THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

DEPARTMENT OF
COMPUTER SCIENCE
Te Tari Rorohiko

2024

Contributors

J. Turner

V. Moxham-Bettridge

J. Bowen

SELF-DRIVING MAQUEEN

The goal of the Micro:bits and Maqueen session was to get your robot car to complete the course and cross the finish line using almost any technique that did not involve physically pushing the Maqueen. This week is similar in that you have a course to complete, however your robot car must be fully autonomous and complete the track without your input. This can be done using the line following sensors on the underside of the robot.

Line Following Theory

There are 2 line sensors on the Maqueen robot which are located on the underside of the circuit board at the front near the single white wheel (refer back to figure 27 in the previous session for a diagram). Each sensor consists of an IR emitter and receiver. They work by emitting IR light onto a surface (e.g. the floor) and then capturing the reflection using the receiver. *If the receiver detects reflection, the sensor will return 1, else if little to no reflection is detected (like the case with a black line) the sensor will return 0.* This is because a black object will absorb all wavelengths of light whereas a white surface is the opposite in that it reflects all wavelengths of light[4]. If you would like to know more, have a look at the links in the 'Useful Resources' section at the end of this session.

Getting started

Firstly, download the .hex file on the Slack channel and import it into the Microsoft MakeCode editor (feel free to seek assistance from staff about this). The file already includes the extensions you will need for this session.

Now, to see how the sensor works in practice and not just in theory, drag the `read line tracking sensor` tile into your code (see figure 34).

[4] A summary from the UC Santa Barbara website: http://scienceline.ucsb.edu/getkey.php?key=3873

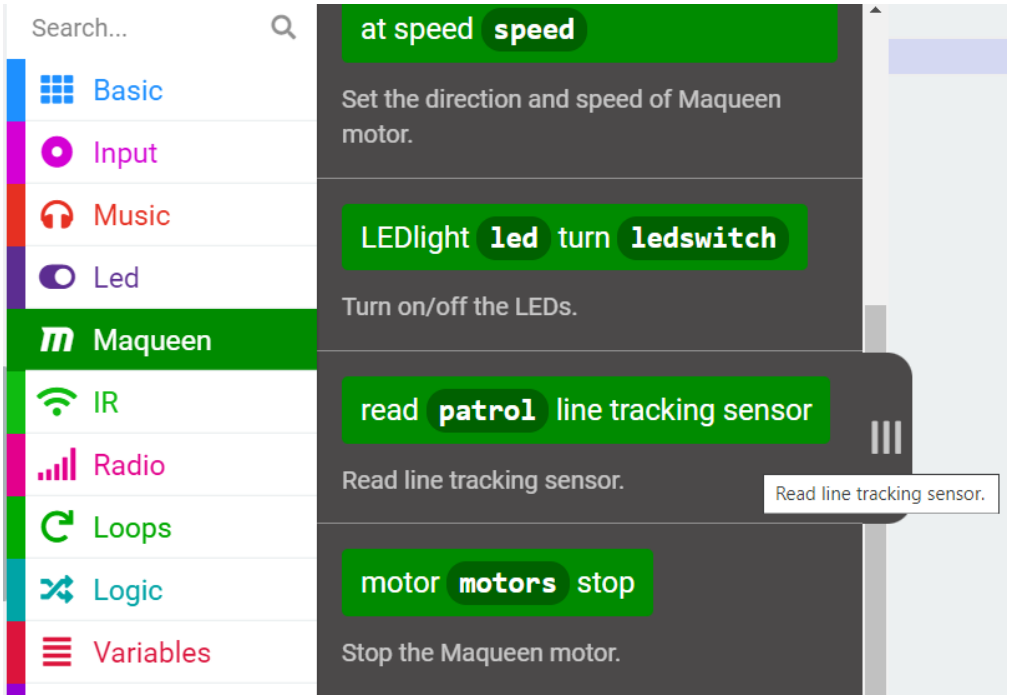

Figure 34: Adding the 'read line tracking sensor' tile to our code

You should now see the code in figure 35 in your workspace.

```
1
2 maqueen.read_patrol(maqueen.Patrol.PATROL_LEFT)
```

Figure 35: The code for reading the left line sensor value

What this line of code does is read the left line sensor, however by itself, this isn't very useful so let's light up the Maqueen's LEDs instead (see figure 36).

```
1  def on_forever():
2      left_sensor = maqueen.read_patrol(maqueen.Patrol.PATROL_LEFT)
3      right_sensor = maqueen.read_patrol(maqueen.Patrol.PATROL_RIGHT)
4
5      if left_sensor == 0:
6          maqueen.write_led(maqueen.LED.LED_LEFT, maqueen.LEDswitch.TURN_OFF)
7      if right_sensor == 0:
8          maqueen.write_led(maqueen.LED.LED_RIGHT, maqueen.LEDswitch.TURN_OFF)
9      if left_sensor == 1:
10         maqueen.write_led(maqueen.LED.LED_LEFT, maqueen.LEDswitch.TURN_ON)
11     if right_sensor == 1:
12         maqueen.write_led(maqueen.LED.LED_RIGHT, maqueen.LEDswitch.TURN_ON)
13
14 basic.forever(on_forever)
```

Figure 36: The code for lighting up the LEDs based on the line sensor readings

Download your program to the Micro:bit, and fold out the black and white circular track. What happens when you move the Maqueen robot over this track?

If you had copied the previous code correctly, then you should have seen the LEDs turn on when the Maqueen was over a light surface (e.g. the white tables) and turn off when over a dark surface (e.g. a black line). Knowing this along with what we did in the Micro:bits and Maqueen session, how do you think we can instruct the robot to follow a line?

There are 4 things that can happen when the robot follows a line, these are: both sensors are on the line, left sensor is on the line (right sensor is not), right sensor is on the line (left sensor is not), and neither sensor is on the line (i.e. both are off). The easiest case to handle is the first one because if both sensors are on the line, then the robot should move forward.

Delete your current code and replace it with the following:

```
1 def on_forever():
2     if maqueen.read_patrol(maqueen.Patrol.PATROL_LEFT) == 0
      and maqueen.read_patrol(maqueen.Patrol.PATROL_RIGHT) ==
0:
3         maqueen.motor_run(maqueen.Motors.ALL,maqueen.Dir.CW,
    25)
4 basic.forever(on_forever)
```

Line 1 defines a *function* called `on_forever`, line 2 checks if the sensor values are equal to 0, if they are (meaning they are both on the line), line 3 executes causing the Maqueen to move

forward with a speed of 25. Line 4 uses the `basic.forever` function to run the `on_forever` function infinitely (i.e. never stops).

Place the robot onto the circular track and turn it on, what happens?

If your robot started on the circle and then drove off in a straight line, don't worry that was meant to happen, but this isn't ideal if we are wanting the robot to follow the line. The reason it did this is because of the other cases mentioned earlier, after it moved forward one of the sensors would have been off of the line and as there is no check for that, the motors weren't told to stop. To fix this, we need to have checks for each of the cases, but let's start with the robot veering to the right.

If the robot has gone off the track towards the right, it means the left sensor will have a value of 0 and the right sensor will have the opposite value (i.e. 1). This means the Maqueen has to spin anticlockwise to find the track again (see the code below).

```
1 def on_forever():
4     if maqueen.read_patrol(maqueen.Patrol.PATROL_LEFT) == 0
      and maqueen.read_patrol(maqueen.Patrol.PATROL_RIGHT) ==
1:
5     maqueen.motor_run(maqueen.Motors.M2, maqueen.Dir.CW, 25)
6     maqueen.motor_stop(maqueen.Motors.M1)
7 basic.forever(on_forever)
```

Please note that lines 1-3 are the same as the previous code snippet (don't delete these as you still need them) and line 4 of the previous code has been moved to line 7 in the current code. Feel free to ask staff for clarification about this.

Now you know how to deal with the robot going off to the right of the track, how about for the left side? The code will be very similar to what you've already written, just remember to write it within the function (i.e. between def `on_forever()` and `basic.forever(on_forever)`).

What happens if the robot goes completely off the track? How will you deal with it? You should have a check for this case too (it's up to you to decide what actions the robot should do).

Test your robot on the circular test track, make sure it actually follows the line and ends up in the same place as it starts. Once you're happy with the performance, move on to the advanced exercises and let your robot hit the (paper) streets!

Summary

This week we have expanded our knowledge of Micro:bits & Maqueen by making self-driving robots that follow a line. Hopefully it has been a fun and fairly easy activity even if you haven't used Micro:bits before. Next week, we move on data structures and introduce you to how computers store data.

Advanced exercises

Have you got your robot already following the circular track smoothly? Want to take on the city circuit? If so, have a look at these extra exercises to expand the functionality of your robot.

1. When the front distance sensors detect a stationary obstacle, can you make your robot come to stop?
2. When stopped, your robot's rear RGBs should glow red (see the 'neopixel' module).
3. When the front distance sensors detect an object moving towards your robot, it should reverse.
4. When reversing, your robot should make a reversing noise.

Useful Resources

- BBC Micro:bits Introduction page: https://microbit.org/get-started/first-steps/introduction/
- MicroPython Home page: https://micropython.org/
- DFRobot Education homepage (the manufacturers of Maqueen): https://edu.dfrobot.com/tag-497.html
- Why do black objects absorb more light than other colours: http://scienceline.ucsb.edu/getkey.php?key=3873
- Why do black materials absorb light and white materials reflect light: https://www.columbiatribune.com/story/lifestyle/family/2013/08/07/why-do-black-materials-absorb/985683007/
- A more technical description about how line sensors work (example is using a Raspberry Pi but the concepts are applicable in general).: https://www.futurelearn.com/info/courses/robotics-with-raspberry-pi/0/steps/75899#:~:text=Infrared%20(IR)%20detection-,Line%20sensors%20detect%20the%20presence%20of%20a%20black%20line%20by,a%20light%20sensor%20(receiver).