Extension: Co-op Bop!

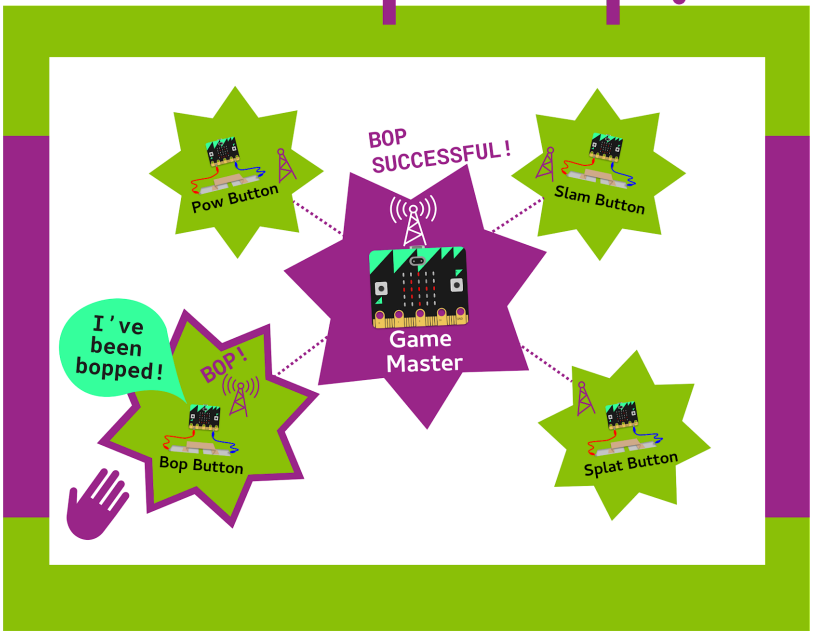
Make a Bigger, Better Bop It, and play as a team!

Spread out your game by making buttons that talk to your game with radio!

This extension has 2 parts.

- Making radio buttons.
- Changing your game to listen for the radio buttons.

You can work in groups!



Learning about Radio

We'll need to know how to use the radio for this extension. Here's some commands

Action	Code
Set the channel, we set it to 6.	`radio.config(channel=6)`
Turn the radio on	`radio.on()`
Send a message, we sent "bop"	`radio.send("bop")`
Receive a message, check if it matches 'bop'	`if radio.receive() == "bop":`

Part 1

Making Radio Buttons

A new file that sends a radio message every time you press a button.

Use the Micro:Bit buttons or craft your own!

Part 2

Listening for Radio Buttons

Adapt your game to use radio messages to complete actions.

You can connect lots of buttons, so work as a team to make more.

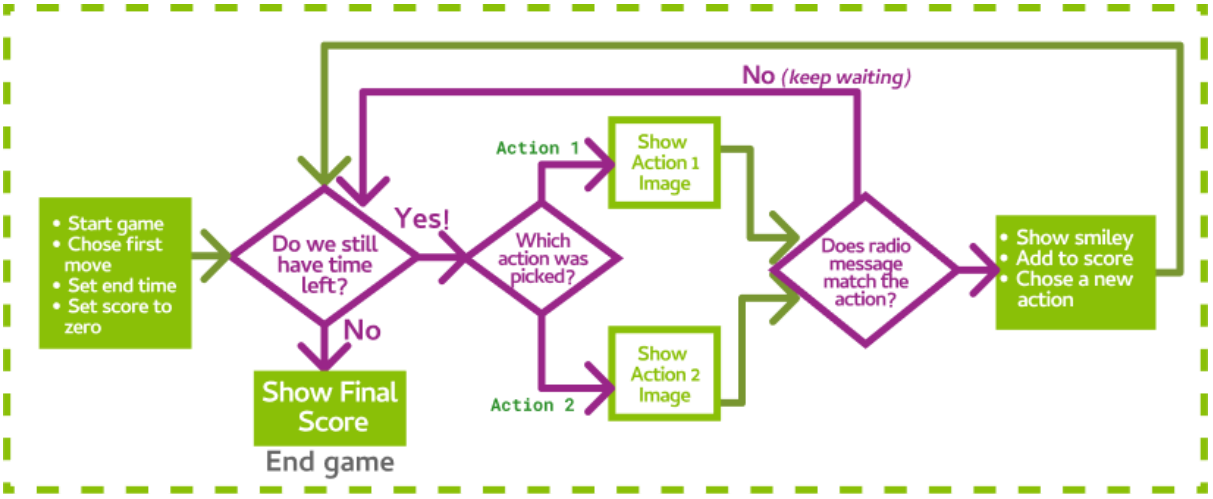
Part 2: Listening for Radio Buttons

We'll re-write our game so it uses radio to complete actions

It will:

- Choose actions, keep score, and loop, just like before.
- Listen for radio messages that match the action it is waiting for.

This new code will look a lot like your original game code!



Task 2.1: Configure the Radio

We'll need to start a new file for our game master!

1. Click the **Project** button to the left, and then click **Create file**.
2. Name your new "game_master" file.
3. At the top of your file, `import` the `microbit`, `random` and `radio` modules.
4. Turn the radio on with `radio.on()`.
5. Configure the radio to use the channel that the room coordinator gave you.

Task 2.1: Setting up the game

1. **Create a list of all the actions** you want to connect to your game.
   You can add more buttons from your friends here later!
2. Show the starting image
3. Create a score variable and set it to 0
4. Like in your first game, calculate the ending_time for the game using `running_time()`
5. Chose the first random action from the list, assign it to a variable called `action`

Task 2.2: Play the game!

1. Like your first game, create a `while` loop that keeps running until the finish time.
2. Create if statements to show the actions.
   - The if statement should check what the action is
   - Inside the if statement show the image you associate with that button
   - Make sure you have an if statement for every action in the list

Task 2.2: Checking for messages

We need to check if we've received a message about button presses!

1. Go to the place in the code after your if statements show the image.
2. Create a new if statement to check on incoming messages
   - The if statement should check if the current action is equal to `radio.receive()`
   - If it is, add to the score, show a smiley and choose a new action randomly.
3. Add an else statement to deal with the case that it is not a match.
   - If it doesn't match, use `continue` to loop again

Task 2.3: Game over!

1. After your loop add the code to scroll the final score. Time to give your game a go!

☑ CHECKPOINT ☑

If you can tick all of these off you have finished this Extension:

☐ You have configured your radio channel with the number the tutor gave you

☐ Your game chooses actions which can be completed by radio button presses