Project 2

ECE 10

Team Name: Just me

Jacob Sickafoose (jsickafo)

May 10, 2022

Foreword

I'm really not sure what we are supposed to be turning in on Canvas. The submission type says .pdf and .py file types so I'm turning in my lab write-up and controller code. This pdf document is just a copy-paste from my github readme.

1. What are the names of everyone in your lab group?

Jacob Sickafoose (jsickafo) -Master repo

2. What happens (in terms of the robot's behavior) during the robot.step(TIME_STEP) statement?

On every time step the robot reads from the line sensors, makes a decision on which direction to move and calculates the change in world coordinates resulting from that.

3. What happens if your robot's time step is not exactly TIME_STEP long, but slightly varies?

The only time the time step is looked at is in updating the odometry. If the time step is not exactly TIME_STEP long, then the calculation for distance traveled during that time step would be inaccurate. Small errors such as this would accumulate over time the way this controller is programmed.

4. What is the ePuck's average speed (in m/s) from Part 1?

In Part 1, the ePuck's average speed was found by taking the change in distance only in the X direction while the ePuck was moving forward at max speed for approximately 4 seconds. I calculated EPUCK_MAX_WHEEL_SPEED = $0.1262 m/s$. Calculating this with the ePuck datasheet, MAX_SPEED * EPUCK_WHEEL_RADIUS = $6.28(rad/s) * 0.0205(m) = 0.12874 m/s$.

5. In an ideal world, what should the ePuck's pose show each time it crosses the starting line?

The Webots simulator says that the ePuck translation(x, y, z) = $(0.498596, -6.39506e-05, 0.0637583)$ at the startline so I think it would be that same pose every time. Except in webots, y is out of the board so it should be pose_x = x, and pose_y = z.

6. How did you implement loop closure in your controller?

I implemented the loop closure as the hints suggested. I made sure all the sensors reading the line was the startline by making it read that for >8 time steps. After these time steps and I know it's on the line, I waited for the start line to not be read anymore before implementing any pose changes.

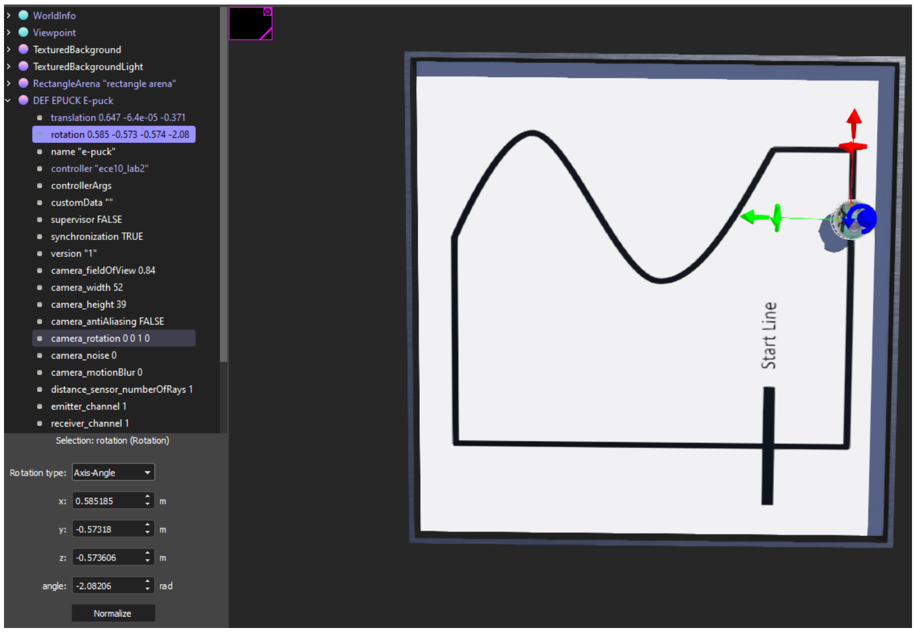
After I was sure I was only triggering once per lap and at the startline, I simply set the pose to the previous pose. I also had the previous pose actually change because I wasn't sure if it should just always reset to the pose of the very first lap. It's really easy to change it back to that but I made a guess for what was expected.

7. Roughly how much time did you spend programming this lab?

I probably spent around 5 hours working on this lab. I've been working on it here and there because I'm super busy with ECE129 and ECE167 which I also have big deadlines for this week.

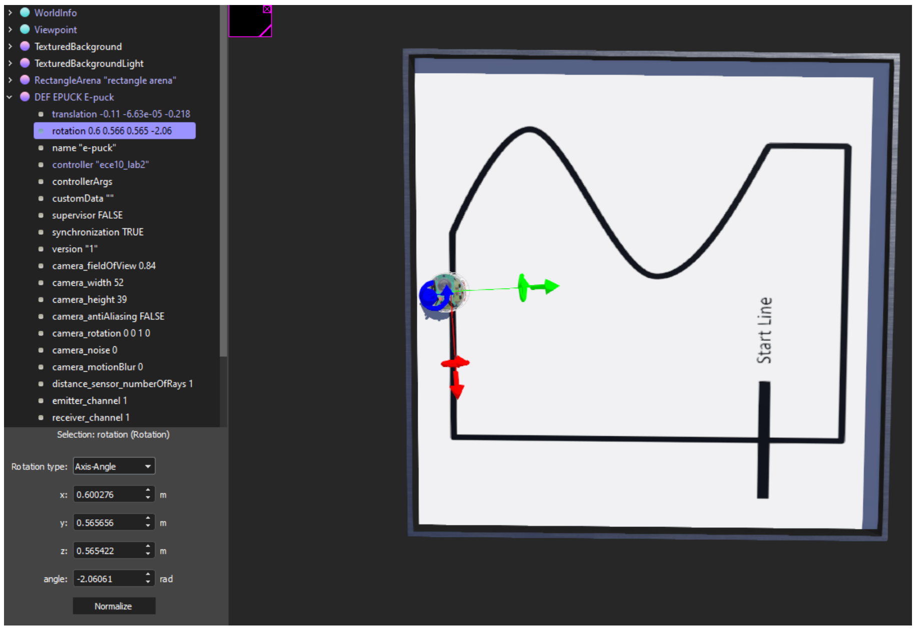
8. Does your implementation work as expected? If not, what problems do you encounter?

In my implementation of the pose, I was thrown for several loops. I started out trying to align my theta to that of the simulator coords because I know X and Y are calculated from that. It took me a long time to realize that the radians field in the rotation of the ePuck is not the theta I was looking for. It would results that were not at all consistent with the ePuck's orientation. Here is a screenshot of the ePuck facing up and down with the same value of rotation:



```
WorldInfo
Viewpoint
TexturedBackground
TexturedBackgroundLight
RectangleArena "rectangle arena"
DEF EPUCK E-puck
    translation 0.647 -6.4e-05 -0.371
    rotation 0.585 -0.573 -0.574 -2.08
    name "e-puck"
    controller "ece10_lab2"
    controllerArgs
    customData ""
    supervisor FALSE
    synchronization TRUE
    version "1"
    camera_fieldOfView 0.84
    camera_width 52
    camera_height 39
    camera_antiAliasing FALSE
    camera_rotation 0 0 1 0
    camera_noise 0
    camera_motionBlur 0
    distance_sensor_numberOfRays 1
    emitter_channel 1
    receiver_channel 1
Selection: rotation (Rotation)
Rotation type: Axis-Angle
x: 0.585185 m
y: -0.57318 m
z: -0.573606 m
angle: -2.08206 rad
```

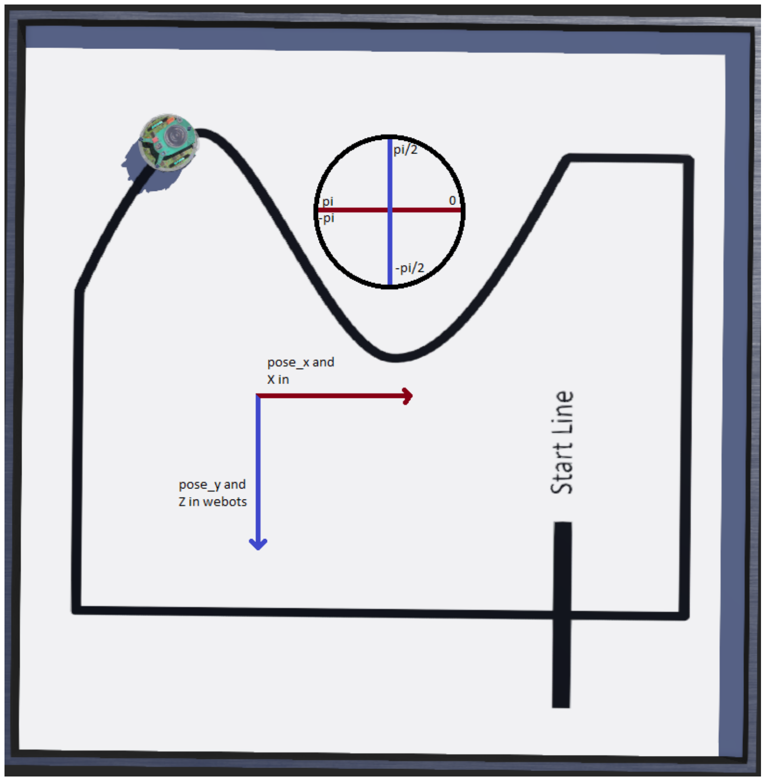
(a) ePuck Pointing Up



```
WorldInfo
Viewpoint
TexturedBackground
TexturedBackgroundLight
RectangleArena "rectangle arena"
DEF EPUCK E-puck
    translation -0.11 -6.63e-05 -0.218
    rotation 0.6 0.566 0.565 -2.06
    name "e-puck"
    controller "ece10_lab2"
    controllerArgs
    customData ""
    supervisor FALSE
    synchronization TRUE
    version "1"
    camera_fieldOfView 0.84
    camera_width 52
    camera_height 39
    camera_antiAliasing FALSE
    camera_rotation 0 0 1 0
    camera_noise 0
    camera_motionBlur 0
    distance_sensor_numberOfRays 1
    emitter_channel 1
    receiver_channel 1
Selection: rotation (Rotation)
Rotation type: Axis-Angle
x: 0.600276 m
y: 0.565656 m
z: 0.565422 m
angle: -2.06061 rad
```

(b) ePuck Pointing Down

I just decided to look at how Webots was counting the coordinates when the ePuck moved by looking at the translation tab. I settled on making the axis the following:



(c) Axis' I used for Pose variables

I wasn't sure what we were really supposed to do, but at least my pose was close to what Webots was saying.