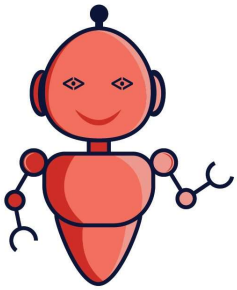
2018



KidzCanCode

IGNITE, INSPIRE, INNOVATE

www.kidzcancode.com

# Fireflies

KIDZCANCODE

TREVOR WARREN

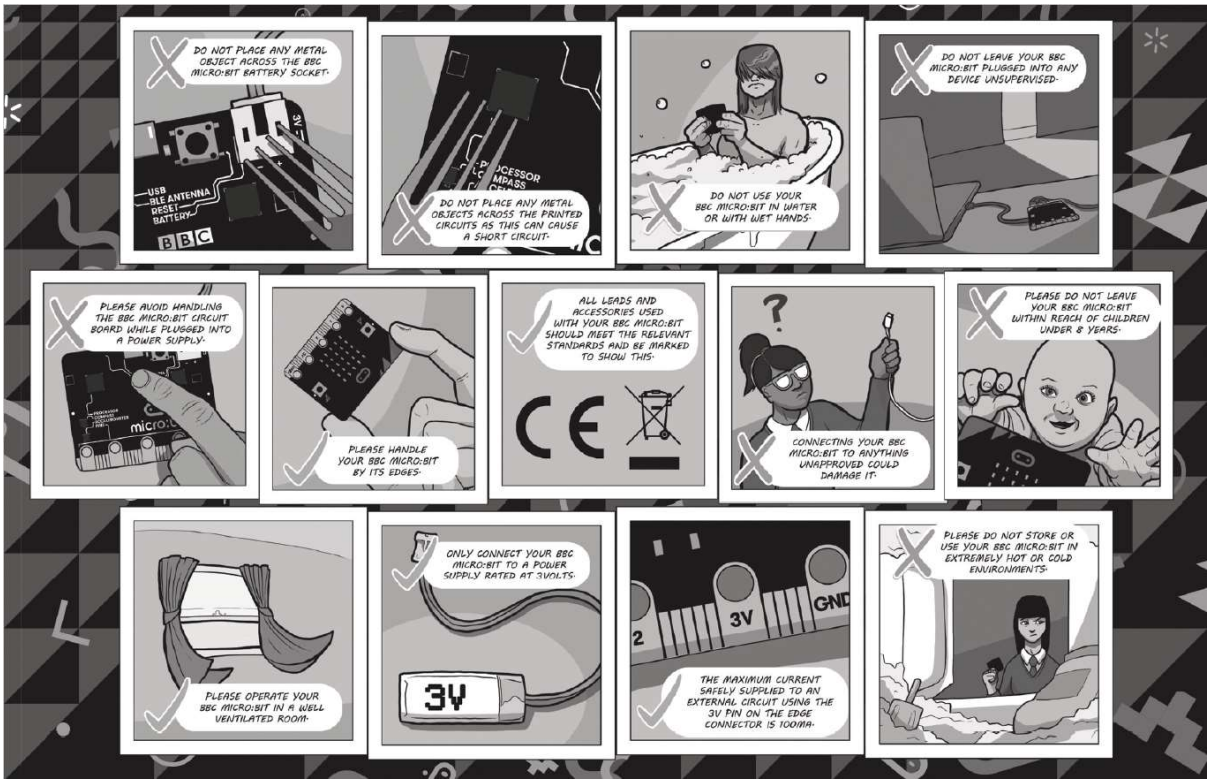
**Tutorial 2.1e – Fireflies**

# 1. Safety Warnings

THE MICRO:BIT IS AN EXPOSED BOARD, TO BE USED WITH CARE PLEASE RETAIN THIS INFORMATION FOR FUTURE REFERENCE. You can read the detailed document at - [http://microbit.org/guide/safety-advice/](http://microbit.org/guide/safety-advice/)

## a. General Safety Warnings

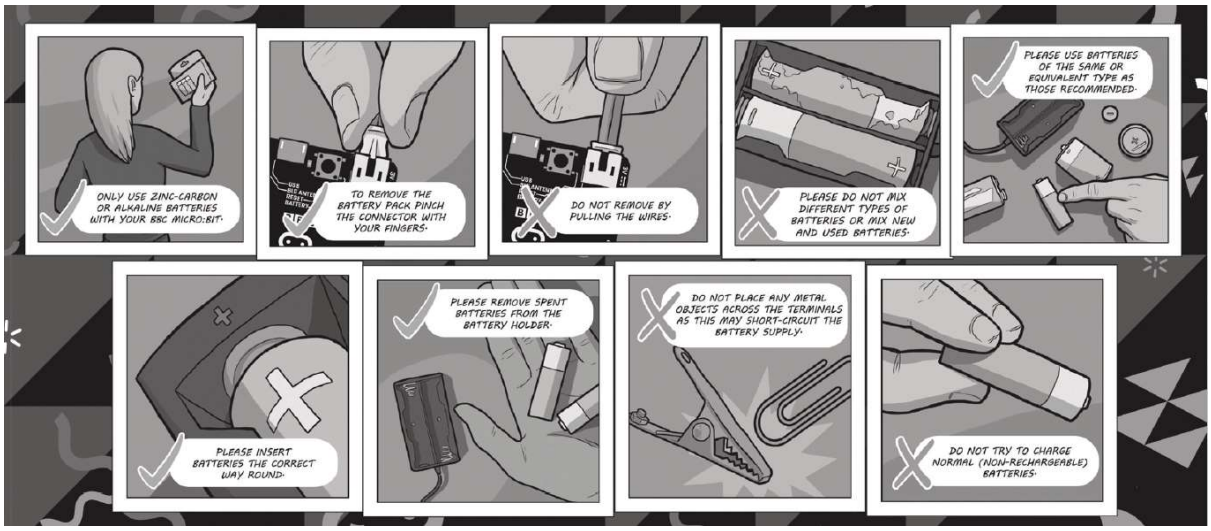
Using the BBC micro:bit is easy to use but is designed to have all the electrical parts on display. This does mean there's a small risk that the parts can be damaged and even overheat with a risk of injury but a little bit of care and caution will ensure you and your micro:bit will stay fit and healthy.




1. Always keep your BBC micro:bit in the anti-static bag when not in use. It's good practice for students to earth themselves before handling it.
2. Please handle your BBC micro:bit by its edges. This minimises the risk of damage through an electrostatic discharge.
3. Please use the battery pack and the USB lead provided to power your micro:bit. Do not use portable battery chargers or USB charging ports (often marked with a lightning bolt or 'SS'), to power your micro:bit. Using these may damage your micro:bit and stop it working properly.
4. Please avoid handling the BBC micro:bit circuit board while plugged into a power supply.
5. All peripherals (for example: USB cable, battery holder, sensors) used with your BBC micro:bit should comply with the relevant standards and should be marked accordingly.
6. Connecting your BBC micro:bit to any unapproved peripherals could damage your BBC micro:bit
7. Please do not attempt to keep using faulty micro:bits. If a school-issued micro:bit develops a fault, contact the vendor immediately.

8. The maximum current safely supplied to an external circuit using the 3V pin on the edge connector is 100mA. Please make sure this limit is not exceeded.
9. Please do not store or use your BBC micro:bit in extremely hot or cold environments.
10. Do not place any metal objects across the printed circuits on the board as this can cause a short circuit damaging your BBC micro:bit. This can cause risk of burn or fire.
11. Do not use your BBC micro:bit in water or with wet hands.
12. Do not leave your BBC micro:bit plugged into a computer or any other device unsupervised.
13. Please do not leave your BBC micro:bit within reach of children under 8 years of age.
14. Please operate your BBC micro:bit in a well ventilated room To remove the battery pack, pinch the connector with your fingers. Do not remove by pulling the wires.

## b. Battery Warnings



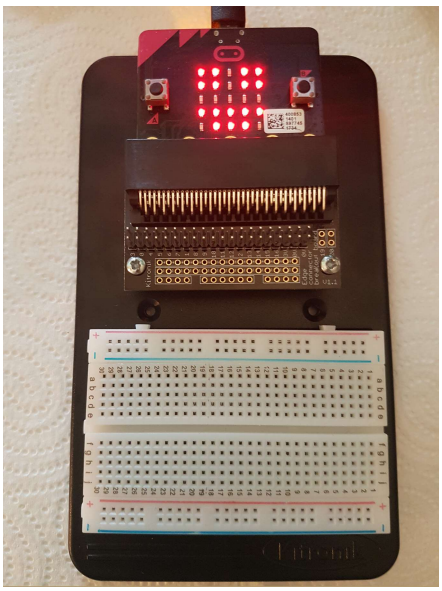

1. Do not try to charge normal (non-rechargeable) batteries
2. Please do not mix different types of batteries or mix new and used batteries.
3. Please use batteries of the same or equivalent type as those recommended.
4. Please insert batteries the correct way round (with the correct polarity).
5. Please remove spent batteries from the battery holder.
6. Do not short-circuit the battery supply terminals, for example by placing a metal object across the terminals.
7. Only use Zinc or Alkaline batteries with your BBC micro:bit.
8. Please do not use rechargeable batteries

## 2. Pre-requisites

If you have questions with the assembly of the micro:bit, edge connector breakout board, mounting board and the breadboard please drop us a note at help@kidzcancode.com. The edge connector board, mounting board and the breadboard are part of the Kitronix Inventors kit which needs to be purchased separately.

To be able to perform this tutorial you will need the following components –

1. Parts required –
    a. 1 x BBC Micro:bit
    b. 1 x Mounting Plate
    c. 1 x Edge connector breakout board
    d. 1 x Bread board
2. Assembly required –
    a. Bread board mounted on top of the mounting plate
    b. BBC Micro:bit inserted into the Edge Connector breakout board



Before proceeding please check your setup and confirm that all the required parts are configured as demonstrated in the above picture.

## 3. Learning Objectives

The objectives of this tutorial are to introduce the student to the following concepts –

- Use of plot and unplot commands to turn LED's on and off across the board
- Understanding the use of custom functions for purposes of programming
- Declaring variables and assigning values to variables at different times of execution in the program
- Coding for use of the buttons on the micro:bit, perform a given action when a button is pressed.
- Understanding the meaning and structure of arrays.
- Learn to make use of arrays and how to add elements to an existing array.

The BBC micro:bit is a powerful little computer. Through programming these games kids explore more advanced computer science concepts. Along the way kids are encouraged to share, create and extend the games using their own imagination and creativity.

This tutorial builds upon concepts introduced in previous tutorials so please make sure you have covered the previous tutorials before you dive into this one. So overall this tutorial intends to build upon concepts learnt in previous tutorials while exploring new concepts.

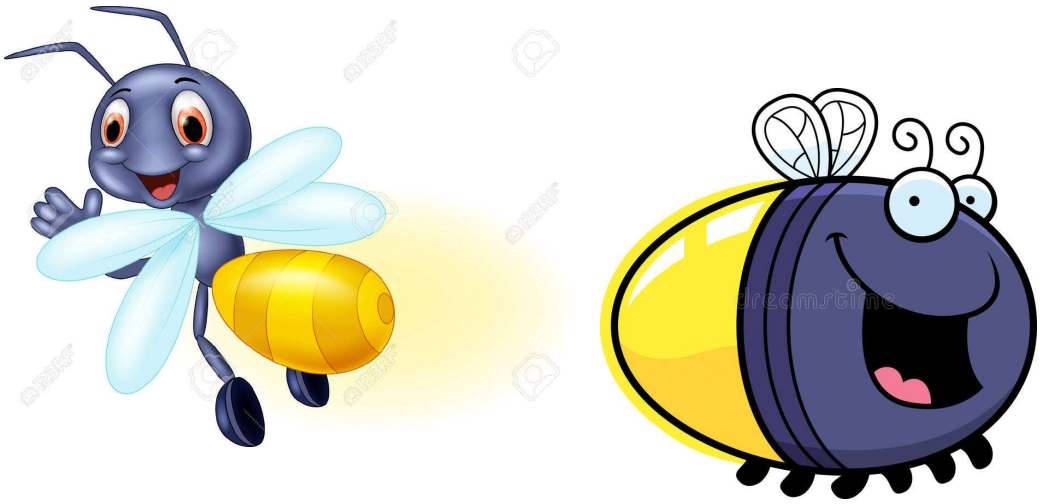
In future tutorials we will continue to build upon the concepts learned here and will build more complex interactive games using the functionality provided by the micro:bit.

# 4. Activity

## a. Activity

This activity involves designing a program using a combination of variables and custom functions to create a firefly effect by randomly lighting up LED's across the board to simulate fireflies dancing around in the dark.

The main challenge here is to turn on and turn off the LED's using the plot command to light up LED's at random positions on the board. The objective is to write code making sure one doesn't use the "show leds" code block. The "show leds" option allows the user to easily just check a box and have the LED light up which is the easy way out and we will avoid doing that instead using the plot command to switch the LED's on and off.



Using the plot command forces the programmer to think through the coding challenge, learn the use of variables, understand and work through the co-ordinate system and implement all of plot/un-plot commands using coding logic making it a really interesting challenge.

The activity is designed to have additional challenges which allow the developer to keep pushing the boundaries. The main challenges in this activity involve –

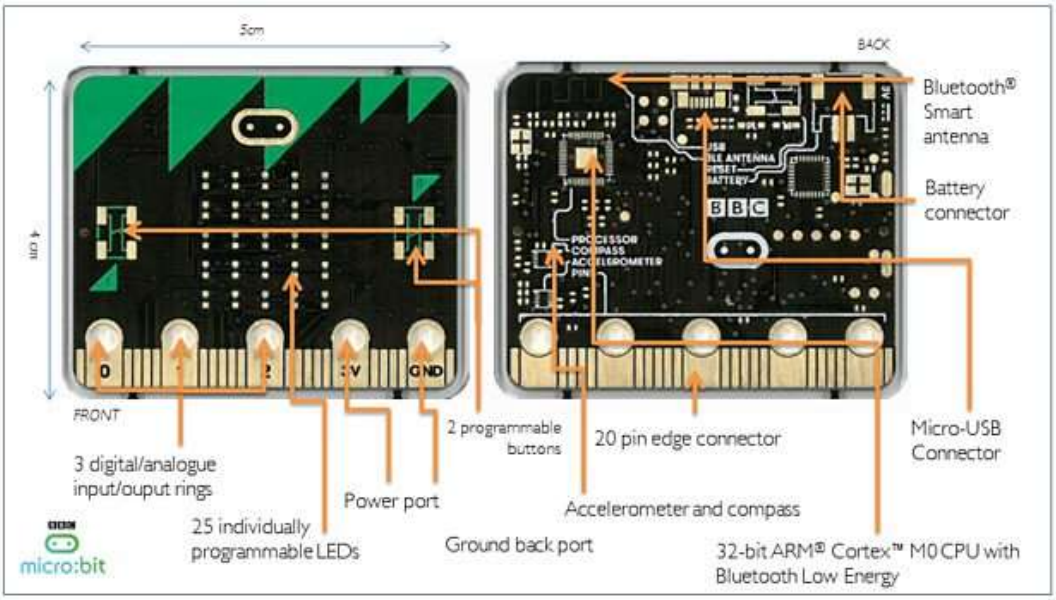
- Using a number of custom functions and calling on them as needed.
- Working with array lists and custom variables.
- Using if-Then-Else if multiple times to check an array and then calling custom functions.

## b. How Does It Work

This section talks about the BBC micro:bit and what it's made up of. If you have already read through this section then feel free to skip directly to the next section. The BBC micro:bit is a powerful little board and has various types of sensors on board. Here's what makes up the BBC micro:bit.

1. Size: approx. 5cm x 4cm.
2. Weight: 8g.
3. Processor: 32-bit ARM Cortex M0 CPU.
4. Bluetooth Low Energy.
5. Digital Compass.
6. Accelerometer.
7. Micro-USB controller.
8. 5x5 LED matrix with 25 red LEDs.
9. Pins for connecting external sensors, LED's, etc.

Here's what the micro:bit looks like –




**Front of the board (left hand side)**

1. Button A (left button with edge connector at the bottom) – labelled A on the board
2. Button B (right button with edge connector at the bottom) – labelled B on the board
3. P0 (left large pin (crocodile clip port) with edge connector at the bottom) - labelled 0 on the board
4. P1 (middle large pin (crocodile clip port) with edge connector at the bottom) - labelled 1 on the board
5. P2 (right large pin (crocodile clip port) with edge connector at the bottom) - labelled 2 on the board
6. +3V - labelled 3V on the board. This is 3V PWR OUT
7. GND
8. P3 – P22 pins from left to right with edge connector at the bottom. Referred to as Pins when referencing that part of the board. Text will talk about 'pins' when referring to

individual connections or the general way of connecting to the board – not labelled on the front of the board
9. LED matrix referred as the 'screen' - not labelled on the board
10. LED coordinates starting at 0,0 top left corner and ending at 4,4 at the bottom corner - not labelled on the board

The order of the large pins as follows: P0 P1 P2 3V GND labelled 0, 1, 2, 3V GND on the board

**Rear of the board (Right hand side)**

1. 1. USB Plug (Micro-USB plug) – labelled USB on the board
2. Button R (reset button) – labelled Reset on the board
3. Status LED – not labelled on the board
4. Battery socket – labelled Battery on the board

**Other components on the board include**

1. Accelerometer
2. Compass
3. Bluetooth Smart Technology Antenna
4. AAA Battery Holder - not labelled on the board
5. Processor (Cortex M0)

The BBC micro:bit is programmable in a few different languages. You can write code for the micro:bit using the Makecode block coding interface, Javascript, Python of even in C. Most of our tutorials will cover the use of the Makecode block coding interface built by Microsoft for the micro:bit.

## 5. Let's write some code

It's time to write some code and get going with coding our game. So let's head over to the micro:bit block code editor page (https://makecode.microbit.org/) and get coding!!!

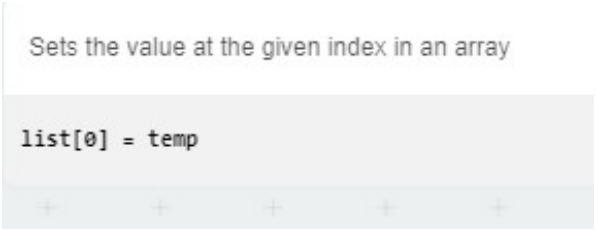
### a. On Start

In the following section we will dive into the code you will put together for this tutorial. The code is split into various different sections just to make things a bit easier to comprehend. The first code block below covers off concepts that you would have covered in previous tutorials so while we cover the relevant blocks, we will not dive into a lot of depth here.

In the first code block, we use the "on start" block provided by the micro:bit which is run only "once" during a given program. You can only trigger the "on start" block once again when you hit the re-set button or pull the power plug and reboot the board. So please do keep that in mind with regards to the "on start" block of code. You want to put stuff into the "on-start" code block that you want run at the start of the program.


a) The "on start" block of code simply shows a string
b) We then pause briefly to ensure that the reader has been able to see the text being displayed
c) We then show a pre-defined icon
d) We use the "set" commands to initialize a series of variables required in the image above.

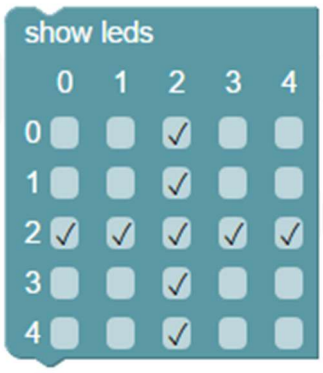
e) We set the first element of the “list” array equal to the value of our custom variable ”temp”
   a. This last line of code is equivalent to this as below (You can hover over a block to see what the code version of it is):




This brings the on-start block of code to an end. Feel free to dive in and customize the “on-start” code block with additional code that you want to put in. Please note that the addition of the “pause” commands (similar to our use of the wait commands in scratch) is intended to inject wait and slow down processing so that the flow of the program makes sense to the human being. To the computer, not having the wait command just lets it breeze through all the code one instructions after another.

Before we dive into this tutorials let’s quickly have a look at the X, Y co-ordinates for the LED’s on the microbit. Refer the image below which shows the LED matrix with X and Y co-ordinates marked.

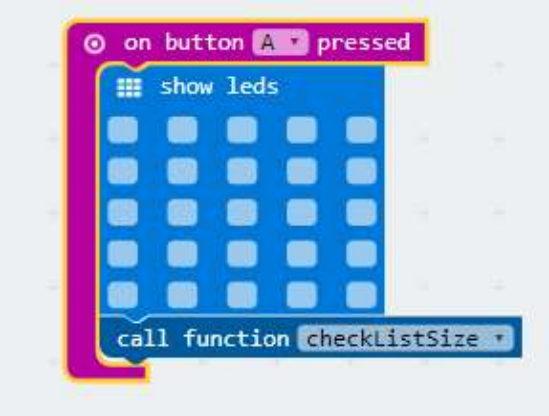
- The X axis is the horizontal axis with numbering from 0 – 4
- The Y axis is the vertical axis with numbering from 0 - 4




In this tutorial we will use variables (Remember the data section in Scratch?) to represent the X, Y co-oridinates and light up the LED’s as we increase and decrease the value of the variables. There’s a lot more to X, Y co-ordinates that just what’s been covered here however we’ll limit ourselves to the short X, Y axis covered above for purposes of working with the micro:bit.

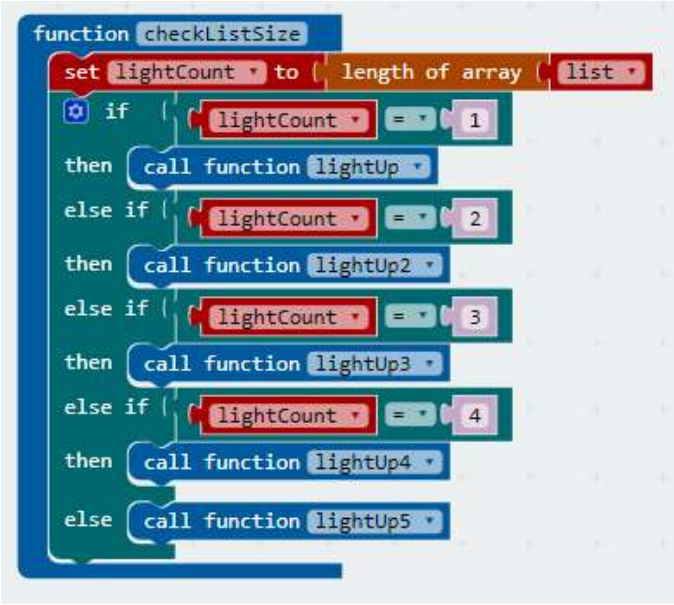
## b. On Button A Pressed

Let's now put together the code for the "on Button A pressed". The code in this block is automatically executed everytime a user clicks on button A. Our intention here is to program the micro:bit such that everytime the user presses button A the current LED's on the board are reset to show a clean board with no LED's showing. After this has happened, the micro:bit will call a custom function we will create called "checkListSize".




## c. Custom Function – Checking List Size

The next code block is focused on a custom function we will use to check list size.

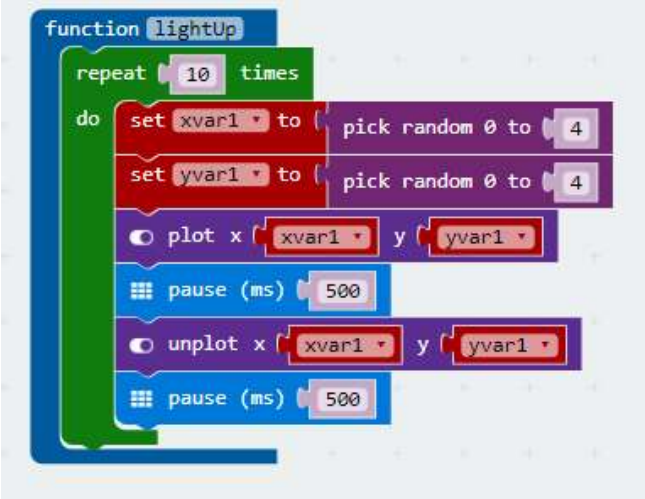



This block of code will check the size of the array "list" and call other custom functions depending on the length of the array

a) Set the variable "lightCount" equal to the length of the array "list"
b) Use an if – else if – else logic statement to check if the variable "lightCount" is equal to a certain number i.e. 1. Depending on the number it will call a custom function we will create next

## d. Custom Function – Light Up

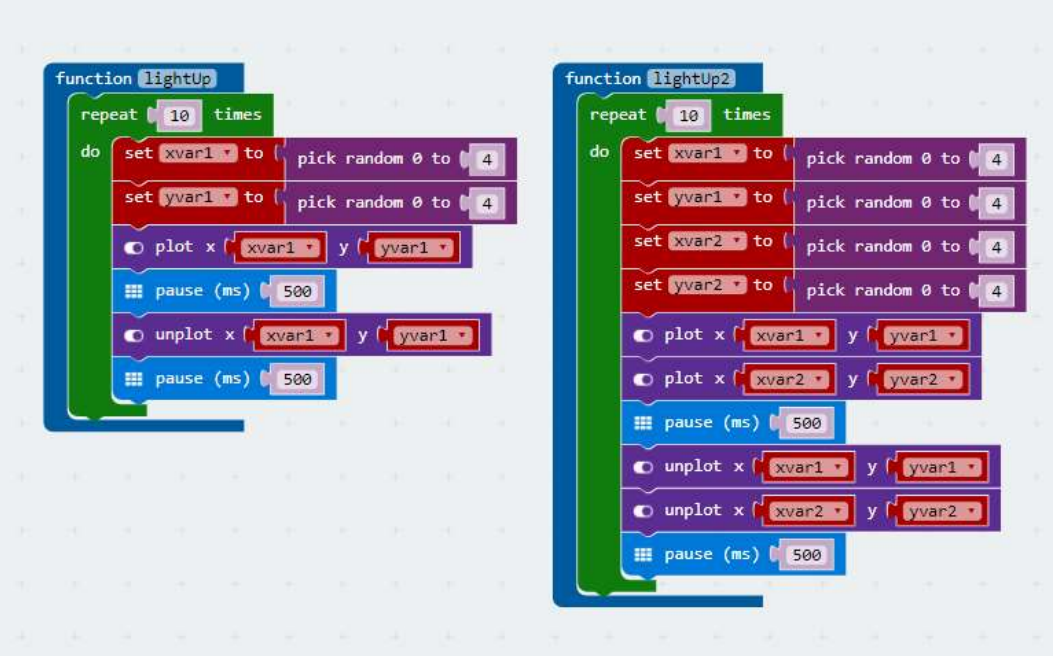
Lets now put together the code for our “lightUp” functions that we will use to plot LED’s at random locations on the board depending on the number of items in the “list” array. These functions will repeat 10 times and set the variables “xvar1” and “yvar1” to a random spot on the board at the x and y coordinate.




a) Set a repeat block to run the code inside it 10 times
b) Set the variable “xvar1” and “yvar1” to a random point on the LED board between 0 and 4
c) Plot an LED point at the x and y coordinate using the variables that were set previously.
d) Pause for 500ms
e) Remove the LED point at the x and y coordinate
f) Pause again for 500ms

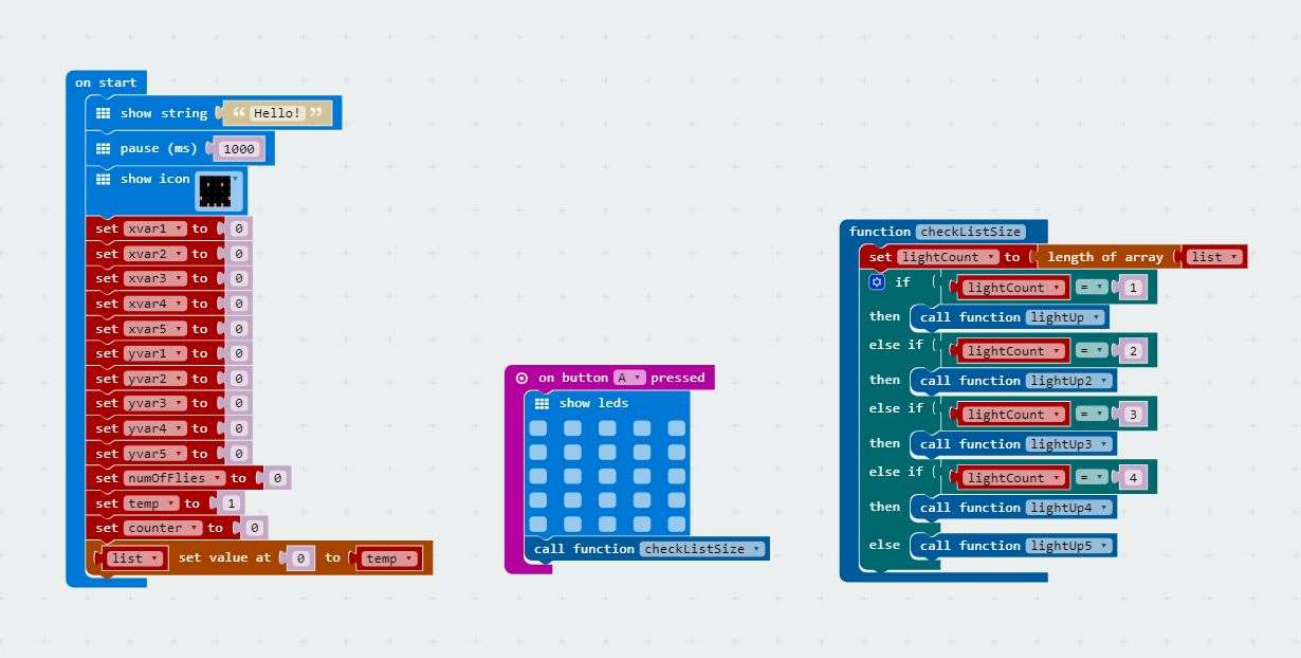
## e. Custom Function –Light Up (number)

We've set up the code for the "lightUp" function previously but when we want to add more than one LED on the board, we must alter the code slightly to allow for more than one point be plotted.

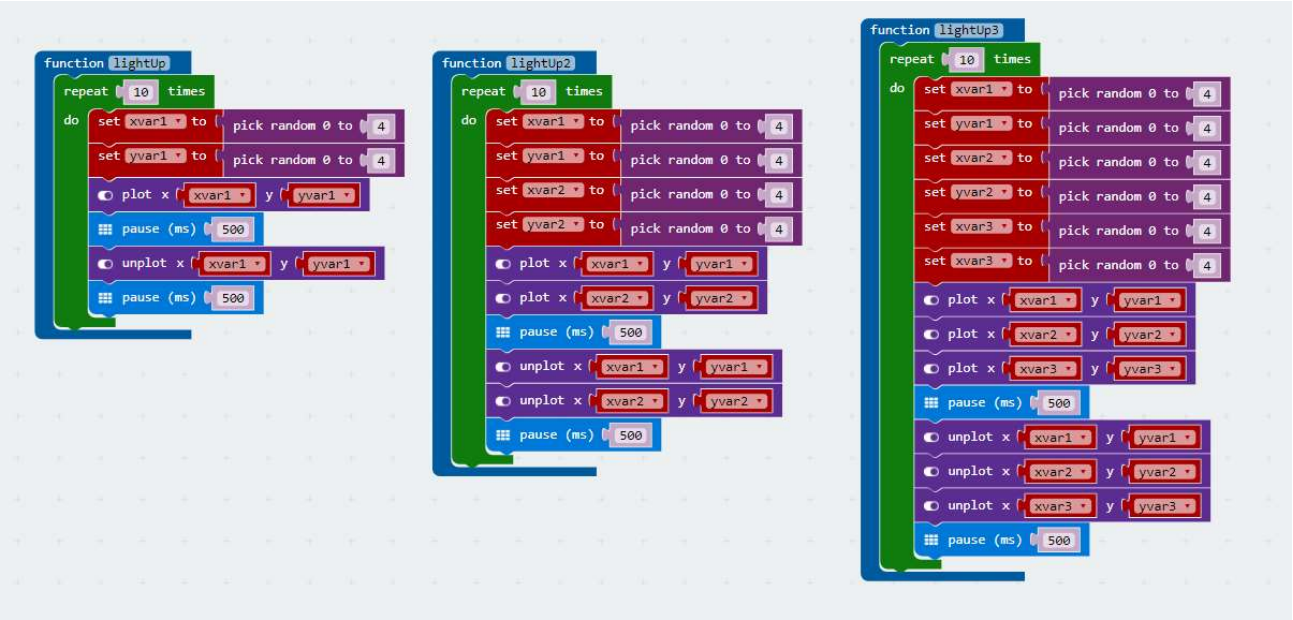


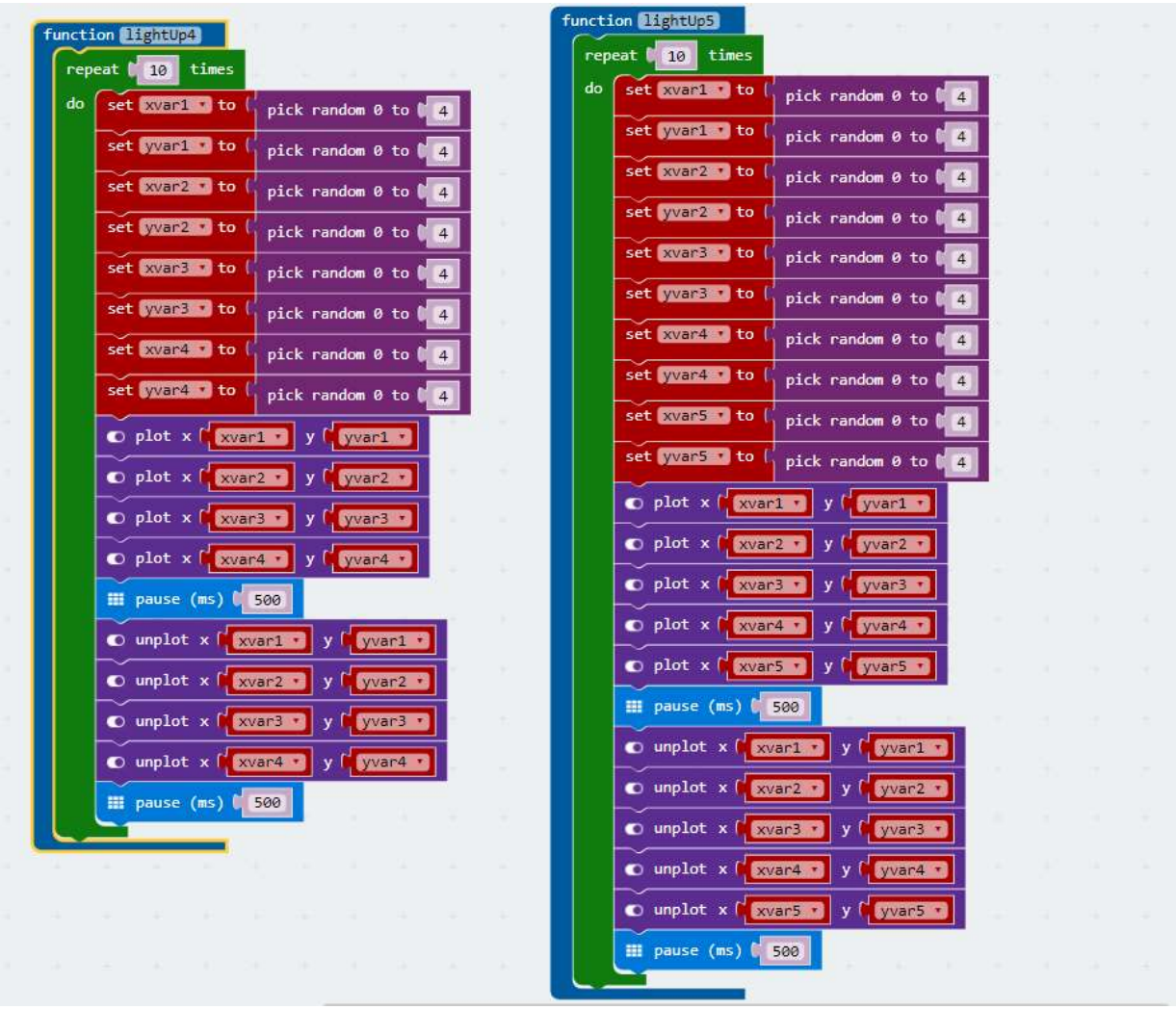

We make use of the "xvar" and "yvar" (1 – 5) depending on the number of light up LED we want to light up. For example in the function as shown above "lightUp2" we use two "xvar" and "yvar" because we want to show two LEDs on the board at once.

The next section provides a combined view of the all the code used for the game.



## Tutorial 2.1e – Fireflies

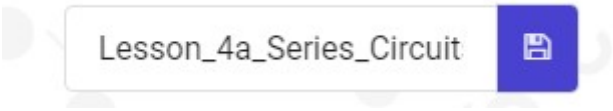



You are encouraged to make changes, improvise and customize the game using your own ideas.

## 6. Downloading Your Code To The micro:bit

Once you have completed the program, enter a name for your program using the option provided below i.e. in the text box adjacent to the small save button.




Now that you have given your program a name and saved it you can download it your micro:bit. But before we do that let's confirm what drive your micro:bit shows up as. On most machines the micro:bit will show up as an additional USB drive. So head over into windows explorer and confirm what drive name (D:, E:, F:. G:, etc. ) the micro:bit shows up as. You need to absolutely be sure what drive the micro:bit shows up as. Once you've confirmed what drive the micro:bit shows up as on your machine you can select the right drive when downloading the code to the micro:bit. If in doubt please ask the volunteer/mentor/session facilitator helping out.


To download the newly written code to the micro:bit, hit the download button shown above. You should now see a dialog box open up and you will be asked to save the file somewhere on your machine. Please choose the drive your micro:bit shows up as i.e. D: or E: or F: or whatever it shows up as on your machine. A sign of success is when you see the lights on the rear (orange) lighting up in quick succession suggesting that the code is being written to the micro:bit. On completion the micro:bit reboots and you should now see the code in action on the micro:bit.

If hitting the download button shown above does not open up a dialog box asking you to save to the micro:bit please save the file (you will have a <filename.hex> file) to your desktop. Then open up windows explorer and drag that file onto the drive which is your micro:bit. A sign of success is when you see the lights on the rear (orange) lighting up in quick succession suggesting that the code is being written to the micro:bit. On completion the micro:bit reboots and you should now see the code in action on the micro:bit.

Please feel free to customize the code blocks, have a play. Add your own custom code and re-download the code to the micro:bit. Give yourself a tap on the back, you've just completed your first circuit!!!!!

## 7. Challenges

Well done for completing the tutorial. There's a lot of ground we have covered in this tutorial so please feel free to make notes, come back to the tutorial at some point down the line and ask your learning facilitator any questions or doubts you might have on the concepts covered thus far.

Let us now stretch it a bit further -

1. Implement code to allow use of button A to increase the pause time (between LED blinks) by 100 ms every time the button is clicked
2. Implement code to allow use of button B to decrease the pause time (between LED blinks) by 100 ms every time the button is clicked OR on button B pressed add another variable and plot an extra LED at a random X and Y coordinate
3. Implement code to detect shake of the board and re-start the program i.e. re-run the entire program from the start.
4. Code the LED's to light up as a stream of lights moving around the board in a sequence to make it look like a snake is slithering around the screen. i.e. don't turn off the LED's as you progress through the inner and outer square