Exp_Lab_Assignment_3

Generated by Doxygen 1.8.11

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 camera_manager.camera_manager_fnct Class Reference

Class to manage the camera.

Public Member Functions

- def __init__ (self)

  Init function.
- def callback (self, ros_data)

  Callback function for the camera image.

Public Attributes

- **image_pub**
- **cam_pub**
- **to_send**
- **subscriber**

4.1.1 Detailed Description

Class to manage the camera.

Definition at line 85 of file camera_manager.py.

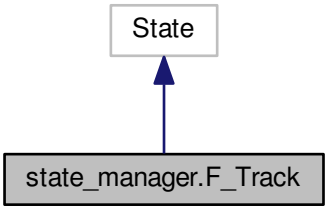
The documentation for this class was generated from the following file:

- scripts/camera_manager.py

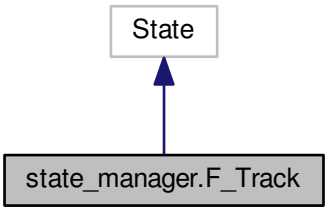
4.2 state_manager.F_Track Class Reference

Find track state of the smach machine.

Inheritance diagram for state_manager.F_Track:



Collaboration diagram for state_manager.F_Track:



Public Member Functions

- def __init__ (self)

  Init function for smach machine normal state.
- def execute (self, userdata)

  Execute function of the state:

4.2.1 Detailed Description

Find track state of the smach machine.

Definition at line 467 of file state_manager.py.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 def state_manager.F_Track.__init__ (*self*)

Init function for smach machine normal state.

Definition at line 470 of file state_manager.py.

```
470     def __init__(self):
471
472         smach.State.__init__(self,
473                              outcomes=['find_command', 'play_command'])
474
```

4.2.3 Member Function Documentation

4.2.3.1 def state_manager.F_Track.execute (*self, userdata*)

Execute function of the state:

- Go close to the ball: is it the desired position? (no need to check if saved: u enter here only if it's not)
  - Yes: exit PLAY
  - No: exit FIND

Definition at line 480 of file state_manager.py.

```
480     def execute(self, userdata):
481         global moveTo, vel_pub, closeBall, justDetected, position_, ballsPos, desiredRoom
482
483         rospy.logerr('f_track')
484
485         # Move closer to the ball
486         while closeBall == -1:
487             vel_to_ball.angular.z = -0.002*(circCenter-400)
488             vel_to_ball.linear.x = -0.01*(radius-100)
489             vel_pub.publish(vel_to_ball)
490
491         # Stop dog
492         rospy.logerr('Stopping in front of ball')
493         for i in range(0, 3):
494             vel_to_ball.linear.x = 0
495             vel_pub.publish(vel_to_ball)
496
497         # If the ball is the one the human asked for, go back to play...
498         if colorName[justDetected] == colorName[desiredRoom]:
499             rospy.logerr('I found the right ball')
500
501             # If not saved yet, save ball's position
502             if ballsPos[justDetected] == [0, 0]:
503                 ballsPos[justDetected] = [position_.x, position_.y]
504                 rospy.logerr(
505                     'Saved position of the %s ball, will not forget it!', colors[justDetected])
506
507             return 'play_command'
508
509         # ... if it is not, save its position anyway
510         else:
511             rospy.logerr(
512                 'I found the wrong ball')
513
514             # If not saved yet, save ball's position
515             if ballsPos[justDetected] == [0, 0]:
516                 ballsPos[justDetected] = [position_.x, position_.y]
517                 rospy.logerr(
518                     'Saved position of the %s ball anyway, may turn out useful', colors[justDetected])
519
520             return 'find_command'
521
```

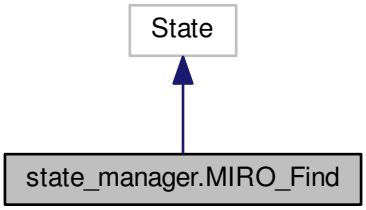
The documentation for this class was generated from the following file:

- scripts/state_manager.py

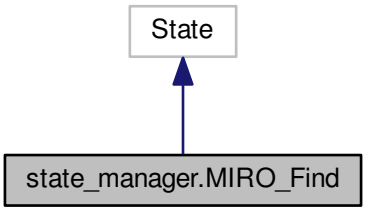
4.3 state_manager.MIRO_Find Class Reference

Find state of the smach machine.

Inheritance diagram for state_manager.MIRO_Find:



Collaboration diagram for state_manager.MIRO_Find:



Public Member Functions

- def __init__ (self)

  Init function for smach machine normal state.
- def execute (self, userdata)

  Execute function of the state:

4.3.1 Detailed Description

Find state of the smach machine.

Definition at line 421 of file state_manager.py.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 def state_manager.MIRO_Find.__init__ (*self*)

Init function for smach machine normal state.

Definition at line 424 of file state_manager.py.

```
424     def __init__(self):
425
426         smach.State.__init__(self,
427                              outcomes=['play_command', 'f_track_command'])
428
```

4.3.3 Member Function Documentation

4.3.3.1 def state_manager.MIRO_Find.execute (*self, userdata*)

Execute function of the state:

- In a loop:
- Move towards goal (may change it): ball?
  - Yes: exit F_TRACK
  - No: continue
- End of the loop: exit PLAY

Definition at line 436 of file state_manager.py.

```
436     def execute(self, userdata):
437
438         global justDetected
439
440         rospy.logerr('find')
441
442         # Launch explore for autonomous exploration
443         command2 = ["roslaunch", "explore_lite", "explore.launch"]
444         p = subprocess.Popen(command2)
445
446         for loops in range(0, 10):
447             time.sleep(1)
448
449             for loops2 in range(0, LOOPS_EXPLORE):
450                 time.sleep(3)
451
452                 # Check if ball is detected
453                 if justDetected != -1:
454                     rospy.logerr('I found something!')
455                     p.terminate()
456                     time.sleep(7)
457
458                     return 'f_track_command'
459
460         p.terminate()
461         time.sleep(7)
462
463         return 'play_command'
464
```

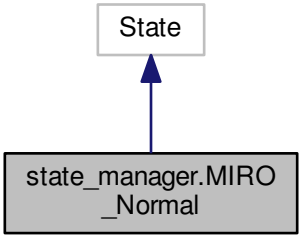
The documentation for this class was generated from the following file:

- scripts/state_manager.py

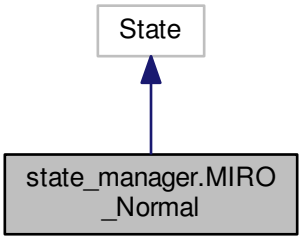
4.4 state_manager.MIRO_Normal Class Reference

Normal state of the smach machine.

Inheritance diagram for state_manager.MIRO_Normal:



Collaboration diagram for state_manager.MIRO_Normal:



Public Member Functions

- def __init__ (self)

  Init function for smach machine normal state.
- def execute (self, userdata)

  Execute function of the state:

4.4.1 Detailed Description

Normal state of the smach machine.

Definition at line 238 of file state_manager.py.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 def state_manager.MIRO_Normal.__init__ (*self*)

Init function for smach machine normal state.

Definition at line 241 of file state_manager.py.

```
241     def __init__(self):
242
243         smach.State.__init__(self,
244                              outcomes=['sleep_command', 'play_command', 'n_track_command'])
245
```

4.4.3 Member Function Documentation

4.4.3.1 def state_manager.MIRO_Normal.execute (*self, userdata*)

Execute function of the state:

- Move around: ball?
  - Yes: exit N_TRACK
  - No: Continue
- Listen to human: play command?
  - Yes: exit PLAY
  - No: Continue
- End of the loop: exit SLEEP

Definition at line 255 of file state_manager.py.

```
255     def execute(self, userdata):
256         global justDetected, lastDetected
257
258         rospy.logerr('normal')
259         time.sleep(2)
260
261         # Launch explore for autonomous exploration
262         command = ["roslaunch", "explore_lite", "explore.launch"]
263         p = subprocess.Popen(command)
264
265         for loops in range(0, 10):
266             time.sleep(1)
267
268             for loops2 in range(0, LOOPS_EXPLORE):
269                 time.sleep(3)
270
271                 # Check if ball is detected
272                 if justDetected != -1:
273                     rospy.logerr('a ball was detected')
274
275                     # Check if it's a new ball or the same as before
276                     if justDetected != lastDetected:
277                         lastDetected = justDetected
278                         rospy.logerr('and it is %s', colors[justDetected])
279
280                         # Go to track state
281                         p.terminate()
282                         time.sleep(7)
283
284                         return 'n_track_command'
285
```

```
286                     else:
287                         rospy.logerr('but i already knew it')
288
289             # Listen to user
290             if user_says(0) == 'play':
291                 rospy.logerr('user said play')
292
293                 p.terminate()
294                 time.sleep(7)
295                 return 'play_command'
296
297         p.terminate()
298         time.sleep(7)
299         return 'sleep_command'
300
```

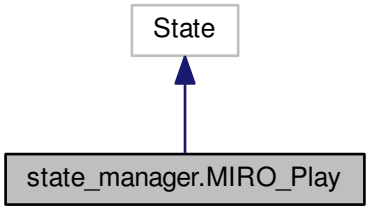
The documentation for this class was generated from the following file:

- scripts/state_manager.py

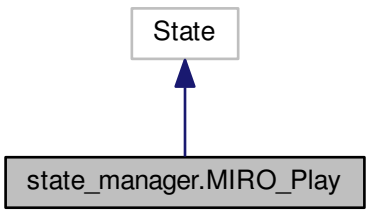
4.5 state_manager.MIRO_Play Class Reference

Play state of the smach machine.

Inheritance diagram for state_manager.MIRO_Play:



Collaboration diagram for state_manager.MIRO_Play:



Public Member Functions

- def __init__ (self)

  Init function for smach machine play state.
- def execute (self, userdata)

  Execute function of the state:

4.5.1 Detailed Description

Play state of the smach machine.

Definition at line 350 of file state_manager.py.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 def state_manager.MIRO_Play.__init__ (*self*)

Init function for smach machine play state.

Definition at line 353 of file state_manager.py.

```
353     def __init__(self):
354
355         smach.State.__init__(self,
356                              outcomes=['normal_command', 'find_command'])
357
```

4.5.3 Member Function Documentation

4.5.3.1 def state_manager.MIRO_Play.execute (*self, userdata*)

Execute function of the state:

- In a loop:
- Go to human
- Wait for a goto command
- Compare command to the known ball positions: is position known?
  - Yes: go to position
  - No: exit FIND
- Wait to be arrived
- End of the loop: exit NORMAL

Definition at line 368 of file state_manager.py.

```
368     def execute(self, userdata):
369         global moveTo, ballsPos, colorName, justDetected
370 
371         rospy.logerr('play')
372         lastDetected = -2
373 
374         if ballsPos[0] != [0, 0] and ballsPos[1] != [0, 0] and ballsPos[2] != [0, 0] and ballsPos[3] != [0,
    0] and ballsPos[4] != [0, 0] and ballsPos[5] != [0, 0]:
375             rospy.logerr('ALL BALLS HAVE BEEN DETECTED! Positions:')
376             rospy.logerr(ballsPos)
377 
378         for loops in range(0, 10):
379 
380             # Move to the human, unless dog is already near him
381             rospy.logerr('moving to human')
382             if (position_.x > -7 or position_.x < -3) and (position_.y > 10 or position_.y < 6):
383                 #move_dog([-5, 8, 0])
384                 time.sleep(1)
385 
386                 # Listen to human
387             rospy.logerr('listen to user')
388             user_command = user_says(1)
389 
390             # Save the desired room
391             if 'go' in user_command and 'to' in user_command:
392                 for i in range(0, 6):
393                     if colorName[i] in user_command:
394                         desiredRoom = i
395                         break
396 
397             else:
398                 rospy.logerr('Wrong command received')
399                 return 'normal_command'
400 
401             rospy.logerr('user said to go to %s which is color %s',
402                          colorName[desiredRoom], colors[desiredRoom])
403 
404             # If position of room (i.e. ball) is known, go there...
405             if ballsPos[desiredRoom] != [0, 0]:
406                 rospy.logerr(
407                     'i know this ball! i will go to it right away! it is in')
408                 moveTo = np.append(np.asarray(ballsPos[desiredRoom]), 0)
409                 move_dog(moveTo)
410 
411             # ... if not: look for it
412             else:
413                 rospy.logerr('i do not know this position, i will look for it')
414                 return 'find_command'
415 
416         return 'normal_command'
417 
418 
```

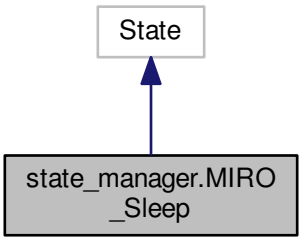
The documentation for this class was generated from the following file:

- scripts/state_manager.py

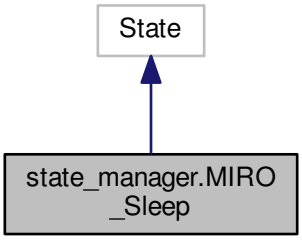
4.6 state_manager.MIRO_Sleep Class Reference

Sleep state of the smach machine.

Inheritance diagram for state_manager.MIRO_Sleep:



Collaboration diagram for state_manager.MIRO_Sleep:



Public Member Functions

- def __init__ (self)

  Init function for smach machine sleep state.
- def execute (self, userdata)

  Execute function of the state:

4.6.1 Detailed Description

Sleep state of the smach machine.

Definition at line 207 of file state_manager.py.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 def state_manager.MIRO_Sleep.__init__ (*self*)

Init function for smach machine sleep state.

Definition at line 210 of file state_manager.py.

```
210     def __init__(self):
211
212         smach.State.__init__(self,
213                              outcomes=['normal_command'])
214
```

4.6.3 Member Function Documentation

4.6.3.1 def state_manager.MIRO_Sleep.execute (*self, userdata*)

Execute function of the state:

- Go to kennel
- Stay still for 5 seconds
- Exit NORMAL

Definition at line 220 of file state_manager.py.

```
220     def execute(self, userdata):
221         global lastDetected
222         rospy.logerr('sleep')
223
224         # Go to kennel
225         move_dog([-5, 6, 0])
226         time.sleep(1)
227
228         # Reset previously seen ball variable
229         lastDetected = -2
230
231         rospy.logerr('exit sleep')
232         return 'normal_command'
233
234
```

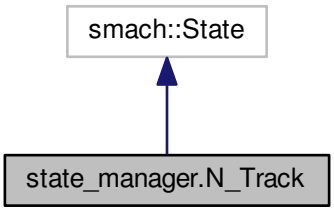
The documentation for this class was generated from the following file:

- scripts/state_manager.py

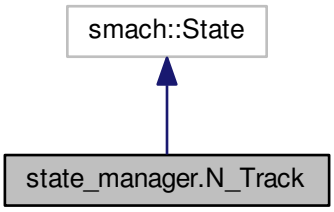
4.7 state_manager.N_Track Class Reference

Normal track state of the smach machine.

Inheritance diagram for state_manager.N_Track:



Collaboration diagram for state_manager.N_Track:



Public Member Functions

- def __init__ (self)

  Init function for smach machine normal state.
- def execute (self, userdata)

  Execute function of the state:

4.7.1 Detailed Description

Normal track state of the smach machine.

Definition at line 303 of file state_manager.py.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 def state_manager.N_Track.__init__ (*self*)

Init function for smach machine normal state.

Definition at line 306 of file state_manager.py.

```
306     def __init__(self):
307
308         smach.State.__init__(self,
309                              outcomes=['normal_command'])
310
```

4.7.3 Member Function Documentation

4.7.3.1 def state_manager.N_Track.execute (*self, userdata*)

Execute function of the state:

- Go close to the ball: did you know its position yet?
  - Yes: continue
  - No: save it
- Exit NORMAL

Definition at line 317 of file state_manager.py.

```
317     def execute(self, userdata):
318         global vel_pub, closeBall, colorName, circCenter
319
320         rospy.logerr('N_track')
321         time.sleep(2)
322
323         # Move closer to the ball
324         rospy.logerr('Moving closer to ball')
325         while closeBall == -1:
326             vel_to_ball.angular.z = -0.002*(circCenter-400)
327             vel_to_ball.linear.x = -0.01*(radius-100)
328             vel_pub.publish(vel_to_ball)
329
330         # Stop dog
331         rospy.logerr('Stopping in front of ball')
332         for i in range(0, 3):
333             vel_to_ball.linear.x = 0
334             vel_pub.publish(vel_to_ball)
335         time.sleep(3)
336
337         # If not saved yet, save ball's position
338         if ballsPos[justDetected] == [0, 0]:
339             ballsPos[justDetected] = [position_.x, position_.y]
340
341             rospy.logerr('Saved position of %s ball as %d, %d approximately',
342                          colors[justDetected], position_.x, position_.y)
343
344         return 'normal_command'
345
346
```

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

- scripts/state_manager.py

Chapter 5

File Documentation

5.1 scripts/camera_manager.py File Reference

Manages the camera, then sends relevant information (related to the ball) of the environment.

Classes

- class camera_manager.camera_manager_fnct

  Class to manage the camera.

Functions

- def camera_manager.camera_manager ()

  Ros node that subscribes to the target_pos and odom topic and publishes on the cmd_vel topic.

Variables

- tuple **camera_manager.blackLower** = (0, 0, 0)
- tuple **camera_manager.blackUpper** = (5, 50, 50)
- tuple **camera_manager.redLower** = (0, 50, 50)
- tuple **camera_manager.redUpper** = (5, 255, 255)
- tuple **camera_manager.yellowLower** = (25, 50, 50)
- tuple **camera_manager.yellowUpper** = (35, 255, 255)
- tuple **camera_manager.greenLower** = (50, 50, 50)
- tuple **camera_manager.greenUpper** = (70, 255, 255)
- tuple **camera_manager.blueLower** = (100, 50, 50)
- tuple **camera_manager.blueUpper** = (130, 255, 255)
- tuple **camera_manager.magentaLower** = (125, 50, 50)
- tuple **camera_manager.magentaUpper** = (150, 255, 255)
- list **camera_manager.lowerValues**
- list **camera_manager.upperValues**
- list **camera_manager.colorName**
- list **camera_manager.colors** = ['black', 'red', 'yellow', 'green', 'blue', 'magenta']
- list **camera_manager.blackPos** = [0, 0]
- list **camera_manager.redPos** = [0, 0]

- list **camera_manager.yellowPos** = [0, 0]
- list **camera_manager.greenPos** = [0, 0]
- list **camera_manager.bluePos** = [0, 0]
- list **camera_manager.magentaPos** = [0, 0]
- list **camera_manager.ballsPos** = [blackPos, redPos, yellowPos, greenPos, bluePos, magentaPos]
- int **camera_manager.justDetected** = -1
- int **camera_manager.closeBall** = -1
- int **camera_manager.desiredRoom** = -1
- **camera_manager.position_** = Point()
- **camera_manager.pose_** = Pose()
- int **camera_manager.yaw_** = 0
- int **camera_manager.yaw_precision_2_** = 1
- list **camera_manager.moveTo** = [0, 0, 0]
- int **camera_manager.radius** = 0
- int **camera_manager.circCenter** = 0

5.1.1 Detailed Description

Manages the camera, then sends relevant information (related to the ball) of the environment.

5.1.2 Function Documentation

5.1.2.1 def camera_manager.camera_manager ( )

Ros node that subscribes to the target_pos and odom topic and publishes on the cmd_vel topic.

Definition at line 163 of file camera_manager.py.

```
163 def camera_manager():
164
165     rospy.init_node('camera_manager', anonymous=True)
166
167     camera_manager_fnct()
168
169     rospy.spin()
170
171     pass
172
173
```

5.1.3 Variable Documentation

5.1.3.1 list camera_manager.colorName

Initial value:

```
1 = ['entrance', 'closet', 'kitchen',
2             'livingRoom', 'bedroom', 'bathroom']
```

Definition at line 52 of file camera_manager.py.

5.1.3.2 list camera_manager.lowerValues

Initial value:

```
1 = [blackLower, redLower, yellowLower,
2                greenLower, blueLower, magentaLower]
```

Definition at line 48 of file camera_manager.py.

5.1.3.3 list camera_manager.upperValues

Initial value:

```
1 = [blackUpper, redUpper, yellowUpper,
2                greenUpper, blueUpper, magentaUpper]
```

Definition at line 50 of file camera_manager.py.

5.2 scripts/state_manager.py File Reference

Manges the dog behaviour states and the human commands.

Classes

- class state_manager.MIRO_Sleep
  Sleep state of the smach machine.
- class state_manager.MIRO_Normal
  Normal state of the smach machine.
- class state_manager.N_Track
  Normal track state of the smach machine.
- class state_manager.MIRO_Play
  Play state of the smach machine.
- class state_manager.MIRO_Find
  Find state of the smach machine.
- class state_manager.F_Track
  Find track state of the smach machine.

Functions

- def state_manager.clbk_cam (msg)
  Callback function for the camera information.
- def state_manager.user_says (stateCalling)
  Function to simulate the human speaking.
- def state_manager.clbk_odom (msg)
  Callback function for the odometry of the robot.
- def state_manager.normalize_angle (angle)
  Function to normalize the yaw angle.
- def state_manager.move_dog (target)
  Function to move the dog to a target position.
- def state_manager.main ()
  Ros node that subscribes to camera_info and odometry.

Variables

- list **state_manager.colorName**
- list **state_manager.colors** = ['black', 'red', 'yellow', 'green', 'blue', 'magenta']
- list **state_manager.blackPos** = [0, 0]
- list **state_manager.redPos** = [0, 0]
- list **state_manager.yellowPos** = [0, 0]
- list **state_manager.greenPos** = [0, 0]
- list **state_manager.bluePos** = [0, 0]
- list **state_manager.magentaPos** = [0, 0]
- list **state_manager.ballsPos** = [blackPos, redPos, yellowPos, greenPos, bluePos, magentaPos]
- int **state_manager.justDetected** = -1
- int **state_manager.closeBall** = -1
- int **state_manager.desiredRoom** = -1
- int **state_manager.lastDetected** = -2
- int **state_manager.LOOPS_EXPLORE** = 20
- **state_manager.position_** = Point()
- **state_manager.pose_** = Pose()
- int **state_manager.yaw_** = 0
- float **state_manager.yaw_precision_2_** = 0.5
- list **state_manager.moveTo** = [0, 0, 0]
- **state_manager.vel_pub** = rospy.Publisher("/cmd_vel", Twist, queue_size=1)
- **state_manager.vel_to_ball** = Twist()
- **state_manager.x**
- **state_manager.y**
- **state_manager.z**
- int **state_manager.radius** = 0
- int **state_manager.circCenter** = 0

5.2.1 Detailed Description

Manges the dog behaviour states and the human commands.

5.2.2 Function Documentation

5.2.2.1 def state_manager.move_dog (*target*)

Function to move the dog to a target position.

It first sets the right yaw angle by directly publishing on the /cmd_vel topic then implements a client to the MoveBase service to move the robot to the target position while avoiding obstacles.

Definition at line 147 of file state_manager.py.

```
147 def move_dog(target):
148     global yaw_, pub, yaw_precision_2_, vel_pub, position_
149 
150     kp_a = 3.0
151     ub_a = 0.6
152     lb_a = -0.5
153 
154     rospy.logerr('move dog started with data %d %d', target[0], target[1])
155 
156     # Adjust yaw angle
157     while True:
158         desired_yaw = math.atan2(
159             target[1] - position_.y, target[0] - position_.x)
160         err_yaw = normalize_angle(desired_yaw - yaw_)
161 
162         twist_msg = Twist()
163 
164         if math.fabs(err_yaw) > yaw_precision_2_:
165             twist_msg.angular.z = kp_a*err_yaw
166             if twist_msg.angular.z > ub_a:
167                 twist_msg.angular.z = ub_a
168             elif twist_msg.angular.z < lb_a:
169                 twist_msg.angular.z = lb_a
170 
171         # Publish velocity directly on cmd_vel topic
172         vel_pub.publish(twist_msg)
173 
174         if math.fabs(err_yaw) <= yaw_precision_2_:
175             break
176 
177     # Move to target position
178     rospy.logerr('now set target')
179 
180     # Call MoveBase service
181     client = actionlib.SimpleActionClient('move_base', MoveBaseAction)
182     client.cancel_goal
183     client.wait_for_server()
184 
185     goal = MoveBaseGoal()
186     goal.target_pose.header.frame_id = "map"
187     goal.target_pose.header.stamp = rospy.Time.now()
188     goal.target_pose.pose.position.x = target[0]
189     goal.target_pose.pose.position.y = target[1]
190     goal.target_pose.pose.position.z = 0
191     goal.target_pose.pose.orientation.x = 0
192     goal.target_pose.pose.orientation.y = 0
193     goal.target_pose.pose.orientation.z = 0
194     goal.target_pose.pose.orientation.w = 1.0
195 
196     client.send_goal(goal)
197 
198     while math.fabs(position_.x - target[0]) > 0.3 or math.fabs(position_.y - target[1]) > 0.3:
199         time.sleep(1)
200     rospy.logerr('arrived')
201     client.cancel_all_goals()
202     return 1
203 
204 
```

5.2.3 Variable Documentation

5.2.3.1 list state_manager.colorName

Initial value:

```
1 = ['entrance', 'closet', 'kitchen',
2              'livingRoom', 'bedroom', 'bathroom']
```

Definition at line 36 of file state_manager.py.

Index