

First lessons with MakeCode and the micro:bit

**Lessons:** 6

**Programming languages:** MakeCode

**Target age:** 7-11 yrs

**Subjects & topics:**

* Computational thinking: Algorithms
* Computer systems: Input/output, Sensors
* Programming: Iteration, Selection, Variables, Randomisation

# Unit of work summary

A sequence of six lessons that provide a pathway through six projects, ideal for getting started with the micro:bit. Students will develop their use of some core computing concepts through coding and making practical projects.

Projects include a name badge, animation, using inputs and outputs to make an emotion badge, using sensors to make step counters and nightlights, and finally combining skills from previous lessons to make a rock, paper, scissors game.

## Overall key learning

* Gain practical skills for creating, testing, and **transferring code to micro:bits**.
* Understand that computers need instructions in a sequence, also known as **algorithms,** and that these are written as programs in **code**, a language the computer can understand.
* Learn how **sequences and loops** can be used to make animations and control programs.
* Gain practical experience and understanding of **inputs, outputs,** and **variables** in real-world contexts.
* Learn how **logic (‘if…then’ instructions)** and **sensors** combine to make a simple control system.
* Combine skills and knowledge gained through the previous lessons to **create computer simulations** of real-world games of chance.
* **Evaluate** what you have made.

## Additional skills

Debugging, evaluation, modifying, prototyping

## Lesson 1: Name badge

Students create their first programs and transfer them to their micro:bits.

**Key learning:**

* Understand the micro:bit is a tiny computer which needs instructions in code to make it work.
* Understand that sets of instructions for computers in a sequence are also called algorithms or programs.
* Use the MakeCode editor to create instructions in code that the micro:bit can understand and then transfer them to the micro:bit.
* Know the micro:bit has an LED display output which it can use to show words (as well as numbers and pictures).

**Subjects & topics:**

* Computational thinking: Algorithms
* Computer systems: Input/output
* Programming: Iteration

## Lesson 2: Beating heart

Create a simple animation to learn about sequence and simple loops.

**Key learning:**

* Understand that sequence and timing is important when making an animation.
* Understand that animations create an illusion of movement by showing a sequence of still images.
* Code the micro:bit to show simple animations on its LED display output.
* Use loops to make animations run longer using fewer instructions.

**Subjects & topics:**

* Computational thinking: Algorithms
* Programming: Iteration

## Lesson 3: Emotion badge

Start learning about inputs and outputs using buttons and icons on the display.

**Key learning:**

* Code the micro:bit to make different outputs happen depending on different inputs. (This is a very simple kind of selection. We look at selection in more detail in lesson 5, Nightlight.)
* Understand that inputs and outputs involve the flow of data in and out of computers.
* Apply this knowledge using the micro:bit’s button inputs and display output.

**Subjects & topics:**

* Computer systems: Input/output
* Programming: Selection

## Lesson 4: Step counter

Introduce variables to track your step count and begin to use the accelerometer input.

**Key learning:**

* Understand how sensor inputs from the accelerometer can be used to detect movement, such as when a step is taken.
* Understand that variables are used to keep track of the current step count.
* Understand that the order of instructions is important: display the variable’s value after updating it, not before.
* Apply this learning to build a practical, real-world project.

**Subjects & topics:**

* Programming: Variables
* Computer systems: Input/output, Sensors

## Lesson 5: Nightlight

Make an automatic nightlight and discover how logic, conditionals and inputs and outputs combine to make a simple control system.

**Key learning:**

* Understand how inputs, outputs, and computer code work together to make control systems.
* Understand how logic (conditional ‘if… then… else’ instructions) is used to make different outputs happen depending on changes in data from a sensor.
* Use ‘forever’ infinite loops to keep control systems responding to changes in the environment.
* Practise testing and improving a project to make the nightlight work better in specific local lighting conditions.

**Subjects & topics:**

* Programming: Selection
* Computer systems: Input/output, Sensors

## Lesson 6: Rock, paper, scissors

Combine skills from the previous lessons to turn your micro:bit into an electronic simulation of a popular game of chance.

**Key learning:**

* Use the accelerometer via the ‘on shake’ block to start the code running.
* Code the creation of random numbers in a fixed range.
* Use variables so they can be tested using logic.
* Make use of more complex logical ‘if… then… else if…’ conditional instructions.
* Apply these concepts to make a computer simulation of a real-world game.
* Evaluate the fairness of computer simulations.

**Subjects & topics:**

* Programming: Selection, Randomisation
* Computer systems: Input/output, Sensors

# Curriculum links

## England National Curriculum

#### KS2 Computing

Curriculum aims:

* [Pupils] can understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation.
* can analyse problems in computational terms and have repeated practical experience of writing computer programs in order to solve such problems.

Subject content

1. Design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts
2. Use sequence, selection, and repetition in programs; work with variables and various forms of input and output
3. use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs

[Read the full KS2 computing curriculum.](https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/239033/PRIMARY_national_curriculum_-_Computing.pdf)

## Scotland Curriculum for Excellence

#### Technologies: Computer Science, Second level

1. I can understand the operation of a process and its outcome (TCH 2-13a).
2. I can explain core programming language concepts in appropriate technical language (TCH 2-14a).
3. I can create, develop and evaluate computing solutions in response to a design challenge (TCH 2-15a).

[Read the full Curriculum for Excellence: technologies](https://education.gov.scot/Documents/Technologies-es-os.pdf).

## Northern Ireland Curriculum - Primary

#### Prerequisite Skills in Using ICT across the curriculum

1. Explore - investigate, make predictions and solve problems through interaction with digital tools.
2. Evaluate - talk about, review and make improvements to work, reflecting on the process and outcome.

[Read the full Northern Ireland curriculum - primary](https://ccea.org.uk/downloads/docs/ccea-asset/Curriculum/The%20Northern%20Ireland%20Curriculum%20-%20Primary.pdf)

#### CCEA Primary Using ICT Desirable Features

Computational Thinking and Coding, Level 4

Programmable devices (including micro:bit)

Pupils should:

1. look at and talk about examples of coding projects, including the use of motion, looks, lights or sounds, sensors, control and events such as ‘if…then’ and ‘loop until’ (or equivalent) that make the code more efficient.
2. recognise that these projects are composed of different components and break the task into smaller manageable tasks (decomposition).
3. use a range of commands to create a project including triggering commands such as ‘if...then’ and ‘loop until’ to facilitate a more efficient method of interaction.
4. test and debug at regular intervals and collaborate with others to solve problems as they arise.

[Read the CCEA's Using ICT Desirable Features](https://ccea.org.uk/learning-resources/using-ict-desirable-features)

## Curriculum for Wales

#### Science and technology

**Computation is the foundation for our digital world**

**Progression step 1**

I can experiment with and identify uses of a range of computing technology in the world around me.

**Progression step 2**

1. I can safely use a range of tools, materials and equipment to construct for a variety of reasons.
2. I can create simple algorithms and am beginning to explain errors.
3. I can follow algorithms to determine their purpose and predict outcomes.
4. I can follow instructions to build and control a physical device.

**Progression step 3**

1. I can use conditional statements to add control and decision-making to algorithms.
2. I can identify repeating patterns and use loops to make my algorithms more concise.
3. I can explain and debug algorithms.
4. I can use sensors and actuators in systems that gather and process data about the systems’ environment.

[Read the full science and technology curriculum](https://hwb.gov.wales/curriculum-for-wales/science-and-technology/descriptions-of-learning/)

## USA CSTA Standards

**Grades 3-5**

1. IB – AP – 09 Create programs that use variables to store and modify data.
2. IB-AP-10 Create programs that include sequences, events, loops, and conditionals.
3. IB-AP-14 Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.
4. IB-AP-15 Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended.

[Read the CSTA Standards in full](https://csteachers.org/k12standards/).

This content is published under a [Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)](https://creativecommons.org/licenses/by-sa/4.0/) licence.