# Integrating RGBD Camera to TB3 using ROS – A step-by-step Guide.

**Objective**:  
This guide explains how to integrate an RGBD camera (OAK-D Pro) with the TurtleBot3 (Burger model) for static and dynamic map creation using sensor fusion. This system will allow the TurtleBot to perform autonomous navigation using LiDAR and RGB-D sensors.

**System Components**:

1. TurtleBot3 (Burger model)
2. RGBD Camera (OAK-D Pro)
3. LiDAR sensor (Mounted on TurtleBot3)
4. 2 Personal Computers (High-performance recommended)
5. 1 Portable PC

**Software Requirements**:

* ROS Noetic
* libopencv-dev
* python3-rosdep
* depthimage\_to\_laserscan
* ira\_laser\_tools

**Installation Links**:

* [ROS Noetic Installation Guide](https://wiki.ros.org/noetic/Installation/Ubuntu)
* [DepthImage to LaserScan](https://github.com/ros-perception/depthimage_to_laserscan)
* [Ira Laser Tools](https://github.com/iralabdisco/ira_laser_tools)

## Part 1: Static Map (Using LiDAR sensor to create maps and navigation)

This section assumes that the TurtleBot3 has been fully configured. For setup, refer to the [TurtleBot3 Quick Start Guide](https://emanual.robotis.com/docs/en/platform/turtlebot3/quick-start).

1. **Run roscore from the Master Laptop**

Open a terminal on the master laptop and run the following command to start ROS:

A computer screen with white text

Description automatically generated

Diagram 1.0

1. **Connect the Master laptop to the Turtlebot**

Ensure both the TurtleBot and the master laptop are on the same network. Use SSH to connect:****

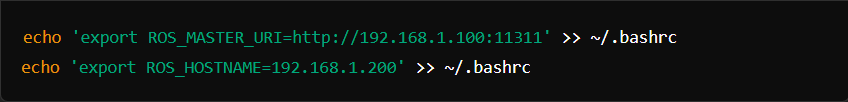
Diagram 2.0

Default username: ubuntu, password: turtlebot.

1. **Configure Network Settings**

Modify the .bashrc file on both the TurtleBot and the laptop. Append the following lines:

* + - * **Master Laptop:**



Replace 192.168.1.100 with the IP address of your ROS master and 192.168.1.200 with the IP address of your laptop.

* + - * **Turtlebot:**

A computer screen with numbers and letters

Description automatically generated

Replace 192.168.1.100 with the IP address of your ROS master and 192.168.1.200 with the IP address of your Turtlebot3.

Once done, apply the changes immediately by using the command ‘source ~/.bashrc’.

1. **Start TurtleBot3 Core**

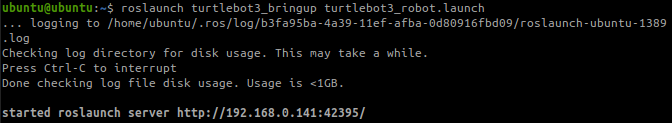
****

Diagram 3.0

* 1. When successfully done, the terminal will print these messages.

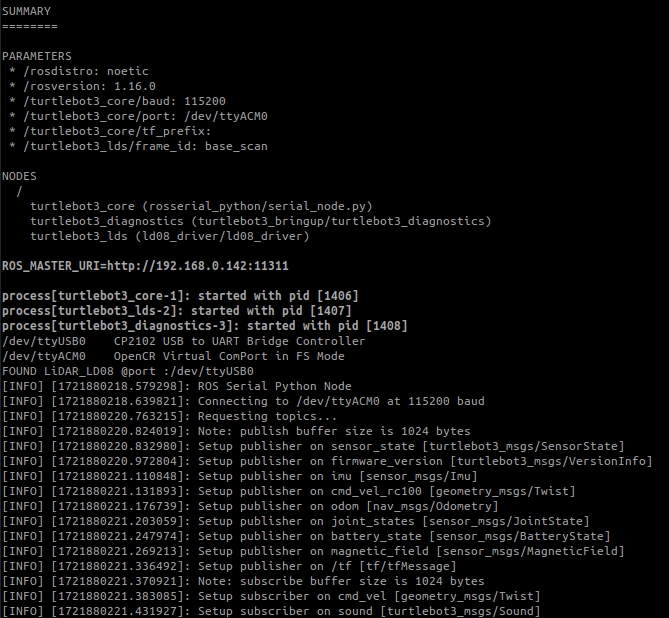


Diagram 3.1

1. **Save the TURTLEBOT3\_MODEL parameter permanently.**

**A black background with white text

Description automatically generated**

Diagram 4.0

1. **Launch SLAM Node:**
   1. Launch a SLAM package (default SLAM method will be gmapping) to generate a 2D occupancy grid map.

A computer screen with white text

Description automatically generated

Diagram 5.0

* 1. 3D visualization tool such as rviz will automatically open displaying the current surrounding around the TB3.

A screenshot of a computer

Description automatically generated

Diagram 5.1

1. **Run the teleoperation node from the Remote PC.**

**A computer screen with text

Description automatically generated**

Diagram 6.0

* 1. When successfully done, a virtual keyboard analog will appear to control the TurtleBot.

**A screenshot of a computer

Description automatically generated**

Diagram 6.1

1. **Collect information by exploring using the TurtleBot3.**
2. **Save the Static Map**:
   1. Once the mapping is complete, use the map\_saver node from the map\_server package to save the generated map as a .pgm and .yaml file for future use.

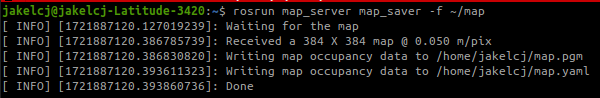


Diagram 8.0

A white object with holes in it

Description automatically generated

Diagram 8.1

A screenshot of a computer program

Description automatically generated

Diagram 8.2

1. **Autonomous Navigation of a Known Map with TurtleBot**:
   1. Test the navigation stack by loading the static map and observing how the TurtleBot3 navigates within the mapped area.
   2. Use RViz to visualize the map and verify its accuracy.

Steps:

* + 1. Launch the Navigation on RViz with the map saved using SLAM.



Diagram 9.0

* + 1. Once the map is opened, check whether the robot is correctly placed and whether the surroundings detected matched the map (There should be red light around the robot representing the objects).
       - If in case the robot did not detect any object near it, check the time synchronization between the Remote PC and the robot.
    2. To move the robot to the correct location, perform the Initial Pose Estimation by selecting the 2D Pose Estimation function and dragging the green arrow toward the direction where the robot is facing.
    3. You can also launch keyboard teleoperation node to precisely locate the robot on the map. Don’t forget to terminate the keyboard to prevent different cmd\_vel values from being published from multiple nodes during Navigation.
    4. Lastly, set the navigation goal by selecting the 2D Nav Goal function and dragging the red arrow toward the direction where you want the robot to go, and the robot will find the shortest path to reach the destination.

A screenshot of a computer

Description automatically generated

Diagram 9.1

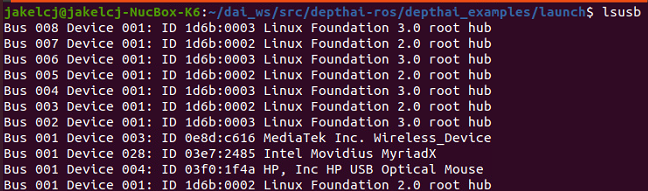
Part 2: Dynamic Map (Integrating Depth Camera for 3D Vision)

# To enable dynamic mapping using the RGBD camera (OAK-D Pro), we will connect the camera to a portable PC and integrate its depth data into the TurtleBot's navigation system.

# Setup and initial procedure

**Connect the Camera and the Portable PC**

* 1. Connect the camera to the Portable PC via USB cable.
  2. Ensure that the camera is successfully detected by the device by using the command ‘lsusb’.



**Install necessary dependencies and setting up the procedures.**

* 1. Once verified, go to <https://github.com/luxonis/depthai-ros> and scroll until you see the link to the newest documentation.
  2. It will bring you to luxonis page which specify the steps to setup DepthAI ROS.
  3. You will need to install from source by just following the steps given.
* If you are using Raspberry Pi, before you try to build your workspace by using the command ‘catkin\_make’, use the command ‘export MAKEFLAGS="-j4" ‘to specify the number of parallel jobs so that the Raspberry Pi won’t crash.
  1. Check if the environment and all the required files are properly built and all directory is available.

**Configure Network Settings between Master and Client.**

* 1. You will need to append the environment variable settings directly into the ~/.bashrc file of those devices.
     + Set ROS Master URI and Hostname:
       - For the Portable PC:

A computer screen with numbers and letters

Description automatically generated

Replace 192.168.1.100 with the IP address of your ROS master and 192.168.1.200 with the IP address of your Portable PC.

* 1. Once done, apply the changes immediately by using the command

‘source ~/.bashrc’.

# Camera Startup

**Launch the DepthAI camera node**

* 1. Run the roscore command from the ‘Master’ Laptop.
  2. In the Portable PC, you will need to source your ROS environment first by using the command ‘source ~/dai\_ws\_ws/devel/setup.bash’.
  3. Next, find the node to activate the camera functionality.

A screen shot of a computer screen

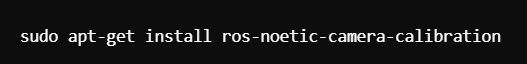
Description automatically generated

* 1. Launch the ‘stereo\_inertial\_node.launch’ file to gain depth image output.

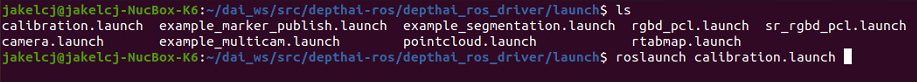


**Camera Calibration Using an 8x6 Checkerboard**

* 1. Install the Camera Calibration Package:
  + Make sure you have the necessary calibration package installed in ROS:

****

* 1. **Prepare the Checkerboard:**
  + **Print an 8x6 checkerboard pattern (8 corners horizontally, 6 vertically) on A3 paper. Ensure that:**
    1. **The checkerboard is flat and without any wrinkles.**
    2. **It is printed to scale without any resizing.**
    3. **Secure the checkerboard on a flat surface for calibration.**
* You can find a printable checkerboard here:  
  https://calib.io/pages/camera-calibration-pattern-generator
  1. **Configure the Camera Calibration Node:**
     1. Connect your camera to the system.
     2. Make sure the camera node is publishing the image topic (e.g., /camera/rgb/image\_raw).
     3. Launch the ROS camera driver to start publishing image data if not already running.
     4. Identify the image topic and camera info topic for your camera (typically /camera/rgb/image\_raw and /camera/rgb/camera\_info).
  2. **Launch the camera calibration node.**
  + Run the following command to start the camera calibration:



* + Ensure that:
    - --size 8x6 corresponds to the checkerboard pattern (8 corners horizontally, 6 vertically).
    - --square 0.025 indicates the size of each square in meters (adjust according to your checkerboard size).
  + Move the checkerboard around the camera's field of view. Ensure you cover different angles and distances until the calibration node indicates success.
  1. **Save and Commit the Calibration:**
  + Once calibration is completed, save the calibration parameters. You’ll be prompted to save the data to a .yaml file. This file will contain your camera's intrinsic parameters (camera matrix, distortion coefficients, etc.).



A computer screen shot of a computer

Description automatically generated

**Correct the Coordinate Transform for Related Sensors with the Robot**

* 1. **Open the Camera Launch File:**
     + Locate the launch file for your camera. This file usually defines the transformation between the camera frame and the robot's base frame.
     + Example: stereo\_inertial\_node.launch
  2. **Measure the Real-World Position of the Camera:**

• Position the Camera: Start with the camera mounted on your robot.

• Measure Offsets: Physically measure the camera's position relative

to the robot’s base.

* Measure the:

1. X (forward/backward distance),
2. Y (left/right distance), and
3. Z (height).

• Also measure the orientation of the camera in terms of roll,

pitch, and yaw.

* 1. **Update the Transformation Parameters:**
     + In your camera’s launch file or in a custom transform node, modify the x, y, z, roll, pitch, and yaw values according to your measurements.
  2. **Test and Verify the Transform:**
     + Launch your camera node and robot.
     + Use rviz to visualize the camera frame and ensure it aligns correctly with the robot's frame.
     + If necessary, adjust the x, y, z, roll, pitch, and yaw values until the camera is correctly positioned and oriented in the real world.
  3. **Save the Transform:**
* Once you are satisfied with the transform, ensure it is permanently saved in the launch file for future use.

# Create a Static Map using SLAM

**Please refer to the previous example.**

# First Layer (Displaying the Laser Scan onto the map)

**Build the Node from Source.**

* 1. Now you will need to clone the ‘depthimage\_to\_laserscan’ repository from the ROS GitHub in your Portable PC.

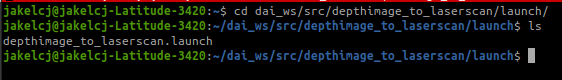


* 1. Then build the package again.

A black background with white text

Description automatically generated

* 1. Check if the ‘depthimage\_to\_laserscan’ node is available in the launch file.

****

* 1. Open depthimage\_to\_laserscan.launch file

A screen shot of a computer

Description automatically generated

* 1. Change the configuration to synchronize with the actual placement.

A screen shot of a computer code

Description automatically generated

**Merging the laser from the camera and LIDaR sensor.**

1. Now, in your ‘Master’ Laptop you will need to clone the ‘ira-laser-tools’ repository from the ROS GitHub.



1. Build the package and source your workspace.

A black background with white text

Description automatically generated

1. Open laserscan\_multi\_merger.launch file and change the configuration to synchronize with the actual placement.

A computer screen shot of white text

Description automatically generated

1. Launch the ‘laserscan\_multi\_merger.launch’ file.



1. Subcribe the right laser scan topic to subcribe to.

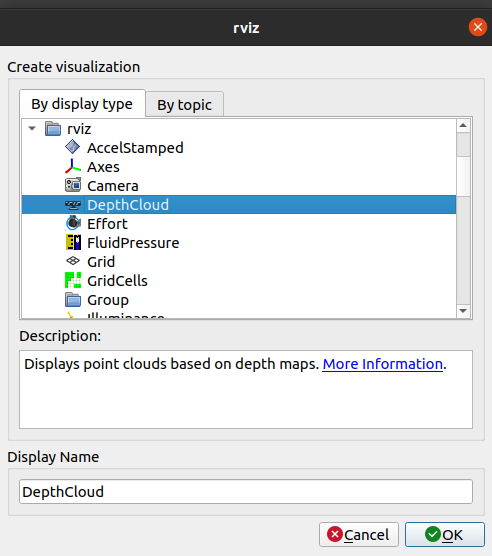
A screenshot of a computer

Description automatically generated

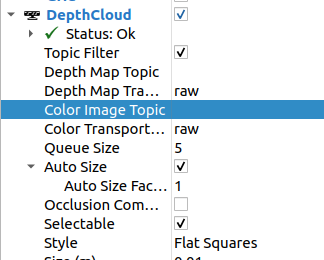
# Second Layer (Displaying the 3D images onto the map)

**Subscribe to the camera node.**

* 1. In the rviz click the button ‘add’ and add either DepthCloud or Pointcloud2 topic.



* 1. Click on the topic and choose the right image topic to subscribe to.



* 1. The image will be displayed on the map.

A screenshot of a computer

Description automatically generated

Autonomous Navigation of a Known Map with TurtleBot

**Please refer to the previous example.**

Configuring the costmap files

**Creating inflation area around the objects detected by the LaserScan.**

* 1. Locate the param directory for the turtlebot3\_navigation and you will find 3 main cost map file that you will need to configure.
     1. Local\_costmap\_params.yaml

A screen shot of a computer

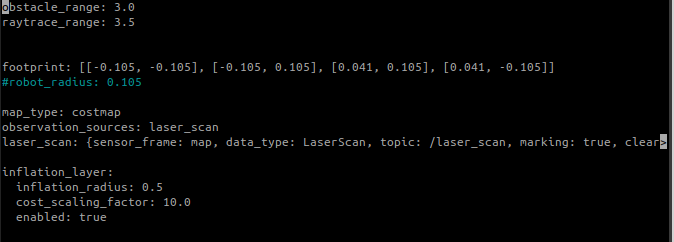
Description automatically generated

* + 1. Global\_costmap\_params.yaml

A screen shot of a computer program

Description automatically generated

* + 1. costmap\_common\_params\_burger.yaml



Results