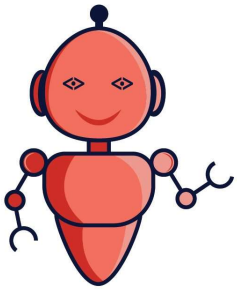
2018



KidzCanCode

IGNITE, INSPIRE, INNOVATE

www.kidzcancode.com

Crossy Roads

KIDZCANCODE

TREVOR WARREN

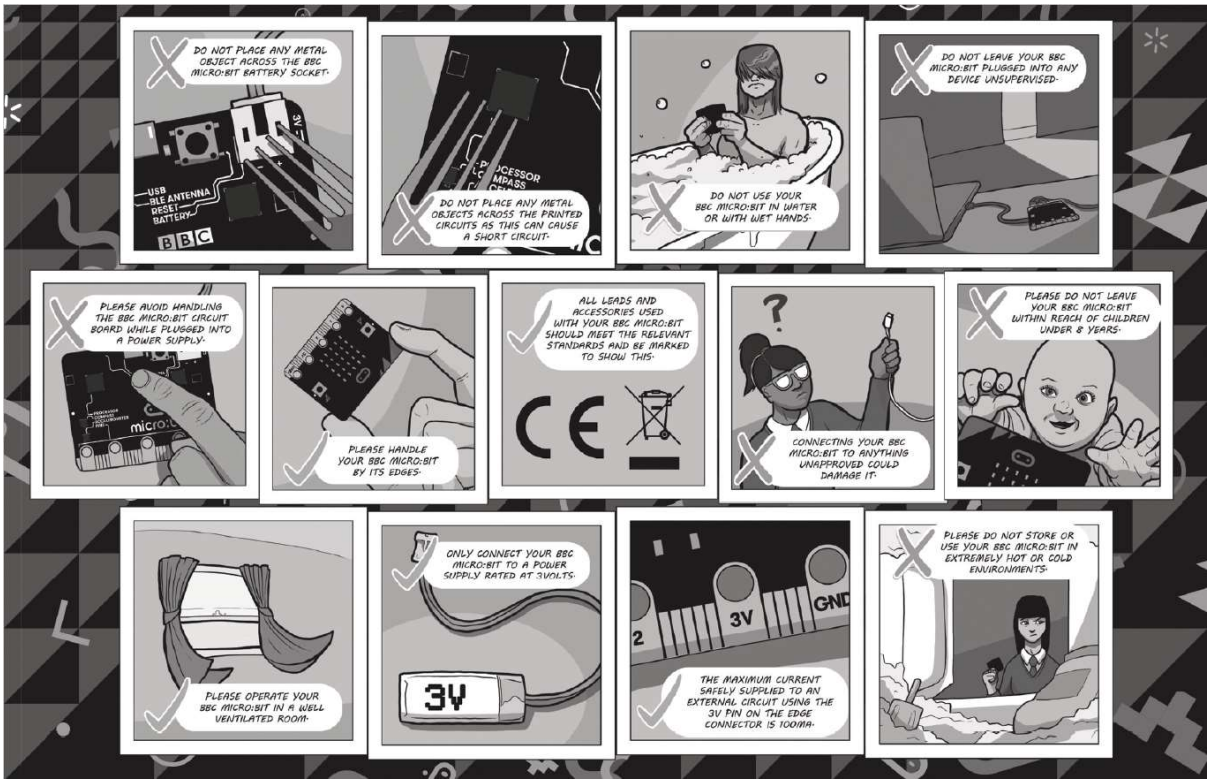
Tutorial 2.1b – Crossy Roads

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/

a. General Safety Warnings

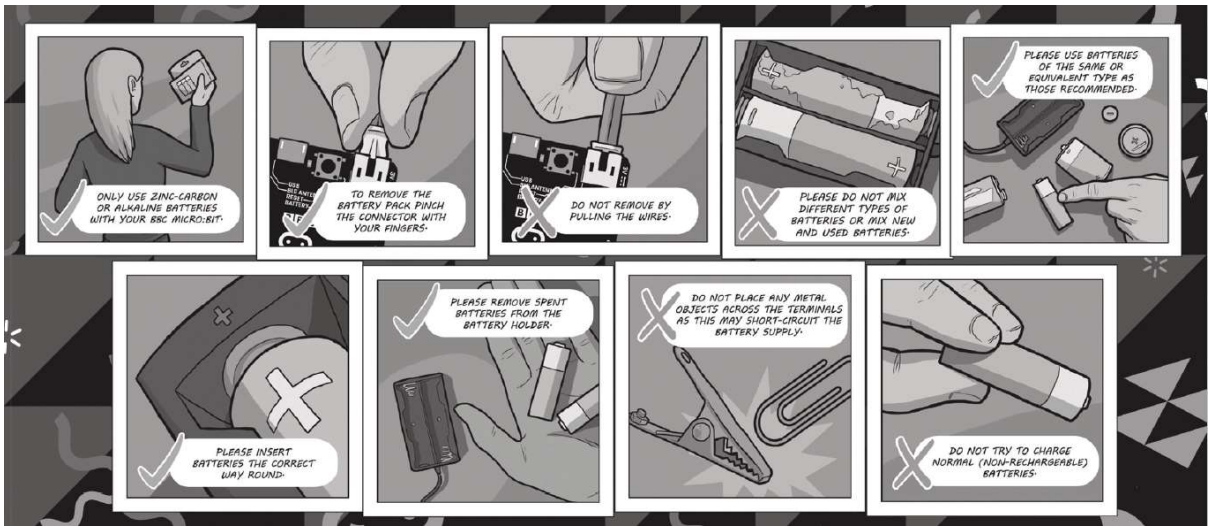
Using the BBC micro:bit is easy to use but is designed to have all the electrical parts on display. This does mean there's a small risk that the parts can be damaged and even overheat with a risk of injury but a little bit of care and caution will ensure you and your micro:bit will stay fit and healthy.



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



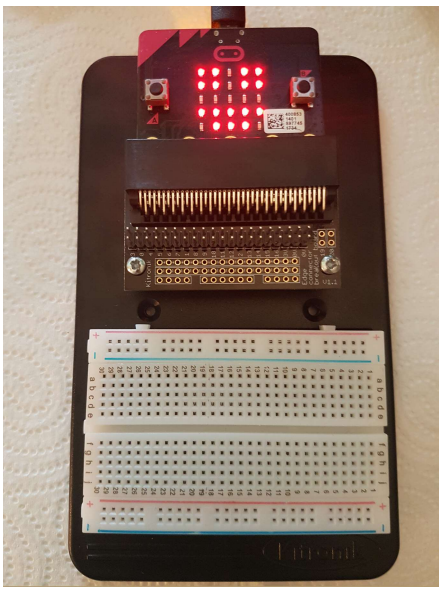
1. Do not try to charge normal (non-rechargeable) batteries
2. Please do not mix different types of batteries or mix new and used batteries.
3. Please use batteries of the same or equivalent type as those recommended.
4. Please insert batteries the correct way round (with the correct polarity).
5. Please 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 LED's on and off across the board
- Understanding the use of the "while do" loop to create logic blocks that make decisions when conditions are met
- Understanding the use of the "if then" loop to create logic blocks that make decisions when certain conditions are met
- 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.

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 creating an interactive game called “Crossy Roads”. The objective of this game is for the player to use the controls on the micro:bit and navigate from the bottom of the board to the top of the board while traffic (seen as a sequence of lights on the board) moves continuously from one side of the board to the other. The picture below is from the Disney crossy roads game, the game we are building on the micro:bit is on similar lines but a lot more simpler…☺

The game is designed to have several levels of challenges which allow the developer to keep pushing the boundaries. The challenges involve –

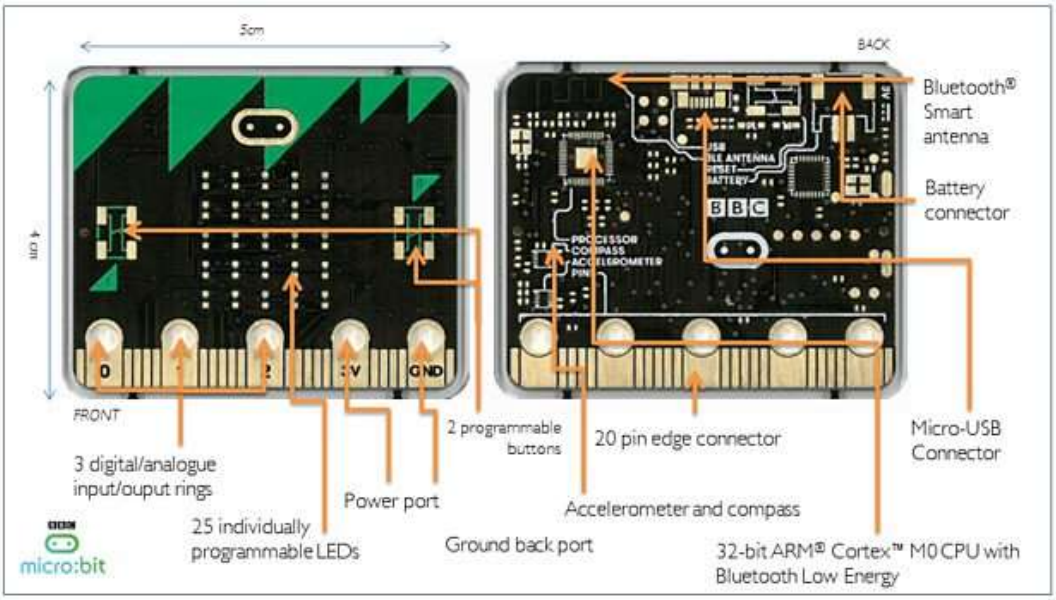
- Coding the LED’s to light up as a stream of lights moving from left to right and back i.e. continuously moving from one end of the board to the other. These moving LED’s will symbolize traffic moving across the road.
- Coding to get the buttons to control vertical movement of the player on the board. The player intends to cross from one side (bottom of the board) of the road to the other (upper end of the board).
- The player is represented by an LED and initially placed at the middle of the last row of the board.
- Coding to randomly change the rate at which the traffic (led lights moving horizontally) moves across the board.
- Coding to randomly increase or decrease the spacing between the lines. This will symbolize traffic changing lanes and moving randomly.
- Coding to interpret if the traffic (lights moving horizontally) collide with the user moving up. The player loses if there’s a collision.

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.

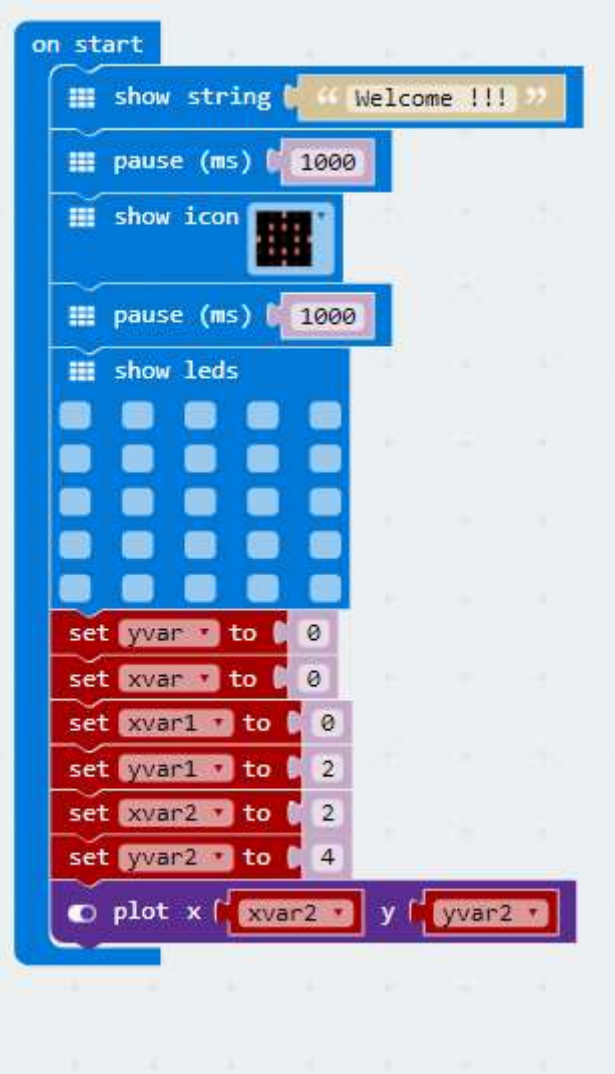
5. Let's write some code

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 Block

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.



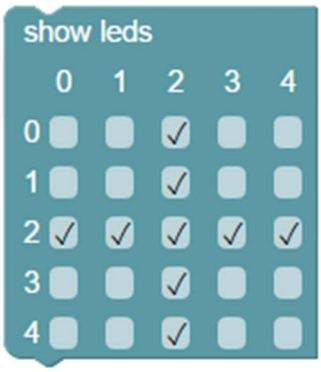
a. The "on start" block of code simply shows a string

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 again pause briefly for a second
e. We then clear the screen
f. We use the "set" commands to initialize a series of variables required
g. Finally we light up an LED using the plot command. This LED will symbolize the person wanting to cross from one end of the street to the other.

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



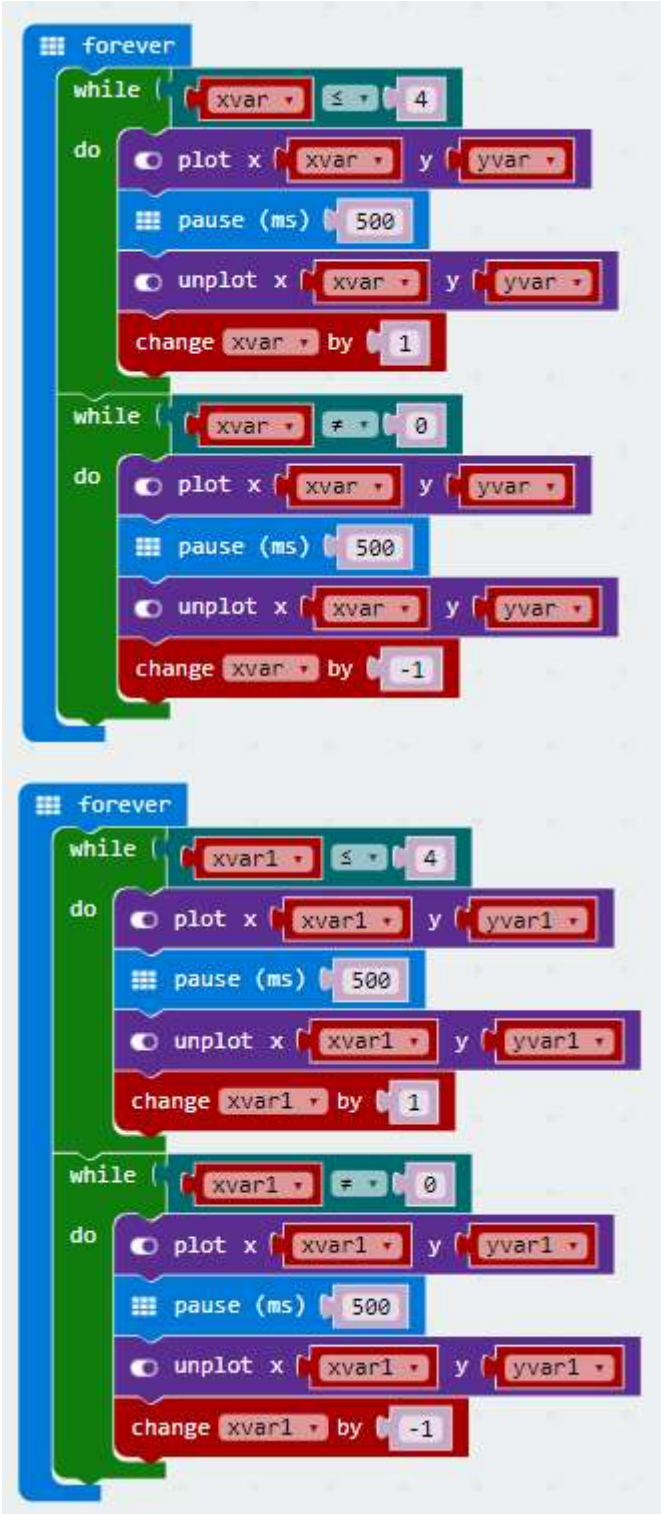
In this tutorial we will use variables (Remember the data section in Scratch?) to represent the X, Y co-oridinates and light up the LED's as we increase and decrease the value of the variables. There's a lot more to X, Y co-ordinates that just what's been covered here however we'll limit ourselves to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

b. Forever Block

The next block of code deals with the main content where we create our "Crossy Roads" game.

In this section we use the "forever" block which runs code repeatedly until the board is reset or the power is turned off. This block of code focuses on creation of the traffic moving from

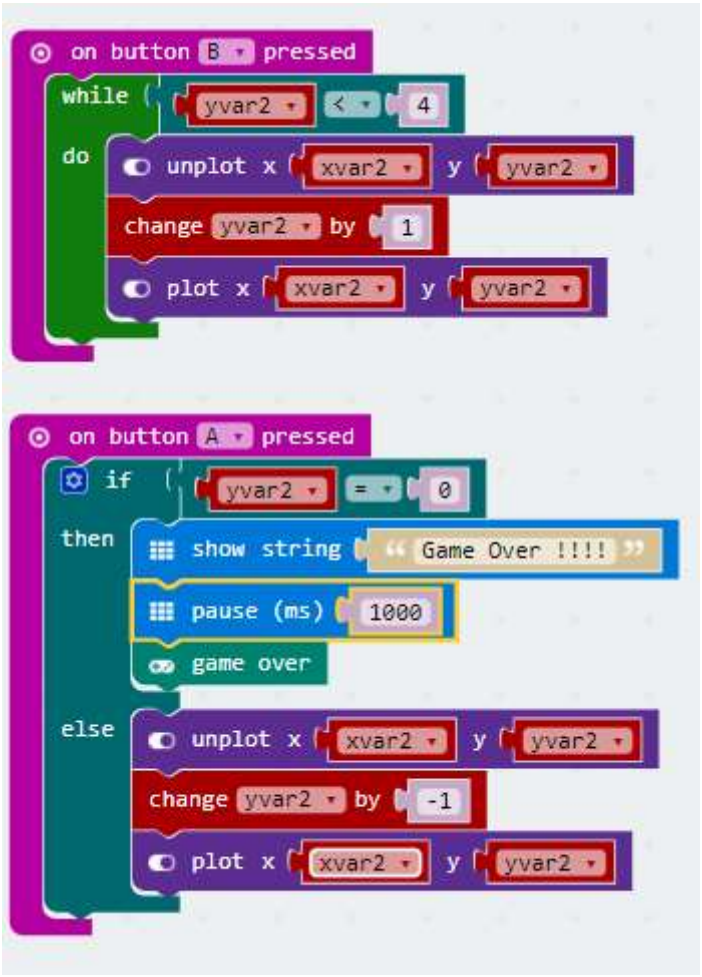
left of the screen to right and back on a continuous basis. The traffic is symblized by the LED's that we will light us using code. We will have two lines of traffic (LED's) to start with. You can add more lines of traffic as required.



a. There are two different sets of forever blocks that we will be making use of. The first set of the forever block will create a line of moving LEDS across the top of the screen.
b. The second forever block is very similar to the first one and will also create a line of moving LED's but a couple of lines down from the top of the micro:bit. So in essence we will be coding two moving LED lines across the micro:bit.
c. Start off by programming the first forever block and see the LED's streaming from one end of the screen to the other and back. Once you've completed coding the first forever block only then move to programming the second LED block.
d. The micro:bit has an inbuilt LED matrix referred as the 'screen'. LED coordinates starting at 0,0 top left corner and ending at 4,4 at the bottom corner will be lit up sequentially using code. Refer to the paragraph at the start of this section which covers off details of the X, Y co-ordinate system applied to the micro:bit.
e. A "while do" loop is used in this case to increase or decrease the value of the variables which form the co-ordinates of the LED's i.e. x, y. The LED co-ordinates are represented by variables xvar, yvar for the first line of LED's and xvar1, yvar1 for the second line of LED's in our program.
f. You'll notice that each of the "While Do" loops iterates between 0 – 4 the reason being that the value of X for the LED's varies from 0 -4. Y remains constant since the requirement is to create a sense of LED's flashing from left to right i.e. from one side of the screen to the other.
g. We have to use two sets of co-ordinates because we are programming two LED lines on the screen. Each LED line requires the use of one set of x,y co-ordinates. If you plan to create additional LED lines please remember to create new variables which you can then use for your own x,y co-ordinates.
h. The "while do" loops are programmed such that they gradually increase the x variable so that the lit up LED moves from left of the screen to the right and then once it reaches the end (right hand side) it decreases the value of the x variables so that the lit up LED's move from right to left of the screen.

c. On Button Press Block

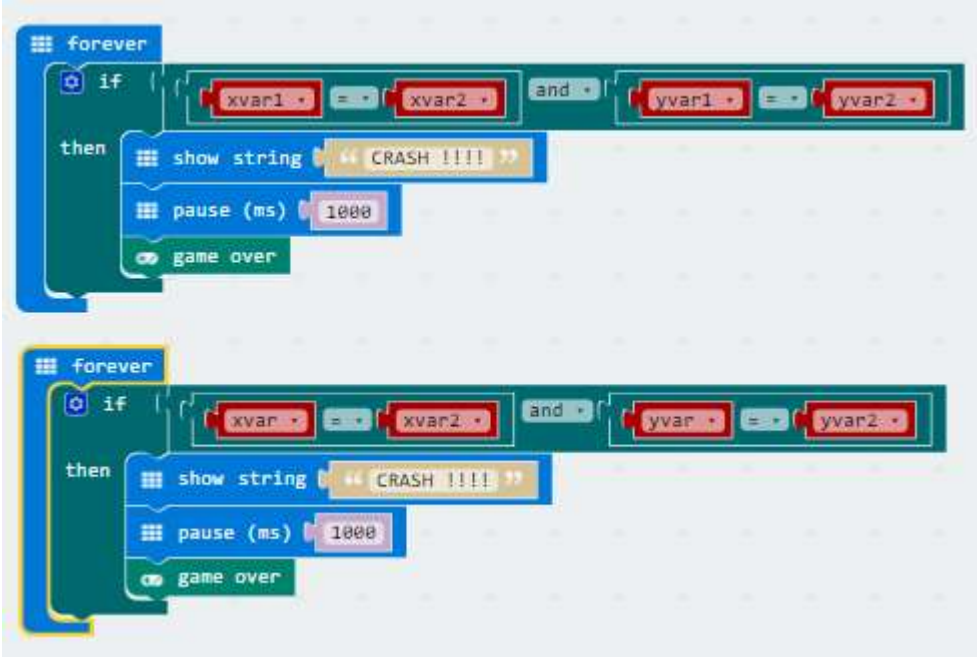
Now that we have created the two lines of traffic lets now create the capability to control the user (represented by the LED light at the bottom row, in the middle) using the two buttons on the micro:bit.



a. From the work we've done earlier you would know that the X, Y co-ordinates for the LED's on the micro:bit range from 0, 0 to 4, 4. This implies that X goes from 0 to 4 and Y goes from 0 to 4 as well.
b. The first code block above uses a "while do" loop to light up (plot command) and turn off (un-plot command) the LED. The first code block is designed to give us the "move downwards" functionality to move the person crossing the road.
c. The "while do" loop for controlling buttong B looks for a condition "yvar2 < 4" (Y co-ordinate lesser than 4) because there's no use moving the LED beyond Y=4 which is the bottom of the screen.
d. Can you think through the need to use the change "yvar2 by 1" and what it's purpose is?
e. The second code block above uses a "if then else" loop to light up (plot command) and turn off (un-plot command) the LED. The second code block is designed to give us the "move upwards" functionality to move the person crossing the road.

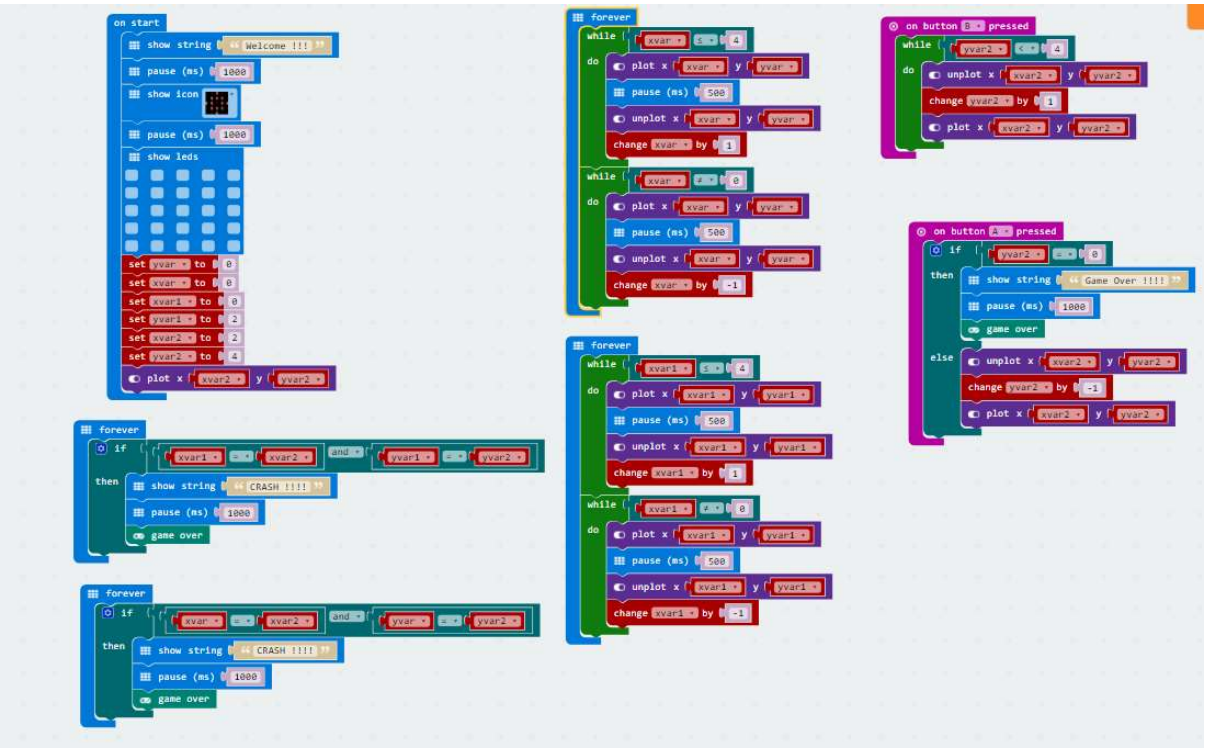
f. The second code block uses the “if then else” block for a condition where “yvar = 0” (Y co-ordinate = 0). Reaching the top of the screen is the end of the objective and the user has successfully crossed the road, resulting in winning the game.
g. Can you think through the need to use the change “yvar2 by -1” and what it’s purpose is?

The final set of code blocks are designed to catch the person crossing the road, bumping into traffic. Without these blocks of code you can still play the game but if the person navigating the road happens to cross the path of the moving traffic (LED’s moving from left to right and back) nothing happens. We obviously want to declare the user as having won the game only if he/she has been able to cross the road from bottom of the screen to the top without bumping into the traffic (LED’s moving from left to right of the screen and back).



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

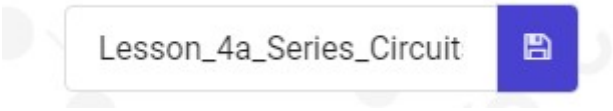
Tutorial 2.1b – Crossy Roads



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 this far.

For those who want to stretch it a bit further -

1. Try making it a bit more challenging by decreasing the time (pause) between the traffic (moving horizontally from left to right and back) LED's turning on and off.
2. Could you use a variable to randomize the time (pause), 100-500ms between the traffic (moving horizontally from left to right and back) LED's turning on and off.
3. Can you code moving both the lines vertically down (X = 0 for first line and X = 2 for second line) for every second iteration. The program is currently designed to keep running with both the lines of traffic moving from left to right at the same X level (X = 0 for first line and X = 2 for second line).
4. Are you able to try adding a new row of traffic moving from left to right? Can you also code it to randomly change the pause time i.e. pause time (100-500 ms) for turning the traffic LED's on and off.
5. Try implementing any other ideas you might have. Show off your ideas during "Show & Tell" in class.