# Lab 2

## Solution

The robot enters the line following state, which has 4 states underneath it: turn left, turn, go straight ahead and run circles if there is no line. It cycles through these states until it detects an object. Then it enters an ‘ultimate avoidance’ state. In this case, turn right for 3 seconds. After this turn it scans the left sensor and enters one of the following states: soft turn left or soft turn right. If the left front sensor detects the wall it will make a hard right turn to avoid it. This is the reason why the robot makes a big turn around a corner. It follows these states until it detects the black line again, then it enters the ‘rejoin’ state. After entering this state the robot will turn clockwise for 4 seconds, when it has completed this task the robot will return to the line following state. It repeats this cycle until there is no line anymore, then it will start running circles until it detects the black line again. I tested the robot on a track with 2 boxes and a cylinder and it ran without a problem so I would give myself a 9 because it isn’t the quickest around an obstacle. Reason for this is that the robot clips through the ground if I decelerate too quickly and as you can imagine, the sensors don’t like that.A picture containing text, monitor, sport

Description automatically generated

## Code

"""Lab2 controller."""

#  from controller import Robot, Motor, DistanceSensor

from controller import Robot, DistanceSensor, Motor

# create the Robot instance.

robot = Robot()

# define max speed

MAX\_SPEED = 6.28

# get the time step of the current world.

TIMESTEP = 32

# set avoid state to 0

avoid = 0

# set runtime to 0 (time since the start of the simulation)

runtime = 0

# set rejoin state to 0

rejoin = 0

# initialize line sensors

gs = []

gsNames = ['gs0', 'gs1', 'gs2']

for i in range(3):

    gs.append(robot.getDevice(gsNames[i]))

    gs[i].enable(TIMESTEP)

# initialize proximity sensors

ps = []

psNames = ['ps0','ps1','ps2','ps3','ps4','ps5','ps6','ps7']

for i in range(8):

    ps.append(robot.getDevice(psNames[i]))

    ps[i].enable(TIMESTEP)

# initz motors

leftMotor = robot.getDevice('left wheel motor')

rightMotor = robot.getDevice('right wheel motor')

leftMotor.setPosition(float('inf'))

rightMotor.setPosition(float('inf'))

leftMotor.setVelocity(0.0)

rightMotor.setVelocity(0.0)

#------------ Main Loop ------------------------------------------------------

while True:

    # start simulation step

    robot.step(TIMESTEP)

    # indicate new info batch

    print('.')

    print('.')

    print('.')

    print('.')

    print('.')

    # define seconds since start simulation, every step is 0.032 seconds

    runtime = round(runtime + 0.032, 4)

    print(runtime)

# Data gathering (distance) --------------------------------------------------

    # Read the distance sensors

    psValues = []

    for i in range(8):

        psValues.append(ps[i].getValue())

    # process distance sensor data

    FLsensor = round(psValues[7])

    FRsensor = round(psValues[0])

    Lsensor = round(psValues[5])

    Rsensor = round(psValues[2])

    # print distance data

    print('sensorFL '+str(FLsensor))

    print('sensorFR '+str(FRsensor))

    print('sensorL '+str(Lsensor))

    print('sensorR '+str(Rsensor))

# Data gathering (Line) ------------------------------------------------------

    # Read the line sensors:

    gsValues = []

    for i in range(3):

        gsValues.append(gs[i].getValue())

    # Process line sensor data here.

    line\_right = round(gsValues[0])

    line\_center = round(gsValues[1])

    line\_left = round(gsValues[2])

    # print line data

    print('line\_right: '+ str(line\_right))

    print('line\_left: '+ str(line\_left))

    print('line\_center: '+ str(line\_center))

#-- Obstacle avoidance -------------------------------------------------------

    #check for obstacle / enter object avoidance state

    if FLsensor > 100 or FRsensor > 100 or avoid == 1:

        if rejoin == 0:

            # set ending variables

            end = runtime - 2

            end2 = runtime - 4

            # turn right when obstacle is detected

            if avoid == 0 or end < start:

                # set avoid state to 1

                print('enter avoid state')

                # start timer

                if avoid == 0:

                    start = runtime

                    avoid = 1

                leftMotor.setVelocity(MAX\_SPEED/2)

                rightMotor.setVelocity(MAX\_SPEED/20)

            # set action trigger to 0

            action = 0

            # turn away from obstacle

            if Lsensor > 150 and action == 0  and end > start:

                leftMotor.setVelocity(MAX\_SPEED/5)

                rightMotor.setVelocity(MAX\_SPEED/20)

                action = 1

                print('turn away object (Lsens)')

            # turn away from obstacle

            if FLsensor > 90 and action == 0  and end > start:

                leftMotor.setVelocity(MAX\_SPEED/5)

                rightMotor.setVelocity(MAX\_SPEED/20)

                action = 1

                print('turn away object (FLsens)')

            # turn towards obstacle

            if Lsensor < 150 and action == 0 and end > start:

                leftMotor.setVelocity(MAX\_SPEED/20)

                rightMotor.setVelocity(MAX\_SPEED/5)

                action = 1

                print('turn toward object (Lsens)')

            # start to rejoin

            line\_check = line\_center + line\_left + line\_right

            if (line\_check < 1100 or rejoin == 1) and end2 > start:

                leftMotor.setVelocity(MAX\_SPEED/5)

                rightMotor.setVelocity(MAX\_SPEED/20)

                print('rejoining')

                rejoin = 1

                start2 = runtime

                print(start2)

        elif rejoin == 1:

            # set ending variables

            end = runtime - 2

            end2 = runtime - 4

            leftMotor.setVelocity(MAX\_SPEED/5)

            rightMotor.setVelocity(MAX\_SPEED/20)

            print('rejoining')

            if end2 > start2:

                avoid = 0

                rejoin = 0

                print('ending avoid obstacle')

#-- Line Following -----------------------------------------------------------

    if avoid == 0:

        # fullspeed ahead

        fullspeed = 0

        if line\_left < 500 and line\_right < 500:

            leftMotor.setVelocity(MAX\_SPEED)

            rightMotor.setVelocity(MAX\_SPEED)

            fullspeed = 1

            print('fullspeed')

        # turn Right

        right = 0

        if line\_right > 500 and line\_left < 500:

            leftMotor.setVelocity(MAX\_SPEED)

            rightMotor.setVelocity(MAX\_SPEED/2)

            right = 1

            print('right')

        # turn Left

        left = 0

        if line\_right < 500 and line\_left > 500:

            leftMotor.setVelocity(MAX\_SPEED/2)

            rightMotor.setVelocity(MAX\_SPEED)

            left = 1

            print('left')

# if there is no line turn right (most probable as it is a right turning oval)

        if fullspeed == 0 and right == 0 and left == 0:

            leftMotor.setVelocity(MAX\_SPEED)

            rightMotor.setVelocity(2)

            print('houston, we have a problem')