Lab 2: PID and Wall Following

CDA 4621
Fall 2024
Lab 2: PID and Wall Following
Due Date: 10-14-2024 by 11:59pm

Objective

This lab will teach you how to apply a PID controller to navigate parallel to a wall and stop at a desired distance from an end wall. The lab will also teach you about Lidars to measure distances to walls.

Requirements

Programming: Python
Robot: Webots 2023b (FAIRIS)

Robot Sensors Utilized

LIDAR

FAIRIS_Lite gives access to the LIDAR sensor mounted on the robot, which is an attribute of the `MyRobot` class and can be accessed via `lidar_sensor`. This sensor scans the surroundings of the robot, providing a full 360-degree range of measurements. Each LIDAR reading represents the distance from the robot to the nearest object at a specific angle, which helps detect obstacles and assist in navigation.

To retrieve the current LIDAR readings, use the function `robot.get_lidar_range_image()`. This function returns a list of 800 distance measurements. The first measurement corresponds to the distance directly behind the robot, the 200th measurement is from the left side, the 400th measurement is from the front, and the 600th measurement is from the right. Refer to Figure 1 for further details.

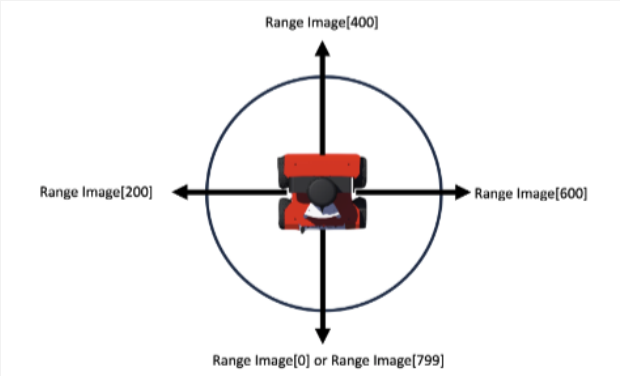


FIGURE 1. RosBot Lidar ranges in Webots

The LIDAR sensor object is a Python object created and maintained by Webots. For more details, see the Webots Docs. The LIDAR can return an array of distances for each angle, which can be processed to map the environment, avoid obstacles, or assist in path planning. More detailed documentation on how to use the LIDAR within FAIRIS_Lite can be found in the FAIRIS_Lite Docs.

PID

The PID controller will use the LIDAR sensor to control robot navigation. You will use the full PID controller to control motor velocities proportional to measured distances from walls, as shown in Figure 2 and Equations 1 to 4.

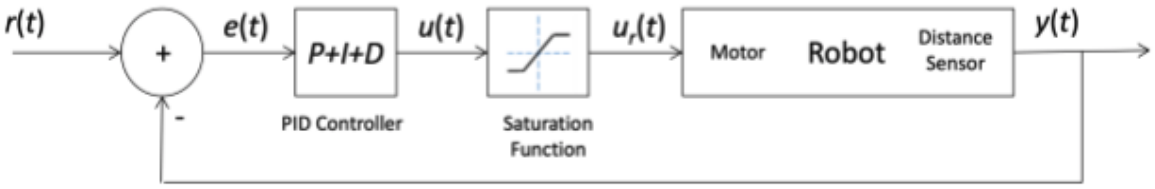


FIGURE 2. PID control of the robot velocity proportional to its desired distance to the wall.

The following equations summarize the PID controller:

$$e(t) = r(t) - y(t) \tag{1}$$

$$u(t) = K_p \cdot e(t) + K_i \cdot \sum_{0}^{t} e(t)\,dt + K_d \cdot \frac{e(t) - e(t-1)}{dt} \tag{2}$$

$$u_r(t) = f_{\text{Sat}}(u(t)) \tag{3}$$

$$f_{\text{Sat}}(u(t)) = \begin{cases} u_r(t) = c_{\max}, & \text{if } u(t) > c_{\max} \\ u_r(t) = u(t), & \text{if } c_{\min} \le u(t) \le c_{\max} \\ u_r(t) = c_{\min}, & \text{if } u(t) < c_{\min} \end{cases} \tag{4}$$

Where:

- $r(t)$ = desired distance to the wall
- $y(t)$ = current distance to the wall
- $e(t)$ = distance error
- K_p, K_i, K_d = gain constants
- $u(t)$ = motor control velocity
- $f_{\text{Sat}}(u(t))$ = motor control saturated velocity function
- $u_r(t)$ = motor control saturated velocity

PID (cont.)

Hint

$$u(t) = K_p \cdot e(t) + K_i \cdot \sum_{0}^{t} e(t)\,dt + K_d \cdot \frac{e(t) - e(t-1)}{dt} \tag{5}$$

To convert this into an iterative (or recursive) form, we break it down as follows:

- The proportional term is straightforward:
$$P(t) = K_p \cdot e(t)$$
- The integral term accumulates the error over time. In iterative form, we can approximate the integral using a running sum:
$$I(t) = I(t-1) + e(t) \cdot dt$$
So the integral part becomes:
$$K_i \cdot I(t)$$
- The derivative term estimates the rate of change of the error. In an iterative approach, we can approximate the derivative using the difference between the current error and the previous error:
$$D(t) = \frac{e(t) - e(t-1)}{dt}$$
So the derivative part becomes:
$$K_d \cdot D(t)$$

Therefore, the iterative or recursive update for $u(t)$ is given by:

$$u(t) = K_p \cdot e(t) + K_i \cdot I(t) + K_d \cdot D(t) \tag{6}$$

Where:

$$I(t) = I(t-1) + e(t) \cdot dt$$
$$D(t) = \frac{e(t) - e(t-1)}{dt}$$

Task 1: PID forward wall stop

The robot should use K_p, K_i, K_d PID "forward" gain constants for motion control, applying them only to the error values related to the forward motion. The control should be applied exclusively towards the robot's front motion, stopping 1m away from the end wall, and should not influence side motions. The robot must avoid hitting walls. Note that you will need to use the LIDAR for this task. The robot should start 6.5m away from the end wall, as shown in Figure 3.

For this task, you must load the maze in the maze2.xml file. This can be achieved by using the following line in your controller `maze_file = '../../worlds/Fall24/maze2.xml'`. Additionally, assume that $dt = 0.032$, representing the duration of a single timestep in Webots. Test your solution by experimenting with the following six values for the gain constants: $0.001, 0.01, 0.5, 1.0, 5.0, 10.0$. Apply these values separately to K_p, K_i, and K_d. Your goal is to find the optimal combination of gain constants (K_p, K_i, and K_d) that provides the best performance. These may differ from the previously suggested values. The robot should approach the wall in front, gradually slow down, and rest 1 meter away from the wall. Test the task starting at different distances from the end wall by clicking and dragging the

Task 1: PID forward wall stop (cont.)

robot. Ensure that if the robot is placed closer than 1 meter from the wall, the robot should back up.

During the test, continuously display the distance measurements from the end wall using sensor readings. For the video submission, select the best gain constant and record the robot moving forward until it stops 1 meter from the wall. Additionally, demonstrate what happens if the robot is manually dragged to a position less than 1 meter from the wall.

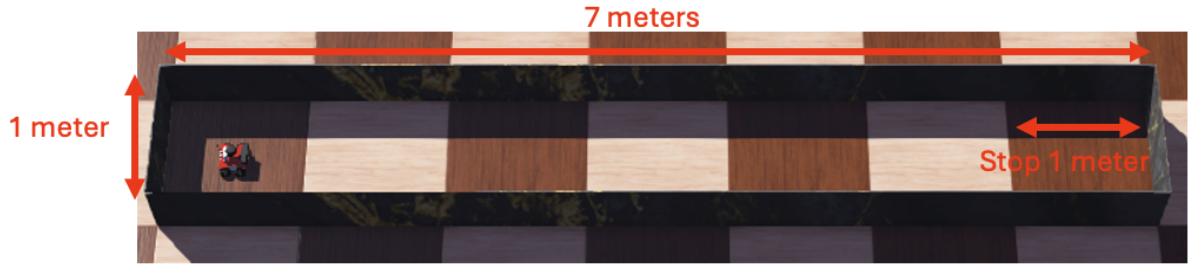


FIGURE 3. Task 1 world setup. This maze can be loaded in Webots by using the maze2.xml file. Note that in Webots, each colored square measures $1 \times 1m^2$ width.

Task 2: Wall following

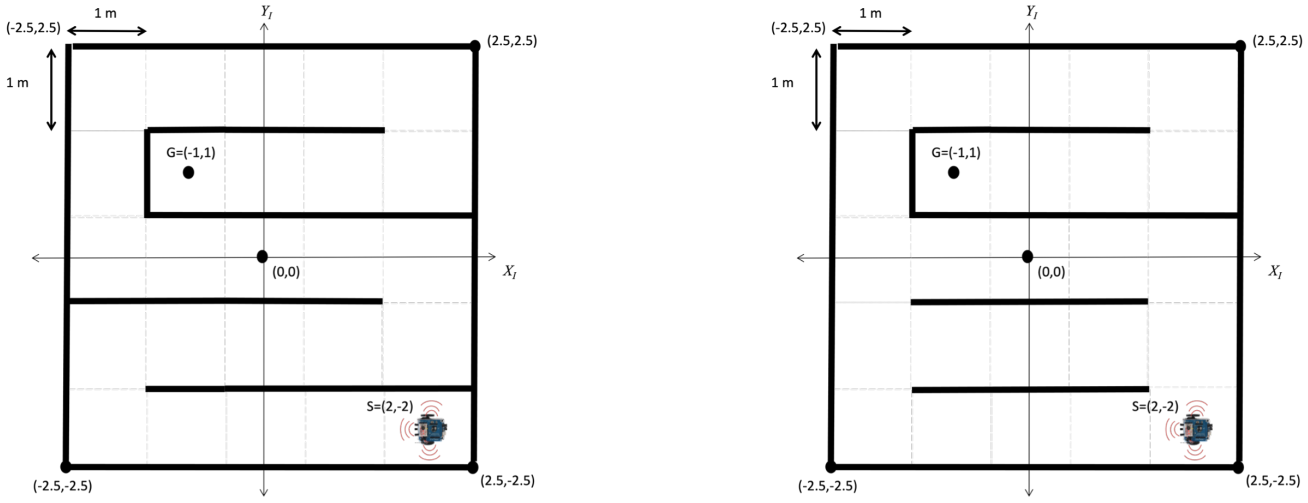
Perform robot wall following on the two mazes: `maze3.xml` and `maze4.xml`. The robot will need to follow either the left or right wall, depending on the specified mode. The task is as follows:

- The robot should perform wall following by either sticking to the left or right wall. For example, in left wall following, the robot must maintain contact with the left wall, navigating by treating the left side as if it is always "touching" the wall.
- When the robot encounters a wall directly in front of it, it should rotate 90 degrees and continue following the same wall (left or right) without switching sides.
- If the wall the robot is following ends abruptly (such as at a corner), the robot should wrap around the sharp edge and continue following the same wall.
- The robot should not switch between walls and should maintain consistent left or right wall following throughout the task.
- The robot needs to complete the wall following for both mazes: `maze3.xml` and `maze4.xml` while adhering to the wall following strategy.
- Some combinations of maze and wall-following modes may result in the robot being unable to reach the goal. For example, there may be cases where following the left wall prevents the robot from ever reaching the goal. If this happens, identify which maze and wall-following combination is impossible in your lab report and explain why the goal in unreachable.
- Compute and have the robot navigate both mazes using both left and right wall following. The robot should predict its current pose (x, y, θ) using the encoders and IMU throughout the navigation for valid combinations. The robot must determine if it has reached the goal position $G = (-1.0, 1.0)$ by checking if its predicted position is within $\pm 0.5m$ of the goal and come to a stop when this condition is met.
- Report the minimal travel time T from the start position $S = (2.0, -2.0, \pi)$ to the goal in maze3.xml for each valid combination of wall following. (10 points will be awarded to the robot navigating at minimum travel time in the class. Requires corresponding analysis in the report.)

Task 2: Wall following (cont.)

Hint

Although not required, consider using an additional PID controller that utilizes the side distances to modify the robot's angular velocity. This will help it maintain a consistent distance from the wall being followed. If done correctly, this will also allow the robot to curve around sharp turns smoothly without the need for hardcoded methods.



(I) maze3.xml (II) maze4.xml

FIGURE 4. These are the two mazes you will test your wall following solutions.

Code Evaluation Task 1 (30%)

Task 1: PID Forward Wall Stop (30 points total)

Robot stabilizes at specified distance from end wall (15 points)

- **Full Marks (15-13 points):** The robot reliably stabilizes exactly 1 meter from the end wall, coming to a steady stop. There is minimal to no oscillation, and the robot does not overshoot.
- **Partial Marks (12-8 points):** The robot stabilizes near 1 meter from the end wall, but there may be slight overshooting or oscillation. It eventually corrects and stops within an acceptable range.
- **Minimal Marks (7-4 points):** The robot struggles to maintain the correct distance from the end wall, frequently overshooting or stopping too early. Stabilization is inconsistent.
- **No Marks (3-0 points):** The robot does not stabilize at the specified distance or fails to stop altogether.

Robot utilizes PID to reduce speeds proportionally to the front wall (10 points)

- **Full Marks (10-9 points):** The robot reduces its speed as it approaches the wall, using PID control to ensure a smooth deceleration. The motion is proportional to the remaining distance.
- **Partial Marks (8-6 points):** The robot reduces its speed, but the deceleration is either too fast or too slow, with noticeable inconsistencies in PID control.
- **Minimal Marks (5-3 points):** The robot reduces its speed, but there is little to no evidence of smooth PID control. The robot slows down too late or too quickly.
- **No Marks (2-0 points):** The robot does not effectively utilize PID control to adjust its speed, or it does not decelerate proportionally.

Robot restabilizes when placed further or closer to the wall (5 points)

- **Full Marks (5 points):** The robot successfully restabilizes when manually moved closer or farther from the wall, adjusting smoothly back to 1 meter.
- **Partial Marks (4-3 points):** The robot restabilizes when moved, but it may take too long or overshoot before returning to the correct distance.
- **Minimal Marks (2-1 points):** The robot struggles to restabilize and only sometimes returns to the correct distance.
- **No Marks (0 points):** The robot fails to restabilize after being moved.

Robot hits walls (-5 points)

Deduction: If the robot hits any walls during the task, 5 points will be deducted from the overall task score.

Code Evaluation Task 2 (60%)

Task 2: Wall Following (60 points total)

Wall Following on Maze 3 (30 points)

- **Full Marks (30-25 points):** The robot follows the correct wall (left or right) in the maze with minimal deviation and successfully navigates through to the goal without hitting walls or switching sides.
- **Partial Marks (24-18 points):** The robot follows the correct wall but deviates at certain points, requiring occasional corrections to stay on course. The robot completes the maze but may brush against walls.
- **Minimal Marks (17-10 points):** The robot follows the wall inconsistently, switching sides or struggling to maintain the wall-following behavior. The robot does not always reach the goal successfully.
- **No Marks (9-0 points):** The robot fails to follow the wall or does not complete the maze.

Wall Following on Maze 4 (30 points)

- **Full Marks (30-25 points):** The robot follows the correct wall in the maze with minimal deviation and successfully navigates through to the goal without hitting walls or switching sides.
- **Partial Marks (24-18 points):** The robot follows the correct wall but deviates at certain points, requiring occasional corrections to stay on course. The robot completes the maze but may brush against walls.
- **Minimal Marks (17-10 points):** The robot follows the wall inconsistently, switching sides or struggling to maintain the wall-following behavior. The robot does not always reach the goal successfully.
- **No Marks (9-0 points):** The robot fails to follow the wall or does not complete the maze.

Additional Deductions for Task 2:

- Robot average travel speed less than .1 meter per second except for corners (-5 points)
- Robot constantly exceeds .4 meters closer or further away from any wall (-5 points)
- Robot hits walls (-5 points)

Report & Video Evaluation (10%)

A project submitted without a report, or without including the corresponding task video, will not be graded and will receive a "0" for the complete lab.

Report Sections: The report should include the following (points will be deducted if anything is missing):

- **Mathematical Computations:** Show how you calculated the speeds of the left and right servos given the input parameters for each task.
- **Conclusions:** Analyze any issues encountered when running the tasks and discuss how these could be improved. Discuss your results, including any navigation errors. Conclusions need to reflect an insight into what was learned during the project. Statements like "everything worked as expected" or "I enjoyed the project" will not count as valid conclusions.
- **Video:** Upload the video to Canvas showing the robot executing the task (details below).
- **Authorship Statement:** Include and sign the following statement at the end of the report:

  "I developed all the code and written report with the following exceptions:
  [Student name: signature, date]
  [Code sections (specify lines): references and additional comments]
  If any external tools were used to generate code, you need to include the prompt and output."

Task 1 Video: Record a single video of the robot performing PID forward wall stop on the maze2.xml file. The video should show the best choice for gain constant and forward distance being printed. Be sure to show the robot coming to a steady state once 1 meter away from the front wall and what would happen if place too close to the wall.

Task 2 Video: Record a single video showcasing the robot performing both left and right wall following on both `maze3.xml` and `maze4.xml`. The video should demonstrate how the robot navigates each maze, following the selected wall consistently, and the printouts of the robot pose estimation during navigation. Ensure that the video highlights successful completion of both wall-following strategies in both mazes, including any adjustments the robot makes when encountering walls or corners. For each combination, show the robot reaching the goal and stopping within the required $\pm 0.5m$ accuracy. If any combination fails to reach the goal, explain why in the video.

Lab Submission

For Lab 2 there are two separate Canvas Assignments: Lab 2 Code Submission and Lab 2 Report Submission. Please follow the Submission policy outlined below. **Note:** Failure to adhere to the submission policy will result in a reduction of 20 points.

Code Submission Policy

Students will need to upload a Python file (i.e. Webots controller) for each Task. The Python file should follow this naming convention *USF_ID_LabX_TaskY.py*, where *X* is the lab number and *Y* is the task number. Example *rockybull5_Lab1_Task1.py*. Additionally, if a student has created any functions that are not present in the controller but rather the *MyRobot.py*, the student will also need to upload that python file as well using the following naming convention *USF_ID_MyRobot.py*.

If custom functions are used in a student's submission and they are not in the controller or the MyRobot Python files, an automatic Zero will be assigned for the code submission.

Each file should be uploaded to the same submission. **DO NOT** upload a zip file, Canvas allows multiple uploads per submission. Failure to meet these submission policies will result in 20 points off.

Report Submission Policy

Students will need to upload their videos to their USF OneDrive or USF Sharepoint and generate a shareable link. Please ensure that you make it viewable to everyone or at the very least the TA and the Professor (use their emails to add them to the access list).

Students will need to upload a PDF file of their lab report. Be sure that the lab report contains the link to the videos and any Metrics requested by the Lab. These Metrics will be used to determine which student receives the extra credit. Use the naming convention *USF_ID_LabX_Report.pdf*. Only PDFs will be accepted.

If the student used ChatGPT to generate code they will also need to provide the conversation used to generate the code. This can either consist of screenshots of the conversation or a shareable link that the system provides. This should be a separate document and not part of the report.

Each file should be uploaded to the same submission. **DO NOT** upload a zip file, Canvas allows multiple uploads per submission. Failure to meet these submission policies will result in 20 points off.

TA may ask you additional questions to gauge your understanding via Canvas Message or MS Teams. Failure to reply may result in point deduction.