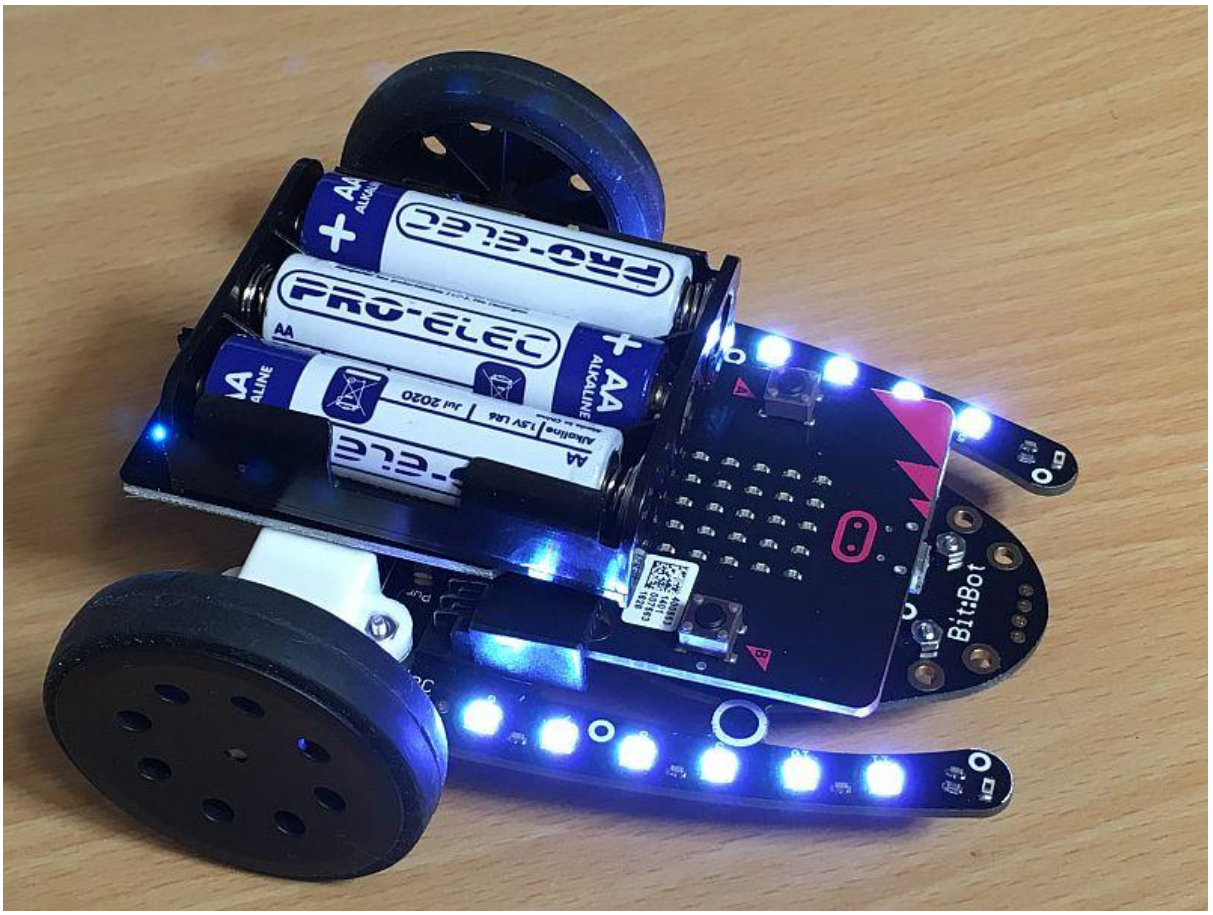
Bitbot Workshop - Robots and Micro:bit coding



By Joe Brown 17th July 2019

Updated 7th February 2023

Intro to robotics & block based coding.

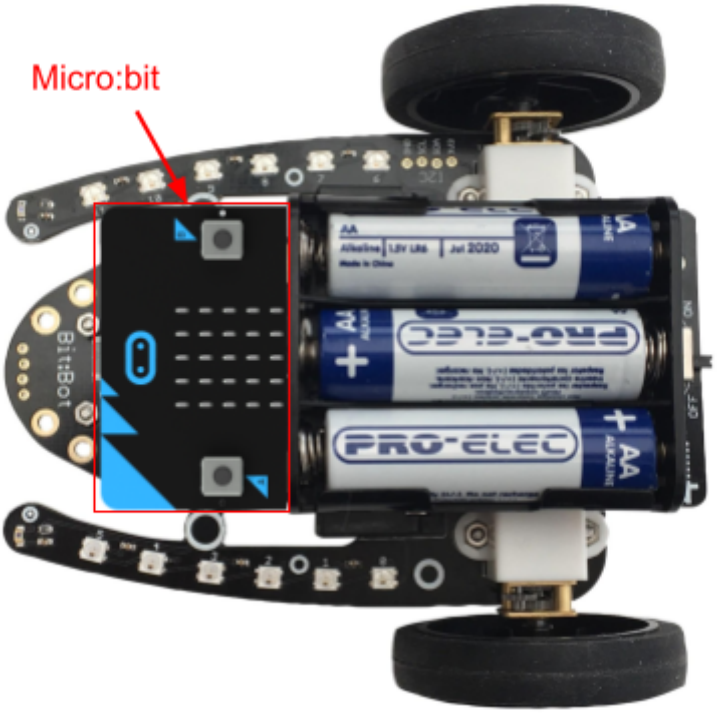
Learning outcomes:

Engineering skills - trial and error, problem solving, creativity
Electronics hardware - LEDs, buttons, motors, line sensors
Computer programming - loops, delays, if statements

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!

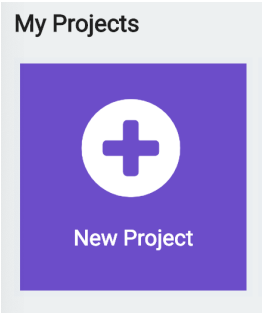


The Micro:bit is basically the brain of your robot.
You will be giving it commands by using blocks of code.

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:



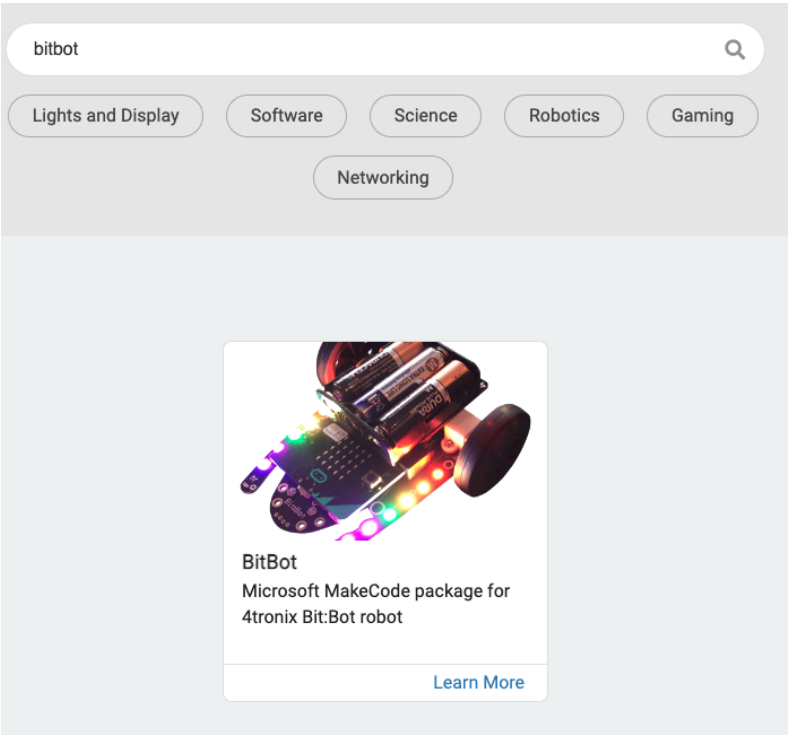
A new project will load.

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 **Extensions** tab in the Block Library:

Search for "bitbot" and
click on 'BitBot - Microsoft MakeCode package for 4tronix Bit:Bot robot'



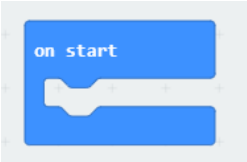
Now you're good to get coding!

2 Lights

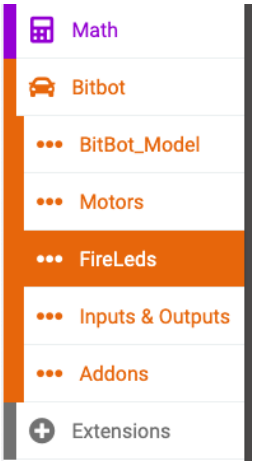
Let's do some coding and test the bitbot is working by turning on it's lights!

2.1 Coding the lights

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



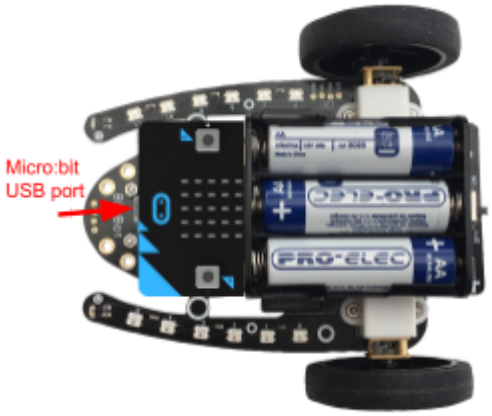
Click on the **Bitbot** tab and then on **FireLeds** (these 'Leds' are the lights on the Bitbot)



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

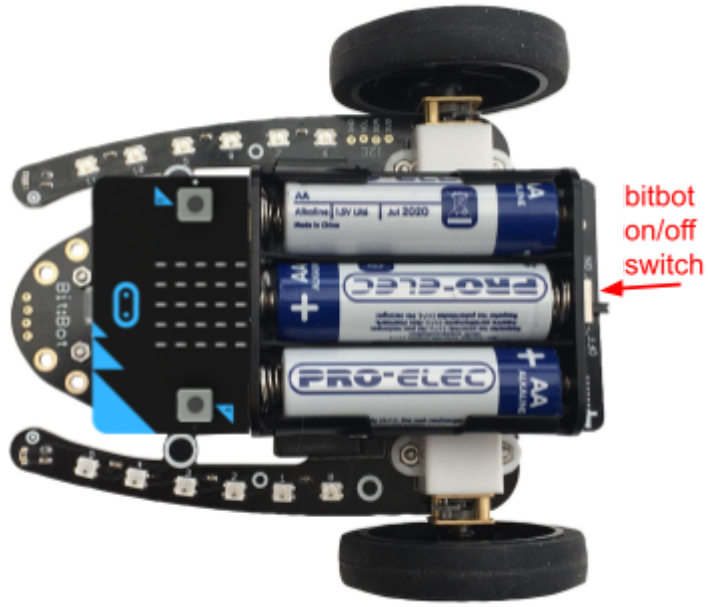
Let's test out our code! Plug the USB cable into the computer and into the micro:bit



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 your Downloads to MICROBIT in ‘My Computer like it’s a memory stick.

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!



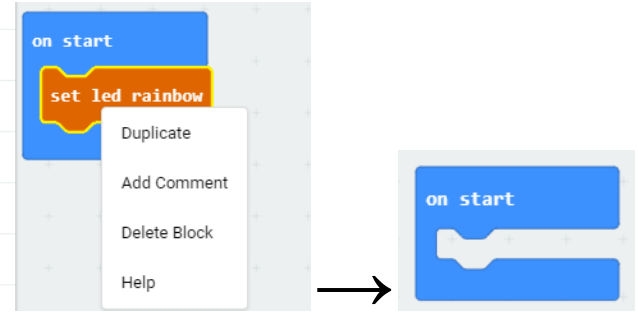
Hopefully when you turn on the bitbot you can see a rainbow of lights like so:

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.

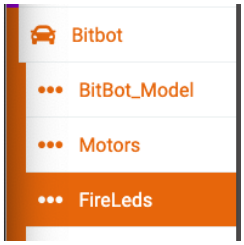
Now switch off the bitbot.

2.3 Customising the lights

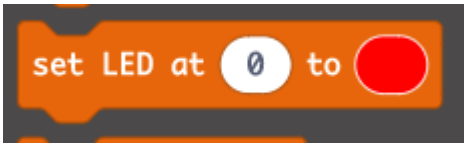
Now the lights are working, it's time to customise them!
Delete the **set led rainbow** block by right clicking on it and clicking 'Delete Block'



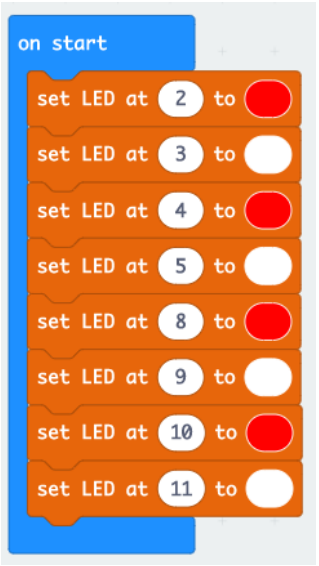
And go back to the **Bibot FireLeds** tab

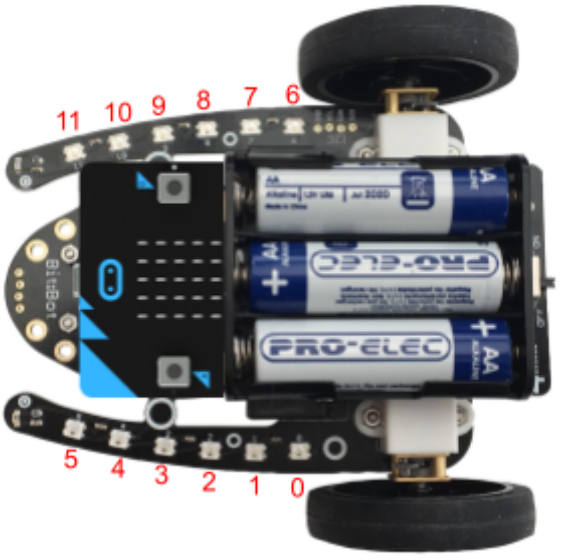


And add the **set LED at** 0 **to** red block:



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?

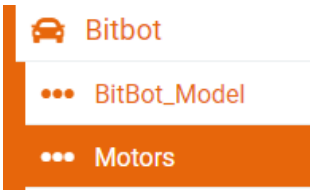
3 Movement

Now switch off the bitbot.

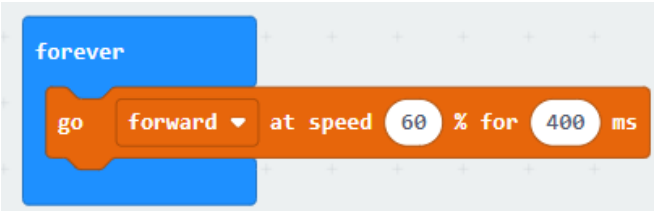
3.1 Forever Loop

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

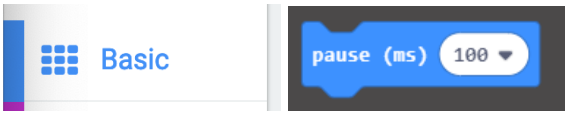
Blocks you put inside this **forever** block will repeat forever or 'loop'.
Let's test this out by getting your Bitbot to move.
Click the **Bitbot** tab and click the **Motors** section.



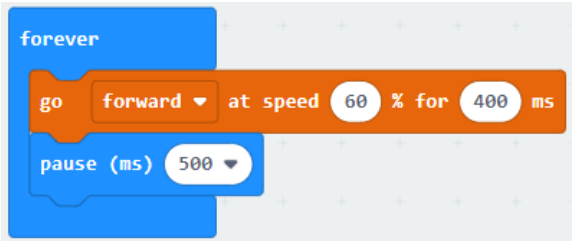
Click the '**go forward at speed** 60**% for** 400 **(ms)**' block, drag and drop it inside the forever block



If you didn't know 'ms' is short for 'milliseconds' or one thousandth of a second.
So 600ms is the same as 0.6 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.

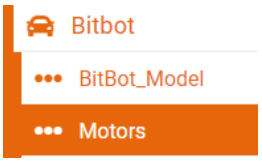
Once the USB light on the micro:bit has stopped flashing, unplug the USB cable and switch on the bitbot

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

3.3 More movement

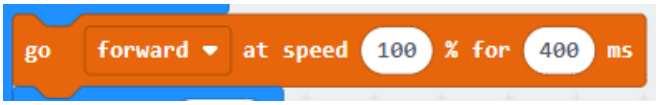
Now you've got your bitbot moving it's time to add some more movements.

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

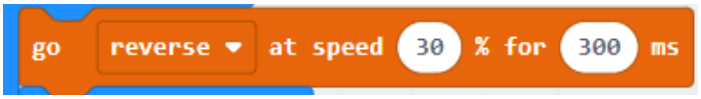


Here are some example movements:

Full speed forwards (for 400ms):



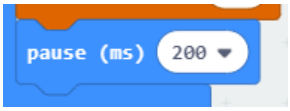
Set the motor to go backwards. Slow speed backwards (for 300ms):



Turn left full speed (for 200ms):

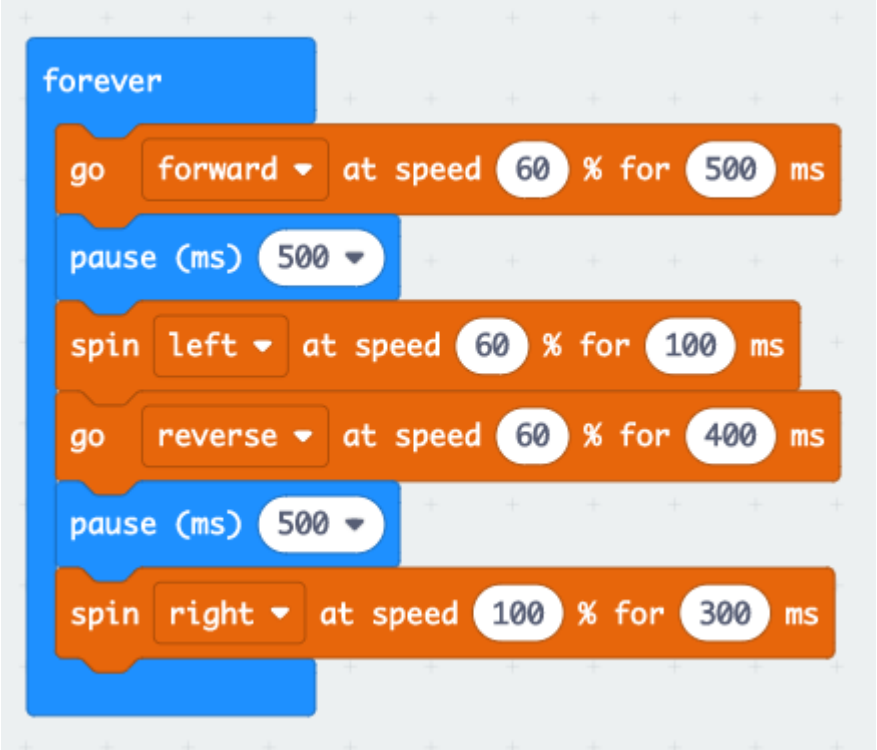
Turn right slowly (for 600ms):

Stop moving for (200ms):



Add a few different motor movements to your bitbot 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 for 0.1s
- Go backwards for 0.4s
- Stop for 0.5s
- 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 on from the **on start** block. If not, add some lights!

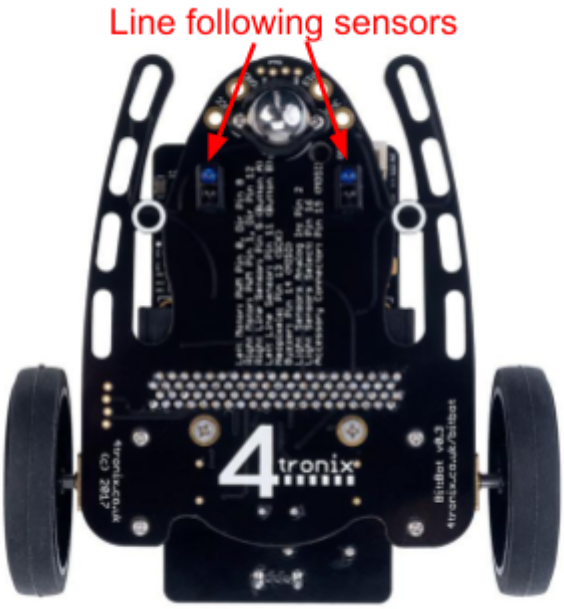
3.4 Optional: Bot Art

At this point the helpers may be able to provide you with a pen and paper so your bitbot can draw some art - the pen attaches to the back of the robot and leaves a trail of pen as it drives around! Try out some different movements and work out what will draw a good pattern as the robot drives around! Can you draw a circle, a square, a flower, a catherine wheel?

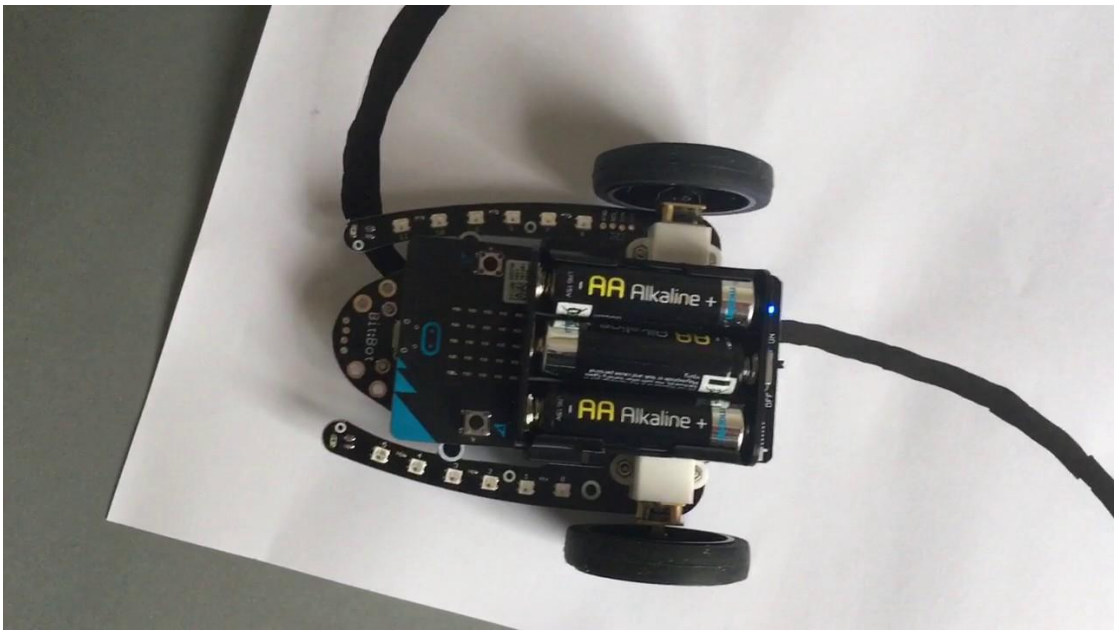
Now switch off the bitbot.

4 Line Following

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



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



4.1 If statements

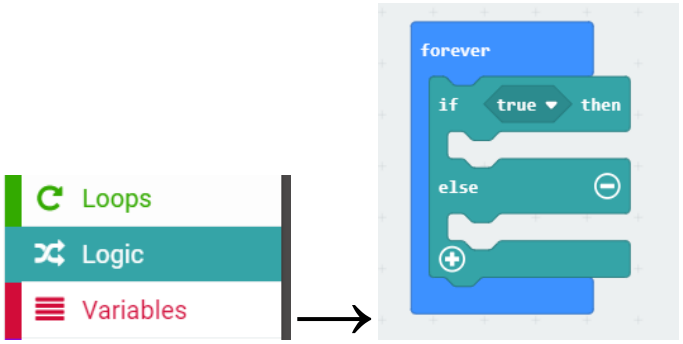
To follow a line, we need to do the following:

- **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.

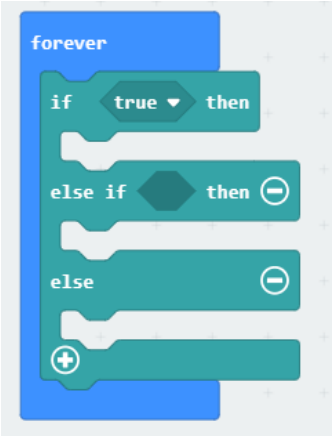
Micro:bit has a block for **if, else if, else** known as "if statements" - If this then do this!

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:



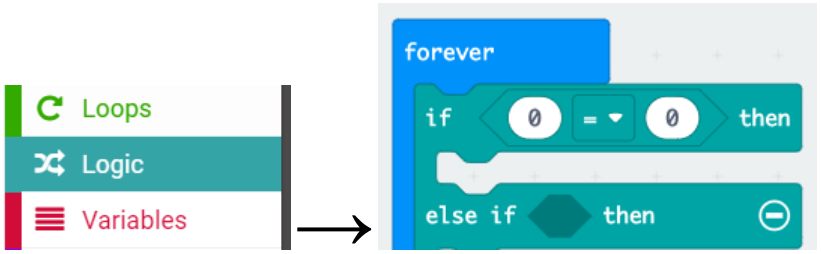
Click the plus  button on the **if else** block so it becomes a **if, else if, else** block:



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

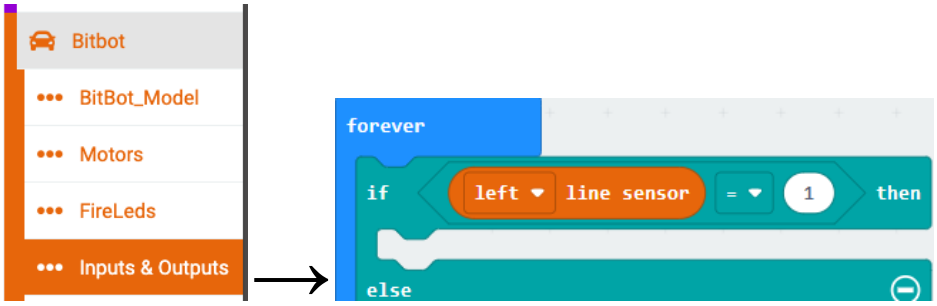
When the sensor detects a line, it gives a 1. So we want to do:
If the left line sensor = 1 then spin left

So to get the "=1" we want the comparison **0 = 0** block from the **logic** tab:

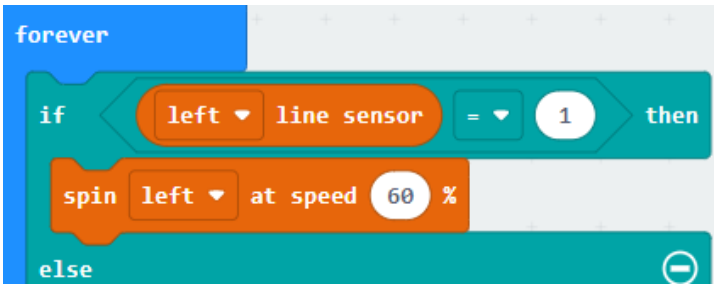


And change the 0 on the right of the block to a 1.

The **line sensor** block is in the **Bitbot Inputs & Outputs** 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.

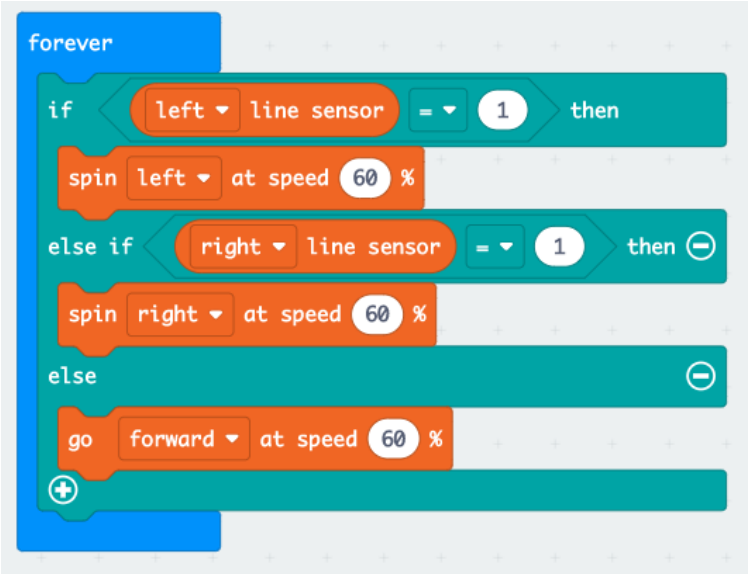
Can you code the final bit too?

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.

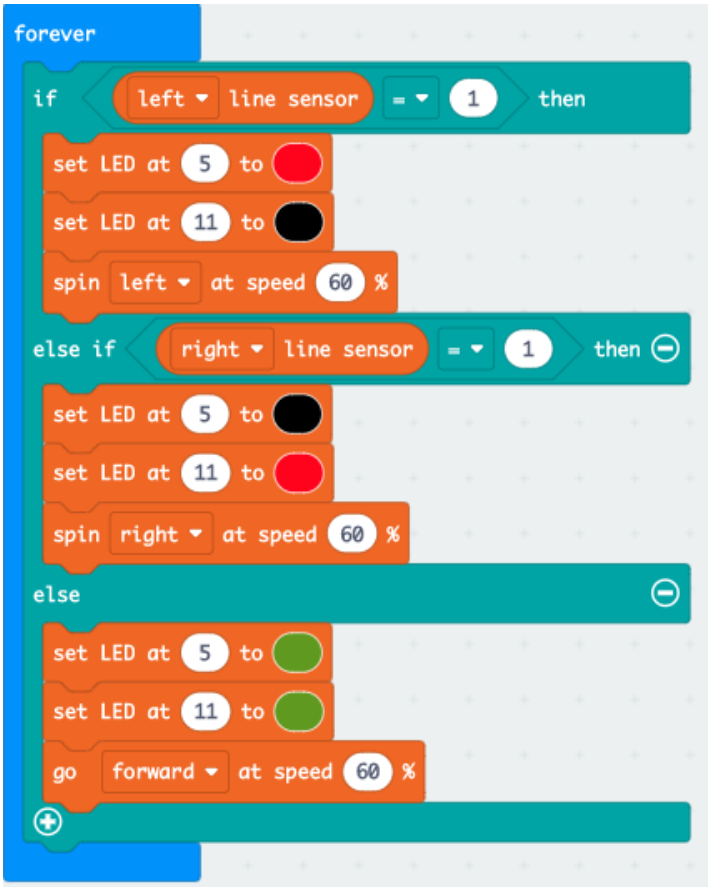
If you're stuck ask for help.
The solution is on the next page, but only look if you're really stuck!

Line following solution:



Hopefully your bitbot follows the line well! If not, try changing the speeds of the motors.

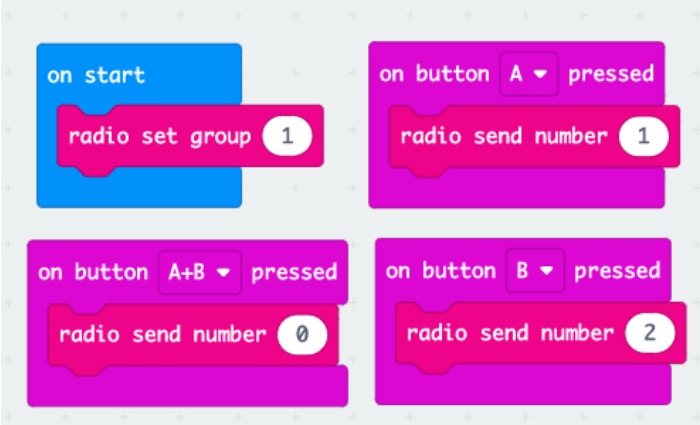
Why not add LEDs to show when the bitbot is turning left and right?
Like indicators on a car! Setting an LED to black turns it off.
Make sure to remove other set LEDs from your code so it doesn't interfere with this.



Try improving your line following - can you make it follow faster without going off the track?
Use a stopwatch - Who can make the fastest time round the track?

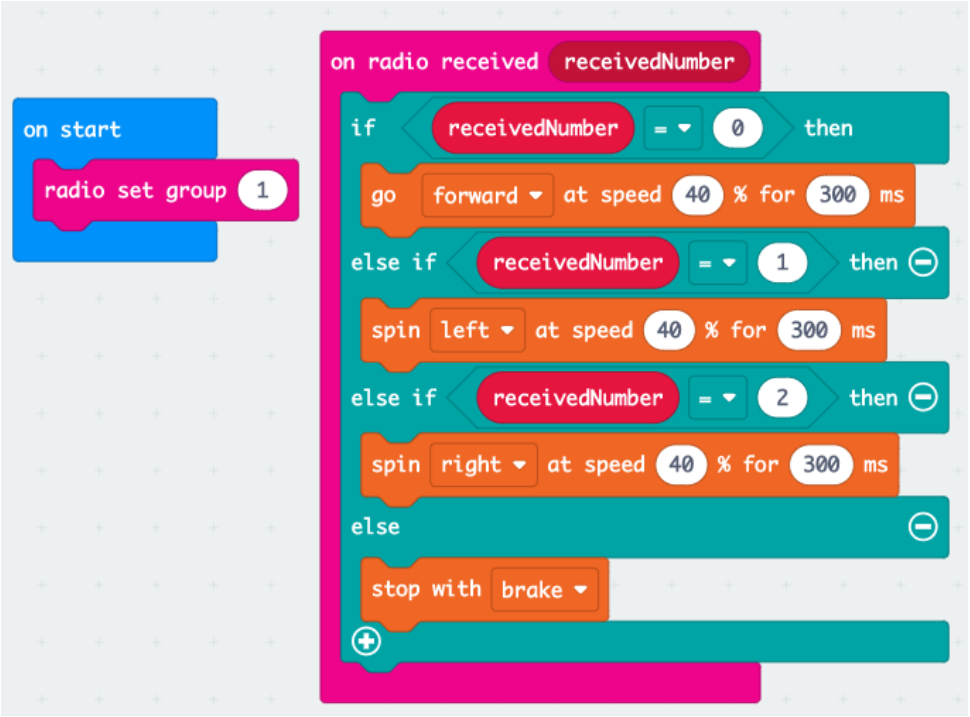
5 Extension: Remote Control

If you've finished the workshop with lots of time to spare this is a great extension project - using two microbits with their radio transmitters you can drive the bitbot using the buttons on a second microbit - ask one of the helpers for a second microbit for your remote controller.
For your **"remote controller" microbit** you'll want to make some code something like this:



Change the group number to one of your choice between 0-255 (you don't want the same group as anyone else in the class!)
Button A will turn left, Button B will turn right, both buttons together will drive forwards.
Plug in the new microbit and download this code.

Now plug in the **original bitbot microbit** and write the code for driving the "car", this is our suggestion, remember to set the radio group number to the same one as your remote.



This will give you a basic car that goes forwards left right in pulses. Can you improve the car code to make it more real time driving like a proper RC car? Maybe Add lights for turning?