3806ICT - Week 2 Lab

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

March 25, 2021

1 Task 1 - Explain ROS

1.1 What is a ros node?

A ros node is an executable in a ros package which communicates through topics to one another.

1.2 What is roscore?

Roscore essentially hosts all the nodes that are running on ros. For any nodes to communicate, ros must be running.

1.3 What is the difference between publisher/subscriber and a service?

A publisher sends information to a topic, a subscriber reads information from a topic and a service is a node that can be requested to reply with information.

1.4 What is the purpose of catkin?

Catkin is a build system that manages workspaces, and within those, packages, and compiling them.

1.5 What does the roscd command do?

Like the cd command in linux, it navigates to the given package or stack in the ros system.

1.6 Show the command for creating a package called lab2 which relies on roscpp, rospy, std_msgs and message_generation

I've already ran it, but its catkin_create_pkg lab2 roscpp rospy std_msgs message_generation

1.7 What do I have to change in my package.xml file to enable message generation?

Its required to uncomment the <build_depend >message_generation</build_depend>line and the <exec_depend >message_runtime</exec_depend>

1.8 What is the difference between a msg and a srv file?

A msg file, or message file, is used to define the format of the messages being communicated whereas services describe the format of a service. A srv is split into two sections, a request and a reponse.

1.9 Define what these types represent

1.9.1 uint32

An unsigned int made of 32 bits or 8 bytes - the same as uint32_t

1.9.2 string

An array of characters, a std::string

1.9.3 float64

A float made of 64 bits - a double.

1.9.4 float32

A float made of 32 bits - a float.

1.9.5 bool

A boolean value - uint8_t - Quick question, wouldn't it have been better if ros actually used a bool type since std::vectors are optimised to compress them? Do some robots really have a 1 bit bool or something?

1.9.6 int64

An int made of 64 bits - int64_t

1.9.7 int32

An int made of 32 bits - int32_t

2 Task 2 - Programming Exercise

This is explained... oddly. Here's my list of assumptions

- The origin is always the same (5.54445). After reading the turtle sim source code, This seems to always be the case as its based on the resolution of the turtle and we can assume that's constant.
- Zero degrees occurs at the far right - like a unit circle system. The way that quadrants are usually defined
- The edge case of when the turtle is exactly on a border between two is that it belongs to the one more upper right. For instance if its between quadrant 1 and 2 then it belongs to 1.

So here's the result.

Figure 1: The publisher

publisher.cpp

publisher.cpp > ...

```cpp
// Written by Nick van der Merwe for 3806ICT
// s515332 - nick.vandermerwe@griffithuni.edu.au
#include <cstdlib>
#include <sstream>

#include "ros/ros.h"
#include "geometry_msgs/Twist.h"

int main(int argc, char** argv) {
    ros::init(argc, argv, "randomWalker");
    ros::NodeHandle nodeHandle;

    ros::Publisher turtle_pub =
        nodeHandle.advertise<geometry_msgs::Twist>("turtle1/cmd_vel", 50);
    ros::Rate loop_rate(2);
    geometry_msgs::Twist msg;

    while (ros::ok()) {
        msg = geometry_msgs::Twist{};
        uint16_t a = rand();  // rand() runs 0 -> 32k

        switch (a % 4) {
            case 0:
                msg.linear.x = 2.0;
                break;
            case 1:
                msg.linear.x = -2.0;
                break;
            case 2:
                msg.angular.z = 2.0;
                break;
            case 3:
                msg.angular.z = -2.0;
                break;
            default:
                break;
        }

        turtle_pub.publish(msg);

        ros::spinOnce();
        loop_rate.sleep();
    }

    return 0;
}
```

Figure 2: The subscriber

publisher.cpp | listener.cpp | package.xml | dynamic_listener.cpp | CMakeLists.txt

src > listener.cpp > {} std > basic_stringbuf::pbackfail(int_type)

```cpp
// Written by Nick van der Merwe for 3806ICT GUGC
// s5151332 - nick.vandermerwe@griffithuni.edu.au
#include "ros/ros.h"
#include "turtlesim/Pose.h"

constexpr uint8_t XORIGIN = 5.544445;
constexpr uint8_t YORIGIN = 5.544445;

int defineQuadrant(const float& x, const float& y) {
    /// If its on the border between two then its at
    /// the more upper right quadrant. This isn't clearly
    /// defined in the tasksheet. We can also shortcut
    /// the checks by searching like this instead.
    if (x > XORIGIN && y > YORIGIN) {
        return 1;
    }
    if (x > XORIGIN) {
        return 4;
    }

    if (y > YORIGIN) {
        return 2;
    }

    return 3;
}

void quadrantPrinter(const turtlesim::PoseConstPtr msg) {
    ROS_INFO("Its currently at quadrant %i", defineQuadrant(msg->x, msg->y));
}

int main(int argc, char** argv) {
    /*
     * After reading the sourcecode of turtle.sim at
     * https://github.com/ros/ros_tutorials/blob/noetic-devel/turtlesim/src/
     * I realised that regardless of the monitor resolution, the turtle always
     * spawns in the same place. We'll use this as the origin
     */

    ros::init(argc, argv, "quadrantReporter");
    ros::NodeHandle nodeHandle;
    ros::Subscriber sub =
        nodeHandle.subscribe("turtle1/pose", 50, quadrantPrinter);

    ros::spin();

    return 0;
}
```

Figure 3: Everything in action