BBC micro:bit Challenges

Micro:bit and MakeCode Getting Started Prompts

1) Program the micro:bit to display a new image or scrolling text

2) Program the micro:bit to display one image when the A button is pressed and a different image when the B button is pressed.

3) Program the micro:bit to display the temperature. Test your thermometer.

   Super duper gifted and talented extra credit: Display the temperature in Fahrenheit

4) Program the micro:bit to display a smiley face when the temperature is above a certain value and sad face when it gets cold.

5) Program one micro:bit to "pass a message" to another micro:bit, using the radio features, when a user presses a button.

   You can't handle this challenge! Program two more micro:bits to pass messages between each other.

6) Program one micro:bit to cause another micro:bit to produce some action, such as light an LED, drive a servo, or run a program.

7) Make a micro:bit stopwatch. (tricky)

1-hour micro:bit Workshop Challenge

1. Program the micro:bit to behave like a die rolled by a player.

2. Instead of displaying a number, display a die face.

3. Add some effects to make it look (or sound) like rolling a die before it settles on a "side"

4. Roll your die and have it send the value to appear on a friend's micro:bit.

5. Make the die rolled in your hand make something else happen on a friend's die, like flash an LED x times.

6. Use your micro:bit die with Scratch to control an animation or interact with a board game you program.

Read more about the thinking behind the design of this activity at https://inventtolearn.com/1-hour-microbit-workshop-challenges/

The Invent to Learn Guide to the micro:bit

Getting Started with micro:bit Radio Communication

By Gary S. Stager, Ph.D.

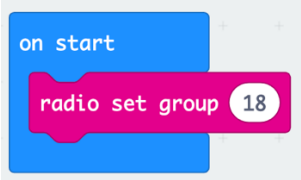
Although the block-based MakeCode environment is quite intuitive, the radio functionality of the micro:bit requires a tiny bit of instruction. You can use the micro:bit's radio functionality to pass messages, as a remote control, or even to model social behavior between robots. Here is a concise guide to get you started.

Imagine the micro:bit as a walkie-talkie

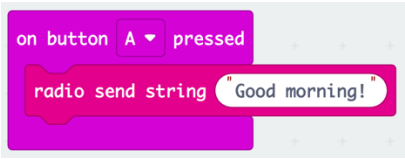
If you talk into a two-way radio, like a walkie-talkie, any other radio within range and using the same frequency or channel should receive your message. Friends should also be able to speak to you. It does not matter how many walkie-talkies are involved. Each radio on the same frequency can participate in the conversation. The micro:bit works in a similar fashion.

Voice is but one of the forms of data a walkie-talkie can share. Some allow users to exchange morse code. The micro:bit doesn't transmit voice, but it does have the ability to share text (strings) and numbers between a seemingly infinite number of micro:bits using the same channel within a limited physical range.

1. Each micro:bit in your "network" needs to be set to the same channel. If you are exchanging private messages, keep the channel extra secret quiet. To set the channel, each micro:bit being used needs to have an `on start` block that sets the *same* channel from 0-255.

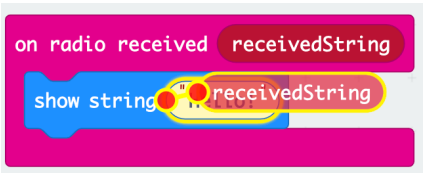


2. Each micro:bit needs code to broadcast messages/data to other listening micro:bits and code for reacting once it "hears" a new message. In this case, when a user presses the A button on the micro:bit, a message of "Good Morning!" is broadcast. If no other micro:bit is listening, that message falls on deaf ears.

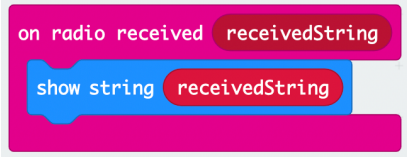


It is critical to decide if you are sending & receiving numbers or strings (text). You must the choose the correct *send* and *receive* blocks according to the data type.

3. If more than one of the micro:bits in your community contain blocks like the first two and the next one, you will have created a simple text-based walkie-talkie system.



Use an `on radio received` block and drag the `receivedString` variable into the input of the `show string` block. This will replace a literal message with the variable one being broadcast by another micro:bit. If dragging the variable into the container is too difficult, the variable may also be found in the blocks under the radio menu.



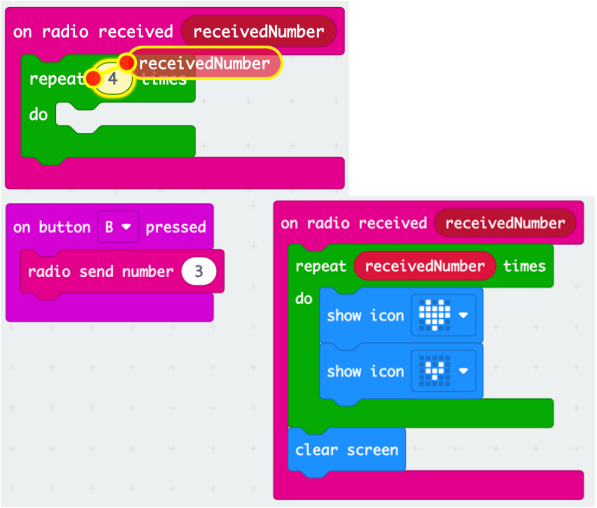
That block has the job of listening until a new message is broadcast (within) range and then doing something. In this case, it displays the message for the owner of the receiving micro:bit to read, just like passing a note or sending a text message.

You can of course program other buttons or micro:bit movements to send other messages.

All of the micro:bits participating in this chat need to have the same three blocks, or reasonable facsimiles downloaded to them.

You can also send numbers!
You might send a numerical value between micro:bits when you wish to trigger an event without a textual (string) message being received and displayed. For example, what if you want to tell another micro:bit to start an animation, flash an LED *X* times, or tell your robot to turn left or right? Sending a number might just do the trick.

1. Set the channel as you did in the previous example.
2. Use an `radio received receivednumber` block to build a simple animation that will repeat the number of times one micro:bit broadcasts to others.



Strings and numbers may be shared via radio in the same program.

Remember that every micro:bit in your network needs to have similar *send* and *receive* programs downloaded to the communicating micro:bits! Any program changes need to be downloaded to all of the micro:bits.

Challenges:
1. Make an LED connected to another micro:bit flash a certain number of times.
2. Shake one micro:bit to display a random dice face on a second micro:bit
3. Tilt one micro:bit to control the motions of a machine containing another micro:bit.
4. Invent a way for micro:bits to send messages across greater distance than their normal range. *Clue*: Think about how telegraphy facilitated Western expansion in the United States.