

COMPSYS 726 Lab manual

Contents

Part 0: Prerequisites.....	2
A. Robots Pioneer P3-DX:	2
B. Charging:	3
C. Laptop:	3
D. The Robot Operating System (ROS)	4
Part 1: Setup.....	7
Setup	7
Clean up.....	8
Part 2: Robot manual control.....	9
Manual movement.....	9
Manual sensor data acquisition	12
Part 3: Simulation – basic.....	15
Part 4: Launch files	20
Part 5: Executing on the robot	22
Part 6: Simulation – SLAM.....	25

COMPSYS 726 Lab manual

Part 0: Prerequisites

A. Robots Pioneer 3-DX:

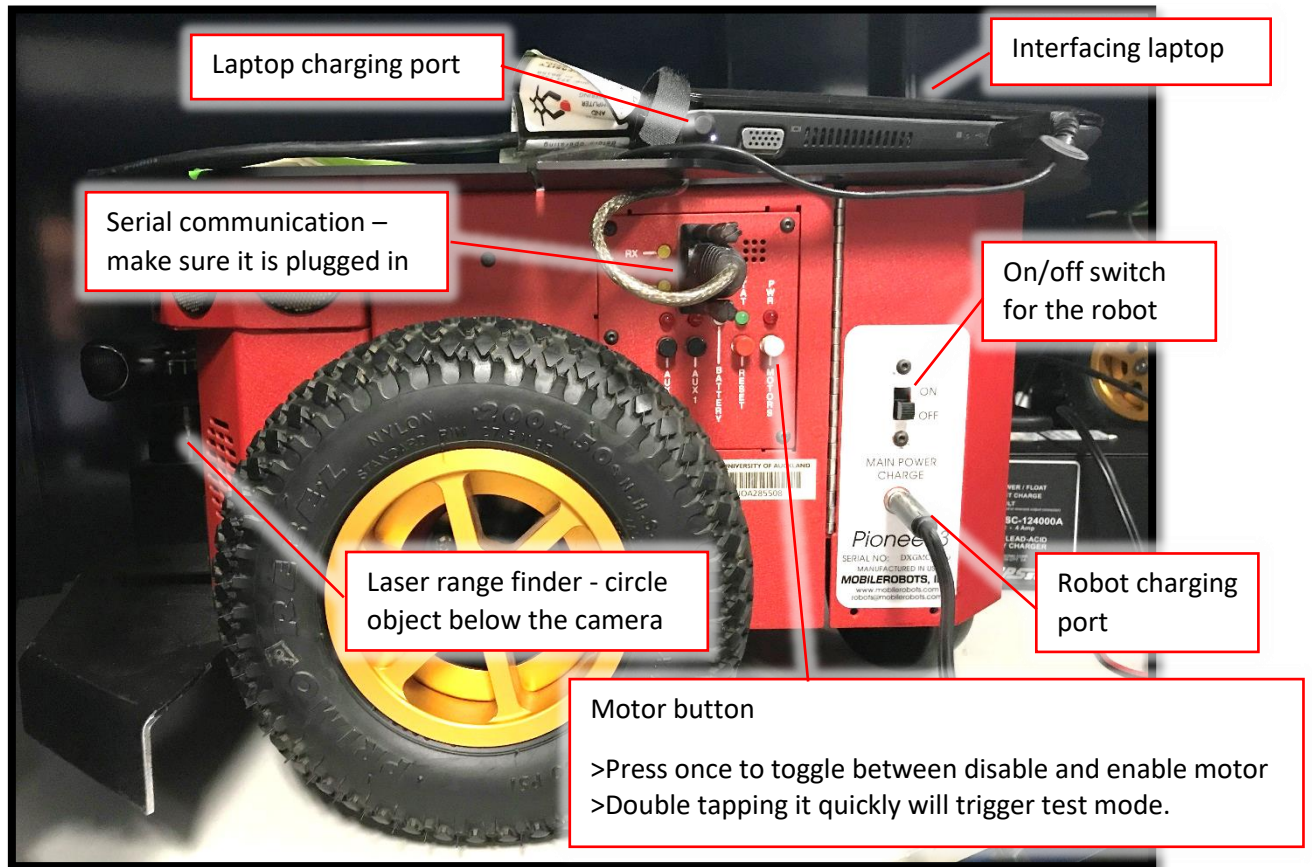


Figure 1: robot side view

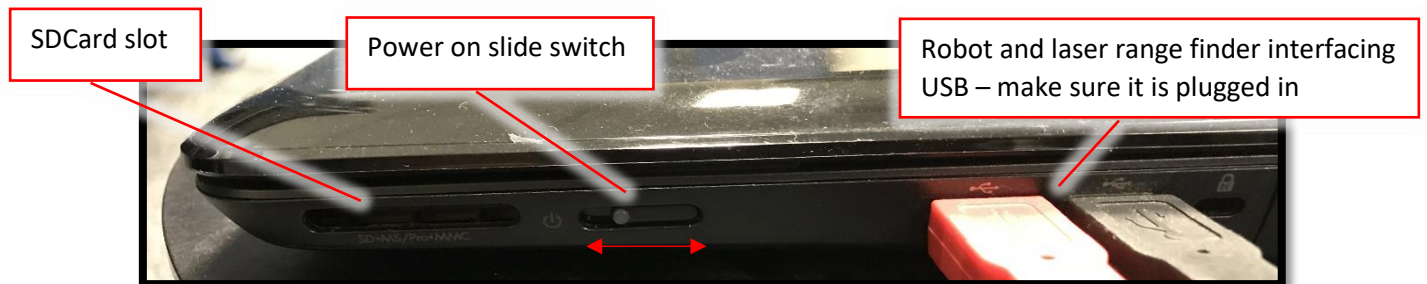


Figure 2: laptop side view

- Please take care when moving the robots as they are very heavy and fragile.
- Leave the robot on the floors and not on desks.
- The robots needs to be completely lifted when moved or it may damage its wheels.
- Motor button (white) to **enable/ disable motors** (see Figure 1: robot side view)
 - Once the robot and laptop are fully booted double tap the motor button to run robot test sequence.
 - See [motor_test.mov](#) in the resources folder or <https://youtu.be/wOOH7Yodkz0>

COMPSYS 726 Lab manual

- The motor speed is set in meters per seconds
 - note that 1.2m is very fast in the lab so please set an appropriate speed
- If the robot starts beeping due to an error, turn it off and on again.
 - See beep_error.mov in the resources folder or <https://youtu.be/3A1UxKMuN5E>
 - After turning it off and on, you will need to restart your ros code (i.e., the nodes, such as, rosaria)
- If the robots are broken please email Fung Yang (fung.yang@auckland.ac.nz) and CC the lecturers. The email should include the following information:
 - The name of the robot (a white sticker on the robot)
 - The name of the laptop (a white sticker on the robot)
 - What is broken on the robot or laptop
 - (if you can put out a label and place it on the broken device)
- **If some part of the lab stopped working, try turning off the robot and then on and restart the ALL ROS nodes.**

B. Charging:

- The robots and laptops have around 4+ hours of runtime but the batteries are old
- **IMPORTANT**: After each lab session return **the robot** and **the laptop** when placing back in the cabinet and make sure that both the robot and laptop charging cable is attached tightly.

C. Laptop:

- The password for sudo is "robot_3c3"
- Pressing "CTRL+C" in a terminal will terminate the current executing program
 - Multiple times may be needed for terminating launch files

COMPSYS 726 Lab manual

D. The Robot Operating System (ROS)

Reference: You will be expected to **utilise** and **read** the online documentation and resources. wiki.ros.org is your best source of detailed information about ROS. Specific pages that may be helpful for COMPSYS 726 include:

1. For general tutorials, in particular, tutorials 1-8 and 11 may help your understanding of ROS.
<http://wiki.ros.org/ROS/Tutorials>
2. For information about how to communicate with Pioneer robots.
<http://wiki.ros.org/ROSARIA>
3. For information on LaserScan message.
http://docs.ros.org/api/sensor_msgs/html/msg/LaserScan.html

ROS is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. ROS is similar in some respects to 'robot frameworks,' such as [Player](#), [YARP](#), [Orocos](#), [CARMEN](#), [Orca](#), [MOOS](#), and [Microsoft Robotics Studio](#).

The ROS runtime "graph" is a peer-to-peer network of processes (potentially distributed across machines) that are loosely coupled using the ROS communication infrastructure. ROS implements several different styles of communication, including synchronous RPC-style communication over [services](#), asynchronous streaming of data over [topics](#), and storage of data on a [Parameter Server](#). These are explained in greater detail in our [Conceptual Overview](#). [<http://wiki.ros.org/ROS/Introduction>]

ROS Computation Graph Level

The Computation Graph is the peer-to-peer network of ROS processes that are processing data together. The basic Computation Graph concepts of ROS are nodes, Master, Parameter Server, messages, services, topics, and bags, all of which provide data to the Graph in different ways.

For these labs we will be focused on nodes, Master, messages and topics. A complete description can be found in <http://wiki.ros.org/ROS/Concepts>.

Nodes: Nodes are processes that perform computation. ROS is designed to be modular at a fine-grained scale; a robot control system usually comprises many nodes. For example, one node controls a laser range-finder, one node controls the wheel motors, one node performs localization, one node performs path planning, one node provides a graphical view of the system, and so on. A ROS node is written with the use of a ROS client library, such as roscpp or rospy.

Master: The ROS Master provides name registration and lookup to the rest of the Computation Graph. Without the Master, nodes would not be able to find each other, exchange messages, or invoke services.

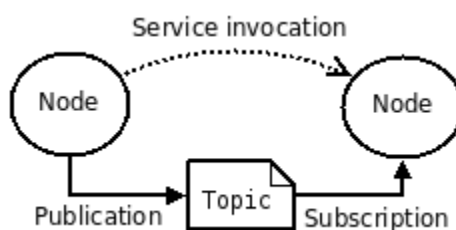
Parameter Server: The Parameter Server allows data to be stored by key in a central location. It is currently part of the Master.

Messages: Nodes communicate with each other by passing messages. A message is simply a data structure, comprising typed fields. Standard primitive types (integer, floating point, boolean, etc.)

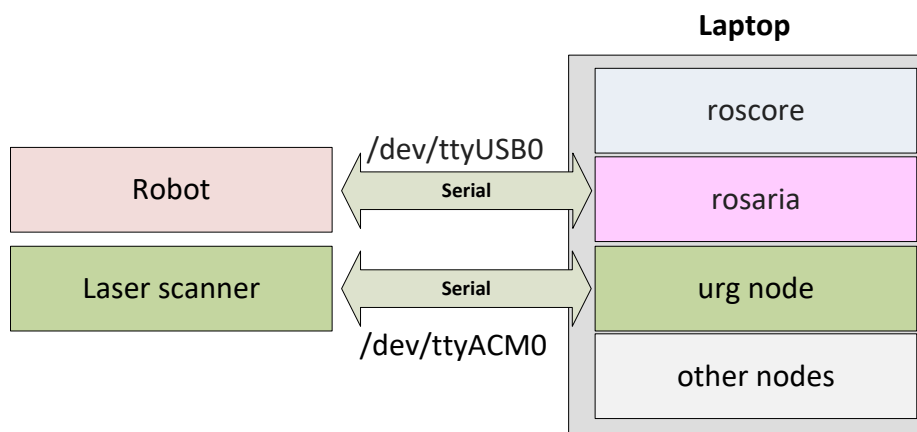
COMPSYS 726 Lab manual

are supported, as are arrays of primitive types. Messages can include arbitrarily nested structures and arrays (much like C structs).

Topics: Messages are routed via a transport system with publish / subscribe semantics. A node sends out a message by publishing it to a given topic. The topic is a name that is used to identify the content of the message. A node that is interested in a certain kind of data will subscribe to the appropriate topic. There may be multiple concurrent publishers and subscribers for a single topic, and a single node may publish and/or subscribe to multiple topics. In general, publishers and subscribers are not aware of each others' existence. The idea is to decouple the production of information from its consumption. Logically, one can think of a topic as a strongly typed message bus. Each bus has a name, and anyone can connect to the bus to send or receive messages as long as they are the right type.



Overall architecture of our system



- **roscore** is a collection of [nodes](#) and programs that are pre-requisites of a ROS-based system. You **must** have a roscore running in order for ROS nodes to communicate. It is launched using the roscore command.
 - NOTE: If you use [roslaunch](#), it will automatically start roscore if it detects that it is not already running.
 - roscore will start up:
 - ROS [Master](#) - The ROS Master provides naming and registration services to the rest of the nodes in the ROS system. It tracks publishers and subscribers to topics as well as services. The role of the Master is to enable individual ROS nodes to locate one another. Once these nodes

COMPSYS 726 Lab manual

have located each other they communicate with each other peer-to-peer.

- ROS [Parameter Server](#) - A parameter server is a shared, multi-variate dictionary that is accessible via network APIs. Nodes use this server to store and retrieve parameters at runtime.
- [rosout](#) - System-wide logging mechanism for messages sent to the /rosout topic.
- **rosaria** node provides a ROS interface for most Adept MobileRobots, MobileRobots Inc., and ActivMedia mobile robot bases including Pioneer2, Pioneer3.
[<http://wiki.ros.org/ROSARIA>]
 - Information from the robot base, and velocity and acceleration control, is implemented via a RosAria node, which publishes topics providing data recieved from the robot's embedded controller by ARIA, and sets desired velocity, acceleration and other commands in ARIA when new commands are received from command topics
- **urg node** provides a ros interface to the laser scanner. The laser scanner is a laser range finder for the robot to determine the distance of objects in front of it.
 - A ROS node to provide access to SCIP 2.2-compliant laser range finders. [http://wiki.ros.org/urg_node]

COMPSYS 726 Lab manual

Part 1: Setup

Setup (do these steps at the start of each session with the robot!):

(You can skip S1 to S6 if you already have the workspace on your SD card)

- S1. Boot into “Teaching Linux” on your desktop
 - a. Restart the computer
 - b. Select Linux when the boot option appears



Figure 3: Boot selection

- S2. Download the “lab_resource.zip” from canvas
- S3. Extract the zip file.
- S4. Copy the “workspace.tar.gz” onto your micro SD card with the USB adapter
- S5. Eject the USB and unplug it from the desktop PC
- S6. Move the micro SD card from the USB adapter onto the SD card adapter
- S7. Turn on the robot and the laptop on top of the robot
- S8. Wait until they are fully booted
- S9. Insert the SD card into the robot’s SD card slot
- S10. Copy the “workspace.tar.gz” onto the Desktop of the robot
- S11. Extract the “workspace.tar.gz”

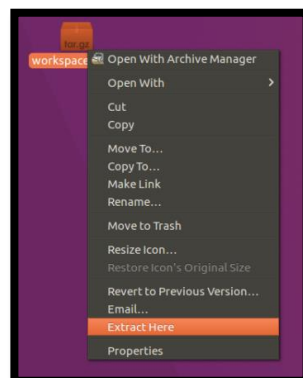


Figure 4: Extract tar.gz file

- S12. You should now have the workspace folder on the desktop
- S13. You are ready to start using the robot

COMPSYS 726 Lab manual

Clean up (do these steps at the end of each session with the robot!):

- S1. Copy any changed files from the workspace folder onto your SD Card
 - a. **IMPORTANT:** copying the workspace folder directly will not work since it contains symbolic links to the laptop.
 - b. If you wish to copy entire workspace folder you will need to compress it first and then copy the compressed file. Follow Figure 5: Compressing the workspace folder to compress the workspace folder.

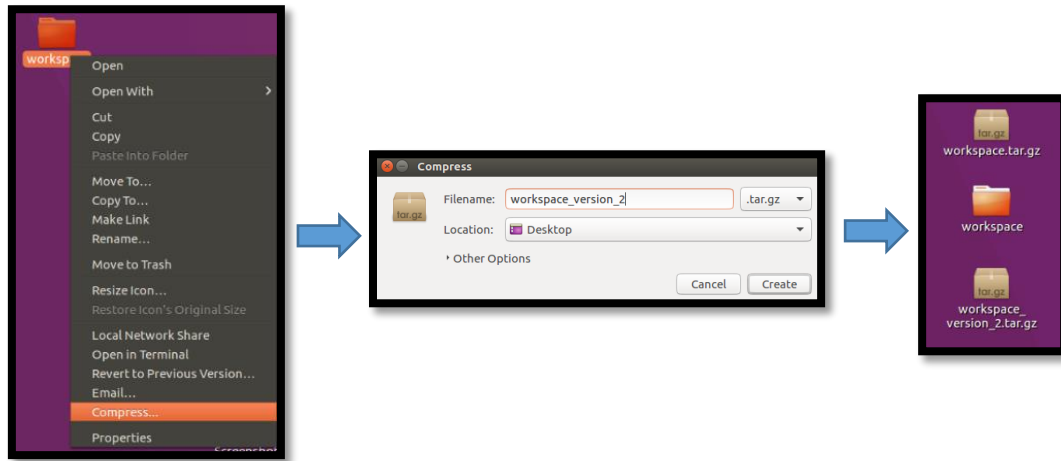


Figure 5: Compressing the workspace folder

- S2. Eject your SD card from the robot's laptop
- S3. Move the micro SD card from the SD card adapter to your USB adapter
- S4. Plug your USB adapter onto you the lab PC
- S5. Backup your SD Card content
 - a. Option 1: Copy everything onto your H Drive
 - b. Option 2: Use GIT for version control. You can commit any changed file to github. A tutorial is in <https://try.github.io/> and <https://guides.github.com/>
- S6. Delete your workspaces folder and anything you added from the laptop.
- S7. Shutdown the laptop on the robot and turn off the robot
- S8. Place the robot back into the cabinet
- S9. Make sure **BOTH** the robot and the laptop has its charging cables plugged in securely.

COMPSYS 726 Lab manual

Part 2: Robot manual control

Manual movement

- S1. Lift and move the robot to your workstation. Make sure there is sufficient space on the floor.
- S2. Turn on the robot and the laptop and wait until it is fully booted. This may take a few minutes.
- S3. Double tap the motor button (white) to test the wheels of the robot.
 - I. If nothing happens wait for 20 seconds and try again
 - II. Make sure both wheels move in accordance to motor_test.mov in the resources folder or <https://youtu.be/wOOH7Yodkz0>
- S4. Open a new terminal on the laptop by clicking on the Ubuntu icon and type in “term” and click on the Terminal icon. **[Let’s call this terminal 1]**

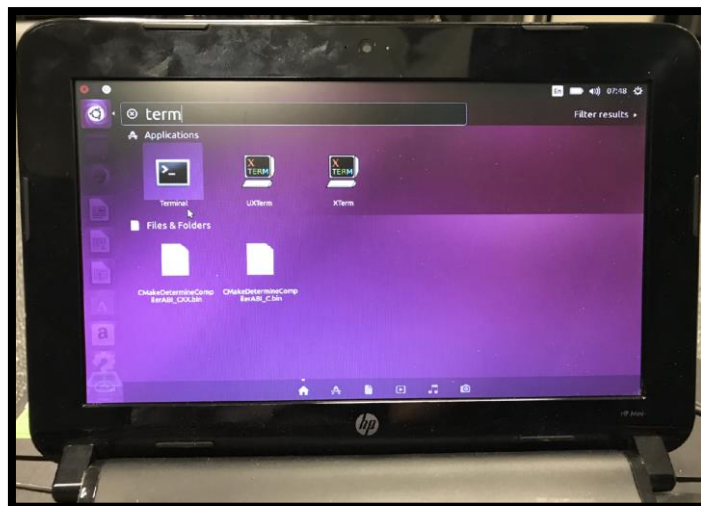


Figure 6: Open a new terminal

IMPORTANT: Only type in the white text into the terminal and not text of any other colour. The \$ symbol indicates a new line.

- S5. To start ROS we first **start the ROS master (using roscore)**. Type in the following into terminal 1 **(MAKE SURE YOU HAVE FOLLOWED THE SETUP instructions in Part 1)**:

```
$cd ~/Desktop/workspace/ //navigate to the workspace folder
$source devel/setup.bash //setup the environment for this terminal
$roscore //start the roscore
```

IMPORTANT: Whenever you start a new terminal always type in the first 2 lines so that the environment of the terminal is setup correctly. If you see errors such as “[rospack] Error: package 'x' not found” make sure you have setup the environment with the “source” command and the case and spelling is correct.

COMPSYS 726 Lab manual

```
roscore http://dopey:11311/
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://dopey:36001/
ros_comm version 1.12.7

SUMMARY
=====
PARAMETERS
* /rostdistro: kinetic
* /rosversion: 1.12.7

NODES
I

auto-starting new master
process[master]: started with pid [2382]
ROS_MASTER_URI=http://dopey:11311/

setting /run_id to ee761dc0-3b65-11e8-9461-cc52af9603cb
process[roscout-1]: started with pid [2432]
started core service [/roscout]
```

Figure 7: roscore expected outputs – should have no errors (in red)

- S6. Open a new terminal on the laptop. **[Let's call this terminal 2].**
S7. **Start the robot interface node (rosaria).** Type in the following in terminal 2:

```
$cd ~/Desktop/workspace/ //navigate to the workspace folder
$source devel/setup.bash //setup the environment for this terminal
$roslaunch rosaria RosAria //start the rosaria node (robot interface)
```

- I. Expected output with no errors:

```
robot@dopey: ~/engenuity_urg
[ INFO] [1523217191.414069515]: RosAria: set port: [/dev/ttyUSB0]
Connecting to robot using TCP connection to localhost:8101...
Could not connect to simulator, connecting to robot through serial port /dev/tty
USB0.
Syncing 0
Syncing 1
Syncing 2
Connected to robot.
Name: Auckland_3820
Type: Pioneer
Subtype: p3dx-sh
ArConfig: Config version: 2.0
Loaded robot parameters from /usr/local/Aria/params/p3dx-sh.p
Robot Serial Number: GMC_3820
ArRobotConnector: Connecting to MTX batteries (if necessary)...
ArRobotConnector: Connecting to MTX sonar (if necessary)...
[ INFO] [1523217192.392943928]: This robot's TicksMM parameter: 128
[ INFO] [1523217192.399865730]: This robot's DriftFactor parameter: 0
[ INFO] [1523217192.407282780]: This robot's RevCount parameter: 16570
[ INFO] [1523217192.559618242]: RosAria: publishing new recharge state 0.
[ INFO] [1523217192.560122149]: RosAria: publishing new motors state 0.
[ INFO] [1523217192.599138175]: rosaria: Setup complete
[ INFO] [1523217194.007167875]: RosAria: publishing new motors state 1.
```

- II. If you get errors like this:

```
robot@dopey:~/engenuity_urg$ roslaunch rosaria RosAria
[ INFO] [1523217059.433515138]: RosAria: set port: [/dev/ttyUSB0]
Connecting to robot using TCP connection to localhost:8101...
Could not connect to simulator, connecting to robot through serial port /dev/tty
USB0.
ArSerialConnection::open: Could not open serial port '/dev/ttyUSB0' | ErrorFromO
SNum: 13 ErrorFromOSSString: Permission denied
Could not connect, because open on the device connection failed.
Failed to connect to robot.
[ERROR] [1523217059.547761035]: RosAria: ARIA could not connect to robot! (Check
-port parameter is correct, and permissions on port device, or any errors repor
ted above)
[FATAL] [1523217059.547969584]: RosAria: ROS node setup failed...
```

- Type in the following (see Part 0c for password):

```
$sudo chmod 777 /dev/ttyUSB0
```

- Retry step 7.

COMPSYS 726 Lab manual

- S8. Open a new terminal on the laptop. **[Let's call this terminal 3].**
- S9. To move the robot, let's send a command to the robot by publishing to the /RosAria/cmd_vel topic. In terminal 3 type (be careful of the spaces):

```
$cd ~/Desktop/workspace/           //navigate to the workspace folder
$source devel/setup.bash           //setup the environment for this terminal
$rostopic pub -1 /RosAria/cmd_vel geometry_msgs/Twist '[0.01,0,0]' '[0,0,0]'
```

The robot should move a little bit forward.

Explanation of the command:

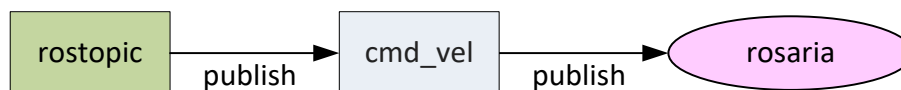


Figure 8: Diagram depicting publishing a message from rostopic to rosaria

- Rostopic (<http://wiki.ros.org/rostopic>)
 - command-line tool displays information about ROS topics. Currently, it can display a list of active topics, the publishers and subscribers of a specific topic, the publishing rate of a topic, the bandwidth of a topic, and messages published to a topic. The display of messages is configurable to output in a plotting-friendly format.
- Pub
 - Publish
- -1
 - Publish once
- /RosAria
 - Namespace for rosaria messages
- /cmd_vel
 - the topic
- geometry_msgs/Twist
 - message type
- '[x speed, y speed, z speed]' '[x angular velocity, y angular velocity, z angular velocity]'
 - Speed is in meters per second
 - Angular velocity is in radians per second
 - Note: for this pioneer robot the only degrees of freedom you can command are the **x linear velocity** and **z angular velocity**.

- S10. Let's try to make the robot spin. If the previous command is still engaged press "CTRL+C" in terminal 3. Now try typing the following into terminal 3:

```
$rostopic pub -r 10 /RosAria/cmd_vel geometry_msgs/Twist '[0,0,0]' '[0,0,0.05]'
```

- -r 10
 - Publish at the rate of 10Hz (i.e., every 100ms)

COMPSYS 726 Lab manual

S11. Stop the robot by pressing “CTRL+C” in terminal 3. If the robot is still spinning type in:

```
$rostopic pub -1 /RosAria/cmd_vel geometry_msgs/Twist '[0,0,0]' '[0,0,0]'
```

- Sending '[0,0,0]' '[0,0,0]' will stop all motion on the robot.
- **Alternatively:** press the white motor button to temporary disable the motor. Remember to press it again to enable it when you are ready to type in a new command.

S12. Try typing in your own arguments (note that 1ms^{-1} for speed is very fast for the lab)

Manual sensor data acquisition

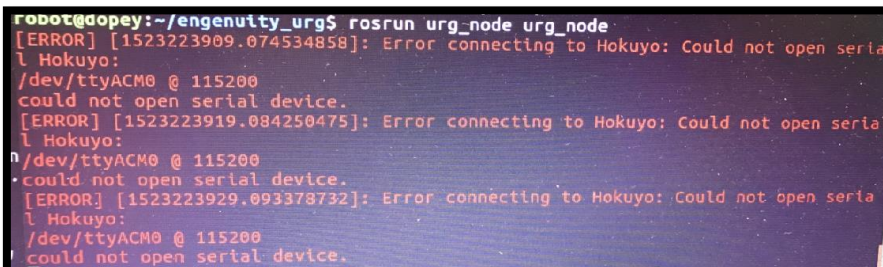
(Continue from previous section)

S13. Stop the robot from moving and terminate what is executing in terminal 3.

S14. **Start the laser scanner node.** Type the following in terminal 3:

```
$roslaunch urg_node urg_node
```

- There should be no errors (red text).
- If you encounter an error like this:



```
robot@dopey:~/engenuity_urg$ roslaunch urg_node urg_node
[ERROR] [1523223909.074534858]: Error connecting to Hokuyo: Could not open serial
l Hokuyo:
/dev/ttyACM0 @ 115200
could not open serial device.
[ERROR] [1523223919.084250475]: Error connecting to Hokuyo: Could not open serial
l Hokuyo:
/dev/ttyACM0 @ 115200
could not open serial device.
[ERROR] [1523223929.093378732]: Error connecting to Hokuyo: Could not open serial
l Hokuyo:
/dev/ttyACM0 @ 115200
could not open serial device.
```

- Type in the following (see Part 0c for password):

```
$sudo chmod 777 /dev/ttyACM0
```

S15. Open a new terminal on the laptop. **[Let's call this terminal 4].**

S16. To view the sensor raw sensor data type the following in terminal 4:

```
$cd ~/Desktop/workspace/ //navigate to the workspace folder
$source devel/setup.bash //setup the environment for this terminal
$rostopic echo /scan
```

- The echo argument for rostopic prints the messages in a topic onto the terminal. The topic we are printing from is the “scan” topic. The laser scans the distance of object within 180 degrees in front of the robot. You should see lots of numbers scrolling on the screen. Each data represents the distance from the robot. The position of the data represents the angle.

S17. Now terminate the rostopic running on terminal 4 by pressing “CTRL+C”

S18. To visualize the data, type in the following in terminal 4:

```
$roslaunch rviz rviz
```

S19. You should now be presented with the RVIZ GUI

S20. Under the “Global Options” change the Fixed Frame from “map” to “laser”

S21. Press “Add” and select “LaserScan” and then press “OK”

COMPSYS 726 Lab manual

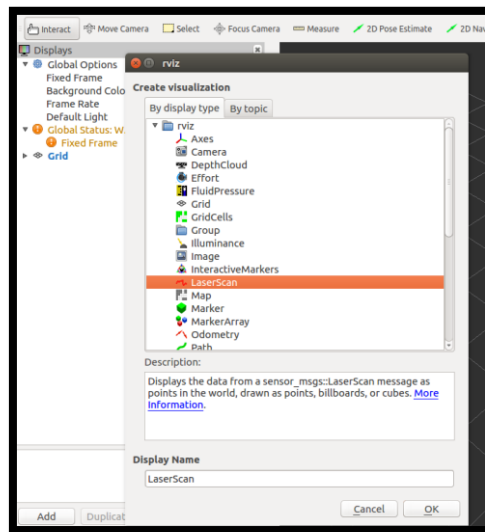


Figure 9: add visualisation of laser scanner

S22. Under the “Topic” in “LaserScan” type in “/scan”. This will get RVIZ to read from the “/scan” topic. You should now see white dots. The white dots represent the distance detected by the laser scanner. Try asking your group partner to step in front of the robot. The white dots should reflect the legs of your group partner.

- I. You can also try different Global Option fixed frames i.e. (base_link)
- Each square in the grid is 1m x 1m. The laser scanner will be at the centre of the grid.
 - Try resizing points and changing views and make sure you can match the data with objects in the environment. To do so drag the map by holding “left mouse button” to rotate and by holding “right mouse button” to zoom.

You should see something like the following:

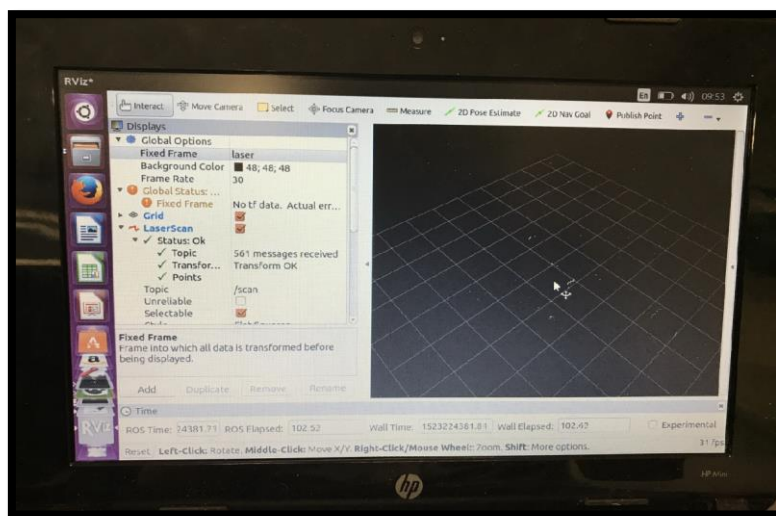


Figure 10: Visualisation of laser scanner data

COMPSYS 726 Lab manual

Note: if you see many white dots close to the robot but you are very sure there are no objects within 6 meters of the robot and those white dots do not change as you move objects in front of the robot. Try wiping the sensor with a clean soft cloth or tissue. If it does not fix the issue you may need to report an issue of the laser scanner. However, some white dots are to be expected due to noise in the sensors.

COMPSYS 726 Lab manual

Part 3: Simulation – basic

Purpose: Create a new workspace and a basic publisher in ROS with example specific to Pioneer robot. The robot workspace have been provided to you in the resource file from Canvas. Here we will be creating a workspace for simulation. These are the steps creating a new workspace on a robot. It is important to know that the robot and the simulation uses different workspaces because they require different ROS node packages. The robot requires RosAria and urg node, whereas, the simulation does not.

- S1. Go to the lab computer and make sure you have booted into Linux
S2. Creating a ROS workspace. Open a new terminal, let's call this Terminal 1.
S3. Type the following in terminal 1:

```
$ nano .bashrc //list the files
```

- S4. Add the following in the end of the file
S5. "source /opt/ros/kinetic/setup.bash"
S6. Press "CTRL+O" and enter to save
S7. Press "CTRL+X" to exit – You have now setup your terminals to the installed path of ROS
S8. Close and open the terminal (**IMPORTANT**).
S9. Type the following in terminal 1:

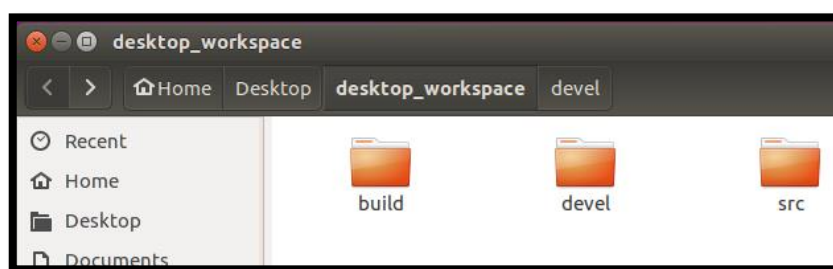
```
$ cd ~/Desktop/ //navigate to the desktop folder
$ mkdir -p desktop_workspace/src //create the workspace and src folder
$ cd desktop_workspace/src/
$ catkin_init_workspace //initialise the workspace
$ ls //list the files
```

- S10. You should see the CMakeLists.txt was created in the src directory
S11. Navigate up one directory and build the project:

```
$ cd ..
$ catkin_make
```

- There should be no errors.

- S12. Using the GUI open the desktop_workspace folder. You should see 3 folders: "build", "devel" and "src".



- S13. Overlay your workspace on top of the ROS environment. This is very important. Get into the habit of doing this when you want to build or run something from your workspace. You will need to type in the following in every new terminal and each time you rebuild the workspace. Type the following in terminal 1:

```
$ source devel/setup.bash //make sure you have already navigated to the workspace folder
```

- **source devel/setup.bash**, setups up the terminal with the environment variables, defined by the built ROS workspace. This includes where to find the compiled ros nodes so that when you type "roslaunch" the correct executable can be found.

COMPSYS 726 Lab manual

S14. From the “lab_resource.zip” downloaded from canvas, copy everything inside the folder “Resource 1” to the “src” folder. Do not include the “Resource 1” folder itself. These contain the source files we will use for simulation and mapping later in the lab. The “src” folder should now contain the folders “openslam_gmapping” and “slam_gmapping”. We will be using those later in the lab.

S15. Rebuild the project by typing the following in terminal 1:

```
$catkin_make
```

- There should be no errors (red text). If there are errors try calling “catkin_make” again 2 to 3 times. Sometimes there are errors due to build dependencies.
- **catkin_make** is combination command of cmake, make and mkdir that build any packages located at its current workspace.

S16. Now let’s create a “robot_driver” package to control a robot. Navigate to the “src” folder by typing the following in terminal 1:

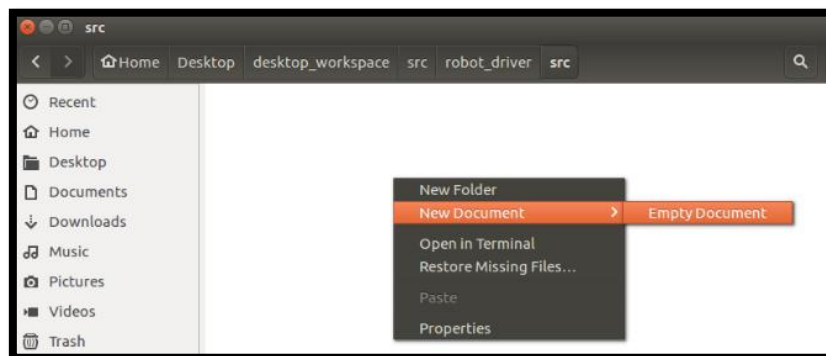
```
$cd src
```

S17. Now let’s create a custom package to control a robot. Navigate to the “src” folder by typing the following in terminal 1:

```
$catkin_create_pkg robot_driver std_msgs roscpp
```

- The package we created is called “robot_driver” and it depends on the std_msgs (standard messages) and roscpp (ros c++).

S18. Using the graphic user interface navigate to the “desktop_workspace/src/robot_driver/src” and create a new text file called “pioneerLaser.cpp”.



S19. Copy and paste the code from “code1.cpp” from the “Resource 3” folder located in the “lab_resource.zip” downloaded from canvas into the “pioneerLaser.cpp” file. Make sure you then save the file.

- I. Read the code and try to understand what it is doing.

Basic Definitions

- I. **Node:** a ROS executable (uses ROS to communicate with other nodes).
- II. **Topics:** Nodes publish and subscribe to topics.
- III. **Messages:** data types for publishing or subscribing to a topic.
- IV. **Master:** mostly helps nodes find each other, part of roscore.
- V. **Publisher `ros::NodeHandle::advertise`**

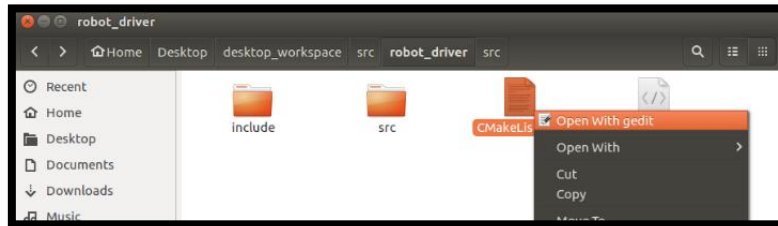
The `advertise()` function is how you tell ROS that you want to publish on a given topic name. This invokes a call to the ROS master node, which keeps a registry of who is publishing and who is subscribing. After this `advertise()` call is made, the master node will notify anyone who is trying to subscribe to this topic name, and they will in turn negotiate a peer-to-peer connection with this node. `advertise()`

COMPSYS 726 Lab manual

returns a Publisher object which allows you to publish messages on that topic through a call to `publish()`. Once all copies of the returned Publisher object are destroyed, the topic will be automatically unadvertised.

The second parameter to `advertise()` is the size of the message queue used for publishing messages. If messages are published more quickly than we can send them, the number here specifies how many messages to buffer up before throwing some away.

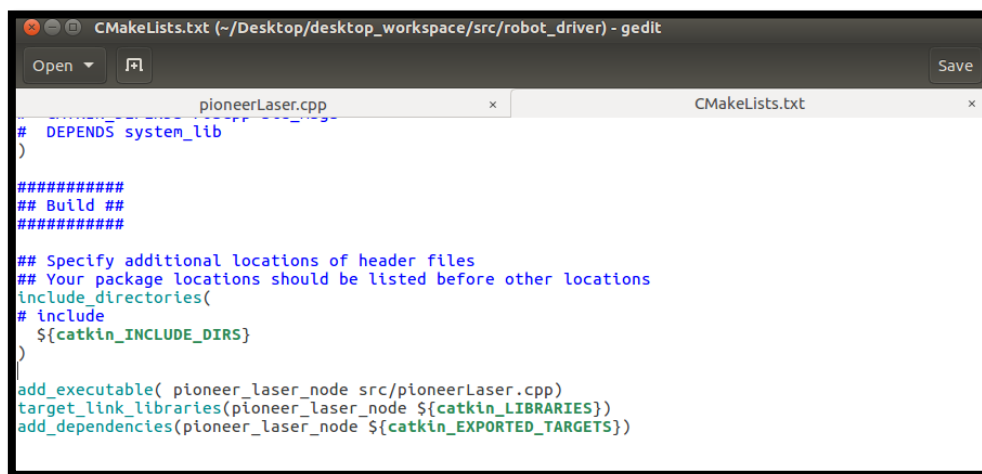
S20. Navigate to the “desktop_workspace/src/robot_driver” folder and edit the “CMakeLists.txt” file.



S16. Add the following lines of code after line 121:

```
add_executable( pioneer_laser_node src/pioneerLaser.cpp)
target_link_libraries(pioneer_laser_node ${catkin_LIBRARIES})
add_dependencies(pioneer_laser_node ${catkin_EXPORTED_TARGETS})
```

S21. You should see the following. Click Save.



S22. Copy everything inside the folder “Resource 2” to the “desktop_workspace/src/robot_driver/” folder. Do not include the “Resource 2” folder itself. It should be the “world” and “launch” folders. We will be using them later in the lab.

S23. Rebuild you workspace by typing the following in terminal 1:

```
$ cd ~/Desktop/desktop_workspace //navigate to the workspace folder
$ catkin_make //build the workspace
$ source devel/setup.bash //setup up the environment
```

COMPSYS 726 Lab manual

S24. Using the GUI navigate to the “desktop_workspace/src/robot_driver/world/” folder. Open the “myworld.world” file using a text editor. World files are files used by the simulator to describe a virtual environment. Read the file and its comments and try to understand it. Details can be found in <http://player-stage-manual.readthedocs.io/en/stable/>

S25. Start the ROS master node by typing in the following in terminal 1:

```
$ roscore //start the master node
```

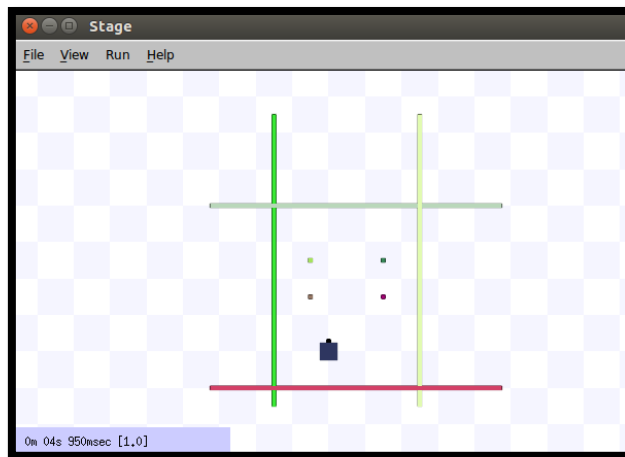
- The master node must always be running for other nodes to communicate

S26. Open a new terminal, let's call it terminal 2

S27. To run the stage simulator in terminal 2 type in the following:

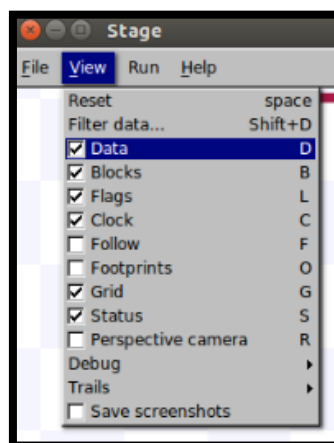
```
$ cd ~/Desktop/desktop_workspace //navigate to the workspace folder
$ source devel/setup.bash //setup up the environment
$ rosrun stage_ros stageros src/robot_driver/world/myworld.world
```

S28. You should see the following:



S29. Try holding the left or right mouse button to drag the map and try scrolling with the mouse wheel. (see <https://youtu.be/8ohDMsRjZ48>)

S30. Click on “view>data” and you can now see the simulated laser scanner.



S31. Open a new terminal, let's call it terminal 3.

S32. To run the our robot control code, in terminal 3 type in the following:

```
$ cd ~/Desktop/desktop_workspace //navigate to the workspace folder
$ source devel/setup.bash //setup up the environment
```

COMPSYS 726 Lab manual

```
$ rosrun robot_driver pioneer_laser_node /RosAria/cmd_vel:=/cmd_vel
```

- You should see the robot in the simulator move forward a little bit
- `roslaunch robot_driver pioneer_laser` execute the `pioneer_laser` node in the robot driver package
- `/RosAria/cmd_vel:=cmd_vel` maps messages publishing to the `/RosAria/cmd_vel` topic to the `/cmd_vel` topic. The simulator uses the `/cmd_vel` topic rather than the `/RosAria/cmd_vel` topic used by the RosAria node.

S33. Type in the following code in terminal 3 a few more times to see the robot move.

```
$ rosrun robot_driver pioneer_laser_node /RosAria/cmd_vel:=/cmd_vel
```

- Each time our node runs, the robot will move for 50 times at 10Hz. (i.e. for 5 seconds).

COMPSYS 726 Lab manual

Part 4: Launch files

By now you're probably getting tired of opening a new terminal for roscore and every new node that needs to be run. So we'll use launch files to run roscore and multiple nodes at once. Launch files will also be important for when we want to set node parameters (for mapping) and for name remapping (for using the same node with stage simulator and rosaria). This lab follows from the barebones subscriber in lab 2. Here we create a launch file to run everything using a single terminal. Continue from Part 3.

- S1. Go to the **lab computer** and make sure you have booted into Linux
- S2. Close all terminals and windows.
- S3. Navigate to the “desktop_workspace/src/robot_driver/launch” folder
- S4. Open and take a look at the file “launchSimulation.launch”
 - The launch file is an XML describing how to run the ROS nodes.
 - See for details <http://wiki.ros.org/roslaunch/XML>
 - The parameter use_sim_time enables simulation time
 - The nodes declare which nodes to execute/
 - The topic published to by pioneer_laser_node is remapped from RosAria/cmd_vel to cmd_vel because Stage uses cmd_vel.
 - Similarly, the topic published to by stageros is remapped so that pioneer_laser_node can use it.
 - There is no urg_node or rosaria in the simulation.
- S5. Navigate to the “desktop_workspace/src/robot_driver/src” folder
- S6. Open and edit the file “pioneerLaser.cpp”
 - Change the for loop in line 29 to **while(ros::ok())**
- S7. Open a new terminal, let's call it terminal 1
- S8. To rebuild the workspace type the following in terminal 1:

```
$ cd ~/Desktop/desktop_workspace //navigate to the workspace folder
$ catkin_make //rebuild the workspace
```

- S9. To execute the roscore, the robot driver and the stage simulator type in the following in terminal 1:

```
$ source devel/setup.bash //setup up the environment
$ roslaunch src/robot_driver/launch/launchSimulation.launch
```

- The robot should now be moving forward in the simulator

- S10. Stop execution by pressing “CTRL+C” a few times.
- S11. Open and edit the file “pioneerLaser.cpp” by copying the code in “code2.cpp” in the “resource 3” folder downloaded from Canvas.
 - Look at the comments and try to understand the code. Figure 11: Dependences between nodes describes the dependencies between the nodes.

COMPSYS 726 Lab manual

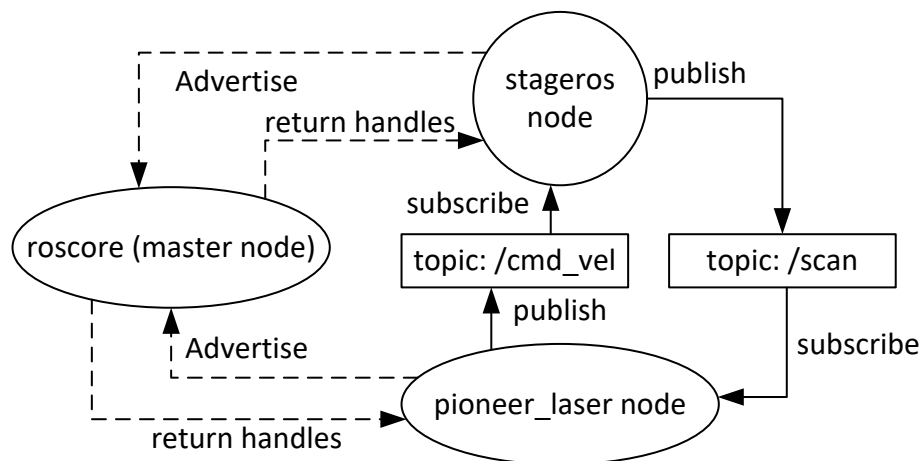


Figure 11: Dependencies between nodes

S12. Build the workspace and run the launch file by typing in the following in terminal 1:

```
$ catkin_make //rebuild the workspace
$ source devel/setup.bash //setup up the environment
$ roslaunch src/robot_driver/launch/launchSimulation.launch
```

S13. If you leave the robot running you can see that it is only moving in the first half of the map. Try changing the rotation to the right by changing 0.1 to -0.1 and then rebuild the project.

The robot should now be able to go to the second half of the map.

These are simple obstacle avoidance algorithm, for more complex problems, it is important to design a better obstacle algorithm which would allow the robot to traverse any map entirely as and avoid the obstacles along the way.

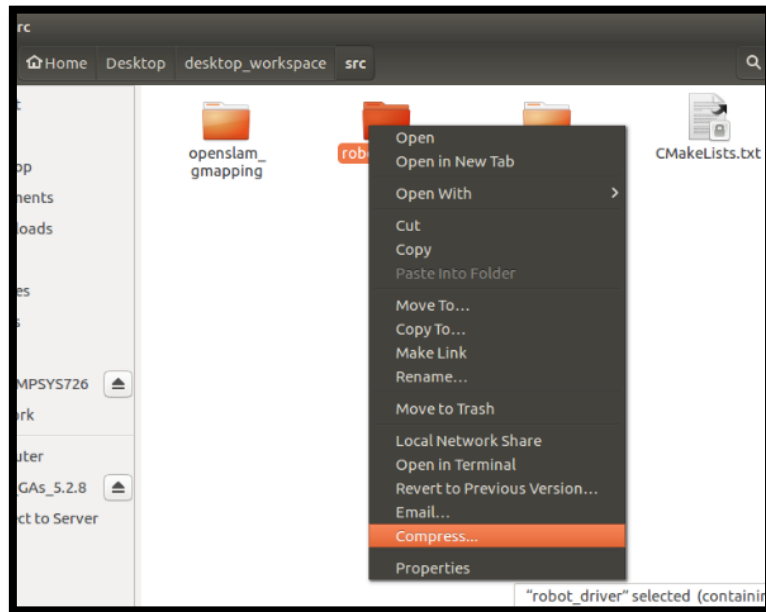
S14. (optional) Examples for reading the odometer can be found in “odomExample.cpp” under the “resource 3” folder. Take a look as you may need to know the robot’s current position to create better obstacle avoidance code. Note that in real-life on the actual robot the odometer can drift and become less accurate over time. If you want to test the code, simply copy the code from “odomExample.cpp” to your “pioneerLaser.cpp” and rebuild the workspace.

COMPSYS 726 Lab manual

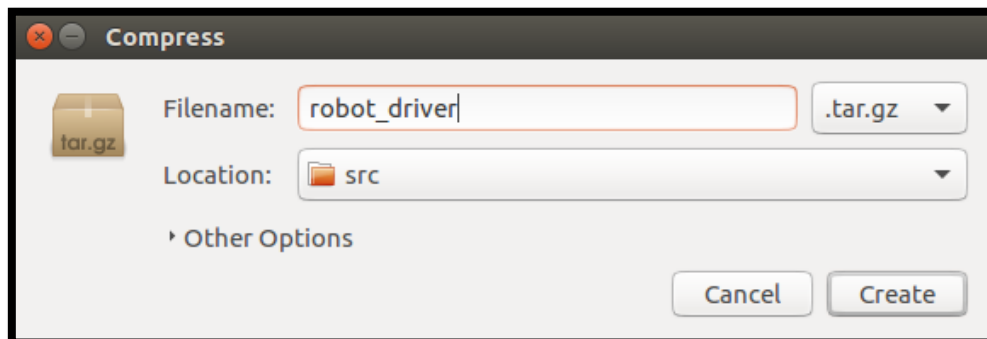
Part 5: Executing on the robot

It time to execute your robot control code on the robot.

- S1. Close all windows on the robot laptop
- S2. Compress the “desktop_workspace/src/robot_driver/” folder



- S3. Type a name i.e., robot_driver



- S4. Insert the micro SD card using the USB adapter into the lab computer and copy the compressed file (i.e., robot_driver.tar.gz).
- S5. Eject the micro SD card and remove it from the lab computer
- S6. Put the micro SD card using the SD card adapter onto the robot
- S7. Delete the “workspace/src/robot_driver” folder located on desktop. Extract your “robot_driver” folder into the “workspace/src/” folder. **If the “workspace” folder is not present do the steps in Part 1: setup.**
- S8. Take a look at the “launchRobot.launch” file inside the “launch” folder.
 - You can see that instead of the simulator it will now run the urg_node for the laser scanner and the rosaria node for the robot interface. No remapping is required since we are now using the actual robot.

COMPSYS 726 Lab manual

S9. To rebuild the workspace type the following in a new terminal:

```
$ cd ~/Desktop/desktop_workspace //navigate to the workspace folder
$ catkin_make //rebuild the workspace
```

- If you get a segmentation fault, try rebuilding it again with “catkin_make -j1”
- If you get the following make error:

```
CMake Error: The current CMakeCache.txt directory /home/canto/Desktop/workspace/build/CMakeCache.txt is different than the directory /home/robot/Desktop/workspace/build where CMakeCache.txt was created. This may result in binaries being created in the wrong place. If you are not sure, reedit the CMakeCache.txt
CMake Error: The source "/home/canto/Desktop/workspace/src/CMakeLists.txt" does not match the source "/home/robot/Desktop/workspace/src/CMakeLists.txt" used to generate cache. Re-run cmake with a different source directory.
Invoking "cmake" failed
```

- Delete the “devel” and “build” folder from the “workspace” folder and the “CMakeLists.txt” file from the “workspace/src” folder and try again.

S10. To run the launch file type in the follow into the previous terminal:

```
$ source devel/setup.bash //setup up the environment
$ roslaunch src/robot_driver/launch/launchRobot.launch
```

- If you get an error press “CTRL+C” and try the following:
- Sometimes rebooting or unplugging the USB cables will cause the permission to the hardware to reset.
- If you get an laser (urg node) error try “\$sudo chmod 777 /dev/ttyACM0”
 - See lab Part 2 s14
- If you get an rosaria error try “\$sudo chmod 777 /dev/ttyUSB0”
 - See lab Part 2 s7
- If you cannot determine the error try running the node using separate terminals instead of the launch file

Terminal 1 (roscore):

```
$ cd ~/Desktop/desktop_workspace //navigate to the workspace folder
$ source devel/setup.bash //setup up the environment
$ roscore
```

Terminal 2 (laser scanner):

```
$ cd ~/Desktop/desktop_workspace //navigate to the workspace folder
$ source devel/setup.bash //setup up the environment
$ roslaunch urg_node urg_node
```

Terminal 3 (rosaria):

```
$ cd ~/Desktop/desktop_workspace //navigate to the workspace folder
$ source devel/setup.bash //setup up the environment
$ roslaunch rosaria RosAria
```

Terminal 4 (our code):

```
$ cd ~/Desktop/desktop_workspace //navigate to the workspace folder
$ source devel/setup.bash //setup up the environment
```

COMPSYS 726 Lab manual

```
$ rosrun robot_driver pioneer_laser_node
```

- S11. The robot should now move around the lab and avoid obstacles at the same time. Try moving in front of it.
- S12. Once you are done make sure you follow the “clean up” instructions to back up your code in Part 1 of the lab.

Part 6: Simulation – SLAM

Simultaneous Localisation and Mapping (SLAM). Here we use an existing ROS package to perform Simultaneous Localisation and Mapping (SLAM). The package we will use will be gmapping. Note that gmapping is slow on the robot laptops especially if you also have rviz running, because the laptops are slow.

- S1. Go to the **lab computer** and make sure you have booted into Linux
- S2. Close all terminals and windows.
- S3. Navigate to the folder “desktop_workspace/src/robot_driver/launch”. Take a look at the “launchSimulationSLAM.launch” file. The following lines has been added to our previous launch file in Part 4.

```
<node pkg="gmapping" type="slam_gmapping" name="slam_gmapping" output="screen">
  <param name="map_update_interval" value="0.5" />
  <param name="linearUpdate" value="0.3" />
  <param name="angularUpdate" value="0.1" />
  <param name="scan" value="scan" />
  <param name="delta" value="0.25" />
  <param name="xmin" value="-2" />
  <param name="ymin" value="-2" />
  <param name="xmax" value="2" />
  <param name="ymax" value="2" />
</node>

<node pkg="tf" type="static_transform_publisher" name="baseLink_to_Laser" args="0.17 0 0.1
0 0 0 base_link laser 50">
</node>

<node pkg="rviz" type="rviz" name="rviz" />
```

- The slam_gmapping node is the node we are using to perform SLAM. We have defined that the map should update every 0.5 seconds, when the robot has travelled 0.3 of a metre or when the robot turns 0.1 rad. We set gmapping to use the scan topic for laser scanner data. We have also defined the initial map size and the resolution of the map. Full details of each argument can be found in <http://wiki.ros.org/gmapping>
- The tf node provides a transform between the robot (base_link) and the laser scanner. The args are x y z translations, followed by x y z rotations, followed by the parent frame id, followed by the child frame id and the period (in ms) of this transform being sent.
- The rviz node provide us a visualization of the robot sensors.

- S4. Execute the launch file with gmapping by typing the following in a new window:

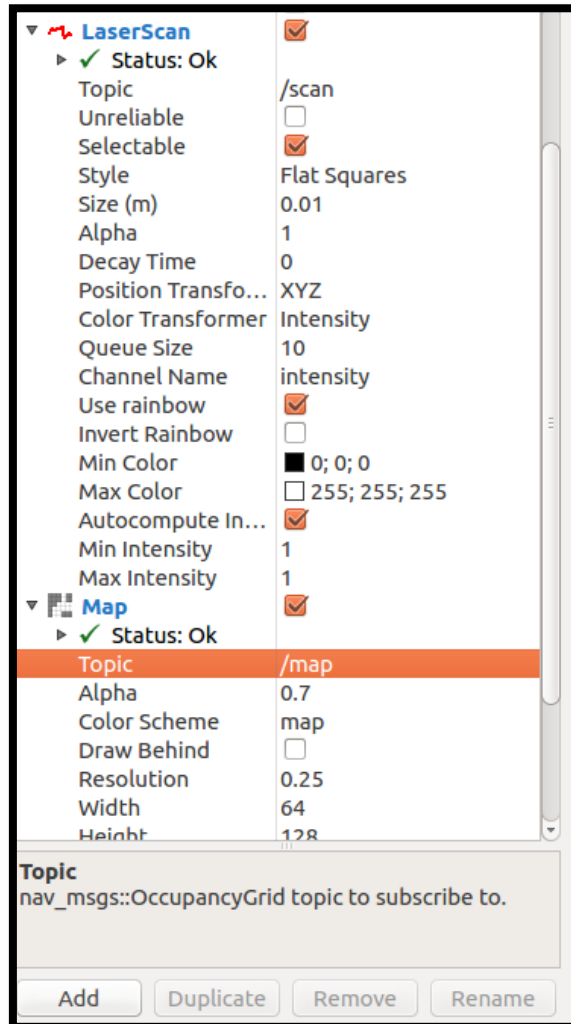
```
$ cd ~/Desktop/desktop_workspace //navigate to the workspace folder
$ catkin_make //rebuild the workspace
$ source devel/setup.bash //setup up the environment
$ roslaunch src/robot_driver/launch/launchSimulationSLAM.launch
```

COMPSYS 726 Lab manual

S5. Do the following in RVIZ:

- Add a LaserScan and set the Topic as “/scan”
- Add a Odometry and set the Topic as “/odom”
- Add a Map and set the Topic as “/map”

You should see something similar on your RVIZ panel:



- S6. You should now see the odometry (arrow), laser (dots) and map data (tilled map). The black squares represent objects detected by the SLAM algorithm. Note that the SLAM algorithm does not well with moving objects. (for more information see the video <https://youtu.be/nRWKUY4TjFk>)
- S7. Try putting this code onto the robot by following what you did in Part 5. This time use the “launchRobotSLAM.launch” file instead. Compare this launch file with “launchRobot.launch” and spot any differences.
- S8. Examples for reading the occupancy grid (i.e., map) can be found in “SLAMExample.cpp” under the “resource 3” folder. Take a look if you would like to the occupancy grid in the assignment (optional). Note that you will need to copy the code inside the “SLAMExample.cpp” to “pioneerLaser.cpp” and replace text inside the “CMakeLists.txt” in the “robot_driver/src” folder with that of the “SLAM_CMakeLists.txt” file in the “Resource 3” folder. The “SLAM_CMakeList.txt” contains the “tf” package which is required to

COMPSYS 726 Lab manual

transform the robot coordinate frame to that of the map. If you were to run the code on the simulator using the “launchSimulationSLAM.launch” file you should see the following:

[illegible]

- The * represent some probability of an object at that cell in the occupancy grid
- The R represent the robot's position in the occupancy grid
- The printed map is rotated 90 degrees compared to the simulated map