# Control of Mobile Robotics
# CDA4621
# Spring 2023
# Lab 3
# Navigation with Distance Sensors
# Total: 100 points

### Due Date: 2-27-2023 by 11:59pm

The assignment is organized according to the following sections: (A) Requirements, (B) Objective, (C) Task Description, (D) Task Evaluation, and (E) Lab Submission.

## A. Requirements

### A.1 Physical Robot

**Robot:** "Robobulls-2018". **Programming**: Python
**Basic Files**: tof.py ("SamplePrograms.zip")

### A.2 Robot Simulator

**Simulator:** Webots. **Robot**: "e-puck". **Programming**: Python
**Basic Files**: "Lab3_epuck.zip"

## B. Objective

This lab will teach you about (1) Distance Sensors, (2) Lidars, and (3) PID. You will learn how to apply a PID controller to navigate and stop at certain distance from a wall.

### B.1 Physical Robot

"Robobulls-2018" uses Adafruit VL53L0X Time of Flight (ToF) Distance Sensor[1].

#### B.1.1 Distance Sensors

There are 3 distance sensors configured in the robot: left, front, and right. Note that in general, measurements change depending on type of surface and reading angles.

#### B.1.2 Lidar

The current robot configuration does not include a Lidar.

### B.2 Robot Simulator

Figure 1 shows the configuration of distance sensors and Lidar in the e-puck robot. Note the different positioning of distance sensors and Lidar in the robot. You will need to take into consideration these offsets in sensor positioning.

#### B.2.1 Distance Sensors

Webot's e-puck uses a generic distance sensor having a single ray. The e-puck robot design used for the labs includes 3 distance sensors (an additional rear sensor has also been added in case needed). Each distance sensor is defined by a lookup table for measuring range and noise level[2]:

[1] https://learn.adafruit.com/adafruit-vl53l0x-micro-lidar-distance-sensor-breakout/downloads
[2] https://cyberbotics.com/doc/reference/distancesensor

```
lookupTable [ 0 0 0, 1 1 0.1]
```

The above lookup table consists of two rows of 3 numbers each, where first number represents a distance $d_i$, second number represents a corresponding distance value $v_i$, and third value represents a noise level $n_i$. Thus, combined two rows define a distance range, $d_i$ (row i) and $d_{i+1}$ (row i+1), i.e., 0 to 1 meter; a corresponding return value range, $v_i$ (row i) and $v_{i+1}$ (row i+1), i.e., 0 to 1 value (in this case matching exact distance range values), and noise level, i.e., 0.1 (10%).

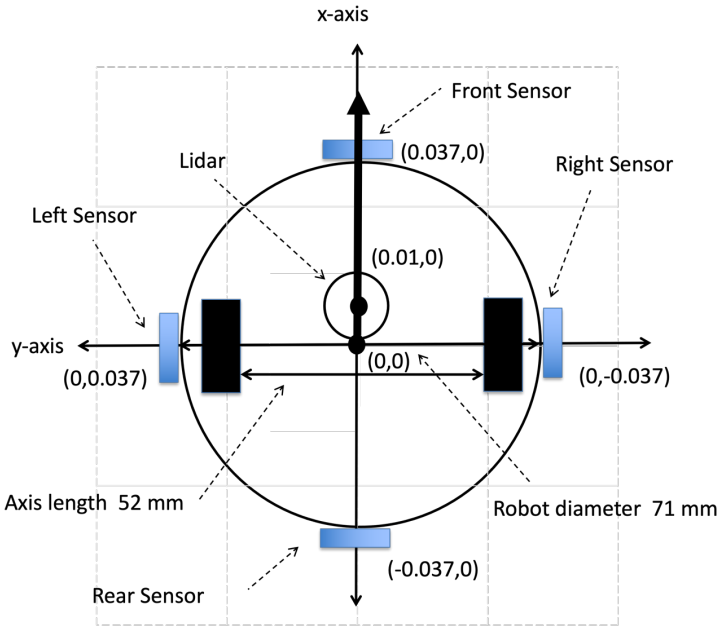


Figure 1. E-puck robot distance sensor and Lidar configuration in Webots.

### B.2.2 Lidars

A Lidar ("Light Detection and Ranging") is shown in Figure 2. The Lidar is a range sensor that sends multiple laser beams (pulsed light waves) to detect distances at different orientations (the traditional distance sensor sends a single beam). The Lidar computes the time it takes each pulse to return to the sensor to calculate distances at each orientation. A Lidar has been added to the Webots robot[3].

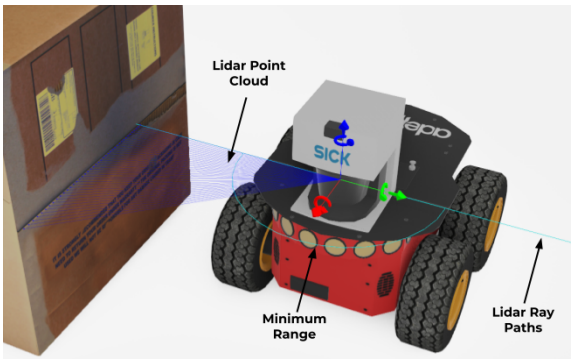


Figure 2. Typical Lidar configuration.

The max range of the Webots Lidar is set to 1 meter, with a 0.01 (1%) noise level. The Lidar is set to a single layer, with 1-deg resolution, and a 360-degree field of view. It will capture 360 measurements (one for every degree) at every time step. A list is of ranges is returned as follows:

```
d = lidar.getRangeImage()[i]
```

[3] https://cyberbotics.com/doc/reference/lidar

The functions returns a 360-element list. Each distance *d* is obtained by *i* from 0 to 359. For example, the [0] element corresponds to the front of the robot, [90] element to the left side, etc.

### B.3 PID

You will program a PID controller to control distances to walls for each of the 3 sensors. You will use only the "P" component. Note that distances to walls will be indirectly controlled by controlling motor velocities, as shown in Figure 3. Velocities should be set proportional to distance error.

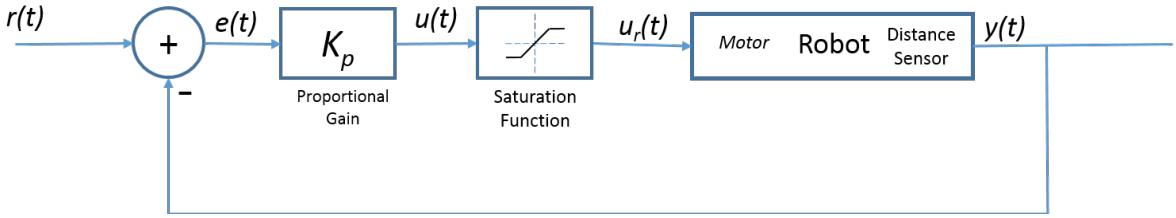


Figure 3. PID control of the robot velocity proportional to its desired distance to the wall.

The following equations summarize the diagram in Figure 2:

$$e(t) = r(t) - y(t) \quad \text{(Eq. 1)}$$
$$u(t) = K_p * e(t) \quad \text{(Eq. 2)}$$
$$u_r(t) = f_{Sat}(\, u(t)) \quad \text{(Eq. 3)}$$
$$u_r(t) = f_{Sat}(\, K_p * e(t)\,) \quad \text{(Eq. 4)}$$
$$u_r(t) = f_{Sat}(\, K_p \left(r(t) - y(t)\right)\,) \quad \text{(Eq. 5)}$$

where:

$r(t)$ = desired distance to the wall
$y(t)$ = distance from robot to the wall
$e(t)$ = distance error
$K_p$ = proportional gain or correction error gain
$u(t)$ = control signal corresponding to robot velocity
$f_{sat}$ = saturation function (see explanation below)
$u_r(t)$ = control signal corresponding to the saturated robot velocity

## C. Task Description

The lab consists of the following tasks:

- Task 1 – Motion with PID ('Lab3_Task1.py')
- Task 2 – Wall following with PID ('Lab3_Task2.py')

### C.1 Task 1 – Motion with PID

The robot should use $K_{pf}$ proportional gain control applied to its front sensor to move towards the end wall and control its stopping distance. Robot should use $K_{ps}$ proportional gain control applied to its left and right distance sensors to avoid hitting the side walls. Robot should keep a min distance of 3 inches from any of the side walls. Use both distance sensors and Lidars to compare results. Note that you can use a single $K_p$ for all Lidar PID computations.

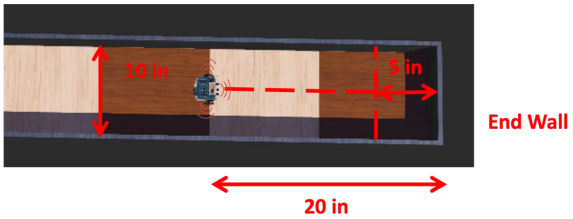


Figure 4: Robot starts 20 inches (0.5 m) away from the end wall and should stop at the 5-inch mark. (Note that in Webots each colored square measures 0.25 m (10 inch) width, where 4 tiles comprise a tile.

Robot should start at 20 inches away front the front wall and should stop at the 5-inch using the front sensor, as shown in Figure 4. Test the program using 6 different values of $K_{pf}$ and $K_{ps}$, (0.1, 0.5, 1.0, 2.0, 2.5, 5.0). Find the one $K_{pf}$ and $K_{ps}$ values that you think works best. Task should run for no more than 30 sec. During evaluation, the TA will start the robot at different distances and orientations from the walls. You need to continuously print distance measurements.

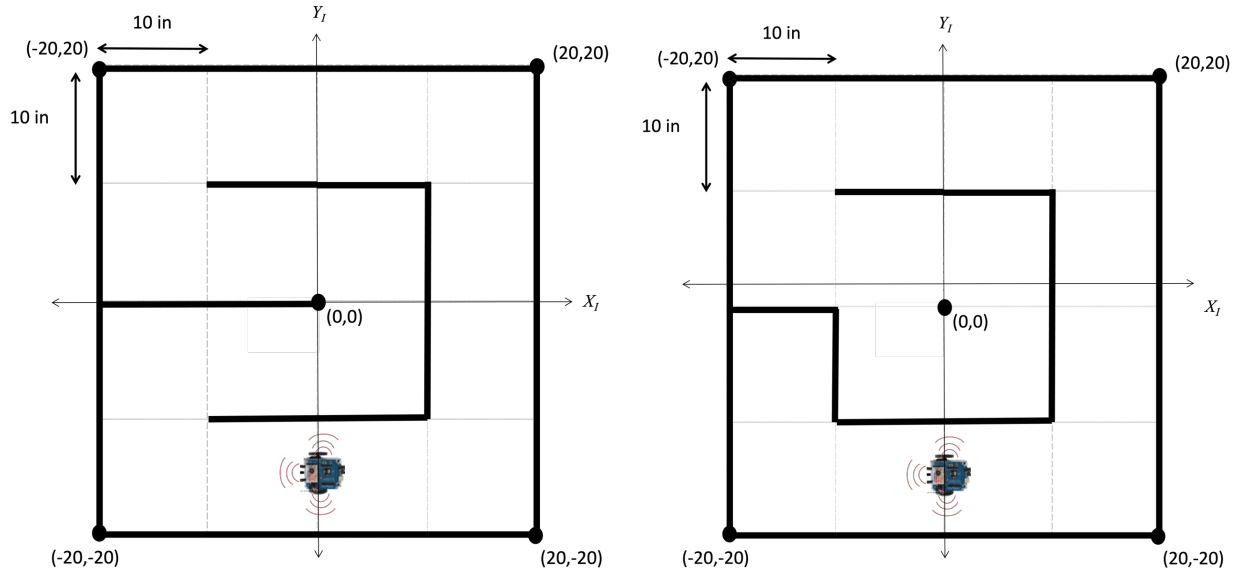


Figure 5 Physical robot maze wall-following task. (Left) Maze (a). (Right) Maze (b).

### C.2 Task 2 – Wall Following with PID

The robot should apply the best $K_p$ proportional gain control from the previous Task. Note that you do not need to use the front sensor controller. If the robot reaches any end wall it should make a 90-degree and continue navigation. If no 90-degree turns are possible, it should make a 180-degree turn, and continue wall following in the opposite direction. Note that depending on the original orientation the robot my end up following a different path. Also note that navigation does not require visiting every single square. The robot can start at any grid cell with any orientation and should run for no more than 1 minute (or less if it already returned to its original starting location). During evaluation, the TA will start the robot at different initial position, orientation, and decide whether the robot should follow left or right walls. Robot should follow one wall at a time. Two different world mazes have been created, as shown in Figure 5, and should be tested using the same controller. Use both distance sensors and Lidars and compare results.

## D. Task Evaluation

Task evaluation is based on: (1) program code, (2) report, including a link to a video showing each different task, and (3) task presentation of robot navigation with the TA. Note that all functions and tasks to be implemented are required in future labs.

### D.1 Task Presentation (90 points)

The following section shows the rubric for the different tasks:

- Task 1 (45 points)
  - Robot stabilizes at specified distance from end wall (10 points)
  - Robot stabilizes at specified distance from side walls (10 points)
  - Robot utilizes PID to reduce speeds proportionally to the front wall (10 points)
  - Robot utilizes PID to updates orientation proportionally to distance from side walls (10 points)

    - Robot restabilizes when placed further or closer to the wall (5 points)
    - Robot hits walls (-5 points)
- Task 2 (45 points)
    - Robot follows left or right walls accordingly (20 points)
    - Robot makes left or right turns accordingly (15 points)
    - Robot follows end of walls accordingly (10 points)
    - Robot average travel speed less than 2 inches per second except for corners (-5 points)
    - Robot constantly exceeds 3 inches closer or further away from any wall (-5 points)
    - Robot hits walls (-5 points)

**D.2 Task Report (10 Points)**

The report should include the following (points will be taken off if anything is missing):

1. Mathematical computations for all kinematics. Show how you calculated the speeds of the left and right servos given the input parameters for each task. Also, show how you decide whether the movement is possible or not. (4 point)
2. Video uploaded to Canvas showing the robot executing the different tasks. You should include in the video a description, written or voice, of each task. You can have a single or multiple videos. Note that videos will help assist task evaluations. (3 point)
3. Conclusions where you analyze any issues you encountered when running the tasks and how these could be improved. Conclusions need to show an insight of what the group has learnt (if anything) during the project. Phrases such as "everything worked as expected" or "I enjoyed the project" will not count as conclusions. (3 point)

**D.3 Task Presentation**

Task presentation needs to be scheduled with the TA. Close to the project due date, a timetable will be made available online to select a schedule on a first come first serve basis. Students must be present at their scheduled presentation time. On the presentation day, questions related to the project will be asked, and the robot's task performance will be evaluated. If it is seen that any presenter has not understood a significant portion of the completed work, points will be deducted up to total lab points. Students that do not schedule and attend presentations will get an automatic "0" for the lab.

NOTE for physical robot presentation: During presentations, robot calibration time is limited to 1 min. It is important that all members of the team attend and understand the work submitted.

## E. Lab Submission

Each student needs to submit the programs and report through Canvas as a single "zip" file.

- The "zip" file should be named "yourname_studentidnumber_labnumber.zip"
- Videos should be uploaded to the media folder in Canvas. Name your files "yourname_studentidnumber_labnumber_tasknumber".
- The programs should start from a main folder and contain subfolders for each task.
- The report should be in PDF and should be stored in the same "zip" file as the programs.