LESSON 29

Radio Messaging Network

40 minutes | Introduction to Python | Outcome 3.11 | Outcome 3.13 | Outcome 3.14 | Outcome 2.6 | Outcome 2.7 | Outcome 2.9 | Outcome 2.16

WHAT THIS LESSON DOES

In this lesson, you'll create a simple chat system with multiple Micro:bits using radio communication. Build a network to send and receive predefined messages, using buttons to select and send them while learning string manipulation and functions.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
3.11	use and control digital inputs and outputs within an embedded system
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
2.9	assemble existing algorithms or create new ones that use functions (including recursive), procedures, and modules
2.16	use data types that are common to procedural high-level languages

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand the principles of wireless communication using radio modules in a network.
- Develop skills in programming multi-device interactions with Micro:bits.
- Apply string manipulation and list handling to personalise and manage messages.
- Implement control structures and functions for message sending and receiving.
- Explore conditional logic to create automated responses based on message content.

BEFORE THE LESSON

- Confirm your editor imports the radio module cleanly and run the full final program yourself once, either in the simulator or on hardware, so you know it works.
- If using physical micro:bits, charge or cable up at least two and have USB leads ready, since a network needs more than one device.
- Agree a single radio group number for the whole class in advance so everyone can hear each other, or split into pairs on separate groups to avoid one noisy channel.

Radio Messaging Network

HOW THE LESSON RUNS

- Introduction to the Radio Messaging Network
- Import Modules and Set Up Radio
- Define Messages List and Sender Name
- Select a Message
- Send a Message
- Receive and Display Incoming Messages
- Add Conditional Logic for Responses
- Download your Code
- Wrap Up and Extensions

TEACHING MOVES

- 2. Import Modules and Set Up Radio. Stress that radio.config(group=1) is what pairs devices. Two micro:bits on different groups will simply never hear each other. Have the class settle on their group now.
- 3. Define Messages List and Sender Name. Point out that messages is a list and sender_name is a string, and that each student must change sender_name to their own name so received messages are identifiable.
- 4. Select a Message. Walk through why selected_index starts at -1 and why the wrap-around check uses len(messages). This is the classic off-by-one point. Test the cycle end to end before moving on.
- 5. Send a Message. Show the string concatenation that prepends sender_name to the chosen message. Ask them to predict exactly what text goes over the air before they press B.
- 6. Receive and Display Incoming Messages. Highlight that radio.receive() returns None when nothing has arrived, which is why the if incoming check is needed before scrolling.
- 7. Add Conditional Logic for Responses. Draw out that 'Help!' in incoming is a string membership test, not equality, so it matches even when a sender name is attached. Test one micro:bit auto-replying to another.
- 8. Download your Code. Remind students the same code goes on every device in the network. Watch for micro:bits that appear connected but were never flashed.
- 9. Wrap Up and Extensions. Restate the four techniques used: lists, string concatenation, a function, and conditionals. Offer the group-channel experiment as a stretch.

WHAT TO WATCH FOR

- Students assume any two micro:bits will communicate, then are puzzled when nothing is received. Trace it back to the group number. Have both devices print or confirm their group value and set them to match.
- The index wrap-around is off by one, so the last message is skipped or an index error is raised. Step through with a five-item list on paper: index 4 is the last valid one, and 5 must reset to 0. Match that against len(messages).
- Students expect radio.receive() to wait for a message and are confused it returns nothing. Explain it is checked repeatedly inside the loop and returns None when empty, which the if handles. Nothing displaying is normal until a message arrives.

DIFFERENTIATION

- Emerging: Pair with a partner who has a working device so they can see messages actually arrive, and have them get one button working before attempting both. Give the final code as a reference and let them retype in stages, comparing after each step.
- Developing: Ask them to explain aloud what text is sent when they press B, tracing the concatenation, before running it.
- Proficient: Have them extend the auto-response to handle a second keyword, or use a gesture such as shake to send, then explain their design choice as evidence for an Applied Learning Task. Challenge them to add a message counter or timestamp to each transmission and justify how they tested it.