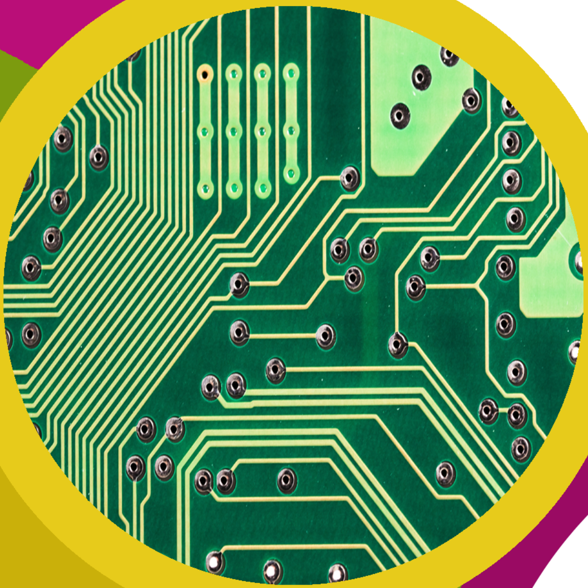
technocamps

Micro:bit Morse Code Teacher Guidance



UNDEB EWROPEAIDD
EUROPEAN UNION
Llywodraeth Cymru
Welsh Government
Cronfa Gymdeithasol Ewrop
European Social Fund

Prifysgol Abertawe Swansea University
Cardiff University Prifysgol Caerdydd
Prifysgol Bangor University
Cardiff Metropolitan University Prifysgol Metropolitan Caerdydd
i.t.wales
Prifysgol Aberystwyth University
Prifysgol Glyndŵr Wrecsam | Wrexham Glyndŵr University
University of South Wales Prifysgol De Cymru

Curriculum Links

Links to Science and Technology AoLE

Computation:

(PS2) I can create simple algorithms and am beginning to explain errors.
(PS2) I can follow algorithms to determine their purpose and predict outcomes.
(PS2) I can explore how different component parts work together.
(PS2) I am beginning to explain the importance of accurate and reliable data to ensure a desired outcome.

(PS3) I can use conditional statements to add control and decision-making to algorithms.

Links to Other AoLEs

Languages, Literacy, and Communication:

(PS2) I can listen to, understand, and communicate the general meaning of what I hear.
(PS2) I can infer meaning from text and images

(PS3) I can compare different things I read.

Mathematics and Numeracy:

(PS2) I have explored patterns of numbers and shape. I can recognise, copy, and generate sequences of numbers and visual patterns.

Learning Rationale

The Four Purposes and Cross-Curricular Skills

This resource provides **Critical Thinking and Problem-Solving** opportunities. Students are required to follow and design an algorithm using block-based programming. They are able to analyse errors in the code, identify solutions, and deduce the next steps in the code.

Learners also use **Creativity and Innovation**. They are encouraged to discuss and implement strategies to improve their program. They are also able to create and send visual patterns in Morse code using their program, and then translate the message.

The **Interacting and Collaborating** and **Data and Computational Thinking** sections of the **DCF** apply to this resource. Students will learn to code efficiently using loops and events to create an algorithm that takes user input to display an output. Learners will work together, communicate using patterns, and translate messages using their programmed micro:bits.

Why Is Learning This Important?

This resource provides learners with the opportunity to create simple algorithms with a demonstrable application, using a block-based programming language. It introduces concepts such as conditionals, loops, and event-based programming which are critical to most common programming languages. The resource also teaches the importance of long-distance communication and allows for collaborative and interactive activities to showcase this. This can be expanded to introduce students to text-based programming such as Python.

Suggested Approaches Key

In this suggested approach we use the following colours to differentiate the types of activities:

- **Yellow - Explain.** Teachers should explain the slide/example to the class.
- **Green - Discuss.** Teachers should start an open discussion with the class to get them to feedback some answers/ideas.
- **Purple - Activity.** Students are expected to complete an activity whether it be in their workbooks or on the computer, followed by a discussion of their solutions.
- **Green - Introduction/Conclusion.** The introduction/conclusion is also colour coded green. Teachers should hand out materials in the introduction and conclude the session and collect materials at the end.

Introduction

Begin with introductions, and a brief explanation of the Technocamps programme, before handing out any resources required by learners and any additional aids for learners with additional learning needs.

Explain: Topics Covered Today

We will be learning how to use micro:bits and block-based programming to create radio devices that can communicate with each other.

Discuss: Micro:bit

Provide each student with a micro:bit and ask them what they know about it. Have they used them before? Can they tell you what some of the components are (e.g. buttons, LED lights, USB connector)?

Explain: What is micro:bit?

The micro:bit is a very small computer that is used to teach how hardware and software work together.
It has several components: 25 Led lights that can be used to display images, sensors that can detect light/temperature/movement, buttons, and radio and bluetooth antenna.
We can program the micro:bit to take input, display output, process information, communicate with other micro:bits and many more things.

Explain: What is programming?

Programming is telling a computer what to do using a set of ordered instructions. The set of ordered instructions is called an **algorithm**. The language used to tell the computer what to do is called a **programming language**.
Introduce the students to the MakeCode editor and explain how to connect their micro:bit devices.

Display Output

Explain: Downloading Programs

Each time you make changes to your program you need to click on the download button before the micro:bit can run the program.

Activity: Smile

Program the micro:bit to display a happy face. Drag the **show icon** block into the **forever** loop and choose the icon from the dropdown menu.

Explain: LED lights

The **show icon** blocks allow us to display default images on the micro:bit. Using the **show leds** block, we can customise the image that is displayed.

Activity: Custom Icons

Ask the students to create their own icons using the **show leds** block. Clicking on each individual square turns that light on or off.

Loops

Explain: Loops

Loops can be used to repeat commands in a program without typing out each action every time. They can be repeated forever, for a certain number of times, or for a given condition.
In MakeCode, these are found in the **Loops** section.
For a loop to have a purpose, an action command needs to be placed inside it.
All MakeCode programs start with a default **forever** loop. This loop will run a set of commands until the micro:bit is unplugged or reset. You can only have one **forever** loop in a micro:bit code.

Activity: Changing Faces

Ask the students to create a program that displays two alternating images - a happy face and a sad face. They can do this in the default **forever** loop.
Suggest that they add a **pause** command between the two icons.

Discuss: Forever vs. On Start

Explain the difference between the **on start** block and the **forever** loop. Ask the students what they think would happen if they put the code from the 'changing faces' activity in the **on start** block instead of the **forever** block.
Answer: The code would run once and then stop without repeating.

Activity: Custom Animation

Ask the students to create their own animation using the **forever** block and the **show leds** command. They can use as many different images as they want to create the animation. Remind them to click download when they are done.

Explain: Events

Each micro:bit has two buttons: A on the left and B on the right.
These buttons allow us to choose which action to take without reprogramming the micro:bit each time.
For example, we can show a happy face when we press button A and a sad face when we press button B.
These commands are found in the **input** section.

Activity: Events

Create a program that displays one image when button A is pressed and a different image when button B is pressed.

Radio

Discuss: Radio Communication

Ask the students what they know about radio communication. What devices can use radio waves to communicate with each other?
- Broadcast radio and TV, mobile phones (bluetooth), air traffic control, medical scanners, remote controlled toys, and… micro:bits.

Explain: Micro:bit Radio

Micro:bits have a Bluetooth feature that allows them to communicate with each other using high frequency radio waves.
We can use this to send messages and images between two (or more) micro:bits.
The blocks that are needed are found in the **radio** section in MakeCode.
For micro:bit devices to be able to send and receive message to each other, they need to be in the same radio group. This can be anything from 1 to 255.

Activity: Radio Groups

Ask the students to get into pairs. Each person must set their micro:bit to the same radio group as their partner. However, this must be different to all other groups in the room.
Get the pairs to call out which group number they are and make sure there are no duplicates.

Sending Messages

Explain: Sending Messages

There are two ways of sending information between micro:bits. You can send a number, or you can send a string.
We can use events to choose when to send the information.

In this case we can add the **radio send string** command to an **on button A pressed** block. Choose a text string to send, for example "smile".

Activity: Sending Messages

Ask the students to recreate the code on the slide. Does the micro:bit do anything when you run the code now?

Discuss: Sending Messages

What do they think the next step will be?
Hint: How will the other micro:bit know what to do with the received message? Tell the students to look in the **radio** section for possible commands to use.

Receiving Messages

Explain: Receiving Messages

Now that the micro:bit has sent a message, the other micro:bit needs to be told what to do with that message when it is received.

Since we sent a text string, we need to use the **on radio received receivedString** block to do this.
Inside the block, drag and drop the **show icon** command and choose the happy face from the drop down menu.
Add a **clear screen** command after the **show icon**.

Activity: Sending Messages

Ask the students to recreate the code on the slide. Does the radio communication work now?

Discuss: Sending Messages

Now we have a working radio communication device. However, we can only send one image. How can we improve this code to have an option to send a second image instead? Hint: We used button A to send the smile string.

Explain: Sending Two Messages

We can use the buttons on the micro:bit to choose between two messages to send.

Add a second **on button pressed** event block and change the button to button B. Insert a **radio send string** command into the second block and change the string to “frown”.

This gives us the choice to send either the text string “smile” or the text string “frown”.

Discuss: Problem Solving

Try running the code and see what happens. Can you identify the problem? How do you think we can solve this?

This may be a difficult question to answer with a class that is inexperienced with coding - the solution is to use conditional statements.

Explain: Logic and If-statements

Conditions help us perform different actions based on different conditions. These conditions should always result in Yes/No or True/False.

For example:
If my homework is done, **then** I can go outside to play.
If I have eaten my dinner, **then** I can have dessert.
Ask the students for more examples.

In MakeCode, the if-statements are found in the **Logic** section.

Using the **if true then - else** block we can tell the micro:bit to display a different image depending on which message it has received. If it receives "smile", display a happy face, and if it receives "frown", display a sad face.

1. Add the **if true then - else** block to the **on radio received receivedString** block.
2. Replace the **true** with a **" " = " "** block.
3. Drag the **receivedString** block to the position before the equals sign. Type the message ("smile") after the equals sign.
4. Drag a **show icon** command to the first indent in the **if** block. Select the happy face from the dropdown menu.
5. Drag another **show icon** to the second indent (after **else**). Select the sad face from the dropdown menu.
6. Drag a **clear screen** block below the if-else block.

Activity: Receiving Two Messages

Ask the students to follow along step-by-step as you create this code. Ask them if they can work out the next step as you go.
Try running the code and see what happens now.

Morse Code

Discuss: Morse Code

Ask the students if they have heard of Morse code. Can they tell you anything about it?

Explain: Morse Code

Before the telephone was invented, we could not transmit complex messages over a distance. In the 1800s, the American Samuel Morse designed a system of communicating using only two symbols - **dots** and **dashes**.
This communication method involved sending either quick electrical pulses (**dots**) or longer electrical pulses (**dashes**) down a wire.
The receiver would then have to translate these **dots** and **dashes** into letters.

Activity: Morse Code

Provide the students with the Morse code translation handouts.
Go through an example Morse code message (**....   .   .-..   .-..   —**) together, one letter at the time, and see if they can work what it says using their handouts.
The message says H E L L O.

Discuss: Morse Code

Even though we can now send complex messages to each other all over the world, Morse code can still sometimes be useful. Can you think of some situations where it might be good to use a coded language like Morse code?

Emergency situations: Sound, light, or even physical materials can be used to display an emergency message such as “SOS”. This can be especially useful if you don’t speak the same language as the person you are communicating with.

Secret messages: Not many people can understand Morse code. You can use it to send secret messages to your friends that other people are unable to understand.

Communication: For people who are non-verbal or have difficulty communicating with speech, codes such as Morse code can be used to help with communicating since it only requires two sounds or images.

Activity: Micro:bit Morse Code

Ask the students to create a program that can allow the micro:bits to communicate using Morse code.

This program will be very similar to the one created earlier that sends a happy face or a sad face. This time, the micro:bits should send a dot or a dash. Dot and dash icons can be made using the **show leds** command.

The string messages sent between micro:bits should also be changed accordingly

Activity: Morse Code

In pairs, get the students to send Morse code messages to each other across the classroom. Each person should have one micro:bit with a working code, both programmed to the same radio group.

Using the printed handout with a translation of the Morse code alphabet to the English alphabet, see what messages they can send to each other and if they can accurately decipher them.

Extension: Word Breaks

Discuss: Morse Code Limitations

Ask the students if they can think of any problems with their Morse Code programs or limitations with Morse code in general.

What if we need to send entire sentences instead of just words? How will the receiver know when one word is ending and another is beginning?

Can the students think of a way to solve this issue?

Explain: Word Breaks

One solution could be to leave a longer gap. However, this could be time consuming. Instead, we can add a third symbol to represent a word ending. We can make this symbol using the **show leds** command, to make a '/' for example.

Extension Activity: Additional Input

We are already using the A and B buttons to send dots and dashes, so what input can we use to send a third signal?

Ask the students to look in the **Input** section for possible blocks.

Some examples are: **on shake, on tilt right, on logo pressed.**

Extension Activity: Word Breaks

Ask the students to add a third input block to their code to send a word break icon.

1. Drag the chosen input block to the code editor and add a **radio send string** command. Give the command a string to send (e.g "space").
2. Expand the if statement to include another **else**. Change the conditional to **else if receivedString = "space"**.
3. Drag a new **show leds** command into the expanded if statement.

Extension Activity: Letter Breaks

If the students want to expand further on their program, they can add letter breaks - a fourth icon to represent each letter ending, since a letter can be made up of four different dot and dash symbols.

This will follow the same process as the word breaks, choosing yet another input block, and adding the required code to the sending and receiving code blocks.

Differentiating for Learners

- Extension tasks have been provided to challenge students to improve their code by adding word breaks and letter breaks so that they are able to communicate more easily with each other using Morse code.

- Some learner may need more guidance in assembling the code than others. Since this workshop is aimed at ages 9-11, there might be several learners with little coding experience so will require step-by-step instructions in assembling the blocks. Other learners can use their previous coding experience to write the algorithm without any code provision.

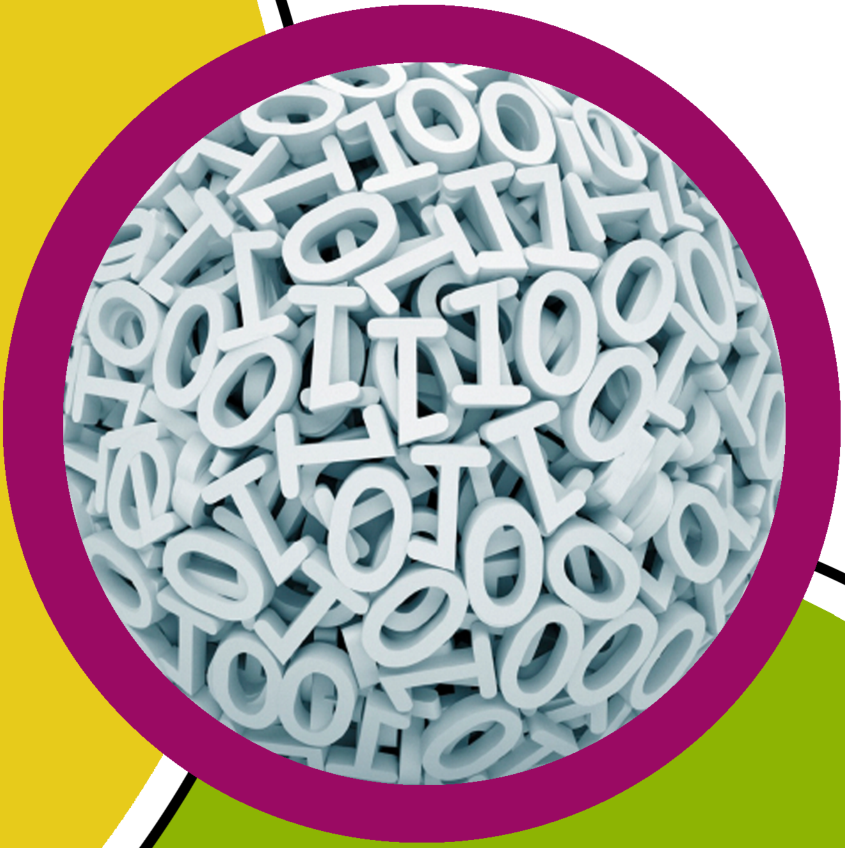
- The micro:bit website and editor allow for navigation using accessibility features such as a screenreader, or speech recognition software.

Where To Go Next

- When adding word breaks in the extension activity, the learners can be given freedom to test the limitations of the micro:bit input sensors, using different input blocks to send radio messages.

- Micro:bit also has a user-friendly Python editor. This workshop can be adapted to introduce learners to a text-based programming language.

- Another extension could be to increase the number of micro:bits in each radio group to send messages to larger groups. They can also add other messages to ask for words to be repeated.



technocamps

@Technocamps

Find us on Facebook