Unit 4.2: We are makers

Coding for micro:bit

Software: Microsoft MakeCode for the micro:bit (online)
Hardware: Laptop/desktop computers, BBC micro:bits (with USB cables and battery packs)

Overview

In this unit, pupils write and test their own **micro:bit** project, after analysing and modifying others. In:

- **Session 1** they explore the **MakeCode** environment and learn about the BBC micro:bit
- **Session 2** they work out how a match-scoring program has been written
- **Session 3** they modify a rock-paper-scissors game to make a sorting hat game
- **Session 4** they modify their sorting hat game to make a dice game
- **Session 5** they plan their own micro:bit project
- **Session 6** they write and test their own micro:bit project.

Alternatives

The MakeCode editor runs in a web browser, and so can be used with any computer. If using iPads, the micro:bit program must be transferred to the micro:bit via **Bluetooth** (see *Useful links* page 22). A micro:bit can be programmed using other programming languages, including Scratch. With Scratch, the micro:bit acts as input and output to a Scratch program running on the computer. Other hardware platforms are available, including Circuit Playground Express, which can be programmed using MakeCode, and Crumble, which has its own programming environment.

Knowledge, skills and concepts

In this unit, pupils will learn:

- about the input – process – output model of computation
- about the inputs and outputs available on a BBC **micro:bit**
- to program using the **MakeCode** block-based environment
- to test and debug programs they write, using an on-screen **simulator** and the micro:bit
- how to convert and transfer a program written on screen to the micro:bit.

Progression
In Key Stage 2:

- Programming here builds on earlier programming work in Scratch, including **Unit 3.1: We are programmers, Unit 3.2: We are bug fixers** and **Unit 4.1: We are software developers**.
- Pupils will continue to develop programming in **Unit 5.1: We are game developers** and **Unit 6.6: We are AI developers**. They return to the micro:bit and MakeCode for programming in **Unit 6.1: We are toy makers**.

Assessment – by the end of the unit:

All pupils can:

- identify some of the inputs and outputs of the **micro:bit**
- predict and test what a **MakeCode** program does
- transfer a MakeCode program to the micro:bit
- implement an **algorithm** as a MakeCode program
- design a program for the micro:bit.

Most pupils can:

- identify all of the inputs and outputs for the micro:bit
- explain what a MakeCode program does
- debug a MakeCode program
- design their own algorithm for the micro:bit
- implement their own algorithm as a program in MakeCode.

Some pupils can:

- suggest applications for the inputs and outputs on the micro:bit
- make changes to a MakeCode program to improve it
- design an algorithm and user interface for a micro:bit application
- implement their own design as a working program on the micro:bit.

Background information

- The BBC **micro:bit** is a small, single circuit board, programmable computer. It was launched by the BBC in 2016 and distributed to every Year 7 pupil in the UK. The hardware includes a system-on chip programmable microcontroller, two input buttons, an **accelerometer**, a magnetometer and a 25-**LED** display. The micro:bit is equipped with a USB connector, a **Bluetooth** radio system and an edge connector with programmable input/output pins, which can be used to connect external switches, sensors, speakers, LEDs or motors.
- A range of programming languages can be used to program the micro:bit, including Microsoft's **MakeCode** and MicroPython. Here, we use the blocks editor in MakeCode, which is like the Scratch editor that pupils will have used in previous units. MakeCode also has a text mode, allowing programs written in blocks to be converted to **JavaScript** and, in some circumstances, JavaScript programs to be converted to blocks.
- MakeCode programs (**source code**) are downloaded as a complete **runtime** environment (.hex files, **object code**) to the computer, and then transferred by USB or Bluetooth to the micro:bit.

Key vocabulary

Accelerometer: hardware component providing data on changes in motion, typically in three directions

Algorithm: a sequence of precise instructions or steps (sometimes a set of rules) to achieve an objective

Bluetooth: wireless digital communication protocol using low energy signals over short distances

If/else if/else: programming selection construct which indicates what code should be run depending on which one of multiple conditions are satisfied

JavaScript: text-based programming language, commonly used to power interactive web pages

LED: light emitting diode, an electronic component that lights up when current flows in one direction

MakeCode: block- and text-based editor from Microsoft, supporting a variety of hardware platforms including the micro:bit

micro:bit: simple, single board programmable computer with integrated input, output and network capabilities

Object code: a version of the program converted (compiled) into the detailed instructions to be followed by the computer's processor

Runtime: the complete software environment (operating system, drivers, interpreter) needed for a program to run on particular hardware

Simulator: software allowing one computer system to behave as another; in this case, the MakeCode editor includes an on-screen simulator of a micro:bit so that programs can be tested

Source code: the program as written, in a language that can be understood by both the programmer and the computer

Variable: lets computer programs store, retrieve or change simple data – typically thought of as a particular bit of the computer's memory that holds a specific bit of data

Differentiation

Pupils can explore how the **micro:bit** can be connected to other devices using the edge connector, how games can be implemented on the micro:bit using its sprite blocks, or how messages can be passed between micro:bits using **Bluetooth**. Some pupils might like to explore the **JavaScript** version of their micro:bit programs, seeing how this mirrors their block code.

If pupils need more support, have them follow step-by-step guidance. Also consider replacing the independent, open-ended projects in Sessions 5 and 6 with a single project, e.g. a whack-a-mole game, a virtual pet or a fitness tracker.

The **MakeCode** editor has support for languages other than English if pupils learning English as an additional language wish to program in their first language.

Cross-curricular opportunities

Maths: Pupils programme a dice game.

English: Pupils create a sorting hat game, based on the *Harry Potter* book series.

Design and technology: If pupils make use of additional hardware components for input or output, then strong links with design and technology can be built.

Other: There is an opportunity to link the open-ended project to other curriculum areas.

Preparation for teaching the unit

Things to do

- Check you have access to MakeCode (see *Useful links*) and sufficient micro:bits for your class. You will also need USB cables and the micro:bit battery packs. If you don't have access to micro:bits you can use the micro:bit simulator in MakeCode instead.
- Read pages 20–21 to get an overview of the unit.
- Read the steps in the unit sessions (pages 23–28) and look at the associated online resources, printing out the worksheets as required.
- Watch the CPD videos (see *Additional resources*).
- Work through the unit yourself, so you know what is expected of pupils.
- If pupils are going to share their work or save their work online, make sure they have accounts set up, that necessary permissions have been obtained and that these are integrated with the iPads they are using.

Resources needed

- **Software:** MakeCode (online)
- **Hardware:** Laptop/desktop computers and micro:bits (with USB cables and battery packs)

Online resources provided

Session resources

- Worksheet 4.2a: Step-by-step instructions for transferring code to the micro:bit
- Worksheet 4.2b: Step-by-step instructions for modifying the sorting hat game
- Worksheet 4.2c: Step-by-step instructions for creating the dice game
- Worksheet 4.2d: End-of-unit quiz
- Worksheet 4.2e: Pupil self-assessment
- Teaching slides 4.2a–4.2f
- Walkthrough videos 4.2a–4.2g
- Example micro:bit programs:
  - Session 2 scoreboard: **makecode.microbit.org/_hg45EbdiDUvc**
  - Session 3 rock-paper-scissors: **makecode.microbit.org/_C0ja8WAT0i6r**
  - Session 3 sorting hat: **makecode.microbit.org/_eaoVhwHUR5z5**
- Interactive end-of-unit quiz 4.2

Additional resources

- CPD video: Controlling hardware
- CPD video: Understanding the development process

Online safety

- The MakeCode editor is online, so pupils will need Internet access for this session. Make sure that the appropriate filters and monitors are in place.
- Pupils can publish their programs to the MakeCode website. If they are to do so, make sure that appropriate parental permission is obtained.
- Pupils might explore the projects uploaded by others to the MakeCode website. Remind them that they must let you know if they come across any inappropriate content when looking at these, even if this is very unlikely.

Collaboration

Give careful thought to how pupils should be organised for this unit. Confident pupils should be able to cover much of this independently, but most will benefit from having a partner to work with. Consider pairing less confident pupils with more confident peers, while reminding them that the aim is for all pupils to learn how to program the micro:bit. Small group work would also be possible, especially if there are limited numbers of micro:bits available.

Useful links

Software and tools

- MakeCode: **www.makecode.microbit.org**
- More about the micro:bit: **www.microbit.org**

Online tutorials

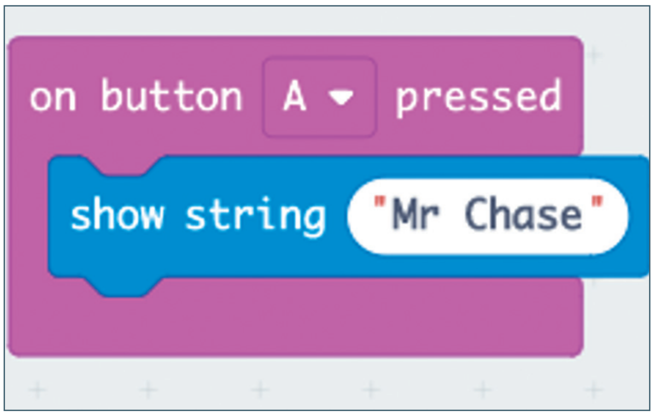
- MakeCode: **makecode.microbit.org/tutorials/getting-started**
- micro:bit: **www.microbit.org/guide/quick**
- micro:bit for iPads: **www.microbit.org/guide/ble-ios**

Information and ideas

- **www.makecode.microbit.org/courses/csintro**
- **www.microbit.org/ideas**
- **www.microbit.org/en/2017-03-07-javascript-block-resources/#lessons_a**
- **www.mb4ps.co.uk/resources**
- **www.microbit.org/teach/iet**
- Rock-paper-scissors strategies: **www.arstechnica.com/science/2014/05/win-at-rock-paper-scissors-by-knowing-thy-opponent**

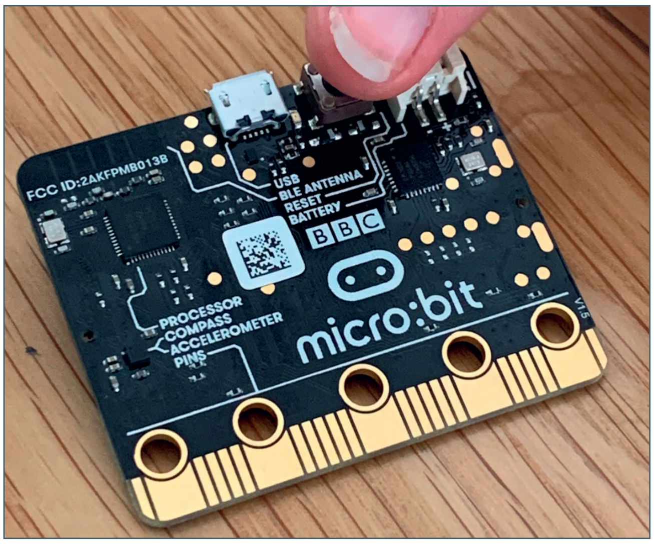
Unit outcomes

Below are some examples of the outcomes you could expect from this unit.

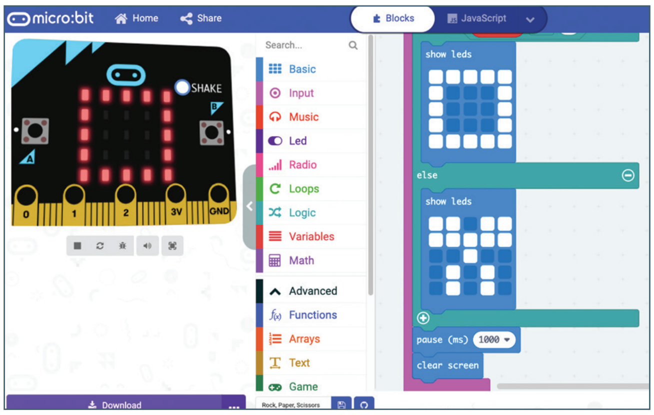


```
on button A pressed
    show string "Mr Chase"
```

Session 1: A simple program to make text scroll across the display

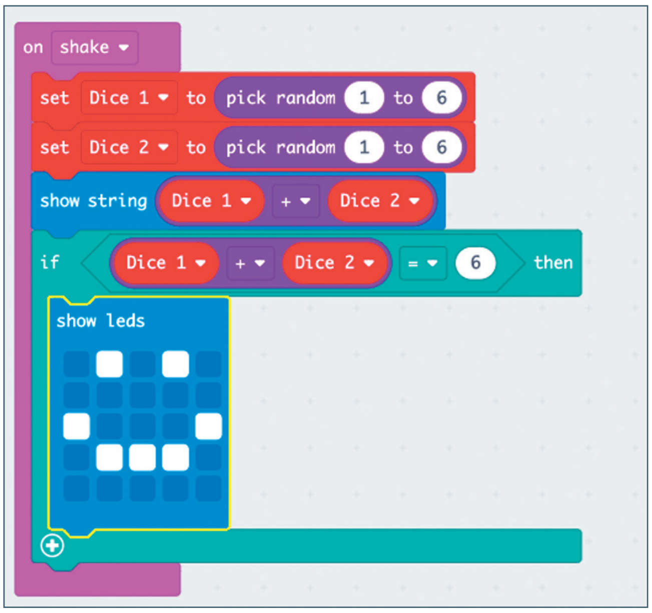


Session 2: Transferring code onto a micro:bit



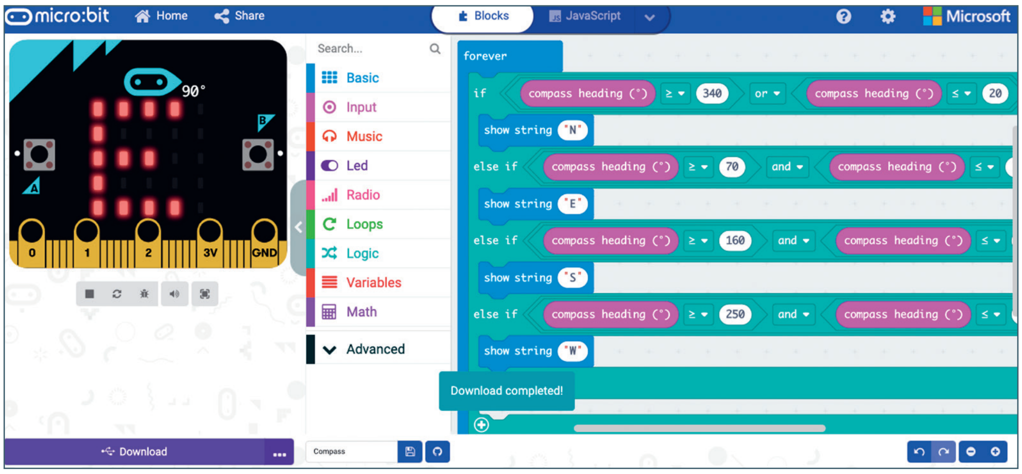
```
show leds
else
show leds
pause (ms) 1000
clear screen
```

Session 3: Exploring the rock-paper-scissors game



```
on shake
    set Dice 1 to pick random 1 to 6
    set Dice 2 to pick random 1 to 6
    show string Dice 1 + Dice 2
    if Dice 1 + Dice 2 = 6 then
        show leds
```

Session 4: Creating a dice rolling game



```
forever
    if compass heading (°) ≥ 340 or compass heading (°) ≤ 20
        show string "N"
    else if compass heading (°) ≥ 70 and compass heading (°) ≤
        show string "E"
    else if compass heading (°) ≥ 160 and compass heading (°) ≤
        show string "S"
    else if compass heading (°) ≥ 250 and compass heading (°) ≤
        show string "W"
```

Download completed!

Sessions 5 and 6: Planning and creating a micro:bit project (example here is a compass)