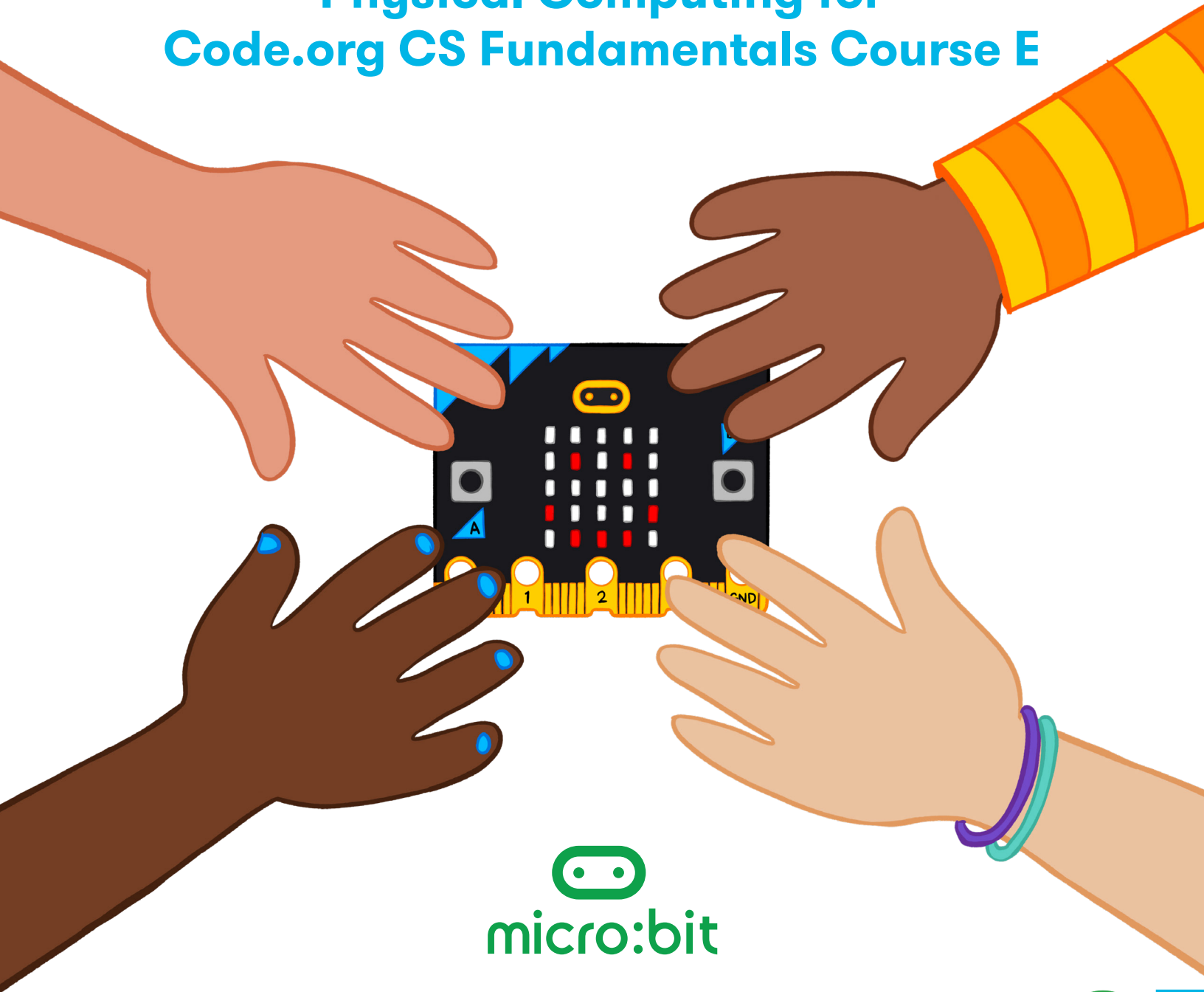




# micro:bit Physical Computing Fundamentals

Physical Computing for  
Code.org CS Fundamentals Course E

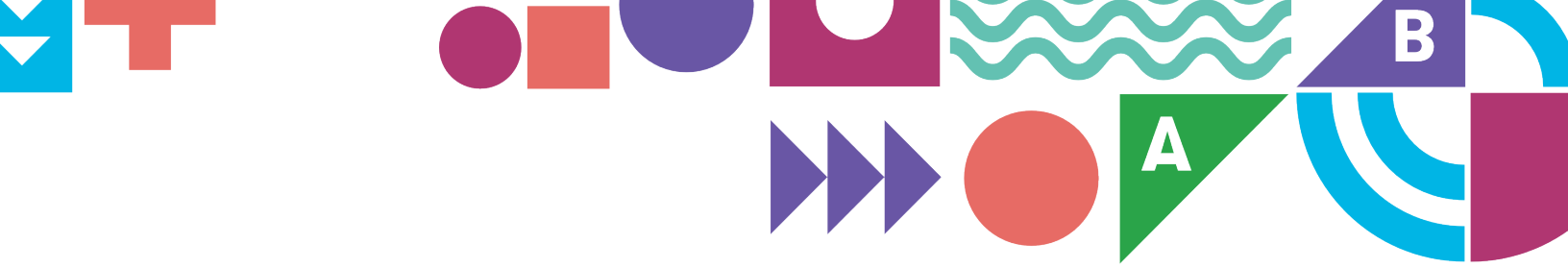


micro:bit



# Contents

<b>1. Welcome to micro:bit Physical Computing Fundamentals</b> .....	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
<b>2. “Meet your micro:bit” exploration</b> .....	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
<b>3. Coding lessons</b> .....	18
Coding lesson menu.....	19
How to teach the coding lessons.....	20
Lesson 2: Sensory toy coding.....	22
Lesson 3: Simple door alarm coding.....	25
Lesson 4: Tilt alarm coding.....	28
<b>4. Vocabulary</b> .....	31



# 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 E](#).

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 E 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.

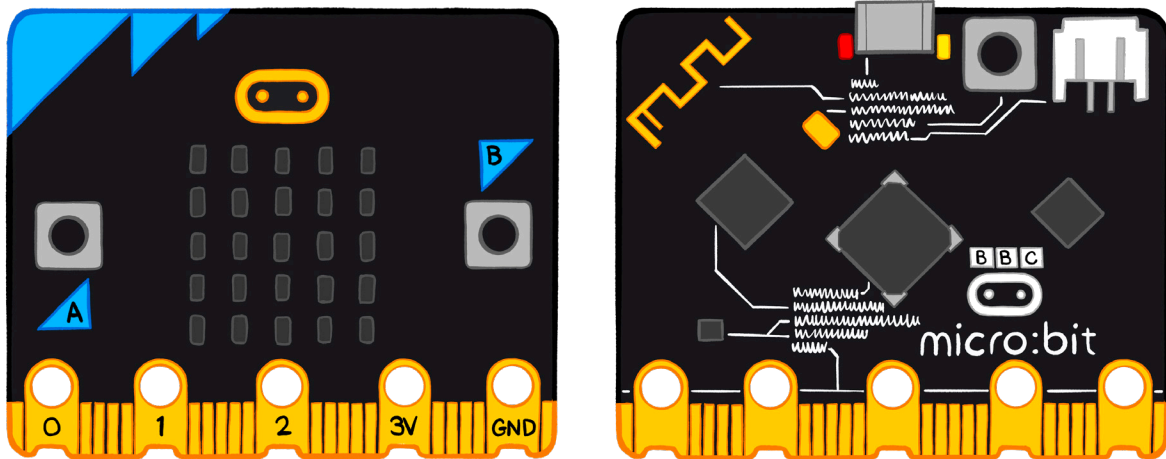
Computing topics from CS Fundamentals covered in these lessons include:

- **Event** – an action that causes something to happen
- **Impacts of computing** – understanding the good and bad effects computing can have on different groups of people, specifically:
  - **Designing for accessibility** – making design decisions that are driven by the needs of different groups of people
- **Conditional** – a statement that only runs under certain conditions
- **Function** – a piece of code that you can call over and over again

Note that **events** are not specifically covered in CS Fundamentals Course E, but are covered in Courses C and D.

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

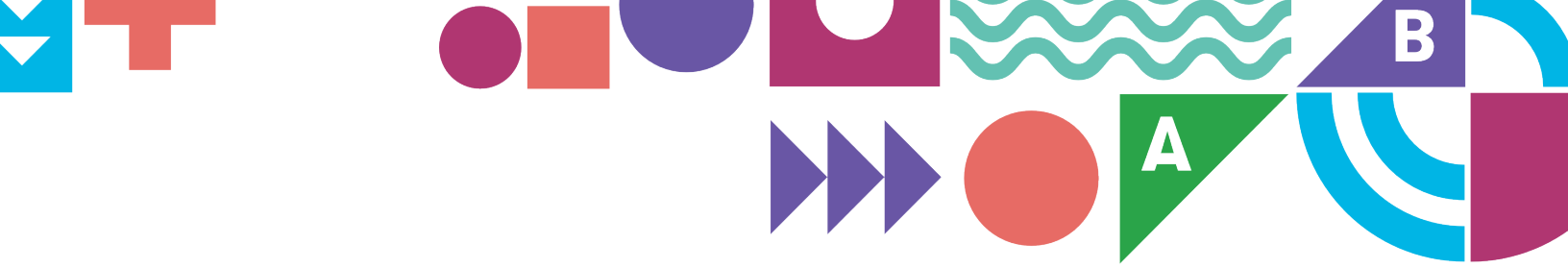


The BBC micro:bit is a tiny computer used by millions of children around the world. It's packed with **inputs** like buttons and sensors for light, movement, temperature, magnetism, and sound. It can also **output** pictures, numbers, and words on its LED display, make sound and music, and even communicate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



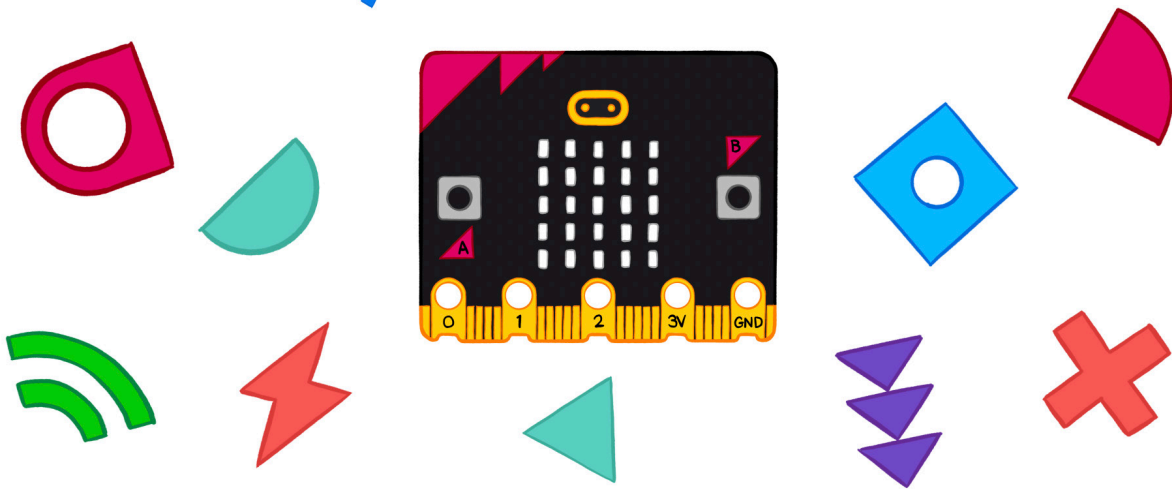
## Section 2

# “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:

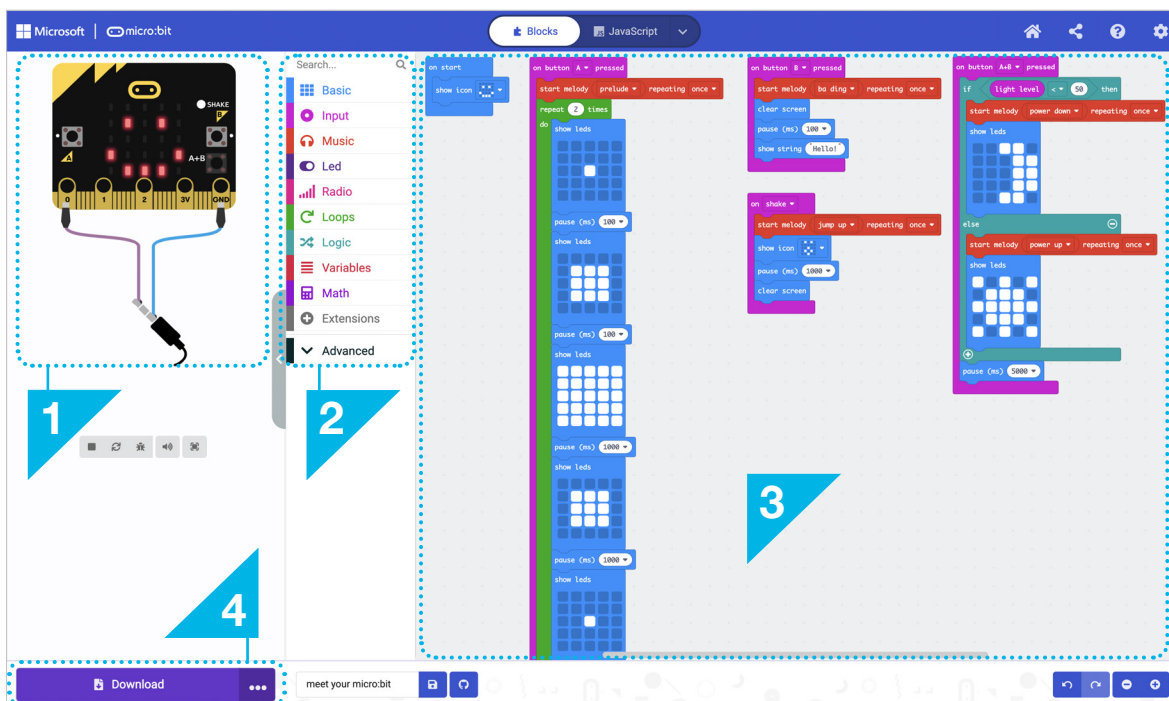
<https://mbit.io/csf-1-lesson-guide>

## 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>

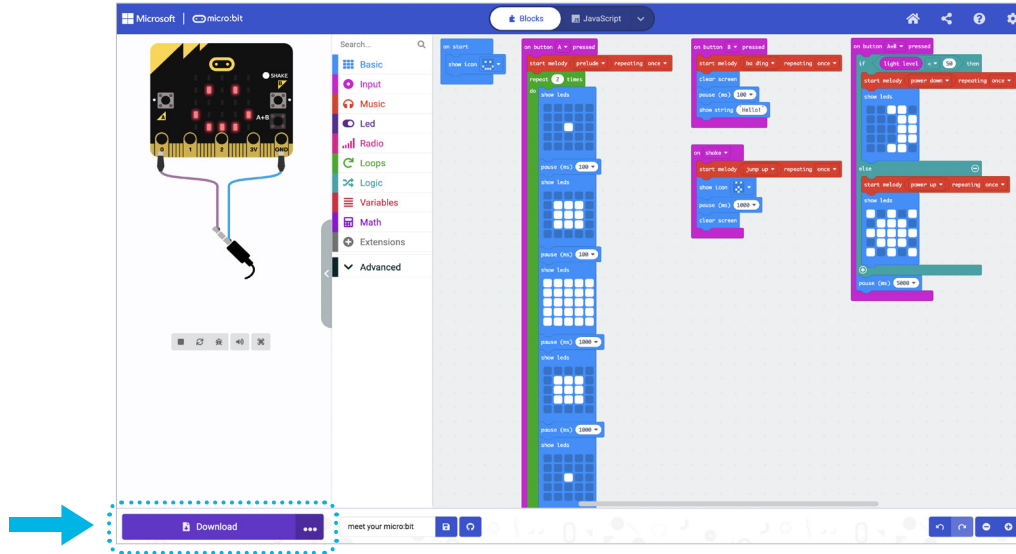


This is also a chance to familiarize yourself with the main parts of the MakeCode editor:

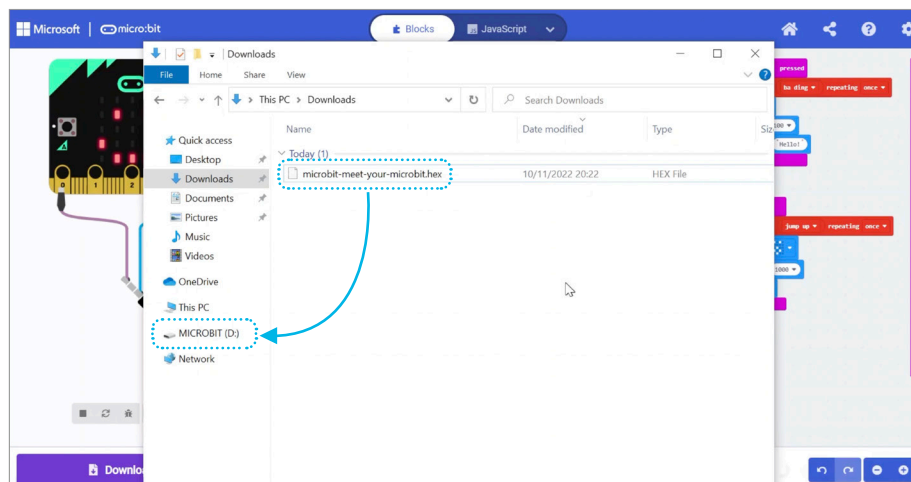
- 1 The **Simulator**, a virtual micro:bit that lets you demonstrate working code to your students, and lets your students test, debug, iterate, and improve their code before transferring it to their micro:bits. Click on button A to try it out.
- 2 The **Toolbox**, where you'll find the code blocks you 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	
<b>Introduction</b>	<p>Introduce the BBC micro:bit to your students:</p> <ul style="list-style-type: none"><li>• The micro:bit is a tiny computer you can program to make self-contained projects to do different things.</li><li>• For it to work, it first needs to be programmed to tell it what to do.</li><li>• 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?</li></ul>
<b>Events</b>	<p>The program on these micro:bits responds to different <b>events</b>—can your students work out what they are? (Pressing different buttons and shaking the micro:bit)</p>
<b>Inputs &amp; Outputs</b>	<p>Your students should consider what <b>inputs</b>—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 <b>outputs</b> 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!**

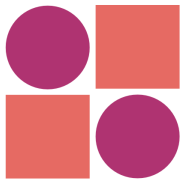
## Wrap up

### 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.

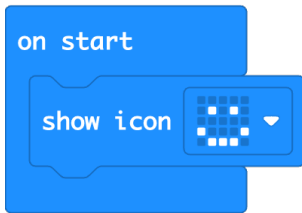
Talk about as many of the event blocks as are appropriate to the time available and your students' prior learning.



**Code Expert!**

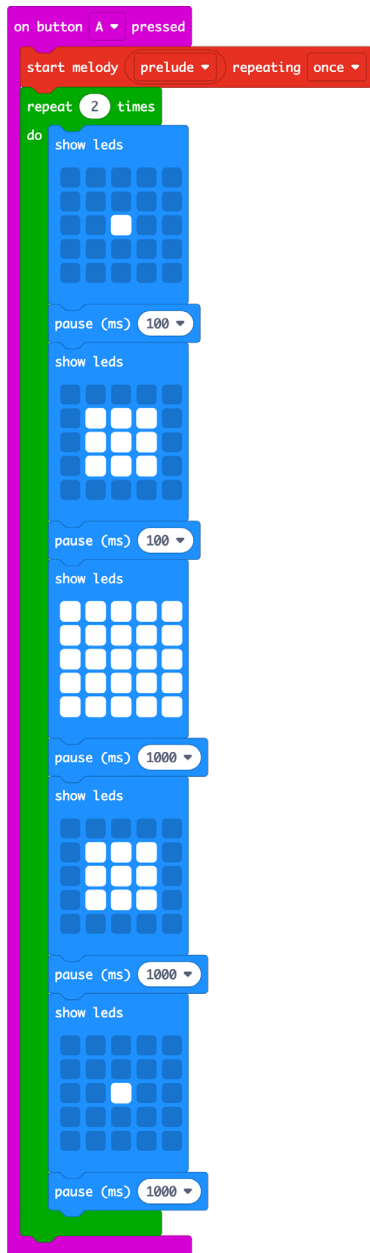


## CS talking points for code



### “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



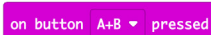
pause (ms) 100



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



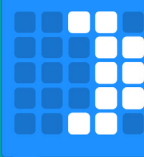
if light level < 50 then



start melody power down repeating once



show leds



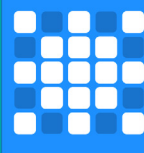
else



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.

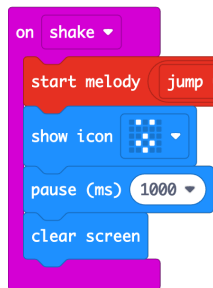
This part of the code shows different pictures on the display output, depending on the amount of light around you.

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.

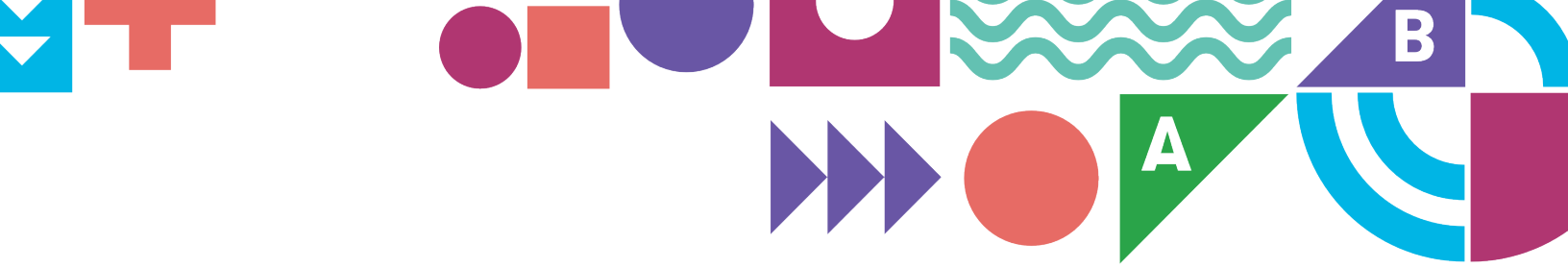
How might you use movement in your own projects?

## 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>



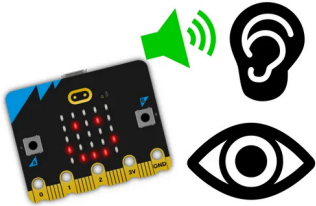
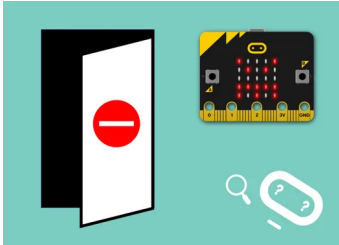
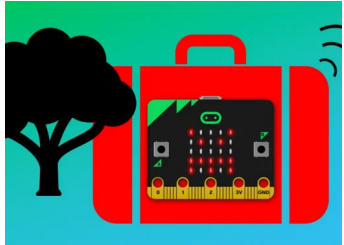
# Section 3

## 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 in sequence—you can pick as many or as few as you like.

	Beginner	Intermediate	Stretch
Title	<p><b>Lesson 2: Sensory toy</b></p> 	<p><b>Lesson 3: Simple door alarm</b></p> 	<p><b>Lesson 4: Tilt alarm</b></p> 
Description	Turn the micro:bit into a toy that responds to movement with sound and lights.	Use the micro:bit's compass and a magnet to tell if someone has been in your room.	Make an alarm to tell if an object has been moved using the micro:bit's radio and accelerometer; learn how functions can make code more compact and easier to read and modify.
Key Concepts	<ul style="list-style-type: none"> <li>• Events</li> <li>• Impacts of computing</li> <li>• Designing for accessibility</li> </ul>	<ul style="list-style-type: none"> <li>• Loops</li> <li>• Conditionals: if...then</li> </ul>	<ul style="list-style-type: none"> <li>• Functions</li> </ul>
CSTA Standards	<p><b>AP - Algorithms &amp; Programming</b></p> <p><b>1B-AP-10</b> - Create programs that include sequences, events, loops, and conditionals.</p> <p><b>IC - Impacts of Computing</b></p> <p><b>1B-IC-19</b> - Brainstorm ways to improve the accessibility and usability of technology products for the diverse needs and wants of users.</p>	<p><b>AP - Algorithms &amp; Programming</b></p> <p><b>1B-AP-10</b> - Create programs that include sequences, events, loops, and conditionals.</p>	<p><b>AP - Algorithms &amp; Programming</b></p> <p><b>1B-AP-11</b> - Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.</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	
<b>Explain the aim</b>	<b>Explain</b> the project aim
<b>Reinforce key learning</b>	<b>Reinforce</b> key learning relevant to CS Fundamentals and make connections with prior learning by either: <ul style="list-style-type: none"><li>• <b>Watching</b> an introduction video together</li><li>- or -</li><li>• <b>Exploring</b> 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"</li></ul>
<b>Examine the code</b>	<b>Examine</b> 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	
<b>Student coding</b>	Student coding activity. Pick whichever method suits your teaching style and students: <ul style="list-style-type: none"><li>• Step-by-step coding <b>videos</b></li><li>• A <b>micro:bit classroom</b> live coding session (see bottom of page 21)</li></ul>

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.

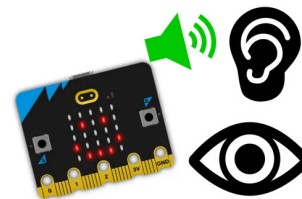
Key features of micro:bit classroom:

- 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: Sensory toy coding

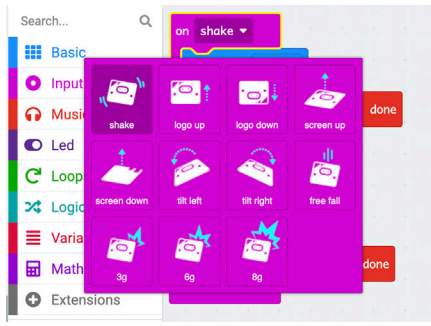
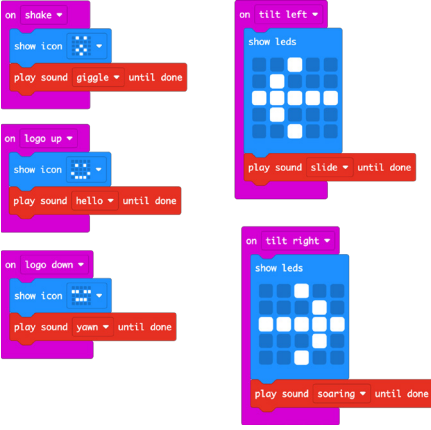
Level: Beginner



Warm up	
<b>Explain the aim</b>	Use the micro:bit's accelerometer to make a toy that reacts to different movements by making different sounds and showing different pictures on the LED display output.
<b>Reinforce key learning</b>	<p><b>Watch</b> the introduction video: <a href="https://mbit.io/csf-e2-intro">https://mbit.io/csf-e2-intro</a></p> <p>- or -</p> <p><b>Explore</b> the project, uploaded onto micro:bits prior to the lesson: <a href="https://mbit.io/csf-e2-project">https://mbit.io/csf-e2-project</a></p> <p>Discuss how a project like this could help learners who respond well to stimulation through touch, light, and sound.</p> <p>Discuss where these concepts are being used and where students may have used them before:</p> <ul style="list-style-type: none"><li>• <b>Events:</b> an action that causes something to happen (e.g., when you move the micro:bit in different ways, it outputs different pictures and sounds)</li><li>• <b>Impacts of computing:</b> understanding the good and bad effects computing can have on different groups of people (e.g., a toy can have a positive impact on users)</li><li>• <b>Designing for accessibility:</b> making design decisions that are driven by the needs of different groups of people (e.g., this toy uses simple movements rather than pressing small buttons, which some users may find difficult)</li></ul>

<b>Examine the code</b>	<p>Share the project working in the simulator with your students and look at the code together prior to students coding the main activity:  <a href="https://mbit.io/csf-e2-project">https://mbit.io/csf-e2-project</a></p> <p>Note that the code for this project uses the “play sound” block that works in the simulator and on the BBC micro:bit V2 with the built-in speaker. If you have the micro:bit V1, you can code this project without sound, or use the “start melody” block instead and connect headphones or an amplified speaker to pins 0 and GND.</p>
<b>Main activity</b>	
<b>Student coding</b>	<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"> <li>• Step-by-step coding <b>video</b>:  <a href="https://mbit.io/csf-e2-coding">https://mbit.io/csf-e2-coding</a></li> <li>• Live <b>micro:bit classroom</b> session:  <a href="https://mbit.io/csf-e2-classroom">https://mbit.io/csf-e2-classroom</a></li> </ul>
<b>Wrap up</b>	
<b>Discussion</b>	<p>Share student work, revisit key concepts used, and explore ideas for extended learning.</p>
<b>CS talking points for code</b>	
<b>Completed program for teachers</b>	<p><a href="https://mbit.io/csf-e2-project">https://mbit.io/csf-e2-project</a></p>



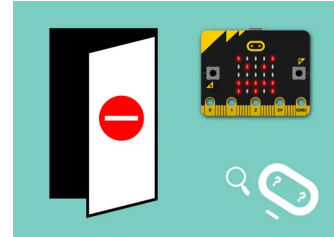
	<p>The program uses <b>input</b> blocks to sense different <b>events</b> using the micro:bit’s accelerometer sensor input to detect different movements or gestures.</p> <p>When you click on “shake” in the “on shake” block, it opens a dropdown menu with diagrams of the different movement events the micro:bit can react to.</p>
	<p>The events trigger different outputs: pictures on the LED display <b>output</b> and sounds on the speaker.</p>
<h3>Extended learning</h3>	
<p><b>Make animations</b></p>	<p>Use loops to create animations when the micro:bit is moved in different ways.</p>
<p><b>Touch sensing</b></p>	<p>Add more inputs such as touching the logo on the micro:bit V2, touching the pins, or making touchable buttons out of tin foil and attaching those to the pins.</p> <p>Learn more about touch pins here:  <a href="https://mbit.io/csf-pins">https://mbit.io/csf-pins</a></p>
<p><b>Find this project and more on microbit.org</b></p>	<p><a href="https://mbit.io/csf-e2-make">https://mbit.io/csf-e2-make</a></p>





# Lesson 3: Simple door alarm coding

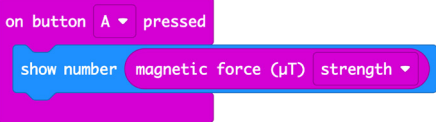
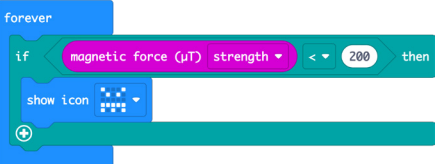
Level: Intermediate



Warm up	
<b>Explain the aim</b>	Create an alarm to show when someone has been in a room using a magnet and the micro:bit's compass as a magnetometer to measure the strength of magnetic fields.
<b>Reinforce key learning</b>	<p><b>Watch</b> the introduction video: <a href="https://mbit.io/csf-e3-intro">https://mbit.io/csf-e3-intro</a></p> <p>- or -</p> <p><b>Explore</b> the project, uploaded onto micro:bits prior to the lesson: <a href="https://mbit.io/csf-e3-project">https://mbit.io/csf-e3-project</a></p> <p>Instead of using real doors, you can model this project in class using a magnet and any box with a lid—for example, a shoe box.</p> <p>Discuss possible uses for an alarm like this. For example, making sure doors are kept closed to keep warm or cool air in a room.</p> <p>Discuss where these concepts are being used and where students may have used them before:</p> <ul style="list-style-type: none"><li>• <b>Loop:</b> the action of doing something over and over again (e.g., a “forever” block keeps the micro:bit checking if the magnet is near by measuring the strength of the magnetic field)</li><li>• <b>Conditional:</b> a statement that only runs under certain conditions (e.g., if the magnetic field strength falls below a certain level because the door has been opened and the magnet has moved away from the micro:bit, it shows an alert on the LED display)</li></ul>

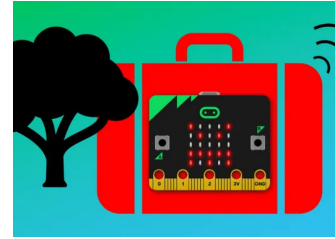
<p><b>Examine the code</b></p>	<p>Share the project working in the simulator with your students and look at the code together prior to students coding the main activity:  <a href="https://mbit.io/csf-e3-project">https://mbit.io/csf-e3-project</a></p> <p>Note that the value you use to trigger the alarm may be different depending on how strong your magnet is, so there are a few steps to making this project:</p> <ul style="list-style-type: none"> <li>• Coding</li> <li>• Transfer and test code on a real micro:bit: does moving the magnet away trigger the alarm? If not, press button A to measure the strength of your magnet when it's close and far away to decide what number to use</li> <li>• Modify the code to work with your magnet</li> <li>• Transfer the code and test again</li> </ul>
<p><b>Main activity</b></p>	
<p><b>Student coding</b></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"> <li>• Step-by-step coding <b>video</b>:  <a href="https://mbit.io/csf-e3-coding">https://mbit.io/csf-e3-coding</a></li> <li>• Live <b>micro:bit classroom</b> session:  <a href="https://mbit.io/csf-e3-classroom">https://mbit.io/csf-e3-classroom</a></li> </ul> <p>Test the code using a real micro:bit and a magnet. If the alarm doesn't work as expected, press button A to measure the strength of your magnet when it's close or far away, and use that number to replace 200 in the code, then transfer and test again.</p>
<p><b>Wrap up</b></p>	
<p><b>Discussion</b></p>	<p>Share student work, revisit key concepts used, and explore ideas for extended learning.</p>





CS talking points for code	
Completed program for teachers	<a href="https://mbit.io/csf-e3-project">https://mbit.io/csf-e3-project</a>
	<p>Button A is used to show a reading of the strength of the magnetic field sensed by the micro:bit's compass sensor <b>input</b>. It uses scientific units—micro Teslas—to measure how strong the magnetic field is. You can use button A to calibrate your code: if your magnet is weaker, it'll show a smaller number when it's close to your micro:bit. You can change the 200 number in the next block to match what you just measured.</p>
	<p>A “forever” <b>loop</b> block keeps checking how strong the magnetic field is.</p> <p>If the door is opened, the magnet moves away from the micro:bit and the magnetic field measured gets weaker. A <b>conditional</b> “if... then...” block shows an angry face on the LED display <b>output</b> if the magnetic field strength falls below 200 micro Teslas.</p>
Extended learning	
<b>Test different magnets</b>	Test the alarm with different magnets and modify the code according to how strong they are.
<b>Make some noise</b>	Add an audible alarm.
<b>Reset</b>	Make the alarm reset itself when the door is closed.
<b>Find this project and more on microbit.org</b>	<a href="https://mbit.io/csf-e3-make">https://mbit.io/csf-e3-make</a>

# Lesson 4: Tilt alarm coding

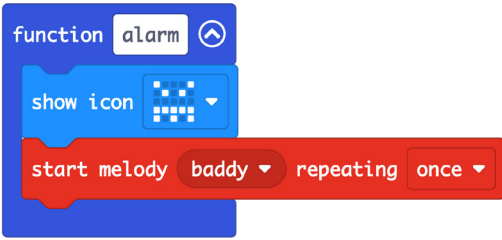
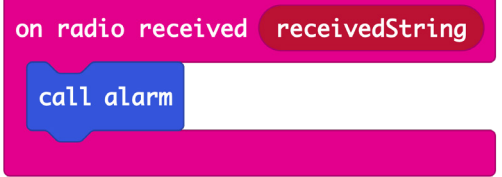
Level: Stretch



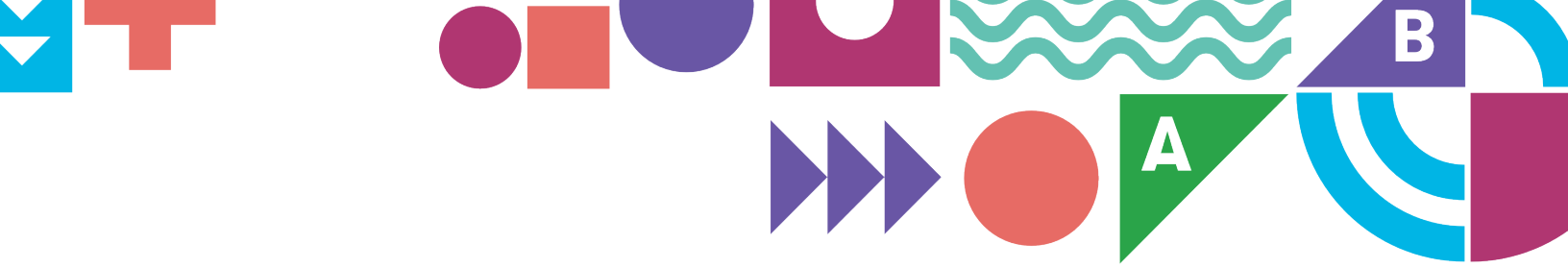
Warm up	
<b>Explain the aim</b>	<p>Use the micro:bit's accelerometer and radio features to make an alarm that warns you when something has been moved—and learn how functions can make code more compact and easier to read and modify.</p> <p>Note: this project works best with two students working in pairs, each with their own micro:bit (two micro:bits per pair of students).</p>
<b>Reinforce key learning</b>	<p><b>Watch</b> the introduction video: <a href="https://mbit.io/csf-e4-intro">https://mbit.io/csf-e4-intro</a></p> <p>- or -</p> <p><b>Explore</b> the project, uploaded onto micro:bits prior to the lesson: <a href="https://mbit.io/csf-e4-project">https://mbit.io/csf-e4-project</a></p> <p>Discuss where these concepts are being used and where students may have used them before:</p> <ul style="list-style-type: none"><li>• <b>Function:</b> A piece of code that you can call over and over again (e.g., the alarm showing an image on the LED display and playing an audible alert is in a function because it's used twice: when the micro:bit is moved and when it receives an alert message by radio)</li></ul>
<b>Examine the code</b>	<p>Share the project working in the simulator with your students and look at the code together prior to students coding the main activity: <a href="https://mbit.io/csf-e4-project">https://mbit.io/csf-e4-project</a></p> <p>Note: you will want to give each pair of students a unique radio group numbers so their alarms only communicate with each other and not the whole class.</p>

Main activity	
<b>Student coding</b>	<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"> <li>• Step-by-step coding <b>video</b>: <a href="https://mbit.io/csf-e4-coding">https://mbit.io/csf-e4-coding</a></li> <li>• Live micro:bit <b>classroom session</b>: <a href="https://mbit.io/csf-e4-classroom">https://mbit.io/csf-e4-classroom</a></li> </ul>
Wrap up	
<b>Discussion</b>	<p>Share student work, revisit key concepts used, and explore ideas for extended learning.</p>
CS talking points for code	
<b>Completed program for teachers</b>	<a href="https://mbit.io/csf-e4-project">https://mbit.io/csf-e4-project</a>
 <p>on start</p> <p>radio set group 1</p>	<p>At the start of the program, the code sets the radio group, a number between 0 and 255. Radio groups are like channels. Any micro:bits in the same group number can communicate with each other, so you will want to give each pair of students their own unique number.</p>
 <p>on shake ▾</p> <p>radio send string "thief!"</p> <p>call alarm</p>	<p>When the micro:bit's accelerometer detects movement (a shake gesture <b>event</b>), it sends a radio message and also calls the alarm function.</p>



	<p>The function called “alarm” shows an angry face icon and plays a warning tune when it is called. We put this code in a function because it’s used twice. This makes the code shorter, and also means if you modify it (for example, to add an animation or different sounds) you only have to change one piece of code.</p>
	<p>If the micro:bit receives a radio message from a nearby micro:bit in the same radio group, it also calls the alarm function. This means you can be alerted to movement of an object that’s out of sight.</p>
<b>Extended learning</b>	
<b>Change the alert</b>	<p>Edit the function to change the alert—for example, you could use animations, scrolling text, or different sounds.</p>
<b>Radio distance</b>	<p>Test out how far away you can get before the radio messages stop working.</p>
<b>Split the code</b>	<p>Split the code into two different programs—one just to send a radio signal when shaken which can be kept in the object you want to keep safe, and another that just receives messages and alerts you.</p>
<b>Find this project and more on microbit.org</b>	<p><a href="https://mbit.io/csf-e4-make">https://mbit.io/csf-e4-make</a></p>





# Section 4

## Vocabulary



# Vocabulary

Here are key computing terms from Code.org's CS Fundamentals Course E 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.
- **Event** – An action that causes something to happen.
- **Function** - A piece of code that you can call over and over again.
- **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. (This term is covered in CS Fundamentals Course F but is included here as inputs and outputs are key concepts in physical computing).
- **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.
- **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 E: <https://studio.code.org/s/coursee-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/>