School of Computing, Engineering and Intelligent Systems

Introduction

What is a Bit:Bot?

The **BitBot** is a hardware extension for the **Microbit** and it turns the **Microbit** into a small robotic car. It has two motors to drive the wheels and a steering wheel. By using two wheels we can control the direction of the bot. It also includes the capacity to add sensors and turn the **Bitbot** into a self-driving vehicle.

The **Bitbot** is a great way to show real world applications of code. It allows us to access and control new sensors and controls as it has motors, lights and sensors which, in turn, allows us to expand on the current limitations of the **Microbit**.

So, we are going to add some new extensions to our Makecode editor and create a robotic car which will traverse a maze.

Let's look at the Bitbot in more detail.

BitBot Specifications.

Here's a list of the main features:

- 2 Motors at the back of the board, can be controlled separately to determine speed and direction.
- Small sized ball to allow it to turn.
- Neopixels on 2 sides, 12 in total. These can be manipulated using the software to set them to light at different times, and determine what colour they light using RGB Values.
- 2 Wheels with Rubber tyres, improving grip.
- Buzzer to create sound other than that of the motors.
- Slot for the BBC Micro:Bit to connect to via its Edge Connector.
- 2 digital sensors for line following, so it can go round a drawn track.
- 2 analog sensors for light, so it can sense light.

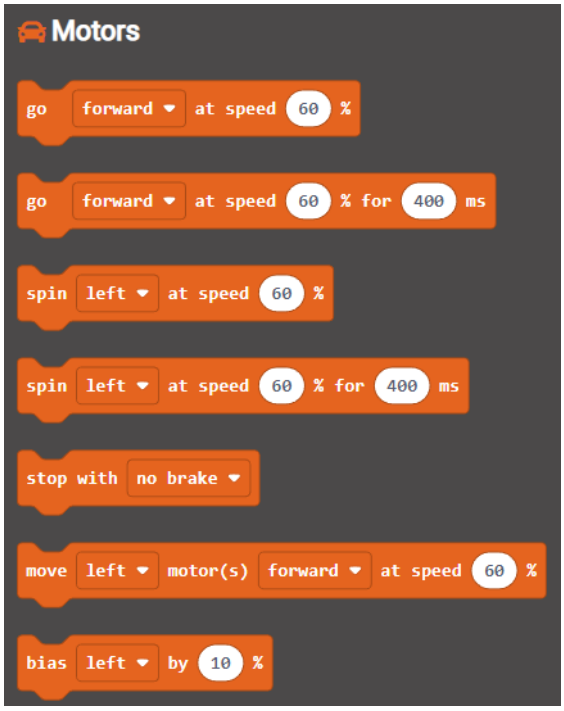
Different pins are used to control the different features:

- Left Motor: PWM Pin 16, Dir Pin 8
- Right Motor: PWM Pin 14, Dir Pin 12
- Right Line Sensor: Pin 5 (Button A)
- Left Line Sensor: Pin 11 (Button B)
- Neopixels: Pin 12 (SCK)
- Buzzer: Pin 14 (MISO)
- Light Sensors Analog In: Pin 2
- Light Sensors Select: Pin 16
- Accessory Connector: Pin 15 (MOSI)

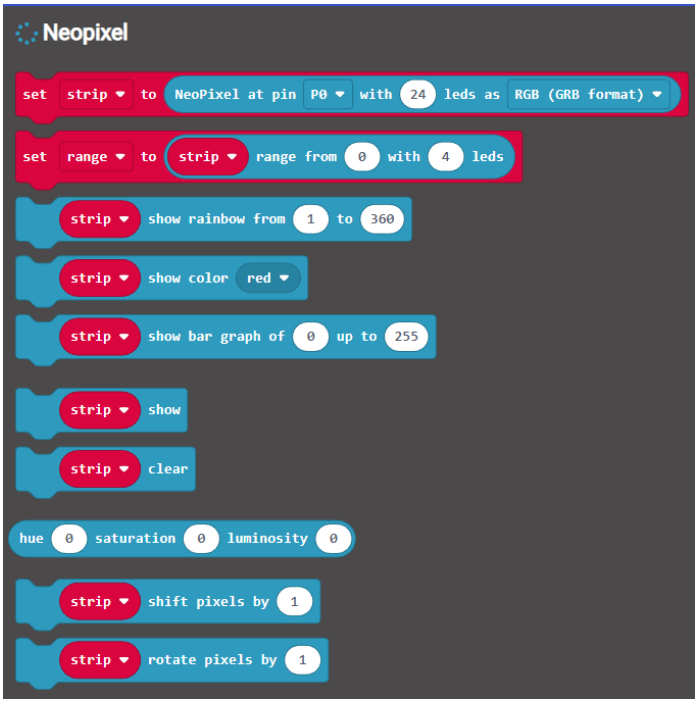
What do I need to do?

The first thing we need to do is add two new extensions to our Make code editor:

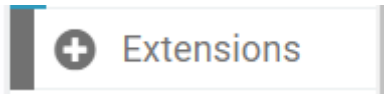
First of all, we are going to add the 4tronix **BitBot** extension which is used with the **Microbit**, this allows us to use the Blocks editor to create code which will control the **BitBot** motors.



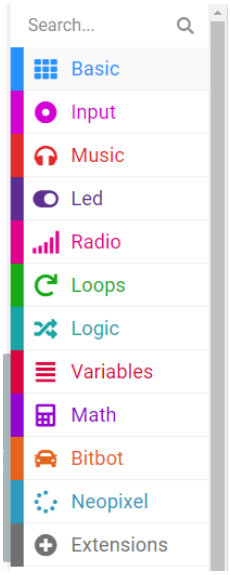
We are also going to use the **Neopixels** extension to have more control over the LEDs. This will allow us to customise our **BitBot**.



So go to Extensions on your code editor and download the two new extensions:



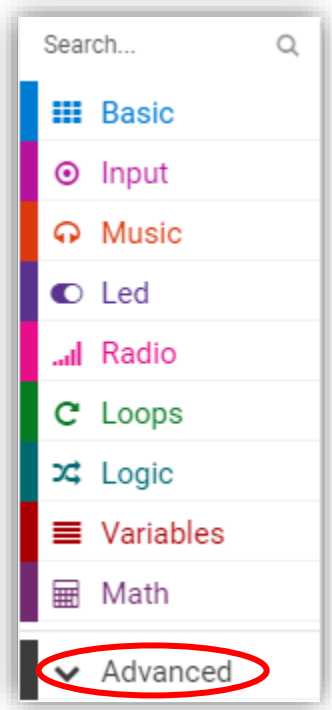
They will now show up on our toolbar:

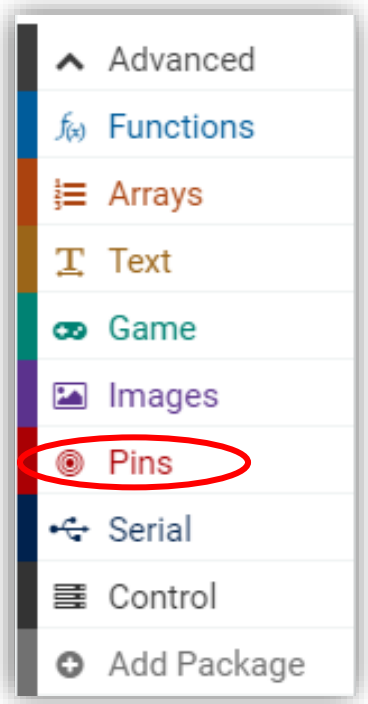


Getting the BitBot Moving

To get it on the move, values need to be assigned to the pins associated with the motors. This will tell the BitBot to go forward or back, what speed it should go and what direction it could go.

Using the Blocks editor, select advanced. Then select the Pin category, this is to access blocks which will control the motor pins.





To make the BitBot move, digital write command needs to be used, it will have a value of 0 or 1.

0 is for forwards and then 1 is used to reverse. The speed is set by analog write command as it sets the speed of the motor.

A speed between 0 and 1023 is used for the analog write command, 1023 being full speed and 0 being stopped.

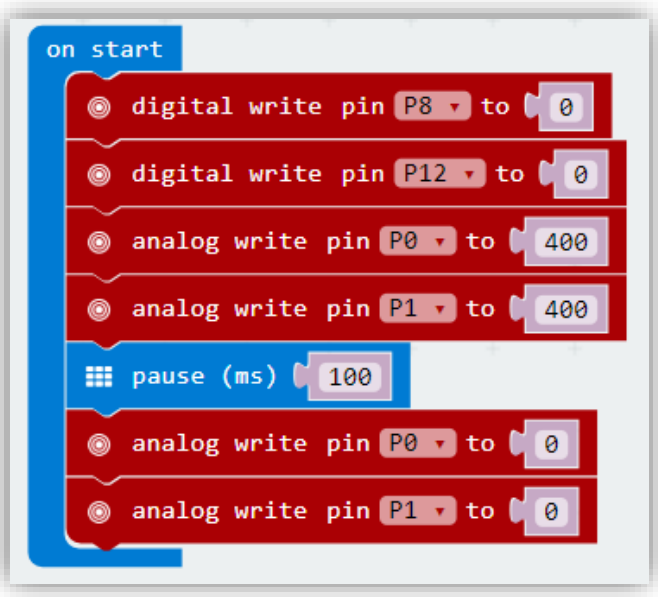
Step 1

The digital write command should be set to P8 and P12, P8 for the left motor to control the left wheel and P12 for the right motor to control the right wheel. The value can be set to either 0 or 1.

0 is forwards and 1 is reverse.

The Analog write commands should be set to Pin0 and Pin1, Pin0 for the left and P1 for the right. The Value should be set anywhere between 0 and 1023.

A Pause can then be used to stop the BitBot then reset its values to stop it moving.



Javascript:

```
pins.digitalWritePin(DigitalPin.P8, 0)
pins.digitalWritePin(DigitalPin.P12, 0)
pins.analogWritePin(AnalogPin.P0, 400)
pins.analogWritePin(AnalogPin.P1, 400)
basic.pause(100)
pins.analogWritePin(AnalogPin.P0, 0)
pins.analogWritePin(AnalogPin.P1, 0)
```

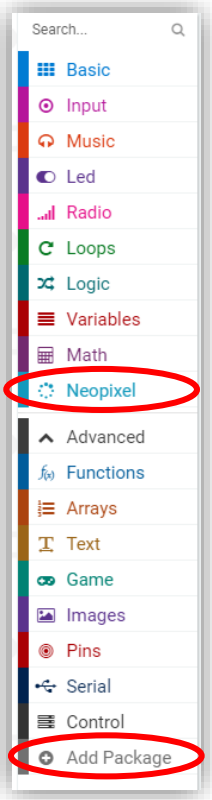
Next try to get it to change direction and pause then go another direction.

Change its speed to see how fast it can go and what the minimum speed is.

Another possibility is to get it to start off and see if you can get it to go round in a full circle or around an object and return to its starting position.

Using the Neopixels

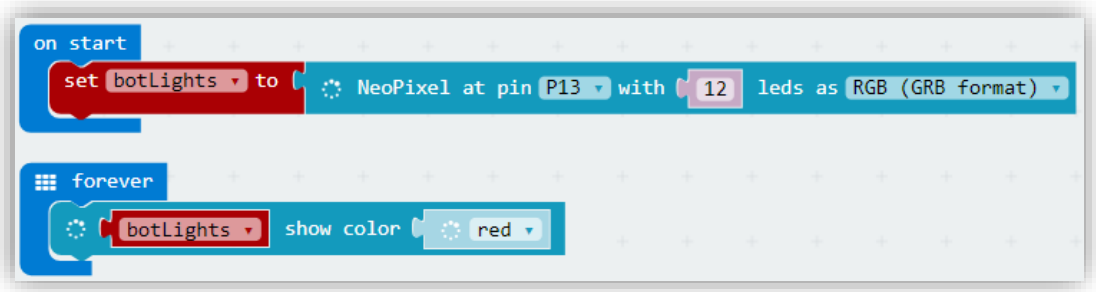
In order to use the Neopixels, a Neopixel package must be added to block editor. This can be done by hitting the Add Package button at the bottom of all the existing categories. Once you select this, type in Neopixel and then click it to add it as a new blocks category. It should then appear below the Math category.



The BitBot has 12 Neopixels, 6 on each side of the board. To allow it to light all of these Neopixels at once, a variable can be used and an RGB value can be assigned to it so all the Neopixels light up the same colour at one time.

To do this, go to basic and get an on start block and a forever block, then variables and select a set item to block. Change the variable name to something which will indicate that it's being used to control the Neopixels e.g. botLights. Then go to Neopixels package and get the Neopixel at pin block, this is connecting the variable to pin13 which is the Neopixel control pin.

Then to set the colour, drag item show colour in then change thee variable to the one which you named previously. Attach this to the forever block of code and then run your code.



JavaScript:

```
let botLights: neopixel.Strip = null

botLights = neopixel.create(DigitalPin.P13, 12, NeoPixelMode.RGB)

basic.forever(() => {

    botLights.showColor(neopixel.colors(NeoPixelColors.Red))

})
```

Individual colours can be set on each Neopixel rather than doing them all together, and the brightness of each can be changed using the more part of the category which shows up just below the Neopixel text.

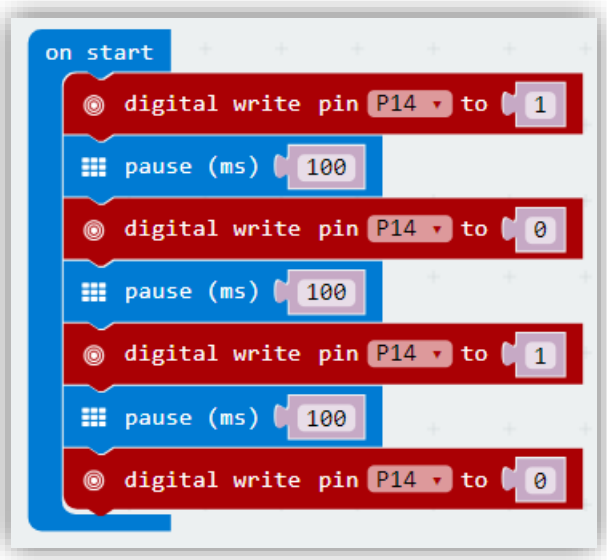
JavaScript:

```
let botLights: neopixel.Strip = null
botLights = neopixel.create(DigitalPin.P13, 12, NeoPixelMode.RGB)
basic.forever(() => {
    botLights.setPixelColor(4, neopixel.colors(NeoPixelColors.Yellow))
    botLights.setPixelColor(6, neopixel.colors(NeoPixelColors.Indigo))
    botLights.setPixelColor(8, neopixel.colors(NeoPixelColors.Purple))
    botLights.show()
})
```

Beeping the Buzzer

The buzzer is a small part of the BitBot, pin14 needs to be set to a value of 1 for it to play any sound using the digital write command.

It could be used to signal the start and end of the objectives of your code, for example if the BitBot is going round an object it could play the buzzer to signal a start then play it again to signal an end. You could use it to make a sound for certain movements, almost like an alert to anyone near than something is moving closer or farther away.



JavaScript:

```
pins.digitalWritePin(DigitalPin.P14, 1)
basic.pause(100)
pins.digitalWritePin(DigitalPin.P14, 0)
basic.pause(100)
pins.digitalWritePin(DigitalPin.P14, 1)
basic.pause(100)
pins.digitalWritePin(DigitalPin.P14, 0)
```