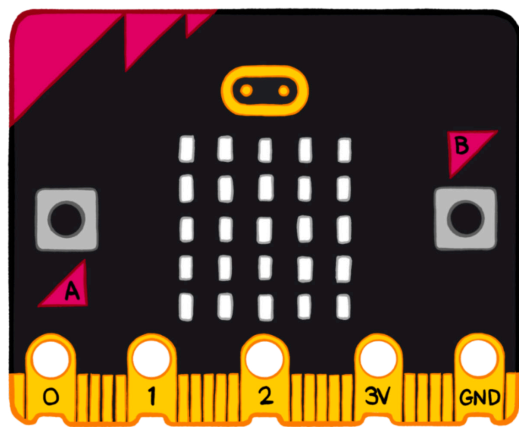


micro:bit Physical Computing Fundamentals

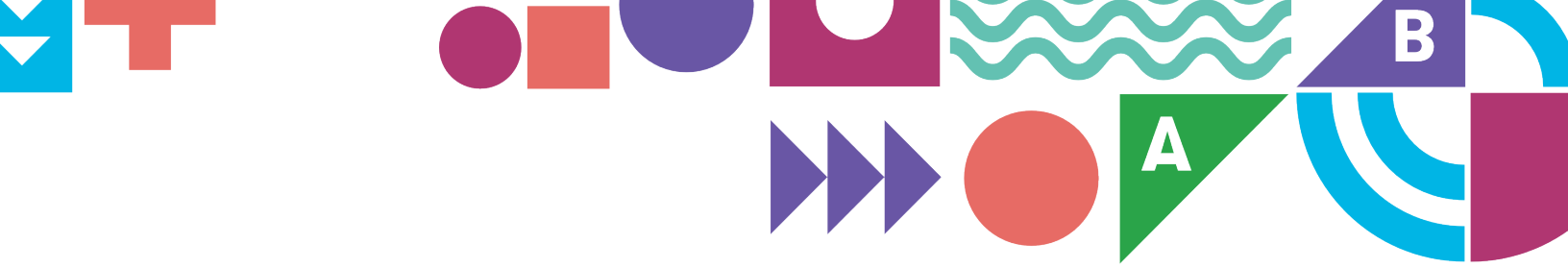
Physical Computing for
Code.org CS Fundamentals Course F




micro:bit

Contents

1. Welcome to micro:bit Physical Computing Fundamentals	3
What you'll find in this guide.....	4
CS Fundamentals and physical computing.....	5
An introduction to the BBC micro:bit physical computing device.....	6
2. “Meet your micro:bit” exploration	7
What your students will learn.....	8
Lesson format.....	9
Equipment list.....	9
“Meet your microbit” video guide.....	9
Before the lesson preparation.....	10
Lesson 1: Meet your micro:bit.....	12
<i>Warm up</i>	12
<i>Main activity</i>	13
<i>Wrap up</i>	14
<i>CS talking points for code</i>	15
<i>Extended learning</i>	17
3. Coding lessons	18
Coding lesson menu.....	19
How to teach the coding lessons.....	20
Lesson 2: Step counter.....	22
Lesson 3: micro:bit pet.....	25
Lesson 4: Max-min thermometer.....	29
4. Vocabulary	34



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 F](#).

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 F 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.

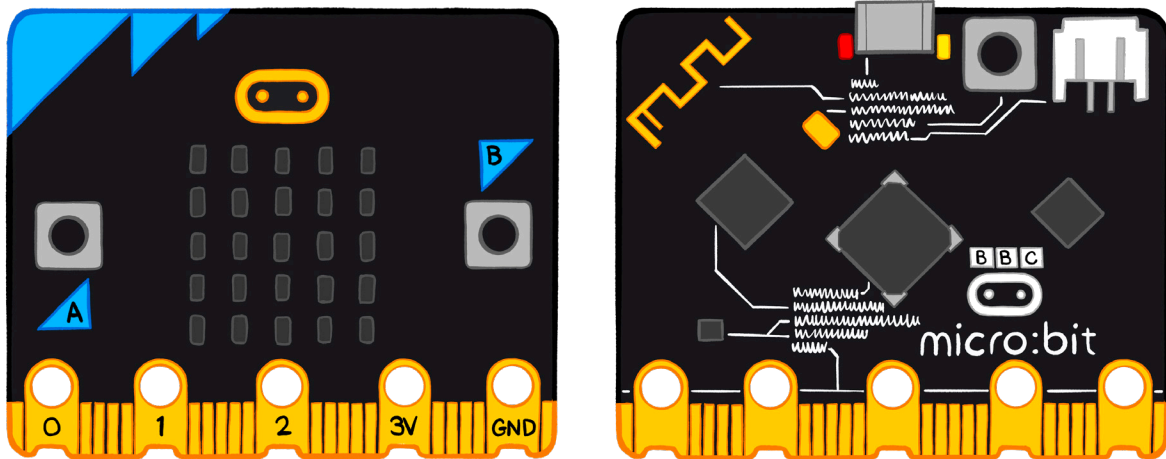
Computing topics from CS Fundamentals covered in these lessons include:

- **Variable** – a label for a piece of information used in a program
- **Data** – a collection of information
- **Simulation** – a program which replicates or mimics key features of a real world event
- **Event** – an action that causes something to happen
- **Loop** – the action of doing something over and over again
- **Conditional** – a statement that only runs under certain conditions

Note that **events**, **loops**, and **conditionals** are not specifically covered as main topics in CS Fundamentals Course F, but are covered in previous courses, and your students should already be familiar with them from prior learning.

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

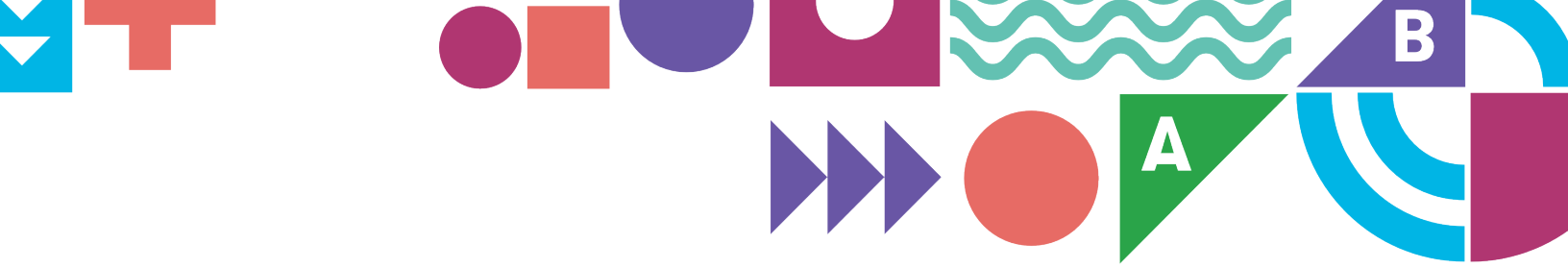


The BBC micro:bit is a tiny computer used by millions of children around the world. It's packed with **inputs** like buttons and sensors for light, movement, temperature, magnetism, and sound. It can also **output** pictures, numbers, and words on its LED display, make sound and music, and even communicate 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/>



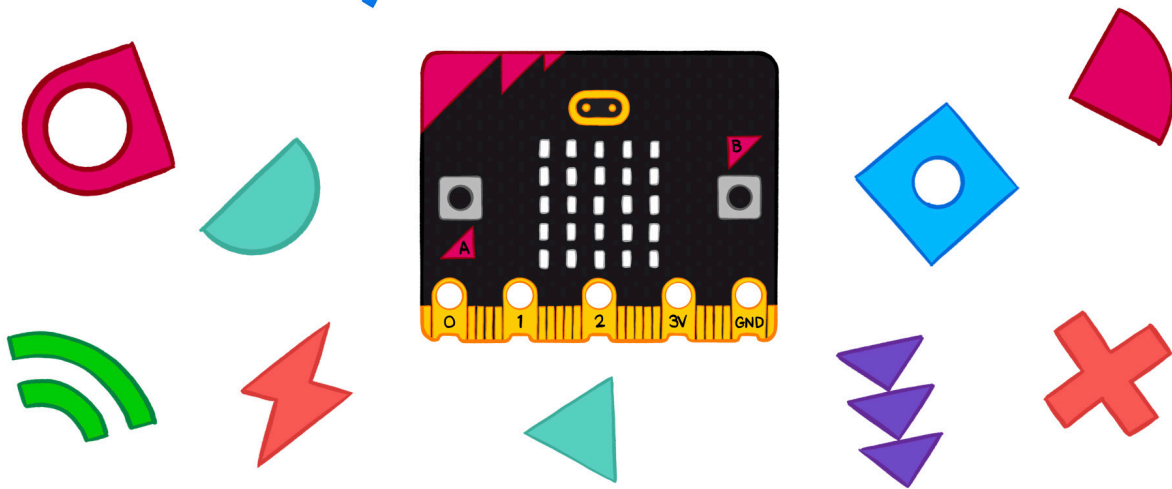
Section 2

“Meet your micro:bit” exploration lesson



“Meet your micro:bit” exploration lesson

meet your micro:bit!



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

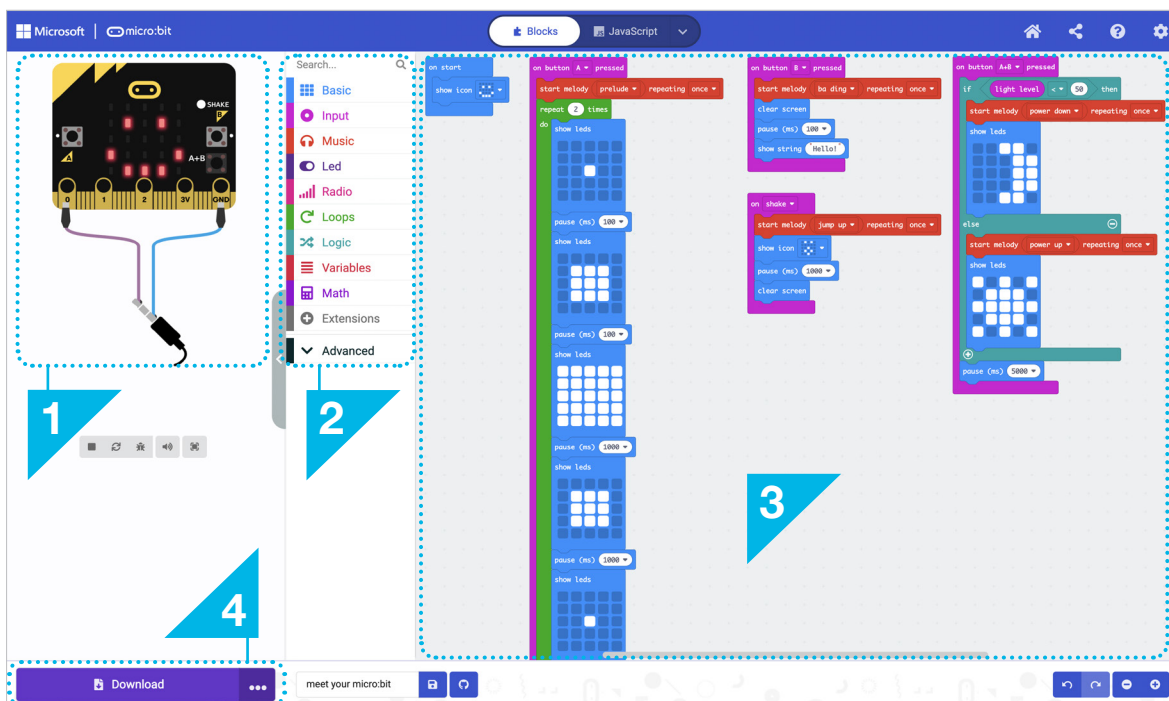
We've created a YouTube video to introduce this first lesson to you:

<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>

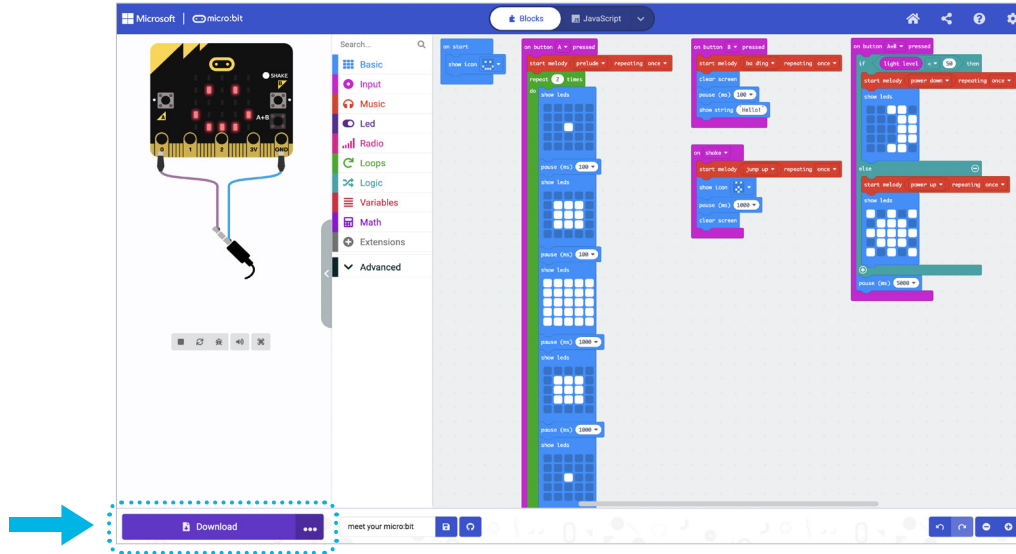


This is also a chance to familiarize yourself with the main parts of the MakeCode editor:

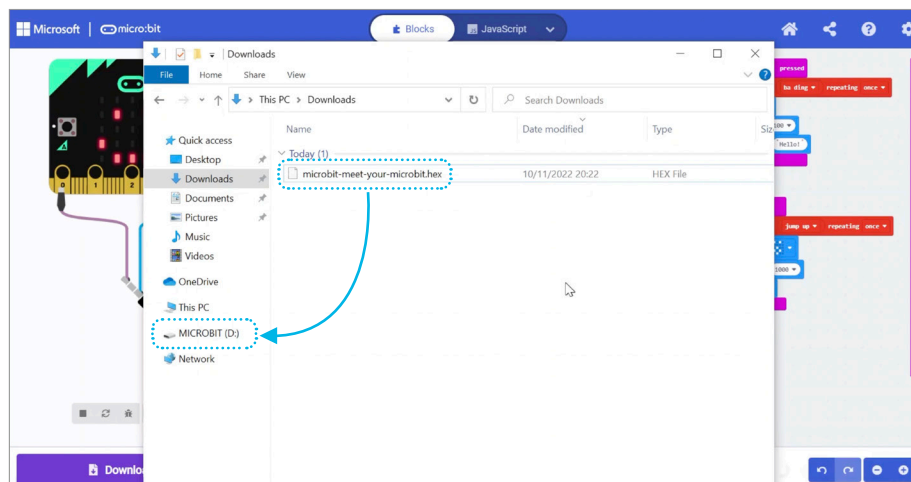
- 1 The **Simulator**, a virtual micro:bit that lets you demonstrate working code to your students, and lets your students test, debug, iterate, and improve their code before transferring it to their micro:bits. Click on button A to try it out.
- 2 The **Toolbox**, where you'll find the code blocks you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

Transfer the project code onto your class micro:bits

Click on “Download” to save the MakeCode blocks program as a HEX file. This is 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	<p>Introduce the BBC micro:bit to your students:</p> <ul style="list-style-type: none">• 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	<p>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)</p>
Inputs & Outputs	<p>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.</p> <p>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.</p>

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.***

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.

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!**

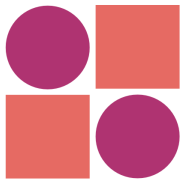
Wrap up

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.

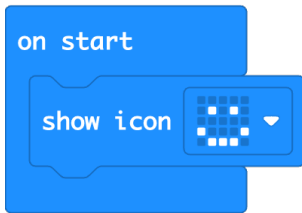
Talk about as many of the event blocks as are appropriate to the time available and your students' prior learning.



Code Expert!

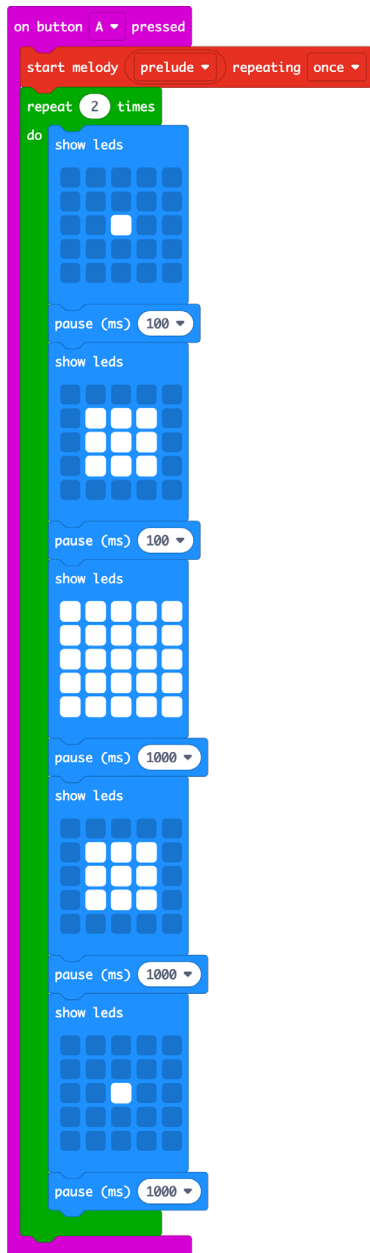


CS talking points for code



“on start” 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” 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



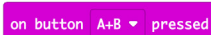
pause (ms) 100



show string "Hello!"

“on button B pressed” event block

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



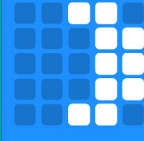
if light level < 50 then



start melody power down repeating once



show leds



else



start melody power up repeating once



show leds



pause (ms) 5000

“on button A+B pressed” event block

The “on button A+B pressed” input block reacts to the event of pressing both buttons at the same time.

The program uses a **conditional** statement:

If the light level is less than 50, **then** it shows a moon on the LED display output.

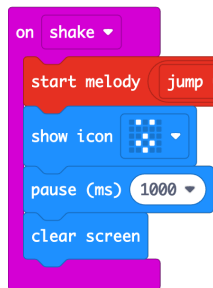
Else (otherwise) it shows a sun.

This part of the code shows different pictures on the display output, depending on the amount of light around you.

The light level is measured by another input, the “light level” block, which measures how much light is falling on the micro:bit.

So, the LED display works as a light sensor input and also as an output to display our pictures and messages.

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.

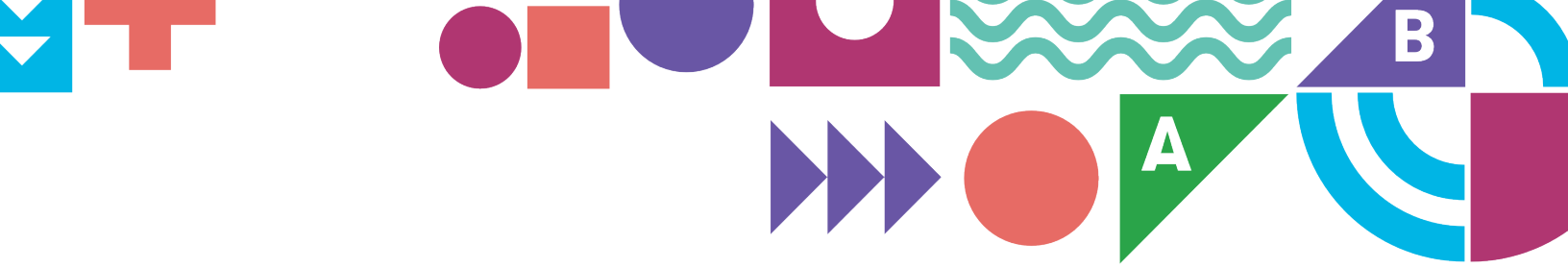
How might you use movement in your own projects?

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>




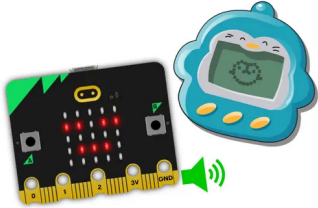
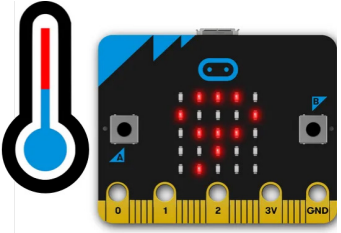
Section 3

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 as a sequence—you can pick as many or as few as you like.

	Beginner	Intermediate	Stretch
Title	<p>Lesson 2: Step counter</p> 	<p>Lesson 3: micro:bit pet</p> 	<p>Lesson 4: Max-min thermometer</p> 
Description	Use variables to track and collect data about how far you walk or move.	Create an electronic pet and learn how variables can be used to make computer simulations of the real world.	Make a thermometer that records highs and lows using conditionals and more advanced use of variables.
Key Concepts	<ul style="list-style-type: none"> • Variables • Data • Events 	<ul style="list-style-type: none"> • Simulation • Variables • Loops • Events • Conditionals 	<ul style="list-style-type: none"> • Variables • Data • Loops • Events • Conditionals
CSTA Standards	<p>CS – Computing systems</p> <p>1B-CS-02 - Model how computer hardware and software work together as a system to accomplish tasks.</p> <p>AP - Algorithms & Programming</p> <p>1B-AP-09 - Create programs that use variables to store and modify data.</p> <p>1B-AP-10 - Create programs that include sequences, events, loops, and conditionals.</p>	<p>CS – Computing systems</p> <p>1B-CS-02 - Model how computer hardware and software work together as a system to accomplish tasks.</p> <p>AP - Algorithms & Programming</p> <p>1B-AP-09 - Create programs that use variables to store and modify data.</p> <p>1B-AP-10 - Create programs that include sequences, events, loops, and conditionals.</p>	<p>CS – Computing systems</p> <p>1B-CS-02 - Model how computer hardware and software work together as a system to accomplish tasks.</p> <p>DA – Data & Analysis</p> <p>1B-DA-07 - Use data to highlight or propose cause-and-effect relationships, predict outcomes, or communicate an idea.</p> <p>AP - Algorithms & Programming</p> <p>1B-AP-09 - Create programs that use variables to store and modify data.</p> <p>1B-AP-10 - Create programs that include sequences, events, loops, and conditionals.</p>

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: <ul style="list-style-type: none">• 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: <ul style="list-style-type: none">• Step-by-step coding videos• A micro:bit classroom live coding session (see bottom of page 21)

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

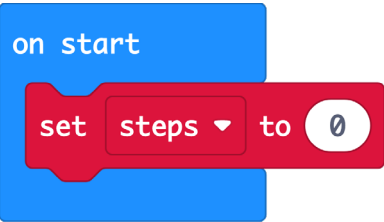
Find out more at <https://classroom.microbit.org/>

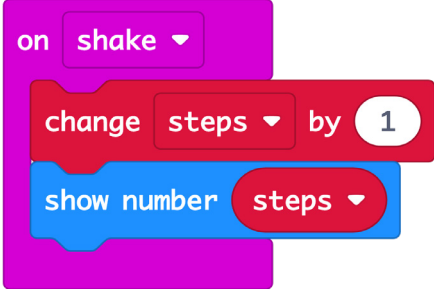
Lesson 2: Step counter coding

Level: Beginner



Warm up	
Explain the aim	Code and make a simple fitness tracker using variables to record how many steps you've taken.
Reinforce key learning	<p>Watch the introduction video: https://mbit.io/csf-f2-intro</p> <p>- or -</p> <p>Explore the project, uploaded onto micro:bits prior to the lesson: https://mbit.io/csf-f2-project</p> <p>Discuss how you might wear the micro:bit and battery pack—for example, by putting it in a pocket or attaching it to your arm, shoe, or leg.</p> <p>Discuss which computing concepts are being used and where students may have used them before:</p> <ul style="list-style-type: none">• Variable: a label for a piece of information used in a program (e.g., the number of steps taken is stored in a variable)• Data: a collection of information (e.g., the number of steps taken is data—you might collect your own step counts over several days, or compare step counts with other people; this is all data)• Event: an action that causes something to happen (e.g., when the micro:bit's accelerometer detects movement, it triggers code to increase the step count variable by 1 and show it on the LED display)

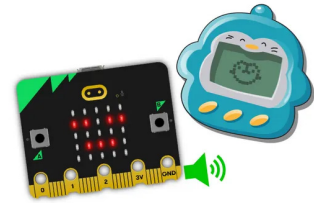
Examine the code	Share the project working in the simulator with your students and look at the code together prior to students coding the main activity: https://mbit.io/csf-f2-project
Main activity	
Student coding	Students make the project themselves using the editor and simulator, then transferring code to micro:bits. Pick one from: <ul style="list-style-type: none"> • Step-by-step coding video: https://mbit.io/csf-f2-coding • Live micro:bit classroom session: https://mbit.io/csf-f2-classroom Students can then try the step counters out with real micro:bits to collect data.
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-f2-project
 <p>The image shows a Scratch code block for an 'on start' event. It is a blue block with the text 'on start' in white. Attached to it is a red 'set' block. The 'set' block has a dropdown menu showing 'steps', followed by the word 'to', and a white circle containing the number '0'.</p>	At the start of the program, the “on start” block sets the value of the “steps” variable to zero. This is called initializing the variable. It’s important to do this so you know what value the variable will have at the start.

	<p>When the micro:bit is shaken (for example, when you take a step) the accelerometer sensor input triggers the “shake” event block to carry out two instructions. The “steps” variable is increased by 1 from its current value (regardless of what that value is).</p> <p>It then shows the updated value of the “steps” variable on the LED display output.</p> <p>The sequence of these instructions is important: you must update the variable before showing it, or the information shown will be out of date and you will see inaccurate data.</p>
<p>Extended learning</p>	
<p>Reset</p>	<p>Make a reset button by adding an “on button A pressed” event block to reset the step counter variable back to 0.</p>
<p>Daily trends</p>	<p>Collect step count data over several days and see if you can spot any patterns suggesting when you are most active.</p>
<p>Calculate the distance</p>	<p>Measure the length of your average step and multiply this by your step count to find out how far you have walked. Can you code your micro:bit to work this out for you using one of the Math blocks in MakeCode?</p>
<p>Make a more sensitive counter</p>	<p>Code a more advanced “Sensitive step counter” project which uses accelerometer strength readings: https://mbit.io/csf-step-counter-2</p>
<p>Data logging</p>	<p>If you have the micro:bit V2 (the version with the built-in speaker), use data logging to make a personalised step counter: https://mbit.io/csf-movement-data</p>
<p>Find this project and more on microbit.org</p>	<p>https://mbit.io/csf-f2-make</p>



Lesson 3: micro:bit pet coding

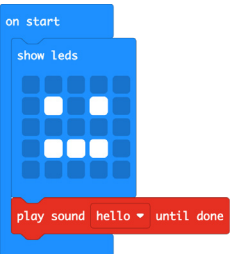
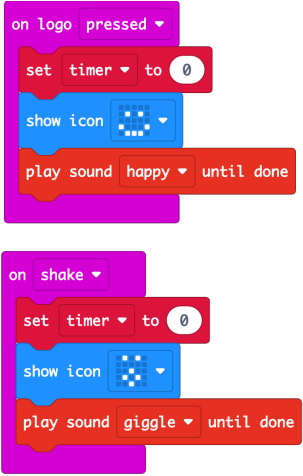
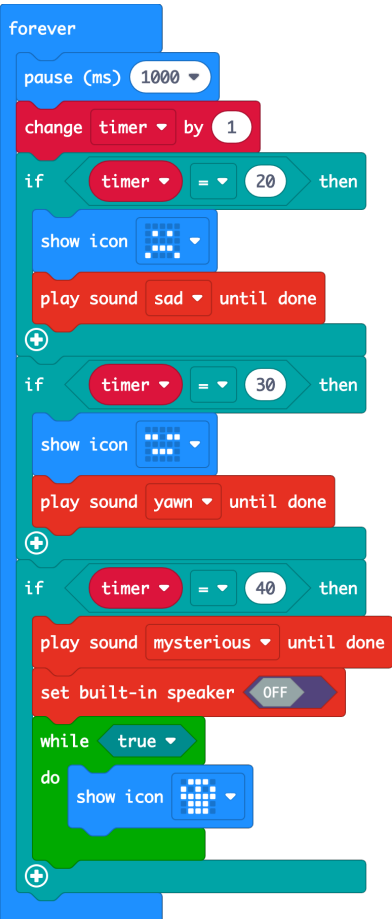
Level: Intermediate



Warm up	
Explain the aim	Code your own electronic pet and learn how loops, variables, events, and conditionals can work together to create a simulation of a pet that craves attention and gets sad if you ignore it.
Reinforce key learning	<p>Watch the introduction video: https://mbit.io/csf-f3-intro</p> <p>- or -</p> <p>Explore the project, uploaded onto micro:bits prior to the lesson: https://mbit.io/csf-f3-project</p> <p>Discuss where these concepts are being used and where students may have used them before:</p> <ul style="list-style-type: none">• Simulation: a program which replicates or mimics key features of a real world event (e.g., the micro:bit code mimics the behaviour of an animal)• Variable: a label for a piece of information used in a program (e.g., a variable is used to store how long you have ignored your pet for)• Loop: the action of doing something over and over again (e.g., a loop keeps the timer running)• Event: an action that causes something to happen (e.g., shaking or touching the micro:bit keeps the pet happy)• Conditional: a statement that only runs under certain conditions (e.g., if the timer count reaches different numbers, the pet gets sad, falls asleep, or even dies)

<p>Examine the code</p>	<p>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-f3-project</p> <p>Note that the code for this project is designed to work in the MakeCode simulator and on the micro:bit V2 (the version with the built-in speaker and gold touch logo).</p> <p>If you have the micro:bit V1, you'll need to make the following changes:</p> <ul style="list-style-type: none"> • Delete the “play sound” blocks or replace them with “start melody” – see “A note about sound” on page 12 • Replace the “on logo pressed” block with “on button A pressed
<p>Main activity</p>	
<p>Student coding</p>	<p>Students make the project themselves using the editor and simulator, then transferring code to micro:bits.</p> <p>Pick one from:</p> <ul style="list-style-type: none"> • Step-by-step coding video: https://mbit.io/csf-f3-coding • Live micro:bit classroom session: https://mbit.io/csf-f3-classroom
<p>Wrap up</p>	
<p>Discussion</p>	<p>Share student work, revisit key concepts used, and explore ideas for extended learning.</p>
<p>CS talking points for code</p>	
<p>Completed program for teachers</p>	<p>https://mbit.io/csf-f3-project</p>



	<p>At the start of the program, the micro:bit shows a neutral face and makes a “hello” sound.</p>
	<p>The pet craves attention, so the event of pressing the logo or shaking the micro:bit resets the “timer” variable back to 0, shows an interested expression on the LED display output, and plays a happy sound.</p>
	<p>A “forever” loop keeps increasing the “timer” variable and checking how long you have ignored the pet for.</p> <p>The code uses conditionals (“if... then...” blocks) to show different images and play different sounds depending on how long the pet has been ignored for.</p> <p>If the timer reaches 40 seconds, the pet dies and a special “while true” loop effectively stops the code running. You can bring your micro:bit back to life by pressing the reset button on the back of the micro:bit or by disconnecting and reconnecting the power (battery pack or USB cord).</p>

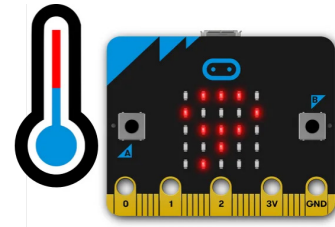


Extended learning	
Add new pet reactions	Add your own expressions and sounds at different values of the “timer” variable.
Talk to your pet	If you have a micro:bit V2 (the version with the built-in speaker), add an “on loud sound” input block so your pet can respond when you talk or sing to it using the built-in microphone.
Two pets are better than one	If you have previously used the micro:bit’s radio feature (https://mbit.io/csf-radio), you can make pets interact with each other; for instance, by making a pet happy when it’s close to another micro:bit pet. You can use the “radio set transmit power” block to reduce the power to 0 so pets have to be very close to each other for this to work.
Find this project and more on microbit.org	https://mbit.io/csf-f3-make



Lesson 4: Max-min thermometer coding

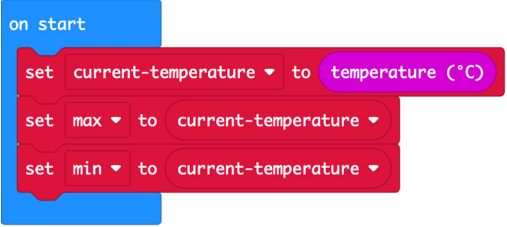
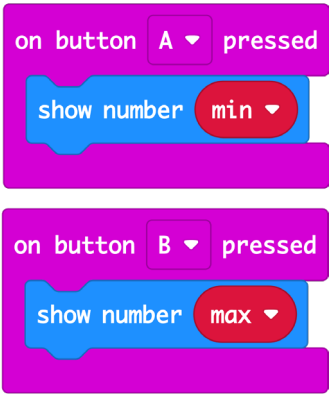
Level: Stretch



Warm up	
Explain the aim	Make a thermometer that records high and low temperatures using conditionals and more advanced use of variables.
Reinforce key learning	<p>Watch the introduction video: https://mbit.io/csf-f4-intro</p> <p>- or -</p> <p>Explore the project, uploaded onto micro:bits prior to the lesson: https://mbit.io/csf-f4-project</p> <p>Discuss how you might use this—where would you place micro:bits to collect this data? How might you gather, visualize, and interpret the data?</p> <p>Discuss where these computing concepts are being used and where students may have used them before:</p> <ul style="list-style-type: none">• Variable: a label for a piece of information used in a program (e.g., the maximum and minimum temperatures are stored in variables so we can keep them updated and recall the information when we need it)• Data: a collection of information (e.g., the values of the maximum and minimum temperature are data)• Events: an action that causes something to happen (e.g., the event of pressing button A or B causes the minimum or maximum temperature to be displayed)

	<ul style="list-style-type: none"> • Loops: the action of doing something over and over again (e.g., the micro:bit uses a loop to keep showing a flashing period and checking if the temperature is lower than the previous minimum or higher than the previous maximum) • Conditional: a statement that only runs under certain conditions (e.g., if the current temperature is lower than the minimum, update the record of the minimum; else if it is higher than the maximum, update the record of the maximum temperature)
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-f4-project
Main activity	
Student coding	Students make the project themselves using the editor and simulator, then transferring code to micro:bits. Pick one from: <ul style="list-style-type: none"> • Step-by-step coding video: https://mbit.io/csf-f4-coding • Live micro:bit classroom session: https://mbit.io/csf-f4-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-f4-project



	<p>At the start of the program, the “on start” block initializes three variables:</p> <ul style="list-style-type: none"> • “current temperature” which will store the temperature when it’s tested every 2 seconds • “max” which will store the maximum temperature • “min” which will store the minimum temperature <p>Note that at the start these all have the same value.</p>
	<p>The event of pressing a button causes the minimum or maximum temperature to be shown on the LED display output.</p>



```

forever
  show string " "
  set current-temperature to temperature (°C)
  if current-temperature < min then
    set min to current-temperature
  +
  if current-temperature > max then
    set max to current-temperature
  +
  pause (ms) 1000
  clear screen
  pause (ms) 1000

```

This is the smart part of the code! A “forever” **loop** keeps the program running.

It shows a period on the LED display so you know it’s running.

The “current-temperature” variable is updated to the most current reading from the micro:bit’s temperature sensor **input**, located inside its processor.

Conditionals (if... then... else if... then...) compare the new reading with the previous “max” and “min” values.

If the “current-temperature” value is less than the previous “min” value, it sets “min” to be the same as the “current-temperature”.

Else, if the “current-temperature” value is more than the previous “max” value, it sets “max” to be the same as the “current-temperature”.

It then waits 1 second, clears the LED display and waits 1 more second.

The loop then starts again by resuming the flashing period on the display.

Extended learning

Show the current temperature

Add an “on button A+B pressed” input block to show the current temperature.

```

on button A+B pressed
  show number temperature (°C)

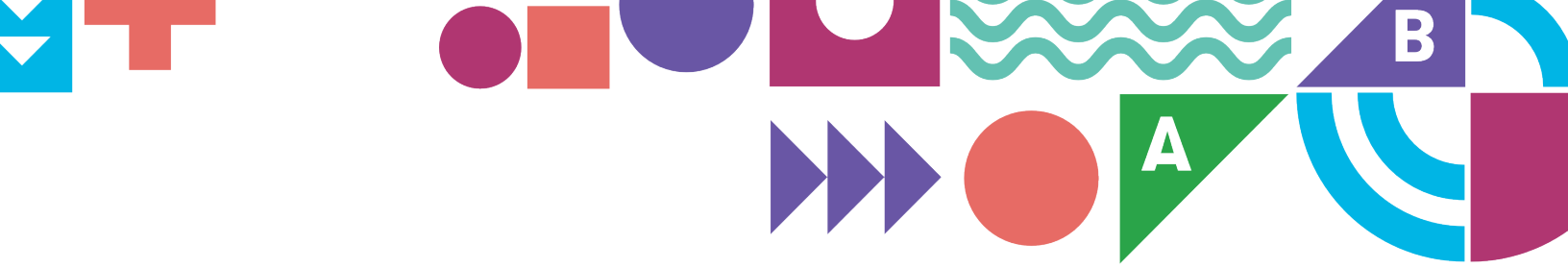
```

Convert Celsius to Fahrenheit

The micro:bit thermometer works in the Celsius temperature scale. Add some code to convert the temperature to Fahrenheit when it’s displayed. You can use a function like in this project or borrow the math blocks from it: <https://mbit.io/csf-fahrenheit>

Data logging	The micro:bit will lose its maximum-minimum data when the power is disconnected. If you have the micro:bit V2 (the version with the built-in speaker), you can use its data logging feature to record data that stays on the micro:bit even when the power is removed, such as in this project that records temperature and light levels: https://mbit.io/csf-data-logger
Find this project and more on microbit.org	https://mbit.io/csf-f4-make





Section 4

Vocabulary





Vocabulary

Here are key computing terms from Code.org's CS Fundamentals Course F 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.
- **Data** – A collection of information.
- **Debugging** – Finding and fixing problems in an algorithm or program.
- **Event** – An action that causes something to happen.
- **Gestures** – Different ways of moving the micro:bit; for example shaking it, placing it display-side up flat on a table, turning it face down, and so on.
- **Hardware** – The physical, electronic parts of a computer system.
- **Input** – The information computers get from users or sensors.
- **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.
- **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.

- **Simulation** – A program which replicates or mimics key features of a real world event
- **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.
- **USB** – Universal Serial Bus, the connection used to connect a computer to a micro:bit to transfer programs to it.
- **Variable** – A label for a piece of information used in a program.
- **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 F: <https://studio.code.org/s/coursef-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/>