Kavita was looking through the input palette inside the MakeCode editor, and found lots of other exciting input blocks. She found one that says 'light level'.

This gave her the idea of building a two-handed glove project that used light level to vary the tone as she moves her hand up and down. She's not quite sure how to make this work though, can you help her with the code? In today's lesson you will help Kavita write a program for her glove, called the 'light glove'.

By doing this, you will use many things you have already learnt about in MakeCode, including:

- ❑ event handlers;
- ❑ variables, including boolean variables;
- ❑ if statements;
- ❑ functions;
- ❑ passing parameters to functions;

You will also learn:

- ❑ how to return a result from a function;
- ❑ how to map a range of numbers to a new range;
- ❑ how to round a number to the nearest digit;
- ❑ and of course, you will learn a little bit more about open source software.

You will need:

- ❑ your assembled MiniMU glove;
- ❑ the MakeCode web coding editor..

You might also need a pair of scissors, or a needle and thread.

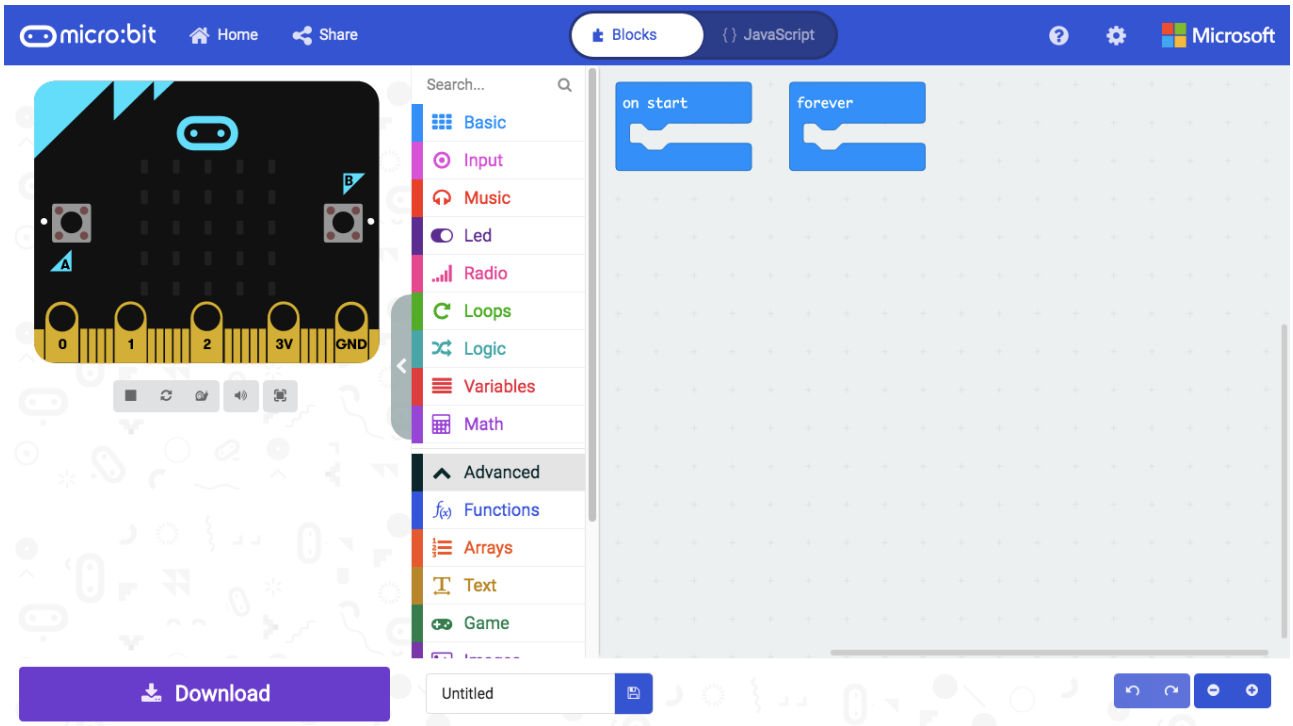

Figure 1: The MakeCode web coding editor.

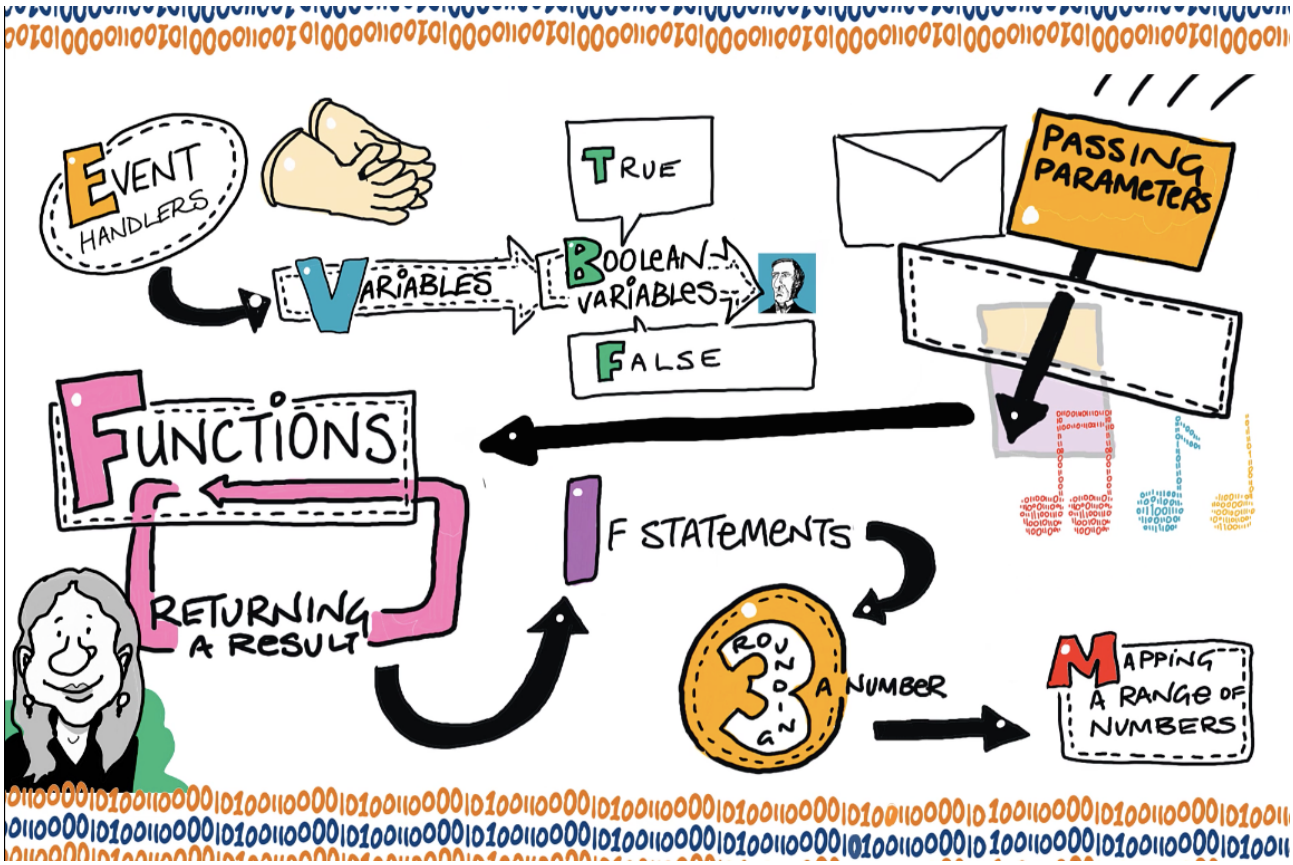

Figure 2: What you will learn this lesson (as a poster).

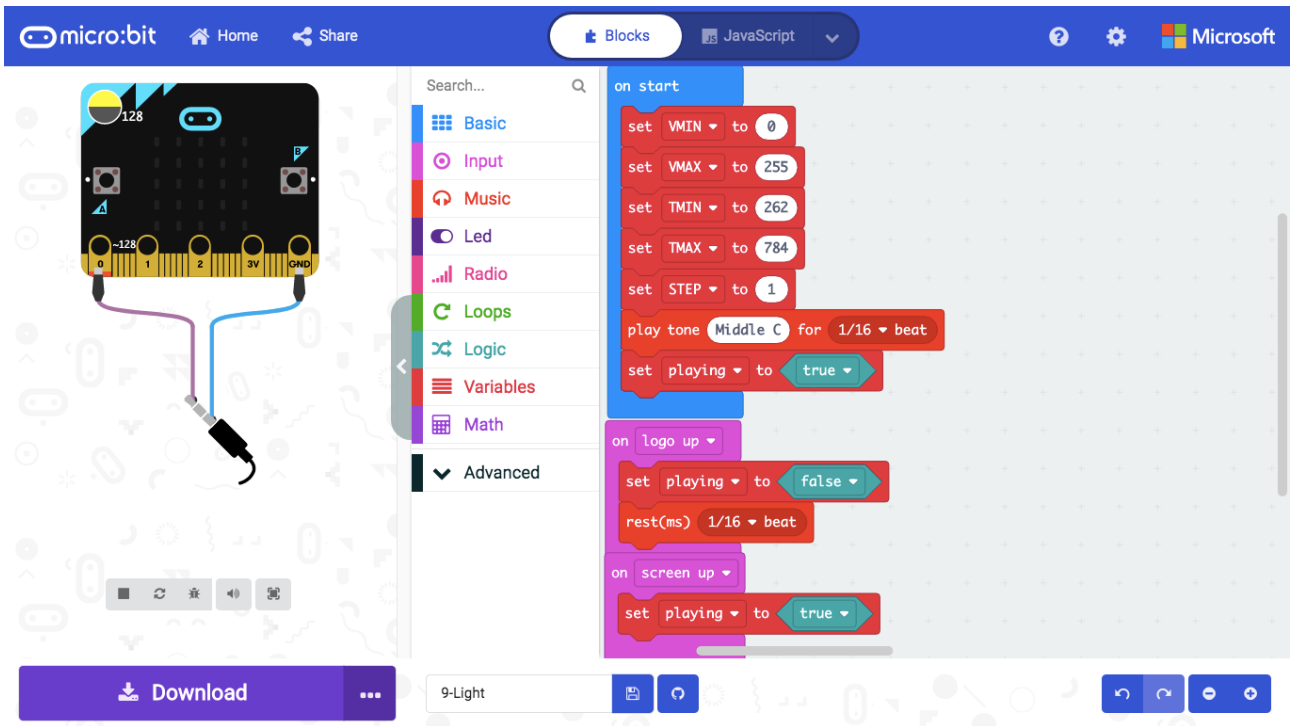

Figure 3: OnStart with all the constants.
OnLogoUp (stops the sound); OnScreenUp (starts the sound).

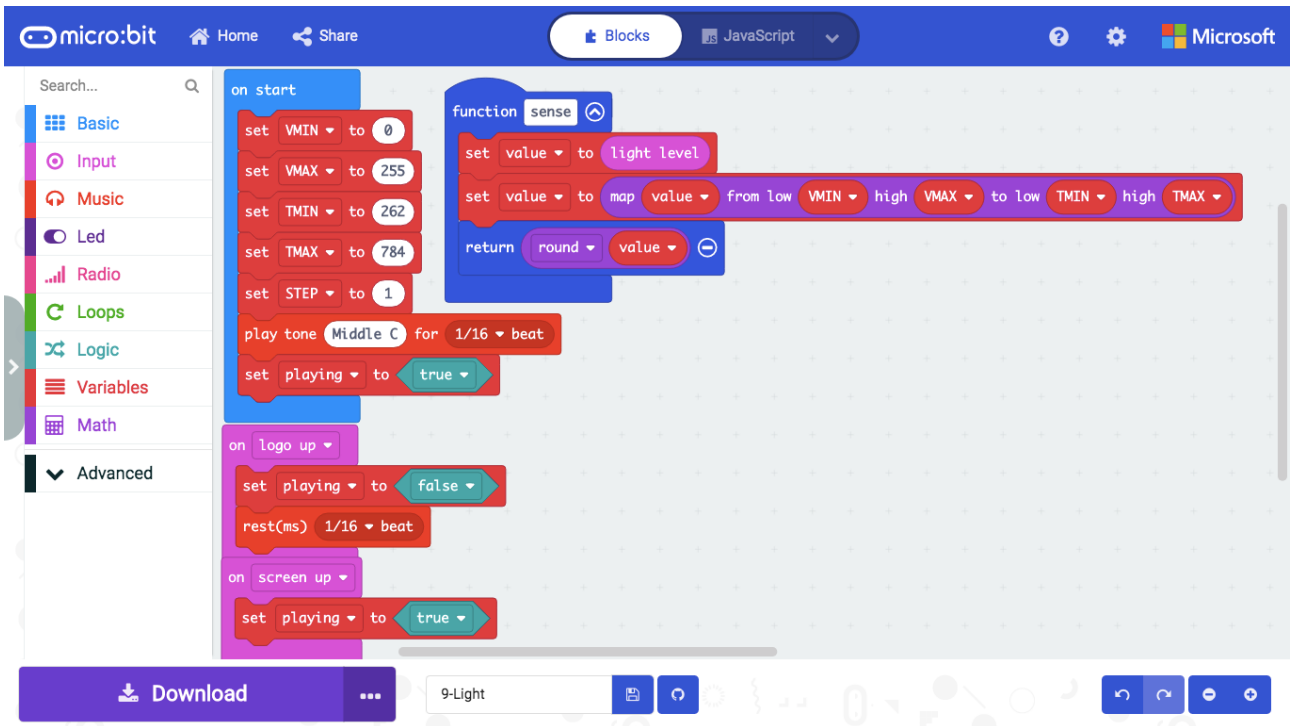

Figure 4: The 'sense' function reads the light level.
It also uses 'map' to convert to a musical tone.
Note the use of the 'return' and 'round' blocks here.

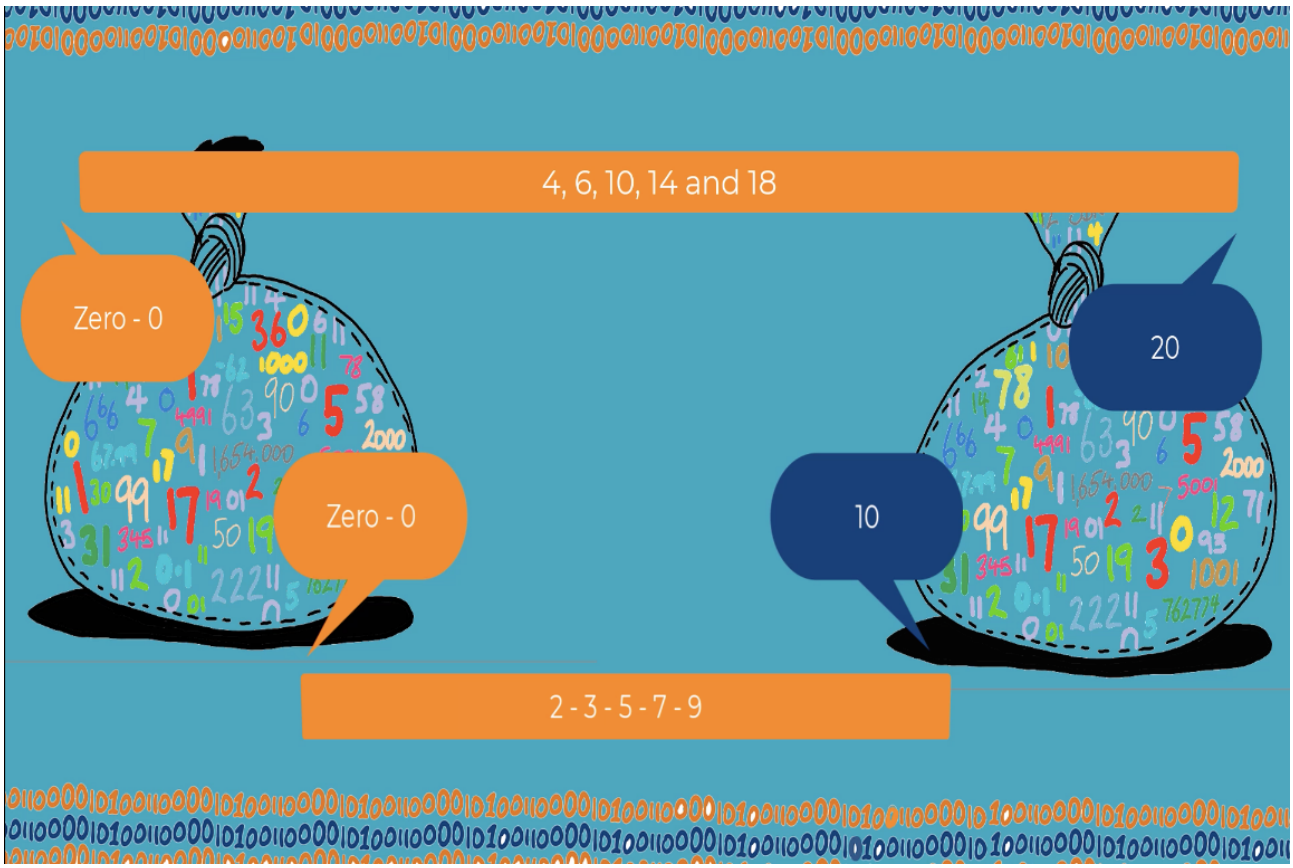

Figure 5: E.g. of a mathematical 'map' between two number bags. In this example, 'map' will multiply by 2*

* Super work, Mr Illustrator!

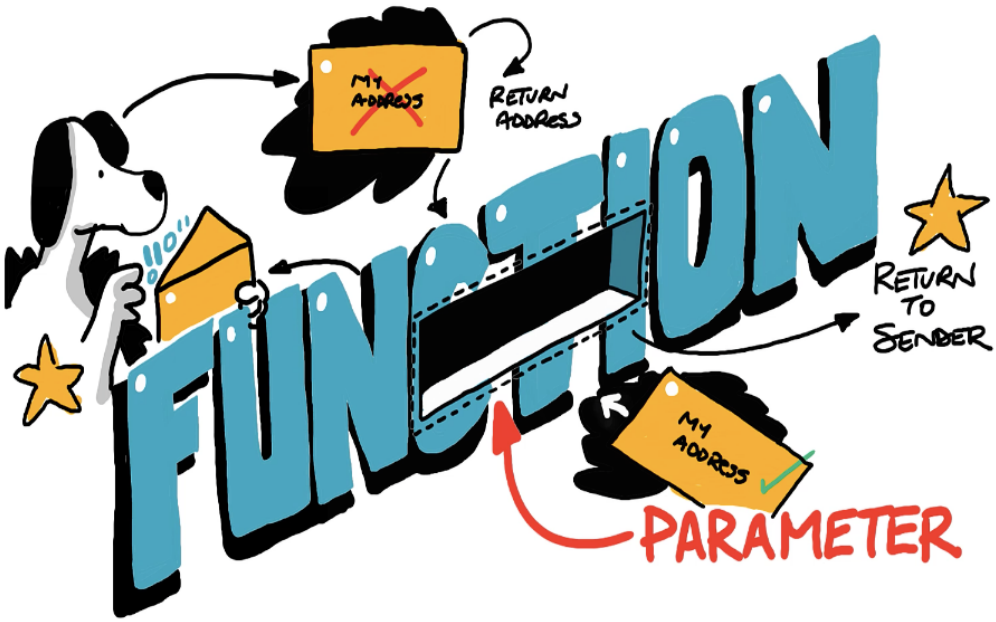

Figure 6: A 'return' is like sending a reply back to the sender*

* See later, you'll learn how your Nan can get involved too!

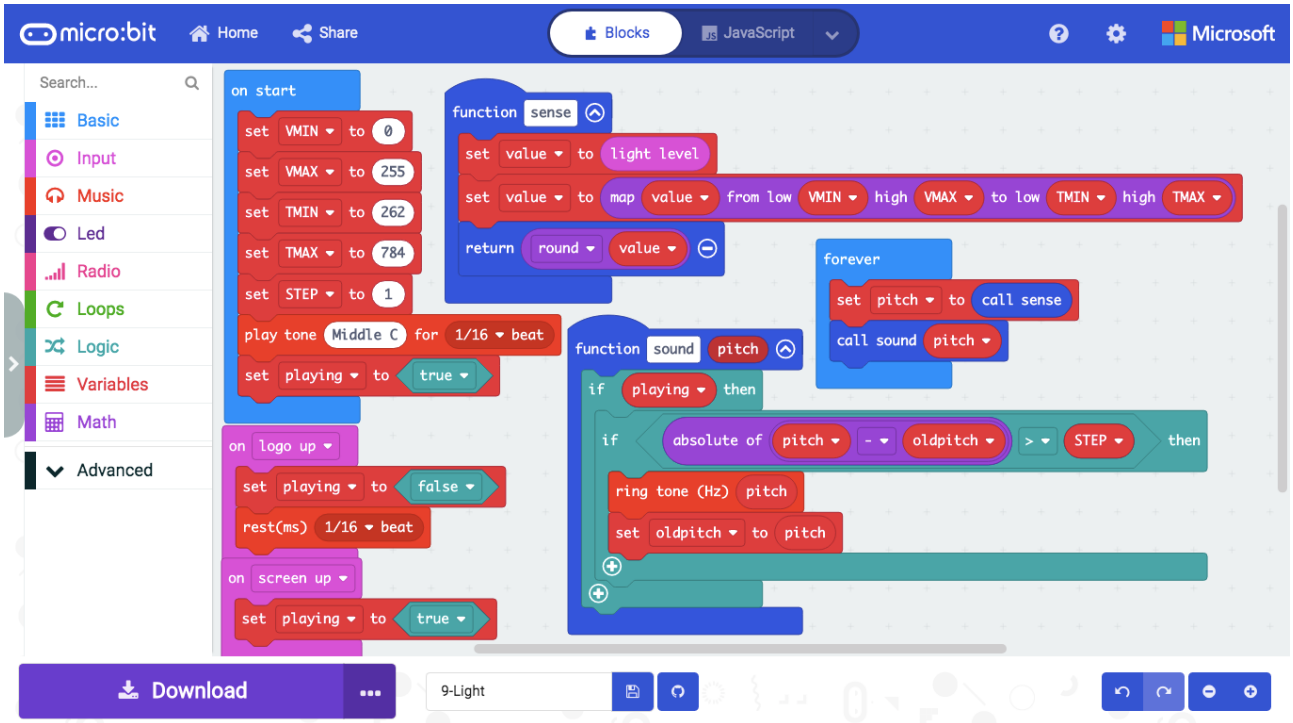

Figure 7: The 'sound' function prevents clicky sounds.
Detecting a change in a value, is a common programming pattern.

Figure 8: Andy's YouTube channel might be handy again!
https://www.youtube.com/watch?v=KOZFWeCic2s

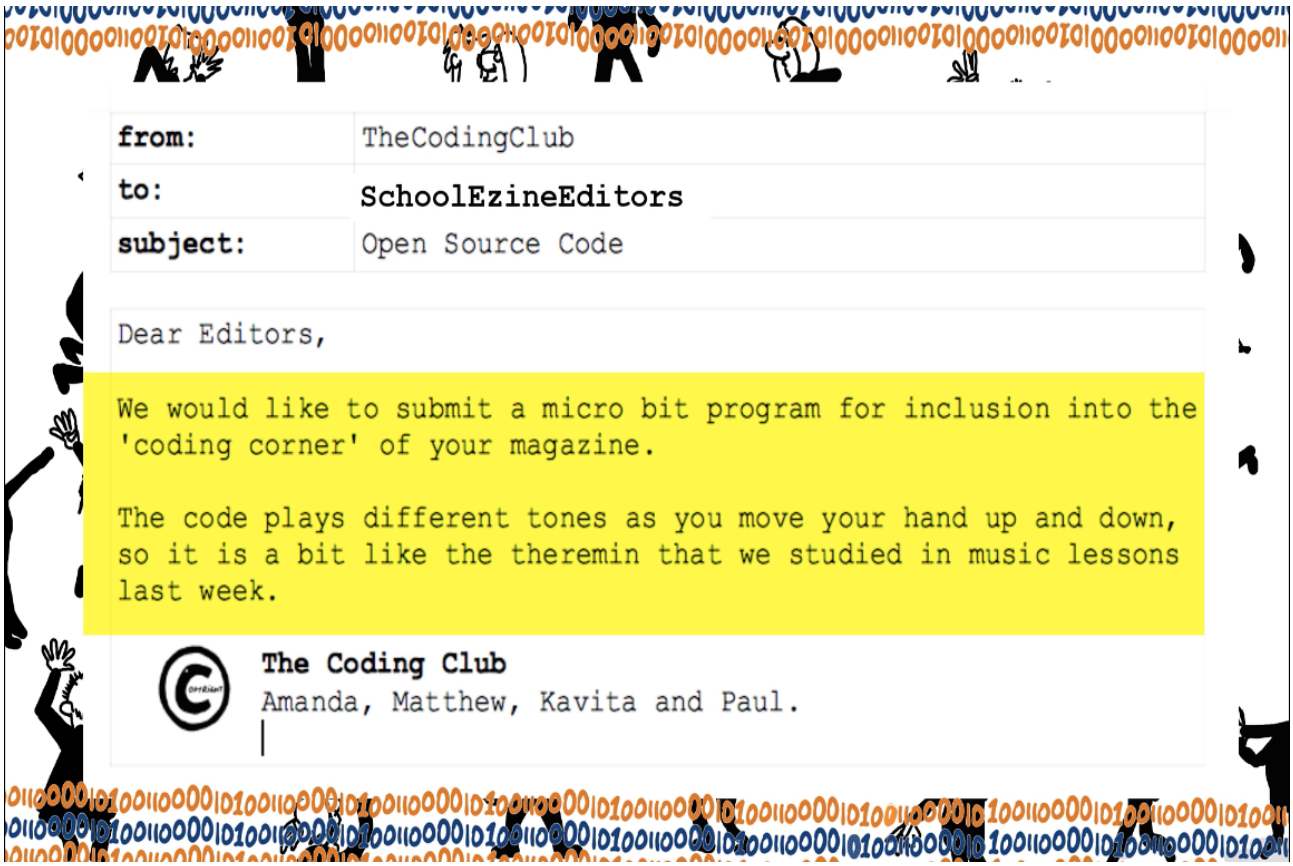

Figure 9: Sharing our code via email with a magazine*

An e-zine is an electronic magazine. But you knew that, right?!

Figure 10: Notice how we always signpost the licence at the end of all of our video lessons.

Additional Information

1. The Round function

'round' is a mathematical operation, that removes the fractional part of a number. So, the number 3.2 would become 3

You may already know how to round numbers from your maths lessons. A digit to the right of the dot of 0 to 4 would round down, and a digit to the right of the dot of 5 to 9 would round up.

So, 3.4 rounds down to 3, and 3.5 rounds up to 4.

The micro:bit uses floating point numbers, which means that the decimal point can 'float' left and right and the micro:bit can represent very small numbers (like 0.0000000001) and very big numbers (like 10000000000).

The original version of MakeCode did not support floating point numbers, it only supported integer numbers. An integer is a positive or negative whole number, like 3 or -5 and it has no decimal point. For integer arithmetic, numbers will automatically round to the nearest whole digit.

The micro:bit Foundation did a lot of research and a lot of talking to teachers and students, and it was decided that using floating point numbers like 3.2 would be better. It would make the micro:bit device easier to use in maths and science lessons, where a wide range of high precision numbers with fractional parts could be used for all sorts of things like temperature, speed, distance, etc. The floating point code was added into MakeCode in a later release, and this does use up some of the micro:bit memory, but it makes the device easier to use.

2. The Magic Number 255

In the video, Stephanie wondered what was important about the strange number 255, when she was talking about the VMIN and VMAX constants.

The light sensor on the micro:bit will return a number in the range of 0 for darkest, and 255 for lightest. Why 255? Why not 100 (like a percentage)?

The micro:bit light sensor software processes readings as a single byte. A byte contains 8 bits (or 8 binary digits). If you line up 8 binary digits together, you end up with a range of numbers from:
00000000 to 11111111

Humans count in 10's as we have 10 fingers, but computers count in 2's as they have 'on' and 'off' (2 states of a digital electronic circuit). So, every time we append another binary column, we double the range of values that can be stored in the memory location.

A single binary digit (with values of 0 and 1) gives you 2 possible values.

2 binary digits give you 00 01 10 11 which is 4 possible values.

8 binary digits give you 2*2*2*2*2*2*2*2 = 256 possible values. Because 00000000 represents zero, those 256 possible values count from 0 (00000000) to 255 (11111111). So, for a single byte of computer memory, it can store the values 0 to 255.

3. Ask Your Nan!

Mr Illustrator loves sneaking in little visual jokes, doesn't he? Did you see this one? Do you know who this person is? Ask your Nan, she will probably know!

It turns out (and Mr Illustrator didn't know this when he sneaked this joke in) that Nans and micro:bits are quite important!

On the 10th March 2016, the BBC ran a small 30 second video segment on prime-time TV called 'BBC micro:bit predictions'. In this video, children of the target age group who were about to be given (for free) a BBC micro:bit device, predict what the future of technology will look like. One of the predictions is about Nans and Holograms – take a look, here!

https://www.youtube.com/watch?v=-y0Yp6W3rP8

(You will see in the credits of this YouTube video that this video was uploaded with the permission of the advertising agency that created this video segment – so it is correctly *attributed* to the original authors.)