Like the last worksheet, we will be using a sensor to determine the movements of our robot.

Sensing Lines

You may be wondering exactly how a sensor determines if it is following a line or not. I'm here to tell you that the name of the type of sensor we are using is actually misleading. To a robot, a line is an abstract concept, something that is hard to define. That may seem silly to us humans but if someone had never seen a line before and asked you to describe it, it would be difficult to define one. After all, there are many different types of lines: straight ones, squiggly ones, jagged ones. This is why the line sensor we are using cheats!

Like the distance sensor you have already used, the line sensor sends out waves and examines what it gets back. However, instead of sound, the line sensor emits light in the infrared spectrum (Vishay, 2022). On its own, this isn't useful. However, if your line is black, it will absorb the light and less will be reflected back for the sensor to detect (BBC, 2022). Therefore, your sensor will determine if it is over a line or not based upon the amount of light reflected back to it. This means that you can use black tape to create a line for your robot to follow (Figure 1). As long as the line is surrounded by a colour that isn't too dark, preferably white, the sensor will be able to tell the difference.

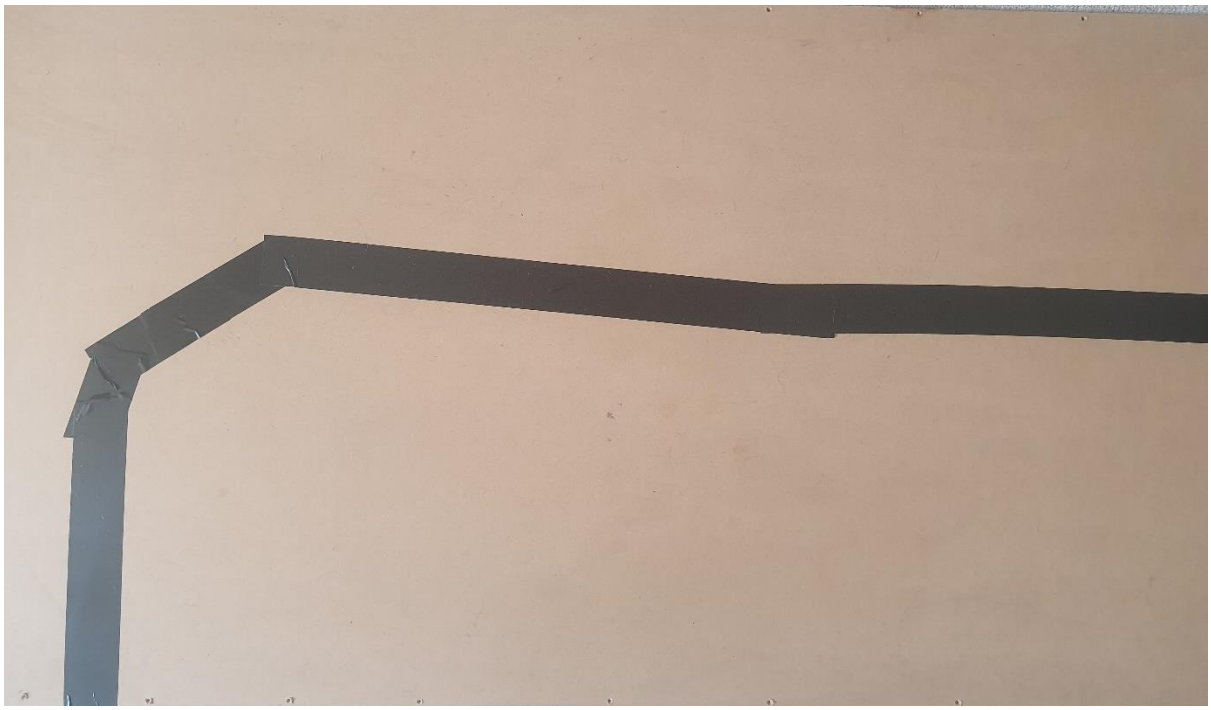

Figure 1 Black tape line

Coding

Just like for our distance sensor program, you will begin by setting up everything you are going to need to use in your main body of code (Figure 2) along with your motor functions (Figure 3).

```
# Import necessary Python libraries
from gpiozero import CamJamKitRobot, LineSensor
import time

# Motor constants (must be in the range 0 < x <= 1)
FORWARD_SPEED = 0.25
BACKWARD_SPEED = 0.25
TURN_SPEED = 0.25

# Component constants
# These could change depending on where you plug your components onto the GPIO pins of your Pi
LINE_SENSOR_PIN = 25

# Setting objects so their associated functions can be called
robot = CamJamKitRobot()
lineSensor = LineSensor(pin = LINE_SENSOR_PIN)
```

Figure 2 Setting up program

```
# Functions to control the motors of the robot
def forward():
    robot.forward(FORWARD_SPEED)

def backward():
    robot.backward(BACKWARD_SPEED)

def left():
    robot.left(TURN_SPEED)

def right():
    robot.right(TURN_SPEED)

def stop():
    robot.stop()
```

Figure 3 Motor functions

After writing the above code for the program, we will create the functions we are going to use to detect the line (Figure 4). The “detectLine” function will turn the robot left and right until the sensor detects it is back over a line. Depending on what the sensor detects, the “follow” function or “search” function will be run. As the “search” function calls to the “detectLine” function, if the robot cannot detect a line, the program will loop here infinitely.

```
34 # Turns the robot left and right to find line if sensor moves off of line
35 def detectLine():
36     robot.left(TURN_SPEED)
37     time.sleep(1)
38     robot.right(TURN_SPEED)
39 
40     lineSensor.when_line = follow
41     lineSensor.when_no_line = search
42 
43 # Follows line when found
44 def follow():
45     print("found")
46     forward()
47 
48 # Activates detectLine() when line is lost
49 def search():
50     print("lost")
51     detectLine()
```

Figure 4 Detect line functions

The functions “when_line” and “when_no_line” that are part of our line sensor are used to trigger which response is necessary for our robot to stay on line (Figure 4).

```
53 while True:
54     # Using "try" with "except" helps handle any errors that occur
55     try:
56         # Waits for a line to be found to begin the line following program
57         lineSensor.wait_for_line()
58         forward()
59 
60         # If robot goes off line, stop and detect line
61         lineSensor.wait_for_no_line()
62         stop()
63         detectLine()
64 
65     # Press ctrl + c to stop the motors and terminate the program
66     except KeyboardInterrupt:
67         stop()
68         quit()
```

Figure 5 Main event loop

Like the rest of the programs we have made, the code that will call our functions is contained within a “while” loop so that our robot will follow a line no matter how long it is (Figure 5). Similar to our distance sensor, we are using “wait_for_line()” and “wait_for_no_line()” to determine when calls to the appropriate functions will be made. The robot will not begin to move until it is placed over a line because of this.

Challenge

- Build a track to test your line following code. Perhaps you can refine the algorithm so that it more accurately follows the line, never accidentally turning back on itself?

END OF WORKSHEET 4

Reference List

BBC, 2022. *What happens when light and sound meet different materials? - OCR 21C.* [Online]
Available at:
https://www.bbc.co.uk/bitesize/guides/zg7jng8/revision/3#:~:text=Absorption%20of%20light&text=When%20waves%20are%20absorbed%20by,colours%20of%20light%20are%20absorbed.
[Accessed 20 04 2022].

Vishay, 2022. *TCRT5000, TCRT5000L.* [Online]
Available at: https://www.vishay.com/docs/83760/tcrt5000.pdf
[Accessed 20 04 2022].