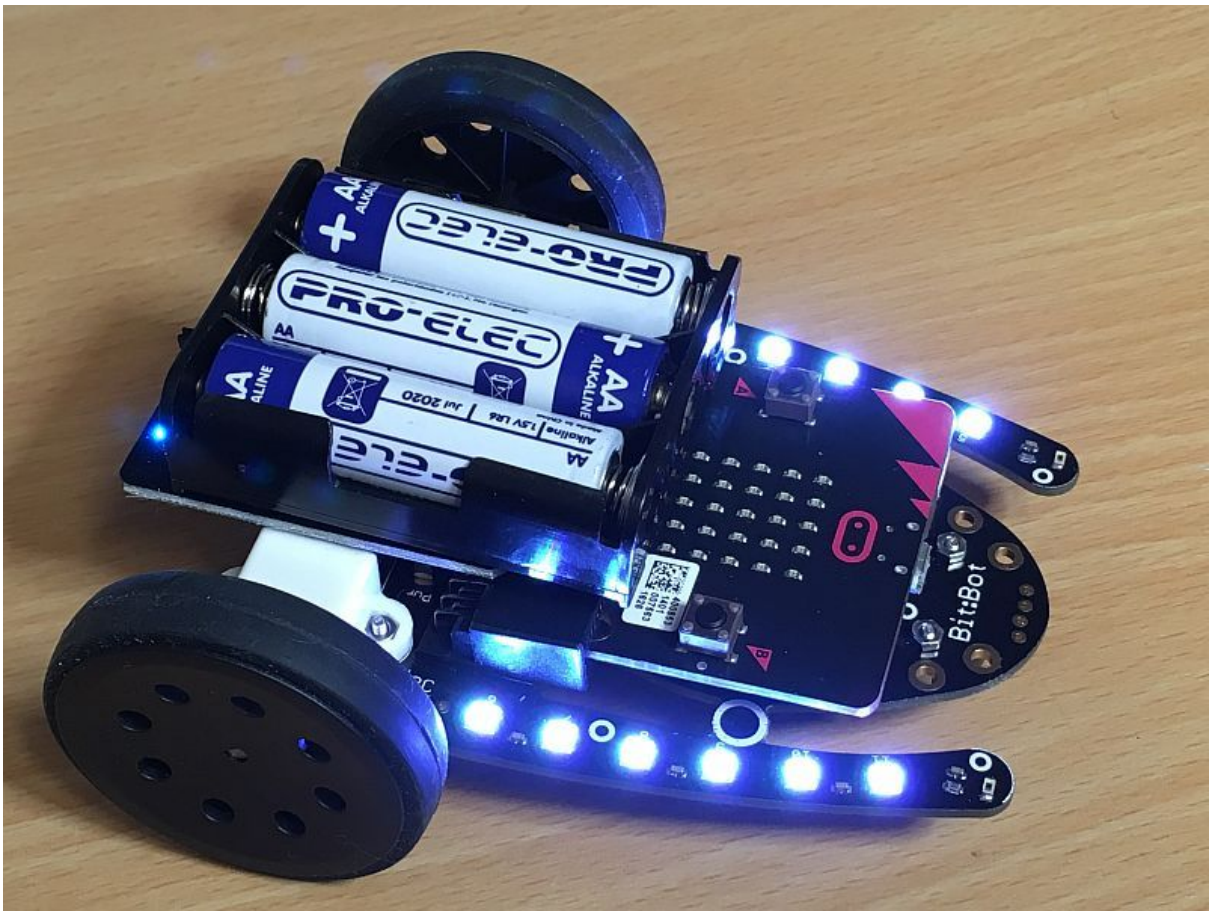
Bitbot Workshop - Robots and Micro:bit coding



By Joe Brown 7th March 2019

Updated 17th July 2019

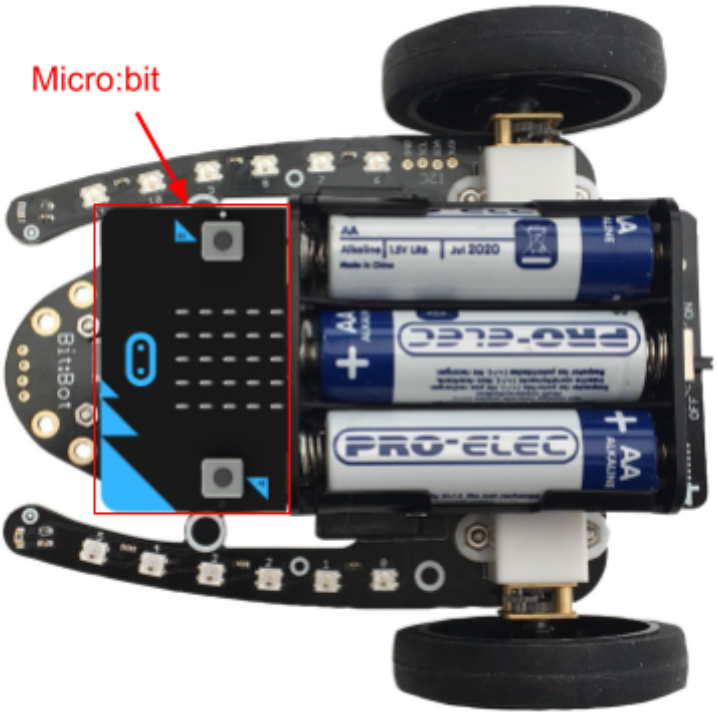
Block based coding.

Text based coding version at: outreach.mrvs.city/workshops/bitbot-intro

1 Introduction

1.1 Introduction

This is a worksheet to teach you some coding concepts using ‘block based editing’ with Micro:Bit through the example of a robot platform called Bitbot!

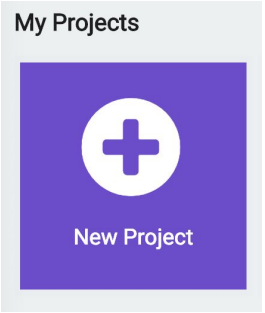


The Micro:bit is basically the brain of your robot.
You will be giving it commands by using blocks of code.

The bitbot is the body of the robot - with motors, sensors and lights.
The Micro:bit will use your code to tell the bitbot what to do.

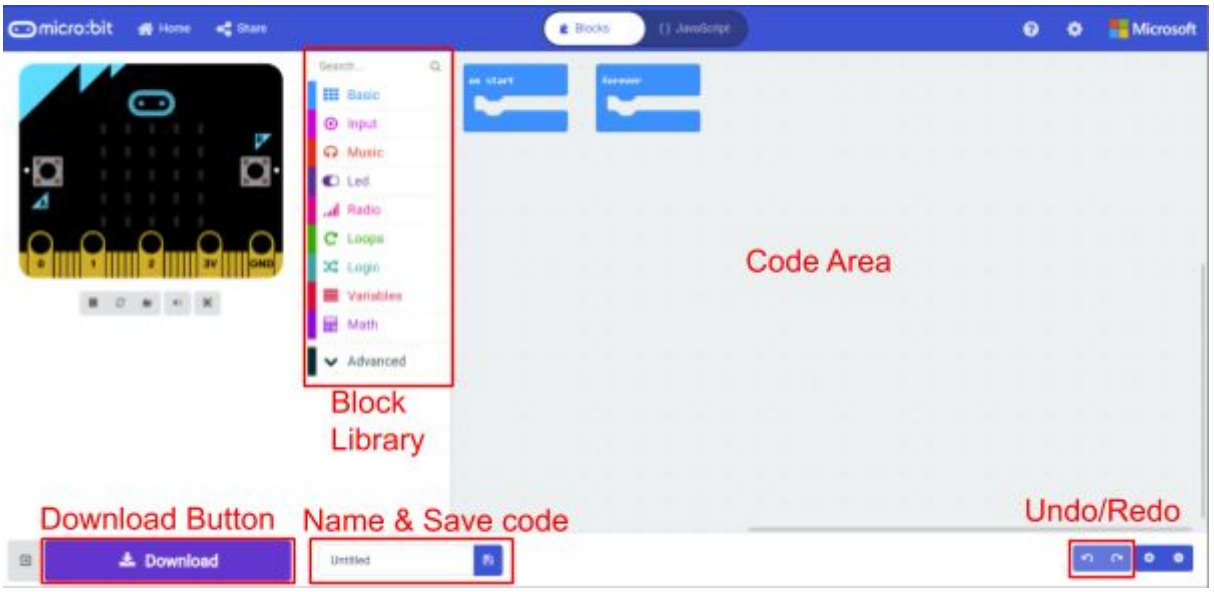
1.2 Setup

Go to https://makecode.microbit.org and click New Project:

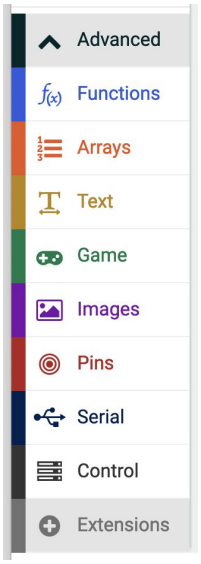


A new project will load.

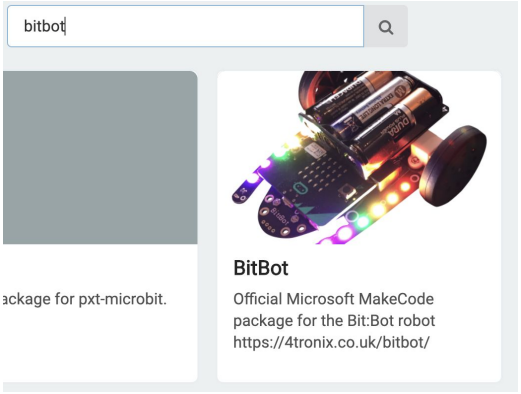
This is the layout of the editor, and all the buttons you'll need:



We just need to add the blocks for the bitbot.
Click **Advanced** > **Extensions** in the Block Library



Search for bitbot and click on 'BitBot - Official Microsoft MakeCode package'



Now you're good to get coding!

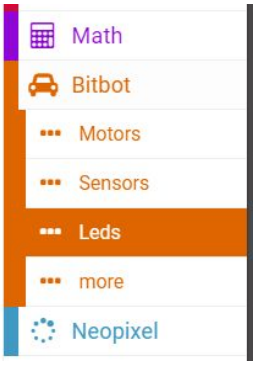
2 Lights

Let's do some coding and test the botbit is working by turning on it's lights!

2.1 Coding the lights

You'll be using the **on start** block seen here:

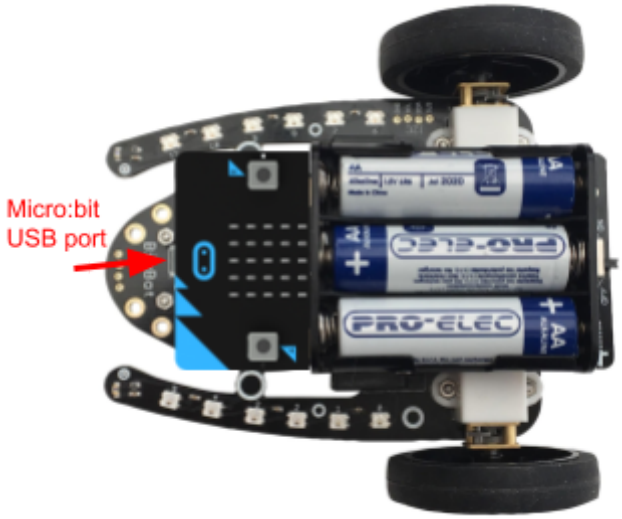
Click on the **Bitbot** tab and then on **Leds** (these 'Leds' are the lights on the Bitbot)



For now let's set the lights to a rainbow! Click and drag the **set led rainbow** block into the **on start** block:

2.2 Testing the code

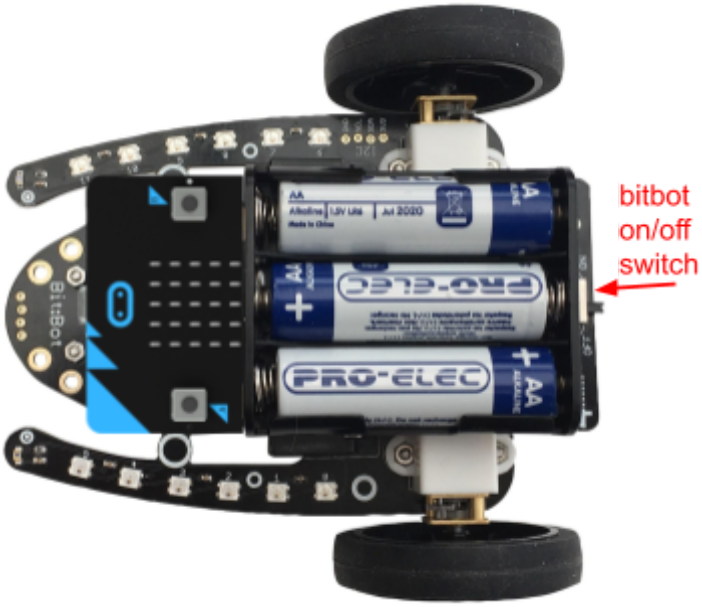
Let's test out our code! Plug the USB cable into the computer and into the micro:bit



Click the download button and the bottom of the screen and save the code (hex file) to the Micro:Bit.

You may need to find it on ‘My Computer’ called MICROBIT - Ask for help if you’re stuck. You may need to copy your hex file from Downloads to MICROBIT.

Once the hex file is copying over to the micro:bit, you should see a little yellow light flashing on it next to the USB port. Once it’s stopped flashing, unplug the USB cable and switch on the bitbot with the on/off switch!



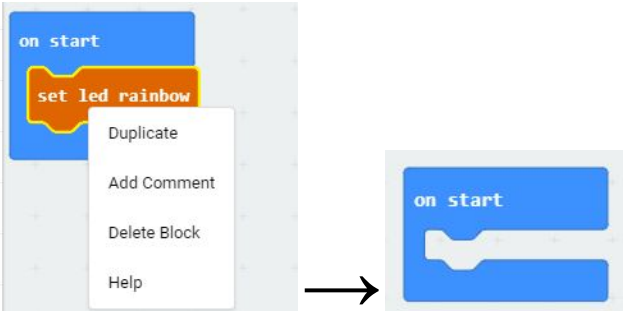
Hopefully when you turn on the bitbot you can see a rainbow of lights like so:

If it doesn’t look like this check your code and make sure it has downloaded to the right place. Also make sure you unplug the USB cable. If it’s still not working ask a helper for help.

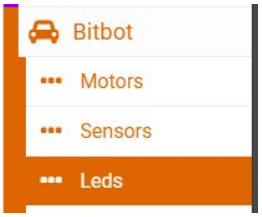
Now switch off the bitbot.

2.3 Customising the lights

Now the lights are working it's time to customise them!
Delete the **set led rainbow** block by right clicking on it and clicking 'Delete Block'

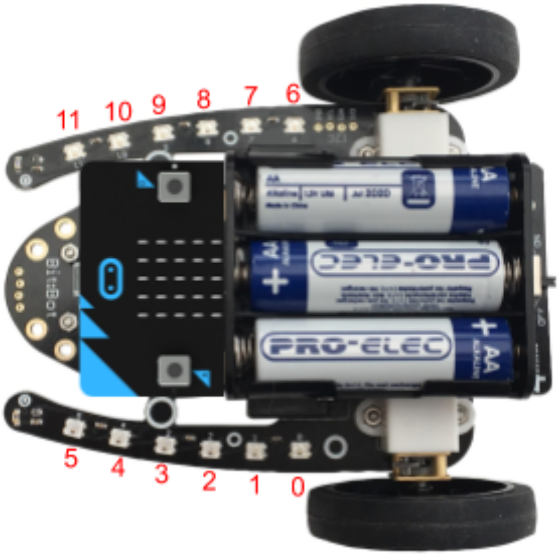


And go back to the **Bibot Leds** tab



And add the **set LED at** 0 **to** red block:

This will set LED 0 to red. Duplicate this **set LED** (right click and click duplicate) for all the LEDs you want to set the colour of. The locations are 0 to 11, set them to whatever colour you like. Here's an example for you, but you can set them to anything you like!



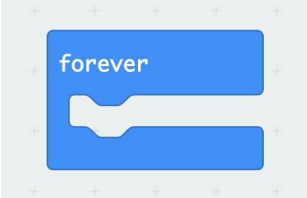
When you're happy with your code, click download and save it to the micro:bit. Once it's finished copying to the micro:bit and the USB light has stopped flashing, unplug the USB cable and switch on the bitbot. Are the lights working like you expected?

3 Movement

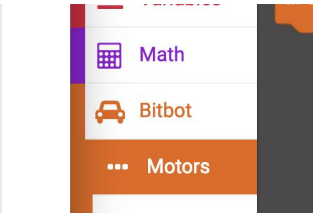
Now switch off the bitbot.

3.1 Forever Loop

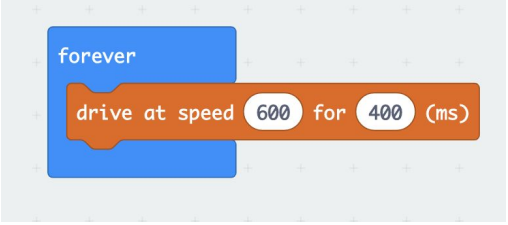
You'll be using the **forever** block seen here:



Blocks you put inside this **forever** block will repeat forever or 'loop'.
Let's test this out by getting your Bitbot to move.
Click the **Botbit** tab and click the **Motors** section.

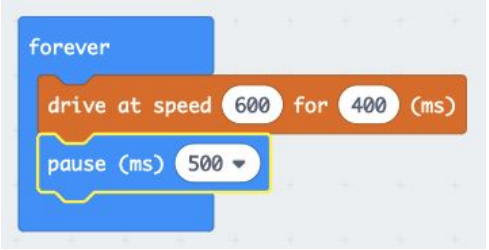


Click the '**drive at speed** 600 **for** 400 **(ms)**' block, drag and drop it inside the forever block



If you didn't know 'ms' is short for 'milliseconds' or one thousandth of a second.
So 400ms is the same as 0.4 seconds.
Now let's add a **pause** from the **Basic** tab:

And set it to 500ms (0.5 seconds)



So this code should make the bitbot:

- On start, show your custom lights!
- drive for 0.4 seconds
- pause for 0.5 seconds
- repeat over and over (forever!).

3.2 Download and test

Make sure your micro:bit is connected via USB to the computer and download the code and save it to the micro:bit.

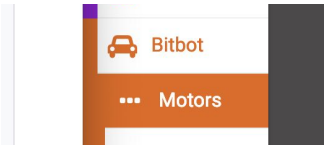
Once the USB light on the micro:bit has stopped flashing, unplug the USB cable and switch on the bitbot

Is the bitbot moving? Does the code behave like you think it should?
If so, now switch off the bitbot.

3.3 More movement

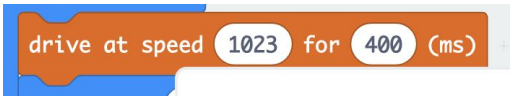
Now you've got your botbit moving it's time to add some more movements.

Add more blocks from the **Bitbot Motors** tab



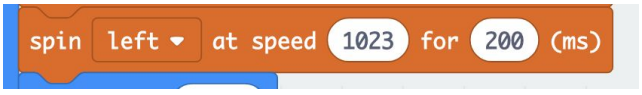
Here are some example movements, (maximum speed is1023, minimum speed is 1):

Full speed forwards (for 400ms):

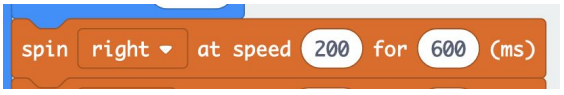


Set the speed to minus to go backwards. Slow speed backwards (for 300ms):

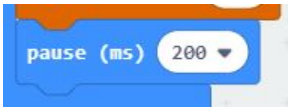
Turn left full speed (for 200ms):



Turn right slowly (for 600ms):

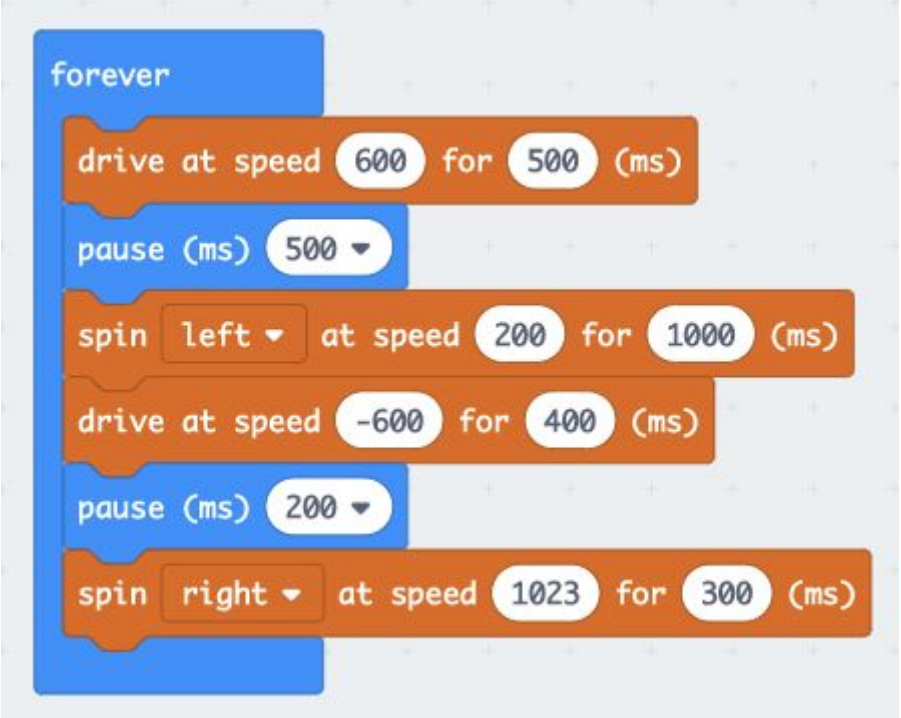


Stop moving for (200ms):



Add a few different motor movements to your botbit code.
Once you're happy with your code, download and switch on the bitbot to test it out.

Here's an example for you:



This example will:

- Go forwards for 0.5s
- Stop for 0.5s
- Turn left slowly for 1s
- Go backwards for 0.4s
- Stop for 0.2s
- Spin right quickly for 0.3s
- Repeat over and over!

At this point you should have a bitbot that can move backwards, forwards, turn and stop, repeating forever because it's in the **forever** block.

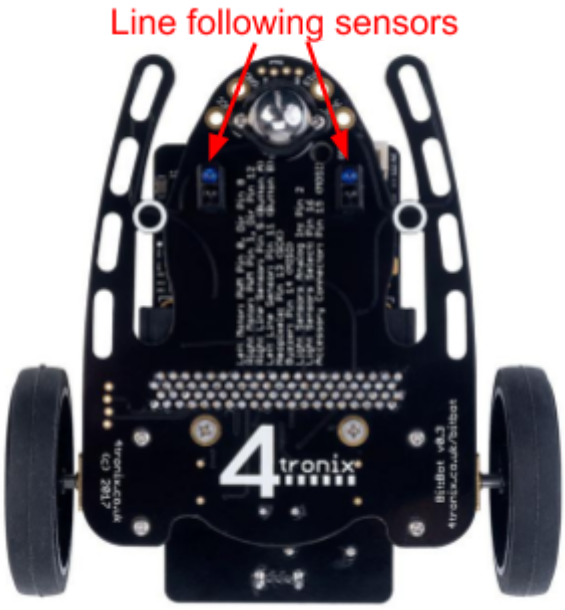
You should still have some lights showing in the **on start** block. If not, add some lights back in!

Now switch off the bitbot.

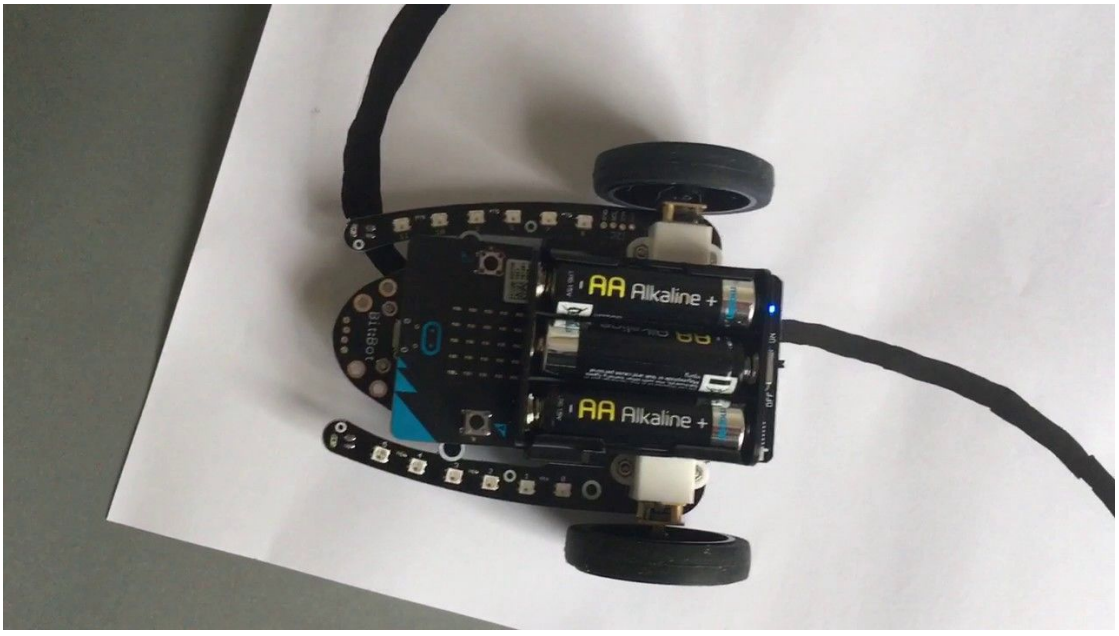
It's time to add some line following.

4 Line Following

On the bottom of your botbit is two line following sensors:



We're going to use this left and right line sensors to tell the botbit to follow a line like this:



4.1 If statements

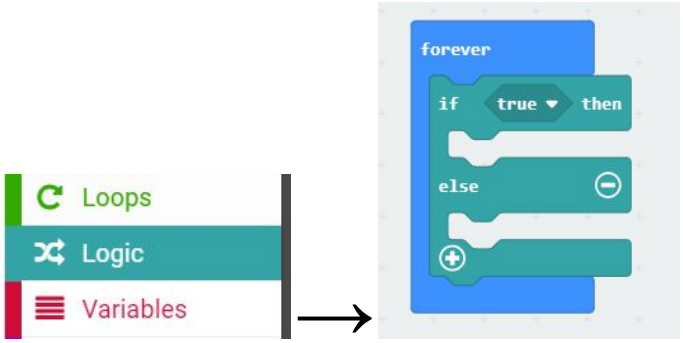
To follow a line, we need to do the following:

- **If** the left line sensor detects a line then spin left
- **Else If** the right line sensor detects a line then spin right.
- **Else** carry straight on.

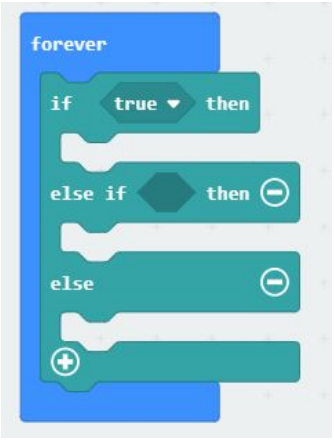
Micro:bit has a block for **if, else if, else** known as "if statements" - If this then do this!

Firstly delete any blocks inside your **forever** block, or start a new program:

And add an **if else** block into the **forever** block from the **logic** tab:



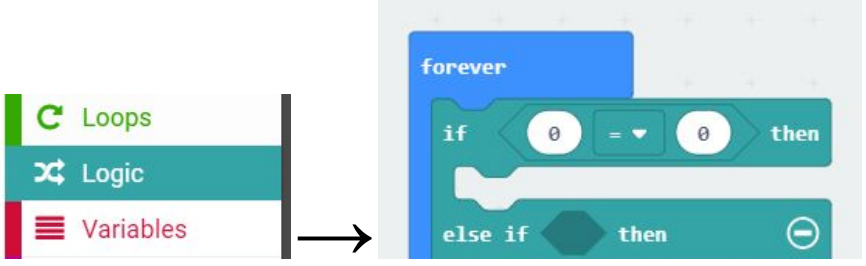
Click the plus button on the **if else** block so it becomes a **if, else if, else** block:



So firstly, we're going to do:
If the left line sensor detects a line then spin left

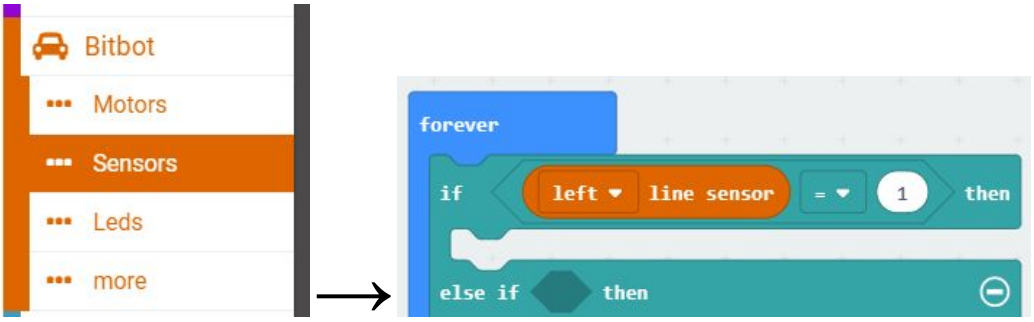
When the sensor detects a line, it gives a 1. So we want to do:
If the left line sensor = 1 then spin left

So to get the "=1" we want the **comparison** block from the **logic** tab:



And set the right number to 1.

The **line sensor** block is in the **Bitbot Sensors** tab. We want to set it to the left sensor:



Finally, we want to make the bitbot turn left if the left sensor = 1:

4.2 Else if, Else

So now you've done the left line sensor, can you do the right line sensor?

Remember we want to do:
Else If the right line sensor = 1 then spin right.

It should be very similar to the left sensor block code.

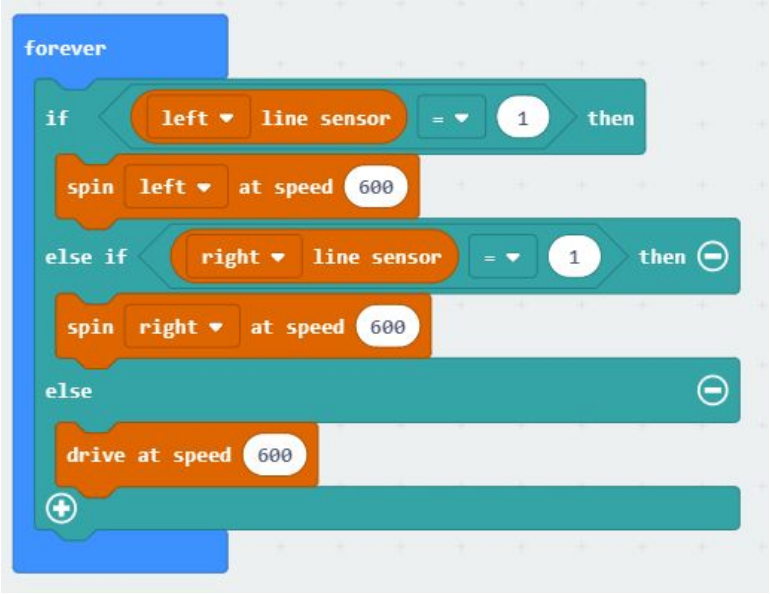
Can you code the final bit too?

We want to do:
Else drive forwards.
This one should be pretty simple.

Once you've finished your code, download it to the micro:bit and switch on the bitbot to test it out.

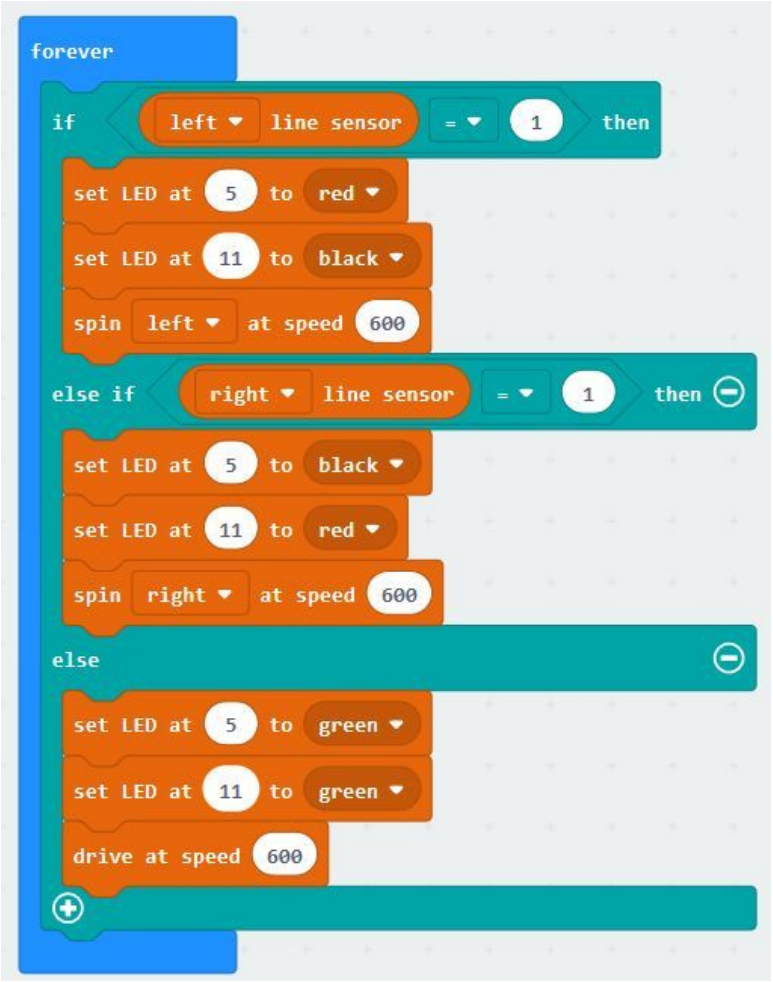
If you're stuck ask for help.
The solution is on the next page, but only look if you're really stuck!

Line following solution:



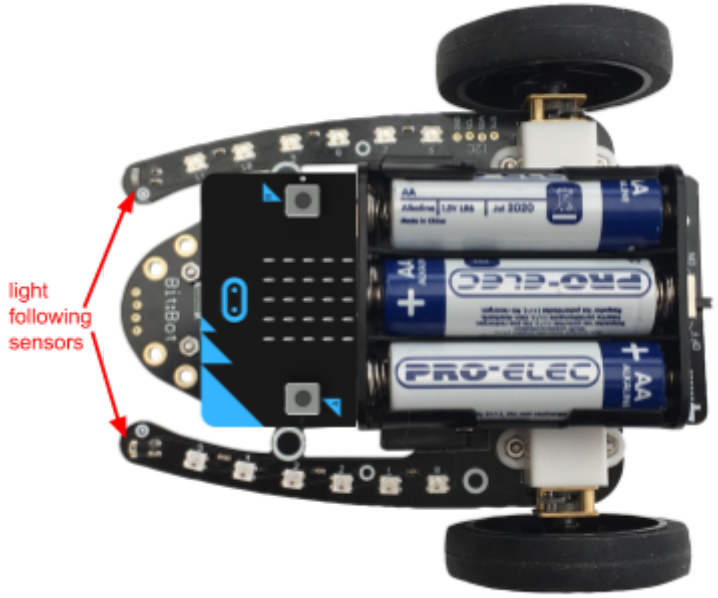
Hopefully your bit bot follows the line well! If not try tweaking the speeds of the motors.

Why not add LEDs to show when the bitbot is turning left and right?
Like indicators on a car! Setting an LED to black turns it off.
Make sure to remove other set LEDs from your code so it doesn't interfere with this.



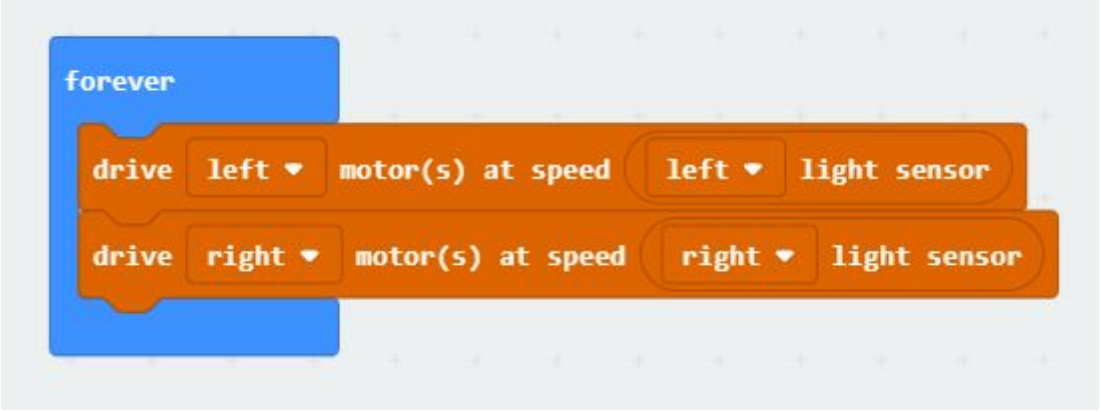
5 Light Following

Now we've done line following, how about light following? Just like the line following sensors, the bitbot has two light following sensors:



To do line following you want to measure the left and right light sensor and drive the motors individually - drive left motor at speed X and drive right motor at speed Y

So have a play around with this function:



Test out the code with the torch light from your phone, point it at each light sensor and see if the bot moves. It may not move much, so use the **math multiply** blocks to tweak how much the motors are driven to make this work nicely - as you point the light more to one side of the robot or the other you should see it follow that direction.