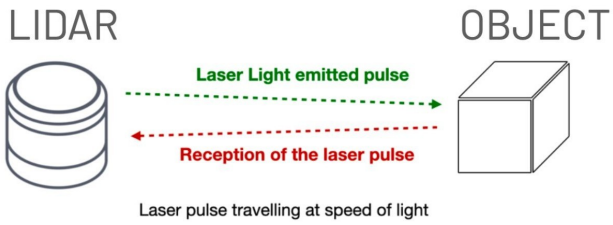
Lab #3: LiDAR-Based Perception and Simple Obstacle Avoidance

In this experiment, you will use the LiDAR sensor on your TurtleBot3 to perceive the environment in real-time and interpret distance measurements. You will also implement a simple collision avoidance behavior based on the perceived environment.

LiDAR (Light Detection and Ranging) sensors provide range measurements of surrounding objects by emitting laser beams and measuring the time it takes for reflections to return. This working principle is shown in Figure 1b.

(a) LDS-02 Sensor.



(b) LiDAR working principle.

Figure 1

By using a measured distance to an object (d) and the angle between the x-axis of the robot and the direction to the object (θ), corresponding x-y coordinates can be obtained simply by:

$$x = d\cos(\theta) \tag{1}$$

$$y = d\sin(\theta) \tag{2}$$

You will also implement a simple algorithm to maintain a safety distance between the robot and an object put in its path. While moving in a straight line, the robot should wait for few seconds. If the detected object is not moving, your robot should avoid it and continue its motion.

Environment Set-up

Refer to the Lab #0 document to connect to the **TurtleBot3** and prepare the MATLAB environment in order to start the algorithm implementation.

Things to do:

1. Write a **LiDARScan.m** script that reads a single data sequence published from the LiDAR sensor and detects the distances to surrounding objects. Change the environment around the robot each time with varying shapes, materials and colors. Ensure that you plot the environment in the Cartesian coordinate system (assuming the robot is located at the origin).

2. Write a **LiveLiDAR.m** script that reads the **real-time** data published from the LiDAR sensor and detects the distances to surrounding objects. For moving the robot, use your keyboard via ros2 terminal (refer to Appendix). Ensure that you plot the environment in a Cartesian coordinate system (assuming the robot is located at the origin).

3. Write a **SafeDistance.m** script. In this task, the robot will move forward while maintaining a safe distance from obstacles in its forward path. If an obstacle is detected within the unsafe range, the robot should stop, wait for a short duration (e.g., 5 seconds) for the object to be removed; in the opposite case, it turns 90° before resuming its movement. This behavior must rely solely on the LiDAR data. Keep the **LiveLiDAR.m** script running during this operation as well (for monitoring purposes).

- **Record a video** demonstration for applications **2 and 3**, upload the videos to a cloud platform (e.g., Google Drive, YouTube), and include the shared links in your report.

- Submit your report **via SUCourse** until the report submission deadline.

Post-Lab Report Deadline: 19 November 2025, 23:59 via **SUCourse**

- Your report must include:
  - **Introduction**
  - **Procedure**
  - **Results**
  - **Conclusion**
  - **Discussion**
  - **Appendix**
  - Provide your **MATLAB** codes in Appendix section appropriately.

Answer the following questions in the Discussion section of your Post-lab report:

1. What do your results look like when you scan the environment without any obstacles? Comment on your result.
2. Once you add an obstacle, how does your resultant plot change? What do the points around the obstacle look like and why? Visualize the detected objects on the Cartesian plot.
3. How does the LiDAR detect reflective or absorptive materials? How do the shape and the colour of the obstacle affect the measurements?
4. What is the maximum range of LiDAR? What happens when a beam returns Inf, and how would you solve this?
5. How did your implemented **SafeDistance.m** work? Comment on your results.

> Important Note
>
> Don't forget to add the links of the videos that you recorded!

Appendix

Controlling The Robot Using Keyboard

1. Open a new CMD terminal.
2. Connect to the robot in the new terminal by typing:
   `ssh me425-1@192.168.XX.1`
   (Don't close the bringup terminal, keep it running!)
3. Once connected, type:
   `ros2 run turtlebot3_teleop teleop_keyboard`
4. Use the [W,A,D, and X] keys to control the linear and angular velocities of the robot, and the [S] key to fully stop the robot.
5. Once done, press [Ctrl + C] to terminate the control.