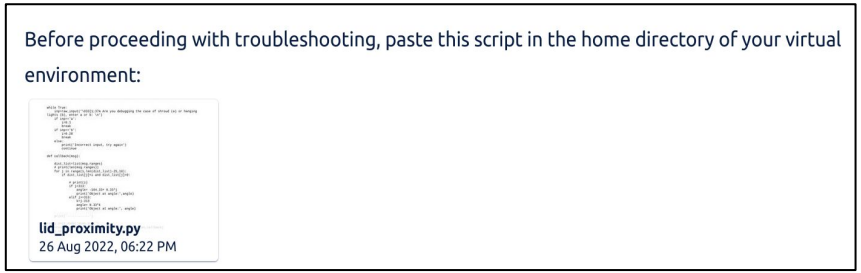
LIDAR PROXIMITY DETECTION

Goals:

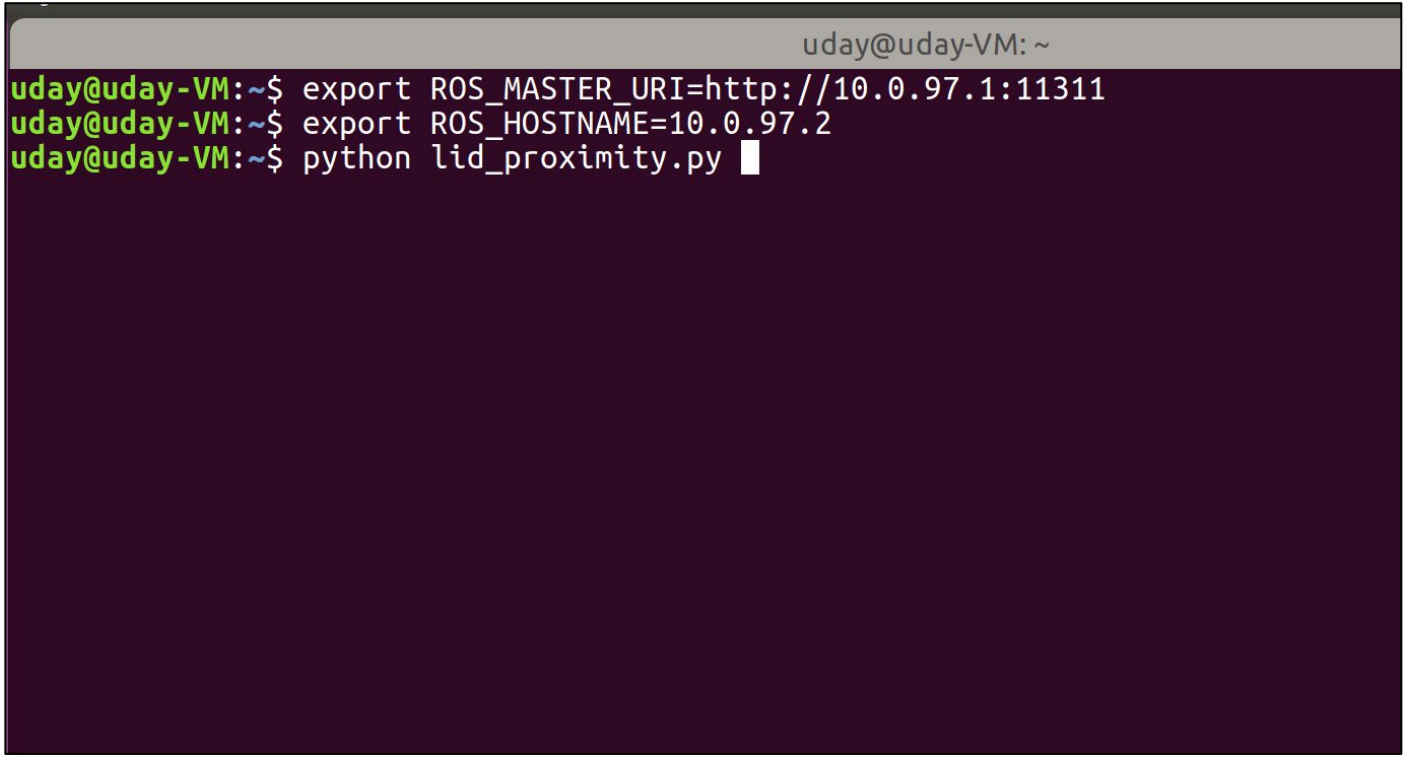
- To detect obstacles that are too close to the lidar, leading to a hardstop.
- For e.g: Dangling headlights and LIDAR shroud (in chk 3s and 4s) and broken chassis when hit by a forklift (chk 5)
- The python node outputs a rough angle at which the object causing hardstop is located.

Steps for running the script:

1. Paste this script in the home directory of your virtual environment/ Linux OS. https://6river.atlassian.net/wiki/spaces/3P/pages/edit-v2/3429629972
2. Turn on the chuck and take it to an empty space (no obstacle within 3 feet in front of chuck).
3. VPN into the chuck using: ‘chuckctl vpn -c <chuck_name>’
4. Copy the ROS_MASTER and ROS_HOSTNAME commands highlighted in green
5. In a new terminal, paste them and run the python script using: ‘Python lid_proximity.py’



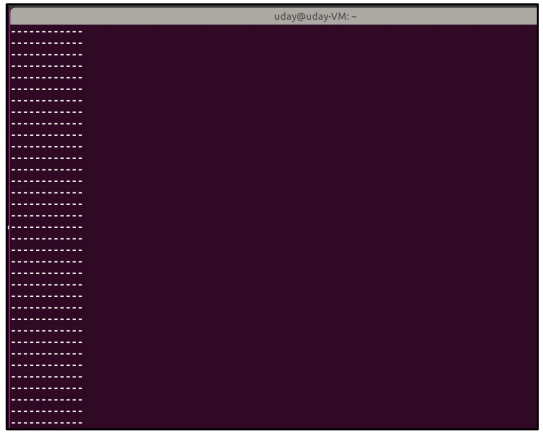
Snippet of the wiki

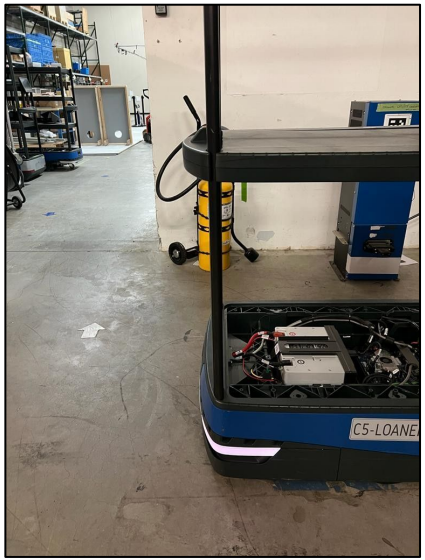


Second terminal

Interpreting the output:

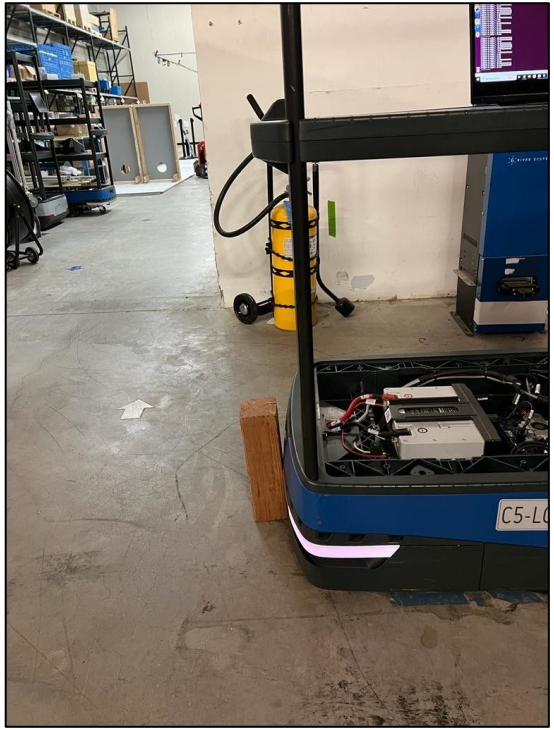
- The code will ask you about the case you are troubleshooting: enter **a** for the case of LIDAR shroud or **b** for the case of hanging lights. Enter your choice and observe the output.





Output when no obstacle

```
uday@uday-VM: ~
('Object at angle:', -1.6999999999999886)
('Object at angle:', 2.64)
('Object at angle:', 5.94)
('Object at angle:', 9.24)
('Object at angle:', 12.540000000000001)
('Object at angle:', 15.84)
('Object at angle:', 19.14)
('Object at angle:', 25.740000000000002)
('Object at angle:', 29.040000000000003)
('Object at angle:', 32.34)
-----------
('Object at angle:', -28.099999999999994)
('Object at angle:', -24.799999999999997)
('Object at angle:', -21.5)
('Object at angle:', -18.19999999999999)
('Object at angle:', -14.899999999999991)
('Object at angle:', -11.599999999999994)
('Object at angle:', -8.299999999999997)
('Object at angle:', -5.0)
('Object at angle:', -1.6999999999999886)
('Object at angle:', 2.64)
('Object at angle:', 5.94)
('Object at angle:', 9.24)
('Object at angle:', 12.540000000000001)
('Object at angle:', 15.84)
('Object at angle:', 19.14)
('Object at angle:', 22.44)
('Object at angle:', 25.740000000000002)
('Object at angle:', 29.040000000000003)
('Object at angle:', 32.34)
-----------
('Object at angle:', -28.099999999999994)
('Object at angle:', -24.799999999999997)
('Object at angle:', -21.5)
('Object at angle:', -18.19999999999999)
('Object at angle:', -14.899999999999991)
('Object at angle:', -11.599999999999994)
```



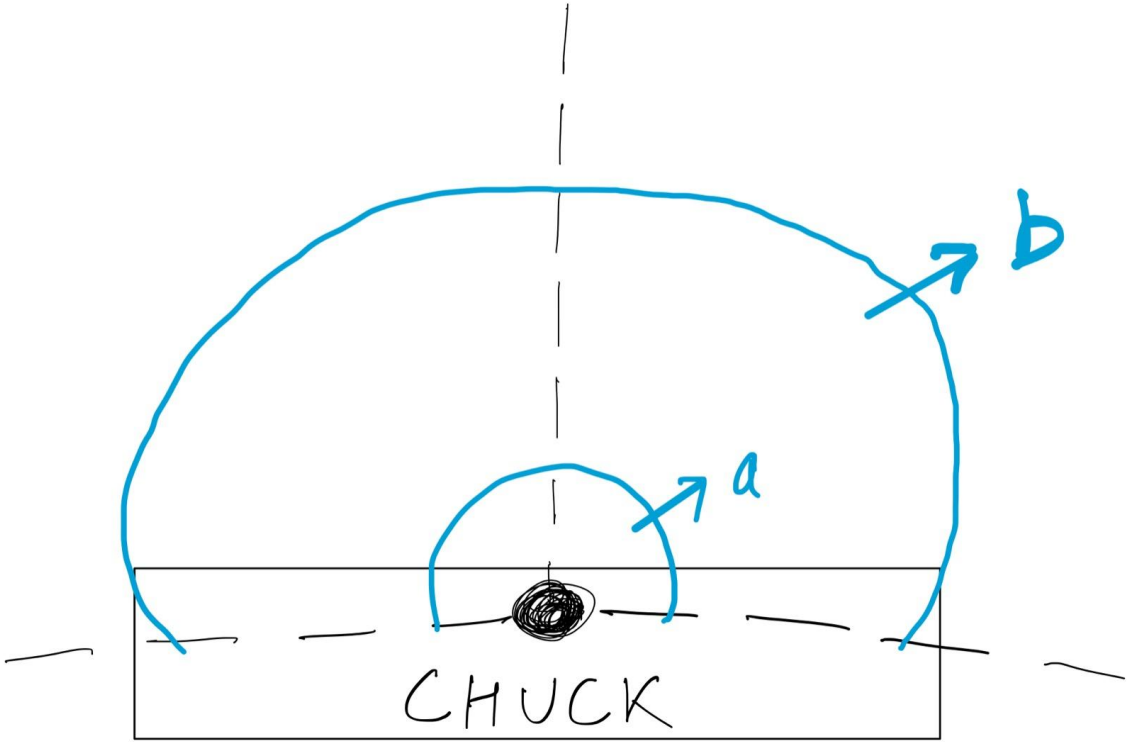
Obstacle right in front of chuck

Python script

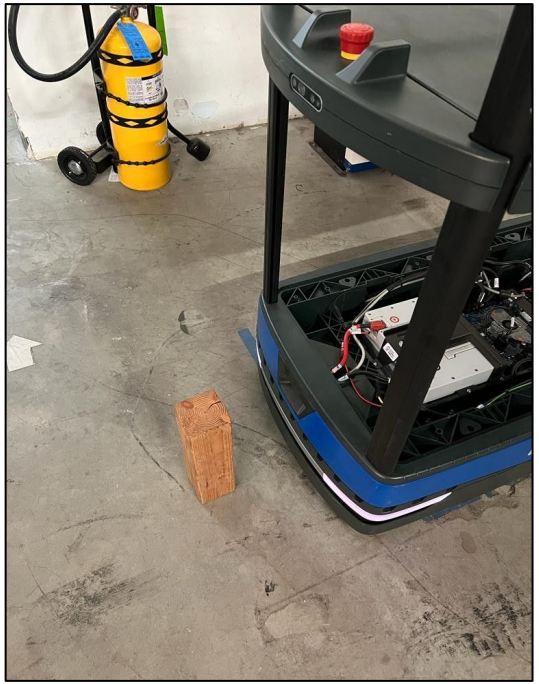
```
import rospy
from sensor_msgs.msg import LaserScan

print('\033[1;32m This program will help you debug a hardstop due to an object that is too close to LiDAR (shroud) \n or far away (hanging lights)]')


while True:
    inp=raw_input('\033[1;37m Are you debugging the case of shroud (a) or hanging lights (b), enter a or b: \n')
    if inp=='a':
        i=0.1
        break
    if inp=='b':
        i=0.28
        break
    else:
        print('Incorrect input, try again')
        continue

def callback(msg):

    dist_list=list(msg.ranges)
    # print(len(msg.ranges))
    for j in range(1,len(dist_list)-25,5):
        if dist_list[j]<i and dist_list[j]>0:

            # print(i)
            if j<313:
                angle= -104.33+ 0.33*j
                print('Object at angle:',angle)
            elif j>=313:
                k=j-313
                angle= 0.33*k
                print('Object at angle:', angle)

    print('------------')

rospy.init_node('scan_dist')
subs=rospy.Subscriber('/sensors/lidar/scan',LaserScan,callback)
rospy.spin()
```

Points to remember and false positives

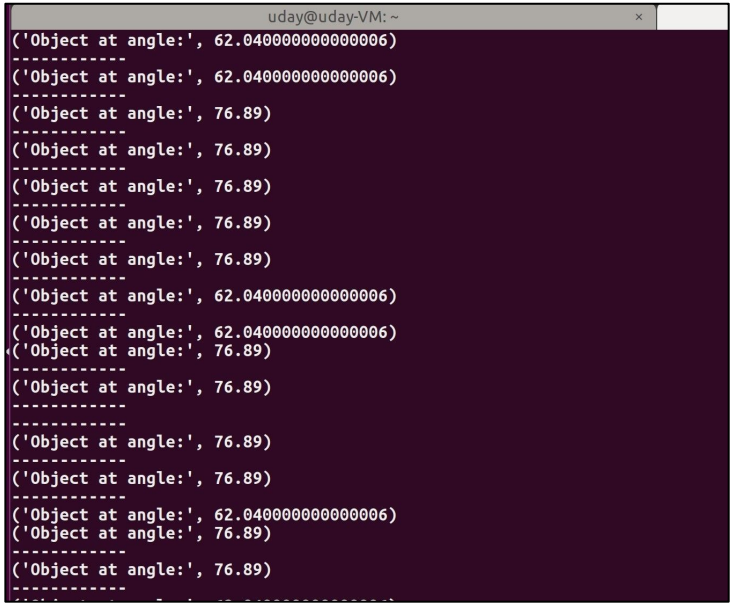
- Sometimes, one or two lines per message are displayed which can be ignored. These can be identified as being repetitive and display angle values that have large differences. If there is an obstacle, the output would be a range of angles
- Always make sure you give the correct input according to the case you are diagnosing. Using **(a)** as input and diagnosing the case for broken headlights would give incorrect results and same for **(b).**
- Having a good internet connection is necessary for accurate results.

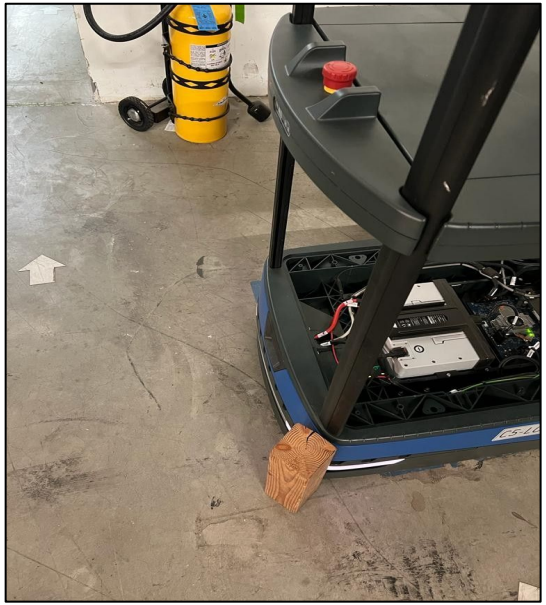


```
uday@uday-VM: ~
('Object at angle:', 5.94)
('Object at angle:', 76.89)
------------
('Object at angle:', -21.5)
('Object at angle:', -18.19999999999999)
('Object at angle:', -16.549999999999997)
('Object at angle:', -14.899999999999991)
('Object at angle:', -13.25)
('Object at angle:', -11.599999999999994)
('Object at angle:', -9.949999999999989)
('Object at angle:', -8.299999999999997)
('Object at angle:', -6.6499999999999915)
('Object at angle:', -5.0)
('Object at angle:', -3.3499999999999943)
('Object at angle:', -1.6999999999999886)
('Object at angle:', 0.99)
('Object at angle:', 2.64)
('Object at angle:', 4.29)
('Object at angle:', 5.94)
------------
('Object at angle:', -18.19999999999999)
('Object at angle:', -16.549999999999997)
('Object at angle:', -14.899999999999991)
('Object at angle:', -13.25)
('Object at angle:', -11.599999999999994)
('Object at angle:', -9.949999999999989)
('Object at angle:', -8.299999999999997)
('Object at angle:', -6.6499999999999915)
('Object at angle:', -5.0)
('Object at angle:', -3.3499999999999943)
('Object at angle:', -1.6999999999999886)
('Object at angle:', 0.99)
('Object at angle:', 2.64)
('Object at angle:', 4.29)
------------
('Object at angle:', -18.19999999999999)
```



False positive





False negative

Thank You!!