

# 4tronix bit:bot makecode starter guide

This is the fourth part of our tutorial, and assumes that you've already completed Part 1 and Part 2. We'll be adding to the **forever** block, so that it repeats endlessly. If you deleted the **forever** block, you can find it by searching at the top of the command categories.

In this tutorial, you'll learn how to:

1. Think about the different conditions that can be detected by the line following sensors.
2. Use data from these sensors to make the Bitbot follow a line, and consider ways to improve the accuracy of this action.

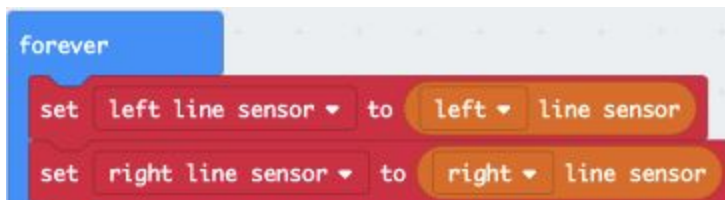
You can find the finished code for this project at [https://makecode.microbit.org/\\_RisDRgPCJh5P](https://makecode.microbit.org/_RisDRgPCJh5P)

## Conditions to detect for line-following

- ❑ Although there can be many different situations that occur when using two sensors to detect light or dark surfaces beneath the Bitbot, we'll start with the two most obvious:
  - a. The left sensor is over the line, and the right sensor is not
  - b. The right sensor is over the line, and the left sensor is not
- ❑ So, we can look at using the **forever** loop and assign the sensor data to variables to take a 'snapshot' at the start of the loop. Then we can use the **if..then..else** block to test for our conditions, and make the Bitbot turn left or right, depending on which case is detected.

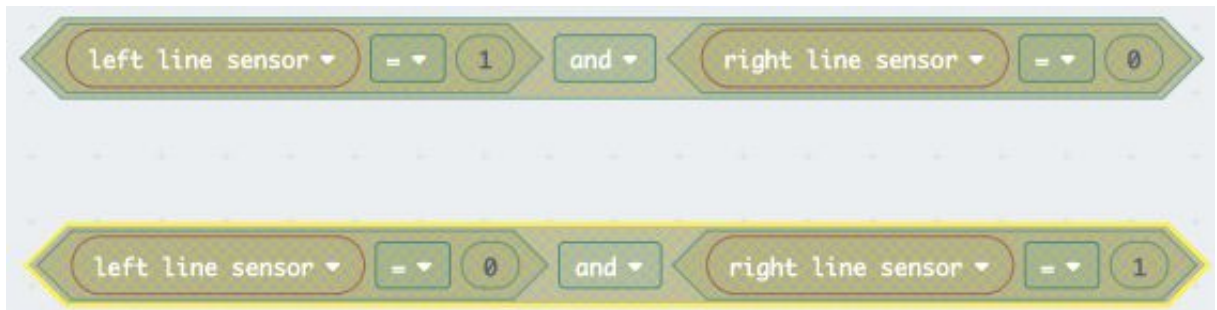
## Store line-following sensor data to variables

- ❑ As in Part 3, we'll need to create variables to store data from the light sensors, so that it doesn't change during each repetition of our loop. Create two variables, and call them **right line sensor** and **left line sensor**.
- ❑ From the **Variables** category, drag two of the the **set..to** blocks into the **forever** block, and add the **Bitbot > ..line sensor** blocks to the variables.



## Test for conditions

- ❑ We'll need the if..then..else block to test for multiple conditions. A good way to clearly visualise these is to build up the conditions with logic blocks:

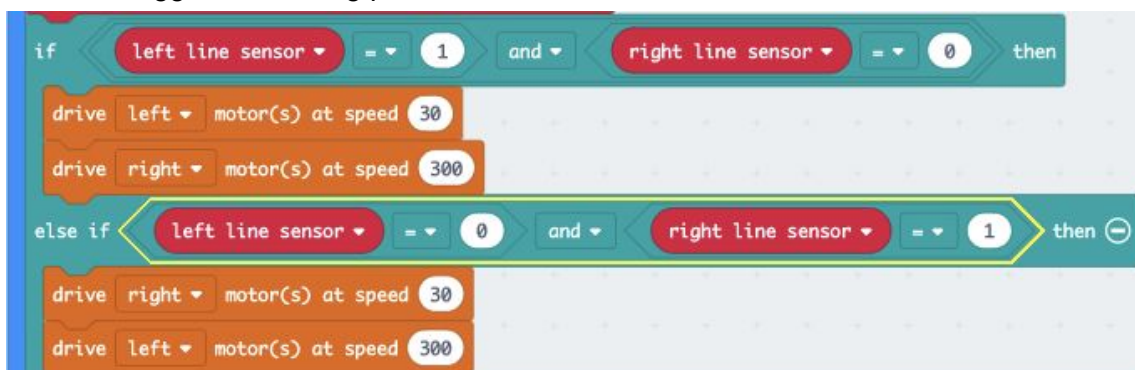


- ❑ The right and left line sensor variables are loaded with sensor data at the start of each repetition of the loop.

Based on the conditions above, to keep the robot on a 1cm wide black line:

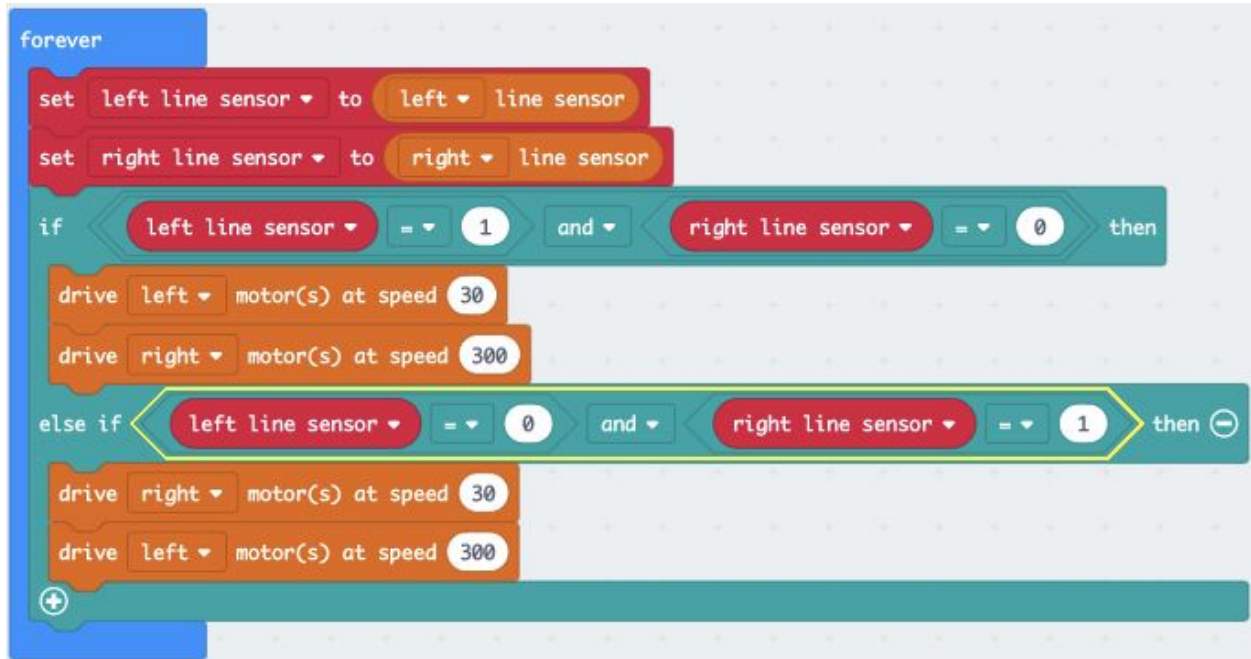
- If the left sensor is on the line, and the right is not, we want to turn the robot left
- If the right sensor is on the line, and left is not, we want to turn the robot right

Knowing exactly how much we want to turn the robot is where the trial and error comes in. We'll provide you with some suggestions to get started. Using the if..then..else block, here is a suggested starting point:



## The finished code

The completed working code looks like this:



```
forever
  set left line sensor to left line sensor
  set right line sensor to right line sensor
  if left line sensor = 1 and right line sensor = 0 then
    drive left motor(s) at speed 30
    drive right motor(s) at speed 300
  else if left line sensor = 0 and right line sensor = 1 then
    drive right motor(s) at speed 30
    drive left motor(s) at speed 300
```

Loading and running this code on your Bitbot will make it:

1. Read the line-following sensor data from the bottom light sensors
2. Drive the motors to correct the robot's path, based on detected conditions

Although this code works for a slightly wavy line, there's lots of room for improvement, and more than just two conditions you could test for. You can also change the amount of correction that the robot performs with its motors, to make the robot run faster, while staying on the line.