LESSON 31

Sound Sampler and Playback Device

40 minutes | Introduction to Python | Outcome 3.11 | Outcome 3.12 | Outcome 3.13 | Outcome 3.14 | Outcome 2.6 | Outcome 2.7 | Outcome 2.14

WHAT THIS LESSON DOES

In this lesson, you'll build a sound sampler and playback device with your Micro:bit. Follow steps to detect sound, record voice as intensity levels, and play them back as tones, using hardware and coding concepts like lists and loops.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
3.11	use and control digital inputs and outputs within an embedded system
3.12	measure and store data returned from an analogue input
3.13	develop a program that utilises digital and analogue inputs
3.14	design automated applications using embedded systems
2.6	construct algorithms using appropriate sequences, selections/conditionals, loops and operators to solve a range of problems, to fulfil a specific requirement
2.7	implement algorithms using a programming language to solve a range of problems
2.14	describe the difference between digital and analogue input

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand how to interface with hardware components like microphones and speakers on a Micro:bit.
- Develop skills in detecting and processing sound events and levels using programming logic.
- Master the use of lists for storing and manipulating data sequences.
- Apply control structures to manage recording and playback functionalities.
- Explore creative sound manipulation by converting sound intensities into playable tones.

BEFORE THE LESSON

- Confirm each micro:bit is a V2 board. The microphone and speaker functions used here do not exist on V1, so the code will fail silently or error on older units.
- Check that micro:bits are cabled to machines and that students can flash a .hex to the physical board. The microphone and speaker only work on real hardware, not the simulator.
- Flash the two-line import setup to one board yourself first to confirm the toolchain and USB connection work before the class starts.

Sound Sampler and Playback Device

HOW THE LESSON RUNS

- Introduction
- Import Modules and Basic Setup
- Detect Sound Events
- Sample Sound Levels
- Record Sound Samples
- Playback the Samples as Tones
- Enhance with Sound Event Trigger
- Wrap Up and Extensions

TEACHING MOVES

- 3. Detect Sound Events. Have students run this and speak near the board to see the happy face appear. Point out the sleep(100) sets how often the loop checks, and that SoundEvent.LOUD is a fixed threshold, not the level itself.
- 4. Sample Sound Levels. Draw the distinction clearly: this returns a number 0 to 255, not a true/false event. Stress it samples loudness, not the audio waveform. This numeric range is what makes the list and playback possible.
- 5. Record Sound Samples. Walk through the for loop with them: 20 samples times 100ms equals 2 seconds. Ask them to predict the list length before running, then check len(samples). This anchors why the loop count controls recording duration.
- 6. Playback the Samples as Tones. Explain level times 10 shifts the 0 to 255 range into an audible frequency band. Have them change the multiplier and listen to the difference, so pitch scaling becomes concrete rather than magic.
- 7. Enhance with Sound Event Trigger. Point out the combined condition: LOUD event or button A now both start recording. Ask why this can misfire, background noise triggers it, before they run it. Good moment to discuss real-world sensor thresholds.

WHAT TO WATCH FOR

- Students think sound_level() records their actual voice like a phone voice memo. Reinforce that it captures loudness as numbers only. Print the samples list to the console so they see a list of intensities, not audio. The playback is a reconstruction, not the original.
- Students expect the microphone or speaker to work in the browser simulator and think their code is broken when nothing happens. Have them flash to a physical V2 board before concluding the code is wrong. Confirm the board is V2 first.
- Confusing the while loop's continuous checking with a one-off run, so they expect recording to happen without pressing a button or making a sound. Trace the loop out loud: it checks the trigger condition every cycle and only records when it is met. Have them explain the trigger back before running.

DIFFERENTIATION

- Emerging: Give the working step 5 code and ask them only to change the number 20 and observe the effect on recording length, before moving on. Pair them with a student whose board is already flashing successfully so environment problems do not stall the whole task.
- Developing: Ask them to predict, in words, what each condition in the step 7 loop does before running it, then confirm against the behaviour.
- Proficient: Have them implement the suggested extensions: adjust pitch scaling or increase sample count, and explain the trade-off between resolution and recording length. Ask them to document this as evidence for an Applied Learning Task on embedded systems, noting hardware interfacing and data storage design decisions.