Lesson Plan

Lesson 4: Step counter

Ages: 7 – 11

Programming language: MakeCode blocks

Topics: Variables (Programming), Input/output, Sensors (Computer systems)
Outcomes: Students turn their micro:bits into step counters (pedometers) using the micro:bit's built-in movement sensor, the accelerometer, and variables to keep track of how far they have walked.

Key learning in this lesson

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

Learning objectives

- I can turn my micro:bit into a step counter using the accelerometer and variables.
- I can explain that accelerometer is a sensor, an input that senses movement.
- I can explain that variables are containers for storing data which can be accessed and updated.

Preparation: before the lesson

What you need

- BBC micro:bits and micro-USB cables – at least one for every two students
- At least one computer (laptop or desktop) for every two students, with internet access to the Microsoft MakeCode editor: https://makecode.microbit.org/
  - Alternatively, you can use iPads with the micro:bit app installed. See our guide: https://mbit.io/lessons-mobile
- micro:bit battery packs – one per micro:bit
- Something to attach a micro:bit and battery pack to a leg, ankle or arm, e.g. elastic bands, pipe cleaners, wrist straps.
- PowerPoint presentation – whole class teaching slides
- Code blocks student handout (optional)

The lesson download also includes an optional '.hex' program file of the completed project, which may be useful if you have limited internet access. You can drag and drop this direct onto the MICROBIT drive when you connect a micro:bit to your computer. You can also drag '.hex' files into the MakeCode editor to examine the code and test it in the simulator.

Differentiation ideas

- If this is one of your first coding lessons with the micro:bit, it may be hard to know which students will need more support. You can use the extension ideas in the teaching section below for students who complete the task more quickly than others.

Decide how to deliver the 'Create' coding activity

You'll share the completed code on screen with your whole class from the slides. Additional options include:

- You (or selected students) model building and testing the code yourself on a large screen. The completed code is in the lesson plan and slide deck.
- Give students printed code blocks handouts to follow or cut out and assemble.
- Share a step-by-step YouTube coding video with the whole class, or individual students.

- If YouTube is blocked in your school, we also provide an animation in the slides showing how to assemble the code.
- Students can individually follow an online step-by-step tutorial.

- You can also choose to manage the whole class coding activity and save every student's code using **micro:bit classroom**. Find out more at https://mbit.io/lessons-classroom

Decide how to deliver the 'Evaluate' activity

Students download their code to real micro:bits and test the project.
You may want your students to answer the evaluation questions:

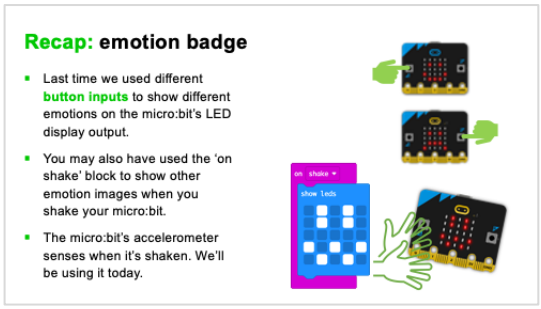
- on paper
- verbally with partners
- as part of a whole-class discussion.

Glossary

accelerometer:	a sensor that detects movement
data:	information collected for use elsewhere
input:	data sent to a computer for processing such as button presses and sensor readings
LED:	light emitting diode - the micro:bit display is made of 25 LEDs
output:	data sent from a computer such as words shown on the display
sensor:	an input that senses things in the real world, such as movement, temperature, and light levels
variable:	a container for storing data which can be accessed and updated while a program is running

Teaching: during the lesson

Recap prior learning (slide 2)

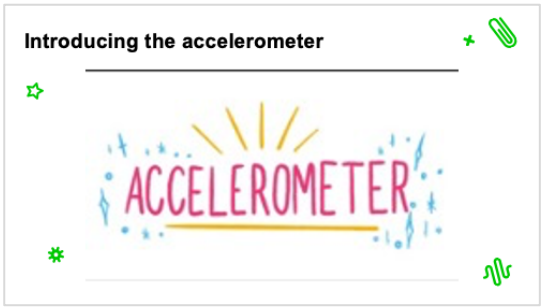


Ask your students what they discovered last time, for example:

- We used different button **inputs** to show different emotions on the micro:bit's LED display output.
- Some of us may also have used the 'on shake' block to show other images when you shake your micro:bit. Explain that the micro:bit's **accelerometer** senses when it's shaken. We're going to be using that today.

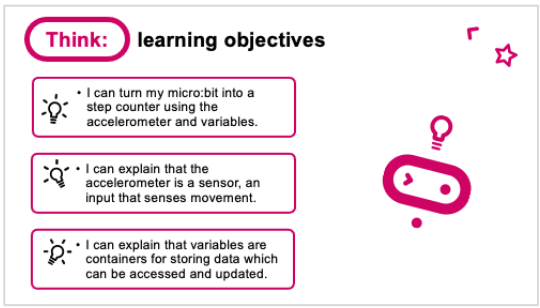
Think: starter activity

Introducing micro:bit's accelerometer (slide 3)



Optionally share a short, animated YouTube video introducing the accelerometer with your class: https://mbit.io/lessons-accel-video
Note that we'll be making a Rock, paper, scissors game using the accelerometer in lesson 6 in this series.

Learning objective (slide 4)



- I can turn my micro:bit into a step counter using the accelerometer and variables.
- I can explain that the accelerometer is a sensor, an input that senses movement.
- I can explain that variables are containers for storing data which can be accessed and updated.

Explain that the **accelerometer** is a **sensor input** that senses when you shake your micro:bit. Students may have already used accelerometers to count steps using phones, watches, or fitness trackers.

The accelerometer is labelled on the back of the micro:bit, so you can see where it is – even though it's tiny, it contains moving parts that react to movement.

Variables help us count how many steps we've taken. They are containers for storing data which can be accessed and updated while a program is running.

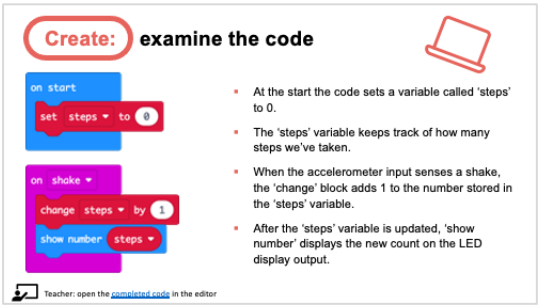
Step counter introduction video (slide 5)

Optionally play the project introduction video: https://mbit.io/lessons-step-intro-video

Create: coding activity

Examine code with students (slide 6)

Explain:

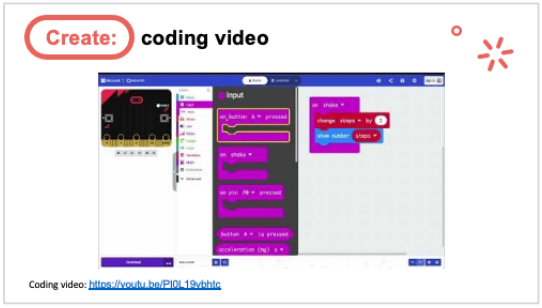


- At the start the code sets a variable called 'steps' to 0. (It's a good idea to set variables to a known value at the start of a computer program.)
- The 'steps' variable keeps track of how many steps we've taken.
- When the accelerometer input senses a shake, the 'change' block adds 1 to the number stored in the 'steps' variable.
- After the 'steps' variable is updated, 'show number' displays the new count on the LED display output. (The order of blocks is important – the code must show the 'steps' variable number on the LED display **after** it's been updated, or the count will be out-of-date!)

You can also follow the link on slide 6 to open the completed code in the editor and model testing it in the simulator: https://mbit.io/lessons-step-code

Model building the code (slides 7-10)

- You can open a new MakeCode project from slide 7 and model building the code.
- Optionally share the YouTube coding video on slide 8 with your class: https://mbit.io/lessons-step-code-video
- Or share the coding animation on slide 9 if YouTube is blocked in your school.

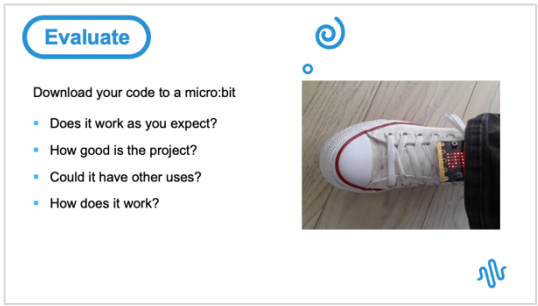


Students recreate the code, testing it in the simulator. They can either:

- Copy the code from slide 6.
- Follow printed code blocks handouts.
- Individually follow a step-by-step online tutorial: https://mbit.io/tutorial-step-counter - you can share the link from slide 10.
- If you are using micro:bit classroom, start a new session and ask your students to join your lesson. You can also open a session with completed code to edit and share with students: https://mbit.io/lessons-step-classroom

Evaluate: (slide 11)

Students transfer code to their micro:bit and test.

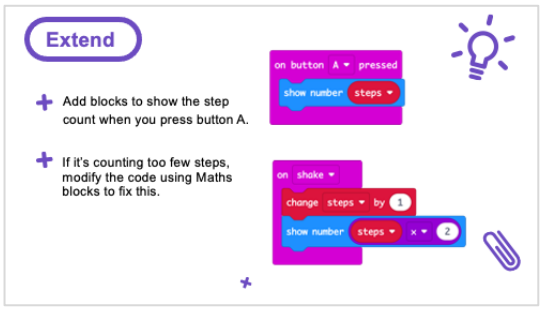


Questions:
Students transfer code to their micro:bit, attach a battery pack, fix the micro:bit to their leg or ankle and test.
Questions:

- Does it work as you expect?
  - If not, do you need to debug the code and download it again?
- How good is the project?
  - Can you think of anyone who would like this project and find it useful of enjoyable?
  - How could you improve it?

  - Students may find the display hard to read, that they have to stamp their legs to record steps, or that it counts fewer steps than they expected. See ‘Extend’ below for some ideas for improving the project.
- Could it have other uses?
- How does it work?
  - Encourage students to think about how it works when using it.

Extend: (optional, slide 12)



- Add an ‘on button A pressed’ block to show the step count when you press button A – this means you can read the step count at any time without having to shake the micro:bit.
- Assess how accurate your step counter is. If the step counter is attached to one leg, it may be counting **strides** rather than steps, counting every **other** step. Students could compensate for this by using Maths blocks to multiply the ‘step’ variable by 2 when it’s shown on the LED display.

Share: revisit learning objectives (slide 13)

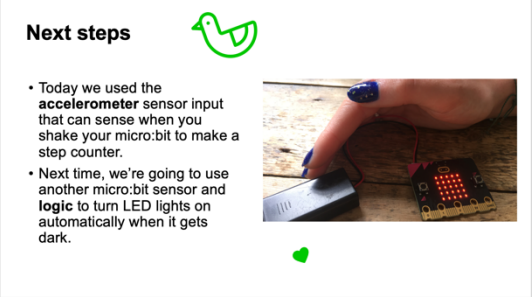
- I can turn my micro:bit into a step counter using the accelerometer and variables.
- I can explain that accelerometer is a sensor, an input that senses movement.
- I can explain that variables are containers for storing data which can be accessed and updated.

Ask:

- How does your step counter work? How accurate is it? (This will vary. It could depend on where or how micro:bits were attached. Students may also discover that the step counter counts every other step, or how many strides they took, rather than steps.)
- How could you improve it? (e.g. multiply the step count by 2, add a button press to show the step count)
- Why do we use variables in the code? (So the micro:bit keeps track of how many steps we took.)
- What job does the accelerometer do in a step counter? (It senses movement)

Next steps (slide 14)

Today we used the **accelerometer** sensor input that can sense when you shake your micro:bit to make a step counter.
Next time, we're going to use another micro:bit sensor and **logic** to turn lights on automatically when it gets dark.



Assessment: after the lesson

When assessing students' work you may find it helpful to ask these questions:

How well did the student follow instructions to code and test a micro:bit step counter?

Can they explain what variables are and why they're used in the step counter code?

What is their understanding of accelerometers and other sensors?

Here are some guiding criteria you might want to include when assessing your students' work:

WORKING TOWARDS the learning objective

The student was able to create a step counter in the editor with assistance, or independently create a program that reacts to movement using the 'on shake' block by showing something on the LED display but may not have transferred it to a micro:bit or tested it.

They can explain that variables hold numbers but may not be able to explain that the numbers change or understand that 'steps' in the code is a variable.

They can explain that the micro:bit reacts to movement, but cannot identify the accelerometer as the part of the micro:bit that does this.

MEETING the learning objective

The student created code to use a variable to count how many times the micro:bit is shaken, transferred the code to a micro:bit and tested it.

They can describe variables as containers that hold data (information) which can be updated and that the code uses a variable so it can keep track of how many steps you have taken.

They can explain that accelerometers sense movement.

EXCEEDING the learning objective

The student created code to use a variable to count how many times the micro:bit is shaken, transferred the code to a micro:bit, tested it to assess its accuracy and suggested ideas or made improvements to the code to make it more accurate.

They can describe variables as containers that hold data (information) which can be updated and that the code uses a variable so it can keep track of how many steps you have taken. They may also explain that counter variables should be set back to zero at the start of a program and should have descriptive names to make the code easier to read.

They can explain that accelerometers are sensors that detect movement and give examples of other devices that use them to count steps and measure other kinds of movement, for example in cars and rockets.