

# THE PARENT'S COMPLETE GUIDE TO THE BBC micro:bit



Includes easy-to-understand  
section targets!

# THE BBC micro:bit STORY



Learn the history and the main aims behind the BBC micro:bit project

**The BBC has a long and proud history when it comes to championing education in the UK. Their involvement has gone well beyond just their TV programs. If like me, you are of a certain age, you will remember the BBC micro – you probably even used them in school.**

Released at the end of 1981, as part of the BBC's Computer Literacy Project, the BBC micro was used in schools, colleges, polytechnics and universities to teach youngsters the skills that would be needed in the emerging world where the use of computers was beginning to become a reality for more and more people. There was no internet back then, just television, teaching institutions and a handful of magazines so it was quite difficult to learn these skills at home on your own. In response to this, and also to help provide resources for educators, the BBC first began to televise live lessons, where they set coding challenges in much the same way as they do now for the BBC micro:bit.

Although a few decades have passed and computer technologies have come along in leaps and bounds, some of the same issues remain. Not enough children are coming out of full-time education fully conversant in the skills that the modern world requires. To help tackle this, the BBC have once again stepped up to the plate and devised a new piece of teaching/learning tech, the BBC micro:bit, to help educators engage and inspire young people in the right ways to help address the skills gap in this country.

In order to facilitate this, the BBC gathered together a team of 29 companies and organisations to help design, make and launch the **BBC micro:bit** and also launched their own **Make It Digital** initiative to focus all of this and deliver resources to the users of the micro:bit. The partners are;

Kitronik, ARM, Barclays, Bluetooth SIG, Cisco, National STEM Centre, Nordic Semiconductor, Institution of Engineering and Technology, Lancaster University, Microsoft, Cannybots, Creative Digital Solutions, Code Club, Code Kingdoms, CoderDojo, CultureTECH, element14, London Connected Learning Centre, MyMiniFactory, NXP, Python Software Foundation, Samsung, ScienceScope, STEMNET, Tangent Design, Technology Will Save Us, TeenTech, The Tinder Foundation, The Wellcome Trust.



^ The original BBC Micro, and a new BBC micro:bit

The BBC microbit project is built on the legacy of the **BBC Micro**, which was put into the majority of schools in the 1980s and was instrumental in the careers of so many of today's technology pioneers. Computing and digital technology have become commonplace since then, but for many, the emphasis has shifted from creation to consumption. The BBC micro:bit and the wider BBC Make it Digital initiative, aims to help redress the balance and turn youngsters from consumers of media into coders and makers.

The BBC managed the project for a further year after the rollout before handing over the reins to the newly formed, not-for-profit BBC micro:bit educational Foundation. Their goal is to continue the great work done by the BBC and to also make the micro:bit easily available and affordable around the world.

# WHAT IS THE BBC micro:bit?



Read an overview of what the BBC micro:bit is and the technology it contains

**The BBC micro:bit is a powerful, handheld, fully programmable computer, designed by the BBC and a number of partners to encourage children to get actively involved in writing software and building new things that will be controlled by it.**

Featuring an onboard 5x5 LED Matrix, two integrated push buttons and a compass, accelerometer and Bluetooth, the device makes a great introduction to the world of programmable components and the wider internet of things.

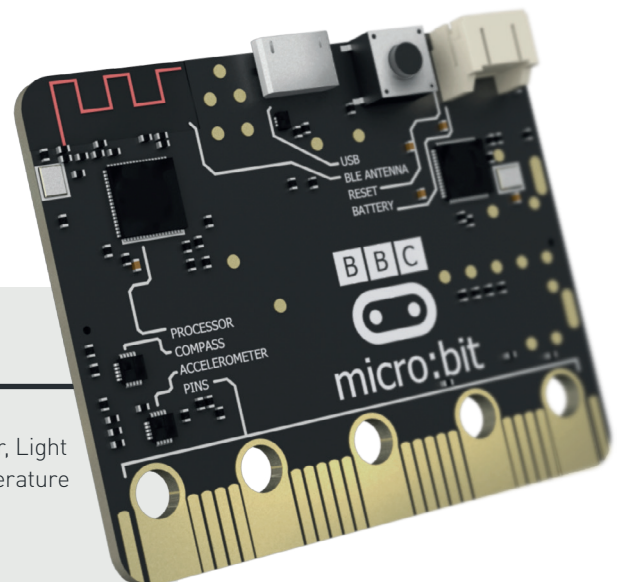
Write your code using one of the easy to use editors provided on the Foundation micro:bit website. Connect your BBC micro:bit to your computer via USB. Press the download button in the editor and then drop the downloaded file directly onto your BBC micro:bit. There are also Apps for Android and iOS devices.

Wirelessly connect and interact with the world around you. Use Bluetooth Low Energy to connect to mobile phones and tablets, take a selfie or drive the music in your playlists. You can also code the micro:bit to interact with other micro:bit's, using the radio blocks in the Microsoft MakeCode Editor. All of this in a device only 5cm wide!

The main I/O rings can be used as outputs to control LEDs, motors and much more, and also as inputs to connect external sensors and switches. The main rings are large enough that they can easily be connected with crocodile clips.

Whilst the main five rings enable you to connect the micro:bit to external devices and components using crocodile clips, accessing the rest of the pins requires something a little more precise than a clip. An edge connector breakout board allows you to conveniently access every pin on the BBC micro:bit. Just slot the micro:bit into place and you can use either M/F or M/M jumper wires to connect the breakout board to your circuit/breadboard.

Although the micro:bit has been designed with connectivity in mind, the amount of current that can be drawn from it is quite limited, so in order to connect the micro:bit to things like motors, a motor driver board is required. This will allow you to drive two motors with full forward and backwards control. Additions such as Motor Driver Boards and Edge Connector Breakout Boards expand the possibilities of the micro:bit's connectivity, paving the way to such things as Robotics.



## BBC micro:bit FACT FILE

<b>INTRODUCED</b>	February 10 <sup>th</sup> 2016	Accelerometer, Light Sensor, Temperature Sensor, Radio.
<b>DIMENSIONS</b>	4cm x 5cm	
<b>LEDs</b>	25 (5 rows of 5)	
<b>CONNECTIVITY</b>	Bluetooth LE, MicroUSB, Edge Connector, Two buttons, Compass,	

# POWERING THE BBC micro:bit



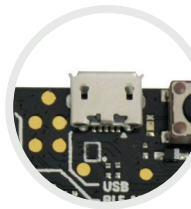
Learn all the ways in which the BBC micro:bit can be powered during use

As has already been highlighted, the BBC micro:bit is a highly versatile bit of educational tech and that versatility is also evident in the variety of ways in which it can be powered.



## JST CONNECTOR

The most common and convenient way of powering the BBC micro:bit would be the onboard JST connector. A simple 2xAAA battery holder with JST connector can be used for this purpose. This frees the micro:bit from your PC and gives you the freedom to take your projects anywhere.



## MICRO USB

The onboard micro USB connector, whilst very commonly available, is not a very portable solution, but could be of great use for projects where the BBC micro:bit stays in the same location as your computer.



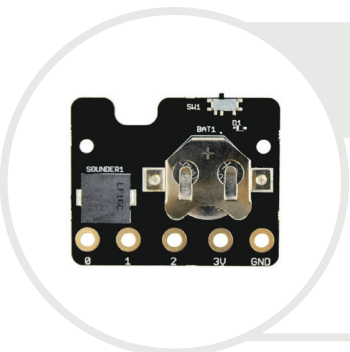
## GPIO PINS

It is also possible to power the BBC micro:bit via its GPIO (General Purpose Input Output) pins on the edge connector. The 3V and GND pins can be seen in the above image.



## POWER PADS


There are two pads on the back of the BBC micro:bit which can be powered with a direct connection. The 3V pin is the one closest to the edge connector, as indicated in the image.



## MI:power BOARD FOR THE BBC micro:bit

The MI:power board for the BBC micro:bit brings real portability to your projects. The stylish, lightweight PCB is designed to fit snugly against the BBC micro:bit and features a built-in buzzer, on/off switch and 3V coin cell holder, making it an ideal option for powering the BBC micro:bit.

# INTRO TO CODING THE micro:bit PART 1

 By the end of this section you will know how to create a program, put it onto the micro:bit, and then run the program.

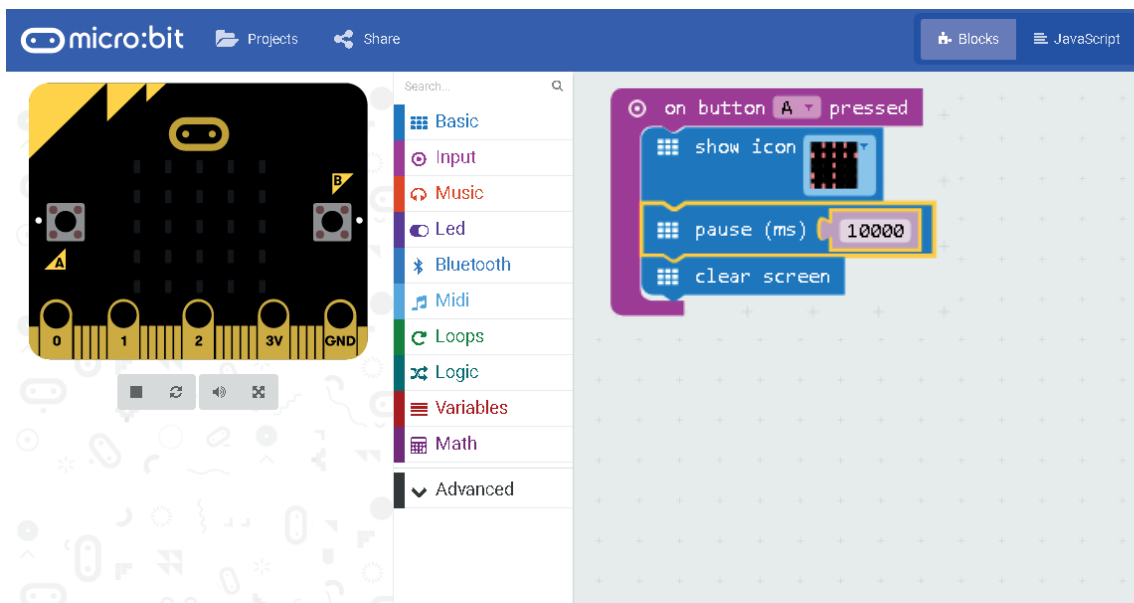
We are going to work through the three steps listed below to show you how easy it is to get a program onto the micro:bit and run it. For more information on using the Editor, visit our online guide at:

<https://www.kitronik.co.uk/makecode>

- 1 Creating a Program.
- 2 Putting the program on the micro:bit.
- 3 Running the program.

**Creating the program:** We are going to use the Microsoft MakeCode Editor to write some simple code that displays a picture on the micro:bit's LED matrix for ten seconds when you press the A button on the front of the micro:bit. After the ten seconds is up the LEDs will turn off.

- 1 Go to [www.microbit.org](http://www.microbit.org).
- 2 Select 'Let's code' in the top menu.
- 3 We are using the JavaScript Blocks Editor (MakeCode), the first editor in the list. Select the 'Let's Code' button. Then, in the top left of the editor, click Projects and select New Project.
- 4 Create the following code:



# INTRO TO CODING THE micro:bit PART 2



By the end of this section you will know how to create a program, put it onto the micro:bit, and then run the program.

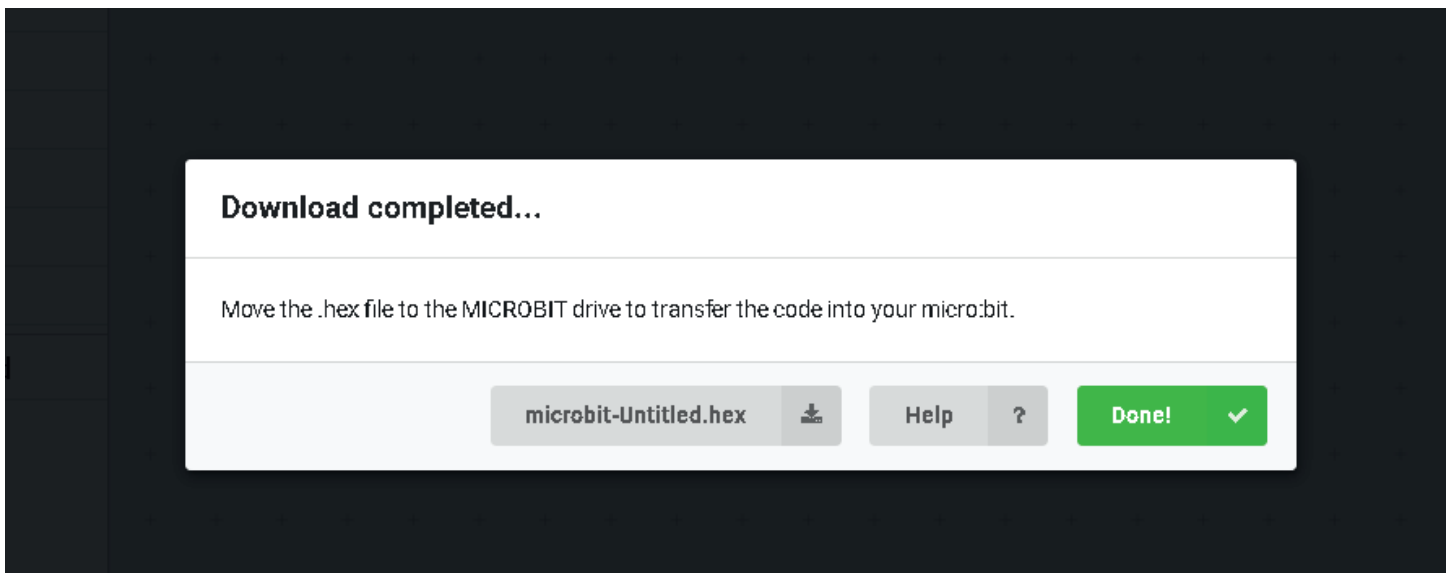
## Hints:

- Find the On Button A pressed block in the input menu and drag and drop it into the editing area.
- The other three blocks can be found in the Basic menu, drag them and drop them into place in the order shown.
- You can change the icon type on the icon block by mouse clicking on the picture.
- 1 second = 1,000 milliseconds, therefore, 10 seconds = 10,000 milliseconds.
- To edit the number, click on it and type in a new value.

## Putting the program on the micro:bit:

Connect the micro:bit to a spare USB input on your computer.

Click the download button at the bottom left of the editor window. On Windows, you will get the following pop up, the pop up may differ slightly on OSX or Linux.



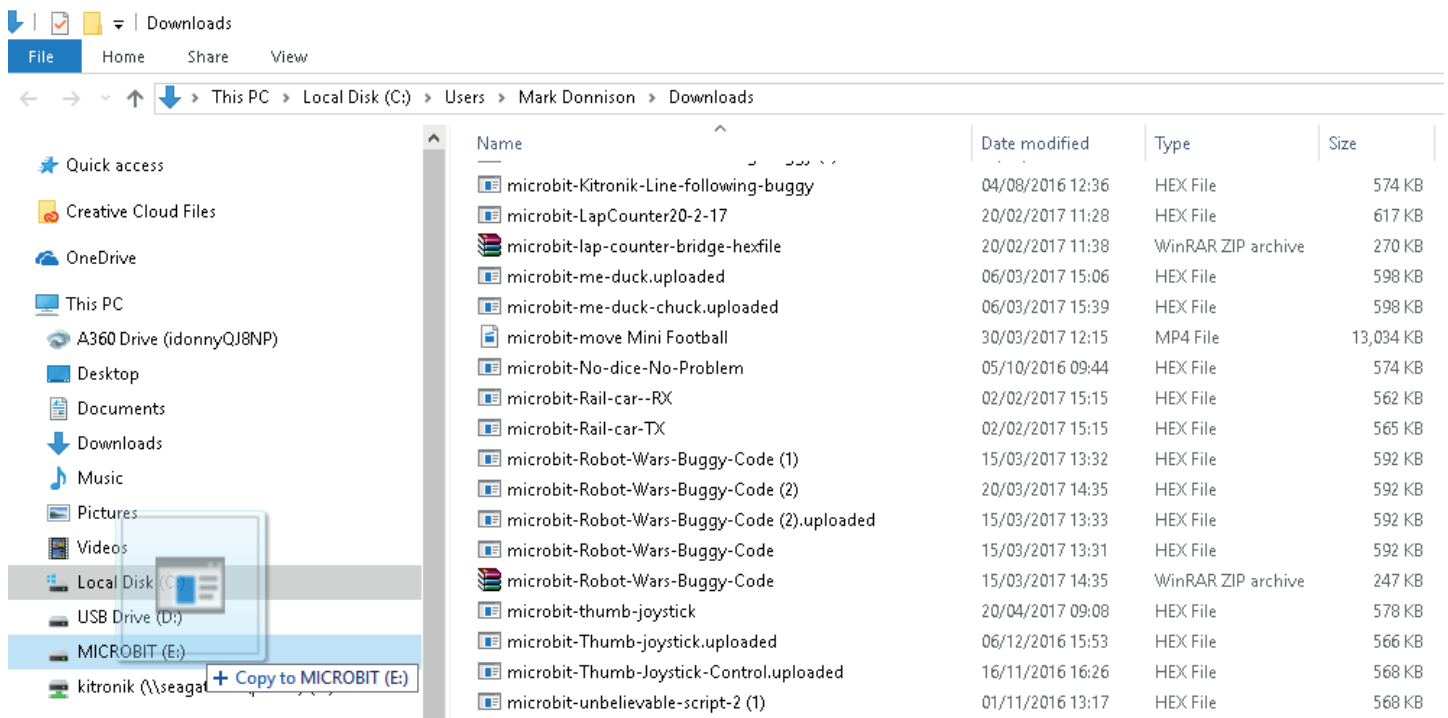
Your program will be downloaded into your default downloads folder. You can either click the 'Done' button or wait for the pop up to auto close.

# INTRO TO CODING THE micro:bit PART 3



By the end of this section you will know how to create a program, put it onto the micro:bit, and then run the program.


Locate the HEX file in your downloads folder using File Explorer (Windows) and drag it onto your micro:bit, which will appear as if it were a storage device. See below.



The light on the back of the micro:bit will begin to flash as the program is transferred, once the light stops flashing you know that the program has been transferred and is ready to run.

**Running the program:** Press the A button on the micro:bit and the icon you chose should be displayed on the LED matrix.

# CODING LANGUAGE OPTIONS & EDITORS

 To introduce you to the different coding options and editors that are available for use with the BBC micro:bit. We will also take a look at useful Android and iOS apps.

Although we are going to primarily use the MakeCode Editor for this guide, it's important to know that there are other coding languages and editor options available, and, some of the reasons why you might choose one over another.

## MakeCode & JavaScript Editor

This editor allows the user to code using blocks or JavaScript, or a mixture of the two. You are free to jump between blocks and JavaScript at any time and the editor will automatically convert all of your code as you do so. This is a great learning aid, not only can you create a program using blocks and then convert it to see what it looks like in JavaScript, you can also write your program in JavaScript and if you don't know how to code something you can pop into blocks and code it there before jumping back to JavaScript. This will speed up your workflow and allow you to learn what JavaScript code is behind the blocks in the block editor.

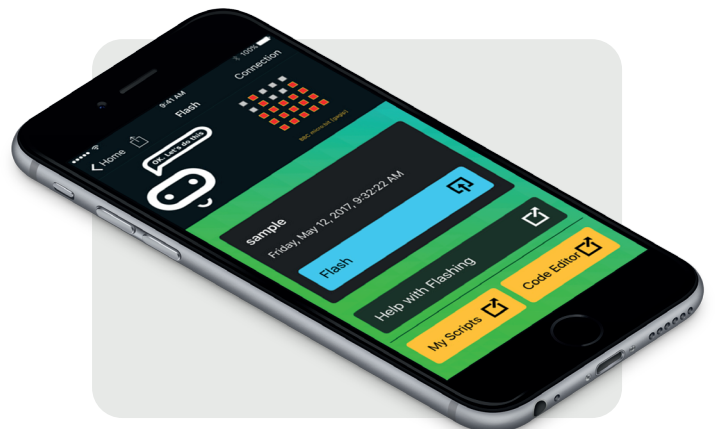
## Python Editor

This is a newer version of the original Python editor; the latest version gives you access to parts of the micro:bits functionality that the original editor didn't and it also has code snippets. The code snippets give you access to a range of pre-made images and music as well as other bits of code designed to help you along and to speed things up.

**Both of the above editors are accessed through a web browser, such as Chrome or Safari. The main pro of this is that no software needs to be installed in order for you to start coding. Traditionally, installing the coding environments and the necessary prerequisites was often enough to put many people off going any further, doing everything through the browser means it's often as easy as following a link. The two main cons of this are that you need an internet connection to use the editors and that they are designed with desktop and laptops in mind. But, as with most things, there is an App for that;**

## The Mobile App

The official micro:bit mobile App allows you to create code on either your phone or tablet and then flash it to your powered and paired micro:bit over Bluetooth, no wired connection required. This handy app means you can continue to code on the go, making it ideal for someone who might be travelling in the back of a car. It's got to be better than "I spy" or counting red cars. The official App by Samsung is available for both Android and iOS, via the respective App Stores.





# HELLO WORLD PART 1

## WHAT WILL WE LEARN

This challenge requires that we make use of the buttons on the front of the micro:bit to tell the micro:bit to do something, in this case to display things on the micro:bit's LED matrix. This is easy to code using the blocks in the Microsoft MakeCode Editor.



## THE INPUT MENU

In the above diagram, you can see that the 'on button pressed' block can be found in the "Input" menu of the editor. Once you drag this block into your program, you can then choose which button you wish the block to listen for by clicking on the 'A' and changing it accordingly.

This block will 'listen' for the correct key press and then run through any blocks that are placed inside it and perform any actions that they dictate.

## THE LED MATRIX

The aim in this challenge is to make things appear on the LED matrix. There are a number of ways to make this happen. For now, we are going to concentrate on using blocks found within the "Basic" menu.

The Show LEDs block and the Show String blocks offer the easiest way of delivering this particular challenge. We can type any text we like into the Show String block and we click in the 25 squares on the Show LEDs block to draw a pattern.

Continued on next page >

# HELLO WORLD

 PART 2

## THE CHALLENGE

Using the MakeCode Editor, write code to...

- 1 When the A button is pressed, display a smiley face on the micro:bit's LED matrix.
- 2 When the B button is pressed, display the "Hello World" message across the LED Matrix.
- 3 When both A + B are pressed together, clear the screen (LED Matrix).



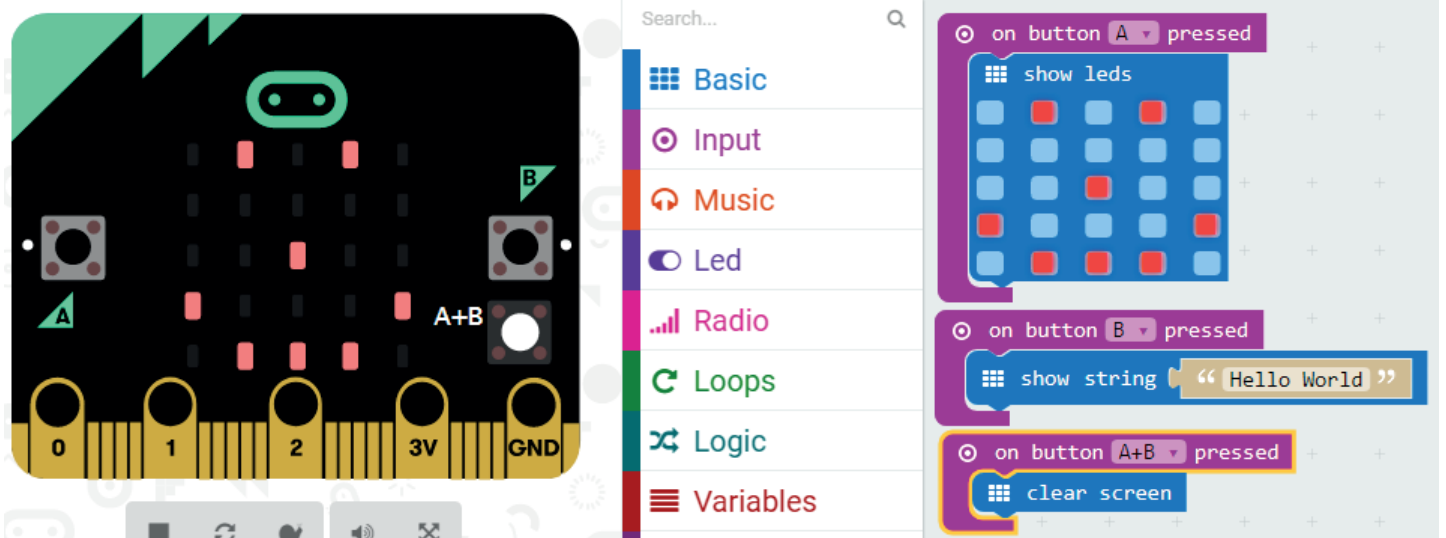
HOW ARE YOU ENJOYING THE CHALLENGES?

Tweet us  @kitronik with the hashtag #kitronikcodechallenge

# HELLO WORLD

**SOLUTION – PART 1**

## THE SOLUTION



You can see in the above example one method of delivering this challenge’s objectives. The step by step instructions below will show you how we did it and the Other Information section at the end will cover any further useful information.

- 1 In the Input menu find the ‘on Button A pressed’ block and drag three of them into the editing area. 2 of them will automatically be greyed out. We will fix this in step 2.
- 2 Change one of the three blocks to ‘on B pressed’ by clicking the A and selecting B from the list. Change the last greyed out block to ‘on A+B pressed’ in the same way.
- 3 In the Basic menu, get a Show LED block and place it in the ‘on button A pressed’ block and use your mouse to draw the smiley face.
- 4 Also in the Basic menu, get a ‘show’ string block and place it in the ‘on button B pressed’ block. Click in the box and use your keyboard to replace the string to ‘Hello World’.

Continued on next page >

# HELLO WORLD

SOLUTION – PART 2

5 Click on the Basic menu. Below the Basic block it should now display the sub menu 'More'. In this sub menu, you will find the Clear Screen block, which you should place in the 'on A+B'.

6 Test your program on the simulator before transferring the code to your micro:bit.

## OTHER INFORMATION

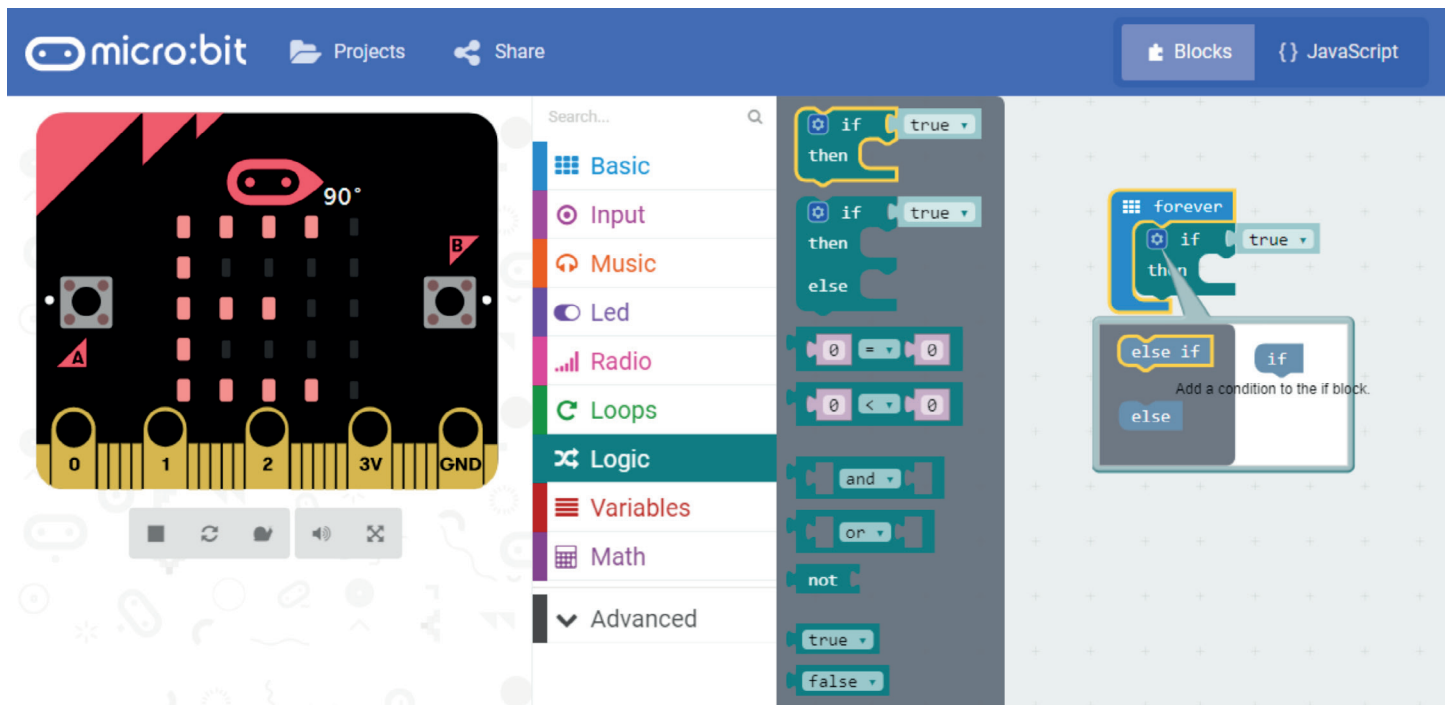
You may have noticed that the A+B button on the simulator wasn't there until you changed one of the three 'on button A pressed' blocks to an 'on A+B button pressed' block. This is a super useful feature of the editor that ensures the simulator doesn't get too cluttered with things you don't need. As you make use of them in your code they will automatically appear on the simulator.

Here are some of the blocks that you might find useful for this challenge.

# WHERE ARE WE GOING? SETUP – PART 1



In this challenge, we are going to learn how to use the micro:bit's magnetometer. The magnetometer chip detects where magnetic north is which allows you to determine which way the micro:bit is facing, from 0 to 360 degrees. We will learn how to then convert the degrees into N, S, E or W and display the result on the LED Matrix.



## IF TRUE/FALSE THEN DO LOOP

This loop can be found in the **Logic** menu and it allows us to run code when the conditions that we set are met. For our program, if the heading is a specific number of degrees then it will need to display the correct heading. This type of loop offers us the easiest way to make this happen.

As we have four possible directions we will need to add a couple more **else if**'s to the block. To do this, click the blue cog icon and a box will open up. You can drag the option you want from the dialogue box and drop it under the **if** in the dialogue box. The actual loop in the workspace will change automatically as you add more statements.

## LOOPING FOREVER

In the above diagram, you can see that the **forever** loop block can be found in the **Basic** menu of the editor. This block will start working as soon as the micro:bit is powered up and will keep executing the code placed inside it until the micro:bit is powered off. This is the perfect loop type for programs that need to perform the same instructions over and over again, as is often the case for programs that are monitoring something.

Continued on next page >

Here are some of the blocks that you might find useful for this challenge.

# WHERE ARE WE GOING?

SETUP – PART 2

## VARIABLES

A variable is like a box that you can put things in. You can also add to or subtract from its contents at any time. You can place numbers, names, or text strings into variables that can be used by your program. Variables also have a label or a name that we can use to call them up at any time in our program.

If you look in the **Variables** menu you will see the **make a variable** button at the top. When you click this, it opens a dialogue box that asks you for the name (label) of your variable. Once it is named you can then write code to determine what is to be put in the box and how it will be used in your program. When choosing a name for your variable, it makes sense to choose a name that describes what the variable is. If your program is going to contain multiple variables it can become quite confusing if the variables aren't suitably named.

## SHOW STRING

The show string block allows us to display text on the micro:bit's LED Matrix. For the purposes of this challenge we only need to display one letter on the LED Matrix at a time; N, S, E or W. To replace the default **Hello**, just delete "Hello" and replace it with the letter you need to display.

## THE CHALLENGE

Using the MakeCode Editor, write code to...

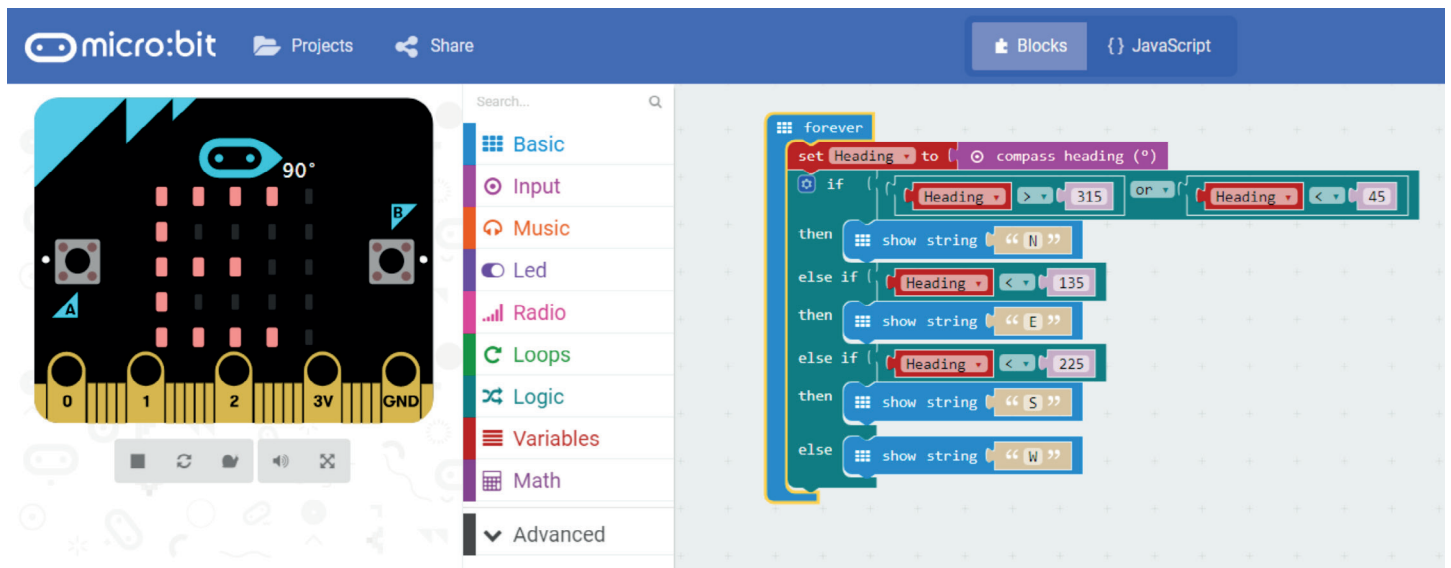
- 1 Monitor the compass heading and store it in a variable.
- 2 Convert the stored degrees value in terms of N, S, E or W.
- 3 Display the direction on the micro:bit's LED Matrix.
- 4 Ensure that the display updates in real time as the micro:bit is moved.

HOW ARE YOU ENJOYING THE CHALLENGES?

Tweet us  @kitronik with the hashtag #kitronikcodechallenge

# WHERE ARE WE GOING? SOLUTION – PART 1

## THE SOLUTION

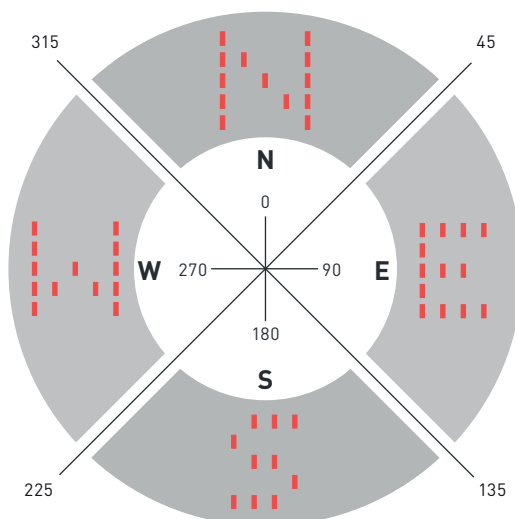


- 1 When you start a new project the forever loop and **start** block will already be in the editor's workspace. You can also find the forever loop in the **Basic** menu.
- 2 In the variables menu at the top is the 'Make a variable' button. Click the button and name your variable. We chose 'Heading' but you can give it any name you like.
- 3 In the **Variables** menu you will find the **Set variable to** block. By default, it will be set to item, to change this click on item and choose **Heading** from the list.
- 4 Now we need to put the compass heading into our variable. You will find the **compass heading** block in the **Input** menu. You can pick this block up and attach it directly to the **set Heading to** block.
- 5 The **if true then** block can be found in the **Logic** menu and once we place it into the workspace we need to extend it with another two **if else** lines. To do this, click on the blue cog icon and a dialogue box will open up. You can drag **else if** from the dialogue box and drop it under the **if** in the dialogue box.

# WHERE ARE WE GOING? SOLUTION – PART 2

**5** In the diagram you can see how the degrees that the micro:bit magnetometer outputs relates to the points on the compass. We do that by comparing our variable **Heading** with the numbers on the below circle.

In English, it would look like; If the variable **Heading** is more than 315 degrees but less than 45 degrees **then** the direction is North. **Else if** it's less than 135 **then** the direction is East. **Else if** it's less than 225 **then** the direction is South. **Else** it's West.



**6** The block that contains the **or** and the block that displays the less than **<** and greater than **>** symbols can be found in the **Logic** menu. You can change whether it's a greater than or less than symbol by clicking on the symbol and choosing the correct one from the dropdown list. You can see how they slot together in the code diagram above step 1.

### OTHER INFORMATION

- You may have noticed that the micro:bit logo on the simulator can now be moved around with your mouse to simulate the micro:bit facing in a different direction. This is a super useful feature of the editor that ensures the simulator doesn't get too cluttered with things you don't need, as you make use of them in your code they will automatically appear on the simulator.
- When running code that uses the micro:bit's onboard Compass for the first time, the micro:bit will ask you to draw a circle by tilting the micro:bit. This is to calibrate the micro:bit.

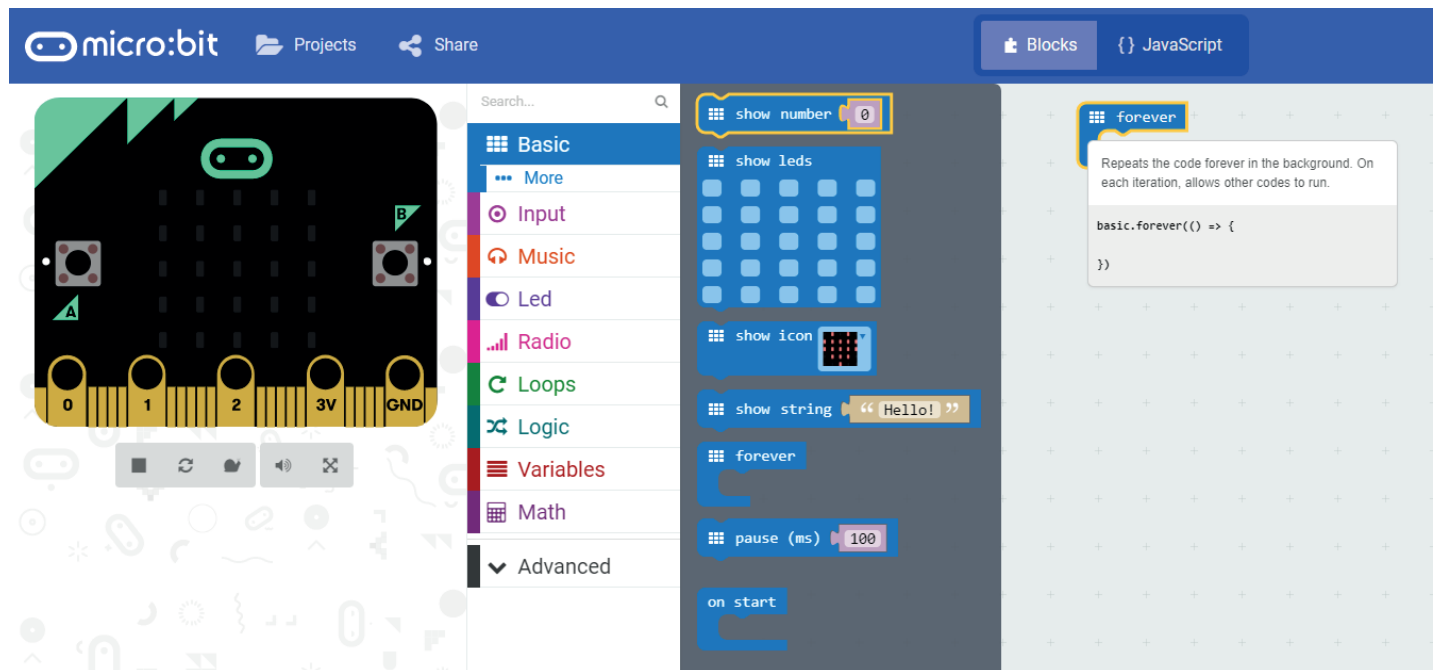


# IS IT GETTING HOT IN HERE? PART 1

**WHAT WILL WE LEARN**

This sensor challenge requires that we make use of the micro:bit's temperature sensor to measure the temperature of a room and then display it on the LED matrix.

The temperature sensor is actually a part of the main CPU but as the CPU usually runs fairly cool, we can use its temperature sensor to return a reading for the ambient temperature. We wouldn't go as far as to say that the temperature will be accurate, but it should register changes in temperature fairly accurately.


**LOOPING FOREVER**

In the above diagram, you can see that the 'forever' loop block can be found in the Basic menu of the editor. This block will start working as soon as the micro:bit is powered up and will keep executing the code placed inside it until the micro:bit is powered off. This is the perfect loop type for programs that need to perform the same instructions over and over again, as is often the case for programs that are monitoring something.

Continued on next page >

# IS IT GETTING HOT IN HERE? PART 2

## WHAT WILL WE LEARN

For this challenge, rather than display a number, we will write code to display the temperature as a graph on the micro: bit's LED Matrix. It will react to changes in real time and update its display, without the need for someone to press buttons or shake the micro: bit.

## PLOT BAR GRAPH

The plot bar graph block can be found in the LED menu and will display a graph based on the parameters we set. In this case, we want it to display the temperature, and as we are storing the temperature reading in a variable we can use the variable directly for the value and choose an upper limit that will result in a meaningful display. Ideally, we would want the display to be fully lit up when the temperature reached the maximum likely temperature for your part of the world.

## A VARIABLE

A variable is like a box that you can put things in, you can also add to or subtract from its contents at any time. You can place numbers, names, or text strings into variables that can be used by your program. Variables also have a label or a name that we can use to call them up at any time in our program.

If you look in the variables menu you will see the 'make a variable' button at the top, when you click this it opens a dialogue box that asks you for the name (label) of your variable. Once it is named you can then write code to determine what is to be put in the box and how it will be used in your program. When choosing a name for your variable, it makes sense to choose a name that describes what the variable is. If your program is going to contain multiple variables it can become quite confusing if the variables aren't suitably named.

Let's see if we can use what we've learnt above to create code to deliver the challenge objectives below;

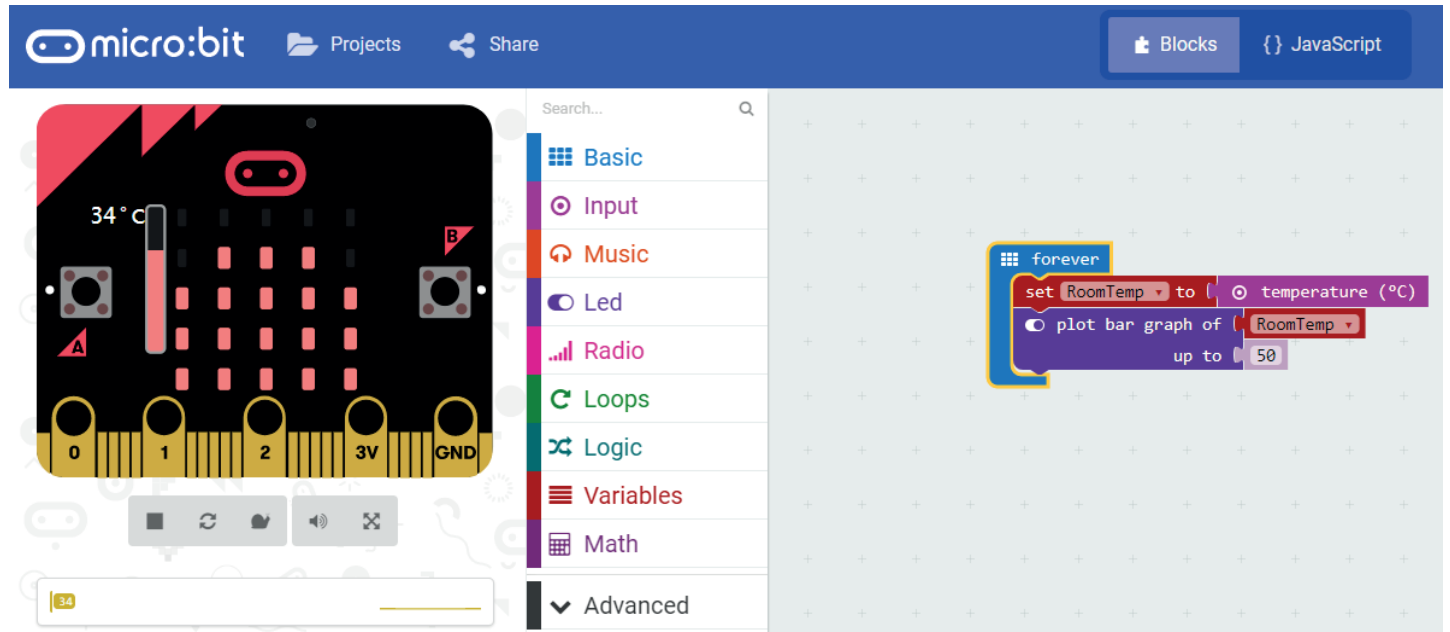
## THE CHALLENGE

Using the MakeCode Editor, write code to...

- 1 Monitor the ambient temperature and store the reading in a variable.
- 2 Display the reading on the LED Matrix as a Bar Graph.
- 3 Ensure that the display is updated in real time.

# IS IT GETTING HOT IN HERE? SOLUTION – PART 1

## THE SOLUTION

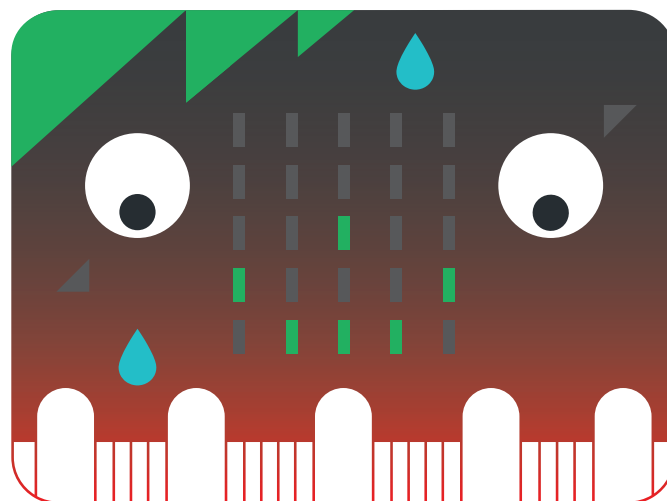


- 1 When you start a new project the forever loop and 'on start' block will already be in the editor's workspace. You can also find the forever loop in the Basic menu.
- 2 In the variables menu at the top is the 'Make a variable' button. Click the button and name your variable. We chose 'RoomTemp' but you can give it any name you like.
- 3 In the Variables menu you will find the 'Set variable to' block. By default, it will be set to item, to change this click on item and choose RoomTemp from the list.
- 4 Now we need to put the temperature reading into our variable. You will find the temperature block in the 'Input' menu. You can pick this block up and attach it directly to the 'set RoomTemp to' block.

Continued on next page >

# IS IT GETTING HOT IN HERE? SOLUTION – PART 2

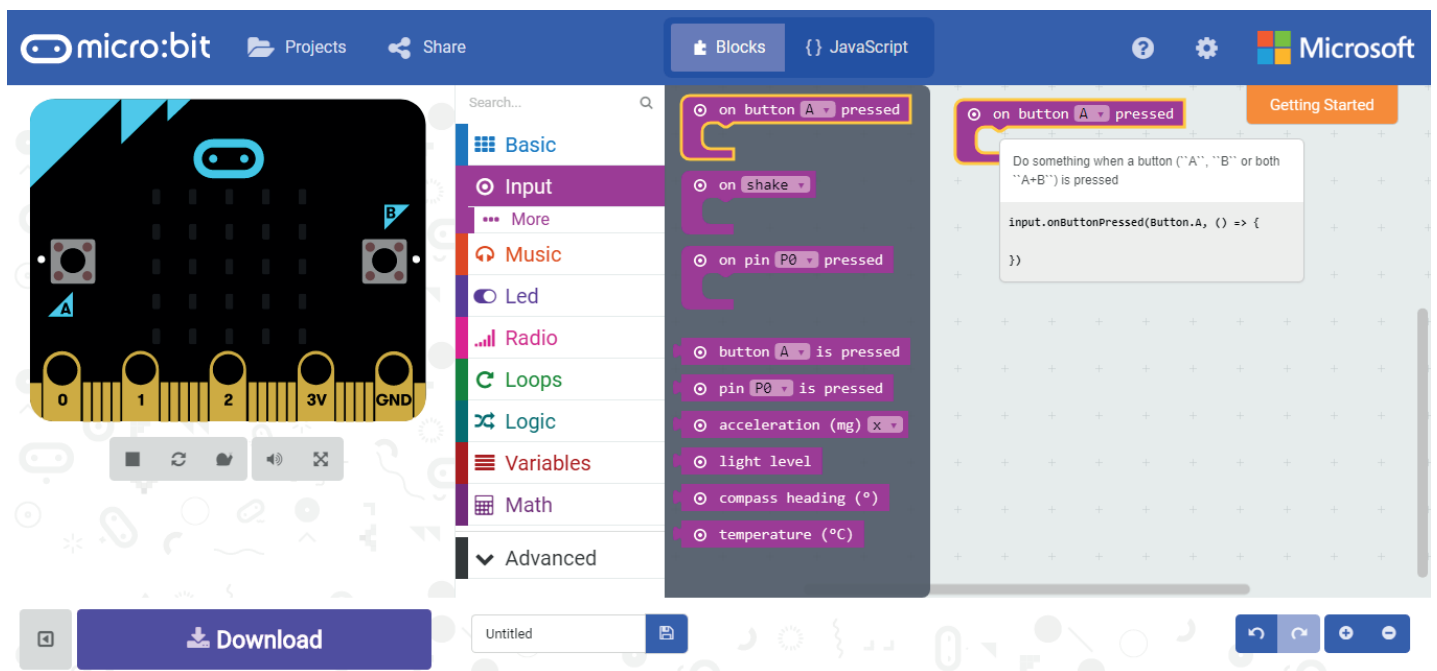
- 5 The 'plot bar graph of' block can be found in the 'LED' menu. Once we've chosen the parameters, this will automatically be displayed on the LED matrix when the code is run.
- 6 The 'plot bar graph' has two parameters that we can set, the value and the high value. In the Variables menu, find the RoomTemp variable block. This should be at the top directly below the 'Item variable' block. Drag it over to the 'plot bar graph' block and place it in the value slot.
- 7 For the high value we would ideally like to choose a number that was the maximum expected temperature so that the display is meaningful. A full display means hot and a nearly empty display means cold. We chose 50, not because we expect it ever to reach 50 but because the LED Matrix contains 25 LEDs, so each LED would then represent 2 degrees C.



# COUNT ON IT PART 1



In this challenge, we will learn how to turn the micro:bit into a button activated counter, such as you might use to count people in and out of an event. We will need to create a variable to store the counted number in and we will have to create code that will allow us to add people as they enter and subtract people as they leave. We will also limit the number of people allowed into the event and we will also ensure that there can never be a minus number of people.



## THE INPUT MENU

In the above diagram, you can see that the 'on button pressed' block can be found in the "Input" menu of the editor. Once you drag this block into your program, you can then choose which button you wish the block to listen for by clicking on the 'A' and changing it accordingly.

This block will 'listen' for the correct key press and then run through any blocks that are placed inside it and perform any actions that they dictate.

## THE LED MATRIX

The aim in this challenge is to make things appear on the LED matrix. There are a number of ways to make this happen.



Continued on next page >

**HOW ARE YOU ENJOYING THE CHALLENGES?**

Tweet us @kitronik with the hashtag #kitronikcodechallenge

# COUNT ON IT

## PART 2

### VARIABLES

A variable is like a box that you can put things in. You can also add to or subtract from its contents at any time. You can place numbers, names, or text strings into variables that can be used by your program. Variables also have a label or a name that we can use to call them up at any time in our program.

If you look in the variables menu you will see the 'make a variable' button at the top. When you click this, it opens a dialogue box that asks you for the name (label) of your variable. Once it is named you can then write code to determine what is to be put in the box and how it will be used in your program. When choosing a name for your variable, it makes sense to choose a name that describes what the variable is. If your program is going to contain multiple variables it can become quite confusing if the variables aren't suitably named.

The Boolean Comparison blocks can be found in the Logic Menu and they are used to compare one thing with another. We use them in this challenge to ensure that our count cannot go above 10 or below 0. We drag and drop a variable into either of the two values or type a value in. You can change the condition, Greater than > or less than < , by clicking on the condition and selecting the desired option from the list.

### LOOPING FOREVER

In the above diagram, you can see that the 'forever' loop block can be found in the Basic menu of the editor. This block will start working as soon as the micro:bit is powered up and will keep executing the code placed inside it until the micro:bit is powered off. This is the perfect loop type for programs that need to perform the same instructions over and over again, as is often the case for programs that are monitoring something.

### IF TRUE/FALSE THEN DO LOOP

This loop can be found in the Logic menu and it allows us to run code when the conditions that we set are met. In this challenge, we will use one of these loops to ensure that we never have a minus number of people.

### ON START

The 'on start' block can be found in the basic menu and is used to run a command, or a set of commands, when the micro:bit is powered on. This block is often used to declare a variable and give it a starting value or to display a message or picture on the LED Matrix.

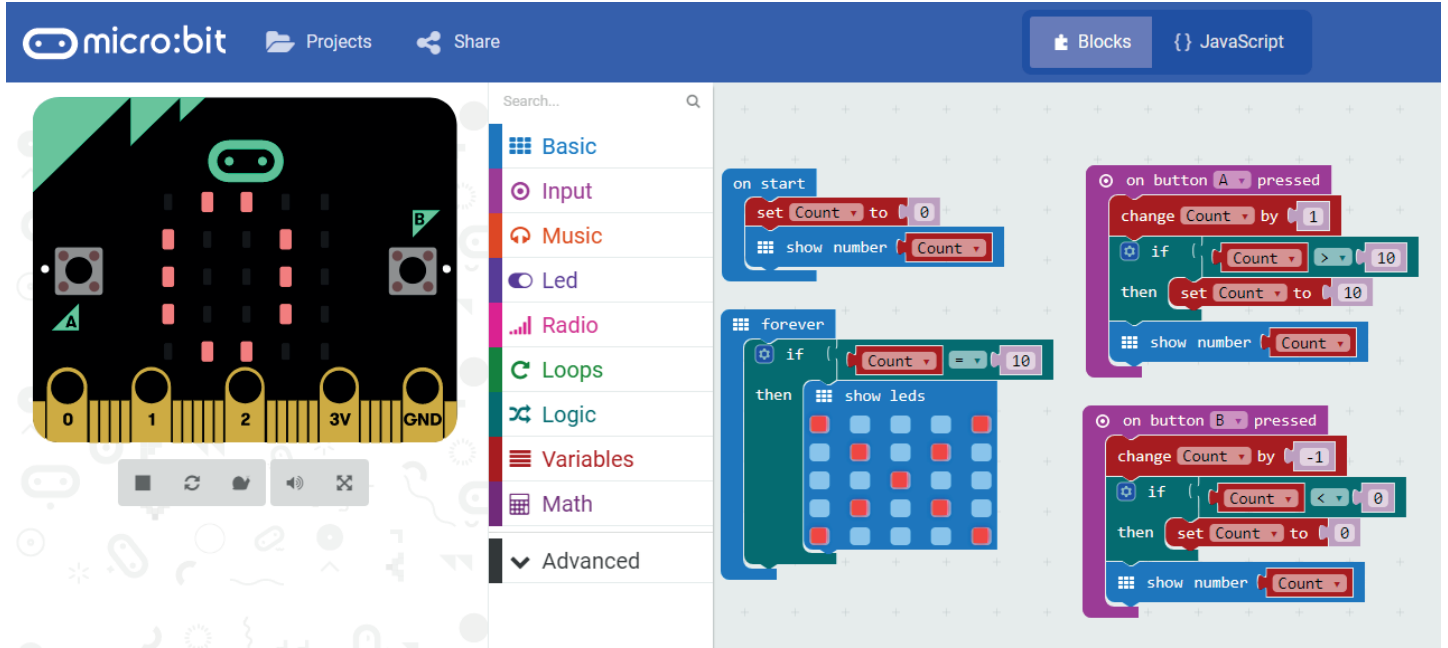
## THE CHALLENGE

Using the MakeCode Editor, write code to...

- 1 Count up by one with a button press.
- 2 Count down by one with a button press.
- 3 Limit the number of people allowed to 10.
- 4 Ensure that the number of people cannot exceed 10 or be less than zero.

# COUNT ON IT SOLUTION – PART 1

## THE SOLUTION



- 1 For this challenge we want our display to always show the number of people inside the event, including when we first turn the micro:bit on. The best way to achieve displaying the number on start-up is to use the 'on start' block which can be found in the basic menu. This block is also already present when you start a new project.

```

on start
  set Count to 0
  show number Count
  
```

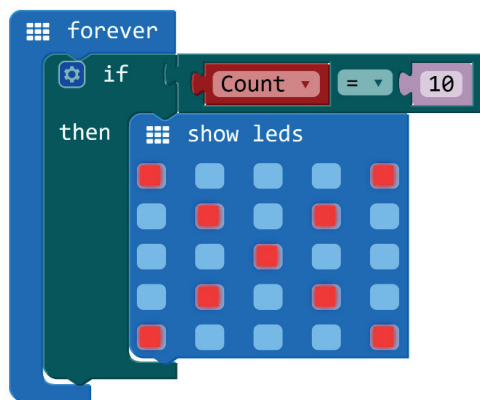
- 2 In the variables menu at the top is the Make a variable button. Click the button and name your variable. We chose 'Count' but you can give it any name you like. This is the variable that will keep track of how many people are inside the event.

Continued on next page >

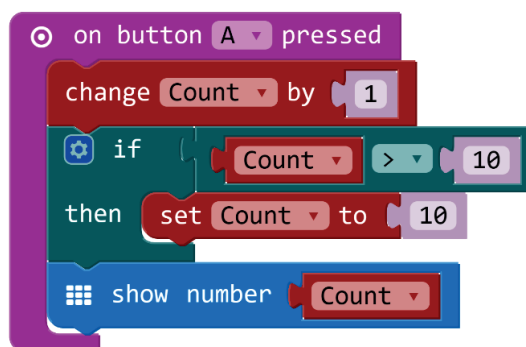
# COUNT ON IT

## SOLUTION – PART 2

- 3 In order to display our starting number, we first need to declare our variable and set it to 0 using the 'set variable to block' from the Variables menu. Remember to change the block from 'item' to 'Count'.
- 4 We can then use the 'show number' block from the Basic menu. In the Variables menu grab the block for our 'Count' variable and drop it into the 'show number' block in place of the number.



- 5 We are going to use a 'forever loop' to set our maximum number of people allowed into the event and also to display a big X on the LED Matrix when our limit is reached. The 'forever loop' can be found in the Basic menu.
- 6 We are going to place an 'if then' loop from the Logic menu into the 'forever loop'. The 'if' statement will monitor the Count variable and when Count reaches the number we have chosen, in this case 10, a 'show leds' block from the Basic menu will display a big X. The Boolean comparison block that checks when Count reaches 10 can also be found in the Logic menu.



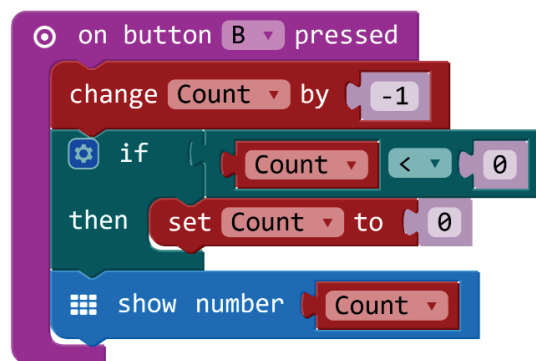
Continued on next page >



# COUNT ON IT

## SOLUTION – PART 3

- 7 We are going to use button A to count people in. Every time Button A is pressed, 1 will be added to the Count variable. The 'on button A pressed' block can be found in the Input menu.
- 8 The 'change variable by' block can be found in the Variables menu and we will change its value by 1 each time the A button is pressed.
- 9 We need another 'if then' to ensure that our button presses don't result in a count greater than 10, after which we have another 'show number' block to ensure that the LED Matrix is updated with each press of the A button.



- 10 As we are going to use all of the same blocks for our button B presses, as we used for the button A presses, we are going to copy the whole block and edit it rather than go through all of the various menus again. Right click on the 'on button A pressed' block in your code and select 'Duplicate' from the menu. The block will be greyed out initially.
- 11 Change the button pressed from A to B and the block and its contents will no longer be greyed out.
- 12 As we are going to be counting down with button B we will need to change the 'change Count by' block to -1 so it deducts 1 from the Count every time the B button is pressed.

Continued on next page >

# COUNT ON IT

SOLUTION – PART 4

- 13 The 'if then' loop in this group of blocks needs to ensure that the count cannot drop below 0. If you press the B key when the Count is at 0 it will reset Count to 0 rather than let it go to -1.



**NOTE:** If we want to change the number of people that are allowed in the venue to another number we would have to edit both the 'forever loop' and the 'on button A pressed' block.

We could simplify our code by doing away with the 'forever' loop and one of the 'if then' loops and instead; add an 'else' statement to the 'if then' loop in the 'on button A pressed' block. Move the 'show leds' block to underneath the 'set count to 10' block and then move the 'show number' block into the 'else' part of the loop. We chose to code it the long way around to aid understanding but it has resulted in extra code (bloat).

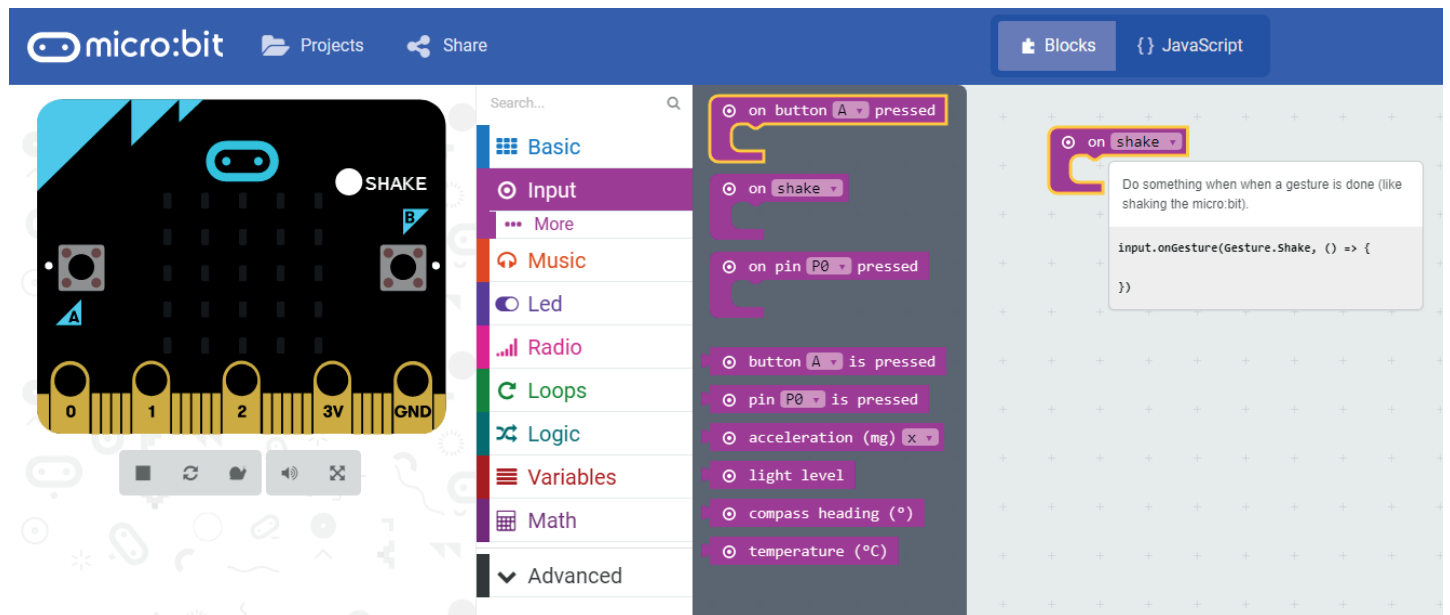
HOW ARE YOU ENJOYING THE CHALLENGES?

Tweet us  @kitronik with the hashtag #kitronikcodechallenge

# ON A ROLL OF THE DICE PART 1



In this challenge, we are going to simulate the roll of a dice. We will learn how to generate a random number, store it in a variable and then display it on the LED matrix. To aid the simulation and make it a little more dice-like, you have to give the micro:bit a shake to make it all happen.



## THE INPUT MENU

In the above diagram, you can see that the 'on shake' block can be found in the "Input" menu of the editor. This block will wait for the micro:bit to be shaken and then run through any blocks that are placed inside it and perform any actions that they dictate.

## VARIABLES

A variable is like a box that you can put things in, you can also add to or subtract from its contents at any time. You can place numbers, names, or text strings into variables that can be used by your program. Variables also have a label or a name that we can use to call them up at any time in our program.

If you look in the variables menu you will see the make a variable button at the top. When you click this, it opens a dialogue box that asks you for the name (label) of your variable. Once it is named you can then write code to determine what is to be put in the box and how it will be used in your program. When choosing a name for your variable, it makes sense to choose a name that describes what the variable is. If your program is going to contain multiple variables it can become quite confusing if the variables aren't suitably named.

Continued on next page >

# ON A ROLL OF THE DICE

 PART 2

## RANDOM NUMBERS

The 'pick random 0 to ?' block can be found in the Math menu. This block not only makes it easy for us to simulate a dice but it also has many possible uses within coding. Everything from Scientific modelling to computer games make use of randomness within their code.

## DISPLAYING THE NUMBER

The 'show number' block in the Basic menu can do more than just display a number that we type into it. We can also use it to display the number that is in our variable by using the variable label instead of a number. The contents of the variable will be displayed on the LED Matrix and will remain there until the micro:bit is shaken again.

## THE CHALLENGE

Using the MakeCode Editor, write code to...

- 1 When the micro:bit is shaken, generate a random number from 1 – 6
- 2 Display the random number on the LED matrix.



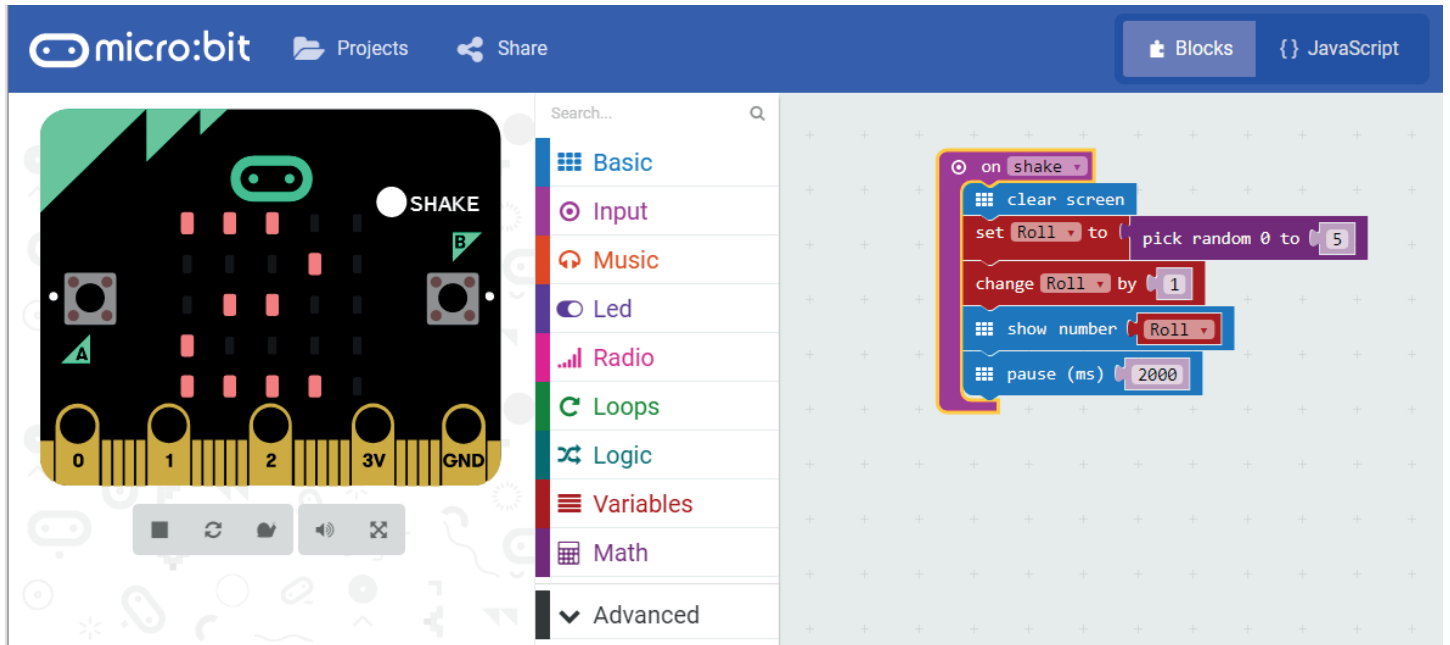
HOW ARE YOU ENJOYING THE CHALLENGES?

Tweet us  @kitronik with the hashtag #kitronikcodechallenge

# ON A ROLL OF THE DICE

**SOLUTION – PART 1**

## THE SOLUTION



You can see in the above example one method of delivering this challenge's objectives. The step by step instructions below will show you how we did it and the 'Other Information' section at the end will cover any further useful information.

- 1 In the variables menu at the top is the 'Make a variable' button. Click the button and name your variable. We chose 'Roll' but you can give it any name you like.
- 2 In the Input menu find the 'on shake' block and drag it into the workspace.
- 3 In the basic menu find the 'clear screen' block and place it in the 'on shake' block. This will clear the last number from the LED matrix with each new shake of the micro:bit.

Continued on next page >

# ON A ROLL OF THE DICE

 SOLUTION – PART 2

- 4 We now need to place a random number into our variable. In the Variables menu, you will find the 'Set variable to' block. By default, it will be set to item. To change this, click on item and choose 'Roll' from the list.
- 5 In the 'Math' menu find the 'pick random 0 to ?' block and add it to the 'set roll to' block in place of the default 0, by dragging and dropping.
- 6 As the 'pick random' block has a start number of zero, we need to change the second number to 5 so that there are 6 possible numbers.
- 7 In the variables folder, find the 'change variable by' block and change the number to 1. To convert our random number from 0-5 into the correct format for a standard dice 1-6, we need to add 1 to our variable.
- 8 In the Basic menu find the 'show number' block.
- 9 In the Variables menu find the 'Roll variable' block, this should be at the top directly below the 'Item variable' block. Drag it over to the 'show number' block and drop it into place.
- 10 As the 'pick random' block has a start number of zero, we need to change the second number to 5 so that there are 6 possible numbers

## OTHER INFORMATION

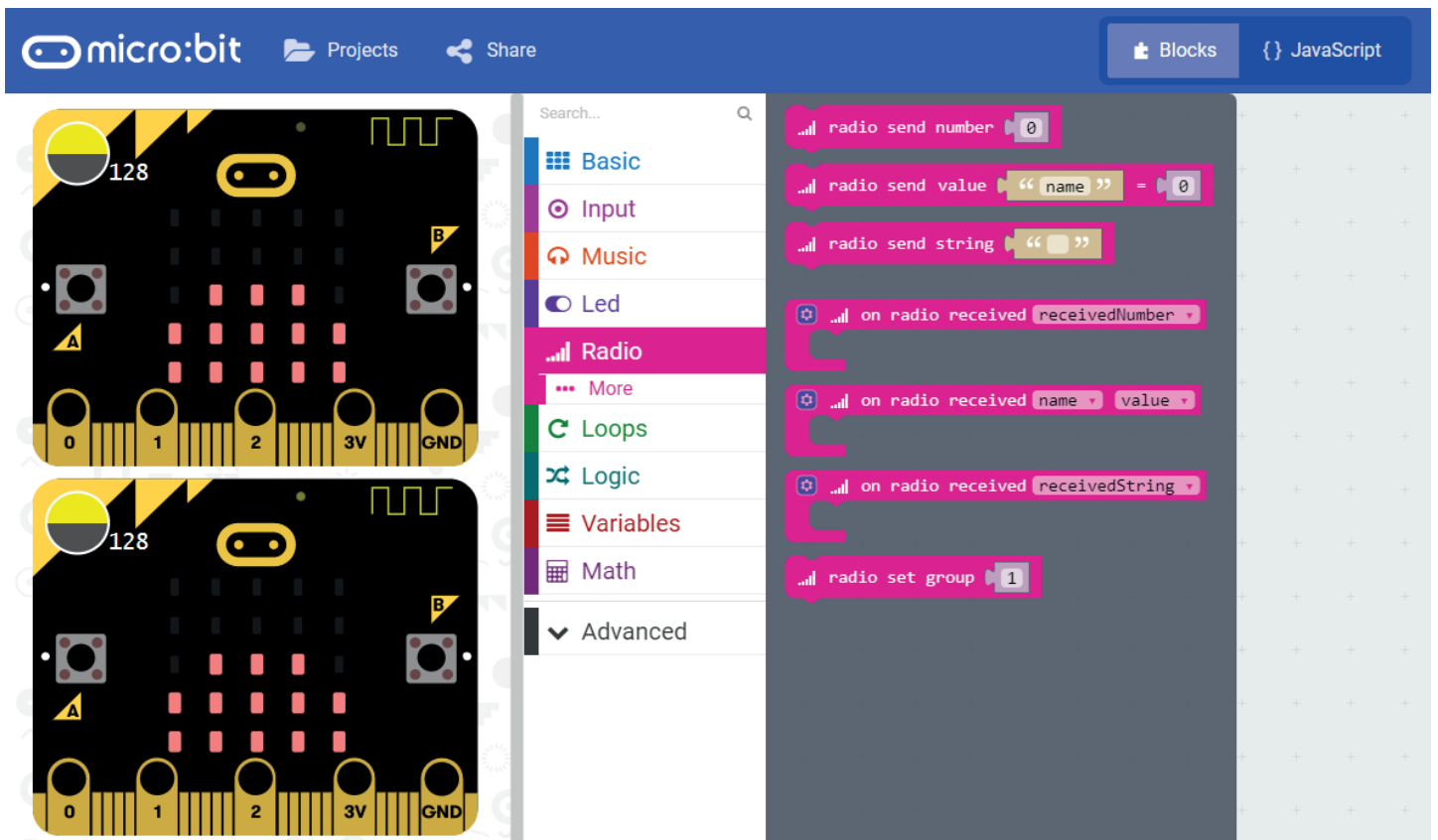
You may have noticed that the SHAKE button on the simulator wasn't there until the 'on shake' block was used. This is a super useful feature of the editor that ensures the simulator doesn't get too cluttered with things you don't need. As you make use of them in your code they will automatically appear on the simulator.

# RADIO TRANSMISSIONS PART 1

**🎯** In this challenge, we will learn how to get two micro:bits to talk to each other using the built in Radio Module. The radio module on the micro:bit is super useful, information gathered by one micro:bit can be transmitted to other micro:bits in the same radio group, which can then perform actions based on the information received.

We are going to measure the light level of a room with a micro:bit. The micro:bit will then transmit that information to a specific radio group. We will have another micro:bit in the same radio group that will pick up that broadcast and then convert the information to a real-time graph displayed on its own LED matrix.

**i NOTE:** This is the only lesson in this guide that requires having more than just a single micro:bit. If you don't have two micro:bits you can still follow along and write the code in the MakeCode Editor.



Continued on next page >

# RADIO TRANSMISSIONS

 PART 2

## VARIABLES

A variable is like a box that you can put things in, you can also add to or subtract from its contents at any time. You can place numbers, names, or text strings into variables that can be used by your program. Variables also have a label or a name that we can use to call them up at any time in our program.

If you look in the variables menu you will see the 'make a variable' button at the top. When you click this, it opens a dialogue box that asks you for the name (label) of your variable. Once it is named you can then write code to determine what is to be put in the box and how it will be used in your program. When choosing a name for your variable, it makes sense to choose a name that describes what the variable is. If your program is going to contain multiple variables it can become quite confusing if the variables aren't suitably named.

## SET RADIO GROUP

When using the radio module on two or more micro:bits, it is important that you set both micro:bits to use the same radio group, one will send to that group and the other will listen out for transmissions to that group. You can choose any number from 0 to 255. This block can be found in the Radio Menu in the editor.

## ON RADIO RECEIVED

This block is placed on the receiving micro:bit and runs part of a program based on the information it receives. This block can be found in the Radio Menu in the editor.

## LOOPING FOREVER

The 'forever' loop block can be found in the Basic menu of the editor. This block will start working as soon as the micro:bit is powered up and will keep executing the code placed inside it until the micro:bit is powered off. This is the perfect loop type for programs that need to perform the same instructions over and over again, as is often the case for programs that are monitoring something.

## RADIO MODULE

The key point to note about the micro:bit's radio module is that it allows you to code one micro:bit to either measure something with its own onboard sensors or process input from its edge connector. This transmits this information to another micro:bit that has been coded to listen out for these radio transmissions. You can code the second micro:bit to react to this incoming data and to perform a task that you specify. The two micro:bits could be in another room.

## RADIO SEND NUMBER

You can use this block to broadcast a number/sensor reading/variable to other micro:bits that are set to the same radio group. This block can be found in the Radio Menu in the editor.

## PLOT BAR GRAPH

The plot bar graph block can be found in the 'LED' menu and will display a graph based on the parameters we set. In this case, we want it to display the light level that it receives from another micro:bit in another room.

## THE CHALLENGE

Using the MakeCode Editor, write code to...

- 1 With one micro:bit, measure the light level of a room and store it in a variable.
- 2 Broadcast the variable to a radio group.
- 3 On a second micro:bit set to the same radio group, receive the light data and plot a bar graph of it on its LED Matrix.

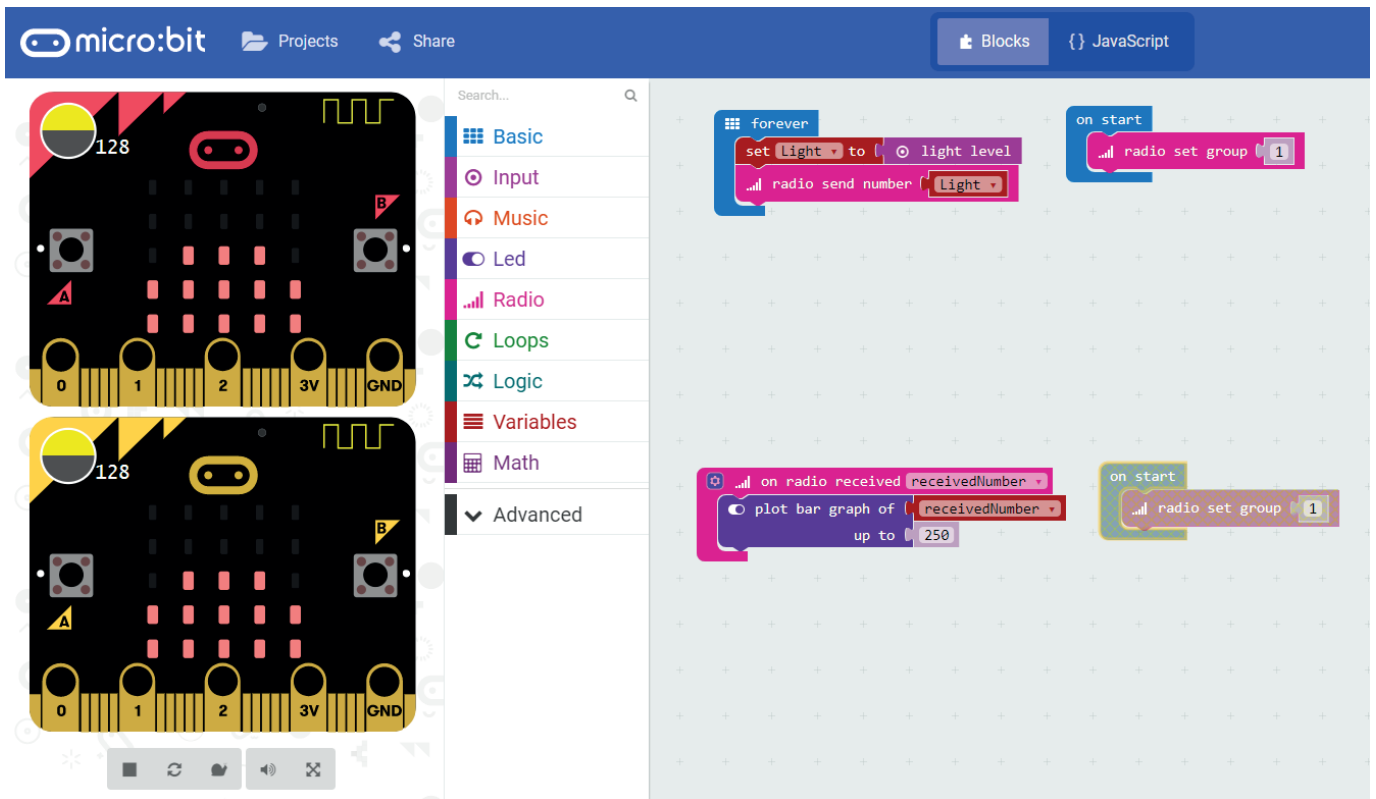


# RADIO TRANSMISSIONS SOLUTION – PART 1

## THE SOLUTION

Before you start with the step by step guide, take a moment to study the code below. The brief might lead you to believe that this challenge might be difficult to code but that couldn't be further from the truth.

The top two blocks of code are for the transmitting micro:bit and the bottom two are for the receiving micro:bit. So, there are really two separate programs shown. We put them in the same project for illustration purposes.



### Coding the transmitting micro:bit

- 1 In the variables menu at the top is the 'Make a variable' button. Click the button and name your variable. We chose 'Light' but you can give it any name you like.

Continued on next page >

# RADIO TRANSMISSIONS SOLUTION – PART 2

- 2 When you start a new project, the project opens with two blocks already in the editing space; the 'on start' block and a 'forever loop', which is exactly what we will need.
- 3 In the 'on start' block we are going to place a 'radio set group' block and set it to a number between 0 and 255. We chose 1. The block can be found in the Radio menu. This will tell our transmitting micro:bit which radio group it should broadcast to, when told to.
- 4 In the 'forever' block we need to write the light level, as measured by the micro:bit, into our variable Light. We can do this by using the 'set variable to' block, changing it to our previously created Light variable and attaching a 'light level' block from the Input menu. This will measure the ambient light levels in real time and write the result to the Light variable.
- 5 We now need to use the 'radio send number' block found in the Radio menu to broadcast the number held in our Light variable.

## Coding the receiving micro:bit

- 6 In the 'on start' block we are going to place a 'radio set group' block and set it to a number between 0 and 255. We chose 1. The block can be found in the Radio menu. This will tell our receiving micro:bit which radio group it should listen to.
- 7 We need the 'on radio received' block from the radio menu. This defaults to 'received Number', which is what we want. It also automatically creates a 'received Number variable' that contains the number broadcast by our transmitting micro:bit.
- 8 In the 'on radio received block' we need to place 'plot bar graph of' block from the Led Menu and replace the default value with the received Number variable.
- 9 For the 'up to' value of the 'plot bar graph of' we chose a value of 250. This suited our light levels and ensured that when the ambient light was at its maximum the bar graph displayed on the LED Matrix was fairly full. You may need to experiment with this number and tweak it a little.