Here’s a step-by-step “setup recipe” for a TurtleBot 4 running ROS 2 Humble on Ubuntu 22.04, tailored to everything our cube-seeker node needs:

**1. Base ROS 2 & tooling**

bash

CopyEdit

sudo apt update

sudo apt install -y \ build-essential \ python3-colcon-common-extensions \ python3-pip \

python3-vcstool \

git

Make sure you’ve installed ROS 2 Humble itself and sourced it in your shell:

bash

CopyEdit

sudo apt install -y ros-humble-desktop

echo 'source /opt/ros/humble/setup.bash' >> ~/.bashrc

source ~/.bashrc

**2. TurtleBot 4 bringup & navigation stacks slamtoolbox Tutorials**

sudo apt update

# TB4 hardware drivers + bringup

sudo apt install -y ros-humble-turtlebot4-bringup

# TB4 navigation (maps → Nav2 planners)

sudo apt install -y ros-humble-turtlebot4-navigation

# Nav2 core & bringup

sudo apt install -y ros-humble-navigation2 ros-humble-nav2-bringup

# SLAM for map building

sudo apt install -y ros-humble-slam-toolbox

# TurtleBot4 Python tutorials (includes TurtleBot4Navigator helper)

sudo apt install -y ros-humble-turtlebot4-tutorials

# CV support

sudo apt install -y ros-humble-cv-bridge python3-opencv

**5. Computer vision support**

# 1. Make sure the ROS 2 repositories are configured (only needs to be done once):

# https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debians.html

sudo apt update

# 2. Install the vision\_opencv stack (cv\_bridge + image\_geometry)

sudo apt install ros-humble-vision-opencv

# 3. Install image transport (for subscribing / publishing compressed images)

sudo apt install ros-humble-image-transport

# 4. Install the Python OpenCV bindings

sudo apt install python3-opencv.

# this should run without error

python3 - <<EOF

import cv\_bridge, cv2

print("cv\_bridge and OpenCV are available!")

EOF

**6. (Optional) Gazebo simulation**

# 1. Make sure ROS 2 repos are set up & Ubuntu universe is enabled

sudo add-apt-repository universe

sudo apt update

sudo apt update

sudo apt install -y \

ros-humble-turtlebot4-simulator \

ros-humble-irobot-create-nodes \

ros-humble-ros-ign-bridge

# 2. Install the TurtleBot 4 simulator and its HMI bridge

sudo apt install -y \

ros-humble-turtlebot4-simulator \

ros-humble-irobot-create-nodes \

ros-humble-gazebo-ros-pkgs

7. Run code in simulation:

sudo sh -c 'echo "deb [arch=$(dpkg --print-architecture)] \

http://packages.osrfoundation.org/gazebo/ubuntu-stable \

$(lsb\_release -cs) main" \

> /etc/apt/sources.list.d/gazebo-stable.list'

wget https://packages.osrfoundation.org/gazebo.key -O - \

| sudo apt-key add -

sudo apt update

sudo apt install -y \

ros-humble-turtlebot4-simulator \

ros-humble-ros-ign-bridge

ros2 launch turtlebot4\_ignition\_bringup ignition.launch.py

ros2 launch turtlebot4\_bringup robot.launch.py slam:=True use\_sim\_time:=True

ros2 run fielname

**7. Your workspace**

Finally, in your own ~/ros2\_ws:

bash

CopyEdit

cd ~/ros2\_ws

# clone your package (if you haven’t already)

# git clone <your-repo-url> src/tb4\_nav\_cube

rosdep install --from-paths src --ignore-src -r -y

colcon build --packages-select tb4\_nav\_cube

source install/setup.bash

**Quick verify**

After installing, confirm that:

bash

CopyEdit

ros2 pkg list | grep turtlebot4\_python\_tutorials

ros2 pkg list | grep slam\_toolbox

Both should appear. Then launch your stack:

bash

CopyEdit

# real robot

ros2 launch turtlebot4\_bringup robot.launch.py slam:=True use\_sim\_time:=False

# (or in simulation)

ros2 launch turtlebot4\_gazebo turtlebot4\_world.launch.py

ros2 launch turtlebot4\_bringup robot.launch.py slam:=True use\_sim\_time:=True

And in a separate terminal:

bash

CopyEdit

ros2 run tb4\_nav\_cube nav\_to\_cube

With those packages installed, your cube-seeker node will have everything it needs: SLAM, Nav2, obstacle avoidance, camera feeds, depth images, and the TB4 navigator helper.

**You said:**

the I should I execute this code after this on the bot

**ChatGPT said:**

Thought for 4 seconds

To run this on your real TurtleBot 4 (or in simulation), follow these steps **once** you’ve placed the updated nav\_to\_cube.py in your tb4\_nav\_cube package and adjusted setup.py (entry point) accordingly:

1. **Make sure your node script is executable**

bash

CopyEdit

cd ~/ros2\_ws/src/tb4\_nav\_cube/tb4\_nav\_cube

chmod +x nav\_to\_cube.py

1. **Build your workspace and source it**

bash

CopyEdit

cd ~/ros2\_ws

rosdep install --from-paths src --ignore-src -r -y # first time only

colcon build --packages-select tb4\_nav\_cube

source install/setup.bash

1. **Launch the TurtleBot 4 bringup (real or simulated)**
   * **Real hardware**:

bash

CopyEdit

ros2 launch turtlebot4\_bringup robot.launch.py slam:=True use\_sim\_time:=False

* + **Gazebo simulation**:

bash

CopyEdit

ros2 launch turtlebot4\_gazebo turtlebot4\_world.launch.py

ros2 launch turtlebot4\_bringup robot.launch.py slam:=True use\_sim\_time:=True

1. **Run your cube seeker node**  
   In a new terminal (remember to source ~/ros2\_ws/install/setup.bash if it’s not already):

bash

CopyEdit

ros2 run tb4\_nav\_cube nav\_to\_cube

Your robot will now:

* Record its current pose as “home”
* Wait for Nav2 and SLAM to spin up
* Explore to build out the map
* Step forward in increments until it sees the cube at a valid depth
* Compute the cube’s map‐frame location and drive there
* Return to the recorded home pose

If you hit any errors, double-check that:

* turtlebot4\_python\_tutorials is installed (so import TurtleBot4Navigator works)
* All required topics (/camera/color/image\_raw, /camera/aligned\_depth\_to\_color/image\_raw, /camera/color/camera\_info) are publishing
* Your package’s setup.py has the correct console\_scripts entry and you’ve rebuilt with co