micro:bit Physical Computing Fundamentals
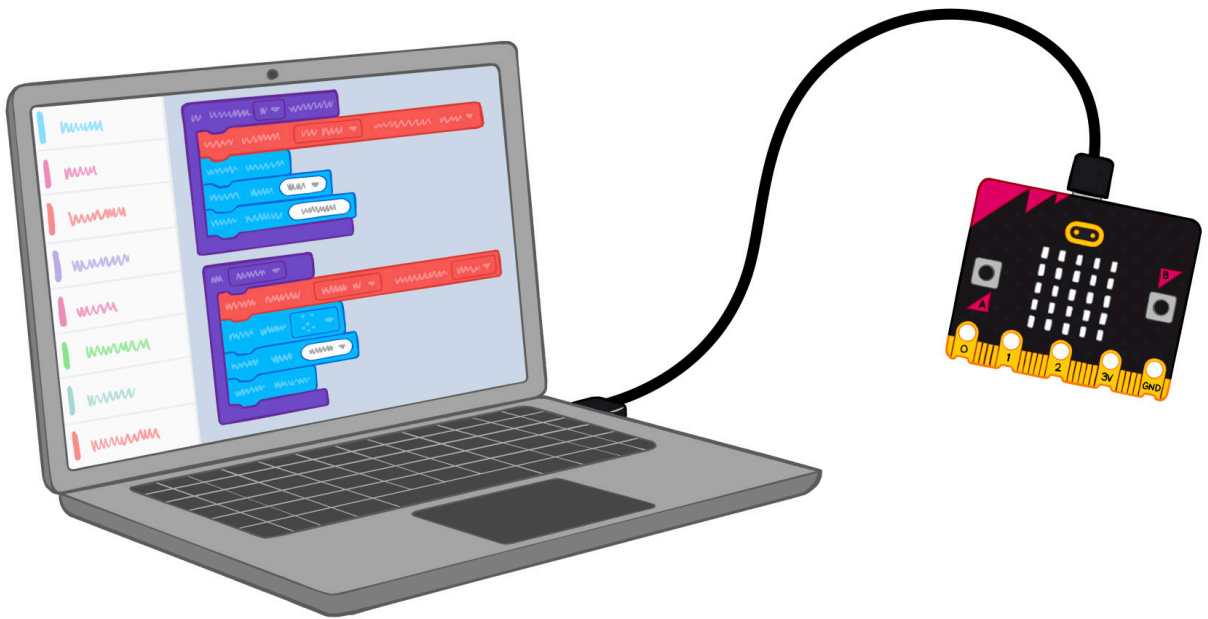
Physical Computing for Code.org CS Fundamentals Course D
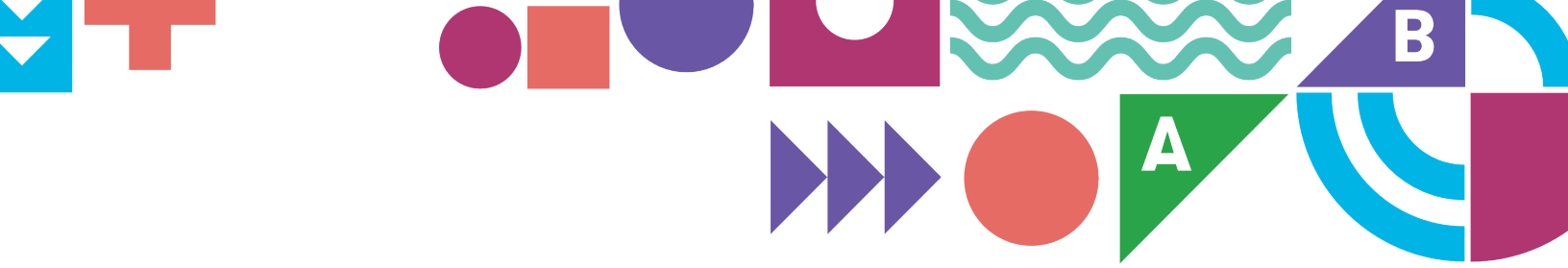

micro:bit

Contents

micro:bit

Section 1

Welcome to micro:bit Physical Computing Fundamentals

Welcome to micro:bit Physical Computing Fundamentals

What you'll find in this guide

This guide contains everything you need to use the BBC micro:bit to add the immersive power of physical computing to your teaching of Code.org's CS Fundamentals Course D.

You'll find:

- An **introductory exploration lesson** so you and your students can get to know some of the micro:bit's features and start making links with prior learning.

- A **coding lesson menu** to help you choose lessons that suit your students.

- A **guide to teaching the coding lessons**, which explains how you can use different resources that suit your students, such as step-by-step coding videos and micro:bit classroom sessions.

- **Three coding lessons** to choose from matched to relevant CS topics.

- Key **vocabulary** relevant to CS Fundamentals Course D and physical computing with the micro:bit.

What your students will learn — CS Fundamentals and physical computing

Lessons in this guide build on what your students are already learning and allow them to transfer that from the screen into physical projects they can code and hold in their hands.

The projects in these lessons are all designed to reinforce the computing topic of **conditionals**—statements that only run under certain conditions. If you work though each project sequentially, your students will gain experience of coding and using progressively more complex "if...then...else" statements.

Other computing topics from CS Fundamentals covered in these lessons include:

- **Events** – actions that cause something to happen
- **Loops** – the action of doing something over and over again
- **Input** – the information computers get from users or sensors
- **Output** – the information users get from computers
- **Variables** – labels for pieces of information used in a program

Note that **inputs** and **outputs** are covered in CS Fundamentals Course F but are included here as they are key concepts in physical computing.

Variables, used in the "Rock, paper, scissors" and "Activity picker" lessons, are not covered explicitly in CS Fundamentals Course D, but your students may have used them as a game score counter in step 11 of Course C Lesson 13: Mini Project: Chase Game.

There are four micro:bit physical computing guides for CS Fundamentals Courses C through F, so you can use micro:bit projects with students from second grade through fifth grade.

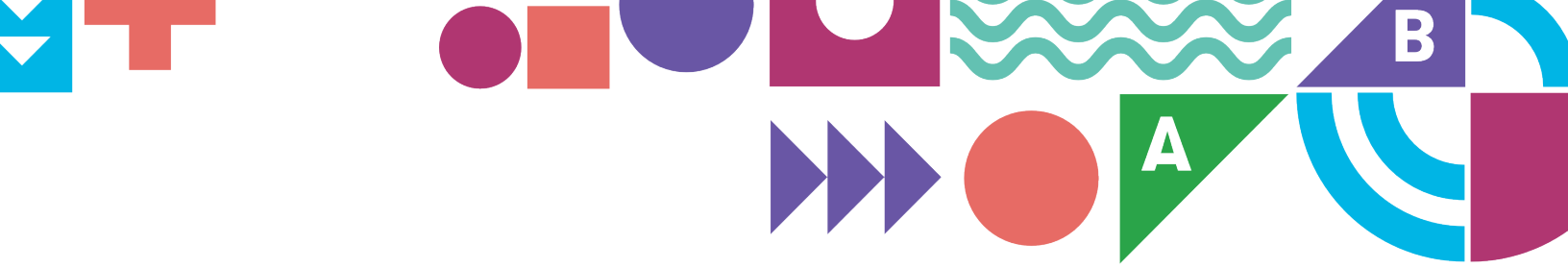# An introduction to the BBC micro:bit physical computing device
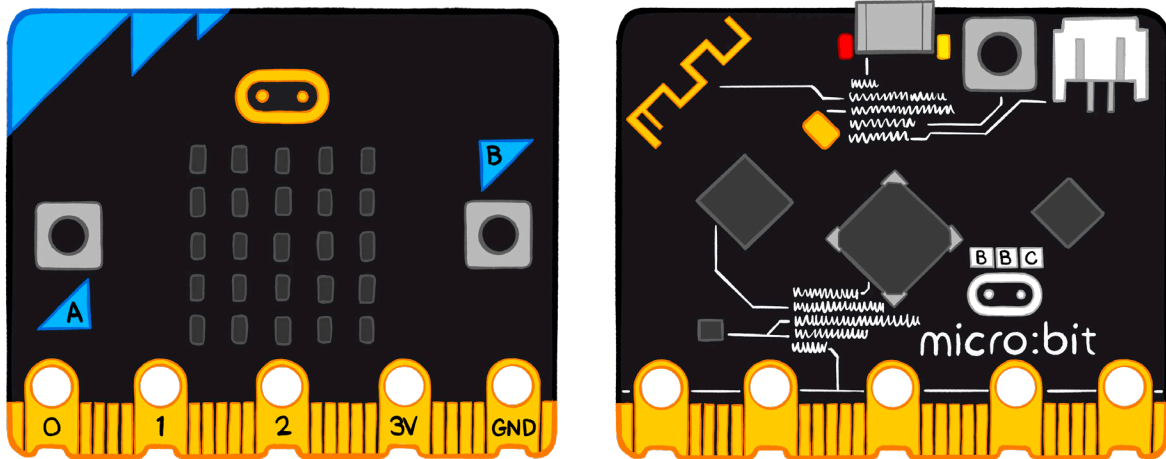
The BBC micro:bit is a tiny computer used by millions of children around the world. It's packed with **inputs** like buttons and sensors for light, movement, temperature, magnetism, and sound. It can also **output** pictures, numbers, and words on its LED display, make sound and music, and even communicate with other micro:bits using radio.

The micro:bit needs instructions—**programs**—to tell it what to do. Using the online Microsoft MakeCode block editor, your students will be able to create working code in seconds which they can test out in the **simulator** before transferring them to a real micro:bit over a USB cable. They can then unplug the micro:bit from the computer, attach a battery pack, and use their projects anywhere.

By making micro:bit projects, your students can take their code off the screen and make self-contained physical devices they can hold in the palms of their hands, making abstract computing concepts tangible.

You can find out more about the BBC micro:bit, including more projects, lessons, and support, on our website: https://microbit.org/

Section 2

"Meet your micro:bit" exploration lesson

"Meet your micro:bit" exploration lesson
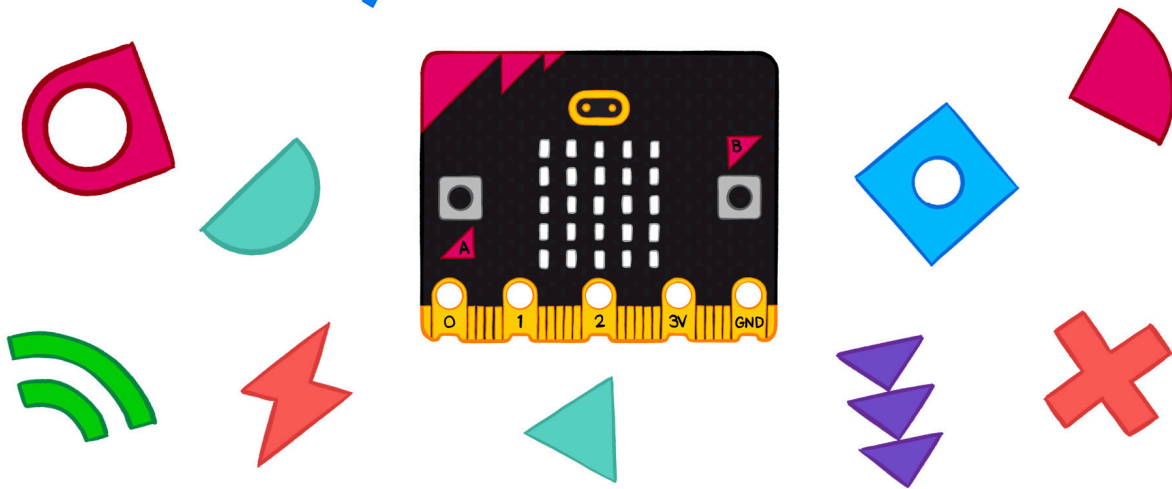
What your students will learn

This lesson is a pre-requisite for teaching the coding lessons in this guide. It gives your students an early hands-on experience to discover the excitement that learning with the micro:bit offers.

It helps reinforce what your students already know about code and computing concepts by transferring them to the physical world through exploring pre-programmed micro:bits.

The exploration is also designed for you to model reviewing code together, helping your students make links between familiar computing concepts and their practical application by programming a physical device.

Lesson format

The lesson requires some short **preparation** to transfer the exploratory project onto micro:bits to share with your students:

- Watch the video
- Put code onto micro:bits

Then **teach the lesson**:

- Warm-up: introduce the micro:bit and the activity.
- Main activity: students work in pairs to explore pre-programmed micro:bits. They'll explore different physical inputs and outputs while you challenge them to think about what computing concepts might be being used to make the program work.
- Wrap-up: discuss what they've discovered and look at the project code together. You'll start to familiarize yourselves with the online Microsoft MakeCode block editor.

You can optionally follow this with another lesson where students recreate the code for themselves.

Equipment list

You will need:

- Access to the MakeCode online editor on the teacher's computer.
- Several micro:bits with micro USB cords. One micro:bit for every two to three students is ideal.
- A power source for the micro:bits. Battery packs are best, but you can also power them from computer USB ports.

"Meet your microbit" video guide

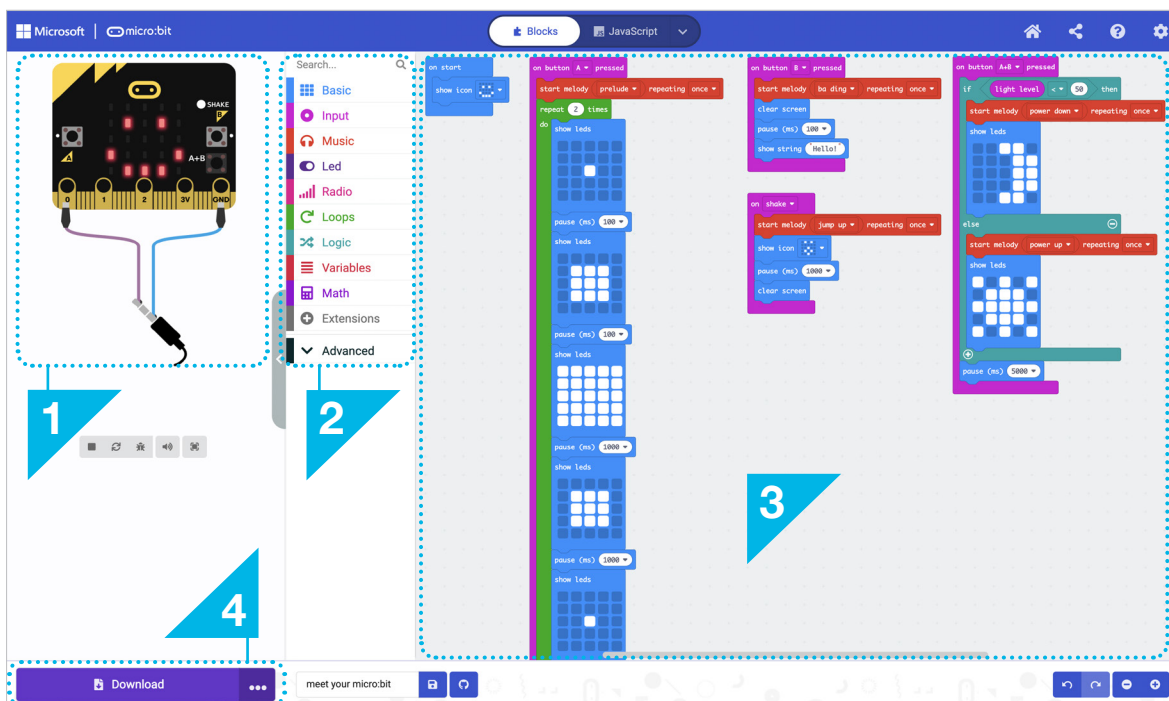**We've created a YouTube video to introduce this first lesson to you:**
https://mbit.io/csf-1-lesson-guide

Before the lesson preparation

Get to know the MakeCode editor

Follow this link to open the "Meet your micro:bit" MakeCode project:
https://mbit.io/csf-1-project

This is also a chance to familiarize yourself with the main parts of the MakeCode editor:

1 The **Simulator**, a virtual micro:bit that lets you demonstrate working code to your students, and lets your students test, debug, iterate, and improve their code before transferring it to their micro:bits. Click on button A to try it out.

2 The **Toolbox**, where you'll find the code blocks you need.

3 The **Workspace**, where you'll assemble program code blocks.

4 The **Download button**. Click on this when you're ready to transfer code to a micro:bit connected by a micro USB cable to your computer.

micro:bit

Transfer the project code onto your class micro:bits

Click on "Download" to save the MakeCode blocks program as a HEX file. This a special version of the program the micro:bit can understand.
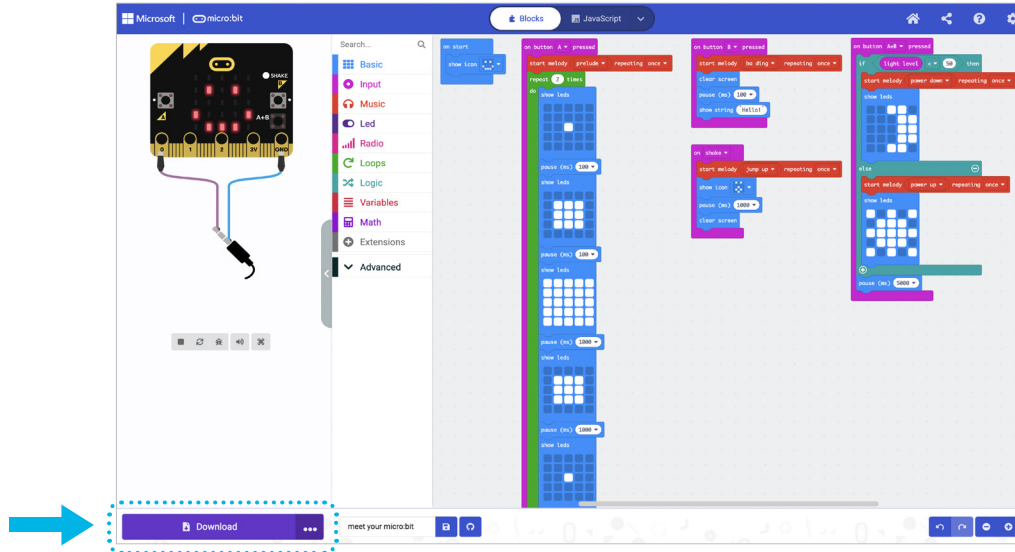
Plug a micro:bit into your computer's USB port. It should appear on your computer like a USB flash storage drive called MICROBIT.
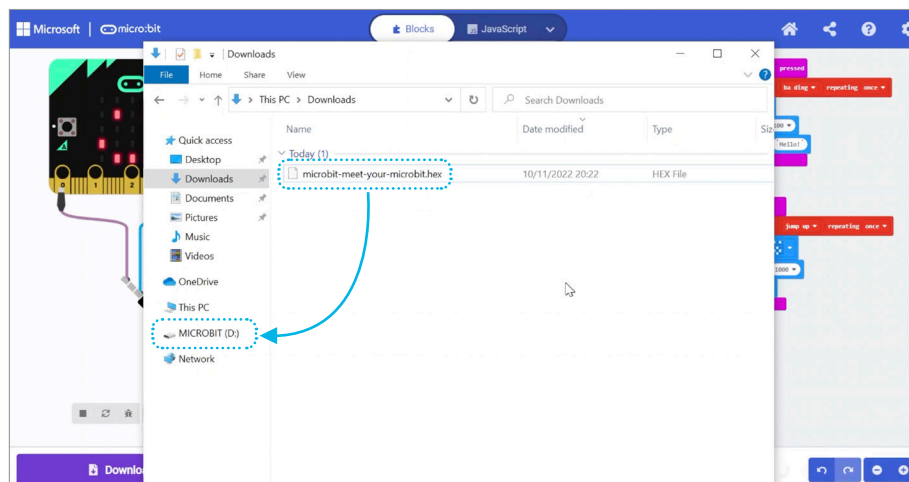
Drag and drop the "Meet your micro:bit" HEX file from your computer's downloads folder to the MICROBIT drive. You should see a light on the back of your micro:bit flash as it copies. The program will start running on the micro:bit as soon as the copying is complete. Note that programs stay on the micro:bit even when the power is disconnected.

Copy this HEX file onto several micro:bits—one for every two to three students is ideal.

Lesson 1: Meet your micro:bit

Warm up	
Introduction	Introduce the BBC micro:bit to your students: • The micro:bit is a tiny computer you can program to make self-contained projects to do different things. • For it to work, it first needs to be programmed to tell it what to do. • Today you'll be given micro:bits that have already been programmed. What can you figure out about the micro:bit and the code that makes it work?
Events	The program on these micro:bits responds to different **events**—can your students work out what they are? (Pressing different buttons and shaking the micro:bit)
Inputs & Outputs	Your students should consider what **inputs**—ways of getting information into the computer—this micro:bit project is using: the buttons, the accelerometer that senses when you shake the micro:bit, and the light sensor that measures how much light is falling on the micro:bit. Also ask your students what **outputs** it's using. Outputs are used to send information from a computer out into the world—for example pictures and text on the micro:bit's LED display.

A note about sound

If your students have the BBC micro:bit V2, they'll also hear different sounds on the built-in speaker output when they press different buttons and shake the micro:bit. You could ask your students to think about what kinds of information they can communicate with sound. Can sounds be happy? Sad? Fast? Slow? Can sounds even say "hello" or "goodbye"?

*If your students have the micro:bit V1, they can hear the sounds by connecting headphones or an amplified speaker with alligator clips to pin 0 and pin GND—the diagram in the MakeCode simulator shows you how to make the connection. **This is not essential—you can run this activity completely without sound.***

micro:bit

Main activity	
Examine the micro:bit	Students work in pairs or small groups to investigate the micro:bit and identify events, inputs, outputs, and any coding concepts they recognize from prior learning. They can optionally record their findings in any way you wish—for example on paper, whiteboards, or electronically. It can also be informal or formal—for example in a table. ![Hand-drawn table] | Event	Input	Output	Coding Concept
---	---	---	---
Press button A	Button A	LED display shows Zooming square animation	• Sequence • Loops	
Explore the "Meet your micro:bit" project	**Students should...** • Connect a power source (battery pack or USB cord) and notice what happens (a happy face appears on the LED display **output**). • Press button A to see a zooming square animation. Does it repeat? How many times? Does it get faster or slower? What computing concepts might be making this work? (A **sequence** to make an animation; **loops** to make the animation repeat). • Press button B to see text scroll across the display. Where else do they see visual information displayed like this? What might they use it for on the micro:bit? • Press buttons A and B together to make a sun or moon appear. Can your students figure out what is making the image change? (The LED display output can also work as a light sensor **input**, so if they cover the micro:bit they'll see a moon, and if they shine light on it they'll see a sun).

Get hands on!

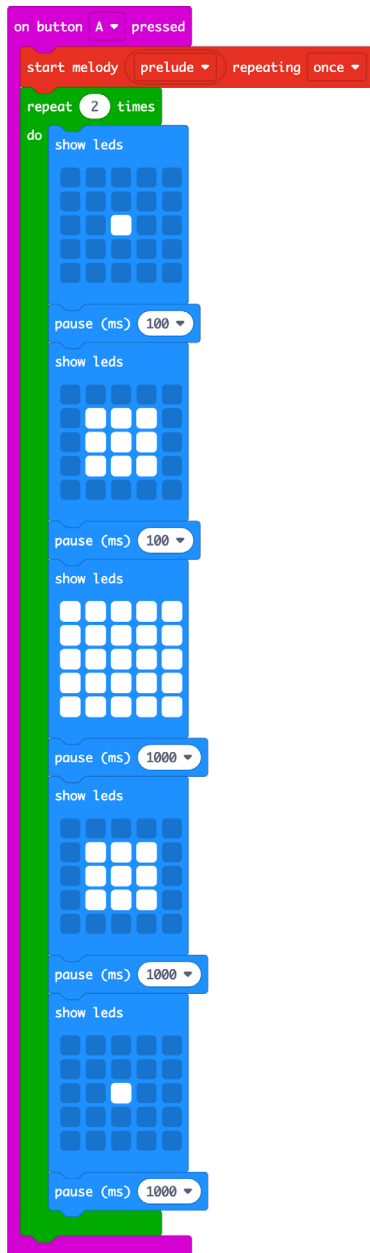| Wrap up	
Discussion	Ask your students to discuss what they discovered with you and the class.
	Share the code (https://mbit.io/csf-1-project) with your students and see how many computing concepts they already know that are used to make the project work. You can use the simulator as you go.
	Talk about as many of the event blocks as are appropriate to the time available and your students' prior learning.
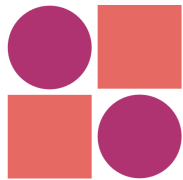
Code Expert!

CS talking points for code
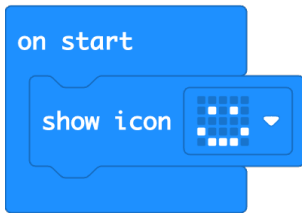
on start show icon [happy face icon]	**"on start" event block** The micro:bit starting to run the program is the **event** that triggers the "on start" block. "show icon happy" is the first instruction. So, we see a happy face on the micro:bit when it's powered up. It's an opportunity to talk about visual displays as **outputs**, which are how computers send information out into the world.
on button A ▾ pressed start melody prelude ▾ repeating once ▾ repeat 2 times do show leds [LED pattern - small square] pause (ms) 100 ▾ show leds [LED pattern - medium square] pause (ms) 100 ▾ show leds [LED pattern - large square] pause (ms) 1000 ▾ show leds [LED pattern - medium square] pause (ms) 1000 ▾ show leds [LED pattern - small square] pause (ms) 1000 ▾	**"on button A pressed" event block** "on button A pressed", "on button B pressed", "on button A plus B pressed" and "on shake" are all **input** blocks, triggered by different events. The "on button A pressed" block is triggered by the event of pressing the button A input on the micro:bit. It then carries out the instructions to play the sound output and display the zooming square animation. If you used sound in your exploration, look at the "start melody" block and listen to the sounds when you click on the buttons in the simulator. After the sound starts playing, a **loop** starts, repeating an animation two times. The display shows a square getting bigger quickly and getting smaller slowly. The **sequence** of images makes up the animation. The pause blocks keep images on the screen for different times—smaller numbers make the animation faster, larger numbers make it slower.

| ```
on button B ▾ pressed
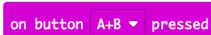start melody ( ba ding ▾ ) repeating once ▾
clear screen
pause (ms) 100 ▾
show string "Hello!"
``` | **"on button B pressed" event block**<br><br>The "on button B pressed" input block uses another event to trigger instructions that clear the screen, pause briefly, then show the word "hello" as a greeting on the LED display output. What else do your students think they could use that for? |
| ```
on button A+B ▾ pressed
if ( light level < ▾ 50 ) then
  start melody ( power down ▾ ) repeating once ▾
  show leds
  [moon pattern]
else
  start melody ( power up ▾ ) repeating once ▾
  show leds
  [sun pattern]
pause (ms) 5000 ▾
``` | **"on button A+B pressed" event block**<br><br>The "on button A+B pressed" input block reacts to the event of pressing both buttons at the same time.<br><br>The program uses a **conditional** statement:<br><br>**If** the light level is less than 50, **then** it shows a moon on the LED display output.<br><br>**Else** (otherwise) it shows a sun.<br><br>This part of the code shows different pictures on the display output, depending on the amount of light around you.<br><br>The light level is measured by another input, the "light level" block, which measures how much light is falling on the micro:bit.<br><br>So, the LED display works as a light sensor input and also as an output to display our pictures and messages.<br><br>What could your students build with a tiny computer that knows when it's light or dark around you? |
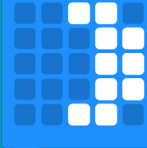
	"on shake" event block
	The "on shake" input block is triggered by an event when the micro:bit's accelerometer sensor detects movement. The micro:bit shows a surprised face, pauses for one second, and clears the screen.
	Ask your students what other technology they know that reacts to movement.
	How might you use movement in your own projects?

Extended learning

Other resources	You can use any of these resources to support an optional follow-up lesson, where your students recreate the code for themselves and practice transferring code to their micro:bits. You can further challenge them to remix the code to add more inputs and outputs.
	• Introduction video: https://mbit.io/csf-1-intro
	• Step-by-step coding video: https://mbit.io/csf-1-coding
	• Completed MakeCode project: https://mbit.io/csf-1-project
	• micro:bit classroom session: https://mbit.io/csf-1-classroom
	• View the project page on the microbit.org website: https://mbit.io/csf-1-make

Section 3

Coding lessons

Coding lesson menu

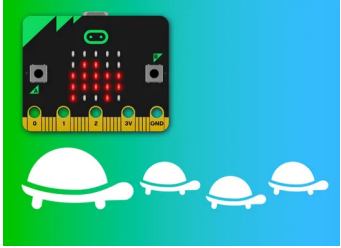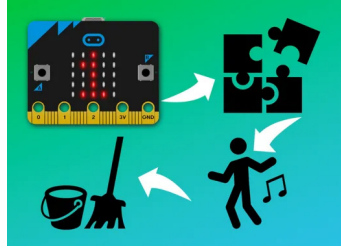Use this table to pick the projects that will work best for your students. They don't need to be taught in sequence—you can pick as many or as few as you like.

	Beginner	Intermediate	Stretch
Title	**Lesson 2: Saving sea turtles**	**Lesson 3: Rock, paper, scissors**	**Lesson 4: Activity picker**
Description	Make a prototype of automated lighting that switches on when it gets dark.	Use conditionals to make an electronic version of a classic game of chance.	Learn how to code an activity picker using random numbers and conditionals.
Key Concepts	• Loops • Conditionals: if...then... else • Inputs and outputs	• Events • Conditionals: if...then... else if...else • Variables*	• Events • Conditionals: if...then... else if...else • Variables*
CSTA Standards	**CS - Computing Systems** **1B-CS-02** - Model how computer hardware and software work together as a system to accomplish tasks. **AP - Algorithms & Programming** **1B-AP-10** - Create programs that include sequences, events, loops, and conditionals.	**CS - Computing Systems** **1B-CS-02** - Model how computer hardware and software work together as a system to accomplish tasks. **AP - Algorithms & Programming** **1B-AP-10** - Create programs that include sequences, events, loops, and conditionals.	**CS - Computing Systems** **1B-CS-02** - Model how computer hardware and software work together as a system to accomplish tasks. **AP - Algorithms & Programming** **1B-AP-10** - Create programs that include sequences, events, loops, and conditionals.

*Note: variables are not covered explicitly in CS Fundamentals Course D, but your students may have used them as a game score counter in Course C Lesson 13: Mini Project: Chase Game.

Coding lessons

How to teach the coding lessons

Make sure you've completed the "Meet your micro:bit" exploration lesson, then use the coding lesson menu (on page 19) to choose which lessons are the best fit for your students.

Follow this format when you're teaching any of the projects that follow:

Warm up	
Explain the aim	**Explain** the project aim
Reinforce key learning	**Reinforce** key learning relevant to CS Fundamentals and make connections with prior learning by either: • **Watching** an introduction video together *- or -* • **Exploring** the micro:bit project. Transfer the project code from the editor before the lesson to some micro:bits, which you can pass around your class like you did in "Meet your micro:bit"
Examine the code	**Examine** a completed program in the online simulator with your class by projecting the simulator as a giant virtual micro:bit, which gives you the option to look at the code blocks together
Main activity	
Student coding	Student coding activity. Pick whichever method suits your teaching style and students: • Step-by-step coding **videos** • A **micro:bit classroom** live coding session (see bottom of page 21)

micro:bit

Wrap up	
Discussion	For reflection on key learning
CS talking points for code	
Code blocks	Completed program blocks so you know where your students need to get to and can judge at a glance how complex each project is; explanations are provided so you can talk about how the code works with your students and help them debug
Extended learning	
Idea	Suggestions for additional learning that build off the lesson

micro:bit classroom

Each lesson activity's code can be opened directly in **micro:bit classroom**, our free tool for teaching live coding lessons. Before the lesson, you can view the code for yourself and decide what starter code to give your students. You can break the code blocks apart so they have to reassemble them, add instructions as comments (https://mbit.io/csf-comments), remove certain blocks, or give them a blank canvas.

Key features of micro:bit classroom:

- Free of charge

- No logins, registration or passwords needed for teachers or students

- Set starter code for your students

- View students' code from your computer in real time

- Download a snapshot of all students' code at any time as a Word document

- Save the whole lesson as a single file so you can resume it at a later date
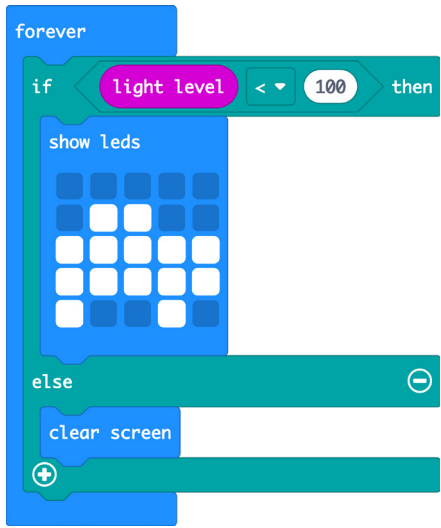
- Keep control of all your students' data

Find out more at https://classroom.microbit.org/

Lesson 2: Saving sea turtles coding

Level: Beginner

Warm up	
Explain the aim	Build a prototype of beach lighting for paths that guides humans safely but doesn't distract turtles by coding the micro:bit to light up the display only when it gets dark.
Reinforce key learning	**Watch** the introduction video: https://mbit.io/csf-d2-intro *- or -* **Explore** the project, uploaded onto micro:bits prior to the lesson: https://mbit.io/csf-d2-project Discuss where these concepts are being used and where students may have used them before: • **Loop:** The action of doing something over and over again (e.g., the "forever" block keeps running the code to check the light level) • **Conditional:** A statement that only runs under certain conditions (e.g., if the light level is below 100, show the turtle image on the LED display) • **Input:** The information computers get from users, or as in this case, from a sensor (e.g., light level reading) • **Output:** The information users get from computers (e.g., LED lights showing turtle image)
Examine the code	Share the project working in the simulator with your students and look at the code together prior to students coding the main activity: https://mbit.io/csf-d2-project

Main activity	
Student coding	Students make the project themselves using the editor and simulator, then transferring code to micro:bits. Pick one from: • Step-by-step coding **video**: https://mbit.io/csf-d2-coding • Live **micro:bit classroom** session: https://mbit.io/csf-d2-classroom
Wrap up	
Discussion	Share student work, revisit key concepts used, and explore ideas for extended learning.
CS talking points for code	
Completed program for teachers	https://mbit.io/csf-d2-project
	A forever **loop** keeps running the code that follows. A **conditional** statement is used to test **if** the light level measured by the light sensor **input** is below 100. **If** it is less than (<) 100 then it shows a turtle on the LED display **output**. **Else** (otherwise) it clears the screen. Then the forever loop runs the code again.

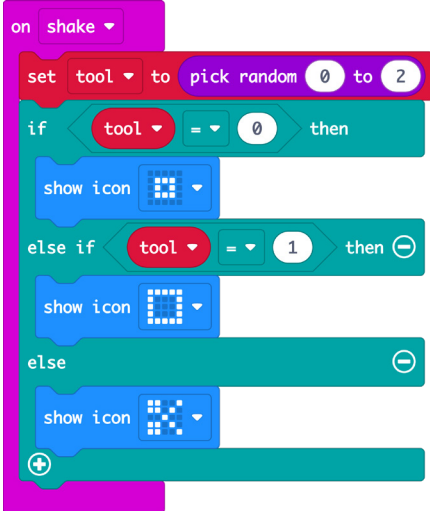| Extended learning	
Experiment with light levels	Vary the light level number to work better in your environment. Numbers higher than 100 will turn the light on even when it's brighter, numbers lower than 100 will only turn the light on when it gets darker.
Make an animation	Show an animation by making a sequence of different images—for example, a turtle walking.
Find this project and more on microbit.org	https://mbit.io/csf-d2-make

Lesson 3: Rock, paper, scissors coding

Level: Intermediate

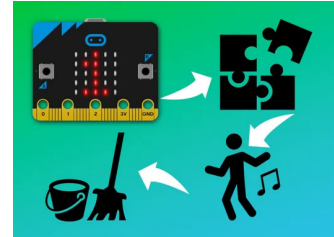| Warm up	
Explain the aim	Program a micro:bit to play a popular game of chance using events and conditionals.
Reinforce key learning	**Watch** the introduction video: https://mbit.io/csf-d3-intro *- or -* **Explore** the project, uploaded onto micro:bits prior to the lesson: https://mbit.io/csf-d3-project Discuss where these concepts are being used and where students may have used them before: • **Events:** actions that cause something to happen (e.g., when the accelerometer senses a shake movement, the "on shake" block carries out the instructions inside it) • **Conditional:** a statement that only runs under certain conditions (e.g., if the random number is 0, it shows a rock icon; if it's 1 it shows paper; otherwise it shows scissors) • **Variable:** a label for a piece of information used in a program (e.g., the code stores a random number between 0 and 2 in a variable called "tool" so it can be tested using a conditional "if" statement)
Examine the code	Share the project working in the simulator with your students and look at the code together prior to students coding the main activity: https://mbit.io/csf-d3-project

micro:bit

Main activity	
Student coding	Students make the project themselves using the editor and simulator, then transferring code to micro:bits. Pick one from: - Step-by-step coding **video**: https://mbit.io/csf-d3-coding - Live **micro:bit classroom** session: https://mbit.io/csf-d3-classroom

Wrap up	
Discussion	Share student work, revisit key concepts used, and explore ideas for extended learning.

CS talking points for code	
Completed program for teachers	https://mbit.io/csf-d3-project
	When the micro:bit's accelerometer senses it's been shaken, an "on shake" **event** block causes the code inside it to run. It sets a **variable** called "tool" to be a random number: 0, 1 or 2. It then tests the variable's value using a **conditional** statement: **if** it is 0, then it shows a rock icon. **If** it is 1, **then** it shows paper. **If** it is not 0 or 1, it must be 2, so **else** (otherwise), it shows scissors.

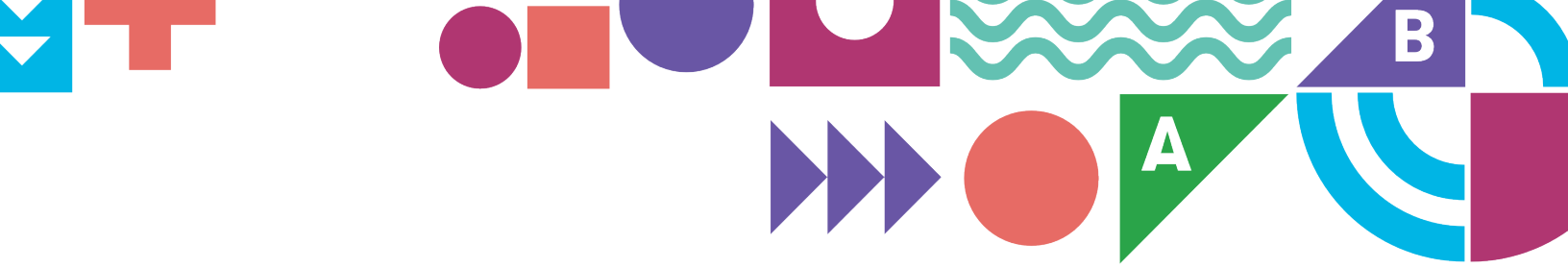| Extended learning	
Design your own icons	Use the "show leds" block to design your own icons for rock, paper, and scissors or use the "show string" block in the "Basic" section to show words instead of pictures.
Make a new game	Invent your own game using different objects and rules.
Find this project and more on microbit.org	https://mbit.io/csf-d3-make

Lesson 4: Activity picker coding

Level: Stretch

Warm up	
Explain the aim	Make an activity picker using random numbers and conditionals.
Reinforce key learning	**Watch** the introduction video: https://mbit.io/csf-d4-intro *- or -* **Explore** the project, uploaded onto micro:bits prior to the lesson: https://mbit.io/csf-d4-project Discuss where these concepts are being used and where students may have used them before: • **Events:** actions that cause something to happen (e.g., when you press button A on the micro:bit, the "on button A pressed" block carries out the instructions inside it) • **Conditional:** a statement that only runs under certain conditions (e.g., showing different activities on the display depending on the value of the random number created) • **Variable:** a label for a piece of information used in a program (e.g., the code stores a random number between 1 and 6 in a variable called "random_number" so it can be tested using a conditional "if" statement)
Examine the code	Share the project working in the simulator with your students and look at the code together prior to students coding the main activity: https://mbit.io/csf-d4-project

micro:bit

Main activity	
Student coding	Students make the project themselves using the editor and simulator, then transferring code to micro:bits.
	Pick one from:
	• Step-by-step coding **video**: https://mbit.io/csf-d4-coding
	• Live micro:bit **classroom session**: https://mbit.io/csf-d4-classroom
Wrap up	
Discussion	Share student work, revisit key concepts used, and explore ideas for extended learning.
CS talking points for code	
Completed program for teachers	https://mbit.io/csf-d4-project

Every time the **event** of the user pressing button A happens, the "on button A pressed" block runs the code inside it.

It sets a **variable** called "random_number" to be a random number between 1 and 6.

It uses multiple **conditional** "**if**" statements to test the variable. It shows different activities **if** the variable is equal to different numbers.

If the variable is not 1, 2, 3, 4 or 5, it must **else** (otherwise) be 6, so it shows the final option.

Extended learning	
Customize it	Customize your "Activity picker" by putting your own activities in the code.
More activities	Add more activities.
Avoid boring chores!	How could you make it less likely that it tells you to tidy your room?
Find this project and more on microbit.org	https://mbit.io/csf-d4-make

Section 4

Vocabulary

Vocabulary

Here are key computing terms from Code.org's CS Fundamentals Course D which are relevant to the lessons in this guide, along with some words frequently used in physical computing.

- **Algorithm** – A list of steps to finish a task.

- **Bug** – Part of a program that does not work correctly.

- **Conditional** – A statement that only runs under certain conditions.

- **Debugging** – Finding and fixing problems in an algorithm or program.

- **Hardware** – The physical, electronic parts of a computer system.

- **Input** – The information computers get from users or sensors. (This term is covered in CS Fundamentals Course F but is included here as inputs and outputs are key concepts in physical computing).

- **Event** – An action that causes something to happen.

- **LED** – Light Emitting Diode. The micro:bit has 25 LEDs on the front arranged in a 5 x 5 grid for showing pictures, numbers, and words.

- **Loop** – The action of doing something over and over again.

- **MakeCode** – The Microsoft block editor used for creating programs for your micro:bit. It's similar to Scratch and the block code editors used in CS Fundamentals.

- **micro:bit** – A tiny computer packed with sensors, inputs, and outputs.

- **Output** – The information users get from computers. (This term is covered in CS Fundamentals Course F but is included here as inputs and outputs are key concepts in physical computing).

- **Program** – An algorithm that has been coded into something that can be run by a machine.

- **Programming** – The art of creating a program.

- **Repeat** – To do something again.

- **Sensor** – A device that detects or records changes in the environment, such as the micro:bit's sensors for temperature, light, movement, and magnetism.

- **Simulator** *(MakeCode)* – A pretend, or virtual, micro:bit in the MakeCode editor that lets you test your programs before transferring them to a real micro:bit.

- **Software** – Programs made of code that tell computer what to do.

- **Toolbox** *(MakeCode)* – The middle part of the MakeCode editor where you find all the code blocks you need to build your micro:bit programs.

- **Variable** – A label for a piece of information used in a program. (This concept is introduced in CS Fundamentals Course F but is used in the "Rock, paper, scissors" and "Activity picker" projects in this booklet).

- **USB** – Universal Serial Bus, the connection used to connect a computer to a micro:bit to transfer programs to it.

- **Workspace** *(MakeCode)* – The right-hand part of the MakeCode editor where you assemble code blocks into programs.

Further reading

You can find more computing vocabulary for Code.org's CS Fundamentals Course D: https://studio.code.org/s/coursed-2023/vocab

The Micro:bit Educational Foundation web site also has a list of terms useful when teaching physical computing: https://microbit.org/teach/for-teachers/glossary/

micro:bit