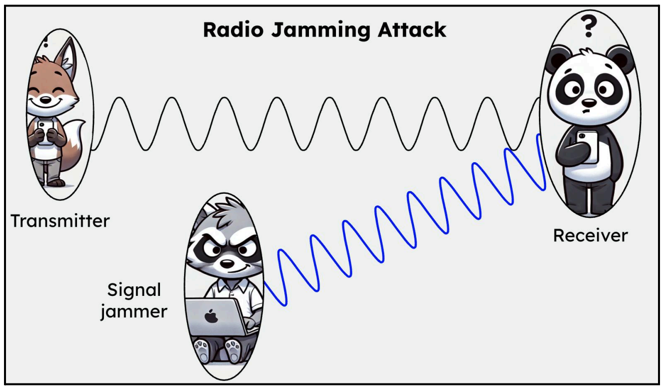
MAKING WAVES with radio

FACILITATOR GUIDE
Signal Jamming



Learning Objectives

1. Explore how interference can affect radio communication.
2. Investigate how changing frequency, radio channels, and physical barriers can affect signal transmission.
3. Use the micro:bit radio feature to test signal jamming and troubleshoot wireless disruptions.
4. Reflect on how jamming relates to real-world systems like Wi-Fi, Bluetooth, and drones.

Materials

Prepare the following for each small group (2-3 participants):

- 3 micro:bit devices per group (at least one Sender, one Receiver, and one Jammer)
- Computers with access to MakeCode (https://makecode.microbit.org)
- USB cables to connect micro:bits to computers
- Battery packs (optional)
- Optional materials to act as barriers (books, boxes, aluminum foil)

Safety

- N/A

Advance Preparation

- Confirm that all micro:bits can send and receive radio messages.
- Use three micro:bits to test the codes for the transmitter, receiver, and jammer.
- If your goal is to help learners build their coding skills, follow the step-by-step instructions provided below. We've included brief explanations with each step, which can be helpful for learners.
- If your goal is to focus more on the concepts of signal jamming and radio communication, consider giving learners the pre written codes and guiding them to tweak the programs as they explore ways to protect a device from jamming in Part C.

Activity Procedure

Part A: Introduce learners to the idea of interference.

1. Start with the following prompts to elicit ideas about what happens when communication gets disrupted or blocked:
    - Have you ever been on a video call and someone's voice kept cutting out or freezing? What do you think caused that?
    - Have you ever used Bluetooth headphones or a speaker and it suddenly stopped working or made strange sounds?
    - Have you ever tried to send something with AirDrop and it went to the wrong person nearby, or didn't send at all?
    - Have you ever been talking to someone and another person kept interrupting or talking over you? What happened to your message?

    Note: These prompts will help learners surface familiar experiences with interference—even if they don't yet know the term. Let learners share stories, then guide the discussion toward the idea that devices, like people, can sometimes have trouble communicating when too many signals are sent at once or when something gets in the way of the message.

2. Tell learners that all these are examples of interference—when signals overlap, compete, or are blocked, causing messages to be lost or delayed.

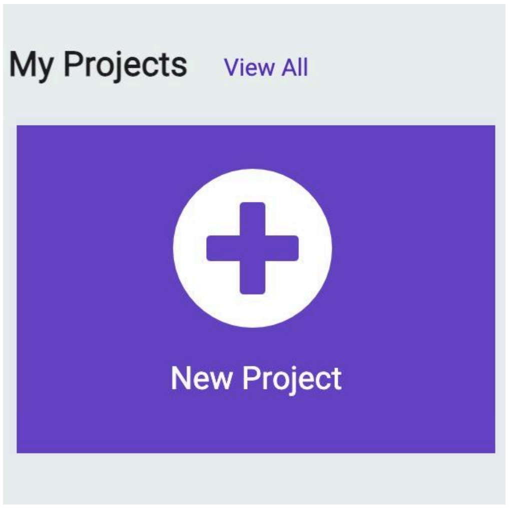
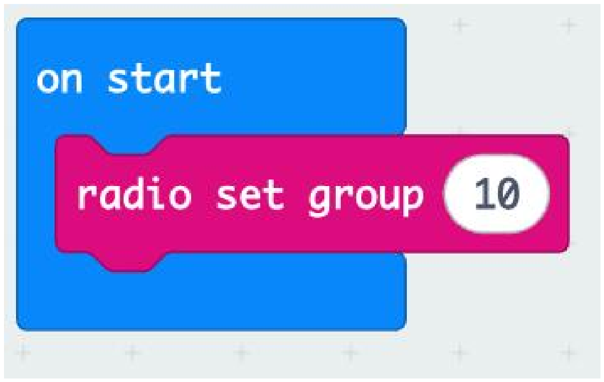
3. **Introduce the purpose of this activity:** Let them know that the purpose of this activity is to explore why interference happens and what we can do to reduce it, avoid it, or design systems that work even when signals are disrupted.

Note: Learners will investigate this by using micro:bits to simulate both communication and jamming, and test different ways to protect a signal or make it more reliable. Do not introduce the idea of jamming before testing the code for the sender and receiver.

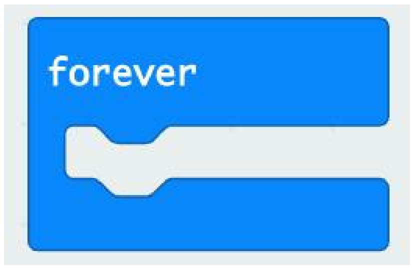
Part B: Set up a working radio communication system between two micro:bits

1. Organize 2-3 learners into a group. Provide them with the materials listed in the material section above.
2. Instruct learners to code one micro:bit as a sender and one as a receiver using the same radio group.

Coding the transmitter (sender)

1. **Open MakeCode** (https://makecode.microbit.org/) and create a new project. Give it a meaningful name, such as Transmitter" or "Signal Sender" so you can keep track of which device this is. 	**2. Set up a radio channel:** Go to the Radio category and drag the radio set group block into the on start block. Set the group number to any number between 1 and 255. Make sure the receiver micro:bit is using the same group. This block sets the "channel" or group your micro:bit will use to send messages. Only micro:bits using the same group number can communicate with each other. 

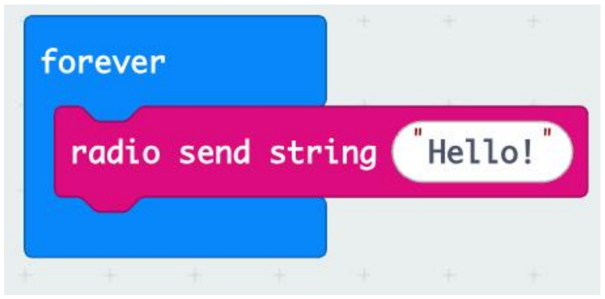
3. **Add a forever loop:** From the Basic category, drag a forever block into the workspace.
This block runs the code inside it over and over again without stopping.



```
forever
```

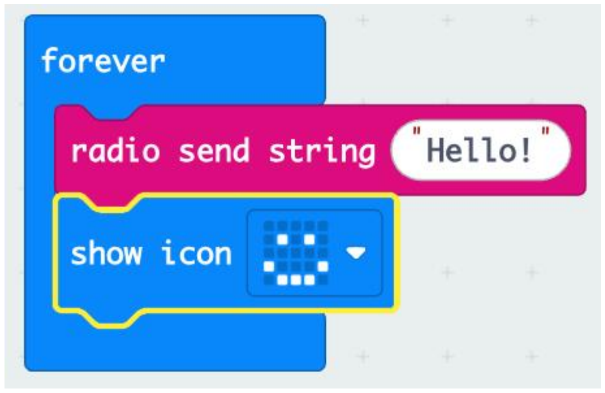
4. **Send a string over the radio:** Inside the forever block, go to the Radio category and drag in a radio send string block. Change the text inside the block to something short, like "Hello!" . This is the signal the receiver will be listening for.

This block sends a message to other micro:bits on the same radio group.



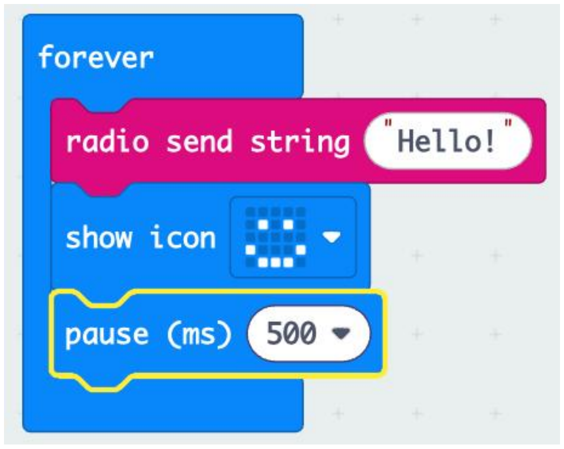
```
forever
    radio send string "Hello!"
```

5. **Add visual feedback:** Go to the Basic category and drag in a show icon block. Place it below the radio send string block. This shows a smiley face on the LED screen each time the message is sent, so you know it's working.



```
forever
    radio send string "Hello!"
    show icon
```

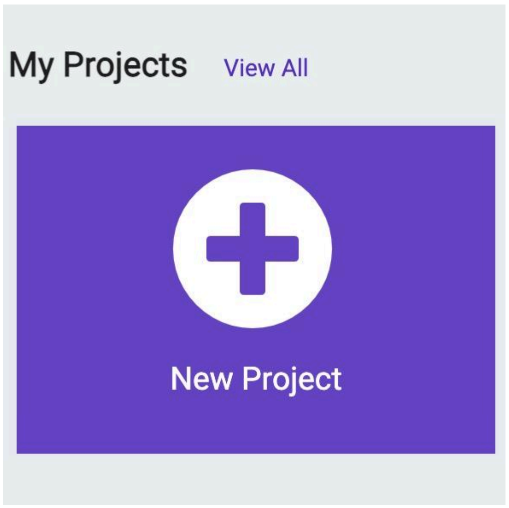
6. **Add a short pause:** Still inside the forever block, drag in a pause (ms) block from the Basic category and set it to 500 milliseconds. This short pause keeps the micro:bit from sending messages too quickly and gives time for the receiver to process each signal.



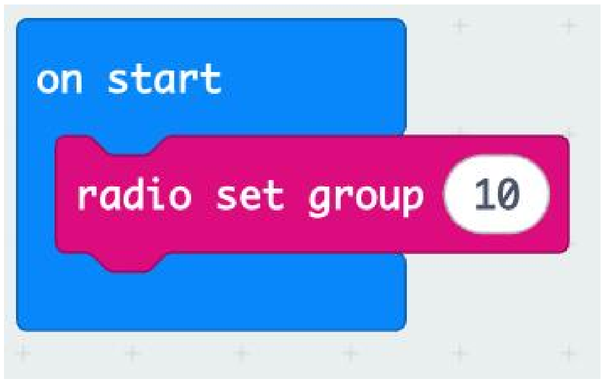
```
forever
    radio send string "Hello!"
    show icon
    pause (ms) 500
```

Coding the receiver

1. **Start a new project on MakeCode:** Give it a meaningful name, such as “Receiver” or “Signal Listener” so you know this micro:bit is for receiving messages.

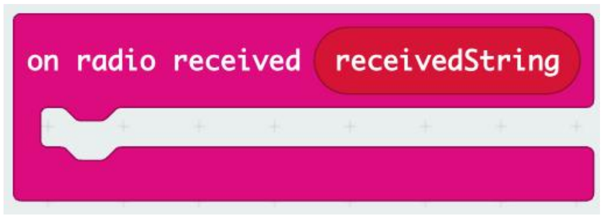


2. **Set a radio group channel:** Go to the Radio category and drag the radio set group block into the on start block. Set the group number to any number between 1 and 255. Make sure the receiver micro:bit is using the same group (channel) as the transmitter.



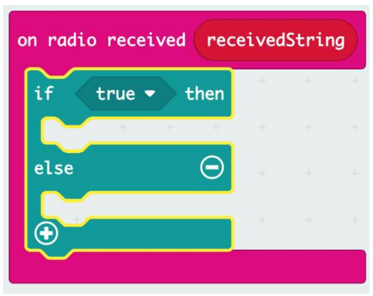
3. **Listen for incoming messages:** From the Radio category, drag a radio received receivedString block into the workspace.

This block will run any time the micro:bit receives a message over radio.

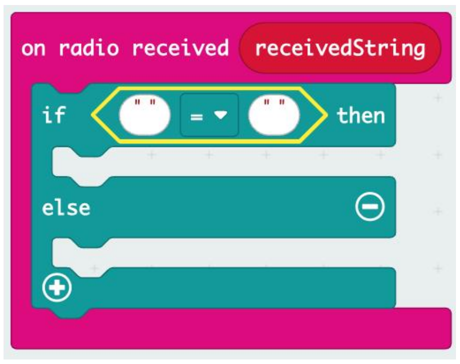


4. **Decide what to do with the message:** Go to the Logic category and drag an if true then else block into the on receivedString block.

The if true then else block lets your micro:bit make a choice. It checks if something is true. If it is, it does something; if it is not, it does something else. It is like asking: “If it is raining, use an umbrella, Else, wear sunglasses”.



5. Still in Logic, drag the “ ” = “ ” comparison block into the “true” part of the if block.

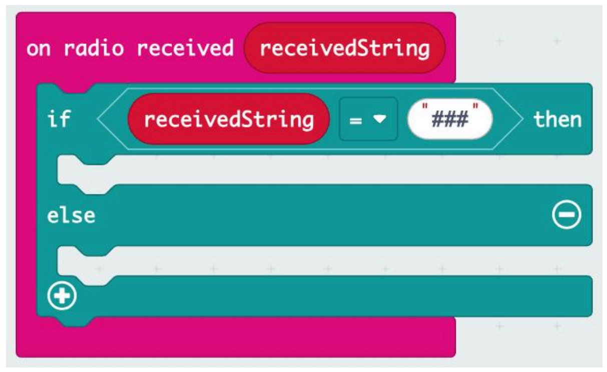


```
on radio received  receivedString
  if  " " = ▾ " "  then

  else                          ⊖

  ⊕
```

6. Drag the variable receivedString (you’ll see it at the top of the on radio received block) into the left side of the “ ” = “ ” comparison block. Click inside the text on the right side of the “ ” = “ ” comparison block and change it to ###.
Now your micro:bit will check: Did it receive the message “###”?



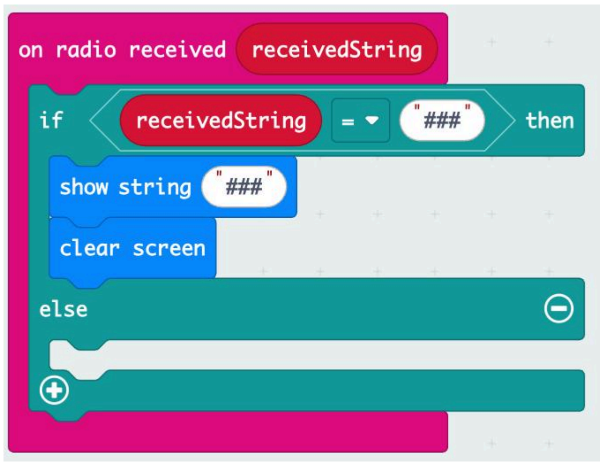
```
on radio received  receivedString
  if  receivedString = ▾ "###"  then

  else                          ⊖

  ⊕
```

7. **Show interference or a normal message:** In the then part of the if true then else block, Go to Basic and drag a show string block. Type in ###. Add a clear screen block after that.

This is what shows if the jamming signal is received.



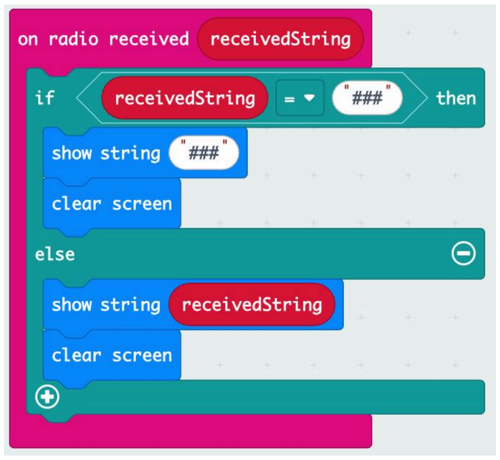
```
on radio received  receivedString
  if  receivedString = ▾ "###"  then
    show string "###"
    clear screen
  else                          ⊖

  ⊕
```

8. **Show interference or a normal Message:** In the else part of the if true then else block, add another show string block, but this time drag the receivedString variable into it.

This will display the message that was sent by the transmitter.

Add a clear screen block after that too.

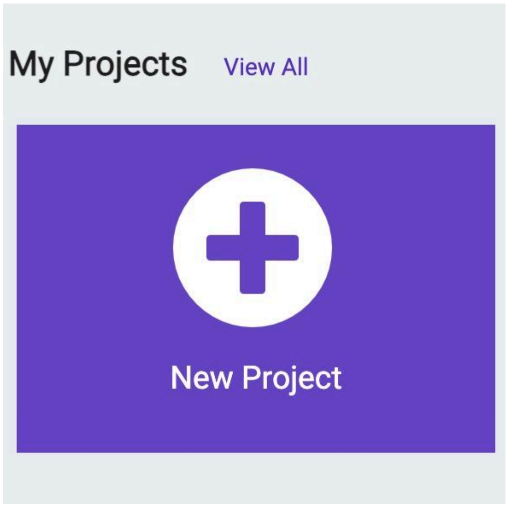
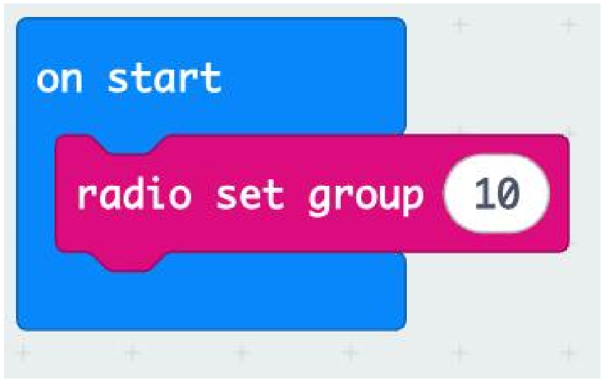


```
on radio received  receivedString
  if  receivedString = ▾ "###"  then
    show string "###"
    clear screen
  else                          ⊖
    show string receivedString
    clear screen
  ⊕
```

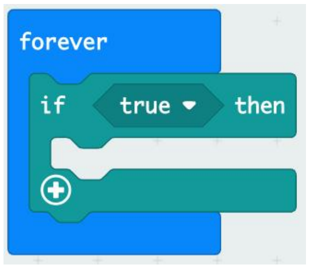
Test the communication between the sender and the receiver: When Button A is pressed on the sender, the receiver should show "Hello".

Part C: Disrupt the communication between the sender and a receiver by programming a jammer

Introduce the idea of jamming. Say something like, *Now that you've seen how the sender and receiver can communicate, what if we wanted to block that signal? Remember when we talked about interference earlier—like when a voice cuts out on a video call or Bluetooth suddenly stops working? That happens when signals get mixed up or disrupted. In the real world, there's a special kind of interference called jamming—it's when something sends out a signal on the same channel to intentionally block communication. Let's try making our own jammer to see if we can disrupt the message being sent between the sender and receiver."*

1. Start a new project on MakeCode. Give it a meaningful name, such as "Jammer". This will help you keep track of which device will block the signal. 	2. Go to the Radio category and drag the radio set group block into the on start block. Set the group number to any number between 1 and 255. Make sure the jammer is using the same group (channel) as the transmitter and receiver. 

3. Go to the Logic category and drag an if true then block into the forever block.

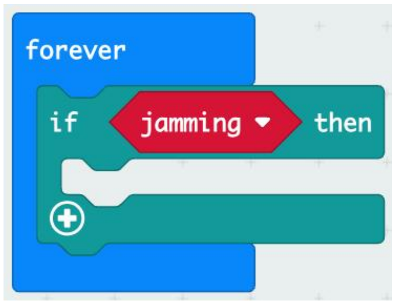


```
forever
  if true then
```

4. **Create a Variable Called "jamming":**

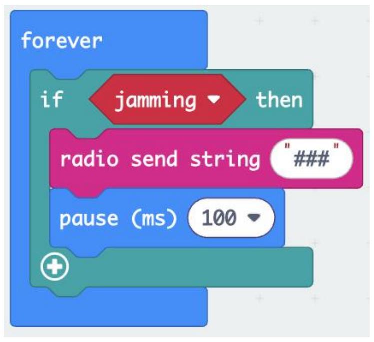
   a. Go to the Variables category.
   b. Click **Make a Variable**, and name it jamming.
   c. Drag a jamming block into the "true" part of the if block.

This variable keeps track of whether the jammer is turned on (1) or off (0).



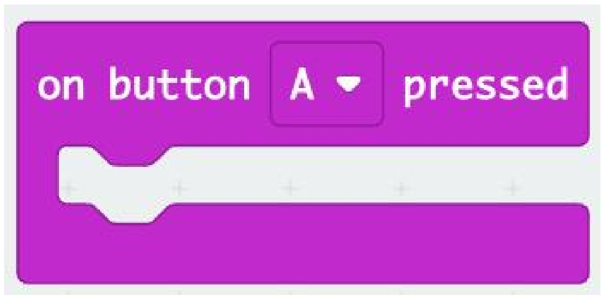
```
forever
  if jamming then
```

5. **Display message when jamming:** Go to the Radio category and add a radio send string block after the "then" in the if true then block. Add a pause (ms) block after that too.
With this code, we are telling the micro:bit: "If jamming is ON, keep sending the jamming signal ('###') over and over." The pause helps it send the signal repeatedly without going too fast. But how do we turn jamming on?



```
forever
  if jamming then
    radio send string "###"
    pause (ms) 100
```

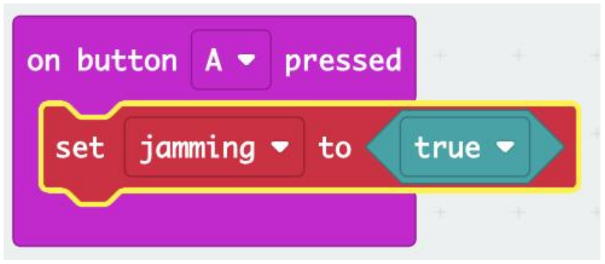
6. **Use a button to turn jamming on.** Go to the Input category and add an on button A pressed block.



```
on button A pressed
```

7. Drag a set jamming to 0 block from the Variables category and place it inside the on button A pressed block. Then, drag a true block from the Logic category and add it into the set jamming to block (replacing the 0).

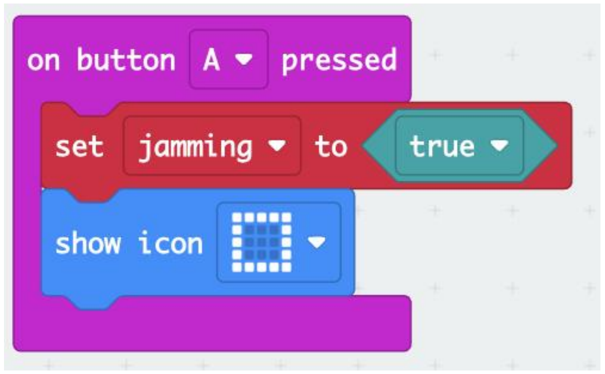
This code tells the micro:bit what to do when you press **Button A**. The block **set jamming to true** means you are turning jamming **ON**. You're saying, "Now it's time to start jamming!"



```
on button A pressed
    set jamming to true
```

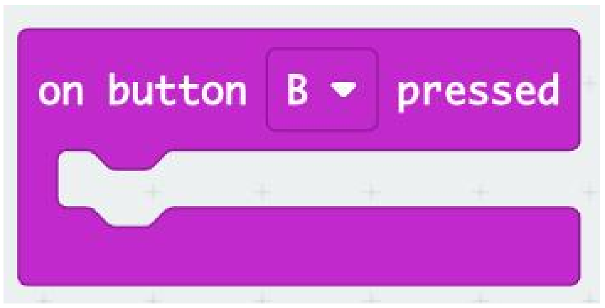
8. Go to the Basic category and drag a show icon block. Choose an icon you want the jammer to display when you press Button A and start sending the jamming signal.

The icon is just a way to show on the micro:bit's screen that the jammer is active — like a visual signal that says "I'm working!"



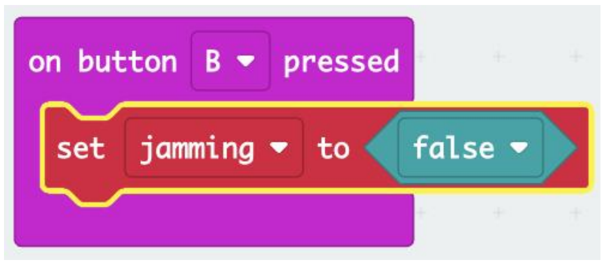
```
on button A pressed
    set jamming to true
    show icon
```

9. **Turn off the Jamming:** Add a on button B pressed from the Input category.



```
on button B pressed
```

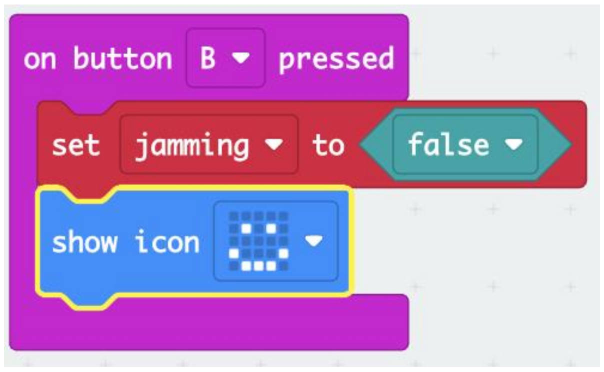
10. Drag a set jamming to 0 block from the Variables category and place it inside the on button A pressed block. Then, drag a false block from the Logic category and insert it into the set jamming to block (replacing the 0).



```
on button B pressed
    set jamming to false
```

11. Go to the Basic category and drag a show icon block. Choose an icon you want the jammer to display when you press Button B to turn off the jamming signal.

The icon is just a way to show on the micro:bit's screen that the jammer is not active — like a visual signal that says "I'm not working!" This means that the micro:bit is not sending the jamming signal ('###') over and over.



```
on button B ▾ pressed
    set jamming ▾ to false ▾
    show icon ▾
```

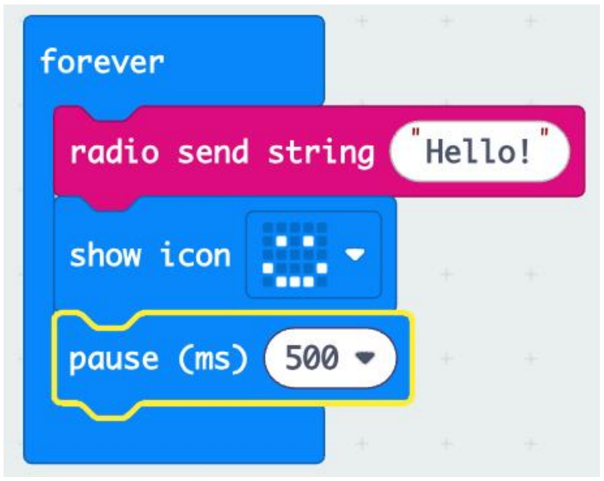
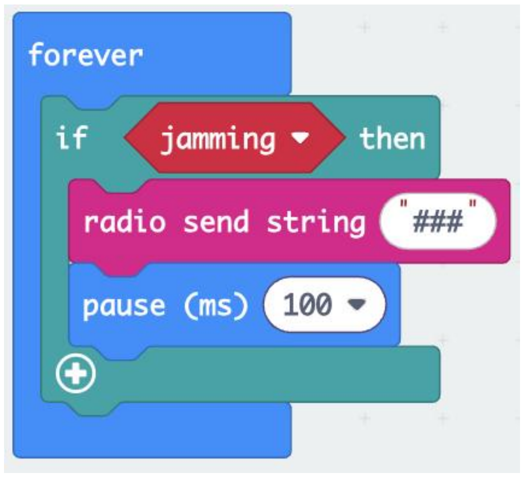
Test the jammer. When Button A is pressed on the jammer, the receiver should begin showing "###". When you press Button B, the jammer should stop sending "###". Note that there may be a short delay on the receiver, so it might continue displaying "###" for a few seconds afterward.

Make sense of how the signals are sent and received in the system. Use the following prompts:

- Why would the receiver display the jamming message ('###') more often than the transmitter message ('Hello!')?
- Based on your observations or the code that we developed for the micro:bits, what could explain why the message of the transmitter is shown less than the message of the jammer?

Point to the codes of the jammer and the transmitter to explore the transmission frequencies of both devices. You can ask:

- How does the pause length affect how often both devices send a signal?
- What can we do to the code to get more 'Hello!' messages to the receiver?

Transmission frequency of the transmitter	Transmission frequency of the jammer
 forever / radio send string "Hello!" / show icon / pause (ms) 500	 forever / if jamming then / radio send string "###" / pause (ms) 100

Part E: Explore solutions to improve communication between the sender and the receiver

Two ways to test the role of transmission frequency is by changing the frequency of the transmitter, or the frequency of the jammer.

Variable to Test	Example Test Setup	Question to Investigate
Jamming frequency	Change delay from 100 ms to 500 ms	Does slower jamming allow more "Hello" messages through?
Sender frequency	Make the sender transmit every 200 ms instead of 500 ms	Can sending a message more often fight off interference? Why or Why not?

Encourage learners to count the number of 'Hello!' messages vs. '###' messages that the receiver is displaying on the screen. After the groups have collected enough information, use the following prompt to debrief their findings:

- What did you notice when you changed how fast the jammer sent signals?
- What about when you changed how fast the sender sent signals?

Part D: Investigate physical barriers

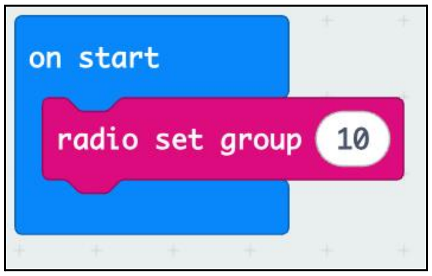
To motivate the role that different materials can play in blocking or weakening radio signals, use some of the following prompts to spark learners' thinking and discussion.

- What could we use to block the signal coming out of the jammer?
- Have you ever noticed your phone or Wi-Fi acting weird in certain places? What do you think causes that?
- What kinds of materials might interfere with or weaken a signal?
- Could we protect a device by covering it with something? What would work best?
- Do metal, plastic, or fabric affect radio signals in the same way? How could we test that?
- Can we design something that helps the receiver ignore the jammer?

Provide materials learners can test to block the signal. Aluminum foil works especially well and can completely block the signal—try wrapping the jammer with it.

Part F: Explore radio channels (frequency hopping).

Point out the code block that sets the radio channel (radio group) on the sender, receiver, and jammer.



```
on start
    radio set group 10
```

Explain that radio signals can travel on different "channels," like walkie-talkies or Wi-Fi networks. Devices on the same channel can talk to each other—but devices on different channels can't. This feature can be used to reduce interference or even protect messages from being jammed.

Introduce the idea of **frequency hopping**—a strategy where devices quickly switch between channels to avoid interference or jammers. While real systems do this automatically, learners can simulate it manually or program simple channel changes into their code.

Notes to the Presenter

Discuss with learners:

- How could we use radio channels to protect a message from being jammed?

- What might happen if two devices are on different channels?
- How could switching channels help us avoid interference from a jammer?
- How could we test whether changing the radio group makes communication more reliable?
- What challenges might come up when sender and receiver switch channels?

Encourage learners to experiment by assigning different radio groups and observing how it affects communication.

Troubleshooting:

- Check that all devices are on the same radio group. If they're on different channels, they won't communicate or jam each other.
- Make sure the jammer is turned on. Press Button A to start jamming; it should show a different icon when active.
- Try moving the jammer closer to the receiver. Jamming works best at short distances.
- Check battery levels. Weak batteries may cause delays or failure in sending/receiving signals.
- Restart or reset the micro:bits. Sometimes restarting helps if devices stop responding.
- Check your code. Make sure the sender is broadcasting, the jammer is looping properly, and the receiver is listening for messages.

Conversational Prompts

- What did you notice when you changed how fast the jammer sent signals?
- How does signal jamming relate to real-world systems like Wi-Fi or drones?
- If you wanted to design a system that resists jamming, what would you try?

Content Background

Signal jamming is a form of interference that occurs when a device intentionally disrupts radio communications by transmitting signals on the same frequency. When two or more signals occupy the same channel, the original message may be lost, delayed, or distorted. This concept is especially important in wireless communication, where signals travel through the air and are more vulnerable to disruption.

In the real world, signal jamming has both protective and malicious uses. For example:

In military settings, jammers may block enemy communications or GPS signals to prevent coordination. This was the case during World War II, when radio-controlled torpedoes could be easily jammed to go off course. Hollywood actress Hedy Lamarr and pianist George Antheil used a player piano mechanism to design a frequency hopping system.

Today, theaters and exam halls sometimes use jamming devices to discourage phone use. For homeland security communication systems use techniques like frequency hopping (rapidly switching channels) or encrypted signals to protect against jamming. Self-driving cars use frequency hopping to enable multiple cars to operate at the same time. Emergency services and aviation rely on secure, interference-resistant systems to maintain communication during critical moments.

Physical barriers also play a role in managing and protecting against interference. Radio waves can be absorbed, reflected, or weakened by certain materials—especially metals. For instance, Faraday cages (enclosures made of conductive material) are used in scientific labs, secure buildings, and some vehicles to shield sensitive equipment from external radio signals. In everyday life, this is why phones may lose signal in elevators or basements with thick concrete or metal walls. In some secure environments, materials are intentionally used to block or limit wireless communication to prevent unauthorized access or ensure privacy.

Micro:bits communicate wirelessly using radio signals. Each signal is sent on a radio group, which acts like a communication channel. Only devices set on the same group can talk to each other. If two groups are different, they can't hear each other's messages—this is useful for avoiding interference. In the real world, devices like Wi-Fi routers and Bluetooth headphones also use different channels to avoid clashing signals. Some advanced systems even use a strategy called frequency hopping, where the devices rapidly switch between channels to avoid jamming or interference. In this activity,learners can simulate that idea by changing the channel manually or through code, and investigate whether this helps protect a signal from being jammed.

List of Terms Related to this Activity

Frequency: This is the rate at which a radio signal oscillates. Different frequencies can carry different signals.

Frequency hopping: A technique used to avoid jamming by rapidly switching between channels.

Interference: When two or more signals overlap that make it difficult to receive or send information clearly.

Receiver: A device that listens for and receives transmitted radio signals.

Radio group (channel): A shared setting that determines which micro:bits can communicate with each other.

Radio signal: A type of electromagnetic wave used to transmit data wirelessly.

Signal jamming: The intentional disruption of communication by sending interfering signals on the same frequency. When a strong signal blocks or scrambles another signal to messages can't get through.

Transmission: The act of sending data (a message) from one device to another.

Transmitter (Sender): A device that sends out a radio signal.

Physical barrier: A material that can block, interfere or weaken a radio signal (e.g., aluminum foil, metal, or thick walls).