

## Purpose

In this project, you will use the robot's sensors to find a nearby wall, follow it by keeping it on the robot's right side. On reaching the next perpendicular wall, your robot must follow a path back, like mowing a lawn.

## Task

You must create a small Python script that publishes and subscribes to topics generating ROS2 messages to drive a simulated differential-drive robot (the iRobot Create 3). The robot's behavior must respond to simulated button presses as follows:

- When the “Power” button is pressed, the robot must begin execution. Execution consists of the following steps:
  1. The Create initiates a spiral search for the nearest wall (if done correctly it will find the docking wall).
  2. Upon finding the wall, the Create aligns itself parallel to it, keeping the wall on its right side.
  3. Once aligned, the Create translates forward until it detects the next perpendicular wall in counterclockwise (CCW) order (the lab wall containing the hall door).
  4. Upon detecting that wall, the robot:
    - (a) Reverses approximately 5 cm, then
    - (b) Executes a U-turn consisting of:
      - i. Rotating  $+\frac{\pi}{2}$  radians,
      - ii. Driving forward approximately 40 cm,
      - iii. Rotating another  $+\frac{\pi}{2}$  radians, and
      - iv. Resuming forward motion to begin a “boustrophedon” (back-and-forth) search pattern—like mowing a lawn.
- When the “Power” button is pressed **during any step** of execution, the robot must halt and reset to a state ready to **restart** execution. In other words, if the button is pressed twice during a run, the robot will restart the spiral search from the beginning.
- If the robot comes into contact with any object with its bump sensor, it must immediately stop, drive backwards  $\approx 5\text{cm}$ , and
  - If executing spiral search, begin aligning parallel.
  - If executing wall-following before the second wall is reached, attempt to recover and resume.
  - If executing end turn, attempt to recover and resume.
  - If executing second pass, resets i.e., ready to execute spiral search again.

**If the robot does not immediately stop and cease hitting any obstacle, a 20% penalty will be applied.**

## Deliverables

You may provide the following files and are required to provide `main.py`. These are the same as provided from:

```
boustrophedon/
+-- boustrophedon/
|   +-- main.py
|   +-- button_node.py
|   +-- hazard_node.py
|   +-- ir_node.py
|   +-- odom_node.py
```

You must provide a zipped archive of the files you intend to submit, even if that is just `main.py`. If you submit files other than `main.py`, ensure those files are correct. **If** resubmissions are considered, there will be a non-trivial penalty.

Any file (other than `main.py`) that you do not provide, will be provided for you.

## Points

Your code will be tested using the `create3_gazebo` simulation using the provided environment with the dock node deleted/removed.

The robot will be assigned a random initial position and orientation. The starting point will lie on or to the right of the line  $x = y$  ( $m = 1$ ) and on or to the left of the line  $x = 0.5$  ( $m = \text{undefined}$ ).

### Correctness

`boustrophedon/boustrophedon/main.py` 10%

+ 5% for correctly responding to a “Power Button” press.

+ 5% for correctly publishing Twist messages to `cmd_vel` in response.

-100% if it crashes; ever.

### Performance

Correctly spiraling into the wall on button press. 15%

+ 5% for initiating spiral motion.

+ 10% for detecting the correct wall and performing stop + backup maneuver.

Correctly halting and resetting on button press. 15%

+ 5% for halting.

+ 10% for resetting (verified by pressing the button again to re-initiate spiral behavior).

Correctly aligning with the wall. 20%

+ 10% for attempting alignment (within  $\pm 45^\circ$  of  $\theta = 0^\circ$ ).

+ 10% for completing alignment (within  $\pm 5^\circ$  of  $\theta = 0^\circ$ ).

Correctly following the wall. 20%

- + 10% for attempting wall-following (translates  $\approx 40$  cm without collision).
- + 10% for detecting the second wall and performing stop + backup maneuver.

Correctly performing U-turn at the second wall. 20%

- + 5% for attempting the U-turn (within  $\pm 45^\circ$  of  $\theta = \pi$ ).
- + 5% for completing the U-turn (within  $\pm 15^\circ$  of  $\theta = \pi$ ).
- + 10% for correctly returning along the intended boustrophedon path.

## Appendix

- README.md

Included are example commands for starting the simulation with different parameters.

To keep them copy/paste-able, they are provided as plain-text in this file.

- IR Sensor: Note that the IR sensor node as provided assumes that sensor returns:

[left, left-front, center-left-front, center, center-right-front, right-front, right]

The order of the Create 3. The simulation returns a different order. I would recommend looking into how to map them to the same ordering, *see boustrophedon/boustrophedon/ir\_node.py* for details.

- Stateful Design

It is highly recommended that you employ a stateful design including the states:

0. INITIALIZED
1. SPIRAL\_SEARCH
2. ALIGN\_WALL
3. FOLLOW\_WALL
4. MAKE\_UTURN
5. RETURN

This will allow you to respond to different sensor input in different ways.

- Align with wall

The goal here, essentially, is to maximize your right IR beam.

If you find the 5cm backup guide too far, you may reduce it. Be careful. If you do not back up enough and fast enough to turn off your bump sensor, you may have mixed results.

- Wall-following

You may use a PD-controller here to maintain a given IR intensity.