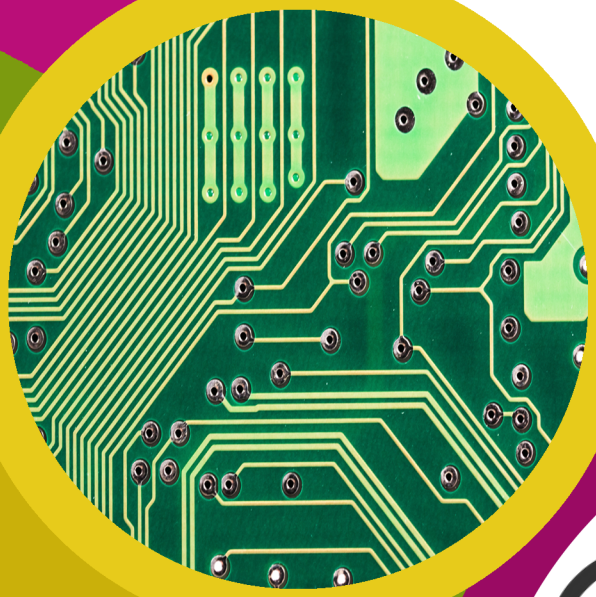
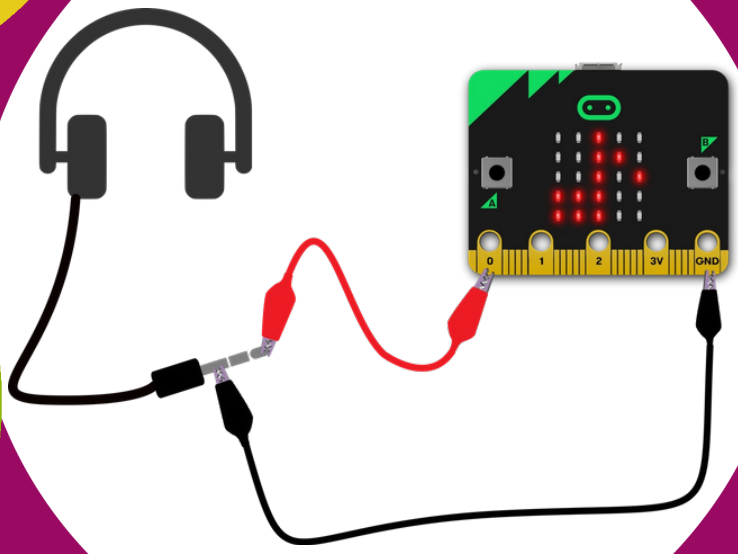
technocamps

Musical Micro:bit
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

Links to Science and Technology AoLE

Computation:

(PS2) I can create simple algorithms and am beginning to explain errors.
(PS2) I can follow instructions to build and control a physical device.
(PS2) I can follow algorithms to determine their purpose and predict outcomes.
(PS3) I can identify repeating patterns and use loops to make my algorithms more concise.

Design Thinking:

(PS2) I can explore how different component parts work together.

Links to Other AoLEs

Expressive Arts:

(PS2) I can explore and experiment with and then select appropriate creative techniques, practices, materials, processes, resources, tools and technologies.
(PS2) I am beginning to apply techniques in my creative work with guidance and direction.
(PS2) I can create my own designs and work collaboratively with others to develop creative ideas.

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 will also use **Creativity and Innovation**. They are encouraged to discuss and implement strategies to improve their program. Learners are also tasked with designing and creating music playing devices and melodies. This workshop gives learners the opportunity to explore chords and pitches, design their own micro:bit piano, and combine programming and music to create melodies.

The **Data and Computational Thinking** section of the **DCF** applies to this resource. Students will learn to create and analyse algorithms, detect and solve coding errors, and identify possibilities for improved efficiency using loops and events.

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 loops and event-based programming which are critical to most common programming languages. The resource also explores basic music theory such as chords and pitch changes, challenging the learners to design and create their own music playing devices and melodies. 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 about musical chords and pitch changes, and how we can use programming to create our own micro:bit pianos to make music.

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.

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: Melody

Program the micro:bit to play a piece of music. Drag the **play tone** command into the **on start** block and choose a note from the dropdown menu.

Micro:bit also has preset melodies. Try dragging the **start melody** command to the on **on start** block and choose a melody from the drop-down menu.

Headphones

Explain: Headphones

Micro:bits have a built in speaker that can output sound. However in a class of students all playing different notes and melodies, we can connect headphones to make it easier to hear your own.

The headphones need to be connected to the pins at the bottom of the micro:bit using crocodile clips.

Activity: Connecting Headphones

Connect the headphones to the micro:bit. Connect pin 0 to the tip of the headphone plug and the GND (ground) pin to the longer part of the headphone, as shown in the diagram on the slide.

Activity: Make Your Own Music

Ask the students to create their own melody using several **play tone** commands, one after the other in the **on start** block.

Give them some time to be creative and see what melodies they can make.

Activity: Extended Melody

Ask the students to build and run the program on the slide. It plays a longer melody with repeated block segments.

Discuss: Extended Melody

The code only plays a short tune but already includes a lot more blocks.

Discuss what the code would look like if the song was two, five, or hundreds of times longer.

Ask the students how we could reprogram this without adding blocks for each individual note.

The solution is to use loops. Our melody has two repeated blocks of notes: CDEC and EFG. These segments can be repeated using loops instead of writing out the code twice.

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: Melody Loop

Ask the students to create a program that repeats a short melody. They can do this by adding the **play note** commands to the **forever** loop.

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 melody loop activity in the **on start** block instead of the **forever** block.
Answer: The code would run once and then stop without repeating.

Explain: Jukebox

A jukebox is a music playing device that gives you the choice of multiple songs to play.

So far, the micro:bits can only play one song. We need to program it to play several different songs, and give the user the choice of which one to play. To do this we use events.

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 play one song when we press button A and a different song when we press button B.

These commands are found in the **Input** section.

Activity: Events

Create a program that plays a note when button A is pressed.

Add a **play tone** command to the **on button A pressed** block.

Activity: Jukebox

Ask the students to add another input block to play a different note when button B is pressed.

Extend this program to play a song when the buttons are pressed, instead of single note. This can either be a preset melody, or a custom one.

Now the students will have their own jukebox.

Explain: Instrument Input

Instead of using buttons to play notes or a melody, we can connect our input device. To do this we use the pins at the bottom of the micro:bit, just like we did to connect the headphones.

We can make our instrument so that, when we play it, an input signal will be sent to the micro:bit. The micro:bit will then produce a sound output.

Activity: Building a Piano

We're going to make our own cardboard piano that we can connect to the micro:bit.

Instruct the students on how to make these:

Cut out three tin foil rectangles. Glue the tin foil rectangle to each side of the cardboard, wrapping them around one side. The foil will represent keys on a piano.

Activity: Connecting the Piano

Attach three crocodile clips to the cardboard piano, one to each piano "key". Connect the other ends of the crocodile clips to pin 1, pin 2, and the GND pin, as seen on the slide.

Make sure all the students have successfully constructed and connected their cardboard pianos.

Discuss: Piano Code

Discuss which blocks will be required to build the code for playing micro:bit music on the cardboard piano.

Instead of using the buttons as input, we need to use the pins. There are **on pin P0 pressed** blocks in the **Input** section.

When we press on one of our piano keys together with the ground pin, we complete an electrical circuit that sends a signal to the micro:bit.

Activity: Piano Code

Ask the students to write a code that allows them to play notes using their cardboard pianos.

Add **on pin P1 pressed** and **on pin P2 pressed** blocks to the code. In each block add a different musical note.

Allow the students to experiment with some different musical outputs using the pins.

Extension: Another Note

Explain: Additional Input

So far, we can get our piano to play one note when pin P1 is pressed and another note when pin P2 is pressed.
We have run out of pins to use to create more sound, but there is a way around this.

We can program the micro:bit to play a third different output when both pins are pressed at the same time. There is no input block for this, so how do we program it?

We need to use conditionals.

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.

Activity: Another Note

Ask the students to write a code that allows us to play a third note using conditionals. You will probably have to talk through building this code, step-by-step. You can ask the students to predict what the next step will be.

1. Add an **if true then** block to the **forever** loop and extend it to allow for three statements: pin 1 pressed, pin 2 pressed, pin 1 and 2 pressed together.

2. Add a **pin P1 is pressed** and **pin P2 is pressed** command to the **if** block.

3. Add two different **play tone** commands to the **if** statement.

4. Drag an **and** block from the **Logic** section to the final segment of the **if** block. Add a **pin P1 is pressed** and a **pin P2 is pressed** command to the **and** block.

5. Add another **play tone** command.

Discuss:

Ask the students what they know about chords.

Explain: Chords

Chords are a combination of multiple notes playing simultaneously, to make a sound. Usually they consist of three notes played together.

The micro:bit can not really play a chord since it can only play a single note at the time. However, if we play three notes quickly, one after the other, it can sound like a "broken" chord.

We can do this by adding three notes back-to-back to our input blocks.

Activity: Chords

Ask the students to make a program that plays a different chord when a different pin is pressed.

Two chord examples are shown on the slide (F major and A minor)

Explain: Pitch

Our micro:bit pianos can now play notes and chords, but we can't yet change these sounds without reprogramming and downloading the code again.

It is possible to add more input blocks to higher or lower the pitch of the note or chord without changing the code every time. The pitch of the note doubles (gets higher) when you move up an octave and halves (gets lower) when you move down an octave.

To do this we need to define the notes as variables that can be changed.

Explain: Variables

Variables are items that can remembered and changed by the micro:bit. It gives us a place to store something.

Variables can take several different forms such as a number or a text.

We can change the variable's value and use it in many ways in our code. But first, we need to define it.

In MakeCode, they are found in the **Variables** section.

Activity: Creating Variables

Instruct the students on how to create variables.

To create a variable, click on the **Variables** section and **Make a Variable.** Create four variables: A, C, E, F representing four musical notes.

There should now be a few more commands available for selection in the **Variables** section.

Activity: Changing Pitch

Instruct the students on how to make a micro:bit piano that can change the pitch of the chords. The steps are shown on the slides. You will probably need to talk the students through each step but you should ask them for ideas at each step.

1. Start by setting each of the variables to the frequency of their note. A = 440, C = 523, E = 659, F = 349.

2. Now we need to tell the micro:bit what musical output to play when the pins are pressed. You have already done this when you programmed the chords. Instead of the preset notes, drag your new variables to the **play tone** blocks.

3. Drag and drop the **on button A pressed** block to your code and add four **set variable to 0** commands to the block. To make the pitch higher, we need to double the frequency of the notes.

4. Drag and drop four multiplication commands from the **Math** section to the **set** commands.

5. Drag and drop the note variables from the **Variables** section to the multiplication commands, and multiply them by two.

6. Repeat steps 3-6 but using division commands for when button B is pressed.

Now you should have a working micro:bit piano that can play two chords when pressing the keys, and change the pitch higher or lower when pressing buttons A and B, respectively.

Differentiating for Learners

- An extension task has been provided to challenge students to improve their code using by adding a third output note, using conditional statements.

- 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. This is especially true for the last program that changes the pitch of the output. 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

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

- There are several additional extension tasks that can teach learners about different micro:bit features. For instance they can produce visual output to display which note is being played. Learners can also create a metronome program that teaches about musical tempo and beat.

technocamps

@Technocamps

Find us on Facebook