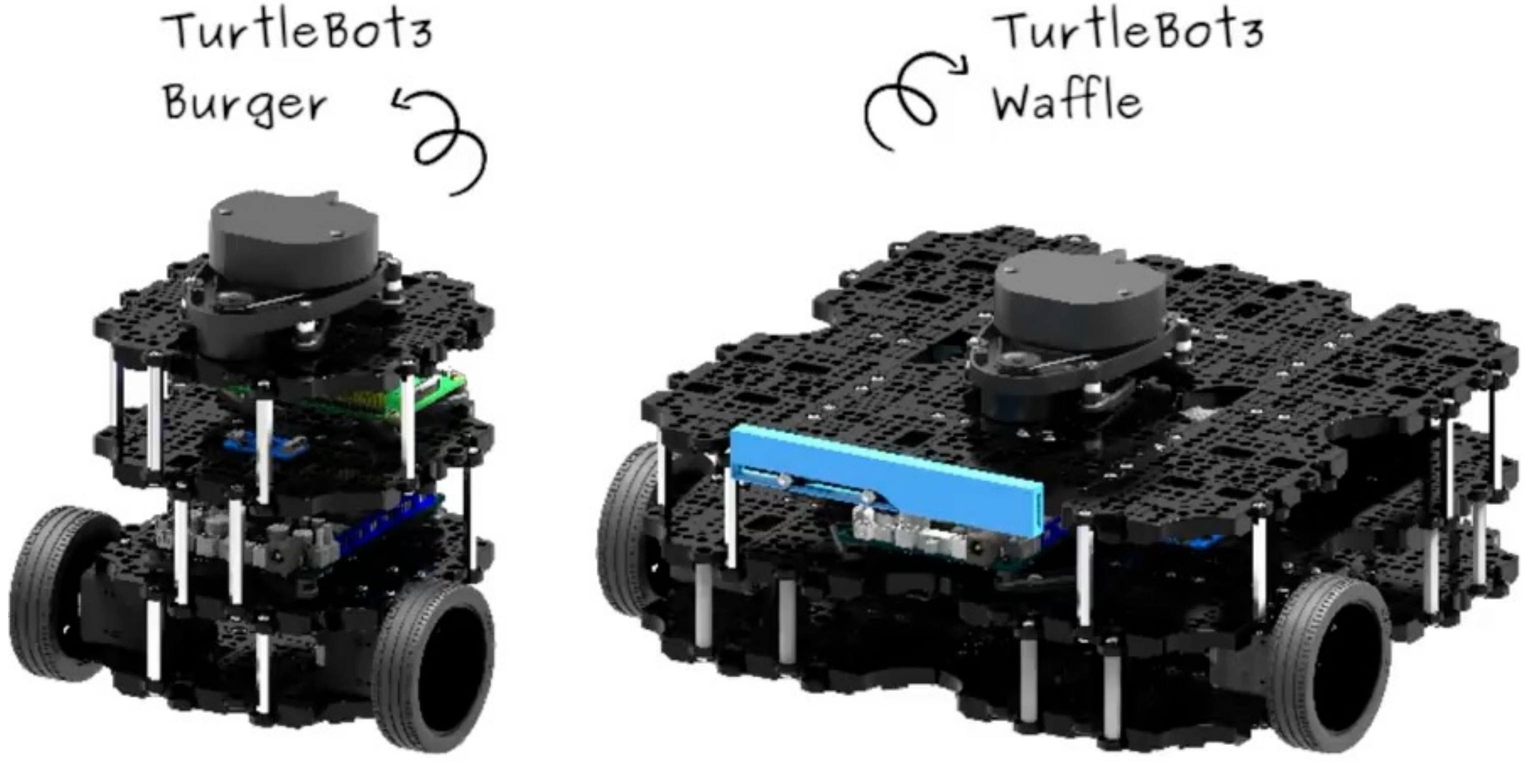
Lab 4: Person Following and Homing

- Part I: install and test TurtleBot3’s “follower” package.
- Part II: creates a ROS package that (a) detects a “pole” object, and (b) moves toward and stops in front of the pole object.



TURTLEBOT3

Follow Demo

Lab 4 – Part I: turtlebot3_follower

(https://github.com/ROBOTIS-GIT/turtlebot3_applications/tree/master/turtlebot3_follower)

- “follower” node: reads a lidar scan and computes a command for turtlebot3 to follow the leading person.
- “ploting”: loads LiDAR data from two files and trains a classifier that defines the leading person’s position as one of eight (8) cases. This is an independent Python program that does not use ROS, and you do not need to run it to use the follower node.
- “laser_subscriber”: works with “ploting” node to create training data for the person location classifer.

“ploting” is not a ROS program.

Python’s pickle class

- In Python, we work with high-level data structures such as lists, tuples, and sets. However, when we want to store these objects in memory, they need to be converted into a sequence of bytes that the computer can understand. This process is called **serialization**.
- The next time we want to access the same data structure, this sequence of bytes must be converted back into the high-level object in a process known as **deserialization**.
- To serialize an object hierarchy, you simply call the dumps() function.
- Similarly, to de-serialize a data stream, you call the loads() function.

How the follower node works …

1. initializes including defining a /cmd_vel publisher and loading a pre-trained person classifier.
2. Reads a LiDAR scan and classifies the observed person into one of eight possible poses (position and orientation), with an angular range of (-70°, 70°), where each class is formatted as “`dist_dir`”.
3. If a person is found (classifier returns a non-empty class), issues command to the robot by publishing a command to /cmd_vel, where the command is determined by the class label `dist_dir`.
4. If no person is detected (the case of an “empty” class), stops the robot.

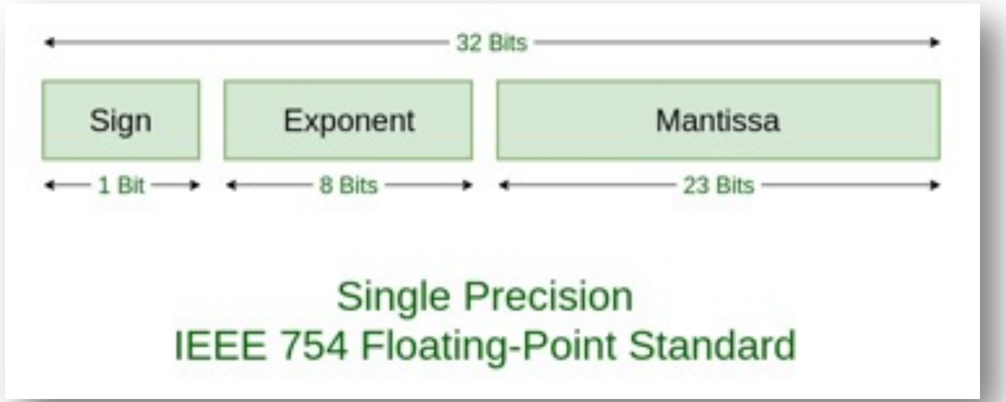
rospy.wait_for_message (“scan_filtered”, LaserScan)

- It (a) creates a new subscription to the topic, (b) receives one message, then (c) unsubscribes.
- In pseudo-code:
    1. creates a rospy.Subscriber
    2. initializes a flag to False
    3. waits for a single message, which
    4. triggers its callback to be called
    5. this callback sets the flag to True
    6. the True flag will cause the function to cancel its subscription, and
    7. returns the message it has received

Python’s np.nan_to_num( )

- Replace NaN with zero and infinity with large finite numbers (default behavior)
- NaN: not a number
- LaserScan:

```
…
float32 ranges[ ]
…
```



- In IEEE floating-point representation, NaN is encoded with e being all 1’s in $(-1)^{s} \bullet 1.f \bullet 2^{e\text{-}bias}$ format.

Lab 4 - Part 2: Homing

- Pole/Pillar detection
  - Assumes that pillar is the closest object, for simplification.
  - Rotates toward the pillar to indicate the pole/goal direction.

- Homing to the detected pillar
  - Implements Siegwart’s algorithm.
  - Could learn from the simple algorithm in the example below.

Refer to https://github.com/hnqiu/ros-ethz/blob/master/course3/course3.md for an example.