Using Different Inputs with BBC micro:bit

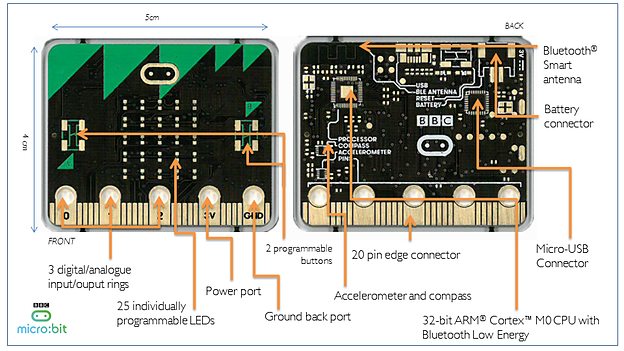
Physical Computing

# Making a Dice

The purpose of this activity is to introduce you to the accelerometer in the BBC micro:bit which can be used to detect movement. In this activity we will make a random number generator or 6-sided dice with the BBC micro:bit. This activity and a copy of the finished micro:bit program will be available on the CS4S website after the workshop.

## BBC micro:bit

The diagram below shows the key components of the BBC micro:bit. During this tutorial we will be using the *accelerometer* as our input and the *25 individually programmable LEDs* as the output.

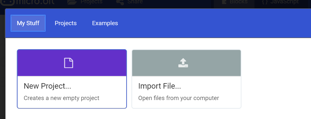




## Creating the code

To get started open a new window in ***Google Chrome*** and go to [https://makecode.microbit.org/#](https://makecode.microbit.org/)

If you already have a project open, click on Projects in the top menu and then click on New Project

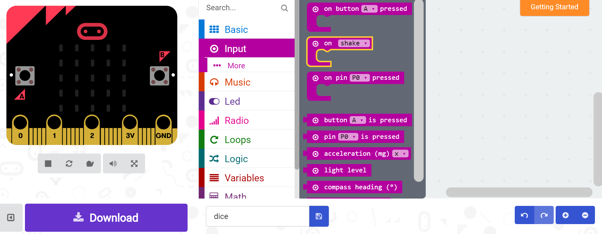




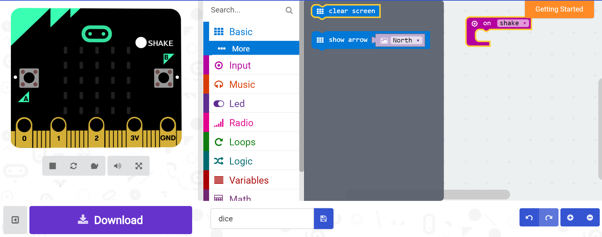
Name your project ‘Dice’.



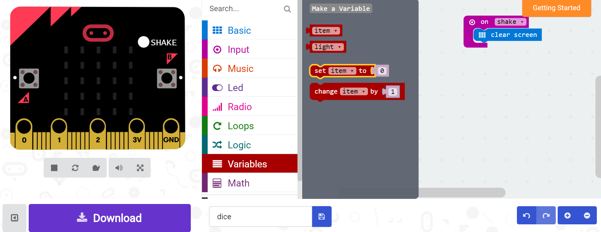
Click on the Input button in the *toolbox* menu and drag an on shake block into the *workspace*.



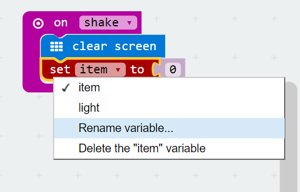
Once the on shake block is in place, click the Basic button in the *toolbox* and then open the More menu. Drag a clear screen block inside the on shake block.



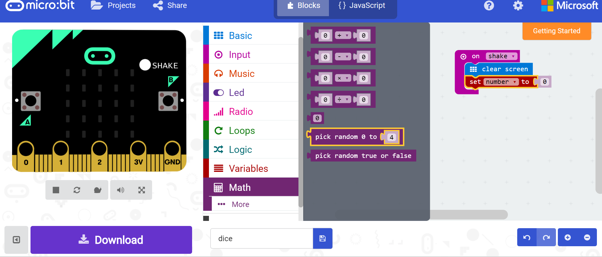
Next drag a set item to block under the clear screen block.



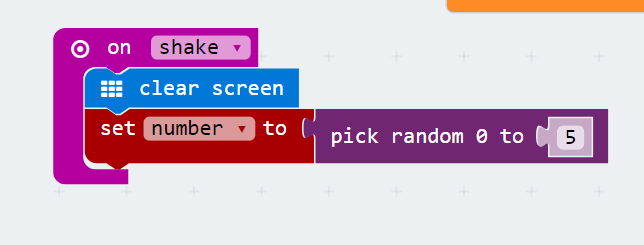
Use the drop down arrow next to the word ‘item’ to rename the variable ‘number’.



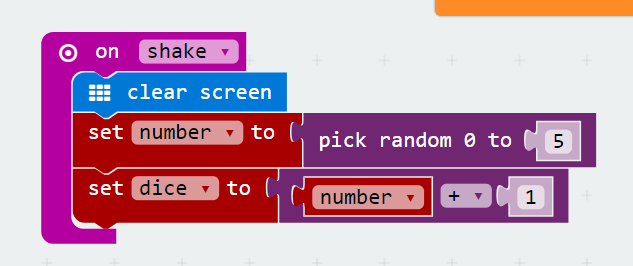
Now we need to set the value of the ‘number’ variable. We can do this using a pick random block from the Math menu in the *toolbox*.



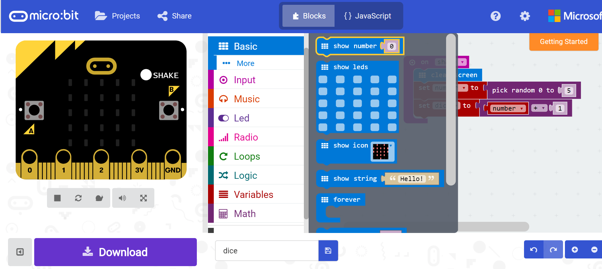
Since our dice will have 6 sides, and the pick random block starts at 0, we need to input ‘5’ as the maximum value for the ‘number’ variable.



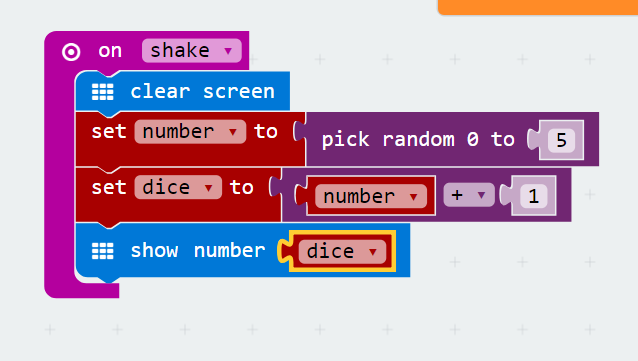
However, since we want out dice to display the numbers 1 to 6, not 0 to 5, we need to create a second variable. Let’s call this variable ‘dice’. Use blocks from the Variables and Math menus of the toolbox to create the following line of block code.



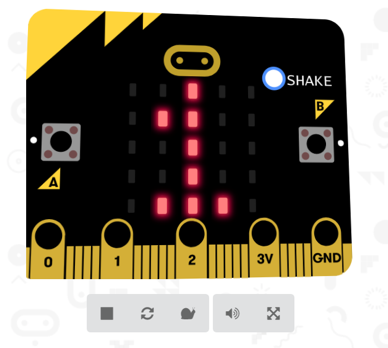
Finally, we need to drag a show number block from the Basic menu below the line of code created above.



Instead of inputting a number manually, we want the LED display to show the randomly generated number. To do this we need to insert the dice variable as shown below.



Your code is now complete. To test it you can use the *emulator* by clicking the shake button.



Now we will test the code on the micro:bit itself. To do this we need to download a .hex file by clicking on the big purple download button at the bottom left of the screen and transfer it to our micro:bit just like we did in the interactive badge tutorial.

You can test this program by shaking the micro:bit. Each time you shake the micro:bit, a random number (between 1 and 6) should appear on the LED display. You may also want to try connecting up the battery pack to the micro:bit and use it as a dice while it is not connected to your computer.

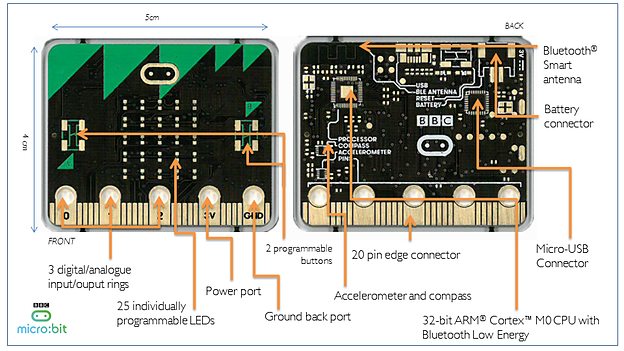
# Using External Outputs

The purpose of this activity is to introduce you to the pin edge connectors in the BBC micro:bit which can be used to connect external devices and create circuits. In this activity we will make a simple music player using our BBC micro:bit. You may work individually or in pairs during this activity. This activity and a copy of the finished micro:bit program will be available on the CS4S website after the workshop.

You will need 2 alligator clips and a pair of headphones or a piezo buzzer.

## BBC micro:bit

The diagram below shows the key components of the BBC micro:bit. During this tutorial we will be using the *2 programmable buttons* as our input and the *20 pin edge connectors* as the output.





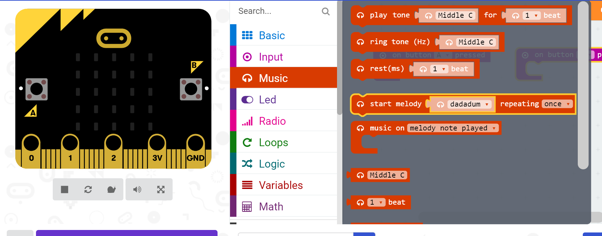
## Creating the code

Create a new project in makecode and name it Music Player.

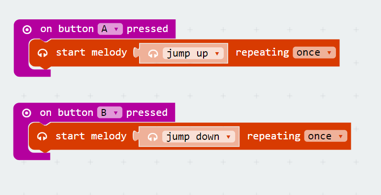
Drag an On Button A and an On Button B block into the workspace.



Next we need to drag a start melody block from the Music menu in the *toolbox* inside each of the on button blocks in the *workspace*.



Choose a melody that you would like to have play when each button is pressed. In the example below we have used jump up and jump down.



Trial this in the *emulator* and then load it onto your microbit.

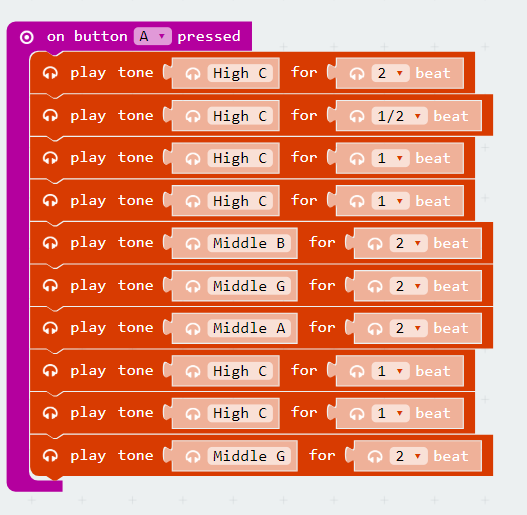
The micro:bit does not have inbuilt speakers so we will have to connect our earphones or a piezo buzzer to the pin edge connectors using alligator clips. You can find a tutorial showing you how to do this at <https://www.microbit.co.uk/blocks/lessons/hack-your-headphones/activity>

We have also included two images below that you can use a guide when connecting the earphones or buzzers. These images show a micro:bit connected to a pair of earphones (on the left) and a piezo buzzer (on the right).



Note that, in the image above, the alligator wire that is connected to the GND port of the micro:bit is connected to the end of the earphones’ plug, which you should do as well.

Now we have had a go at connecting our earphones or piezo buzzer we are going to try to get a little creative and use the play tone blocks in the Music menu to create a tune. Copy the code sequence below, load it onto your micro:bit and see if you can guess the tune.



You may have noticed that it doesn’t sound quite right. That is because the original tune used 1.5 beats, which cannot be programmed in makecode.

# Using the Light and Temperature Sensors

The purpose of this activity is to introduce you to the light and temperature sensors in the BBC micro:bit. In this activity we will first create a simple light level detector and then a simple thermometer using our BBC micro:bit. This activity and a copy of the finished micro:bit program will be available on the CS4S website after the workshop.

## BBC micro:bit

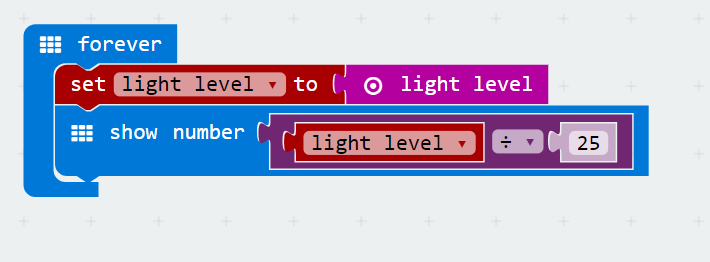
During this tutorial we will be using the *25 programmable LEDs* to detect ambient light. The level of ambient light is assigned a value from 0 to 255. Detecting light using LEDs is possible because, if the current is inverted on an LED it becomes sensitive to light. The microbit is particularly sensitive to red light because LEDs are most sensitive to the same wavelength of light they emit. A more detailed explanation may be found here <https://lancaster-university.github.io/microbit-docs/extras/light-sensing/>

We will also use the micro:bit to detect ambient temperature. However, the micro:bit does not have a dedicated temperature sensor. The temperature is calculated from the temperature of various chips on the circuit board. Because the micro:bit does not get very warm when it runs this will be a fairly accurate estimate of the ambient temperature. A more detailed explanation may be found here <https://www.microbit.co.uk/functions/temperature>

## Creating the code

Create a new project in makecode and name it Light Sensor.

Use blocks from the Basic, Variables, Input and Math menus in the *toolbox* to create the below code.



Load the .hex file onto your micro:bit and see what happens if you shine light directly at the LEDs of the microbit. Why do you think we divided the light level by 25 in the above block code? Hint: we would have divided it by 25.5 but unfortunately the micro:bit block code does not support decimal places.

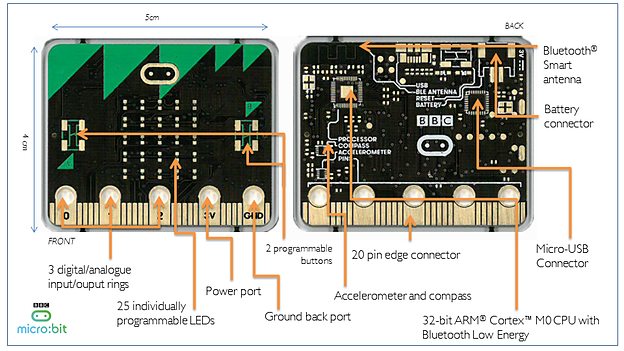
*Create a new project and name it Temperature Sensor.*

*Try to create your own code for displaying the current ambient temperature on the LED screen of the micro:bit*.

# Creating a Compass

## BBC micro:bit

The diagram below shows the key components of the BBC micro:bit. During this tutorial we will be using the *compass*. For more information about how the compass works see <https://www.microbit.co.uk/functions/compass-heading>



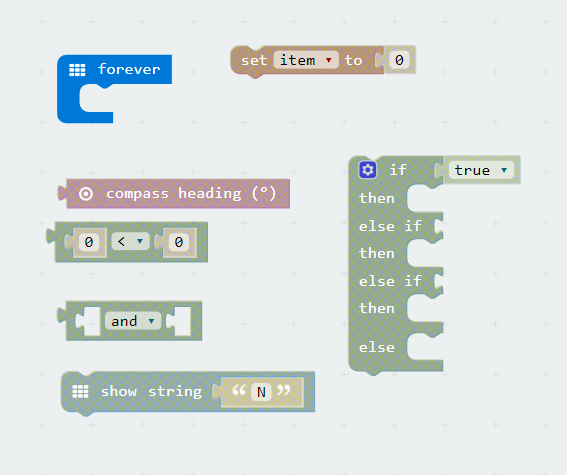


## Creating the Code

Using block code try to create a compass that displays the following directions on screen:

* North
* South
* East
* West

You will need to use the following blocks (see next page). Hint: you may need to use some of them more than once



The if, then, else if, then, else if, then, else block can be created by selecting an if then block from the logic menu in the *toolbox* and using the small cog icon to reveal further options. Ask one of the demonstrators for help if you are unsure.

When you are happy with your code i.e. you have tested it in the *emulator*, try loading it onto you micro:bit.

The first time that you use the compass heading block in a program run on the micro:bit, you may be asked to calibrate the compass on the device. If the LED displays shows the message “DRAW A CIRCLE” when you turn on the micro:bit, this means that you will have to calibrate the compass. To calibrate the compass, you have to tilt the micro:bit around so that a circle is drawn on the display. For more help with calibrating the compass, please see this [support article](https://support.microbit.org/support/solutions/articles/19000008874-calibrating-the-micro-bit-compass-what-does-it-mean-when-the-micro-bit-says-draw-a-circle-or-tilt).

# Acknowledgements

The following images were used in this tutorial:

* BBC micro:bit diagram: <https://codeclubprojects.org/en-GB/resources/microbit-intro/>

Tutorial tasks inspired by:

<https://microbit.org/assets/posts/2018-01-19-train_the_trainer/topic_3.pdf>