CamJam EduKit Robotics - Line Following

Project Line Following

Description You will learn how to get your robot to follow a black line.

Equipment Required

For this worksheet you will require:

- Your robot with the line follower attached to the bottom (see Worksheet 5).
- Paper with a 1cm wide black line - you can print out a short course *CamJam EduKit 3 - Robotics - Line Following Course.pdf*, which is supplied with these worksheets.

How it Works

In Worksheet 5, you learnt how to connect the line detection module to your Raspberry Pi and use it to detect whether it was over a black line or a piece of white paper. In this worksheet you will learn how to get your robot to move along the black line.

Here is the logic for how the line follower will work in this worksheet:

- If the line detector detects that it is over a black line, then drive forward
- If the line detector detects that it is over white, then stop and search for the black line by turning left a little, then right a little, and repeating left/right until it finds the line again.

There are other ways you can use, like scanning one way until it finds the line, and if it doesn't after a while, it turns the other way.

Code

The code will be based on the code you completed at the end of Worksheet 5, so copy it and save it as `8-linefollower.py`.

You're going to update the two functions that are used when the sensor is over a black line or not.

```
def lineseen():
    global isoverblack, linelost
    print("The line has been found.")
    isoverblack = True
    linelost = False
    robot.value = motorforward

def linenotseen():
    global isoverblack
    print("The line has been lost.")
    isoverblack = False
```

You also need to write a function that will turn the robot left and right until it finds the black line again.

```
# Search for the black line
def seekline():
    global direction, linelost
    robot.stop()

    print("Seeking the line")

    seeksize = 0.25  # Turn for 0.25s
```

```
    seekcount = 1  # A count of times the robot has looked for the line
    maxseekcount = 5  # The maximum time to seek the line in one direction

    # Turn the robot left and right until it finds the line
    # Or we have looked long enough
    while seekcount <= maxseekcount:
        # Set the seek time
        seektime = seeksize * seekcount

        # Start the motors turning in a direction
        if direction:
            print("Looking left")
            robot.value = motorleft
        else:
            print("Looking Right")
            robot.value = motorright

        # Save the time it is now
        starttime = time.time()

        # While the robot is turning for seektime seconds,
        # check to see whether the line detector is over black
        while (time.time() - starttime) <= seektime:
            if isoverblack:
                robot.value = motorforward
                # Exit the seekline() function returning
                # True - the line was found
                return True

        # The robot has not found the black line yet, so stop
        robot.stop()

        # Increase the seek count
        seekcount += 1

        # Change direction
        direction = not direction

    # The line wasn't found, so return False
    robot.stop()
    print("The line has been lost - relocate your robot")
    linelost = True
    return False
```

Read through the code above and try to understand what it is doing. Here is an explanation:

- The code uses `direction`, a 'Boolean' variable (True or False) to set the direction the robot will turn.
- It then sets a few variables that it uses to limit the time the robot turns for. The first limit here is `seeksize` (0.25) seconds. It also counts the number of times it can turn left or right; `seekcount`. It also sets the maximum number of times the robot can change direction; `maxseekcount` – here it is 5. Each time the robot turns, the turn count increases by 1, and the turn time (`seektime`) increases to `seeksize * seekcount`.
- The function then starts the motors turning either left or right.
- A timer is started, and while the difference between the time and the start time is smaller than the `seektime`, the code checks to see whether the line detector is seeing the black line.

- If it does see the black line, it stops, and returns the value 'True' to say it has found the line.
- If it does not see the black line within the time limit, `SeekTime`, it increases the seek count, which increases the seek time, and changes the direction of the turn.
- After a certain number of changes in direction (`maxseekcount`), the robot gives up searching. If it has not found the line by that time, it assumes that it will not be able to and stops, returning `False` as the result of the function.

The code that controls the robot then replaces everything under `try`:

```
try:
    # repeat the next indented block forever
    robot.value = motorforward
    while True:
        if not isoverblack and not linelost:
            seekline()
```

This code runs the code within the `try:` section until the Ctrl+C keys are pressed.

Within the `try:` is a `while True:`. Since True is always true, the code within that while will run forever – or until Ctrl+C is pressed.

Within the `while True:`, the code checks whether the line detector is over the black line. If it is, the robot moves forward. If it is not, it searches for the line.

Your code should now look like this:

```
# CamJam EduKit 3 - Robotics
# Worksheet 8 - Line Following Robot

from gpiozero import CamJamKitRobot, LineSensor  # Import the GPIO Zero
Library
import time  # Import the Time library

# Set variables for the line detector GPIO pin
pinLineFollower = 25

linesensor = LineSensor(pinLineFollower)
robot = CamJamKitRobot()

# Set the relative speeds of the two motors, between 0.0 and 1.0
leftmotorspeed = 0.5
rightmotorspeed = 0.5

motorforward = (leftmotorspeed, rightmotorspeed)
motorbackward = (-leftmotorspeed, -rightmotorspeed)
motorleft = (leftmotorspeed, -rightmotorspeed)
motorright = (-leftmotorspeed, rightmotorspeed)

direction = True  # The direction the robot will turn - True = Left
isoverblack = True  # A flag to say the robot can see a black line
linelost = False  # A flag that is set if the line has been lost


# Define the functions that will be called when the line is
# detected or not detected
def lineseen():
    global isoverblack, linelost
    print("The line has been found.")
    isoverblack = True
```

```
    linelost = False
    robot.value = motorforward


def linenotseen():
    global isoverblack
    print("The line has been lost.")
    isoverblack = False


# Search for the black line
def seekline():
    global direction, linelost
    robot.stop()

    print("Seeking the line")

    seeksize = 0.25  # Turn for 0.25s
    seekcount = 1  # A count of times the robot has looked for the line
    maxseekcount = 5  # The maximum time to seek the line in one direction

    # Turn the robot left and right until it finds the line
    # Or we have looked long enough
    while seekcount <= maxseekcount:
        # Set the seek time
        seektime = seeksize * seekcount

        # Start the motors turning in a direction
        if direction:
            print("Looking left")
            robot.value = motorleft
        else:
            print("Looking Right")
            robot.value = motorright

        # Save the time it is now
        starttime = time.time()

        # While the robot is turning for seektime seconds,
        # check to see whether the line detector is over black
        while (time.time() - starttime) <= seektime:
            if isoverblack:
                robot.value = motorforward
                # Exit the seekline() function returning
                # True - the line was found
                return True

        # The robot has not found the black line yet, so stop
        robot.stop()

        # Increase the seek count
        seekcount += 1

        # Change direction
        direction = not direction

    # The line wasn't found, so return False
    robot.stop()
```

```
    print("The line has been lost - relocate your robot")
    linelost = True
    return False


# Tell the program what to do with a line is seen
linesensor.when_line = lineseen
# And when no line is seen
linesensor.when_no_line = linenotseen

try:
    # repeat the next indented block forever
    robot.value = motorforward
    while True:
        if not isoverblack and not linelost:
            seekline()


# If you press CTRL+C, cleanup and stop
except KeyboardInterrupt:
    exit()
```

Running the Code

Place your robot over the black line and run the code.

The robot should then follow the line. You are likely to find that it loses the black line – you will need to adjust your code to set the speed of the motors (the `motorspeed` variables), the size of each turn (`seeksize`) and the number of turns made (`maxseekcount`).

Challenge

Your robot is perfectly able to follow a line with only one line detector. It is possible to use more than one to make line detection quicker and easier. For example, with two line detectors, the robot would look for the white surface each side of the black line. If it sees black on the left-hand detector, it knows it needs to turn left a little. If it sees black on the right-hand detector, it knows it needs to turn right a little. This works quicker than the robot searching left and right for the line.

With three line detectors, the middle detector would look for the line. When it loses sight of the line, the other two detectors can then be used to tell it which way to turn, depending on what they see.

If you want to make your robot follow a line using more detectors, why not buy more, or buy one device that has two, three or five detectors on a single board.