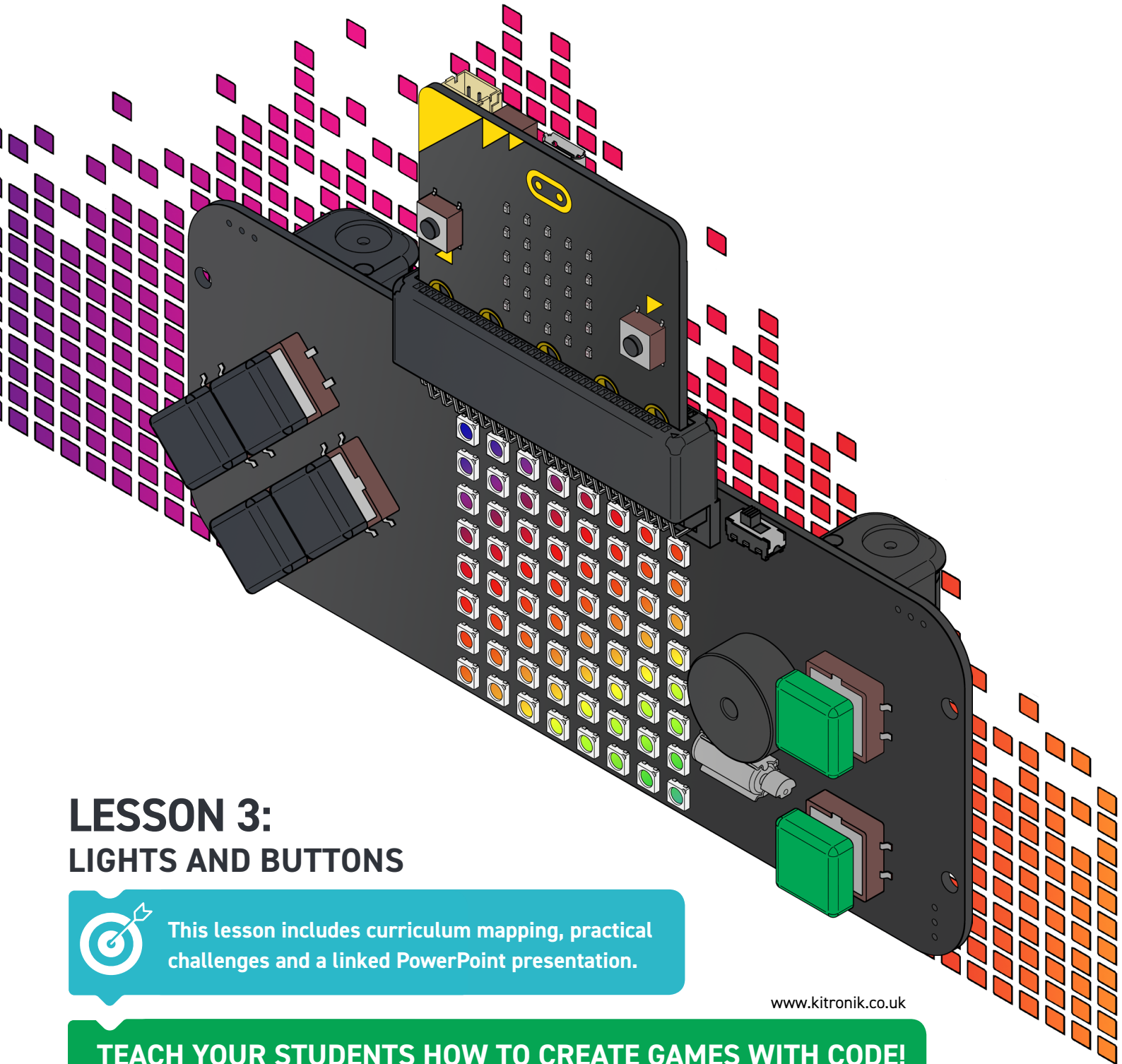


THE TEACHERS LESSON GUIDE TO THE :GAME ZIP 64



LESSON 3: LIGHTS AND BUTTONS



This lesson includes curriculum mapping, practical challenges and a linked PowerPoint presentation.

www.kitronik.co.uk

TEACH YOUR STUDENTS HOW TO CREATE GAMES WITH CODE!

LESSON 3

LIGHTS AND BUTTONS



This is the third lesson for students in Key Stage 3 to the :GAME ZIP64 controller. This lesson involves combining the skills learnt from the previous two lessons together. The students will use the buttons to move a light around the controller. At the end of the lesson the students will create a simple game. Recommended ratio of students to :GAME ZIP64 is 2:1.

Classroom setup

Students will work be working in pairs. They will need:

- A computer/laptop with a USB port and Internet access
- A :GAME ZIP64
- A BBC micro:bit
- 3 x AA batteries
- A micro USB cable

The teacher will be writing on the board as well as demonstrating code on a projected board (if available).

Timings

The lesson is expected to take 1 hour. It can be shortened or lengthened if needed.

Suggested shortening

In the lesson students start coding before you point out a bug in the code to them. You could skip this step and give them the final working algorithm on slide 8.

They also start coding before realising there are boundaries. Again you could skip this step and give them the final working algorithm on slide 10.

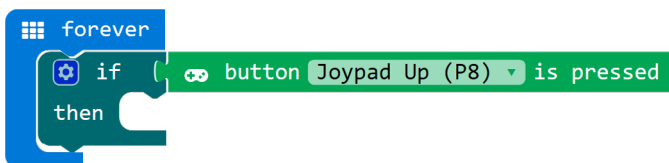
Suggested lengthening:

The final game is quite complicated and long. If students have completed it before the end of the lesson you could challenge them to add extra lights and add extra points, e.g. a green light is 5 points and a blue light is 10 points.

Differentiation

For younger or less able students there is differentiation in the lesson plan.

The controller can be controlled using the NeoPixel and digital read blocks described below. There are simpler custom blocks available to use. See www.kitronik.co.uk/blog/game-zip-64-makecode-blocks/ for more details.



KEY



Teacher asks this question to the class and discusses the answers with them.



Where this content maps onto the curriculum.



Teacher writes the text in this box on the board.



Students take notes, either on paper or in their electronic workbooks.



This part of the lesson aligns to Slide 1 of the attached PowerPoint.



Bug Alert - Check for problems.



LIGHTS AND BUTTONS

INTRODUCTION



This week we're going to look at combining the buttons and the lights on the zip controller. You will continue to work in pairs (reminder of pairs if you needed).

Remind students of the buttons and their pins:



Button	How it's connected
Joypad Up	P8
Joypad Down	P14
Joypad Left	P12
Joypad Right	P13
Fire SW 1	P15
Fire SW 2	P16



Students should have these in their notes/workbook already.

Let's look at the lights more closely.



0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63



Which of these pieces of code will turn light 0 on?



```

on button A pressed
  matrix show color red
  
```

```

on button B pressed
  matrix set pixel color at 0 to red
  matrix show
  
```


LESSON 3

LIGHTS AND BUTTONS

We're going to start moving lights around the using the buttons on the controller. How can we do this?



Algorithm

When the right button is pressed: move the light to the right.

This algorithm is too vague! First off, before we start, we need to know where we are.

New Algorithm:

Set CurrentPosition = 0

When the right button is pressed:



Change the CurrentPosition by ?

To go from 0 to 1 we need to change the CurrentPosition by 1.

Let's get all the direction buttons working.



Ask the students to write down these algorithms.

Show the algorithms without the numbers. Encourage the students to work out these themselves.



Show the full algorithms.



When the up button is pressed

Change the CurrentPosition by -8

Set the CurrentPosition light red

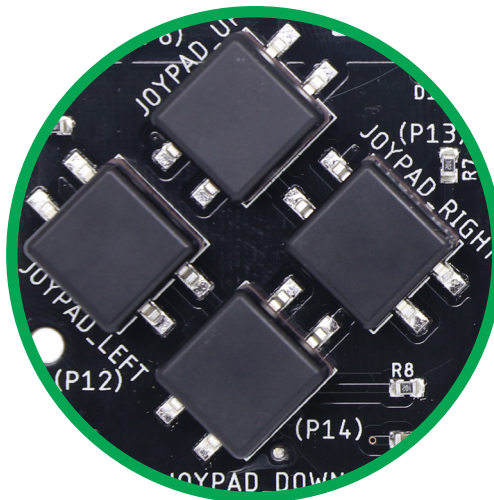
Show the lights.

When the left button is pressed

Change the CurrentPosition by -1

Set the CurrentPosition light red

Show the lights.



When the right button is pressed

Change the CurrentPosition by 1

Set the CurrentPosition light red

Show the lights.

When the down button is pressed

Change the CurrentPosition by 8

Set the CurrentPosition light red

Show the lights.

LESSON 3

LIGHTS AND BUTTONS

Main Lesson

Let's get coding



Curriculum mapping

Use two or more programming languages, at least one of which is textual, to solve a variety of computational problems.

Ask students to look back at their previous notes of detecting a button press and challenge them to implement the algorithms above.

Differentiation: You may want to remind students how to create and assign variables. (They created a variable in lesson 1).

Allow the students time to create at least two button presses. At some stage they might realise the bug below: Stop the class to discuss it.



BUG ALERT!

The algorithm was not specific but the light should move from square to square. Currently we are not clearing the light as it goes, e.g. light 0 is on and when they press right light 0 **and** light 1 are now on.

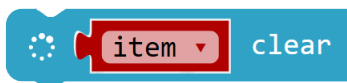
The new algorithm for the right button, for example, is:



When the right button is pressed

Clear the CurrentPosition light
Change the CurrentPosition by 1
Set the CurrentPosition light red
Show the lights

NOTE: This block clears the entire grid. We do not want this!



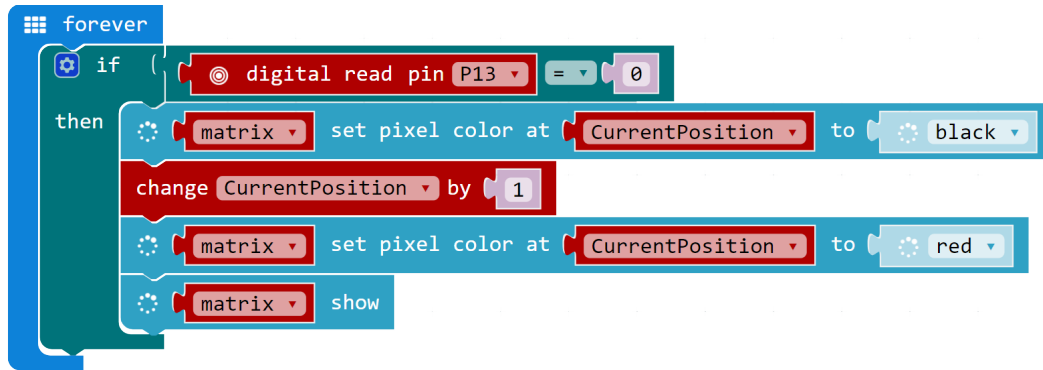
LESSON 3

LIGHTS AND BUTTONS

Ask the students to try and figure out how to clear one light.
(They could set its RGB colours to 0 0 0 OR they could colour it black).



Answer to moving the light to the right.



Give the students lots of time to get all 4 buttons working and the light moving around the screen.
Speed is a problem! They might want to experiment with adding pauses to the buttons.



BUG ALERT!

When we get to light 56, for example, and press down – what happens?
The light disappears!
Same with when we press right at light 64: The light is going outside the range of the matrix.



How can we fix this?

Look at the algorithm and put in a conditional statement. The light only moves if it's under 64:



Advanced Challenge

If the edge is hit you could vibrate the motor.



When the right button is pressed

```
IF CurrentPosition + 1 is less than 64 THEN
    Clear the CurrentPosition light
    Change the CurrentPosition by 1
    Set the CurrentPosition light red
    Show the lights
END IF
```

When the right button is pressed

```
IF CurrentPosition + 1 is greater than 64 THEN
    Clear the CurrentPosition light
    Change the CurrentPosition by ?
    Set the CurrentPosition light red
    Show the lights
ELSE
    Turn the motor on
    Pause for half a second
    Turn the motor off
END IF
```

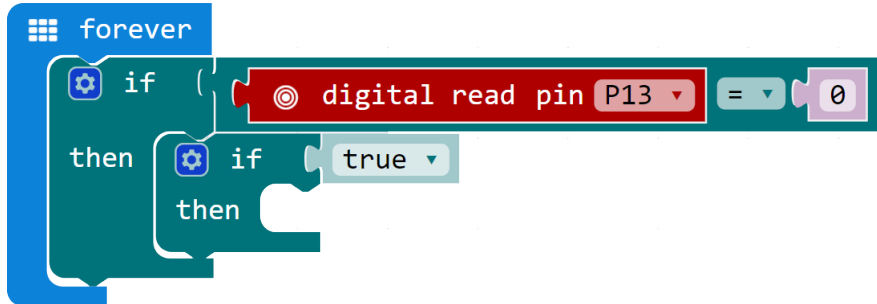
Continued on next page >

LIGHTS AND BUTTONS

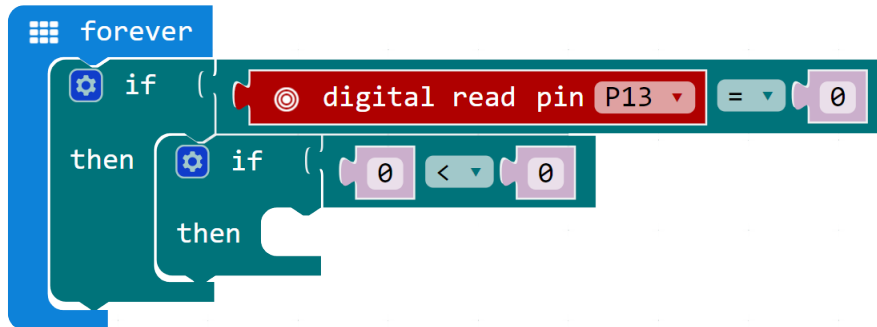
Answer for moving the light to the right



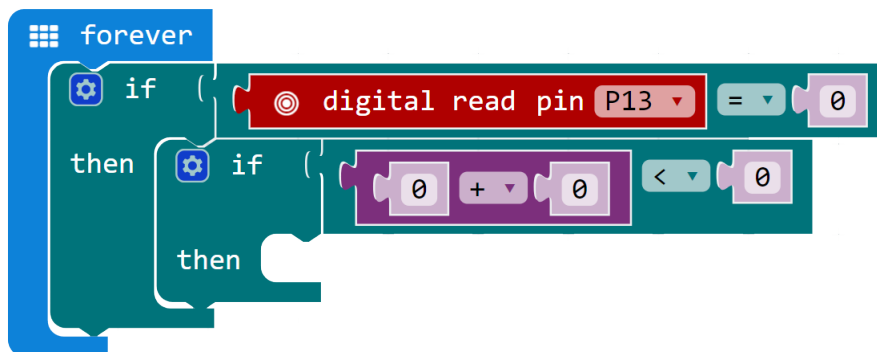
From Logic:



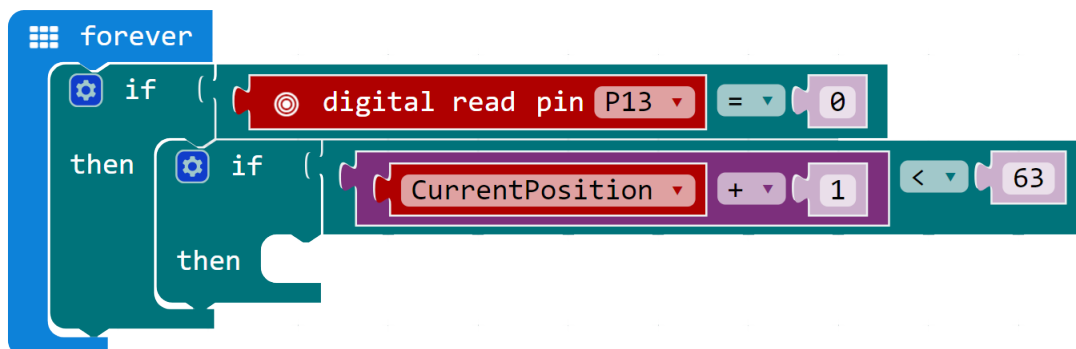
From Logic:



From Math:



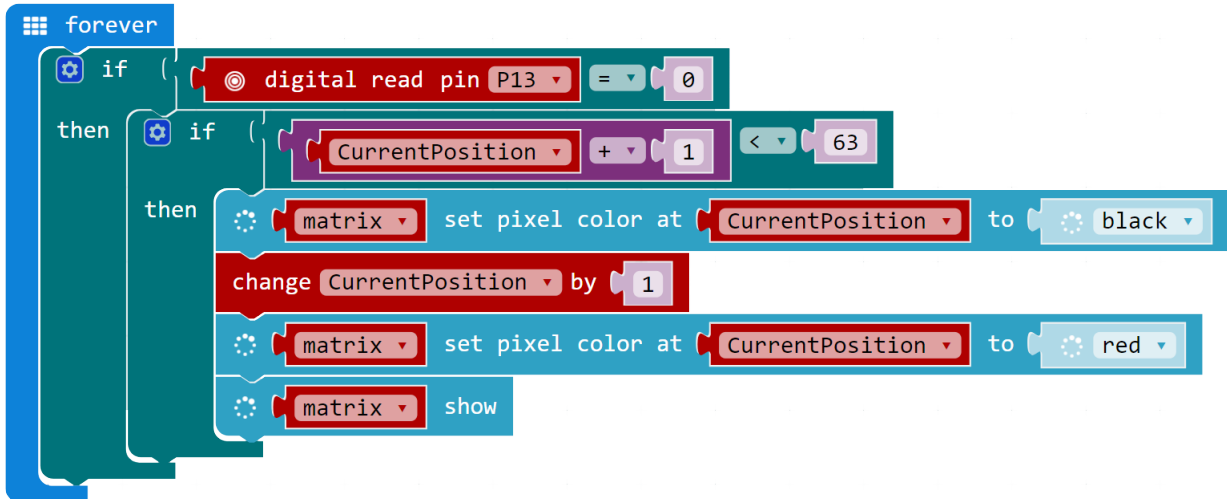
From Variables drag out CurrentPosition and change the second 0 to 1 and the third 0 to 63:



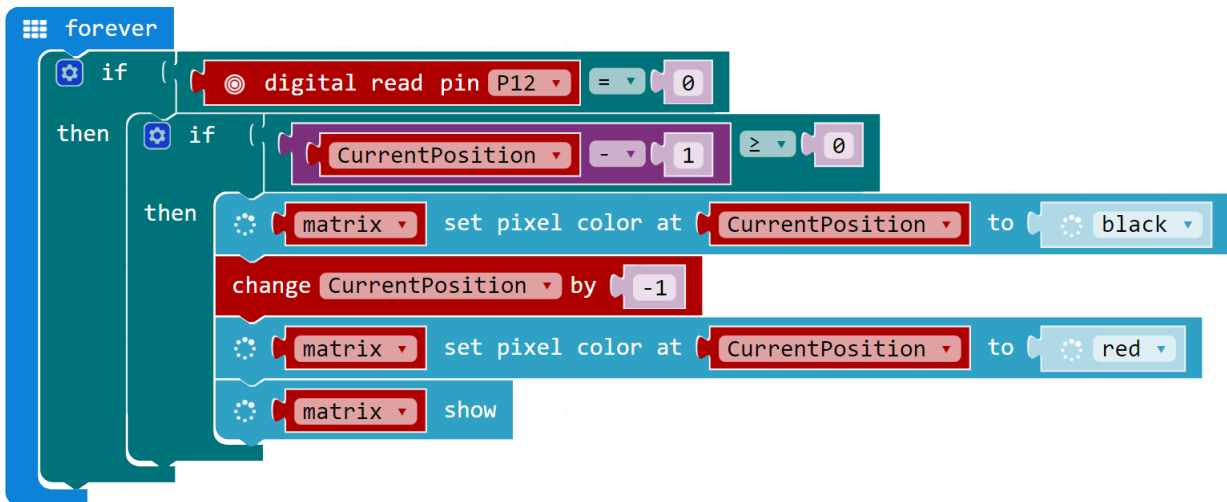
Continued on next page >

LIGHTS AND BUTTONS

Add in the original code:



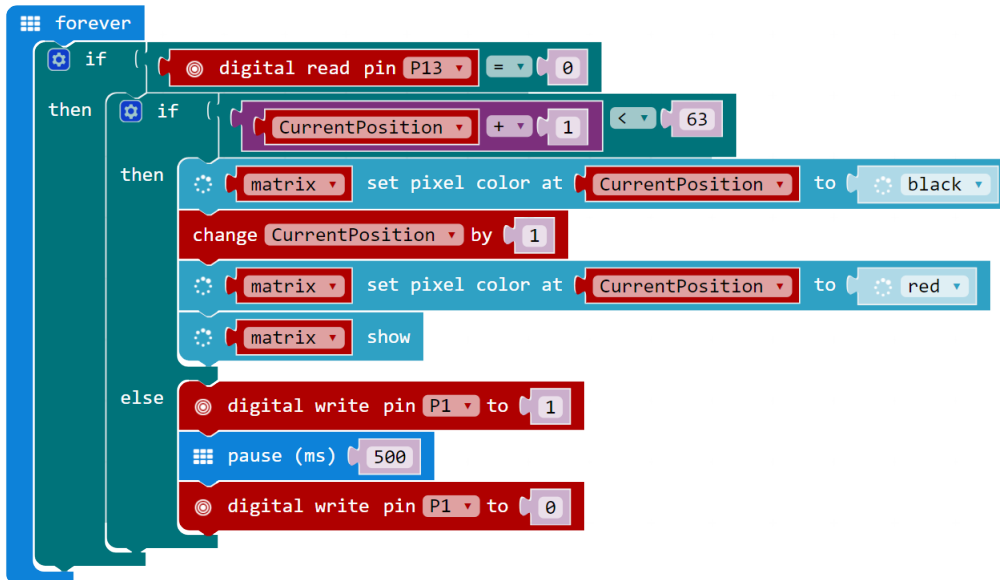
Right and down need to worry about going over 63. Up and left need to worry about going under 0. This is the code for left:



Continued on next page >

LIGHTS AND BUTTONS

The advanced challenge answer:



Challenge

Once the students have the light moving around the matrix without falling off the matrix, set them a challenge of creating a simple game.



Algorithm

(Keep your code of moving the light around the matrix using the direction buttons).

Set CurrentPosition to 0
 Turn the CurrentPosition light red
 Set score to 0
 Display the score on the micro:bit
 Set Ball to 10
 Turn the Ball light blue

Forever:
 When the red light reaches the Ball
 Change the score by 1
 Display the score on the micro:bit
 Set Ball to a random number between 0 and 63
 Turn the Ball light blue
 Show the lights

This is a real challenge! But broken down into parts the students should be able to solve it.

LIGHTS AND BUTTONS

Answer:

```

on start
  set matrix to NeoPixel at pin P0 with 64 leds as RGB (GRB format)
  set CurrentPosition to 0
  matrix set pixel color at CurrentPosition to red
  set Score to 0
  show number Score
  set Ball to 10
  matrix set pixel color at Ball to blue
  matrix show

forever
  if CurrentPosition = Ball
  then
    matrix set pixel color at CurrentPosition to red
    change Score by 1
    show number Score
    set Ball to pick random 0 to 63
    matrix set pixel color at Ball to blue
    matrix show
  
```

Before the end of the lesson:



Ask the students to take notes.

They will need a copy of the algorithms and their code for each challenge they've completed today. Take screenshots of their code and explain it.

Summary

This lesson students combined the skills learnt in previous lessons to move lights around the controller using the buttons on the controller. They have started to code boundaries and a simple game.



This is the third lesson for students in **Key Stage 3** for the **Kitronik :GAME ZIP64** controller for **BBC micro:bit**. This lesson involves combining the skills learnt from the previous two lessons together. The students will use the buttons to move a light around the controller. At the end of the lesson the students will create a simple game. The recommended ratio of students to :GAME ZIP 64 is 2:1. Please find additional lesson plans, accompanying **PowerPoint** presentations and more at www.kitronik.co.uk/5626.

The Kitronik :GAME ZIP 64 is the ultimate retro gaming accessory for the BBC micro:bit. This all in one hand held gaming platform includes a built in 64 (8x8) individually addressable full colour ZIP LED screen.

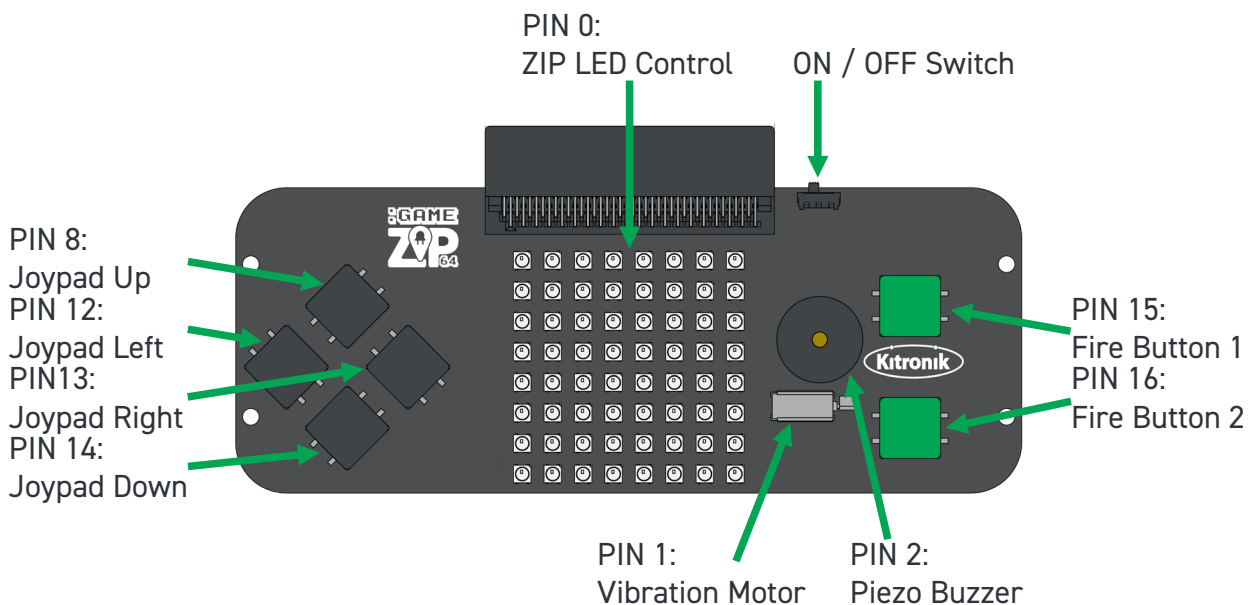
It features on-board sound, 4 directional buttons, 2 fire buttons, and haptic feedback. All features are fully programmable. Breakout points are also included allowing for shoulder buttons or I2C devices to be added, as well as the use of additional LEDs.

All the BBC micro:bit features are still available when plugged in to the :GAME ZIP 64, so your games can still make use of the LED matrix, accelerometer etc.



LESSON REQUIRES:

- Pen & Paper including coloured pens (grid paper)
- A computer/laptop with a USB port and Internet access
- A :GAME ZIP64
- A BBC micro:bit
- 3 x AA batteries
- A micro USB cable



WARNING : Contents may inspire creativity

T: 0845 8380781

W: www.kitronik.co.uk

E: support@kitronik.co.uk

🇬🇧 Designed & manufactured
in the UK by



[kitronik.co.uk/twitter](https://twitter.com/kitronik)



[kitronik.co.uk/facebook](https://facebook.com/kitronik)



[kitronik.co.uk/youtube](https://youtube.com/kitronik)



[kitronik.co.uk/google](https://google.com/kitronik)