

DT Mini Challenge Year 5/6 Blockly Micro:bit Networking

<https://groklearning.com/course/aca-dt-mini-56-bk-microbit-radio/>

About this activity

In this challenge, students will learn how to program the micro:bit's radio to communicate between devices.

This is a DT Mini Challenge - it moves through content faster than other DT Challenges, getting through similar amounts of material in less than half the time.

Networked devices are an essential component of digital systems. They impact the world through their use in remote sensing, entertainment systems, monitoring and communication systems. They are used in diverse applications, from consumer electronics like gaming consoles, smart homes, baby monitors and wearables, to industry and research purposes, like relaying remote sensor information from farm equipment and ocean buoys or storing and transmitting research data from satellites.

The BBC micro:bit is a simple but powerful programmable computer that has several built-in peripheral devices and sensors, along with a bluetooth radio that allows it to easily communicate with other micro:bits.

This challenge uses radio communication as a motivator to learn programming concepts, while exposing students to the concept of networking within digital systems.

Programming concepts

- Algorithms
- Variables
- Branching (if statements)
- User inputs - buttons, temperature, receiving messages
- Boolean logic
- Data manipulation

For further resources that might extend, enrich or deepen the classroom experience or support assessment, we recommend visiting the Digital Technologies Hub at <https://www.digitaltechnologieshub.edu.au>.

Age

This challenge targets students in years 5/6, though it can also be used as an introductory course for students in later years who have not yet been exposed to basic programming concepts.

Language

Blockly, a visual programming language designed for teaching students programming concepts.

Time

The course is designed to be completed in approximately 4-6 hours of class time.

Objectives, Content Descriptions & Key Concepts

Digital Technologies

Content Descriptor Code	Content Descriptor	Key Concepts	Addressed by Micro:bit networking through:
ACTDIK014	Examine the main components of common digital systems and how they may connect together to form networks to transmit data	Digital Systems	Connecting devices through radio.
ACTDIK015	Examine how whole numbers are used to represent all data in digital systems	Data Representation	Learn that the pictures on an LED display can be controlled with numbers.
ACTDIK016	Acquire, store and validate different types of data, and use a range of software to interpret and visualise data to create information	Data Collection	Collect temperature data from a remote sensor. Validate data by filtering strings
ACTDIP017	Define problems in terms of data and functional requirements drawing on previously solved problems	Specification	Decompose problems into smaller components.
ACTDIP018	Design a user interface for a digital system	Interactions	Create a user interface using the LED screen to display information and the buttons to input information.

ACTDIP019	Design, modify and follow simple algorithms involving sequences of steps, branching, and iteration (repetition)	Algorithms	<p>Students follow, design and modify algorithms to solve problems.</p> <p>Most problems require branching and iteration.</p>
ACTDIP020	Implement digital solutions as simple visual programs involving branching, iteration (repetition), and user input	Implementation	<p>Implement algorithms to solve problems.</p> <p>Students implement a digital solution using programming constructs within the blockly paradigm.</p>

What are we learning? (Abstract)

At the conclusion of these activities students will be able to:

- Write programs using a visual programming language
- Recognise that steps in algorithms need to be accurate and precise
- Utilise branching (if statements) in programs
- Utilise user input
- Utilise variables
- Define the term *algorithm*
- Define the term *branching*
- Define the term *state*
- Understand that data can be transmitted wirelessly between devices

Module outline

The course consists of two core modules:

1. Radio send and receive

This module introduces displaying text on the micro:bit's 5x5 LED display, loops and user input through pressing buttons.

It shows how to send and receive information over the radio.

It uses that information to create a remote temperature sensor.

2. Images and states











This module introduces displaying images on the micro:bit, and configuring the radio by changing channels.

It then shows how to filter information by checking received images, and introduces changing the program's state to give the user greater control.

New Vocabulary

Algorithm: A set of rules or step by step instructions to solve a problem or achieve an objective. A recipe is an example of an algorithm - it sets out what you need and the steps you follow to combine everything to create your food item(s).

Branching: Changing the instructions executed by the program based on a certain condition. This allows you to specify that your program should behave one way in some cases, but a different way in others. In blockly, this is achieved through the use of `if` blocks.

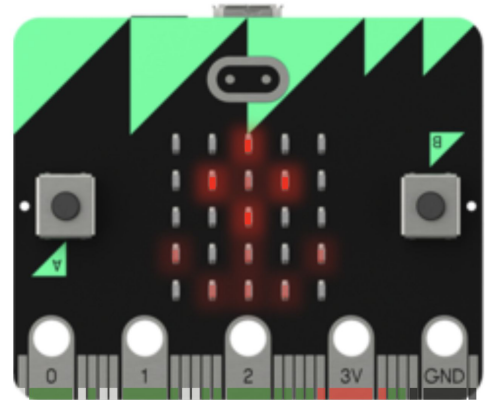
Types of component:			
 Discussion	 Worksheet	 Plugged Activity	
 Group Activity	 Unplugged Activity	 Video	
 Animation	 Reflection	 Game	 App

Overarching Activity: The Network Radio Challenge

Preparation and timing

The challenge consists of 2 modules, with an additional extension investigation for students who have had some experience with programming before or who move through the material quickly.

Completing both modules should take around 4-6 hours.



Overview

- The challenge assumes no programming knowledge
- It teaches programming concepts, radio networking and blockly syntax concurrently.

Suggested Implementation



Plugged Activity

Challenge modules

To get the most out of the modules, students should always:

- Read the interactive notes, including running any example code provided
- Attempt all problems and review questions

Clicking the **Run** button (▶) allows students to check their code by running it and observing the output. When they believe they have the solution correct, pressing the **Mark** button (★) will check to see if the code passes the test cases, and will provide feedback if it does not.

You can interleave the unplugged activities above with completion of the online modules, especially if you find students are struggling with a concept explored in one of the activities.

Many slides and all problems include “Teacher Notes” that verified teachers can access. These provide additional information, suggestions and activities for teaching each concept and exploring ideas further with students in class.

Activity 1 — Introducing the micro:bit



Plugged Activity

[Challenge Module 1 - Radio send and receive](#)

Module 1 introduces the students to the micro:bit and gives them the programming tools needed to send and receive information over the micro:bit radio. It should make them think about how they interact with computers, and how user inputs, control the flow of data.

Focus point: Draw the students attention to the “Put it together!” slide. At this point students will be able to use micro:bits to physically send and receive information across the classroom.

Activity 2 — Unplugged: Abstract Drawing

Preparation and timing

Before starting this activity, the students will need four pieces of plain paper (post-its work well) and a print out of the 5 X 5 grid (linked below).

This activity can be done prior to or during Module 1, which covers how images are shown on the micro:bit display. It could also be done during Module 2, where custom images are introduced.

Overview

- How much information is really needed to convey an idea?
- Abstraction requires the removal of unnecessary information.
- Can an animal be drawn with 25 dots?

Suggested Implementation

Unplugged Activity

Abstract Drawing

Give the students 2 minutes to draw a pig on one piece of paper and a dog on another one.

The short time frame is to try to stop them from trying to draw something too detailed.

Ask the students to show only one picture to the student next to them. Can they guess which animal it is? (encourage a little discussion about funny drawings)

Discussion

Did anyone fail to identify the drawn animal? What were the things that helped you to identify the difference between a pig and a dog?

Suggestions should include curly tail, snout, perhaps long waggy tail, tongue, pointy or floppy dog ears.

Take 2:

Try the activity again, this time get the students to see if they can represent the pig and the dog on the two remaining pieces of paper with the fewest lines possible.

Have them show the pictures to their partner to guess. Was it as easy? Did anyone do a particularly good job of representing a pig or a dog in a tiny number of lines?

 **Discussion**

Talk about the fact that the students have abstracted all but the most identifiable details away. Computer science is all about abstraction — making sure we ignore the unimportant and distracting detail to focus on what's necessary to solve a problem.

As humans, we abstract details all the time: if someone asks us how we are in the morning, we generally only give them a short response (e.g. “good, thank you”). We don't go into the details of the toe we may have stubbed, or the bus to school that braked suddenly causing you to stumble on the way out. Similarly, for computers to solve problems effectively, they need the solution to be general (or abstract) enough that it can solve for all problems of a certain type. Imagine if a calculator could only add 2+2 and nothing else!

Faces in 25 dots

Ask the students to suggest some facial expressions that have very identifiable characteristics.

Suggestions should include things that are represented by emoji like happy, sad, surprised, angry etc.

Get the students to colour squares on the 5 x 5 grid to represent an animal that their partner can guess.

- [Link to the 5 x 5 grid printable \(Google\).](#)
- [Link to the 5 x 5 grid printable \(PDF\).](#)

Rules:

1. They may only use one colour
2. They must colour the whole square of any square they colour

Show the pictures and ask for guesses from the group. (Encourage enthusiastic showing around the room.)

Discussion questions:

- Was it hard to abstract away the least important information?
- Was it easier or harder when there was a very specific limit for how much you could draw?
- Extension: what are some other examples of abstraction (either in how humans communicate or how computers work)?

Activity 3 — Branching: “Simon says ...”

Preparation and timing

For very young students you may need to explain and/or demonstrate the rules of [Simon Says](#). Most students will have played this game before.

This activity could be delivered prior to or concurrently with Module 1.

Overview

- What is branching?
- How does it relate to programming?
- How do we determine the conditions that we need to check to perform different instructions?

Suggested Implementation

 **Unplugged Activity**  **Group Activity**

Simon says...

Ask the student if they’ve all played “Simon says” before and, if any of them haven’t, explain the rules of the game. You should then play a game with the group. Older students may not make many mistakes, and you may choose to end the game early if they aren’t being knocked out very quickly.

You should then say that you’ll play another game, but this time make the rule a little more difficult. Instead of the rule being *does the instruction start with “Simon Says”*, try something like *does the instruction include the word “you”* or *does the instruction have exactly five words in it*. You are demonstrating that the condition can change, but regardless what determines if the instruction should be performed is some condition that you’ve set as your rule.

Branching in programming is what allows us to define different behaviour in the same program. It allows us to check some value or condition in our program, and respond differently based on what the result is. Without branching, all of our programs would perform exactly the same thing every time, and we would need to write brand new programs every time the tiniest little thing changed. Our programs would not be able to respond differently to new users or data.

Blockly provides a few different ways of performing branching, and some of these are covered in modules 1 and 2 of the challenge.

 **Plugged Activity****Challenge modules 1 & 2: branching/decision questions**

In module 1, you may find it useful to walk through the examples as a group, demonstrating the use of branching for students and explaining the thinking process. You can then have students complete the problems on their own or with a peer, giving them a chance to apply the knowledge and what they've seen demonstrated.

Discussion questions:

**Reflection****Group Activity****Branching is everywhere!**

Students share some examples where they think branching occurs in the programs they use regularly, or other situations in life. Things might include:

- If the phone rings for 30 seconds and isn't answered, it automatically goes to voicemail
- If you choose a particular song on a CD or streaming app, it plays that chosen song
- If you choose a certain character in a game, then it loads that character so that you can play as that person
- If you try to use an ability in a game and it isn't charged, it doesn't let you do it
- If the user types in different answers to a question in a program, different options are presented to them for their next action

All of these actions require the program to check some value or condition. The way the computer responds is determined either by the user themselves, or due to the state of the program at the time the check is performed.