degree_to_radians
robot_movement.radians _to_degree_list
robot_movement.transform _list_into_range

6.11.4 Field Documentation

6.11.4.1 joint_matrix

```
robot_movement.MoveJ.joint_matrix
```

Definition at line 486 of file robot_movement.py.

Referenced by robot_movement.MoveJ.move_joint().

6.11.4.2 pos_w_time

```
robot_movement.MoveJ.pos_w_time
```

Definition at line 467 of file robot_movement.py.

Referenced by robot_movement.MoveJ.convert_coordinates_to_joint_movements(), and robot_movement.MoveJ.move_joint().

6.11.4.3 robot

```
robot_movement.MoveJ.robot
```

Definition at line 466 of file robot_movement.py.

Referenced by main.Main.dispense_allergen(), robot_movement.MoveJ.move_joint(), main.Main.place_allergen(), main.Main.run(), main.Main.set_mode(), and main.Main.take_allergen().

6.11.4.4 start_joint_values

```
robot_movement.MoveJ.start_joint_values
```

Definition at line 468 of file robot_movement.py.

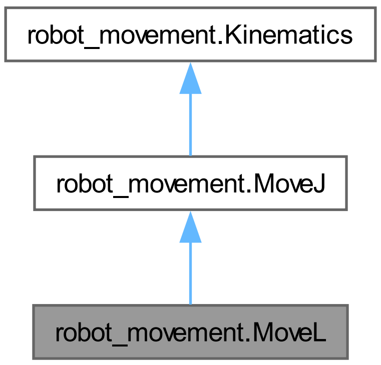
Referenced by robot_movement.MoveJ.convert_coordinates_to_joint_movements().

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

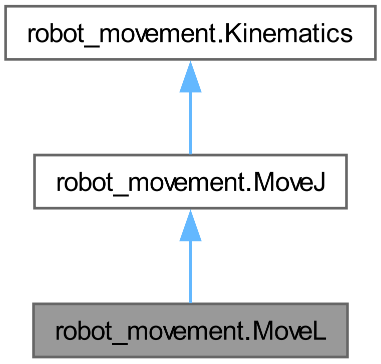
- robot_movement.py

6.12 robot_movement.MoveL Class Reference

Inheritance diagram for robot_movement.MoveL:



Collaboration diagram for robot_movement.MoveL:



Public Member Functions

- __init__ (self, robot_pointer, positions_with_time)
- get_linear_via_points (self)

Public Member Functions inherited from robot_movement.MoveJ

- convert_coordinates_to_joint_movements (self)
- move_joint (self)

Public Member Functions inherited from robot_movement.Kinematics

- get_joint_values (self, coordinate, orientation)
- kin_calculate_theta_1 (self, T_link0_link6)
- kin_calculate_theta_2_3 (self, T_link2_link3)
- kin_calculate_theta_4 (self, rot_matrix, theta_5)
- kin_calculate_theta_5 (self, rot_matrix)
- kin_calculate_theta_6 (self, rot_matrix, theta_5)

Data Fields

- robot
- pos_w_time_linear
- move_time
- start_pos_cartesian
- pos_w_time

Data Fields inherited from robot_movement.MoveJ

- robot
- pos_w_time
- start_joint_values
- joint_matrix

Data Fields inherited from robot_movement.Kinematics

- robot_controller_settings
- dh_value_alpha
- dh_value_a
- dh_value_d
- dh_value_theta

6.12.1 Detailed Description

```
Class for moving in cartesian space.
```

Definition at line 583 of file robot_movement.py.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 __init__()

```
robot_movement.MoveL.__init__ (
              self,
              robot_pointer,
              positions_with_time )
```

```
Constructor for MoveL class
:param robot_pointer: Pointer to robot controller
:param positions_with_time: Position for where the robot should move to.
```

Reimplemented from robot_movement.MoveJ.

Definition at line 587 of file robot_movement.py.

6.12.3 Member Function Documentation

6.12.3.1 get_linear_via_points()

```
robot_movement.MoveL.get_linear_via_points (
              self )
```

```
Method for getting all via points on the path, the robot moves on.
:return: Returns the coordinates which will be used as via points in the joint movement.
```

Definition at line 618 of file robot_movement.py.

References robot_movement.MoveL.move_time, robot_movement.MoveL.pos_w_time_linear, robot_movement.Kinematics.robot_cor
and robot_movement.MoveL.start_pos_cartesian.

6.12.4 Field Documentation

6.12.4.1 move_time

```
robot_movement.MoveL.move_time
```

Definition at line 595 of file robot_movement.py.

Referenced by robot_movement.MoveL.get_linear_via_points().

6.12.4.2 pos_w_time

```
robot_movement.MoveL.pos_w_time
```

Definition at line 616 of file robot_movement.py.

Referenced by robot_movement.MoveJ.convert_coordinates_to_joint_movements(), and robot_movement.MoveJ.move_joint().

6.12.4.3 pos_w_time_linear

```
robot_movement.MoveL.pos_w_time_linear
```

Definition at line 594 of file robot_movement.py.

Referenced by robot_movement.MoveL.get_linear_via_points().

6.12.4.4 robot

```
robot_movement.MoveL.robot
```

Definition at line 593 of file robot_movement.py.

Referenced by main.Main.dispense_allergen(), robot_movement.MoveJ.move_joint(), main.Main.place_allergen(), main.Main.run(), main.Main.set_mode(), and main.Main.take_allergen().

6.12.4.5 start_pos_cartesian

```
robot_movement.MoveL.start_pos_cartesian
```

Definition at line 613 of file robot_movement.py.

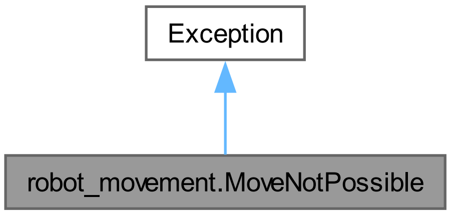
Referenced by robot_movement.MoveL.get_linear_via_points().

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

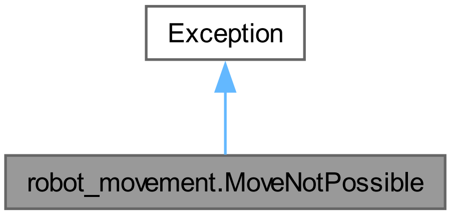
- robot_movement.py

6.13 robot_movement.MoveNotPossible Class Reference

Inheritance diagram for robot_movement.MoveNotPossible:



Collaboration diagram for robot_movement.MoveNotPossible:



6.13.1 Detailed Description

```
Used for error handling. Also the reason for the inheritance of Exception. The following error is described as

MoveNotPossible: The given movement command can not be executed by the program. Please make sure that it dies
```

Definition at line 25 of file robot_movement.py.

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

- robot_movement.py

6.14 robot_movement.Robot Class Reference

Public Member Functions

- __init__ (self, rdk_link)
- set_joints (self, position, robot_speed=100)

Data Fields

- robot_settings
- rdk_link
- prev_position
- time_between_move
- max_joint_speed

6.14.1 Detailed Description

```
Robot class used to talk with RoboDK and the correct robot
```

Definition at line 651 of file robot_movement.py.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 __init__()

```
robot_movement.Robot.__init__ (
              self,
              rdk_link )
```

```
Constructor for class
```

Definition at line 655 of file robot_movement.py.

6.14.3 Member Function Documentation

6.14.3.1 set_joints()

```
robot_movement.Robot.set_joints (
              self,
              position,
              robot_speed = 100 )
```

```
Sets the joint values for the robot
:param position: Position in joint space.
:param robot_speed: Not used due to it not being neccessary, but has to be there in order to use the class tog
```

Definition at line 666 of file robot_movement.py.

References robot_movement.Robot.max_joint_speed, robot_movement.Robot.prev_position, robot_movement.Robot.rdk_link, robot_movement.RobotMyCobotAndSim.rdk_link, and robot_movement.Robot.time_between_move.

6.14.4 Field Documentation

6.14.4.1 max_joint_speed

`robot_movement.Robot.max_joint_speed`

Definition at line 664 of file robot_movement.py.

Referenced by robot_movement.Robot.set_joints().

6.14.4.2 prev_position

`robot_movement.Robot.prev_position`

Definition at line 662 of file robot_movement.py.

Referenced by robot_movement.Robot.set_joints().

6.14.4.3 rdk_link

`robot_movement.Robot.rdk_link`

Definition at line 661 of file robot_movement.py.

Referenced by robot_movement.Robot.set_joints(), and robot_movement.RobotMyCobotAndSim.set_joints().

6.14.4.4 robot_settings

`robot_movement.Robot.robot_settings`

Definition at line 659 of file robot_movement.py.

6.14.4.5 time_between_move

`robot_movement.Robot.time_between_move`

Definition at line 663 of file robot_movement.py.

Referenced by robot_movement.Robot.set_joints().

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

- robot_movement.py

6.15 robot_movement.RobotMyCobot Class Reference

Public Member Functions

- __init__ (self, robo_link)
- set_joints (self, position, robot_speed=100)

Data Fields

- robo_link

6.15.1 Detailed Description

```
Robot class used to talk with MyCobot 320 PI
```

Definition at line 684 of file robot_movement.py.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 __init__()

```
robot_movement.RobotMyCobot.__init__ (
              self,
              robo_link )
```

```
Constructor for class
```

Definition at line 688 of file robot_movement.py.

6.15.3 Member Function Documentation

6.15.3.1 set_joints()

```
robot_movement.RobotMyCobot.set_joints (
              self,
              position,
              robot_speed = 100 )
```

```
Sets the joint values for the robot
:param position: Position in joint space.
:param robot_speed: Value between 0-100 setting the robot's speed.
```

Definition at line 694 of file robot_movement.py.

References robot_movement.RobotMyCobot.robo_link, and robot_movement.RobotMyCobotAndSim.robo_link.

6.15.4 Field Documentation

6.15.4.1 robo_link

```
robot_movement.RobotMyCobot.robo_link
```

Definition at line 692 of file robot_movement.py.

Referenced by robot_movement.RobotMyCobot.set_joints(), and robot_movement.RobotMyCobotAndSim.set_joints().

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

- robot_movement.py

6.16 robot_movement.RobotMyCobotAndSim Class Reference

Public Member Functions

- __init__ (self, robo_link, rdk_link)
- set_joints (self, position, robot_speed=100)

Data Fields

- rdk_link
- robo_link

6.16.1 Detailed Description

```
Robot class used to talk with MyCobot 320 PI and RoboDK
```

Definition at line 709 of file robot_movement.py.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 __init__()

```
robot_movement.RobotMyCobotAndSim.__init__ (
                self,
                robo_link,
                rdk_link )
```

```
Constructor for class
```

Definition at line 713 of file robot_movement.py.

6.16.3 Member Function Documentation

6.16.3.1 set_joints()

```
robot_movement.RobotMyCobotAndSim.set_joints (
                self,
                position,
                robot_speed = 100 )
```

```
Sets the joint values for the robot
:param position: Position in joint space.
:param robot_speed: Value between 0-100 setting the robot's speed.
```

Definition at line 720 of file robot_movement.py.

References robot_movement.Robot.rdk_link, robot_movement.RobotMyCobotAndSim.rdk_link, robot_movement.RobotMyCobot.robo
and robot_movement.RobotMyCobotAndSim.robo_link.

6.16.4 Field Documentation

6.16.4.1 rdk_link

```
robot_movement.RobotMyCobotAndSim.rdk_link
```

Definition at line 717 of file robot_movement.py.

Referenced by robot_movement.Robot.set_joints(), and robot_movement.RobotMyCobotAndSim.set_joints().

6.16.4.2 robo_link

```
robot_movement.RobotMyCobotAndSim.robo_link
```

Definition at line 718 of file robot_movement.py.

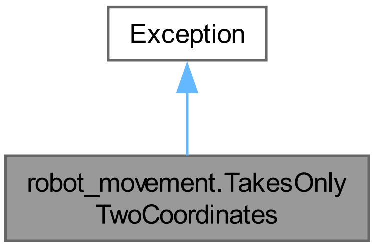
Referenced by robot_movement.RobotMyCobot.set_joints(), and robot_movement.RobotMyCobotAndSim.set_joints().

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

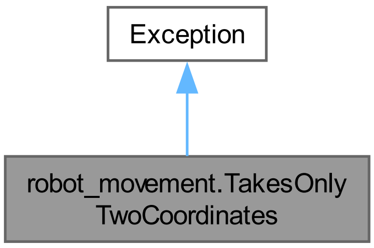
- robot_movement.py

6.17 robot_movement.TakesOnlyTwoCoordinates Class Reference

Inheritance diagram for robot_movement.TakesOnlyTwoCoordinates:



Collaboration diagram for robot_movement.TakesOnlyTwoCoordinates:



6.17.1 Detailed Description

```
Used for error handling. Also the reason for the inheritance of Exception. The following error is described as

TakesOnlyTwoCoordinates: This class takes only two coordinates.
```

Definition at line 43 of file robot_movement.py.

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

- robot_movement.py

Chapter 7

File Documentation

7.1 commands.py File Reference

Namespaces

- namespace commands

Variables

- str commands.HOME_GRIPPER = "home -g"
- str commands.CLOSE_GRIPPER = "close"
- str commands.HOME_PRESSER = "home -p"
- str commands.PREPARE_PRESSING = "prepare pressing"
- str commands.PRESS_ALLERGEN = "press -"
- str commands.NEW_TEST = "new test"
- str commands.RESPONSE = "b'Done'"
- str commands.EMPTY = "b''"

7.2 commands.py

Go to the documentation of this file.

```
HOME_GRIPPER = "home -g"
CLOSE_GRIPPER = "close"

HOME_PRESSER = "home -p"
PREPARE_PRESSING = "prepare pressing"
PRESS_ALLERGEN = "press -"

NEW_TEST = "new test"

RESPONSE = "b'Done'"
EMPTY = "b''"
```

7.3 communication.py File Reference

Data Structures

- class communication.SerialCommunicationUnknownResponse
- class communication.CommunicationProtocol

Namespaces

- namespace communication

Variables

- str communication.SERIAL_COMMUNICATION_UNKNOWN_RESPONSE_ERROR_MSG = "Serial↩ CommunicationUnknownResponse: The received serial response was not valid."
- communication.com_test = CommunicationProtocol("COM3", 9600, 0.1)

7.4 communication.py

Go to the documentation of this file.

```
import serial
import commands as cmd
import time as t


SERIAL_COMMUNICATION_UNKNOWN_RESPONSE_ERROR_MSG = "SerialCommunicationUnknownResponse: The received
serial response was not valid."


class SerialCommunicationUnknownResponse(Exception):
    """
    Used for error handling. Also the reason for the inheritance of Exception. The following error is
described as the following:

    SerialCommunicationUnknownResponse: The received serial response was not valid.
    """
    pass


class CommunicationProtocol:
    def __init__(self, com, baudrate, timeout):
        self.com = com
        self.baudrate = baudrate
        self.timeout = timeout
        self.connection = serial.Serial(port=self.com, baudrate=self.baudrate, timeout=self.timeout)

    def send_command(self, command):
        self.connection.write(bytes(command, 'utf-8'))

        while True:
            resp = str(self.connection.readline())
            print(resp)

            if resp == cmd.RESPONSE:
                break

            if resp != cmd.EMPTY:
                raise
SerialCommunicationUnknownResponse(SERIAL_COMMUNICATION_UNKNOWN_RESPONSE_ERROR_MSG)


if __name__ == "__main__":
    com_test = CommunicationProtocol("COM3", 9600, 0.1)
    t.sleep(5)
    com_test.send_command(cmd.HOME_PRESSER)
    com_test.send_command(cmd.HOME_GRIPPER)
    com_test.send_command(cmd.CLOSE_GRIPPER)
    t.sleep(5)
    com_test.send_command(cmd.PREPARE_PRESSING)
    com_test.send_command(cmd.PRESS_ALLERGEN + "2500")

```

7.5 gui.py File Reference

Data Structures

- class gui.Light
- class gui.GUI

Namespaces

- namespace gui

Variables

- gui.test = GUI()

7.6 gui.py

Go to the documentation of this file.

```
from tkinter import *


class Light(Canvas):
    """
    Class to construct an oval element in the GUI
    """
    def __init__(self, master, **kwargs):
        """
        initializer
        :param master: the root of the tkinter menu
        :param kwargs: possible arguments for the canvas.
        """
        Canvas.__init__(self, master, width=50, height=50, **kwargs)
        self.oval_id = self.create_oval(2, 2, 50, 50, fill='gray')

    def turn_on(self):
        """
        Changes the color of the oval to green
        """
        self.itemconfig(self.oval_id, fill='green')

    def turn_off(self):
        """
        Changes the color of the oval to gray
        """
        self.itemconfig(self.oval_id, fill='gray')

    def turn_ready(self):
        """
        Changes the color of the oval to red
        """
        self.itemconfig(self.oval_id, fill='yellow')


class GUI:
    """
    Main class for the GUI
    """
    def __init__(self):
        """
        Initializer for tkinter menu
        """
        self.root = Tk()
        self.pressed_buttons = set()

        self.mode = None
        self.running = False

        self.button_START = Button(self.root, text="START", padx=150, pady=50, state=DISABLED,
command=self.start_pressed, fg="black",
                                   bg="green")
        self.button_STOP = Button(self.root, text="STOP", padx=140, pady=50, state=NORMAL,
command=self.stop_pressed_init, fg="black",
                                   bg="red")
        self.button_SELECT_ROBOT = Button(self.root, text="Execute program on robot", padx=150,
pady=50, state=NORMAL, command=self.robot_pressed, fg="white",
                                   bg="black")
        self.button_SELECT_SIMULATION = Button(self.root, text="Execute program in simulation",
padx=150, pady=50, state=NORMAL, command=self.simulation_pressed,
                                   fg="white", bg="black")
        self.button_SELECT_BOTH = Button(self.root, text="Execute program on robot and in simulation",
padx=100, pady=50, state=NORMAL, command=self.both_pressed,
                                   fg="white", bg="black")

        self.button_START.grid(row=0, column=0, padx=10, pady=10)
        self.button_STOP.grid(row=0, column=2, padx=10, pady=10)
        self.button_SELECT_ROBOT.grid(row=1, column=0, padx=10, pady=10)
        self.button_SELECT_SIMULATION.grid(row=1, column=1, padx=10, pady=10)
        self.button_SELECT_BOTH.grid(row=1, column=2, padx=10, pady=10)

        self.light_SELECT_ROBOT = Light(self.root)
        self.light_SELECT_ROBOT.grid(row=2, column=0, padx=10, pady=10)

        self.light_SELECT_SIMULATION = Light(self.root)
        self.light_SELECT_SIMULATION.grid(row=2, column=1, padx=10, pady=10)

        self.light_SELECT_BOTH = Light(self.root)
        self.light_SELECT_BOTH.grid(row=2, column=2, padx=10, pady=10)

        self.title = Label(self.root, text="Patch test preparer")
        self.title.grid(row=0, column=1)

    def start_pressed(self):
        """
        Button method for when clicking on the start button
        """
        self.button_START["state"] = NORMAL
        self.button_STOP["state"] = NORMAL
        self.button_SELECT_ROBOT["state"] = DISABLED
        self.button_SELECT_SIMULATION["state"] = DISABLED
        self.button_SELECT_BOTH["state"] = DISABLED

        if "Robot" in self.pressed_buttons:
            self.light_SELECT_ROBOT.turn_on()
            self.mode = "robot"
        else:
            self.light_SELECT_ROBOT.turn_off()

        if "Simulation" in self.pressed_buttons:
            self.light_SELECT_SIMULATION.turn_on()
            self.mode = "simulation"
        else:
            self.light_SELECT_SIMULATION.turn_off()

        if "Robot and Simulation" in self.pressed_buttons:
            self.light_SELECT_BOTH.turn_on()
            self.mode = "both"
        else:
            self.light_SELECT_BOTH.turn_off()

        self.running = True

    def stop_pressed_init(self):
        """
        Button method for when clicking on the stop button
        """
        self.mode = None

        self.button_START["state"] = DISABLED
        self.button_STOP["state"] = DISABLED
        self.button_SELECT_ROBOT["state"] = DISABLED
        self.button_SELECT_SIMULATION["state"] = DISABLED
        self.button_SELECT_BOTH["state"] = DISABLED

        self.light_SELECT_ROBOT.turn_off()
        self.light_SELECT_SIMULATION.turn_off()
        self.light_SELECT_BOTH.turn_off()

        self.pressed_buttons.clear()

        if self.running:
            self.running = False
        else:
            self.stop_pressed()

    def stop_pressed(self):
        """
        Button method for when the program has run the stop command and the buttons should be clickable again.
        """
        self.button_START["state"] = DISABLED
        self.button_STOP["state"] = NORMAL
        self.button_SELECT_ROBOT["state"] = NORMAL
        self.button_SELECT_SIMULATION["state"] = NORMAL
        self.button_SELECT_BOTH["state"] = NORMAL

    def robot_pressed(self):
        """
        Button method for when clicking on the 'Execute program on robot'
        """
        self.pressed_buttons.add("Robot")

        self.button_START["state"] = NORMAL
        self.button_STOP["state"] = NORMAL
        self.button_SELECT_ROBOT["state"] = NORMAL
        self.button_SELECT_SIMULATION["state"] = DISABLED
        self.button_SELECT_BOTH["state"] = DISABLED

        self.light_SELECT_ROBOT.turn_ready()
        self.light_SELECT_SIMULATION.turn_off()
        self.light_SELECT_BOTH.turn_off()

    def simulation_pressed(self):
        """
        Button method for when clicking on the 'Execute program in simulation'
        """
        self.pressed_buttons.add("Simulation")

        self.button_START["state"] = NORMAL
        self.button_STOP["state"] = NORMAL
        self.button_SELECT_ROBOT["state"] = DISABLED
        self.button_SELECT_SIMULATION["state"] = NORMAL
        self.button_SELECT_BOTH["state"] = DISABLED

        self.light_SELECT_ROBOT.turn_off()
        self.light_SELECT_SIMULATION.turn_ready()
        self.light_SELECT_BOTH.turn_off()

    def both_pressed(self):
        """
        Button method for when clicking on the 'Execute program on robot and in simulation'
        """
        self.pressed_buttons.add("Robot and Simulation")

        self.button_START["state"] = NORMAL
        self.button_STOP["state"] = NORMAL
        self.button_SELECT_ROBOT["state"] = DISABLED
        self.button_SELECT_SIMULATION["state"] = DISABLED
        self.button_SELECT_BOTH["state"] = NORMAL

        self.light_SELECT_ROBOT.turn_off()
        self.light_SELECT_SIMULATION.turn_off()
        self.light_SELECT_BOTH.turn_ready()

    def run(self):
        """
        Runs the program
        """
        self.root.mainloop()


if __name__ == "__main__":
    test = GUI()
    test.run()
```

7.7 main.py File Reference

Data Structures

- class main.Main

Namespaces

- namespace main

Functions

- main.communication_multi_thread (communicator, command)

Variables

- bool main.BOTH = True
- bool main.SIMULATING = False
- int main.Z_OFFSET = 0
- int main.STORAGE_OFFSET_X = 30
- int main.SYRINGE_OFFSET_X = -18
- int main.SYRINGE_MOVEMENT_Z = 25
- int main.SPEED = 1/0.5
- main.ALLERGEN_AMOUNT = str(150)
- int main.END_EFFECTOR_TILT_TAKE = 85
- int main.END_EFFECTOR_TILT_PLACE = 89
- list main.PATCH_TEST_CORNER_COORDINATES = [-215, -265, 215]
- list main.SYRINGE_COORDINATES
- bool main.performing_command = False
- main.main = Main()

7.8 main.py

Go to the documentation of this file.

```
from robot_movement import MoveJ, MoveL, Robot, Kinematics, RobotMyCobot, RobotMyCobotAndSim
from gui import GUI
from robodk.robolink import *
from pymycobot.mycobot import MyCobot
from communication import CommunicationProtocol


import threading
import commands as cmd
import time as t
import math as m


BOTH = True
SIMULATING = False
Z_OFFSET = 0
STORAGE_OFFSET_X = 30
SYRINGE_OFFSET_X = -18
SYRINGE_MOVEMENT_Z = 25
SPEED = 1/0.5
ALLERGEN_AMOUNT = str(150)
END_EFFECTOR_TILT_TAKE = 85
END_EFFECTOR_TILT_PLACE = 89


PATCH_TEST_CORNER_COORDINATES = [-215, -265, 215]


SYRINGE_COORDINATES = [
    [305, 169.5+5, 153.5+5],
    [310, 94.5, 153.5+5],
    [310, 19.5, 153.5+5],
    [310, -65.5, 153.5+5],
    [305, -140.5, 153.5+5],
    [310, 132+5, 78],
    [310, 57, 76],
    [310, -28-5, 76],
    [310, -106-5, 76],
    [310, -184-5, 78]
]


performing_command = False


def communication_multi_thread(communicator, command):
    """
    Executes a command on the arduino using multi_threading
    :param communicator: instance of the communication class
    :param command: command to execute.
    """
    global performing_command

    performing_command = True
    communicator.send_command(command)
    performing_command = False


class Main:
    """
    Main class containing the whole program.
    """
    def __init__(self):
        """
        Connecting to ROBODK or MyCobot.
        """
        self.communication = CommunicationProtocol("COM3", 9600, 0.1)
        t.sleep(5)

        self.patch_test_config = [0, 1,
                                  2, 3,
                                  4, 5,
                                  6, 7,
                                  8, 9]

        self.robot = None

        self.gui_created = False
        self.gui_thread = threading.Thread(target=self.create_gui)
        self.gui_thread.start()

        while not self.gui_created:
            continue

        while not self.gui.running:
            continue

        self.set_mode(self.gui.mode)

    def create_gui(self):
        """
        Creates the GUI
        """
        self.gui = GUI()

        self.gui_created = True

        self.gui.run()

    def set_mode(self, mode):
        """
        Sets the mode for how the robot should run
        :param mode: either "both", "simulation", "robot".
        """
        if mode == "both":
            RDK = Robolink()
            rdk_link = RDK.Item('My Mechanism')
            my_cobot = MyCobot('COM4', 115200)

            self.robot = RobotMyCobotAndSim(my_cobot, rdk_link)
            self.robot.set_joints([0, 0, 0, 0, 0, 0], robot_speed=20)
            #self.communication.send_command(cmd.HOME_PRESSER)
            #self.communication.send_command(cmd.PREPARE_PRESSING)
            #self.communication.send_command(cmd.HOME_GRIPPER)
            #self.communication.send_command(cmd.CLOSE_GRIPPER)

        elif mode == "simulation":
            RDK = Robolink()
            rdk_link = RDK.Item('My Mechanism')
            self.robot = Robot(rdk_link)

        elif mode == "robot":
            my_cobot = MyCobot('COM4', 115200)

            my_cobot.set_free_mode(0)
            self.robot = RobotMyCobot(my_cobot)
            self.robot.set_joints([0, 0, 0, 0, 0, 0], robot_speed=20)

        self.run()

    def run(self):
        """
        Runs the whole program sequence
        """
        # Homing sekvens

        self.communication.send_command(cmd.HOME_PRESSER)
        self.communication.send_command(cmd.HOME_GRIPPER)

        self.communication.send_command(cmd.NEW_TEST)

        self.robot.set_joints([107.38158881112184, 46.157985403753784, 103.3563681945703, 30.485646401675925, 107.38158881112184, 135.0], robot_speed=20)

        t.sleep(5)

        for patch_text_index_number in range(len(self.patch_test_config)):

            self.take_allergen(self.patch_test_config[patch_text_index_number])

            MoveL(self.robot, [[260, -132-37.5, 153.5-Z_OFFSET, -90, 90, -90, 1.75 * SPEED]]).move_joint()
            MoveJ(self.robot, [[0, -350, PATCH_TEST_CORNER_COORDINATES[2]-Z_OFFSET, -90, 90, 135+45, 0.5 * SPEED], [PATCH_TEST_CORNER_COORDINATES[0], PATCH_TEST_CORNER_COORDINATES[1], PATCH_TEST_CORNER_COORDINATES[2]-Z_OFFSET, -90, 90, 135, 1 * SPEED]]).move_joint()

            self.dispense_allergen(patch_text_index_number)

            MoveJ(self.robot, [[0, -350, PATCH_TEST_CORNER_COORDINATES[2]-Z_OFFSET, -90, 90, 135+45, 0.5 * SPEED], [260, -132-37.5, 153.5-Z_OFFSET, -90, 90, -90, 1 * SPEED]]).move_joint()

            self.place_allergen(self.patch_test_config[patch_text_index_number])

            if not self.gui.running:
                break

        self.gui.running = False
        self.gui.stop_pressed()

        # Waiting on a new start
        while not self.gui.running:
            continue

        self.set_mode(self.gui.mode)

    def dispense_allergen(self, index):
        """
        Runs the movements needed to dispense allergen onto a patch test
        :param index: What index the position on the patch test has
        :return: None
        """
        x_offset = (index % 2) * 20.5
        y_offset = m.floor(index / 2) * 20.5

        MoveL(self.robot, [[PATCH_TEST_CORNER_COORDINATES[0] - x_offset, PATCH_TEST_CORNER_COORDINATES[1] + y_offset, PATCH_TEST_CORNER_COORDINATES[2]-Z_OFFSET, -90, 90, 135, 1 * SPEED]]).move_joint()
        MoveL(self.robot, [[PATCH_TEST_CORNER_COORDINATES[0] - x_offset, PATCH_TEST_CORNER_COORDINATES[1] + y_offset, PATCH_TEST_CORNER_COORDINATES[2]-Z_OFFSET-10, -90, 90, 135, 0.5 * SPEED]]).move_joint()

        while performing_command:
            continue

        self.communication.send_command(cmd.PRESS_ALLERGEN + ALLERGEN_AMOUNT)

        MoveL(self.robot, [[PATCH_TEST_CORNER_COORDINATES[0] - x_offset, PATCH_TEST_CORNER_COORDINATES[1] + y_offset, PATCH_TEST_CORNER_COORDINATES[2]-Z_OFFSET, -90, 90, 135, 0.5 * SPEED]]).move_joint()

        thread = threading.Thread(target=communication_multi_thread, args=(self.communication, cmd.HOME_PRESSER))
        thread.start()

        MoveL(self.robot, [[PATCH_TEST_CORNER_COORDINATES[0], PATCH_TEST_CORNER_COORDINATES[1], PATCH_TEST_CORNER_COORDINATES[2]-Z_OFFSET, -90, 90, 135, 1 * SPEED]]).move_joint()

    def take_allergen(self, index):
        """
        Takes an allergen from the storage
        :param index: Index for the allergen in the storage system
        :return: None
        """

        x = SYRINGE_COORDINATES[index][0]
        y = SYRINGE_COORDINATES[index][1]
        z = SYRINGE_COORDINATES[index][2]

        MoveL(self.robot, [[x + SYRINGE_OFFSET_X - STORAGE_OFFSET_X, y, z - Z_OFFSET, 0, END_EFFECTOR_TILT_TAKE, 0,
                            1.75 * SPEED]]).move_joint()

        MoveL(self.robot, [[x + SYRINGE_OFFSET_X, y, z - Z_OFFSET, 0, END_EFFECTOR_TILT_TAKE, 0,
                            0.75 * SPEED]]).move_joint()

        self.communication.send_command(cmd.CLOSE_GRIPPER)

        thread = threading.Thread(target=communication_multi_thread, args=(self.communication,
cmd.PREPARE_PRESSING))
        thread.start()

        MoveL(self.robot, [
            [x + SYRINGE_OFFSET_X, y, z + SYRINGE_MOVEMENT_Z - Z_OFFSET, 0, END_EFFECTOR_TILT_TAKE, 0,
             0.75 * SPEED]]).move_joint()
        MoveL(self.robot, [
            [x + SYRINGE_OFFSET_X - STORAGE_OFFSET_X, y, z + SYRINGE_MOVEMENT_Z - Z_OFFSET, 0,
END_EFFECTOR_TILT_TAKE, 0,
             0.75 * SPEED]]).move_joint()

    def place_allergen(self, index):
        """
        Places an allergen in storage
        :param index: Index for storage place
        :return: None
        """

        x = SYRINGE_COORDINATES[index][0]
        y = SYRINGE_COORDINATES[index][1]
        z = SYRINGE_COORDINATES[index][2]

        MoveL(self.robot, [[x + SYRINGE_OFFSET_X - STORAGE_OFFSET_X, y, z + SYRINGE_MOVEMENT_Z -
Z_OFFSET, 0, END_EFFECTOR_TILT_PLACE, 0, 1.75 * SPEED]]).move_joint()
        MoveL(self.robot, [[x + SYRINGE_OFFSET_X, y, z + SYRINGE_MOVEMENT_Z - Z_OFFSET, 0,
END_EFFECTOR_TILT_PLACE, 0, 0.75 * SPEED]]).move_joint()
        MoveL(self.robot, [[x + SYRINGE_OFFSET_X, y, z - Z_OFFSET, 0, END_EFFECTOR_TILT_PLACE, 0, 0.75
* SPEED]]).move_joint()

        while performing_command:
            continue

        self.communication.send_command(cmd.HOME_GRIPPER)

        MoveL(self.robot, [[x + SYRINGE_OFFSET_X - STORAGE_OFFSET_X, y, z - Z_OFFSET, 0,
END_EFFECTOR_TILT_PLACE, 0, 0.75 * SPEED]]).move_joint()


if __name__ == "__main__":
    t.sleep(2)

    main = Main()

```

7.9 robot_controller.py File Reference

Data Structures

- class robot_controller.RobotControllerSettings

Namespaces

- namespace robot_controller

7.10 robot_controller.py

Go to the documentation of this file.

```
import math as m


class RobotControllerSettings:
    """
    Robot controller settings class
    """
    def __init__(self):
        """
        Settings for the robot controller
        """
        self.update_frequency = 30 # hz
        self.linear_frequency = 10 # hz

        self.dh_values = {
            "alpha": [0, m.pi / 2, 0, 0, -m.pi / 2, m.pi / 2],
            "a": [0, 0, 135, 120, 0, 0],
            "d": [173.900, 0, 0, 88.78, 95, 65.5],
            "theta": [m.pi / 2, m.pi / 2, 0, -m.pi / 2, 0, 0]
        }


```

7.11 robot_movement.py File Reference

Data Structures

- class robot_movement.MoveNotPossible
- class robot_movement.JointSyntaxError
- class robot_movement.TakesOnlyTwoCoordinates
- class robot_movement.CoordinateSyntaxError
- class robot_movement.InvalidTimeIncrease
- class robot_movement.Kinematics
- class robot_movement.MoveJ
- class robot_movement.MoveL
- class robot_movement.Robot
- class robot_movement.RobotMyCobot
- class robot_movement.RobotMyCobotAndSim

Namespaces

- namespace robot_movement

Functions

- robot_movement.joint_matrix (alpha, a, d, theta)
- robot_movement.get_rot_x_matrix (angle)
- robot_movement.get_rot_y_matrix (angle)
- robot_movement.get_rot_z_matrix (angle)
- robot_movement.orientation_degree_to_radians (orientation)
- robot_movement.radians_to_degree (angle)
- robot_movement.radians_to_degree_list (list_radians)
- robot_movement.transform_list_into_range (untransformed_list, min_value, max_value, adjuster)
- robot_movement.best_end_joint (current_joint, joints)

Variables

- int robot_movement.MIN_ANGLE_LINK_1_5 = -165 / 180 ∗ m.pi
- int robot_movement.MAX_ANGLE_LINK_1_5 = 165 / 180 ∗ m.pi
- int robot_movement.MIN_ANGLE_LINK_6 = -175 / 180 ∗ m.pi
- int robot_movement.MAX_ANGLE_LINK_6 = 175 / 180 ∗ m.pi
- str robot_movement.MOVE_NOT_POSSIBLE_ERROR_MSG = "MoveNotPossible: The given movement command can not be executed by the program. Please make sure that it dies not result in a singularity"
- str robot_movement.JOINT_SYNTAX_ERROR_MSG = "JointSyntaxError: The syntax must be [Joint1, joint2, joint3, joint4, joint5, joint6] for a joint"
- str robot_movement.TAKES_ONLY_TWO_COORDINATES_ERROR_MSG = "TakesOnlyTwoCoordinates↩ : This class takes only two coordinates."
- str robot_movement.COORDINATE_SYNTAX_ERROR_MSG = "CoordinateSyntaxError: The syntax must be: [x, y, z, alpha, beta, gamma, time] for a coordinate."
- str robot_movement.INVALID_TIME_INCREASE_ERROR_MSG = "InvalidTimeIncrease: Later coordinates must have a higher time than those before or the robot will not be able to move to the given point."
- list robot_movement.robot_joint_position = [107.38158881112184, 46.157985403753784, 103.↩ 3563681945703, 30.485646401675925, 107.38158881112184, 135.0]
- list robot_movement.robot_cartesian_position = [280, 132, 74, -90, 135, -90]

7.12 robot_movement.py

Go to the documentation of this file.

```
import math as m
import time as t
import numpy as np
import random as r

from robot_controller import RobotControllerSettings
from numpy.linalg import inv

MIN_ANGLE_LINK_1_5 = -165 / 180 * m.pi
MAX_ANGLE_LINK_1_5 = 165 / 180 * m.pi
MIN_ANGLE_LINK_6 = -175 / 180 * m.pi
MAX_ANGLE_LINK_6 = 175 / 180 * m.pi

MOVE_NOT_POSSIBLE_ERROR_MSG = "MoveNotPossible: The given movement command can not be executed by the
program. Please make sure that it dies not result in a singularity"
JOINT_SYNTAX_ERROR_MSG = "JointSyntaxError: The syntax must be [Joint1, joint2, joint3, joint4,
joint5, joint6] for a joint"
TAKES_ONLY_TWO_COORDINATES_ERROR_MSG = "TakesOnlyTwoCoordinates: This class takes only two
coordinates."
COORDINATE_SYNTAX_ERROR_MSG = "CoordinateSyntaxError: The syntax must be: [x, y, z, alpha, beta,
gamma, time] for a coordinate."
INVALID_TIME_INCREASE_ERROR_MSG = "InvalidTimeIncrease: Later coordinates must have a higher time than
those before or the robot will not be able to move to the given point."


robot_joint_position = [107.38158881112184, 46.157985403753784, 103.3563681945703, 30.485646401675925,
107.38158881112184, 135.0]
robot_cartesian_position = [280, 132, 74, -90, 135, -90]


class MoveNotPossible(Exception):
    """
    Used for error handling. Also the reason for the inheritance of Exception. The following error is
described as the following:

    MoveNotPossible: The given movement command can not be executed by the program. Please make sure
that it dies not result in a singularity
    """
    pass


class JointSyntaxError(Exception):
    """
    Used for error handling. Also the reason for the inheritance of Exception. The following error is
described as the following:

    JointSyntaxError: The syntax must be [Joint1, joint2, joint3, joint4, joint5, joint6] for a joint
    """
    pass


class TakesOnlyTwoCoordinates(Exception):
    """
    Used for error handling. Also the reason for the inheritance of Exception. The following error is described as the following:

    TakesOnlyTwoCoordinates: This class takes only two coordinates.
    """
    pass


class CoordinateSyntaxError(Exception):
    """
    Used for error handling. Also the reason for the inheritance of Exception. The following error is described as the following:

    CoordinateSyntaxError: The syntax must be: [x, y, z, alpha, beta, gamma, time] for a coordinate.
    """
    pass


class InvalidTimeIncrease(Exception):
    """
    Used for error handling. Also the reason for the inheritance of Exception. The following error is described as the following:

    InvalidTimeIncrease: Later coordinates must have a higher time than those before or the robot will not be able to move to the given point.
    """
    pass


np.set_printoptions(suppress=True)


def joint_matrix(alpha, a, d, theta):
    """
    Function to create transformation matrix.

    :param alpha: alpha value from David Hartenberg parameters.
    :param a: a value from David Hartenberg parameters.
    :param d: d value from David Hartenberg parameters.
    :param theta: theta value from David Hartenberg parameters.
    :return: returns a 4x4 matrix containing the transformation matrix for the given parameters.
    """
    matrix = np.matrix([[m.cos(theta), -m.sin(theta), 0, a],
                        [m.sin(theta) * m.cos(alpha), m.cos(theta) * m.cos(alpha), -m.sin(alpha), -m.sin(alpha) * d],
                        [m.sin(theta) * m.sin(alpha), m.cos(theta) * m.sin(alpha), m.cos(alpha), m.cos(alpha) * d],
                        [0, 0, 0, 1]])

    return matrix


def get_rot_x_matrix(angle):
    """
    Get rotation matrix around x axis.
    :param angle: Angle the rotation matrix should rotate an object.
    :return: Returns the rotation matrix.
    """
    rot_x_matrix = np.matrix([[1, 0, 0, 0],
                              [0, m.cos(angle), -m.sin(angle), 0],
                              [0, m.sin(angle), m.cos(angle), 0],
                              [0, 0, 0, 1]])

    return rot_x_matrix


def get_rot_y_matrix(angle):
    """
    Get rotation matrix around y axis.
    :param angle: Angle the rotation matrix should rotate an object.
    :return: Returns the rotation matrix.
    """
    rot_y_matrix = np.matrix([[m.cos(angle), 0, m.sin(angle), 0],
                              [0, 1, 0, 0],
                              [-m.sin(angle), 0, m.cos(angle), 0],
                              [0, 0, 0, 1]])

    return rot_y_matrix


def get_rot_z_matrix(angle):
    """
    Get rotation matrix around z axis.
    :param angle: Angle the rotation matrix should rotate an object.
    :return: Returns the rotation matrix.
    """
    rot_z_matrix = np.matrix([[m.cos(angle), -m.sin(angle), 0, 0],
                              [m.sin(angle), m.cos(angle), 0, 0],
                              [0, 0, 1, 0],
                              [0, 0, 0, 1]])

    return rot_z_matrix


def orientation_degree_to_radians(orientation):
    """
    Converts angle from degree to radians
    :param orientation: Angle in degree.
    :return: Angle in radians
    """
    for angle_index in range(len(orientation)):
        orientation[angle_index] = orientation[angle_index] / 180 * m.pi

    return orientation


def radians_to_degree(angle):
    """
    Converts angle from radians to degree.
    :param angle: Angle in radians.
    :return: Angle in degree.
    """
    angle = angle / m.pi * 180

    return angle


def radians_to_degree_list(list_radians):
    """
    Converts radians in a list to degree
    :param list_radians: List containing angles in radians
    :return: List containing angles in degrees
    """
    response = []

    for angle in list_radians:
        response.append(angle / m.pi * 180)

    return response


def transform_list_into_range(untransformed_list, min_value, max_value, adjuster):
    """
    Takes a list containing angles and checks if they are in a given range.
    :param untransformed_list: List containing angles
    :param min_value: Min value in range
    :param max_value: Max value in range
    :param adjuster: How much the values may be adjusted
    :return: Transformed list. Some of the elements may have been changed to False if the given value is not in the range.
    """
    for element_index in range(len(untransformed_list)):
        if not untransformed_list[element_index]:
            continue

        while untransformed_list[element_index] < min_value:
            untransformed_list[element_index] += adjuster
            if untransformed_list[element_index] > max_value:
                untransformed_list[element_index] = False

        while untransformed_list[element_index] > max_value:
            untransformed_list[element_index] -= adjuster
            if untransformed_list[element_index] < min_value:
                untransformed_list[element_index] = False
    return untransformed_list


def best_end_joint(current_joint, joints):
    """
    Takes the current joint position and a list containing of possible joint positions and returns the closest possible joint position.
    :param current_joint: The current position of the robots joint.
    :param joints: The possible joint positions for a point.
    :return: The closest joint position to the current joint position
    """
    lowest_score_joint = []
    lowest_score_value = -1

    if len(joints) == 0:
        raise MoveNotPossible(MOVE_NOT_POSSIBLE_ERROR_MSG)

    for joint in joints:
        angle_difference = 0

        for angle_index in range(len(joint)):
            angle_difference += abs(current_joint[angle_index] - joint[angle_index])

        if angle_difference < lowest_score_value or lowest_score_value < 0:
            lowest_score_value = angle_difference
            lowest_score_joint = joint

    return lowest_score_joint


class Kinematics:
    """
    Class to compute the inverse kinematics.
    """
    def __init__(self):
        """
        Constructor for class
        """
        self.robot_controller_settings = RobotControllerSettings()
        self.dh_value_alpha = self.robot_controller_settings.dh_values["alpha"]
        self.dh_value_a = self.robot_controller_settings.dh_values["a"]
        self.dh_value_d = self.robot_controller_settings.dh_values["d"]
        self.dh_value_theta = self.robot_controller_settings.dh_values["theta"]

    def get_joint_values(self, coordinate, orientation):
        """
        Method for getting the joint values from a coordinate and orientation.
        :param coordinate: Coordinate in the format [x, y, z].
        :param orientation: Orientation in the format [roll, pitch, yaw].
        :return: Returns a list containing all possible joint configurations for the robot.
        """
        orientation = orientation_degree_to_radians(orientation)

        rot_x_matrix = get_rot_x_matrix(orientation[0])
        rot_y_matrix = get_rot_y_matrix(orientation[1])
        rot_z_matrix = get_rot_z_matrix(orientation[2])

        wrist_rot_x_matrix = get_rot_x_matrix(0)
        wrist_rot_y_matrix = get_rot_y_matrix(0)
        wrist_rot_z_matrix = get_rot_z_matrix(0)

        wrist = wrist_rot_z_matrix * wrist_rot_y_matrix * wrist_rot_x_matrix

        wrist[0, 3] = 0
        wrist[1, 3] = 0
        wrist[2, 3] = 66

        T_link0_wrist = rot_z_matrix * rot_y_matrix * rot_x_matrix
        T_link0_wrist[0, 3] = coordinate[0]
        T_link0_wrist[1, 3] = coordinate[1]
        T_link0_wrist[2, 3] = coordinate[2]

        T_link0_link6 = T_link0_wrist * inv(wrist)

        theta_1 = self.kin_calculate_theta_1(T_link0_link6)  # Contains two angles

        # For calculating theta_4, theta_5 and theta_6 a rotation matrix consisting of theta_1 and the
other angles is needed. This is created:
        rot_z_matrix_theta_1_1 = get_rot_z_matrix(theta_1[0] + self.dh_value_theta[0])
        rot_z_matrix_theta_1_2 = get_rot_z_matrix(theta_1[1] + self.dh_value_theta[0])

        rot_matrix_alpha_link_1 = get_rot_x_matrix(self.dh_value_alpha[0])
        rot_matrix_alpha_link_2 = get_rot_x_matrix(self.dh_value_alpha[1])
        rot_matrix_alpha_link_3 = get_rot_x_matrix(self.dh_value_alpha[2])

        rot_matrix_theta_link_2 = get_rot_z_matrix(self.dh_value_theta[1])
        rot_matrix_theta_link_3 = get_rot_z_matrix(self.dh_value_theta[2])

        rot_matrix_link_1_theta_1_1_inv = inv(rot_matrix_alpha_link_1 * rot_z_matrix_theta_1_1)
        rot_matrix_link_1_theta_1_2_inv = inv(rot_matrix_alpha_link_1 * rot_z_matrix_theta_1_2)
        rot_matrix_link_2_inv = inv(rot_matrix_alpha_link_2 * rot_matrix_theta_link_2)
        rot_matrix_link_3_inv = inv(rot_matrix_alpha_link_3 * rot_matrix_theta_link_3)

        rot_matrix_link_1_to_link_3_theta_1_1_inv = rot_matrix_link_3_inv * rot_matrix_link_2_inv *
rot_matrix_link_1_theta_1_1_inv
        rot_matrix_link_1_to_link_3_theta_1_2_inv = rot_matrix_link_3_inv * rot_matrix_link_2_inv *
rot_matrix_link_1_theta_1_2_inv

        rot_matrix_theta_1_1 = rot_matrix_link_1_to_link_3_theta_1_1_inv * rot_z_matrix * rot_y_matrix
* rot_x_matrix
        rot_matrix_theta_1_2 = rot_matrix_link_1_to_link_3_theta_1_2_inv * rot_z_matrix * rot_y_matrix
* rot_x_matrix
```

```

        # Based on the rotational matrix, it is possible to get the angle theta5 using the inverse cosinus to field
        # (3,3) in the matrix. However due to the nature of cosinus to a point, it will not be possible to estimate if
        # the angle is positive or negative.

        theta_5 = self.kin_calculate_theta_5(rot_matrix_theta_1_1) + self.kin_calculate_theta_5(rot_matrix_theta_1_2)

        theta_6_theta_1_1 = [self.kin_calculate_theta_6(rot_matrix_theta_1_1, theta_5[0]),
                             self.kin_calculate_theta_6(rot_matrix_theta_1_1, theta_5[1])]
        theta_6_theta_1_2 = [self.kin_calculate_theta_6(rot_matrix_theta_1_2, theta_5[2]),
                             self.kin_calculate_theta_6(rot_matrix_theta_1_2, theta_5[3])]
        theta_6 = theta_6_theta_1_1 + theta_6_theta_1_2

        # Moving the

        pre_theta_4_theta_1_1 = [self.kin_calculate_theta_4(rot_matrix_theta_1_1, theta_5[0]),
                                 self.kin_calculate_theta_4(rot_matrix_theta_1_1, theta_5[1])]
        pre_theta_4_theta_1_2 = [self.kin_calculate_theta_4(rot_matrix_theta_1_2, theta_5[2]),
                                 self.kin_calculate_theta_4(rot_matrix_theta_1_2, theta_5[3])]
        pre_theta_4 = pre_theta_4_theta_1_1 + pre_theta_4_theta_1_2

        # Currently theta5 and theta6 consists of 4 different angles whereas theta1 consists of 2 angles.
        # To get each angle to match in the angle, theta1 will be defined as:

        theta_1 = [theta_1[0], theta_1[0], theta_1[1], theta_1[1]]

        # This gives the possibility to make a loop of 4 as each array will consist of this:

        inverse_kinematics_list = []
        theta_1 = transform_list_into_range(theta_1, MIN_ANGLE_LINK_1_5, MAX_ANGLE_LINK_1_5, 2 * m.pi)
        theta_5 = transform_list_into_range(theta_5, MIN_ANGLE_LINK_1_5, MAX_ANGLE_LINK_1_5, 2 * m.pi)
        theta_6 = transform_list_into_range(theta_6, MIN_ANGLE_LINK_6, MAX_ANGLE_LINK_6, 2 * m.pi)

        for i in range(4):
            # Checking if any of them are out of range (Joint space more specifically).
            if not theta_1[i] or not theta_5[i] or not theta_6[i]:
                continue

            T_link0_link1_rotation = joint_matrix(self.dh_value_alpha[0], self.dh_value_a[0], self.dh_value_d[0],
                                                  self.dh_value_theta[0] + theta_1[i])
            T_link4_rotation_link4 = joint_matrix(0, 0, self.dh_value_d[3], 0)
            T_link4_link5 = joint_matrix(self.dh_value_alpha[4], self.dh_value_a[4], self.dh_value_d[4],
                                         self.dh_value_theta[4] + theta_5[i])
            T_link5_link6 = joint_matrix(self.dh_value_alpha[5], self.dh_value_a[5], self.dh_value_d[5],
                                         self.dh_value_theta[5] + theta_6[i])

            T_link1_link3 = inv(T_link0_link1_rotation) * T_link0_link6 * inv(T_link5_link6) * inv(T_link4_link5) * inv(
                T_link4_rotation_link4)

            # Sorting points which are further away than the robot can reach.
            if m.sqrt(T_link1_link3[0, 3] ** 2 + T_link1_link3[2, 3] ** 2) > self.dh_value_a[2] + self.dh_value_a[3]:
                continue

            theta_2_theta_3 = self.kin_calculate_theta_2_3(T_link1_link3)

            theta_2_untransformed = theta_2_theta_3[0]
            theta_3_untransformed = theta_2_theta_3[1]

            theta_2 = transform_list_into_range(theta_2_untransformed, MIN_ANGLE_LINK_1_5, MAX_ANGLE_LINK_1_5, 2 * m.pi)
            theta_3 = transform_list_into_range(theta_3_untransformed, MIN_ANGLE_LINK_1_5, MAX_ANGLE_LINK_1_5, 2 * m.pi)

            theta_4_untransformed = [False, False]

            for angle_index in range(2):
                if theta_2[angle_index] and theta_3[angle_index]:
                    theta_4_untransformed[angle_index] = pre_theta_4[i] - theta_2[angle_index] - theta_3[angle_index]

            theta_4 = transform_list_into_range(theta_4_untransformed, MIN_ANGLE_LINK_1_5, MAX_ANGLE_LINK_1_5, 2 * m.pi)

            # Creates the list.
            for j in range(2):
                # Checking if any of them are out of range (Joint space more specifically).
                if not theta_2[j] or not theta_3[j] or not theta_4[j]:
                    continue

                inverse_kinematics_list.append(
                    radians_to_degree_list([theta_1[i], theta_2[j], theta_3[j], theta_4[j], theta_5[i], theta_6[i]]))

        return inverse_kinematics_list

    # Returns two angles.
    def kin_calculate_theta_1(self, T_link0_link6):
        """
        Calculates theta 1
        :param T_link0_link6: Transformation matrix from link 1 to link 6.
        :return: Returns the two possible values for theta 1
        """
        T_link6_rotation_link6 = joint_matrix(0, 0, self.dh_value_d[5], 0)

        T_link0_link6_rotation = T_link0_link6 * inv(T_link6_rotation_link6)

        x_6 = T_link0_link6_rotation[0, 3]
        y_6 = T_link0_link6_rotation[1, 3]

        theta_1_1 = (m.acos(self.dh_value_d[3] / ((x_6 ** 2 + y_6 ** 2) ** 0.5)) + m.atan2(y_6, x_6))
        theta_1_2 = (-m.acos(self.dh_value_d[3] / ((x_6 ** 2 + y_6 ** 2) ** 0.5)) + m.atan2(y_6, x_6))

        return [theta_1_1, theta_1_2]

    def kin_calculate_theta_2_3(self, T_link2_link3):
        """
        Calculates theta 2 and 3
        :param T_link2_link3: Transformation matrix from link 2 to link 3
        :return: Returns the possible configurations for theta 2 and theta 3.
        """
        x = abs(T_link2_link3.item((0, 3)))
        z = T_link2_link3.item((2, 3))

        mirror_angle_length = m.sqrt(x ** 2 + z ** 2)

        direction = -1 * abs(T_link2_link3.item((0, 3))) / T_link2_link3.item((0, 3))

        theta_2_1 = direction * (m.acos(z / mirror_angle_length) + m.acos(
            (self.dh_value_a[3] ** 2 - self.dh_value_a[2] ** 2 - mirror_angle_length ** 2) / (
                    2 * self.dh_value_a[2] * mirror_angle_length)) + m.pi)
        theta_3_1 = direction * (
            m.acos((mirror_angle_length ** 2 - self.dh_value_a[3] ** 2 - self.dh_value_a[2] ** 2) / (
                    2 * self.dh_value_a[2] * self.dh_value_a[3])))

        theta_2_2 = direction * (m.acos(z / mirror_angle_length) - m.acos(
            (self.dh_value_a[3] ** 2 - self.dh_value_a[2] ** 2 - mirror_angle_length ** 2) / (
                    2 * self.dh_value_a[2] * mirror_angle_length)) - m.pi)
        theta_3_2 = -1 * direction * (
            m.acos((mirror_angle_length ** 2 - self.dh_value_a[3] ** 2 - self.dh_value_a[2] ** 2) / (
                    2 * self.dh_value_a[2] * self.dh_value_a[3])))

        theta_2 = [theta_2_1, theta_2_2]
        theta_3 = [theta_3_1, theta_3_2]

        return theta_2, theta_3

    def kin_calculate_theta_4(self, rot_matrix, theta_5):
        """
        Calculates theta 4
        :param rot_matrix: Rotation matrix describing rotation from link 4 to link 6.
        :param theta_5: Value for theta 5
        :return: Returns the value for theta 4
        """
        # -self.dh_value_theta[3] is subtracted due to the calculated angle is based on theta2+theta3+theta4+self.dh_value_theta4 = calculated angle.
        theta_4 = m.atan2(rot_matrix.item((1, 2)) * m.sin(theta_5), rot_matrix.item((0, 2)) * m.sin(theta_5)) - \
                  self.dh_value_theta[3]

        return theta_4

    def kin_calculate_theta_5(self, rot_matrix):
        """
        Calculates theta 5
        :param rot_matrix: Rotation matrix describing rotation from link 4 to link 6.
        :return: Returns the two values for theta 5
        """
        theta_5_unsigned = m.acos(rot_matrix.item((2, 2)))

        theta_5 = [theta_5_unsigned, -theta_5_unsigned]

        return theta_5

    def kin_calculate_theta_6(self, rot_matrix, theta_5):
        """
```

```
         Calculates theta 6
         :param rot_matrix: Rotation matrix describing rotation from link 4 to link 6.
         :param theta_5: Value for theta 5
         :return: Returns the value for theta 6
         """
        theta_6 = m.atan2(rot_matrix.item((2, 1)) * m.sin(theta_5), -1 * rot_matrix.item((2, 0)) *
m.sin(theta_5))

        return theta_6


class MoveJ(Kinematics):
    """
    Class for moving in joint space
    """
    def __init__(self, robot_pointer, positions_with_time):
        """
        Constructor for the MoveJ class. Takes the robot and position.
        :param robot_pointer: A pointer to the robot class for the robot
        :param positions_with_time: The position with time.
        """
        self.robot = robot_pointer
        self.pos_w_time = positions_with_time
        self.start_joint_values = robot_joint_position

        start_time = 0

        if not len(self.start_joint_values) == 6:
            raise JointSyntaxError(JOINT_SYNTAX_ERROR_MSG)

        for coordinate in self.pos_w_time:
            if not len(coordinate) == 7:
                raise CoordinateSyntaxError(COORDINATE_SYNTAX_ERROR_MSG)

            if coordinate[6] <= start_time:
                raise InvalidTimeIncrease(INVALID_TIME_INCREASE_ERROR_MSG)

            start_time = coordinate[6]

        super().__init__()

        self.joint_matrix = self.convert_coordinates_to_joint_movements()

    def convert_coordinates_to_joint_movements(self):
        """
        Method that converts the coordinates into joint degrees for each via point and end position.
        :return: Returns the transformed matrix.
        """
        start_joint = self.start_joint_values
        joint_values = []

        for position_index in range(len(self.pos_w_time)):
            position = self.pos_w_time[position_index]

            coordinate = position[0:3]
            orientation = position[3:6]
            time_val = position[6]

            joints = self.get_joint_values(coordinate, orientation)

            if position_index == 0:
                joint_values.append([start_joint, 0])

                start_joint = best_end_joint(start_joint, joints)
                joint_values.append([start_joint, time_val])

            else:
                start_joint = best_end_joint(start_joint, joints)
                joint_values.append([start_joint, time_val])

        return joint_values

    def move_joint(self):
        """
        Method for moving the robot in the joint space thrugh the different via points.
        """
        global robot_joint_position, robot_cartesian_position

        self.joint_matrix = self.convert_coordinates_to_joint_movements()

        end_joint_position = []

        for move_index in range(
                len(self.joint_matrix) - 1):  # The reason for -1 is because the function is indexing
1 ahead.
            time_val = self.joint_matrix[move_index][1]
            time_val_plus_1 = self.joint_matrix[move_index + 1][1]

            joint_movement_time = time_val_plus_1 - time_val

            joint_function_values = []

            for joint_index in range(len(self.joint_matrix[move_index][0])):
                v_start = 0
                v_end = 0

                joint_val = self.joint_matrix[move_index][0][joint_index]
                joint_val_plus_1 = self.joint_matrix[move_index + 1][0][joint_index]

                if not move_index == 0:
                    joint_val_minus_1 = self.joint_matrix[move_index - 1][0][joint_index]
                    time_val_minus_1 = self.joint_matrix[move_index - 1][1]

                    v_start = ((joint_val - joint_val_minus_1) / (time_val - time_val_minus_1) + (
                                (joint_val_plus_1 - joint_val) / (time_val_plus_1 - time_val))) / 2

                if not move_index == len(self.joint_matrix) - 2:
                    joint_val_plus_2 = self.joint_matrix[move_index + 2][0][joint_index]
                    time_val_plus_2 = self.joint_matrix[move_index + 2][1]

                    v_end = ((joint_val_plus_1 - joint_val) / (time_val_plus_1 - time_val) + (
                                (joint_val_plus_2 - joint_val_plus_1) / (time_val_plus_2 - time_val_plus_1))) / 2

                a_0 = joint_val
                a_1 = v_start
                a_2 = 3 / (joint_movement_time ** 2) * (
                            joint_val_plus_1 - joint_val) - 2 / joint_movement_time * v_start - 1 / joint_movement_time * v_end
                a_3 = (-2) / (joint_movement_time ** 3) * (joint_val_plus_1 - joint_val) + 1 / (
                            joint_movement_time ** 2) * (v_end + v_start)

                joint_function_values.append([a_0, a_1, a_2, a_3])

            for freq in range(round(joint_movement_time * self.robot_controller_settings.update_frequency)):
                joints_value = []
                for fp in joint_function_values:  # fp: function parameters
                    x = freq / self.robot_controller_settings.update_frequency

                    function_value = fp[0] + fp[1] * x + fp[2] * x ** 2 + fp[3] * x ** 3
                    joints_value.append(function_value)

                self.robot.set_joints(joints_value)
                end_joint_position = joints_value
                t.sleep(1 / self.robot_controller_settings.update_frequency)

        robot_joint_position = end_joint_position
        robot_cartesian_position = self.pos_w_time[-1][:6]


# IKKE OPDATERET UD FRA NYESTE EXCEPTIONS!
class MoveL(MoveJ):
    """
    Class for moving in cartesian space.
    """
    def __init__(self, robot_pointer, positions_with_time):
        """
        Constructor for MoveL class
        :param robot_pointer: Pointer to robot controller
        :param positions_with_time: Position for where the robot should move to.
        """
        self.robotrobot = robot_pointer
        self.pos_w_time_linear = positions_with_time
        self.move_time = self.pos_w_time_linear[0][6]

        super().__init__(self.robotrobot, self.pos_w_time_linear)

        if not len(positions_with_time) == 1:
            raise TakesOnlyTwoCoordinates(TAKES_ONLY_TWO_COORDINATES_ERROR_MSG)

        start_time = 0

        for coordinate in self.pos_w_time_linear:
            if not len(coordinate) == 7:
                raise CoordinateSyntaxError(COORDINATE_SYNTAX_ERROR_MSG)

            if coordinate[6] <= start_time:
                raise InvalidTimeIncrease(INVALID_TIME_INCREASE_ERROR_MSG)

            start_time = coordinate[6]

        self.start_pos_cartesian = robot_cartesian_position

```

```python
        self.pos_w_time_linear = self.get_linear_via_points()
        self.pos_w_timepos_w_time = self.pos_w_time_linear

    def get_linear_via_points(self):
        """
        Method for getting all via points on the path, the robot moves on.
        :return: Returns the coordinates which will be used as via points in the joint movement.
        """
        linear_movement_coords_w_time = []

        function_values = []

        for i in range(6):
            a_0 = self.start_pos_cartesian[i]
            a_1 = 0
            a_2 = 3 / (self.move_time ** 2) * (self.pos_w_time_linear[0][i] - self.start_pos_cartesian[i])
            a_3 = (-2) / (self.move_time ** 3) * (self.pos_w_time_linear[0][i] - self.start_pos_cartesian[i])

            function_values.append([a_0, a_1, a_2, a_3])

        for i in range(1, round(self.move_time * self.robot_controller_settings.linear_frequency + 1)):

            coordinate = []

            x = i / self.robot_controller_settings.linear_frequency

            for j in range(6):
                coordinate.append(function_values[j][0] + function_values[j][1] * x + function_values[j][2] * x ** 2 + function_values[j][3] * x ** 3)

            coordinate.append(x)

            linear_movement_coords_w_time.append(coordinate)

        return linear_movement_coords_w_time


class Robot:
    """
    Robot class used to talk with RoboDK and the correct robot
    """
    def __init__(self, rdk_link):
        """
        Constructor for class
        """
        self.robot_settings = RobotControllerSettings()

        self.rdk_link = rdk_link
        self.prev_position = [0, 0, 0, 0, 0, 0]
        self.time_between_move = 1 / self.robot_settings.update_frequency
        self.max_joint_speed = 180

    def set_joints(self, position, robot_speed=100):
        """
        Sets the joint values for the robot
        :param position: Position in joint space.
        :param robot_speed: Not used due to it not being neccessary, but has to be there in order to use the class together with other classes.'
        """

        # Prints if one of the joins exceed maximum speed.
        for index in range(6):
            speed = abs((position[index] - self.prev_position[index]) / self.time_between_move)
            if speed > self.max_joint_speed:
                print(f"Joint {index+1} exceeded max speed in simulation at {position}")

        self.prev_position = position

        self.rdk_link.setJoints(position)


class RobotMyCobot:
    """
    Robot class used to talk with MyCobot 320 PI
    """
    def __init__(self, robo_link):
        """
        Constructor for class
        """
        self.robo_link = robo_link

    def set_joints(self, position, robot_speed=100):
        """
        Sets the joint values for the robot
        :param position: Position in joint space.
        :param robot_speed: Value between 0-100 setting the robot's speed.
        """
        encoder_val = []
        encoder_dir = [1, 1, -1, 1, 1, 1]

        for joint_val_index in range(len(position)):
            encoder_val.append(round(2048 - position[joint_val_index] / 90 * 1024 *
encoder_dir[joint_val_index]))

        self.robo_link.set_encoders(encoder_val, robot_speed)


class RobotMyCobotAndSim:
    """
    Robot class used to talk with MyCobot 320 PI and RoboDK
    """
    def __init__(self, robo_link, rdk_link):
        """
        Constructor for class
        """
        self.rdk_link = rdk_link
        self.robo_link = robo_link

    def set_joints(self, position, robot_speed=100):
        """
        Sets the joint values for the robot
        :param position: Position in joint space.
        :param robot_speed: Value between 0-100 setting the robot's speed.
        """
        encoder_val = []
        encoder_dir = [1, 1, -1, 1, 1, 1]

        for joint_val_index in range(len(position)):
            encoder_val.append(round(2048 - position[joint_val_index] / 90 * 1024 *
encoder_dir[joint_val_index]))

        self.robo_link.set_encoders(encoder_val, robot_speed)

        self.rdk_link.setJoints(position)

```

7.13 test.py File Reference

Namespaces

- namespace test

Functions

- test.mySqrt (number, guess, step, tol)

Variables

- int test.testVal = 9

7.14 test.py

Go to the documentation of this file.

```
def mySqrt(number, guess, step, tol):
    # We need to take out negative numbers...
    if (number < 0):
        print('Error - we do not work with complex numbers here...')
        return float("NaN")

    # If we set guess to zero, we have to provide a number - we assume this is the initial call
    if (guess == 0):
        if (number > 1):  # If we have numbers larger than one, we can safely guess half as the sqrt
            guess = 0.5 * number
        else:
            guess = number * 2  # If we have numbers smaller than one, we need to double our guess

    tmp = guess * guess  # Now compute the square of our guess
    if ((tmp - number) < tol):  # Check if the (guess^2 - number) is lower than our tolerance level
        return guess
    else:
        if (tmp > number):  # If our guess was too high, then iterate by calling ourselves again with a slightly lower guess
            return mySqrt(number, (1 - step) * guess, step, tol)
        else:  # Else, our guess was too small, we need to increase the guess for our next call
            return mySqrt(number, (1 + step) * guess, step, tol)


testVal = 9
print(’Squareroot of ’ + str(testVal) + ’ is ’)
print(mySqrt(testVal, 0, 0.001, 0.001))
```

Index

__init__
    communication.CommunicationProtocol, 23
    gui.GUI, 26
    gui.Light, 33
    main.Main, 34
    robot_controller.RobotControllerSettings, 40
    robot_movement.Kinematics, 45
    robot_movement.MoveJ, 51
    robot_movement.MoveL, 56
    robot_movement.Robot, 59
    robot_movement.RobotMyCobot, 61
    robot_movement.RobotMyCobotAndSim, 62

ALLERGEN_AMOUNT
    main, 12

baudrate
    communication.CommunicationProtocol, 24
best_end_joint
    robot_movement, 15
BOTH
    main, 12
both_pressed
    gui.GUI, 27
button_SELECT_BOTH
    gui.GUI, 29
button_SELECT_ROBOT
    gui.GUI, 29
button_SELECT_SIMULATION
    gui.GUI, 29
button_START
    gui.GUI, 29
button_STOP
    gui.GUI, 30

CLOSE_GRIPPER
    commands, 9
com
    communication.CommunicationProtocol, 24
com_test
    communication, 11
commands, 9
    CLOSE_GRIPPER, 9
    EMPTY, 9
    HOME_GRIPPER, 9
    HOME_PRESSER, 9
    NEW_TEST, 10
    PREPARE_PRESSING, 10
    PRESS_ALLERGEN, 10
    RESPONSE, 10
commands.py, 65
communication, 10
    com_test, 11
    main.Main, 38
    SERIAL_COMMUNICATION_UNKNOWN_RESPONSE_ERROR_MS
        11
communication.CommunicationProtocol, 23
    __init__, 23
    baudrate, 24
    com, 24
    connection, 24
    send_command, 24
    timeout, 24
communication.py, 65
communication.SerialCommunicationUnknownResponse,
        25
communication_multi_thread
    main, 12
connection
    communication.CommunicationProtocol, 24
convert_coordinates_to_joint_movements
    robot_movement.MoveJ, 52
COORDINATE_SYNTAX_ERROR_MSG
    robot_movement, 19
create_gui
    main.Main, 35

dh_value_a
    robot_movement.Kinematics, 49
dh_value_alpha
    robot_movement.Kinematics, 49
dh_value_d
    robot_movement.Kinematics, 49
dh_value_theta
    robot_movement.Kinematics, 49
dh_values
    robot_controller.RobotControllerSettings, 40
dispense_allergen
    main.Main, 35

EMPTY
    commands, 9
END_EFFECTOR_TILT_PLACE
    main, 12
END_EFFECTOR_TILT_TAKE
    main, 12

get_joint_values
    robot_movement.Kinematics, 45
get_linear_via_points

robot_movement.MoveL, 56
get_rot_x_matrix
robot_movement, 15
get_rot_y_matrix
robot_movement, 16
get_rot_z_matrix
robot_movement, 16
gui, 11
main.Main, 38
test, 11
gui.GUI, 26
__init__, 26
both_pressed, 27
button_SELECT_BOTH, 29
button_SELECT_ROBOT, 29
button_SELECT_SIMULATION, 29
button_START, 29
button_STOP, 30
light_SELECT_BOTH, 30
light_SELECT_ROBOT, 30
light_SELECT_SIMULATION, 30
mode, 30
pressed_buttons, 30
robot_pressed, 27
root, 31
run, 27
running, 31
simulation_pressed, 27
start_pressed, 28
stop_pressed, 28
stop_pressed_init, 28
title, 31
gui.Light, 32
__init__, 33
oval_id, 34
turn_off, 33
turn_on, 33
turn_ready, 33
gui.py, 66
gui_created
main.Main, 38
gui_thread
main.Main, 38

HOME_GRIPPER
commands, 9
HOME_PRESSER
commands, 9

INVALID_TIME_INCREASE_ERROR_MSG
robot_movement, 19

joint_matrix
robot_movement, 17
robot_movement.MoveJ, 53
JOINT_SYNTAX_ERROR_MSG
robot_movement, 19

kin_calculate_theta_1
robot_movement.Kinematics, 46
kin_calculate_theta_2_3
robot_movement.Kinematics, 47
kin_calculate_theta_4
robot_movement.Kinematics, 47
kin_calculate_theta_5
robot_movement.Kinematics, 48
kin_calculate_theta_6
robot_movement.Kinematics, 48

light_SELECT_BOTH
gui.GUI, 30
light_SELECT_ROBOT
gui.GUI, 30
light_SELECT_SIMULATION
gui.GUI, 30
linear_frequency
robot_controller.RobotControllerSettings, 40

main, 11
ALLERGEN_AMOUNT, 12
BOTH, 12
communication_multi_thread, 12
END_EFFECTOR_TILT_PLACE, 12
END_EFFECTOR_TILT_TAKE, 12
main, 13
PATCH_TEST_CORNER_COORDINATES, 13
performing_command, 13
SIMULATING, 13
SPEED, 13
STORAGE_OFFSET_X, 13
SYRINGE_COORDINATES, 13
SYRINGE_MOVEMENT_Z, 14
SYRINGE_OFFSET_X, 14
Z_OFFSET, 14
main.Main, 34
__init__, 34
communication, 38
create_gui, 35
dispense_allergen, 35
gui, 38
gui_created, 38
gui_thread, 38
patch_test_config, 39
place_allergen, 35
robot, 39
run, 36
set_mode, 37
take_allergen, 37
main.py, 69
MAX_ANGLE_LINK_1_5
robot_movement, 19
MAX_ANGLE_LINK_6
robot_movement, 20
max_joint_speed
robot_movement.Robot, 60
MIN_ANGLE_LINK_1_5
robot_movement, 20
MIN_ANGLE_LINK_6

robot_movement, 20
mode
gui.GUI, 30
move_joint
robot_movement.MoveJ, 52
MOVE_NOT_POSSIBLE_ERROR_MSG
robot_movement, 20
move_time
robot_movement.MoveL, 57
mySqrt
test, 21

NEW_TEST
commands, 10

orientation_degree_to_radians
robot_movement, 17
oval_id
gui.Light, 34

patch_test_config
main.Main, 39
PATCH_TEST_CORNER_COORDINATES
main, 13
performing_command
main, 13
place_allergen
main.Main, 35
pos_w_time
robot_movement.MoveJ, 53
robot_movement.MoveL, 57
pos_w_time_linear
robot_movement.MoveL, 57
PREPARE_PRESSING
commands, 10
PRESS_ALLERGEN
commands, 10
pressed_buttons
gui.GUI, 30
prev_position
robot_movement.Robot, 60

radians_to_degree
robot_movement, 18
radians_to_degree_list
robot_movement, 18
rdk_link
robot_movement.Robot, 60
robot_movement.RobotMyCobotAndSim, 63
RESPONSE
commands, 10
robo_link
robot_movement.RobotMyCobot, 61
robot_movement.RobotMyCobotAndSim, 63
robot
main.Main, 39
robot_movement.MoveJ, 53
robot_movement.MoveL, 57
robot_cartesian_position
robot_movement, 20
robot_controller, 14
robot_controller.py, 73
robot_controller.RobotControllerSettings, 39
__init__, 40
dh_values, 40
linear_frequency, 40
update_frequency, 40
robot_controller_settings
robot_movement.Kinematics, 50
robot_joint_position
robot_movement, 20
robot_movement, 14
best_end_joint, 15
COORDINATE_SYNTAX_ERROR_MSG, 19
get_rot_x_matrix, 15
get_rot_y_matrix, 16
get_rot_z_matrix, 16
INVALID_TIME_INCREASE_ERROR_MSG, 19
joint_matrix, 17
JOINT_SYNTAX_ERROR_MSG, 19
MAX_ANGLE_LINK_1_5, 19
MAX_ANGLE_LINK_6, 20
MIN_ANGLE_LINK_1_5, 20
MIN_ANGLE_LINK_6, 20
MOVE_NOT_POSSIBLE_ERROR_MSG, 20
orientation_degree_to_radians, 17
radians_to_degree, 18
radians_to_degree_list, 18
robot_cartesian_position, 20
robot_joint_position, 20
TAKES_ONLY_TWO_COORDINATES_ERROR_MSG, 20
transform_list_into_range, 18
robot_movement.CoordinateSyntaxError, 41
robot_movement.InvalidTimeIncrease, 42
suppress, 43
robot_movement.JointSyntaxError, 43
robot_movement.Kinematics, 44
__init__, 45
dh_value_a, 49
dh_value_alpha, 49
dh_value_d, 49
dh_value_theta, 49
get_joint_values, 45
kin_calculate_theta_1, 46
kin_calculate_theta_2_3, 47
kin_calculate_theta_4, 47
kin_calculate_theta_5, 48
kin_calculate_theta_6, 48
robot_controller_settings, 50
robot_movement.MoveJ, 50
__init__, 51
convert_coordinates_to_joint_movements, 52
joint_matrix, 53
move_joint, 52
pos_w_time, 53
robot, 53

start_joint_values, 54
robot_movement.MoveL, 54
__init__, 56
get_linear_via_points, 56
move_time, 57
pos_w_time, 57
pos_w_time_linear, 57
robot, 57
start_pos_cartesian, 57
robot_movement.MoveNotPossible, 58
robot_movement.py, 74
robot_movement.Robot, 58
__init__, 59
max_joint_speed, 60
prev_position, 60
rdk_link, 60
robot_settings, 60
set_joints, 59
time_between_move, 60
robot_movement.RobotMyCobot, 60
__init__, 61
robo_link, 61
set_joints, 61
robot_movement.RobotMyCobotAndSim, 62
__init__, 62
rdk_link, 63
robo_link, 63
set_joints, 62
robot_movement.TakesOnlyTwoCoordinates, 63
robot_pressed
gui.GUI, 27
robot_settings
robot_movement.Robot, 60
root
gui.GUI, 31
run
gui.GUI, 27
main.Main, 36
running
gui.GUI, 31

send_command
communication.CommunicationProtocol, 24
SERIAL_COMMUNICATION_UNKNOWN_RESPONSE_ERROR_MSG
communication, 11
set_joints
robot_movement.Robot, 59
robot_movement.RobotMyCobot, 61
robot_movement.RobotMyCobotAndSim, 62
set_mode
main.Main, 37
SIMULATING
main, 13
simulation_pressed
gui.GUI, 27
SPEED
main, 13
start_joint_values
robot_movement.MoveJ, 54
start_pos_cartesian
robot_movement.MoveL, 57
start_pressed
gui.GUI, 28
stop_pressed
gui.GUI, 28
stop_pressed_init
gui.GUI, 28
STORAGE_OFFSET_X
main, 13
suppress
robot_movement.InvalidTimeIncrease, 43
SYRINGE_COORDINATES
main, 13
SYRINGE_MOVEMENT_Z
main, 14
SYRINGE_OFFSET_X
main, 14

take_allergen
main.Main, 37
TAKES_ONLY_TWO_COORDINATES_ERROR_MSG
robot_movement, 20
test, 21
gui, 11
mySqrt, 21
testVal, 22
test.py, 84
testVal
test, 22
time_between_move
robot_movement.Robot, 60
timeout
communication.CommunicationProtocol, 24
title
gui.GUI, 31
transform_list_into_range
robot_movement, 18
turn_off
gui.Light, 33
turn_on
gui.Light, 33
turn_ready
gui.Light, 33

update_frequency
robot_controller.RobotControllerSettings, 40

Z_OFFSET
main, 14