CISC849 HW #3

For this assignment your team will implement a wall-following behavior using the lidar. Your program should allow the choice of a preferred wall side (left/right) as well as a desired lateral offset to be chosen at startup. Behaviors your robot should exhibit:

(1) Start in free space, go straight until a blocking wall is encountered in front (at a distance that depends on chosen offset).
(2) If the chosen side = left, make a right turn at this wall and move forward parallel to it at the chosen offset. If side = right, make a left turn and do the same
(3) If a blocking wall is encountered in front, turn away from side wall and continue parallel to blocking wall.
(4) If the side wall is "lost," turn toward that side until a wall is refound, then continue forward

You should not assume that all walls are straight or long, or that all turns are 90 degrees. RANSAC line-fitting is 100% not necessary for this assignment but is something you can consider experimenting with after trying simpler approaches.

As a team

- First see Sep. 26 slides ("HW #3 prep") for some Github repos you should install
- As you did on HW #2, create a package. This time call it `<robot name>_cpp_laserfollow` or `<robot name>_py_laserfollow` depending on whether you are using C++ or Python. You might want to copy over some of your code from `wanderer` to `laserfollow`, because you should stop the robot if the bumper hazard is triggered at any time. As before, you will also be publishing `cmd_vel` messages to move the robot.

- New this time: you will be subscribing to the `/<robot name>/scan` topic (type `LaserScan`). In order to start the node that publishes these messages, you must:
  a. `ssh` in to your robot's Raspberry Pi board (IP addresses are printed on the bulletin board by the door) with username `student` and password `<robot name>`.
  b. After grounding yourself, plug in the blue USB cable on the robot to power up the lidar (make sure to unplug it when you are done!)

c. Start up the lidar node on the Pi board:

```
ros2 run rplidar_ros rplidar_composition /scan:=/<robot
name>/scan --ros-args -p frame_id:=rplidar_link
```

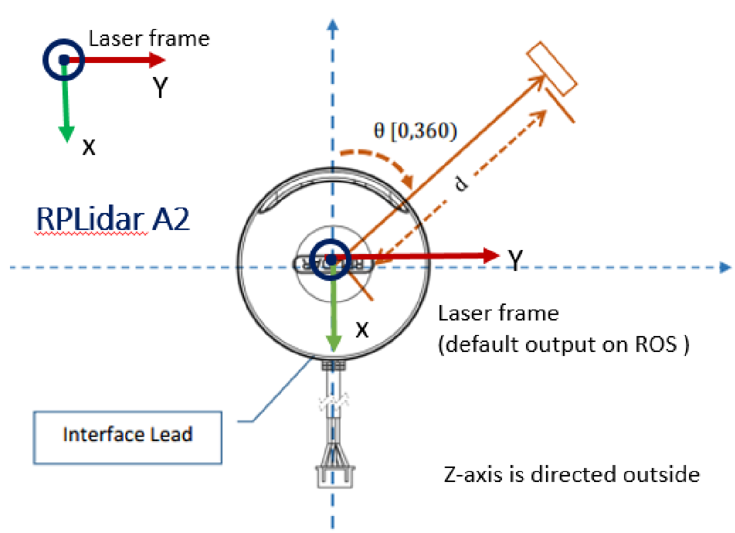
d. You should see something like this in your robot shell:

```
[INFO] [1678404921.927485908] [rplidar_node]: RPLIDAR running on ROS 2
package rplidar_ros. SDK Version: '1.12.0'
[INFO] [1678404924.439203400] [rplidar_node]: RPLIDAR S/N:
B793EDF9C7E29BCEA7E39EF227404304
[INFO] [1678404924.439449711] [rplidar_node]: Firmware Ver: 1.29
[INFO] [1678404924.439589023] [rplidar_node]: Hardware Rev: 7
[INFO] [1678404924.441306516] [rplidar_node]: RPLidar health status : '0'
[INFO] [1678404924.441455587] [rplidar_node]: Start
[INFO] [1678404924.985615468] [rplidar_node]: current scan mode:
Sensitivity, max_distance: 12.0 m, Point number: 7.9K , angle_compensate:
2, flip_x_axis 0
```

e. Verify on your laptop that `/<robot name/scan` messages are being published and that you can `echo` them

f. Now see if you can visualize the `LaserScan` messages in `rviz2`. Your fixed frame should be `base_link`

- Now actually subscribe to the `LaserScan` messages in your `laserfollow` program and start developing your wall-following logic. The diagram below may be helpful in establishing a coordinate system if you want to convert the (r, theta) values in a `LaserScan` to (x, y) coordinates.



- In order to display your program state in `rviz2` while running, publish some kind of [`Marker`] shape or shapes that indicate which of the 4 states above is in effect.

Deliverables

Have one member of your team submit your code on Canvas along with a screen recording (or phone video of laptop screen) showing your robot in `rviz2` with lidar and markers overlaid doing as much of a circuit around the robot area perimeter as possible and a README file describing your approach (PDFs with simple explanatory figures welcome!).