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 6, so lets copy it:

```
cd ~/EduKitRobotics/
cp 7-pwm2.py 8-linefollower.py
nano 8-linefollower.py
```

Add a variable for the line follower GPIO pin to the section where the motor GPIO pins are defined:

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

And add the GPIO setup line to the others:

```
# Set the pinLineFollower pin as an input so its value can be read
GPIO.setup(pinLineFollower, GPIO.IN)
```

You're going to add another function that will return whether the line detector is over a black line or not.

```
# Return True if the line detector is over a black line
def IsOverBlack():
    if GPIO.input(pinLineFollower) == 0:
        return True
    else:
        return 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():
    print("Seeking the line")
```

```
    # The direction the robot will turn - True = Left
    Direction = True

    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 it has been searched for long enough
    while SeekCount <= MaxSeekCount:
        # Set the seek time
        SeekTime = SeekSize * SeekCount

        # Start the motors turning in a direction
        if Direction:
            print("Looking left")
            Left()
        else:
            print("Looking Right")
            Right()

        # 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():
                StopMotors()
                # Exit the SeekLine() function returning
                # True - the line was found
                return True

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

        # Increase the seek count
        SeekCount += 1

        # Change direction
        Direction = not Direction

    # The line wasn't found, so return False
    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 between the comment `# Your code to control the robot goes below this line` and the end of the code.

```
try:
    # Repeat the next indented block forever
    print("Following the line")
    while True:
        # If the sensor is Low (=0), it's above the black line
        if IsOverBlack():
            Forwards()
        # If not (else), print the following
        else:
            StopMotors()
            if SeekLine() == False:
                StopMotors()
                print("The robot has lost the line")
                exit()
            else:
                print("Following the line")

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

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

import RPi.GPIO as GPIO # Import the GPIO Library
import time # Import the Time library

# Set the GPIO modes
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Set variables for the GPIO motor pins
pinMotorAForwards = 10
pinMotorABackwards = 9
pinMotorBForwards = 8
pinMotorBBackwards = 7
```

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

# How many times to turn the pin on and off each second
Frequency = 20
# How long the pin stays on each cycle, as a percent
DutyCycleA = 30
DutyCycleB = 30
# Setting the duty cycle to 0 means the motors will not turn
Stop = 0

# Set the GPIO Pin mode to be Output
GPIO.setup(pinMotorAForwards, GPIO.OUT)
GPIO.setup(pinMotorABackwards, GPIO.OUT)
GPIO.setup(pinMotorBForwards, GPIO.OUT)
GPIO.setup(pinMotorBBackwards, GPIO.OUT)
# Set the pinLineFollower pin as an input so its value can be read
GPIO.setup(pinLineFollower, GPIO.IN)

# Set the GPIO to software PWM at 'Frequency' Hertz
pwmMotorAForwards = GPIO.PWM(pinMotorAForwards, Frequency)
pwmMotorABackwards = GPIO.PWM(pinMotorABackwards, Frequency)
pwmMotorBForwards = GPIO.PWM(pinMotorBForwards, Frequency)
pwmMotorBBackwards = GPIO.PWM(pinMotorBBackwards, Frequency)

# Start the software PWM with a duty cycle of 0 (i.e. not moving)
pwmMotorAForwards.start(Stop)
pwmMotorABackwards.start(Stop)
pwmMotorBForwards.start(Stop)
pwmMotorBBackwards.start(Stop)

# Turn all motors off
def StopMotors():
    pwmMotorAForwards.ChangeDutyCycle(Stop)
    pwmMotorABackwards.ChangeDutyCycle(Stop)
    pwmMotorBForwards.ChangeDutyCycle(Stop)
    pwmMotorBBackwards.ChangeDutyCycle(Stop)

# Turn both motors forwards
def Forwards():
    pwmMotorAForwards.ChangeDutyCycle(DutyCycleA)
    pwmMotorABackwards.ChangeDutyCycle(Stop)
    pwmMotorBForwards.ChangeDutyCycle(DutyCycleB)
    pwmMotorBBackwards.ChangeDutyCycle(Stop)

# Turn both motors backwards
def Backwards():
    pwmMotorAForwards.ChangeDutyCycle(Stop)
    pwmMotorABackwards.ChangeDutyCycle(DutyCycleA)
    pwmMotorBForwards.ChangeDutyCycle(Stop)
    pwmMotorBBackwards.ChangeDutyCycle(DutyCycleB)

# Turn left
def Left():
    pwmMotorAForwards.ChangeDutyCycle(Stop)
    pwmMotorABackwards.ChangeDutyCycle(DutyCycleA)
    pwmMotorBForwards.ChangeDutyCycle(DutyCycleB)
    pwmMotorBBackwards.ChangeDutyCycle(Stop)
```

```
# Turn Right
def Right():
    pwmMotorAForwards.ChangeDutyCycle(DutyCycleA)
    pwmMotorABackwards.ChangeDutyCycle(Stop)
    pwmMotorBForwards.ChangeDutyCycle(Stop)
    pwmMotorBBackwards.ChangeDutyCycle(DutyCycleB)

# Return True if the line detector is over a black line
def IsOverBlack():
    if GPIO.input(pinLineFollower) == 0:
        return True
    else:
        return False

# Search for the black line
def SeekLine():
    print("Seeking the line")
    # The direction the robot will turn - True = Left
    Direction = True

    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 it has been searched for long enough
    while SeekCount <= MaxSeekCount:
        # Set the seek time
        SeekTime = SeekSize * SeekCount

        # Start the motors turning in a direction
        if Direction:
            print("Looking left")
            Left()
        else:
            print("Looking Right")
            Right()

        # 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():
                StopMotors()
                # Exit the SeekLine() function returning
                # True - the line was found
                return True

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

        # Increase the seek count
        SeekCount += 1
```

```
        # Change direction
        Direction = not Direction

    # The line wasn't found, so return False
    return False

try:
    #repeat the next indented block forever
    print("Following the line")
    while True:
        # If the sensor is Low (=0), it's above the black line
        if IsOverBlack():
            Forwards()
        # If not (else), print the following
        else:
            StopMotors()
            if SeekLine() == False:
                StopMotors()
                print("The robot has lost the line")
                exit()
            else:
                print("Following the line")

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

Once you have typed all the code and checked it, save and exit the text editor with "Ctrl + x" then "y" then "enter".

Running the Code

Place your robot over the black line and start the program by typing the following into the terminal window:

```
python3 8-linefollower.py
```

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 `dutycycle` 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.