

Networking with the BBC micro:bit

The following activities use BBC micro:bits to introduce you to some of the core principles of wireless (and wired, to be fair) networking. These principles include:

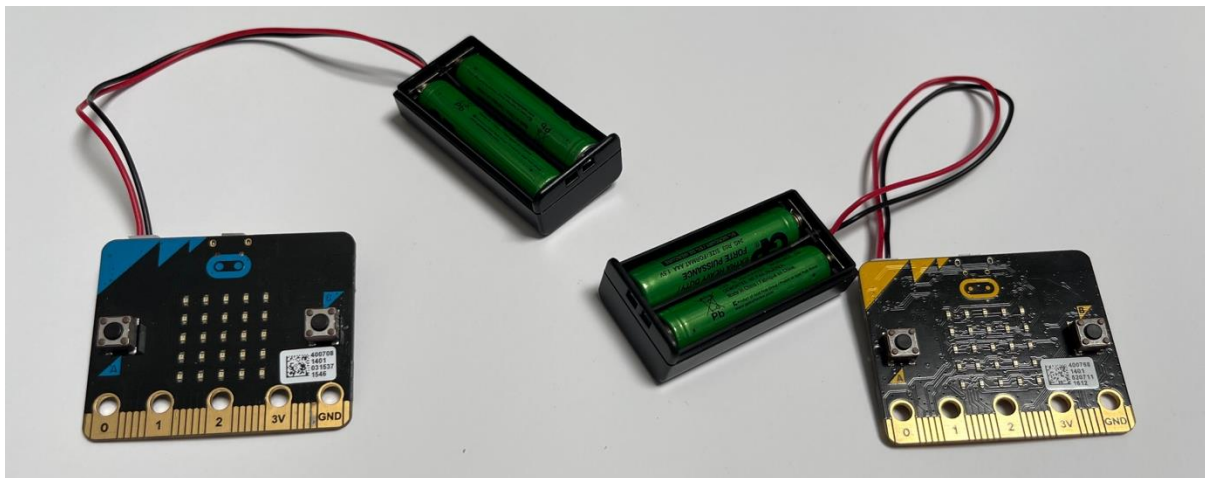
- **Broadcast** – transmit from one to all
- **Multicast** – transmit from one to many
- **Unicast** – transmit point to point, from one device to another specific device.

How to run the activities

Ideally, the activities should be run with those attending split into groups of two. They can be completed individually but educationally; more benefit will be gained using groups.

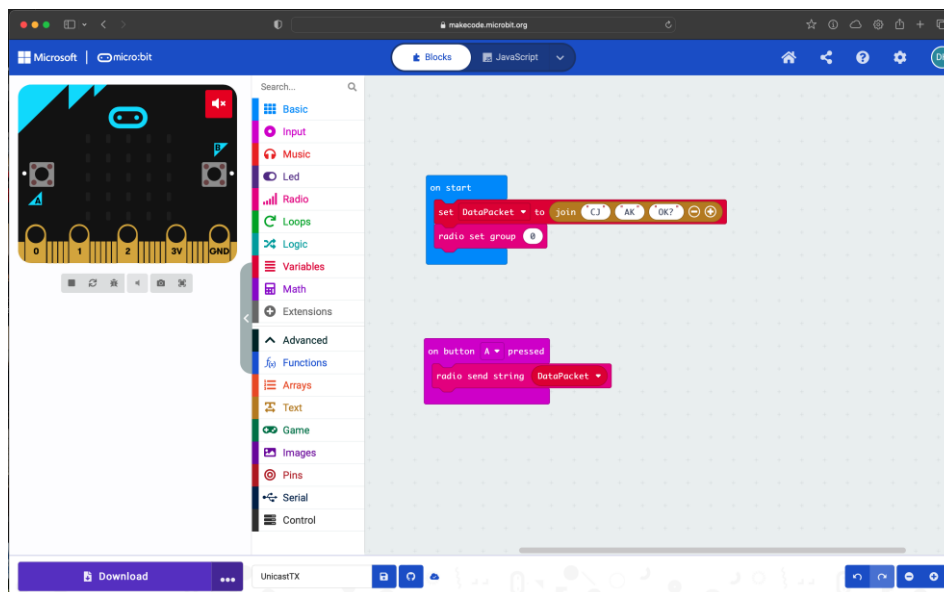


Kit List



For all activities each group will need:

- 2 micro:bits
- 2 battery holders, and 4 AAA batteries – insert batteries and connect a battery pack to each micro:bit
- Whiteboard, marker pens and a board eraser
- 1 teammate
- A computer and web browser – Safari works fine on a Mac
- Access to the online MakeCode Editor - <https://makecode.microbit.org> – make sure you are logged in before we begin.



Prerequisites

It is expected that those attending have a basic understanding of how BBC micro:bits work and that they will complete all of the activities in order.

Activity 1 - Wireless Broadcast

Introduction

Wireless (radio) communication, for example Wi-Fi and mobile phones, is a popular way to connect to the internet. In this activity we will connect our micro:bits using their built-in radio transceivers.

By doing this, you will not only learn how to use your micro:bit's radio but also, broadcast messages. Wireless communication is typically broadcast: one micro:bit can send messages to all micro:bits. In summary, this activity covers:

- Wireless communication and how to configure the micro:bit radio.
- The concept of broadcast and broadcast address.
- When broadcast is useful, and when it isn't.

Background

Wireless communication uses electromagnetic radiation - radio waves and microwaves - to send information. Radio waves are essentially electromagnetic waves radiating from an antenna (like the antennas of a WiFi router). So, wireless communication is always broadcast. In other words, the signals from the WiFi routers can be heard by other WiFi devices tuned into the same radio frequency.

Broadcast

The message of a single sender is transmitted to all receivers in a network.

However, receivers may refuse to receive broadcast messages if they are not labelled with a broadcast address.

Broadcast Address

A special address which says all devices in the network should receive this message.

In a micro:bit, the broadcast address can be configured by setting the group ID of micro:bit's radio. All the micro:bits need to have the same group ID for the broadcast to work.

In this activity, you will learn how you can receive a message from a broadcasting micro:bit and send broadcast messages yourself.

Configure your micro:bit's radio

For broadcast communication to work, you need all your micro:bits to have the same radio group ID. This group ID will be the broadcast address. This is like tuning into the correct channel to receive a TV broadcast.

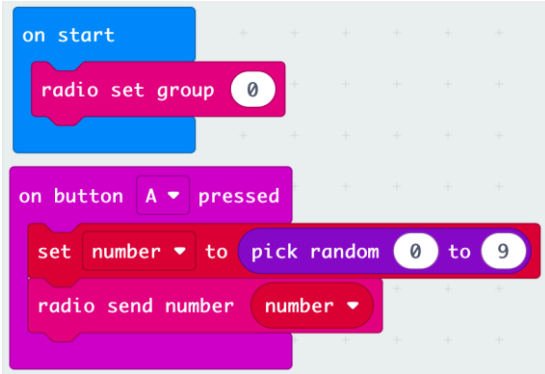
Whilst we are coding our micro:bits we will need to set the group ID to 0. To do this we will use the code block [radio set group] which can be found under the Radio group.

Transmitting and receiving broadcast messages

In this task, you will program one micro:bit to transmit a broadcast message and another to receive the broadcast message.

Programming the broadcast micro:bit

Plug the broadcasting micro:bit into your computer and make sure you can see it in your computer's file manager. Open the MakeCode Editor and create a new project called SendNumber – enter the following code.



```
on start
  radio set group 0

on button A pressed
  set number to pick random 0 to 9
  radio send number number
```

How the code works

Code	Meaning
On start	Execute the code contained within once when the program starts running.
Radio set group 0	Set the radio to receive/transmit from/to other micro:bits with a Group ID of 0 (group 0 contains all the micro:bits on the network).
On button A pressed	Execute the code within when the user presses button A (left).
Set number to pick random 0 to 9	Creates a variable called Number and places a randomly generated number between 0 and 9 in it.
Radio send number number	Transmit the contents of the Number variable.

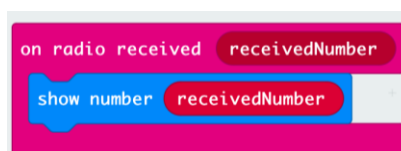
Download the code. This will save the file in the Downloads folder on your computer. Drag the microbit-SendNumber.hex file from the Downloads folder onto the micro:bit using your computer's file manager.

The yellow LED next to the USB socket on the micro:bit will flash whilst the file is being uploaded and stop when complete.

Unplug the broadcasting micro:bit from your computer, it should continue to function on the battery pack.

Programming the receiving micro:bit

Plug the receiving micro:bit into your computer and make sure you can see it in your computer's file manager. Open the MakeCode Editor and create a new project called ReceiveNumber – enter the following code.



How the code works

Code	Meaning
On radio received receiveNumber	When a data packet is received by the radio place the data in a variable called receivedNumber and execute the code contained within.
Show number receivednumber	Display the contents of the receivedNumber variable on the micro:bit's led matrix display.

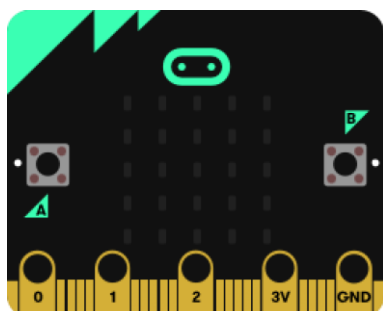
Download the code. This will save the file in the Downloads folder on your computer. Drag the microbit-ReceiveNumber.hex file from the Downloads folder onto the micro:bit using your computer's file manager.

The yellow LED next to the USB socket on the micro:bit will flash whilst the file is being uploaded and stop when complete.

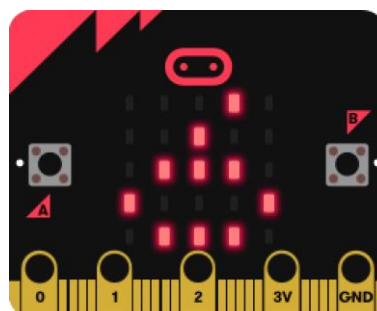
You can leave the receiving micro:bit attached to your computer.

Does it work?

Every time button A on the broadcasting micro:bit is pressed, the number it generated, and broadcasted message should appear on the receiving micro:bit's led matrix display.



Broadcasting micro:bit



Receiving micro:bit

Knowledge check

If you had a third micro:bit and uploaded the microbit-ReceiveNumber.hex file to it too, what do you think would happen when you pressed the A button on the broadcasting micro:bit?

Extension task

Discuss the issues with broadcast transmissions on a wireless network. When would it be useful? Is broadcasting always useful? What about privacy? Is it a problem that all devices on a network receive all messages?

Activity 2 – Wireless Multicast

Introduction

In the previous activity, you experimented with broadcast: sending messages to everybody.

In this activity, you will learn about sending a message so that it just goes to a smaller group of people. This is an activity that is best carried out with a large group of friends or class mates so that you can all experiment with different groups and group sizes.

Group communication (also known as multicast) is an interesting concept and enables several of today's internet technologies. For example, it enables sending videos as fast as possible over the Internet. In this chapter, you will learn:

- The concept of group communication and group or multicast address.
- When group communication is useful and when it isn't.

Background

In the previous activity, all micro:bits received messages from all the broadcasting micro:bit. With the right code loaded, any micro:bit could be the broadcast or receiving micro:bit. Now, let's try limiting who you can send messages to and receive messages from. This is called group communication. Group communication is used on the Internet to send data to many people at the same time. For example, Internet television and video conferencing use group communication.

Group/multicast communication

A message is only sent to the computers in a specific group.

For this, the messages need to be labelled with a group or multicast address.

Group address

A special address which says all devices in the group should receive this message.

In this activity, you need to work together in pairs or small groups, with at least 2 micro:bits in each group. You will complete two tasks to program your micro:bits to send messages to and receive messages from your group.

Create groups

In this task, you will choose a unique group ID for your group. It is especially important that your group number is unique if there are more than one group working within range of each other. What would happen if two groups choose the same group ID? Plan your network, decide which group is going to use which group ID, and note it down on the whiteboard so that everyone knows what they are doing.

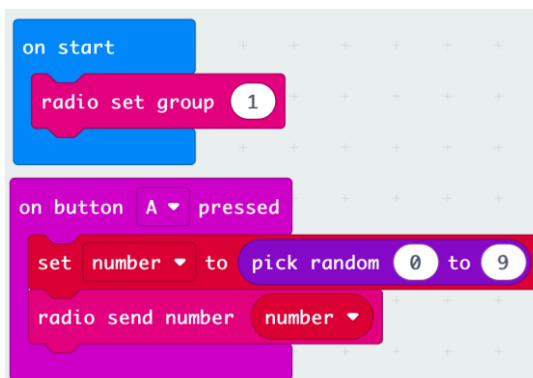
Use the whiteboard to record who is using what addresses so that no two groups of micro:bits are using the same group ID.

Setting up group transmission/multicasting

Select one micro:bit as the multicasting micro:bit and one as the receiving micro:bit.

Programming the transmitting micro:bit

Plug the multicasting micro:bit into your computer and make sure you can see it in your computer's file manager. Open the MakeCode Editor and create a new project called SendToGroup – enter the following code.



This program uses the same code from the first activity, the only change being the use of radio group 1 (you use whichever group ID you were assigned) – only other micro:bits assigned the same group ID will be able to receive the transmitted data packets.

Download the code. This will save the file in the Downloads folder on your computer. Drag the microbit-SendToGroup.hex file from the Downloads folder onto the micro:bit using your computer's file manager.

The yellow LED next to the USB socket on the micro:bit will flash whilst the file is being uploaded and stop when complete.

Unplug the multicasting micro:bit from your computer, it should continue to function on the battery pack.

Programming the receiving micro:bit

Plug the receiving micro:bit into your computer and make sure you can see it in your computer's file manager. We don't need to create a new project for this task.

All we need to do is drag the microbit-ReceiveNumber.hex file, that we created in the first activity, from the Downloads folder onto the micro:bit using your computer's file manager. Remember, this program has no code to set the group ID as it was created to receive broadcast messages.

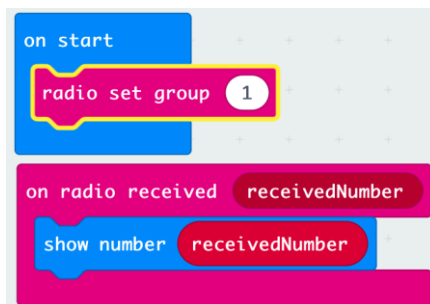
The yellow LED next to the USB socket on the micro:bit will flash whilst the file is being uploaded and stop when complete.

You can leave the receiving micro:bit attached to your computer.

Every time button A on the multicasting micro:bit is pressed, a number is generated, and multicast to all micro:bits set up with a Group ID of 1. You should notice that unlike in the first activity, nothing is displayed on your receiving micro:bit's led matrix display. This is because the receiving micro:bit is not a member of group 1.

Reprogramming the receiving micro:bit

Plug the receiving micro:bit into your computer and make sure you can see it in your computer's file manager. Open the MakeCode Editor and create a new project called ReceiveFromGroup – enter the following code.



How the code works

Code	Meaning
On start	Execute the code contained within once when the program starts running.
Radio set group 1	Set the radio to receive/transmit from/to other micro:bits with a Group ID of 0 (group 1 means only data transmitted by other micro:bits with a group ID of 1 will be received).
On radio received receiveNumber	When a data packet is received by the radio place the data in a variable called receivedNumber and execute the code contained within.
Show number receivednumber	Display the contents of the receivedNumber variable on the micro:bit's led matrix display.

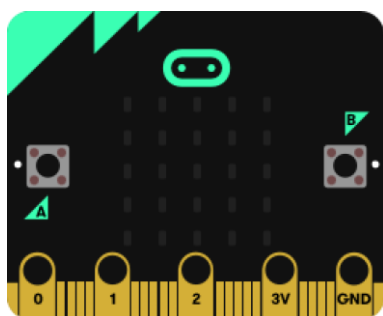
Download the code. This will save the file in the Downloads folder on your computer. Drag the microbit-ReceivedFromGroup.hex file from the Downloads folder onto the micro:bit using your computer's file manager.

The yellow LED next to the USB socket on the micro:bit will flash whilst the file is being uploaded and stop when complete.

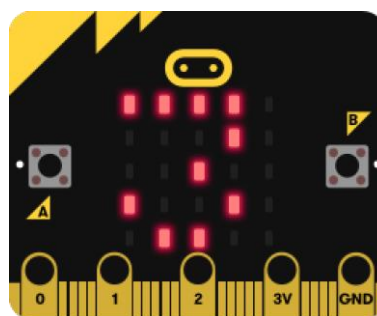
You can leave the receiving micro:bit attached to your computer.

Does it work?

Every time button A on the multicasting micro:bit is pressed, a number is generated, and multicast to all micro:bits set up with a Group ID of 1. The multicast message should appear on the receiving micro:bit's led matrix display.



Broadcasting micro:bit



Receiving micro:bit

Knowledge check

Compared to broadcast, fewer receivers on a network receive a multicast message. True or False?

Extension task

Can a micro:bit be a member of two or more groups? How would you program your micro:bit to do that?

Activity 3 – Wireless Unicast

Introduction

Unicasting is the transmission of messages to a single receiver. It is the typical way we communicate on the internet. For example, to view a web page, we send unicast messages to a server, which in turn send us the page to display on our browser. In this activity, you will send unicast messages to a micro:bit. Doing this, you will learn some basic ideas of computer networking, including:

- the concept of unicast
- the concept of a protocol
- the concept of an address and IP address
- the concept of a data packet and a header

Background

This chapter covers unicast communication.

Unicast

Transmission of a message to a single receiver.

When sending messages to each other, computers use protocols.

Protocol

A set of rules for how messages are sent across networks.

Simply, protocols define how computers should send messages and what they should do when they receive a message. On the internet, every computer or device follows the Internet Protocol (IP). According to Internet Protocol, each device is given a unique address, called an IP address. Remember you have already used special addresses for broadcast and multicast. In this chapter, we consider unicast addresses. IP addresses are used for unicast on the internet.

IP addresses

A unique set of numbers that identify computers that use the Internet Protocol to communicate over a network. This string is made up of 4 decimal numbers (called octets), that range between 0 and 255. Each octet is separated by dots. For example, 213.248.234.11 is an IP address.

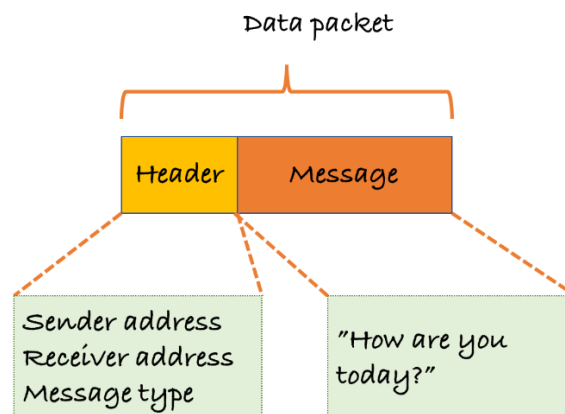
Your micro:bit has a unique address to identify it on a network, but it is a bit different. You already partly changed your micro:bit's address, by changing the group ID.

When two computers communicate, the sender sends a data packet to the receiver.

Data packet

A data packet is a piece of data sent over a network. This piece of data has an actual message part (for example, an image or a text) and one or more header parts. A header contains helpful information for protocols like the sender and receiver IP addresses.

It is important that the header contains at least the sender's and receiver's IP address. This is so that the message can be delivered to the correct recipient and so that the recipient can respond to the sender.



In the example above, as well as the sender and receiver addresses, the header also includes a message type. Message type tells the receiver what type of data is being sent, for example, a text or an image.

In the previous activities you programmed your receivers to receive a specific type of message, a number. If the broadcast/multicast micro:bit sent an image, and the receiving micro:bit was set up to receive a number, the receiving micro:bit would be unable to receive the image. Having a message type in the header of the data packet enables us to develop more advanced code that could handle many different types of data. In this activity, in order to unicast to other micro:bits, and create a data packet by adding a header with source and destination addresses.

To receive any packet, sent by anybody, we need to use broadcast as the underlying communication.

Design your header

The sender micro:bit needs to attach a header to the start of each message before sending and will include:

- Sender address
- Receiver address

First pick two-character strings as addresses for your sending and receiving micro:bit addresses.

In this activity we will use CS for the sending micro:bit and AK for the receiving micro:bit. The addresses you choose should be unique to those on all the other micro:bits that are near to you.

Use the whiteboard to record who is using what addresses so that no two micro:bits have the same address.

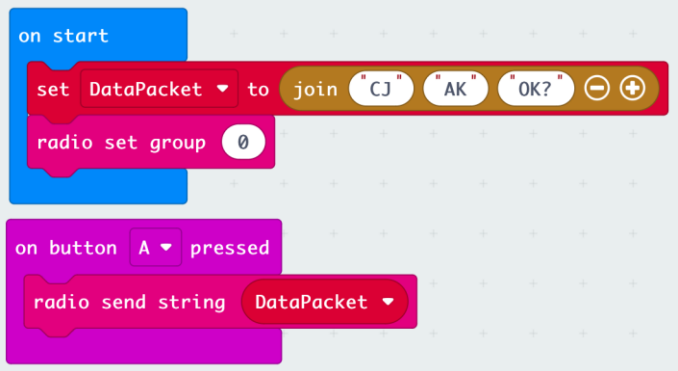
Setting up your unicasting micro:bits

Select one micro:bit as the sender micro:bit and one as the receiving micro:bit.

Programming the sending micro:bit

Plug the multicasting micro:bit into your computer and make sure you can see it in your computer's file manager. Open the MakeCode Editor and create a new project called UnicastSender.

Now create the following code.



```
on start
  set DataPacket to join "CJ" "AK" "OK?"
  radio set group 0

on button A pressed
  radio send string DataPacket
```

How the code works

Code	Meaning
On start	Execute the code contained within once when the program starts running.
Set DataPacket to join "CJ" "AK" "OK?"	Creates a variable called DataPacket. Store the text CJAKOK? In the variable. CJ is the address of the sending micro:bit in the data packet's header. AK is the address of the receiving micro:bit in the data packet's header. OK? Is the message.
Radio set group 0	Set the radio to receive/transmit from/to other micro:bits with a Group ID of 0 (all of them, Group ID is the broadcast group – Activity 1).
On button A pressed	Execute the code within when the user presses button A (left).
Radio send string DataPacket	Transmit the contents of the DataPacket variable (CJAKOK?).

Download the code. This will save the file in the Downloads folder on your computer. Drag the microbit-UnicastSender.hex file from the Downloads folder onto the micro:bit using your computer's file manager.

The yellow LED next to the USB socket on the micro:bit will flash whilst the file is being uploaded and stop when complete.

Unplug the sending micro:bit from your computer, it should continue to function on the battery pack.

Programming the receiving micro:bit

Plug the receiving micro:bit into your computer and make sure you can see it in your computer's file manager. Open the MakeCode Editor and create a new project called UnicastReceiver – enter the following code.

```

on start
  radio set group 0

on radio received receivedString
  set From to join char from receivedString at 0 char from receivedString at 1
  set To to join char from receivedString at 2 char from receivedString at 3
  if To = AK then
    set Message to join char from receivedString at 4 char from receivedString at 5 char from receivedString at 6
    show string Message
  
```

How the code works

Code	Meaning
On start	Execute the code contained within once when the program starts running.
Radio set group 0	Set the radio to receive/transmit from/to other micro:bits with a Group ID of 0 (all of them, Group ID is the broadcast group – Activity 1).
On radio received receivedString	When a data packet is received store the contents of it in a variable called receivedString and execute the code contained within.
Set From ...	<p>Store the results of the following instructions in a variable called From.</p> <p>This will be the address of the sending micro:bit in the header.</p>
Join ...	Join all the following pieces of text together.
Char from receivedString at 0	The first character (to computers, 0 is the first value, 1 the second, etc.) of the received header.
Char from receivedString at 1	The second character of the received header.
Set To ...	<p>Store the results of the following instructions in a variable called To.</p> <p>This will be the address of the receiving micro:bit in the header.</p>
Join ...	Join all the following pieces of text together.
Char from receivedString at 2	The third character of the received header.
Char from receivedString at 3	The fourth character of the received header.
If To = "AK"	<p>If the To variable contains the text AK (the address of the receiving micro:bit) then this message is for us and execute the instructions contained within.</p> <p>If the To variable didn't contain the address of the receiving micro:bit then the instructions contained within would not be executed and the message would just be ignored – it's not for us.</p>
Set Message ...	<p>Store the results of the following instructions in a variable called Message.</p> <p>This will be the message part of the data packet.</p>
Join ...	Join all the following pieces of text together.
Char from receivedString at 4	The first character of the received message.
Char from receivedString at 5	The second character of the received message.
Char from receivedString at 6	The third character of the received message.

Show string message	Display the contents of the Message variable on the micro:bit's led matrix display.
---------------------	--

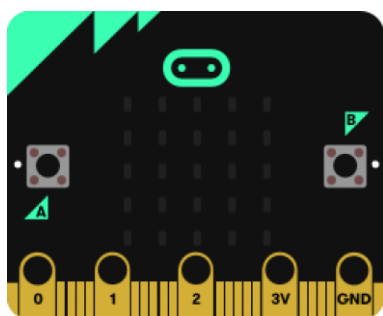
Download the code. This will save the file in the Downloads folder on your computer. Drag the microbit-UnicastReceiver.hex file from the Downloads folder onto the micro:bit using your computer's file manager.

The yellow LED next to the USB socket on the micro:bit will flash whilst the file is being uploaded and stop when complete.

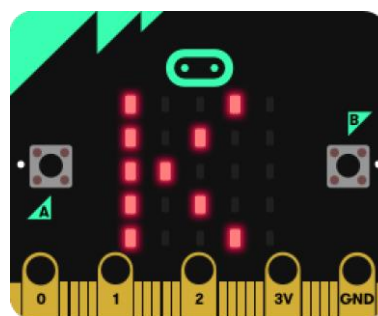
You can leave the receiving micro:bit attached to your computer.

Does it work?

Every time button A on the multicasting micro:bit is pressed, a data packet is transmitted by the sending micro:bit, received by the receiving micro:bit, determined whether the data packet was addressed to the receiving micro:bit, and if it was the message OK? is displayed. Note that because the message is too large to fit on the small display, the message will scroll horizontally and then the display will go blank.



Broadcasting micro:bit

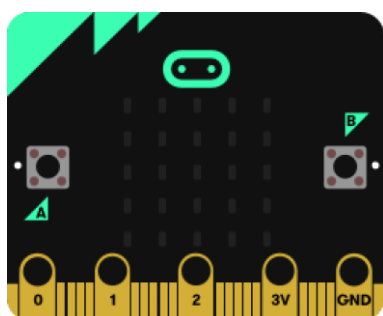


Receiving micro:bit

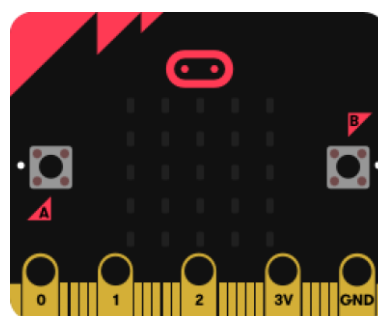
Further proof it works

Change the code in the UnicastSender program so that the address of the receiving micro:bit is something other than AK.

Upload the code to the sending micro:bit, press button A on it and notice that the message is not displayed on the receiving micro:bit – the data packet wasn't addressed to it, so it ignored it.



Broadcasting micro:bit



Receiving micro:bit

Knowledge check

What is the need for all three types of sending data (casting) on a network.

Extension task

There are two extension tasks for this activity.

Firstly: you could program the two micro:bits so that ... when the receiving micro:bit receives the message OK? From the sending micro:bit, it sends the message Yes back to the sending micro:bit.

Secondly: try changing the code on the receiving micro:bit and listen out for packets intended for other receiving micro:bits in the room. Is there a use case for this? Is it the right thing to do? How could you protect your messages from being intercepted by other users on the network.

Knowledge check answers

Activity 1 – Both micro:bits coded with the microbit-ReceiveNumber.hex program would receive the number transmitted by the broadcasting micro:bit.

Activity 2 – True. Since a multicast message is only sent to a sub-group of receiving micro:bits, rather than all micro:bits, multicast messages are received by fewer micro:bits.

Activity 3 – Broadcasting allows messages to be sent to all users of a network. The UK Government is now using this method of messaging users to alert them of major weather and civil events. Multicasting allows messages to be used to select members of a network that are members of the same group. Unicasting allows point-to-point communication between two users – just as we use when we use the internet.

Legal stuff

Copyright

Original work - Copyright © 2017, Nominet

Adaptation of original work – Copyright © 2023, Radio Society of Great Britain

Authors

Original authors: Cigdem Sengul and Anthony Kirby.

<https://microbit.nominetresearch.uk/networking-book/>

Work adapted by Derek Hughes (G7LFC) – RSGB Learning Team

www.rsgb.org/coding

Licence

This work is made available under the Creative Commons Attribution-ShareAlike 4.0 International License.

To view a copy of this licence, visit <http://creativecommons.org/licenses/by-sa/4.0/>.

In summary, you are free to:

- Share — copy and redistribute the material in any medium or format.
- Adapt — remix, transform, and build upon the material for any purpose, even commercially.

Under the following terms:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Acknowledgements

“BBC”, “micro:bit” and the micro:bit emojis are trademarks of the BBC.