

3806ICT - Week 3 Lab

Nick van der Merwe - s5151332 - nick.vandermmerwe@griffithuni.edu.au

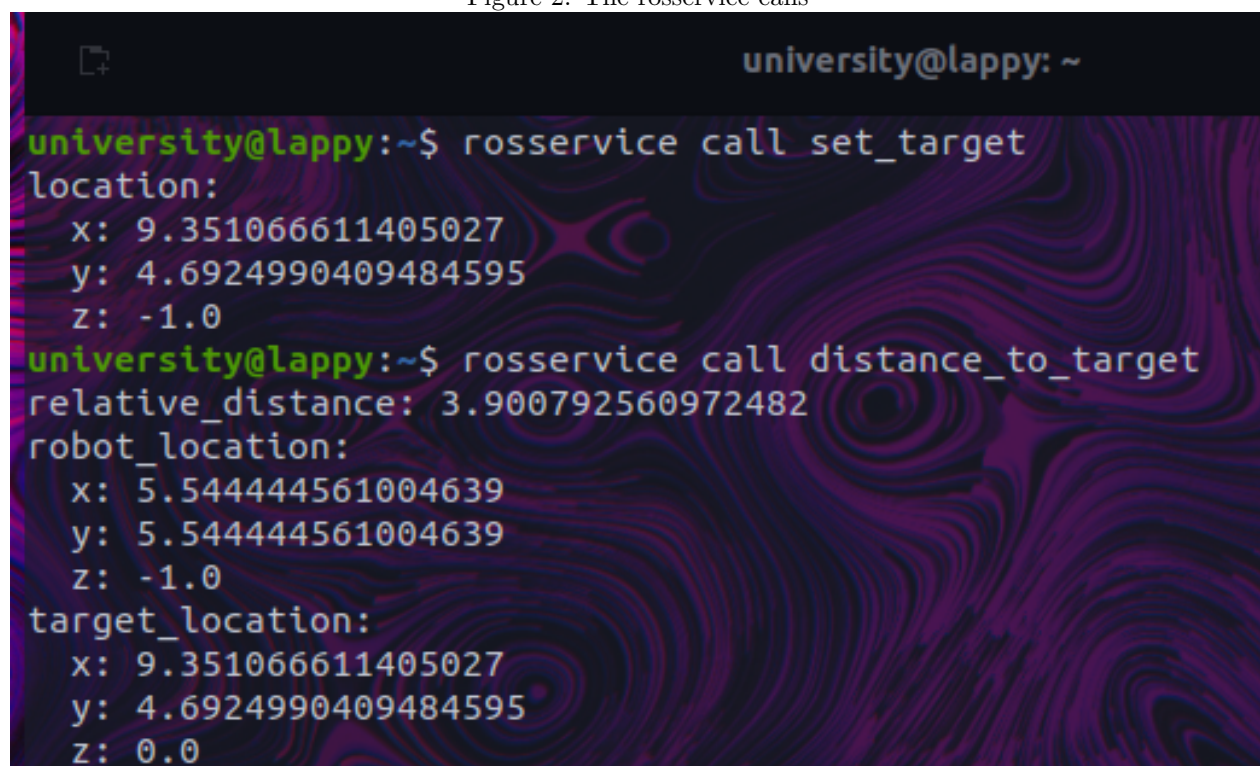
March 25, 2021

1 Task 1 - Programming Exercise

Figure 1: The services file

```
1 services.cpp
2 #include "ros/ros.h"
3 #include "week3/set_target.h"
4 #include "week3/distance_to_target.h"
5 #include "geometry_msgs/Vector3.h"
6 #include "turtlesim/Pose.h"
7 #include <cmath>
8 #include <iostream>
9 geometry_msgs::Vector3 Vector3Ctor(double x, double y, double z){
10     // For some reason the vector3 ctor is broken so here's a custom one
11     geometry_msgs::Vector3 toReturn;
12     toReturn.x = x;
13     toReturn.y = y;
14     toReturn.z = z;
15     return toReturn;
16 }
17
18 geometry_msgs::Vector3 target;
19
20 bool setTargetFunction(week3::set_target::Request & req,
21     week3::set_target::Response & res){
22     // Set the target to always stay the same
23     // World runs from 0-11.0 in both x and y
24     target.x = static_cast<double>(rand()) / static_cast<double>(RAND_MAX) * 11;
25     target.y = static_cast<double>(rand()) / static_cast<double>(RAND_MAX) * 11;
26     res.location = Vector3Ctor(target.x, target.y, -1);
27
28     ROS_INFO("Set target to %f %f", res.location.x, res.location.y);
29     return true;
30 }
31
32
33
34 double euclideanDistance(geometry_msgs::Vector3 & point,
35     turtlesim::Pose & pose){
36     return sqrt(
37         pow(point.x - pose.x, 2) +
38         pow(point.y - pose.y, 2)
39     );
40 }
41
42 turtlesim::Pose turtlePose;
43
44 void setTurtlePose(const turtlesim::Pose & msg){
45     turtlePose = msg;
46 }
47
48 void updateTurtlePose(){
49     ros::NodeHandle nodeHandle;
50     ros::Subscriber turtlePoseSubscriber =
51         nodeHandle.subscribe("turtle1/pose", 50, setTurtlePose);
52     ros::spinOnce();
53 }
54
55 bool distanceToTargetFunction(week3::distance_to_target::Request & req,
56     week3::distance_to_target::Response & res){
57     res.relative_distance = euclideanDistance(target, turtlePose);
58     res.robot_location = Vector3Ctor(turtlePose.x, turtlePose.y, -1);
59     res.target_location = target;
60     return true;
61 }
62
63
64 int main(int argc, char **argv){
65     srand(time(NULL));
66     ros::init(argc, argv, "services_server");
67     ros::NodeHandle nodeHandle;
68
69     ros::ServiceServer setTargetService =
70         nodeHandle.advertiseService("set_target", setTargetFunction);
71     ros::ServiceServer setDistanceToTargetService =
72         nodeHandle.advertiseService("distance_to_target", distanceToTargetFunction);
73     ros::Subscriber turtlePoseSubscriber =
74         nodeHandle.subscribe("turtle1/pose", 50, setTurtlePose);
75     ROS_INFO("Ready to manage erequests");
76
77     ros::spin();
78
79     return 0;
80 }
```

Figure 2: The rosservice calls

A terminal window with a dark background and a purple wavy pattern. The prompt is 'university@lappy: ~'. The first command is 'rosservice call set_target' followed by 'location:' and coordinates 'x: 9.351066611405027', 'y: 4.6924990409484595', and 'z: -1.0'. The second command is 'rosservice call distance_to_target' followed by 'relative_distance: 3.900792560972482', 'robot_location:' with the same coordinates as above, and 'target_location:' with the same coordinates as above.

```
university@lappy: ~  
university@lappy:~$ rosservice call set_target  
location:  
  x: 9.351066611405027  
  y: 4.6924990409484595  
  z: -1.0  
university@lappy:~$ rosservice call distance_to_target  
relative_distance: 3.900792560972482  
robot_location:  
  x: 5.544444561004639  
  y: 5.544444561004639  
  z: -1.0  
target_location:  
  x: 9.351066611405027  
  y: 4.6924990409484595  
  z: 0.0
```