**Department of Electrical Engineering**

**Faculty Member:** Dr. Hafsa Iqbal **Date:** 14/04/24

**Semester:** 8th  **Group:** 01

# EE381 Robotics

**Lab 8: Laser Range Finder and Wall Following**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **PLO5-CLO4** | | **PLO5-CLO5** | **PLO8-CLO6** | **PLO9-CLO7** |
| **Name** | **Reg. No** | **Viva / Quiz / Demo** | **Analysis of Data in Lab Report** | **Modern Tool Usage** | **Ethics** | **Individual and Team Work** |
|  |  | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** |
| Hassan Rizwan | 335753 |  |  |  |  |  |
| Muhammad Abdullah Sohail | 343642 |  |  |  |  |  |
| Muhammad Ahmed Mohsin | 333060 |  |  |  |  |  |
| Muhammad Umer | 345834 |  |  |  |  |  |
| Muhammad Saad | 333414 |  |  |  |  |  |

# Table of Contents

[Table of Contents 2](#_Toc164006422)

[1 Laser Range Finder and Wall Following 3](#_Toc164006423)

[1.1 Introduction 3](#_Toc164006424)

[1.2 Objectives 3](#_Toc164006425)

[1.3 Lab Conduct 3](#_Toc164006426)

[1.4 Theory 4](#_Toc164006427)

[2 Lab Tasks 5](#_Toc164006428)

[2.1 Lab Task 1 – Laser Subscription \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [2] 5](#_Toc164006429)

[2.2 Lab Task 2 – Wandering Around \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [3] 6](#_Toc164006430)

[2.3 Lab Task 3 – Laser Upgrade \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [2] 8](#_Toc164006431)

[2.4 Lab Task 4 – Wall Follower \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [3] 10](#_Toc164006432)

[3 Conclusion 12](#_Toc164006433)

# Laser Range Finder and Wall Following

## Introduction

This laboratory exercise is focused on the sensing aspect of robotics, particularly the laser range finder. A laser range finder is a sensor which fires laser beams into the environment. The response in the form of reflected lasers is measured to determine the distance (range) to objects. The laser range finder is important for building maps, localizing within the map, avoiding unmapped obstacles and wall following algorithms.

## Objectives

The following are the main objectives of this lab:

* Create a subscriber node that can acquire laser range data
* Create an implementation for a simple wandering robot
* Incorporating information from multiple laser values
* Create an implementation for wall following

## Lab Conduct

* Respect faculty and peers through speech and actions
* The lab faculty will be available to assist the students. In case some aspect of the lab experiment is not understood, the students are advised to seek help from the faculty.
* In the tasks, there are commented lines such as #YOUR CODE STARTS HERE# where you have to provide the code. You must put the code between the #START and #END parts of these commented lines. Do NOT remove the commented lines.
* Use the tab key to provide the indentation in python.
* When you provide the code in the report, keep the font size at 12

## Theory

The laser data can be read by subscribing to a ‘scan’ topic which expects messages of type LaserScan. This type of message is available in the sensor\_msgs package. By sensing with the laser, the robot can respond to the obstacles present in the environment. This paves the way for building intelligent robots and also forms the basis for various algorithms. A simple algorithm involves a wandering robot which simply turns away from obstacles. Another implementation involves wall following which is used in simple navigation and bug algorithms.

Common terminal commands for ROS 2 are provided below:

* **colcon build --packages-select <package\_name>**

build a specific package whenever a node is created or modified

* **. install/setup.bash**

make the terminal “aware” of the workspace (notice the dot and space)

* **ros2 pkg create --build-type ament\_python <package\_name>**

create a new package (must be done in src directory)

* **ros2 pkg create --build-type ament\_python <package\_name> --dependencies rclpy std\_msgs geometry\_msgs sensor\_msgs**

create a new package with dependencies

* **ros2 run <package\_name> <node\_name>**

execute a node in the terminal

The first lab task will focus on using the laser range finder in the simulation. Understand the workings of the laser scan properly before proceeding to the next tasks which will involve both laser reading and velocity commanding (twist messages). Tasks 1 and 2 only involve a single laser reading (front) while tasks 3 and 4 involve using all of the laser readings. In the video submissions, show the terminal execution and also ensure your names are appearing in the video otherwise marks may be deducted.

# Lab Tasks

## Lab Task 1 – Laser Subscription \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [2]

In this task, you will get range values from the front laser. Before starting, you need to modify the laser via the **model.sdf** file in the robot package. Edit the laser sensor so that it has 36 samples, 1.57 rad max angle and -1.57 rad min angle. The laser will now span around 180 degrees with 36 laser samples.

Next, create a package called scan\_pkg in the robot workspace. The package must have the dependencies: rclpy, geometry\_msgs, sensor\_msgs. Then, create a subscriber node called scan\_ahead.py in the package. The node must subscribe to the ‘scan’ topic which expects messages of type LaserScan. This message type can be imported as follow:

**from sensor\_msgs.msg import LaserScan**

In the subscriber callback function (assuming that the message instance is called *msg*), use the syntax below to get the range data from a single laser. The index is the number of a specific laser sample (i.e. a single beam).

**msg.ranges[index]**

For this task, you need to start the robot simulation and use the laser sensor. You will also need to put obstacles in the map (such as the default cube/cylinder/sphere obstacles).

Open a terminal and activate the teleoperation program for moving the robot around. Start the scan\_ahead node in another terminal. You need to move your robot towards the obstacle. As the robot moves, the range values from only the front-most laser must be printed by the scan\_ahead node.

For submission, provide the class code for the subscriber node and make a video called labscan\_task1. The video must show the simulation of the moving robot. The terminal where the scan values are printed out must also be shown in the video.

*### SCAN AHEAD CODE STARTS HERE ###*

import rclpy

from rclpy.node import Node

from sensor\_msgs.msg import LaserScan

from std\_msgs.msg import Int32

# Class

class ScanAhead(Node):

    def \_\_init\_\_(self):

        super().\_\_init\_\_("scan\_ahead")

        self.subscription = self.create\_subscription(

            LaserScan, "scan", self.listener\_callback, 10

        )

        self.subscription

    def listener\_callback(self, msg):

        # 36 laser samples, get front-most sample

        val = min(msg.ranges[18], 10)  # 10 is the maximum range of the sensor

        self.get\_logger().info(f"Range ahead: " + str(val))

def main(args=None):

    rclpy.init(args=args)

    scan\_ahead = ScanAhead()

    rclpy.spin(scan\_ahead)

    scan\_ahead.destroy\_node()

    rclpy.shutdown()

if \_\_name\_\_ == "\_\_main\_\_":

    main()

*### SCAN AHEAD CODE ENDS HERE ###*

## Lab Task 2 – Wandering Around \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [3]

In this task, you will implement a robot which wanders around. You will use readings from only the front-most laser. Create a node called wanderbot.py in the package. The node must subscribe to the ‘scan’ topic and must also be able to publish twist messages to the ‘cmd\_vel’ topic.

The wandering algorithm is simple: The robot normally drives in a straight line. When the robot “sees” an obstacle in front within 0.7 meters, it will stop and turn to a new direction. The turn should be more than 90 degrees (it can be either a constant angle or a random angle). The robot then resumes moving in the straight line. This sequence is repeated.

For the submission, place the robot in an enclosed environment and execute the above algorithm. You need to provide the class code and make a video called labscan\_task2. The video must show the node being executed in the terminal as well as the resulting simulation.

*### WANDERBOT CODE STARTS HERE ###*

import random

import rclpy

from geometry\_msgs.msg import Twist

from rclpy.node import Node

from sensor\_msgs.msg import LaserScan

class WanderBot(Node):

    def \_\_init\_\_(self):

        super().\_\_init\_\_("wanderbot")

        self.publisher = self.create\_publisher(Twist, "cmd\_vel", 10)

        self.subscription = self.create\_subscription(

            LaserScan, "scan", self.listener\_callback, 10

        )

        self.subscription

    def listener\_callback(self, msg):

        # 36 laser samples, get front-most sample

        val = min(msg.ranges[18], 10)  # 10 is the maximum range of the sensor

        self.get\_logger().info(f"Range ahead: " + str(val))

        twist = Twist()

        if val < 0.7:  # if obstacle is closer than 0.7 meters

            # stop and turn more than 90 degrees

            twist.linear.x = 0.0

            twist.angular.z = random.uniform(

                1.57, 3.14

            )  # random angle between 90 and 180 degrees

        else:

            # move forward

            twist.linear.x = 1.0

            twist.angular.z = 0.0

        self.publisher.publish(twist)

def main(args=None):

    rclpy.init(args=args)

    wander\_bot = WanderBot()

    rclpy.spin(wander\_bot)

    wander\_bot.destroy\_node()

    rclpy.shutdown()

if \_\_name\_\_ == "\_\_main\_\_":

    main()

*### WANDERBOT CODE ENDS HERE ###*

## Lab Task 3 – Laser Upgrade \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [2]

In tasks 1 and 2, readings from only the front laser were taken. In this task, you will improve the laser scanning so that all of the lasers are being used. Create a subscriber node called scan\_around.py in the package. Divide the lasers into five regions: right, front-right, front, front-left and left. Each region gives out the smallest value among its laser samples. Thus, there will be five values printed on the terminal at any given time.

Additionally, you need to make adjustment for the out-of-range (Inf) values of the laser readings. For this, assign the value of 10 meters (max measurable distance) in place of the infinite value. Thus, the readings will max out at 10 meters.

The task submission is similar to task 1. Use teleoperation to move the robot close to some obstacles while showing the values of all 5 laser regions in a terminal. Provide the class code for the subscriber and make a video called labscan\_task3. The video must show both the simulation and the laser readings on the terminal.

*### SCAN AROUND CODE STARTS HERE ###*

import rclpy

from rclpy.node import Node

from sensor\_msgs.msg import LaserScan

from std\_msgs.msg import Int32

# Class

class ScanAround(Node):

    def \_\_init\_\_(self):

        super().\_\_init\_\_("scan\_around")

        self.subscription = self.create\_subscription(

            LaserScan, "scan", self.listener\_callback, 10

        )

        self.subscription

    def listener\_callback(self, msg):

        R = min(min(msg.ranges[0:7]), 10)

        FR = min(min(msg.ranges[7:14]), 10)

        F = min(min(msg.ranges[14:21]), 10)

        FL = min(min(msg.ranges[21:28]), 10)

        L = min(min(msg.ranges[28:35]), 10)

        self.get\_logger().info(f"Range ahead: " + str(F))

        self.get\_logger().info(f"Range right: " + str(R))

        self.get\_logger().info(f"Range front-right: " + str(FR))

        self.get\_logger().info(f"Range front-left: " + str(FL))

        self.get\_logger().info(f"Range left: " + str(L))

def main(args=None):

    rclpy.init(args=args)

    scan\_around = ScanAround()

    rclpy.spin(scan\_around)

    scan\_around.destroy\_node()

    rclpy.shutdown()

if \_\_name\_\_ == "\_\_main\_\_":

    main()

*### SCAN AROUND CODE ENDS HERE ###*

## Lab Task 4 – Wall Follower \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [3]

Wall following is any algorithm that causes a robot to move along walls without hitting them. Wall following is important for navigating around obstacles and is central to Bug algorithms. In this task, you will use the laser readings to make a moving robot loop around an obstacle via wall following.

In this task, you will be provided models which you will use in your simulation. Go to the Home directory and ensure the “show hidden files” option is checked. Then, go to Home/.gazebo/models and place the model folder in it. After this, you should be able to insert the models in the simulation. In your package, create a node called “wall\_follow.py” that will subscribe to the ‘scan’ topic and publish to the ‘cmd\_vel’ topic.

This task requires that the robot move around the model object either in clockwise or counterclockwise manner (You need to implement only one of these directions). For the start position, place the robot in such a way that the model object is to the right side of the robot (for clockwise) or to the left side of the robot (for counterclockwise). You can use teleoperation to bring the robot in the start position.

From the above mentioned start position, the node will be executed causing the robot to move around the object. The robot must use its lasers to track the wall of the object. In the clockwise case, the right and front-right laser regions need to be used. For the counterclockwise case, the left and front-left laser regions need to be used. You need to use your understanding of the laser range finder from the previous tasks. Provide the class code for the node and make a video called labscan\_task4. The video must show the node being executed in the terminal as well as the resulting simulation.

*### WALL FOLLOWER CODE STARTS HERE ###*

import rclpy

from geometry\_msgs.msg import Twist

from rclpy.node import Node

from sensor\_msgs.msg import LaserScan

class WallFollow(Node):

    def \_\_init\_\_(self):

        super().\_\_init\_\_("wallfollow")

        self.publisher = self.create\_publisher(Twist, "cmd\_vel", 10)

        self.subscription = self.create\_subscription(

            LaserScan, "scan", self.listener\_callback, 10

        )

        self.right\_min\_range = None

    def listener\_callback(self, msg):

        if self.right\_min\_range is None:

            self.right\_min\_range = min(min(msg.ranges[0:14]), 10)

        twist = Twist()

        distance\_error = 0.7 - self.right\_min\_range

        linear\_x = 0.5

        angular\_z = 0.3

        # Prioritize stopping when too close to prevent collisions:

        if distance\_error <= 0.15:

            twist.linear.x = 0.0  # Stop movement

            twist.angular.z = -angular\_z  # Turn away from the wall

        elif abs(distance\_error) < 0.05:

            twist.linear.x = linear\_x

            twist.angular.z = 0

        else:

            twist.linear.x = linear\_x - abs(distance\_error) \* 0.2

            twist.angular.z = angular\_z if distance\_error > 0 else -angular\_z

        self.publisher.publish(twist)

def main(args=None):

    rclpy.init(args=args)

    wall\_follow = WallFollow()

    rclpy.spin(wall\_follow)

    wall\_follow.destroy\_node()

    rclpy.shutdown()

if \_\_name\_\_ == "\_\_main\_\_":

    main()

*### WALL FOLLOWER CODE ENDS HERE ###*

# Conclusion

In conclusion, this lab provided a hands-on exploration of robotics sensing using laser range finders. Tasks ranged from basic obstacle detection to more sophisticated behaviors like wall following. Through implementing subscriber nodes in ROS, participants gained practical experience in processing laser scan data and responding to environmental cues.