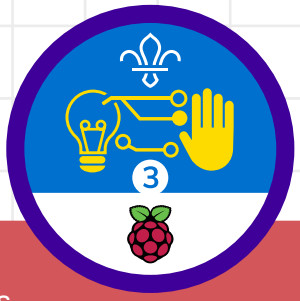
micro:bit step tracker



Overview

This activity fulfills **Stage 3, Requirements 2 and 3** of the **Digital Maker Staged Activity Badge** ('Use a programmable device (such as Arduino, Raspberry Pi, or micro:bit) with electronic components, code, and appropriate materials to create an electronic gadget and use it in a Scouting activity'; 'Design and create digital graphics for use as part of one of the above'). Young people will design pixel art icons and write a Python program to create a digital step tracker (pedometer) using a micro:bit.

30 – 60 minutes

Flexible (ideally two young people per computer)

If you're running this activity without access to WiFi, you will need to download the software ahead of time. You may also wish to print handouts.

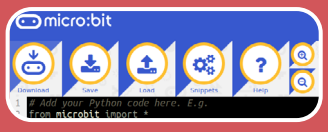
Wherever you have access to computers.

You will need:

- Laptops or desktop computers; tablets or other mobile devices may potentially work as well
- Activity handouts (at least one per group)
- Mu code editor software (an online Python code editor is available, but Mu is preferred)
- Per pair:
  - One micro:bit
  - One micro:bit battery pack
  - One USB A to micro USB B cable

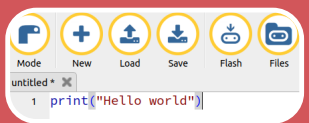
If your meeting place doesn't have WiFi

Download and install the Mu editor ahead of time; you can find instructions at rpf.io/scouts-microbit.



If you want to use the online editor

Open the page python.microbit.org. Young people can write their program in this editor and then download it onto their computers.



Key messages

- The micro:bit has a sensor called an accelerometer, which measures acceleration. This means that it can detect movement, such as walking. It's the same type of sensor that lets your smartphone know which way round you're holding it!
- The micro:bit has a 5×5 grid of individually controllable LED lights on which you can display messages and icons.
- With just a few lines of code, you can demonstrate the principle of how a pedometer works.
- An idea for a project can be prototyped and tested with a small amount of code, and can be developed into a more advanced form.
- Python is a simple text-based programming language that you can use to make a computer carry out instructions.

Scouts Raspberry Pi

micro:bit step tracker

Leader instructions

1. If they have not used them before, introduce the young people to the micro:bit. Explain that micro:bits not only have two buttons (for input) and a small LED display (for output), they also have sensors. Explain the purpose of the sensors, particularly the accelerometer.

2. Explain that the Scouts' challenge is to program a micro:bit so they can use it as a step tracker, a device that can help them measure how far they walk (e.g. per day or on a Scouting trip).

3. Get the young people to connect the micro:bit to the computer with the USB cable, and to open Mu. Help them get started by showing how and where to write the "Hello world" program described at the beginning of the Scouts' instructions, and how to copy (flash) the code to the micro:bit.

4. Let them work through the rest of the instructions. Things to watch out for:
   a. Errors in code can be frustrating — the **Check** button will guide them to what the error is.
   b. Just saving the code file does nothing to the micro:bit — the young people need to click on the **Flash** button each time to transfer the updated code.

5. Close the session with a discussion about what makes a good step tracker, talk about problem solving with the help of computers, and discuss the potential uses for digital devices with sensors in outdoor activities and the daily life of a Scout.

Alternatives

- Instead of writing Python code in the Mu app or the online editor, young people can use the micro:bit MakeCode editor (makecode.microbit.org). This allows them to create their program either with code blocks, or with JavaScript code.

Adaptability

- The simple implementation of the step tracker ('add 1 when shaken') is not very effective as a true pedometer, but it is a good starting point and proof of concept. Some young people may choose to make the more advanced version (as described in the Extension section of the Scouts' instructions), but **this not a requirement.** As long as young people have created a digital asset (a 5×5 pixel icon) and used code to make the proof-of-concept pedometer, they have done enough to complete this activity.
- Try pairing up Scouts of different programming experience levels, without allowing one to take over.
- Depending on their experience, the young people may need extra guidance while writing code to solve the problem. Refer to the code snippets in the instructions, and any technical helpers with coding experience can also lend support.

Community and sharing

Young people should have the opportunity to explain their code, or to explain how they found or fixed a bug, to the rest of the group.

Safety

If the young people are working online, tell them to ask for permission before viewing any other websites. It's a good idea to set up parental controls — you can find instructions for this on the NSPCC website **(rpf.io/scouts-nspcc-online).**
You should also give each young person a Stay Safe leaflet (rpf.io/scouts-staysafe).

micro:bit step tracker

In this activity, you will design pixel art icons and write a Python program to create a digital step tracker (pedometer) using a micro:bit.

Part 1: Test your micro:bit

1. Connect your micro:bit directly to your laptop using the micro USB cable.
2. Open **Mu** and clear the screen by deleting the code.
3. Write some simple Python code to get started:

```
from microbit import *

display.scroll("Hello world")
```

4. Click the **Check** button to check your code for mistakes. Fix any mistakes you find. Look carefully! You should always check your code before sending it to the micro:bit.
5. Click the **Flash** button to send the code to the micro:bit. You should see the message 'Hello world' scroll across the 5×5 LED display.

Trivia

Did you know that the idea of a step tracker, or pedometer, is very old? In the 15th century, Leonardo da Vinci envisioned a mechanical pedometer, and in 1780, Abraham-Louis Perrelet from Switzerland created the first pedometer that could count steps and measure the distance walked.

Trivia

Did you know that 10,000 steps is supposed to be the ideal daily distance to walk? In 1985, Japanese researchers ran experiments suggesting that people who eat a normal amount of food need to take 10,000 steps every day to maintain a healthy body.

micro:bit step tracker

Part 2: A simple step tracker

1 To count steps, you'll use the micro:bit's accelerometer to detect movement. Change the code you have so it looks like this:

```
from microbit import *

while True:
    if accelerometer.was_gesture('shake'):
        display.scroll('shake')
```

2 Send the code to the micro:bit by clicking the **Flash** button again, and test the code by shaking the micro:bit (not your computer!). You should see the word 'shake' scroll across the micro:bit display every time you shake it.

3 In order to count the number of steps, you will use a **variable** to store the number of shakes. Call the variable 'steps' so it's easy to remember what it is for. Your program should increment (add 1 to) the steps variable each time the micro:bit detects movement, and then scroll the step count across the display:

```
steps = 0

while True:
    if accelerometer.was_gesture('shake'):
        steps += 1
        display.scroll(steps)
```

4 Send the code to the micro:bit again by clicking the **Flash** button, and then shake the micro:bit to check whether it shows the number of 'steps'.

Part 3: Design your icons

As well as scrolling text across the micro:bit's display, you can make it show pixel art and animations. Here, you'll create an icon or animation to be displayed whenever you reach a step number target.

1 The micro:bit software had some ready-made examples of icons you can use. Display one at the start of your program to try it out:

```
from microbit import *

display.show(Image.HEART)
```

2 Send your code to the micro:bit with the **Flash** button. If you don't see an icon, make sure the line `display.show(Image.HEART)` is **directly below** the line `from microbit import *`.

3 Try out some more icons by replacing `HEART` with any one of these options: `HAPPY`, `SAD`, `RABBIT`, `DUCK`, `STICKFIGURE`, `TSHIRT`, `SNAKE`

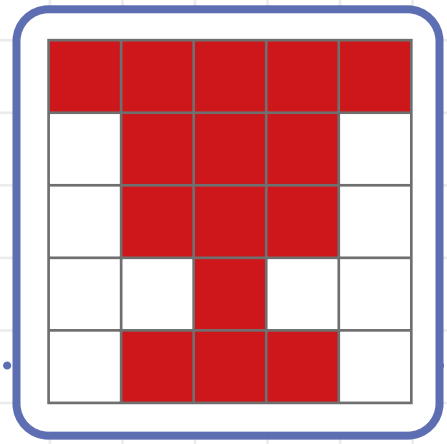
4 To make the icon show up after a certain number of steps, add another if statement to check the number of steps. 10 is a good number for testing purposes, but for the real pedometer you'll probably want to use a higher number.

```
while True:
    if accelerometer.was_gesture('shake'):
        steps += 1
        display.scroll(steps)
    if steps == 10:
        display.show(Image.HEART)
```

micro:bit step tracker

5 Click **Flash** and test the new code: shake the micro:bit repeatedly, and you should see the icon once you've reached 10 steps. The micro:bit should continue counting up beyond 10 too.

6 Now draw your own icon using the paper templates provided. Colour in a selection of squares in the grid, for example like so:



7 Next, move your mouse pointer to below the line of code saying `steps = 0`. Here, write the code for the icon you've designed, using the following format. Each number represents the brightness. 9 is full brightness, 0 is off.

```
icon = Image('99999:'
             '09990:'
             '09990:'
             '00900:'
             '09990:')
```

Tip

Use spaces to align your code.

8 Change the last line of your program to show your icon instead of the one of the ready-made examples:

```
display.show(icon)
```

9 Try adding a second milestone (20 steps, for testing purposes), so it shows the `HEART` at 10 steps, and your new `icon` at 20 steps:

```
while True:
    if accelerometer.was_gesture('shake'):
        steps += 1
        display.scroll(steps)
    if steps == 10:
        display.show(Image.HEART)
    if steps == 20:
        display.show(icon)
```

10 If everything is working, it's time to try your pedometer out. You might want to exchange your milestones 10 and 20 for slightly larger numbers for this! Detach the micro:bit from your computer and connect it to a battery pack. Attach the micro:bit and battery to your wrist and test the effectiveness of the step tracker.

Extension (optional) Part 4: Refine your code

Now that you have some simple accelerometer code and some pixel art, it's time to put them together to make a fully featured pedometer.

1. Reconnect the micro:bit to your computer.

2. In Mu, open a new file by clicking the + sign labelled **New**.

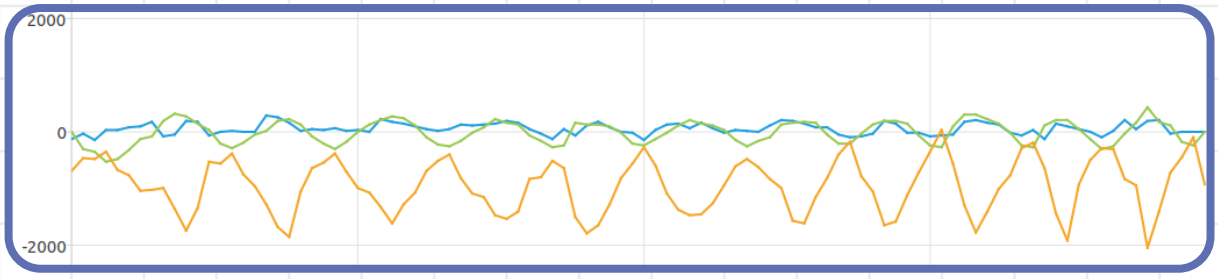
3. Add the following code:

```
from microbit import *
from time import sleep

while True:
    x = accelerometer.get_x()
    y = accelerometer.get_y()
    z = accelerometer.get_z()
    print((x, y, z))
    sleep(0.1)
```

4. Flash the code to the micro:bit and click the **REPL** button in Mu so you can see the printed values. Also click the **Plotter** button, and you'll see a graph of the changing accelerometer values in real time. Leave the micro:bit still on the desk: you should see the x and y values staying at around 0, and the z value staying at around 2000 or -2000. This is the effect of the force of gravity!

5. Slowly shake the micro:bit and see what happens to the graph:

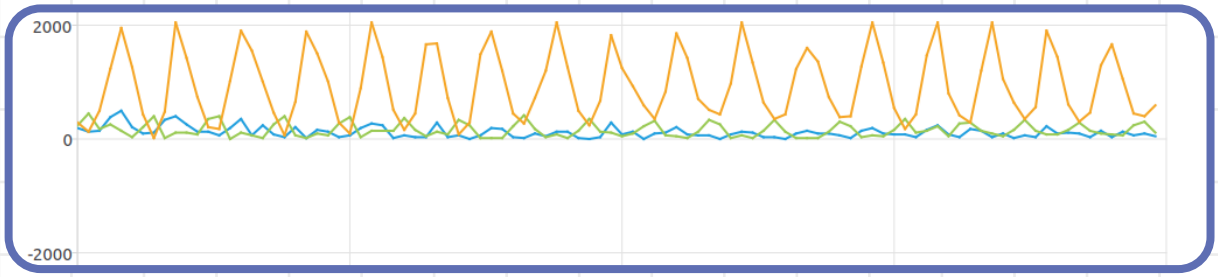


6

You can ignore negative values by using the `abs()` function to get the absolute value of an accelerometer measurement, e.g:

```
x = abs(accelerometer.get_x())
```

If you do this for `x`, `y`, and `z`, and then test your code again, your graph should look something like this:



7

If you have a long enough USB cable, try to simulate a walking motion with the micro:bit to see what pattern the plotter makes. Can you work out a better method of counting steps than using the 'shake' method? You could see what the following `while True` statement results in:

```
while True:
    x = abs(accelerometer.get_x())
    y = abs(accelerometer.get_y())
    z = abs(accelerometer.get_z())
    if x > 256 or y > 256:
        steps += 1
        display.scroll(steps)
```

Try changing the numbers to see whether your step tracker becomes more accurate.

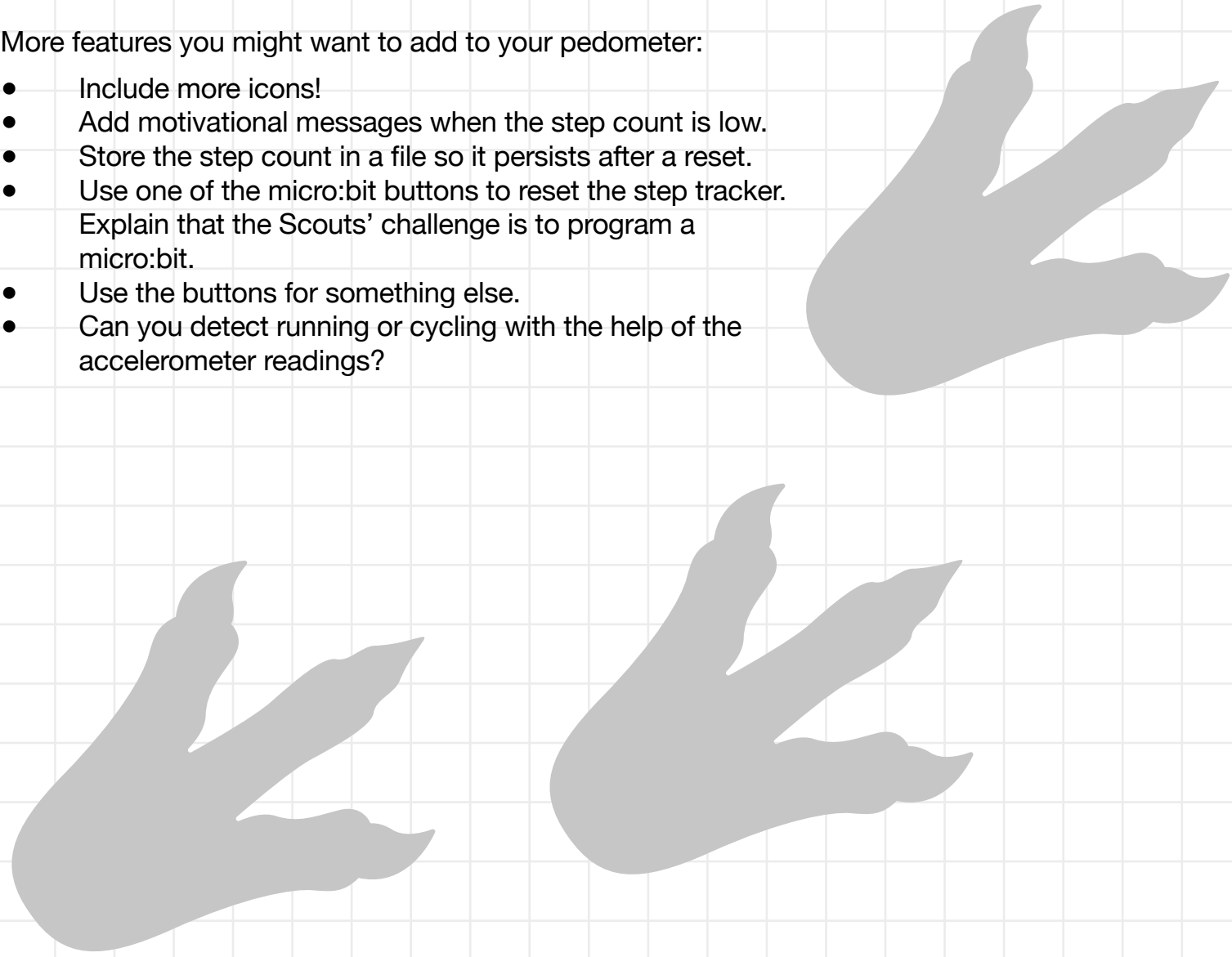
8 When you're happy with your step tracker code, add in your icons so they are displayed at certain step milestones.

9 If you want to further improve your pedometer, try some of the suggestions below.

What next?

More features you might want to add to your pedometer:

- Include more icons!
- Add motivational messages when the step count is low.
- Store the step count in a file so it persists after a reset.
- Use one of the micro:bit buttons to reset the step tracker. Explain that the Scouts' challenge is to program a micro:bit.
- Use the buttons for something else.
- Can you detect running or cycling with the help of the accelerometer readings?



Blank paper template