Control of Mobile Robotics

CDA4621

Spring 2023

Lab 2

Kinematics

Total: 100 points

Due Date: 2-13-2023 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: servos.py, encoders.py, imu.py ("SamplePrograms.zip")

A.2 Robot Simulator

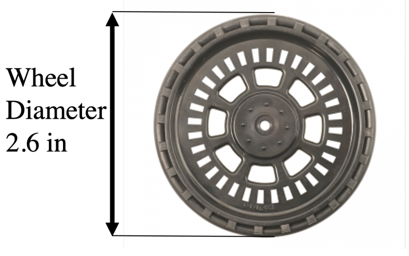
Simulator: Webots. **Robot**: "e-puck". **Programming**: Python
Basic Files: "Lab2_epuck.zip"

B. Objective

This lab will teach you how to compute inverse kinematics and translate them to motion control for the robot to navigate through a set of waypoints. It will also teach you about wheel rotation feedback with an *Inertial Measurement Unit* (IMU) that measures robot rotation angles around *x*-axis (roll), *y*-axis (pitch) and *z*-axis (yaw) with respect to a global coordinate system.

B.1 Physical Robot

"Robobulls-2018" has two wheels, each with approximate diameter 2.6", and approximate distance between wheels 4", as shown in Figure 1.



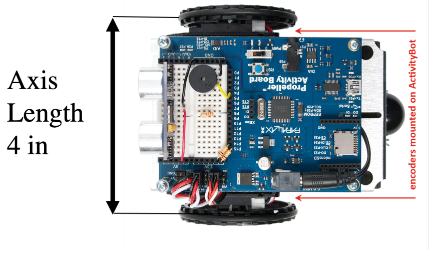


Figure 1. "Robobulls 2018" robot.

B.1.1 IMU

The physical robot is equipped with a 9-DOF Adafruit BN0055 Absolute Orientation IMU sensor[1]. Note that, as shown in Figure 2, the way the IMU sensor was installed in the robot, the *x-axis* points forward, aligned with the front of the robot; the *y-axis* points down from the center

[1] https://www.adafruit.com/product/4646

of the robot; and the *z-axis* points to the left side of the robot. Therefore, if we would like to know which direction the robot is heading, we will look at the rotation reading from the *y-axis*. We will be utilizing the Absolute Orientation (Euler Vector) to get the directional heading.

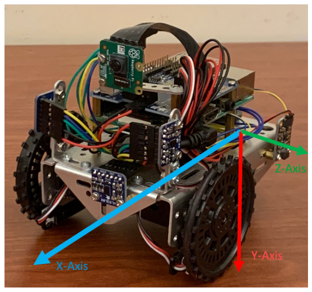


Figure 2. Robot IMU orientations: x-axis, y-axis, and z-axis.

To get readings from the sensor, you will need to call the following variables:

- `sensor.magnetic` : (microteslas) returns three readings corresponding to magnetic direction along the x-axis, y-axis, and z-axis respectively.
- `sensor.acceleration` : (m/s^2) returns three readings corresponding to acceleration along the x-axis, y-axis, and z-axis respectively.
- `sensor.gyro` : (rad/sec) returns three readings corresponding to gyroscopic readings along the x-axis, y-axis, and z-axis respectively.
- `sensor.euler` : (Euler Angles) Three axis orientation data based on a 360° sphere.
- `sensor.quaternion` : (Quaternion) Four point quaternion output for more accurate data manipulation.
- `sensor.linear_acceleration` : (m/s^2) Three axis of linear acceleration data (acceleration minus gravity).

To use this sensor, you will need to include the following lines of code in your python script:

```
import board
import adafruit_bno055
i2c = board.I2C()
sensor = adafruit_bno055.BNO055_I2C(i2c)
```

B.2 Robot Simulator

Webot's "e-puck" robot[2] (you can refer to the original "e-puck"[3] robot) has two wheels, each with approximate diameter 1.6", and distance between wheels 2.28", as shown in Figure 3.

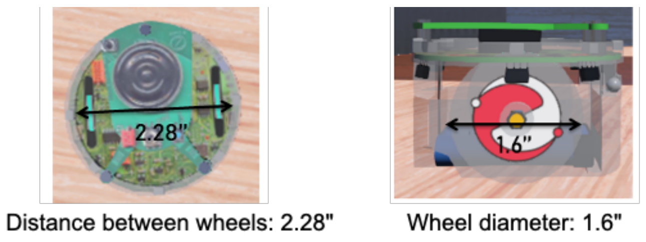


Figure 3. Webot "e-puck" robot.

B.2.1 IMUs

The Webots IMU[4] rotation angles are shown in Figure 4. The robot can rotate around the x-axis (roll), y-axis (pitch) and z-axis (yaw) with respect to a global coordinate system. For this lab, you

[2] https://cyberbotics.com/doc/guide/epuck
[3] http://www.e-puck.org/
[4] Tutorial on Webots' Inertial Unit (IMU) https://cyberbotics.com/doc/reference/inertialunit

are going to be working exclusively with the *z*-axis (yaw) rotation angles with orientation in the interval [-π, π] with respect to the *WorldInfo northDirection*. The *yaw* angle can be used as a compass, where it is 0 when the *InertialUnit x*-axis is aligned with the North direction, π/2 when oriented West, -π/2 when oriented East, and ± π when oriented South. Note that the IMU is located at the center of the robot. Note that orientation of North in the Webots world may not correspond to the starting orientation of the robot.

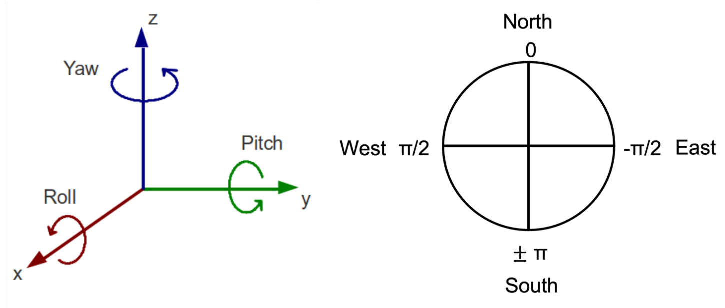


Figure 4. Left: *x*-axis (roll), *y*-axis (pitch) and *z*-axis (yaw). Right: IMU readings.

Use the function *getRollPitchYaw*, returning rotation angle values as an array of 3 components, where only indices 0, 1, and 2 are valid for accessing the returned array, corresponding to *roll*, *pitch* and *yaw*, respectively.

- imu.getRollPitchYaw() : returns all rotation values
- imu.getRollPitchYaw()[2] : returns the Yaw rotation value only

B.3 General Functions

Implement the following IMU general functions as part of this lab.

B.3.1 IMU Functions

IMU functions:

- getIMUDegrees() - function returning the IMU value in degrees with a range of 0-360.

Make sure the IMU sensor readings returned by this function correspond to actual robot pose.

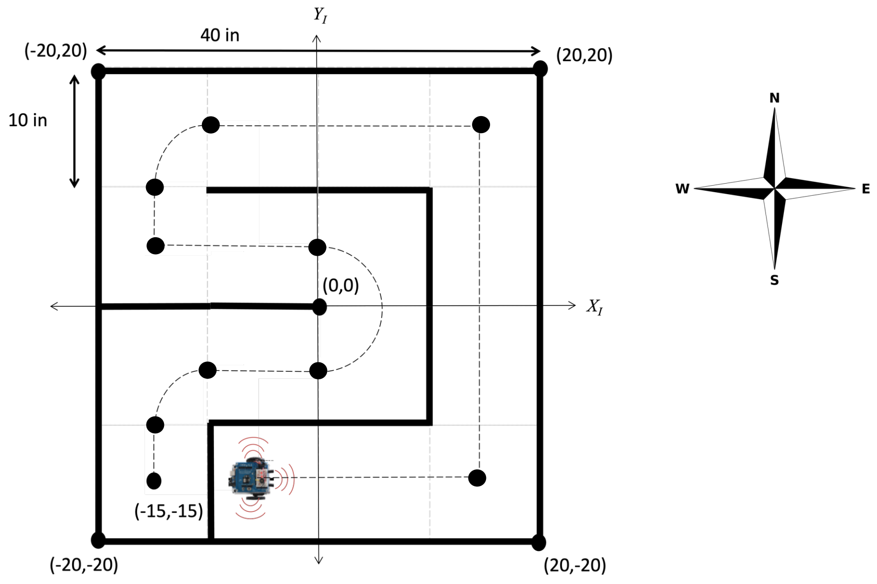


Figure 5. Path to be followed by the robot.

C. Task Description

The lab consists of the following navigation tasks:

- Task 1: Encoders ('Lab2_Task1.py')
- Task 2: IMUs ('Lab2_Task2.py')

The robot should follow the trajectory in dashed lines shown in Figure 5.

C.1 Task 1 – Waypoint Navigation

Follow the precise trajectory shown by the dashed lines in Figure 5 from start P0(-5,-15) to goal P1(-15,-15). The robot will be controlled by an array of waypoints. For example, $A = [P_0, P_1, \ldots, P_9]$, where $P_i = (x_i, y_i)$ is the coordinates in the world frame as shown in Figure 5. Your algorithm should take the waypoints and calculate $V_{li}, V_{ri}, D_{i,}, T_i$, where V_{li} is the left wheel velocity; V_{ri} is the right wheel velocity, D_i is the distance between P_{i-1} to P_i, and T_i is the time it takes to travel D_i. Note that you can decide on V_{li} and V_{ri} as long it follows the trajectory shown by the dashed line. Your code should print $V_{li}, V_{ri}, D_{i,}, T_i$ using both kinematics and the robot's encoders to compute waypoint distances and rotations.

Implement the following function:

- A = [(-15,-15),(15,-15),(15,15),(-10,15),(-15,10),(-15,5),(0,5),(0,-5),(-10,-5),(-15,-10),(-15,-15)]
- Make sure you follow straight lines or curves as illustrated by the dashed line trajectory.
- robotWayPointNav(A) - main task function to control the robot.

C.2 Task 2 - IMUs

Implement a function that keeps track of the robot's pose. The TA will give you a set of time , T_is where you need to track and print the robot's x_w, y_w, θ_w based on your encoder and IMU calculations. You should use the same function used for Task 1 and extend it by updating the robot's pose and printing it while the robot performs waypoint navigation. You are to assume that the starting pose is given to the robot, i.e. robotPose $= [x_0 = -5, y_0 = -15, \theta_0 = 0]$, where $\theta = 0 \rightarrow$ East and $\theta = 90 \rightarrow$ North.

- robotPose(A) – function to update and print the robot pose.

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 Programs (90 points)

The following section shows the rubric for the different tasks:

- Task 1 (45 points)
  - Robot computes and travels the correct path with encoder control (40 points)
  - Robot prints the correct program values (5 points)
- Task 2 (45 points)
  - Robot computes and travels the correct path while keeping track of the robot's pose (40 points)
  - Robot prints the correct program values (5 points)

D.2 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 points)
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 points)

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.