Lab Procedure for Simulink
Obstacle Detection

Setup

1. It is recommended that you review Lab 4 – Application Guide before starting this lab.

2. Launch Quanser Interactive Labs, scroll to the “QBot Platform” menu item and then select the “Warehouse” world.

3. Open the Simulink Model Obstacle_detection.slx, as shown in Figure 1. The loading of the Simulink model also triggers the setup script for QLab. When the script is run successfully, the QBot and the Environment should be spawn in QLab and User LEDs on the virtual QBot will turn white, as shown in Figure 2.

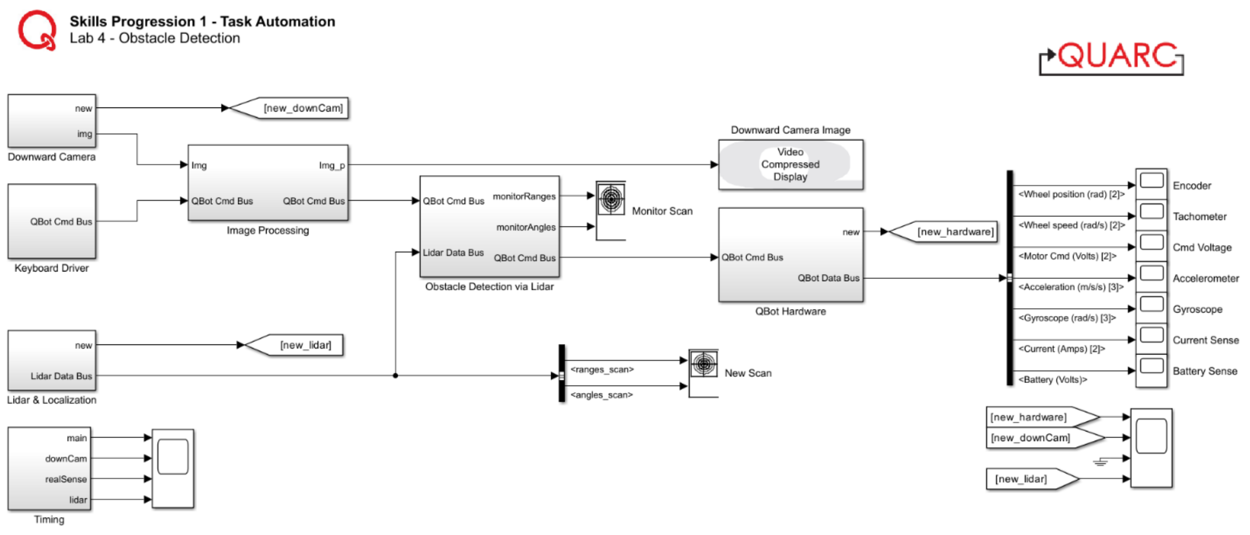

Figure 1. Obstacle Detection Simulink Model

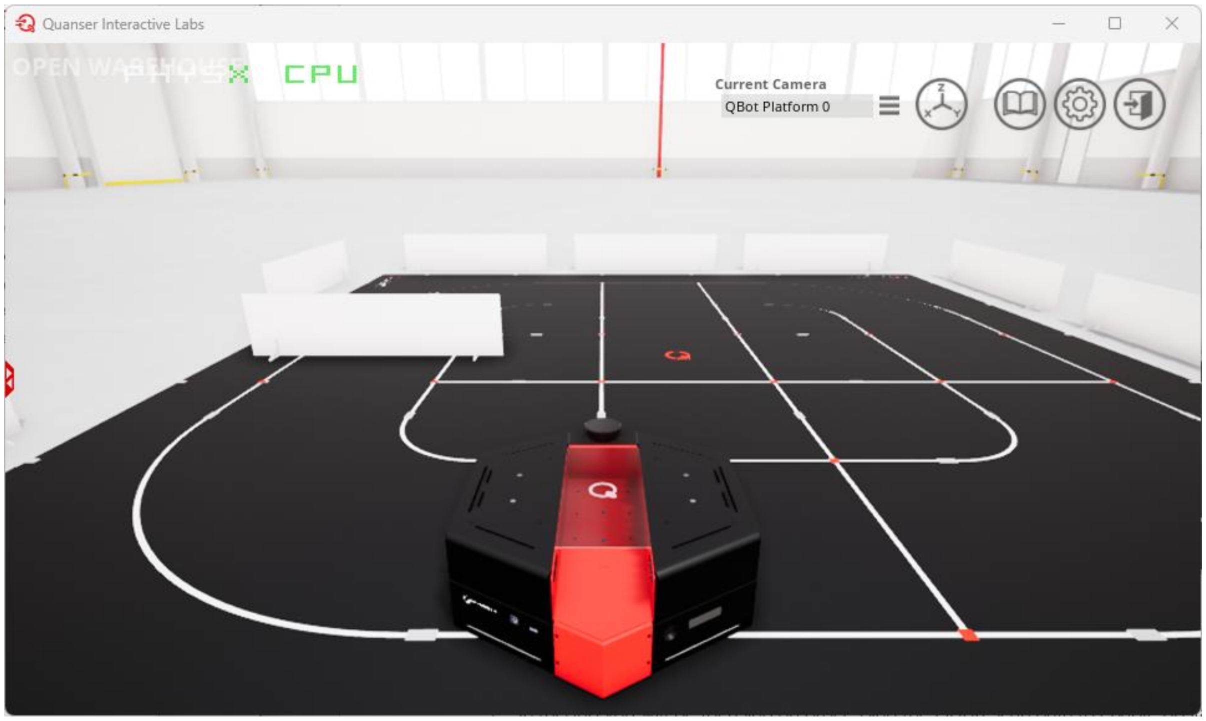

Figure 2. Successful set up of the Quanser Interactive Lab Workspace

LiDAR Data Localization

1. In this lab you will be focusing on processing the LiDAR scan data to enable obstacle detection of the QBot. First, we will take a look at the **Lidar and Localization** subsystem, it should look like figure 3. In this subsystem, the reference frame of the LiDAR data is transferred from the center of the sensor to the center of the QBot. The **Quanser Ranging Sensor** block outputs distances measured in meters and the corresponding headings. Other properties such as the standard deviation (sigma) and quality (qual) of the measurement were not used in our application. To learn more about the output of **Quanser Ranging Sensor** block, right click on the block and select "help" to go to its documentation.

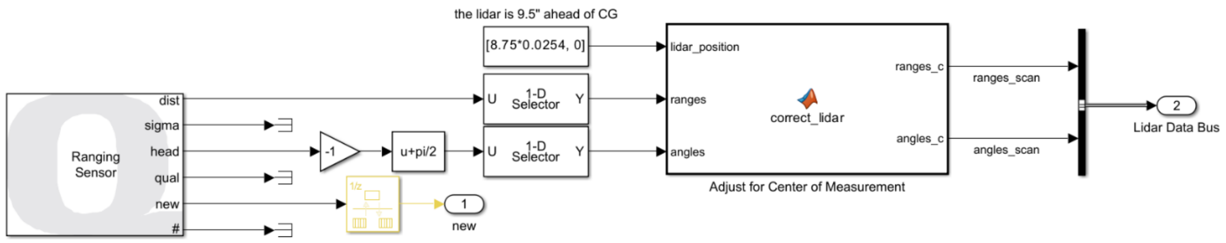

Figure 3. Lidar & Localization Block

2. Notice that the raw headings signal is connected to a **Gain** block with a value of -1 and a **Bias** block with a value of pi/2. This is to correct the headings such that the angle

value of 0 corresponds to the front of the QBot, as shown in Figure 4. In addition, the angles now are now increasing counterclockwise, consistent with the positive convention of our reference frame.

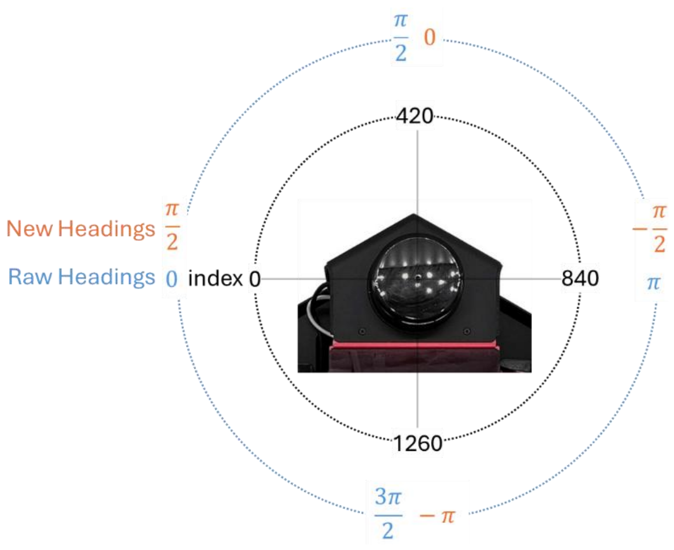

Figure 4. LiDAR measurement correction

3. Two **Selector** blocks are used to downsample the distance and headings signal, which is beneficial in the obstacle detection application as it allows for a faster computation when high resolution is not necessary.

4. Open the **MATLAB Function** block labeled **correct_lidar**. To complete this function, you will use the processed range and angle data and the position of the LiDAR to adjust the current center of measurement (LiDAR) to the center of the QBot.

Dynamic Monitor Region

1. Go back to the main Simulink model and open **Obstacle Detection via Lidar** subsystem, as shown in Figure 5. This subsystem uses the LiDAR measurement to determine when the QBot should stop to avoid collision with an obstacle. In this lab, you will implement a simple obstacle detection algorithm using the **MATLAB Function** block labeled **detect_obstacle**.

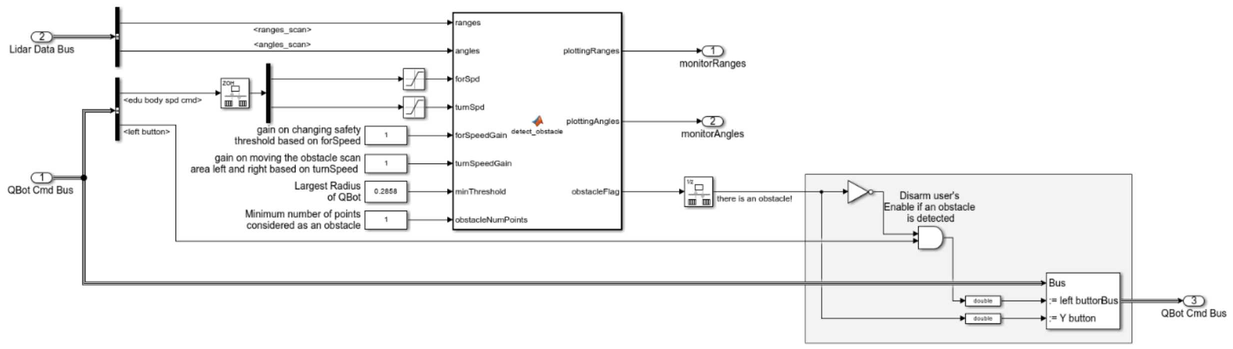

Figure 5. Obstacle Detection via Lidar block

2. Notice that in addition to the LiDAR measurements, QBot body speed commands are also used as input to the function. This is to allow our obstacle detection algorithm to dynamically change monitor region and safety threshold, as shown in Figure 6.

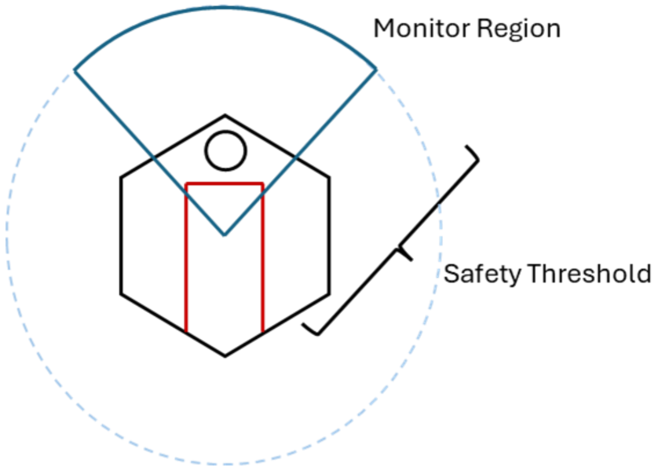

Figure 6. Dynamic Scanning

3. Open the **Saturation Blocks** leading into the *forSpd* and *turnSpd* ports of **detect_obstacle** block. Take notes of the upper and lower limits. They represent the maximum forward and turn speed of the QBot.

4. Open **detect_obstacle** block. In section 1, the variable *startingIndex* corresponds to the westmost angles of the monitor region. First, normalize *turnSpd* using the turn speed limit. Then, derive an equation for *startingIndex* in terms of normalized *turnSpd* and *turnSpeedGain* and complete section 1.

5. Similarly, normalized *forSpd* first, then derive an equation for *safetyThreshold* in terms of normalized *forSpd* and *forSpeedGain* and complete section 2.

6. Close the MATLAB function and go back to the main Simulink model. Verify that the **Image Processing** subsystem is already completed for you. However, feel free to

change the gains in the PD controller or copy over your own subsystem block from the previous lab.

7. Click Run under the Simulation Tab of your model to run the code. When the model is run successfully, the QBot Platform LEDs will turn blue.
8. Open the **Polar Figure** labeled **Monitor Scan**. This figure should display the LiDAR scan in the monitor region, as well as a uniform arc representing the safety threshold.
9. Without arming the QBot, move the movement keys around. Notice that the monitor region and safety threshold may not be changing with the body speed commands. This is most likely due to *forSpeedGain* and *turnSpeedGain* being too small to produce any noticeable responses.
10. Open **Obstacle Detection via Lidar** subsystem again to tune the two gains. Observe how the monitor region and safety threshold dynamically change in response to the varying speed commands. Iterate through this process as much as you need until you are satisfied with the monitor region responses. Press the "U" key to stop the model.

Obstacle Detection

1. Go back to the **Obstacle Detection via Lidar** subsystem. Notice that *minThreshold* has already been define for you. This variable represents the radius of the bounding circle of QBot.
2. Open **detect_obstacle** function. In section 3, the total number of points that lays between *minThreshold* and *safetyThreshold* will be computed and compared to *obstacleNumPoints*. When told number of obstacle point exceeds *obstacleNumPoints*, *obstacleFlag* will be set to true. Complete section 3 and close the function.
3. Click Run under the Simulation Tab of your model to run the code. When the model is run successfully, the QBot Platform LEDs will turn blue.
4. Without arming the QBot, gradually move an obstacle towards the LiDAR sensor to trigger *obstacleFlag*. When an obstacle is detected, the User LED will turn magenta.
5. How sensitive is your obstacle detection algorithm to obstacle? Does it trigger too early or too late?
6. Tune *obstacleNumPoints* until you are satisfied with the obstacle detection response.
7. While arming the QBot, drive it to a line on the mat, and press "7" key to start line following. Observe the obstacle detection response to different scenarios.
    a. The obstacle is directly on the line, as shown in figure 7a.

b. The obstacle is on the side of the line such that the QBot would narrowly drive pass the obstacle, as shown in figure 7b.

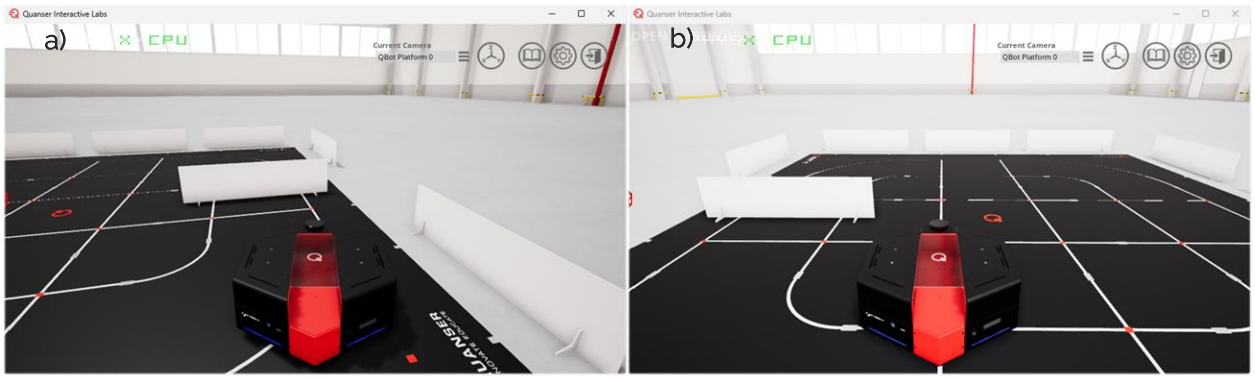

Figure 7. a) Obstacle directly on path; b) obstacle on the side of the path

8. Does the QBot stop in time to avoid collision? Does the QBot stop even if it could move pass the obstacle?
9. Can you move the QBot closer to obstacle after *obstacleFlag* is triggered?
10. Tune the *forSpeedGain*, *turnSpeedGain*, and *obstacleNumPoints* until the QBot can response correctly to the two obstacle configurations in step 7.
11. Stop the Simulink model when complete by pressing the "U" key. Ensure that you save a copy of your completed files for review later. Close Quanser Interactive Labs.