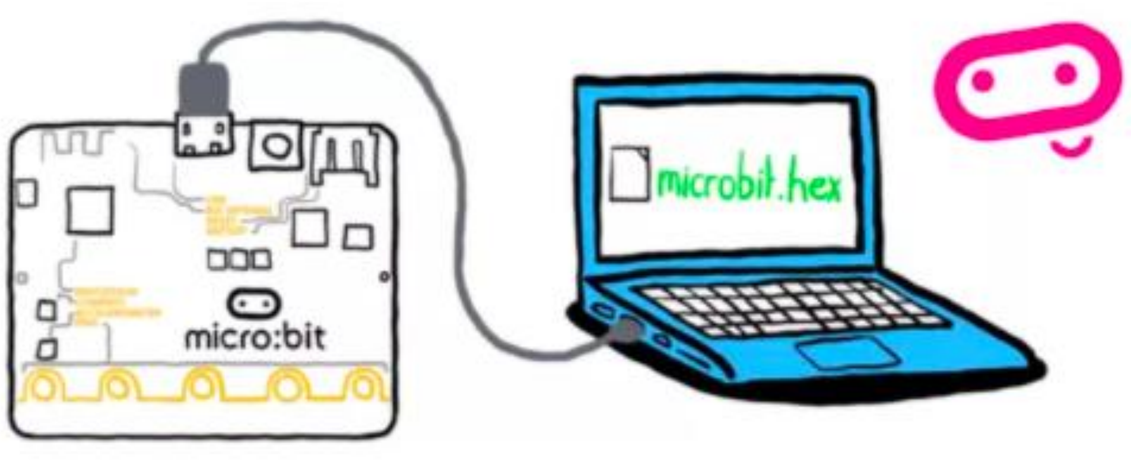
MicroMakers!

Lab Notebook

Property of this Awesome MicroMaker



Lab Notebook Contents

Introduction and Credits

Welcome!

Dear MicroMakers,

I am so honored and delighted to have this opportunity to work with and learn from you, while we solve challenges by building nifty gadgets.

In our Club, we will learn the basic principles of **physical computing** – creating devices that use sensors to collect information, process this information, and then perform an action. For example, a physical computing project may involve using sensors to detect soil moisture, sending the data to a processer to determine whether the soil is too dry, and then activating a motor to pump water into the soil. Physical computing involves interacting with the physical world, in addition to a screen, making learning to code more fun and understandable.

We will learn about physical computing using the **micro:bit** – a microcontroller (a pocket-sized computer) developed by the British Broadcasting Company (BBC) for teaching children how to code. The micro:bit has an LED light display, buttons, sensors, and many input/output features that are programmable using a variety of languages. To get started, we will be using Microsoft **MakeCode**, a visual "drag and drop" programming language designed for kids. Later, as our projects become more advanced, we will try a Python library called MicroPython, designed especially for microcontroller projects.

In MicroMakers, by making new gadgets during every meet up, we will cover the following concepts:

- Computer inputs (especially sensors), outputs, and processors
- Computer memory
- Electric circuits
- Programming fundamentals (variables and constants, functions, loops, logic, and math)
- The "making" process (problem statement, solution design, testing and iteration, documentation, and dissemination)

Thank you again -- I cannot wait to see the wonderful inventions you will make!

❤️,
Ms. Holly

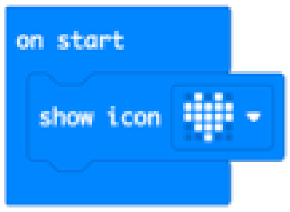
Credits and Resources

- Concepts, Slides, Script, Activities: Holly Krambeck (learningpython64@gmail.com)
- Videos, Illustrations, and Projects (unless otherwise noted): Micro:bit.org (https://microbit.org/get-started)
- More Information about Physical Computing:
    - https://en.wikipedia.org/wiki/Physical_computing
    - http://k12maker.mit.edu/physical-computing.html
    - https://www.microsoft.com/en-us/research/uploads/prod/2020/04/physical-computing.pdf
    - https://guides.temple.edu/c.php?g=419841&p=2863656

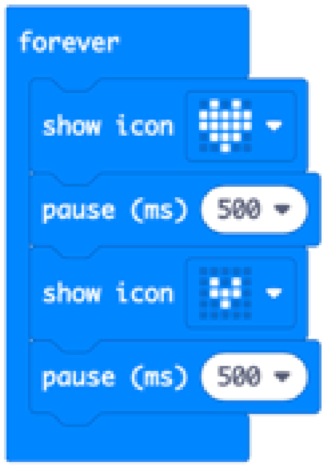
Skill Badges

Date	Badge	Description
	Micro:bit Basics	Identify key parts; recite care instructions; add and remove batteries from battery pack; plug in and remove USB and battery pack; transfer a program to the micro:bit and run; turn micro:bit and battery pack on and off; gently put away micro:bit and components.
	LEDs, Buttons, and Light Sensor	Help little kids sleep more peacefully by using your micro: bit to make a nightlight that turns on when it is dark.
	Accelerometer	Measure the length of your step (with a ruler) and get the micro:bit to multiply this length by the number of steps to calculate the distance you've walked (hint: you'll need the "math" menu). Show number of steps taken when you press button A. Reset number of steps to zero when you press button B. Show distance walked when you press buttons A+B together.
	Magnetometer	Create a snoop detector that uses a variable to count the number of times your door has been opened - you'll need to add code to sense when the door has been opened and when it's been closed.
	Pirate	Invent a treasure hunt game that utilizes both the accelerometer (e.g., a step counter / distance calculator) and the magnetometer (e.g., the compass).
	Radio	Make an indoor/outdoor thermometer. Use the micro:bit's radio to make a remote sensor that sends temperature readings to another micro:bit, for example from outside to inside.
	Secret Agent	Make a radio-controlled remote alarm so when you know someone's turned the lights on, opened a drawer or bag, and/or when someone has opened the door to your room.

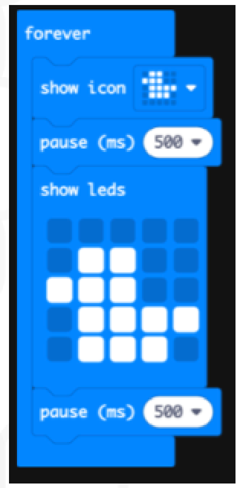
Projects

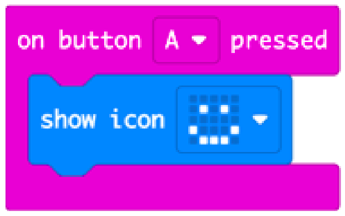
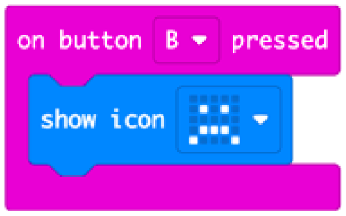
Topic	LEDs and Buttons
Project Name	**Show Your Love**
Description	Light up your micro:bit with love by showing a heart.
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional)
How it Works	• The program shows a heart picture on the micro:bit's LED display output. • The heart stays on the display as long as it has power because you haven't given it any further instructions to show anything else. • The micro:bit has a set of other built-in images you can use in your projects. • Pictures on computers, phones and tablet screens are made using dots called 'pixels'. The micro:bit's display has 25 LED pixels arranged in a 5 x 5 grid.
Code	 on start show icon
Options	• Try choosing other built-in pictures like HAPPY, DUCK or GHOST. • Can you show more than one image?
Notes	

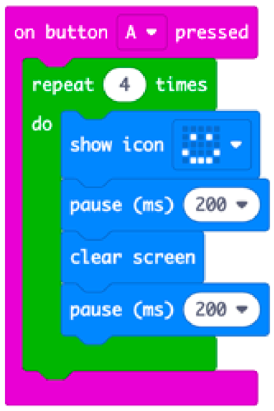
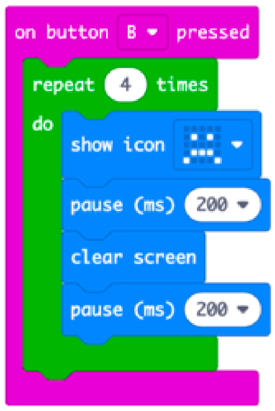
Topic	LEDs and Buttons
Project Name	**Beating Heart**
Description	Make your micro:bit’s heart beat using loops to create an animation.
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional)
How it Works	• The program shows a beating heart using two built-in pictures, a large and small heart, on the micro:bit's LED display. • Different images shown in a sequence create the illusion of movement: a heart getting larger and smaller. • After showing each image, the program **pauses** for half a second (500 milliseconds) before showing the next image. • The animation is kept going **forever** using an infinite **loop**: it repeats the sequence of showing these two images and pausing until you unplug the micro:bit. • Using loops to keep things happening is an important idea in computer programming: we have created an animation that will keep running for as long as the micro:bit has power using only a small amount of code. This is also called **iteration**

Code	 forever
	  show icon ▾
	  pause (ms) 500 ▾
	  show icon ▾
	  pause (ms) 500 ▾

Options	• Make the heart beat faster or slower by changing the delay time. • Try animating other built-in images like the small and large diamond or square. • Create your own animations using your own designs.
Notes	

Topic	LEDs and Buttons
Project Name	**Animal-ation!**
Description	Animate your own pictures on the micro:bit display.
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional) • Graph paper or LED planning sheet to sketch out your own animal designs (optional)
How it Works	• This program creates an animation on the micro:bit's LED display output by using a built-in image and one you create yourself. • First it shows the duck built-in image and then shows a modified version, which is made by moving all the dots (pixels) down one row. • It shows the two different pictures one after the other, with a half second (500 millisecond) delay, to make it look like a duck bobbing up and down on the water. • An infinite **loop** keeps the micro:bit showing the image sequence until you unplug the micro:bit. • Using loops in computer programs is also known as **iteration**. They help you create efficient compact code without needlessly repeating the same instructions.
Code	 forever show icon pause (ms) 500 show leds pause (ms) 500
Options	• Try modifying and animating different built-in images like GIRAFFE and RABBIT. • Create your own images from scratch using graph paper or our LED planning sheet to sketch out your designs. • Make longer animated sequences to tell a story. • *Advanced: In Python, use different numbers to change the brightness of different pixels. 9 is the brightest, 1 is the dimmest and 0 is off*
Notes	

Topic	LEDs and Buttons
Project Name	**Emotion Badge**
Description	Use your micro:bit to tell the world how you're feeling.
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional)
How it Works	• The micro:bit has two input buttons on the front you can use to make things happen. • When you press input button A the program reacts by showing a happy face image on the LED display output. • Pressing button B makes a sad face appear.
Code	 on button A pressed / show icon  on button B pressed / show icon
Options	• Try other emotion images such as ASLEEP, CONFUSED or MEH. • Design your own 'emoticons' using the LED display. • **Use the badge to show if you need help or make class voting badges with ticks or crosses.** • Design a way of wearing your micro:bit badge using thread, or tape. (Don't use safety pins as the metal could damage your micro:bit.)
Notes	

Topic	LEDs and Buttons
Project Name	**Animated Badge**
Description	Make your feelings really stand out with flashing happy and sad faces.
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional)
How it Works	• Like the Emotion badge project, this program shows different emotion images on the LED display output depending on which button input you press. • Loops can make sets of instructions run for ever, but here we use a numbered loop to flash the image 4 times to make it really eye-catching. • Loops are an important idea in computer programming as they save repeating the same code many times, making your program more efficient. This idea is also called **iteration**.
Code	 `on button A pressed` / `repeat 4 times` / `do` / `show icon` / `pause (ms) 200` / `clear screen` / `pause (ms) 200`  `on button B pressed` / `repeat 4 times` / `do` / `show icon` / `pause (ms) 200` / `clear screen` / `pause (ms) 200`
Options	• Make the badge flash more times by making the number 4 larger. • Make the flashing faster or slower by changing the delay of 200 milliseconds (0.2 seconds). • Make it flash forever. • Use different emotion images, or draw your own
Notes	

Topic LEDs and Buttons (+ Accelerometer)

Project Name **Silly Badge!**

Description Make your feelings really stand out with flashing happy and sad faces.

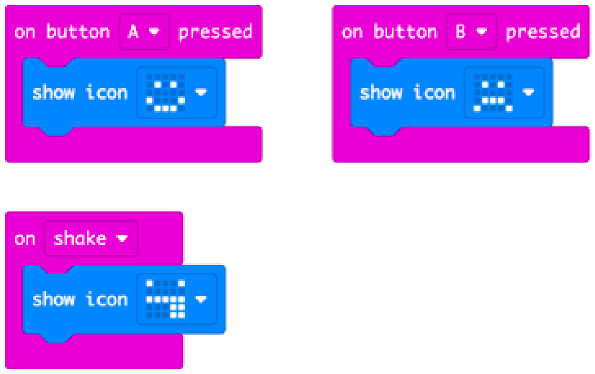
Requirements

- micro:bit (or MakeCode simulator) with USB
- MakeCode or Python editor
- Battery pack (optional)
- Some energy to shake, jump or get silly!

How it Works

- Like the Emotion badge, this program shows a happy or sad face depending on which button input you press, A or B.
- The micro:bit has other inputs, such as sensors.
- This program uses the micro:bit's accelerometer input to measure forces and sense when it's shaken.
- When the accelerometer senses sudden movement the program makes the silly face appear on the LED display output.

Code



```
on button A pressed
    show icon
```

```
on button B pressed
    show icon
```

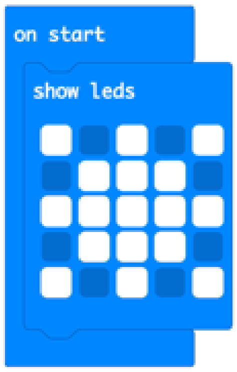
```
on shake
    show icon
```

Options

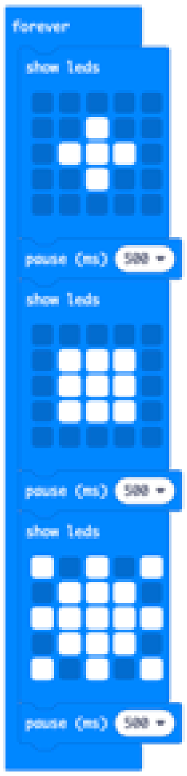
- Use different built-in emotions images like MEH, CONFUSED or ANGRY.
- Show another emotion when you press buttons A and B together.
- Add new emotions using the LED display to draw your own pictures like we did in Animated animals.

Notes

Topic	LEDs and Buttons
Project Name	**My Only Sunshine**
Description	Create your own sun icon for your micro:bit.
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional) • Graph paper for making your own designs (optional)
How it Works	• The micro:bit has lots of pictures built in, but sometimes you'll want to add your own. • This program creates a sun image using the micro:bit display's [5x5 grid of LEDs]. • Computers, phones and tablets use small pictures to represent things and ideas using a small number of dots, or pixels, as icons or emojis.

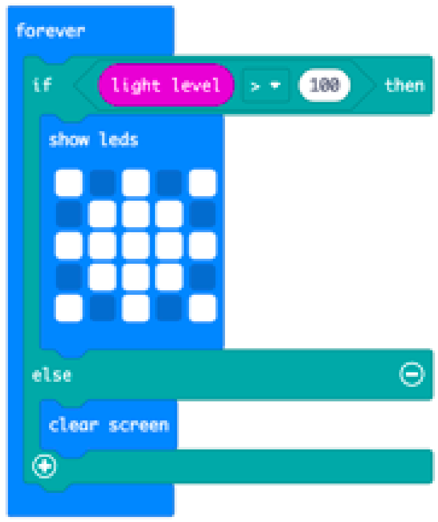
Code	 `on start`
	`show leds`
	`# . # . #`
	`. # # # .`
	`# # # # #`
	`. # # # .`
	`# . # . #`

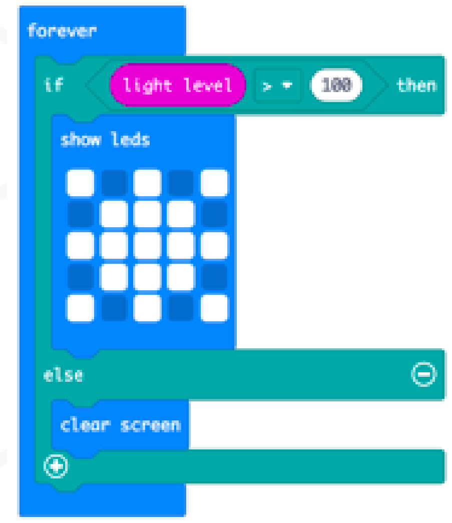
Options	• Draw your own version of the sun, or a star or moon. • Make a sun appear if you press button A, a moon if you press button B. • Use a loop to make the sun blink or rise. • *Advanced: In Python, use a range of numbers (not just 9) to show the sun's rays getting dimmer as they get further from the center.*
Notes	

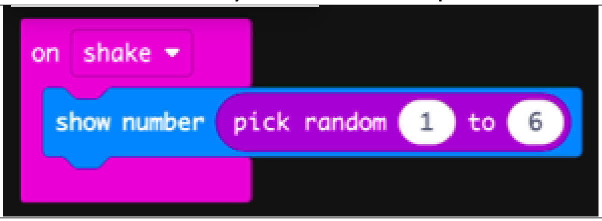
Topic	LEDs and Buttons
Project Name	**Sunbeam Animation**
Description	Use the sun icon from the My Only Sunshine to make a sunbeam animation.
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional) • Graph paper for making your own designs (optional)
How it Works	• The program shows a sequence of sun pictures on the LED display based on the one we made in the Here comes the sun project. • It waits 500 milliseconds (half a second) between showing each image to allow you to see it before displaying the next. • The sequence makes an animation of sunbeams coming from the centre of the sun. • The sequence repeats for as long as your micro:bit has power because the instructions are inside a forever, or infinite, **loop**. • Computers are often used to help animators make cartoons and films, creating an illusion of movement by showing a sequence of slightly different images one after another.
Code	 forever show leds . . . . # . . . # # # . . . # . . . . pause (ms) 500 show leds . . . . . . # # # . . # # # . . # # # . . . . . . pause (ms) 500 show leds # . # . # . # # # . # # # # # . # # # . # . # . # pause (ms) 500
Options	• Speed up or slow down the animation by changing the delay of 500 milliseconds. • Use your own design for the sun and its rays.

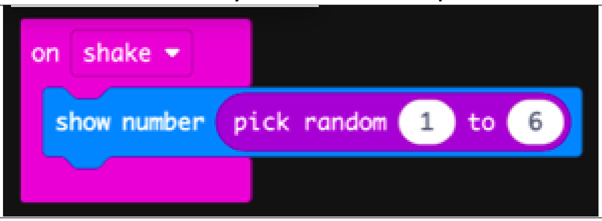
	• *In Python, use a range of numbers from 1 to 9 to show the sun's rays getting dimmer as they get further from the center.*

Notes

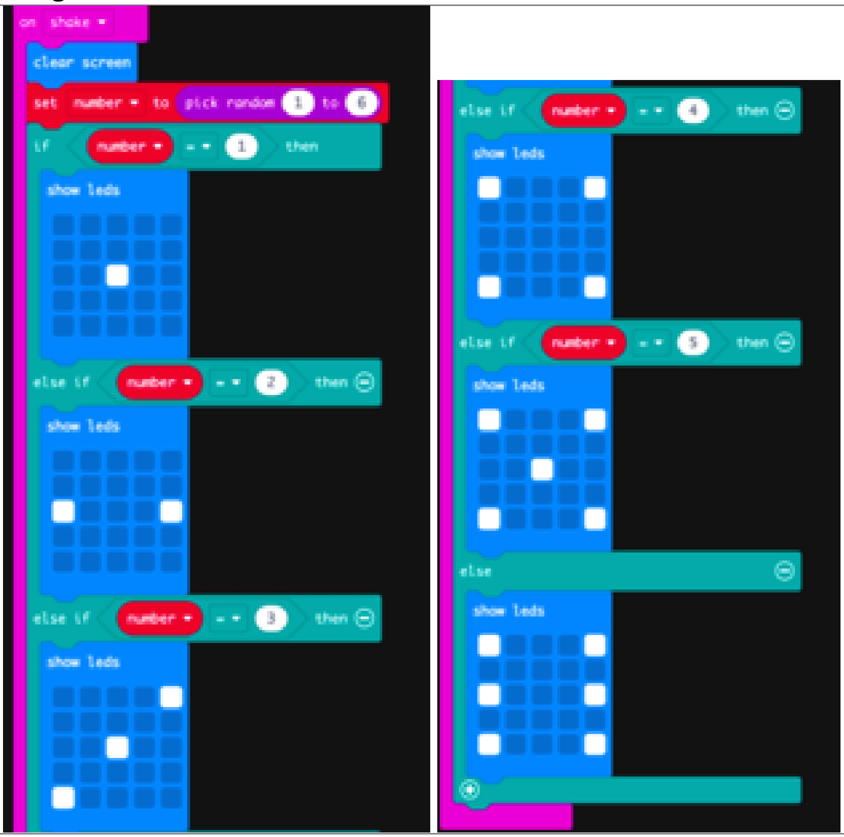
Topic	Sensors: Light
Project Name	**Light Detector**
Description	Turn the LED display into a sensor to make your micro:bit react to light
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional) • A light source and something to cover the micro:bit with – your hand will do!
How it Works	• As well as working as an output, the LEDs on your micro:bit can also work as an input device light sensor, measuring the amount of light falling on them. • This means that micro:bit programs can make different things happen depending on how light or dark it is. • The program uses an 'if… else' statement to show the sun image only **if** the light level is greater than (>) a certain level. This is known as selection – selecting when different things happen. • Flash this program onto your micro:bit and shine a light source, like a torch, daylight or bright ceiling light on to the micro:bit, and you should see the sun appear. • Cover the micro:bit with your hand and the sun icon should vanish. • If it doesn't work, try making the 100 number smaller to suit the lighting where you are.
Code	 forever if light level > 100 then show leds # . # . # . # # # . # # # # # . # # # . # . # . # else clear screen
Options	• Show an animated sun when light falls on your micro:bit.
Notes	

Topic	Sensors: Light
Project Name	**Light Detector**
Description	Turn the LED display into a sensor to make your micro:bit react to light
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional) • A light source and something to cover the micro:bit with – your hand will do!
How it Works	• As well as working as an output, the LEDs on your micro:bit can also work as an input device light sensor, measuring the amount of light falling on them. • This means that micro:bit programs can make different things happen depending on how light or dark it is. • The program uses an 'if… else' statement to show the sun image only **if** the light level is greater than (>) a certain level. This is known as selection – selecting when different things happen. • Flash this program onto your micro:bit and shine a light source, like a torch, daylight or bright ceiling light on to the micro:bit, and you should see the sun appear. • Cover the micro:bit with your hand and the sun icon should vanish. • If it doesn't work, try making the 100 number smaller to suit the lighting where you are.
Code	 forever if light level > 100 then show leds else clear screen
Options	• Show an animated sun when light falls on your micro:bit.
Notes	

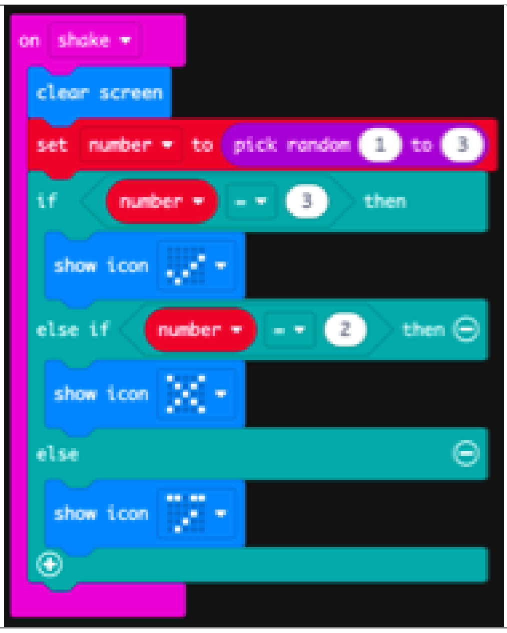
Topic	Sensors: Accelerometer
Project Name	**Random Numbers**
Description	Shake your micro:bit to make random numbers
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional) • Real dice (optional)
How it Works	• Like the Get silly project this program uses the micro:bit's accelerometer to make something happen when you shake it. • When you shake your micro:bit, the program selects a random number between 1 and 6 and shows it on the LED display. • It's really hard for computers to make truly random numbers because they're machines that work precisely and regularly. • Make a tally chart of how often each number comes up. Are these numbers really random? Compare it with real dice.
Code	 on shake show number pick random 1 to 6
Options	• Make the number appear for a few seconds, then clear the LED display to save batteries. • Make it roll 2 dice. You can make a random number between 2 and 12, or you can make two random numbers between 1 and 6 and add them together. • Try both methods and tally how often each score occurs. Does it make a difference? Do some numbers come up more often than others?
Notes	

Topic	Sensors: Accelerometer
Project Name	**Random Numbers**
Description	Shake your micro:bit to make random numbers
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional) • Real dice (optional)
How it Works	• Like the Get silly project this program uses the micro:bit's accelerometer to make something happen when you shake it. • When you shake your micro:bit, the program selects a random number between 1 and 6 and shows it on the LED display. • It's really hard for computers to make truly random numbers because they're machines that work precisely and regularly. • Make a tally chart of how often each number comes up. Are these numbers really random? Compare it with real dice.
Code	 on shake show number pick random 1 to 6
Options	• Make the number appear for a few seconds, then clear the LED display to save batteries. • Make it roll 2 dice. You can make a random number between 2 and 12, or you can make two random numbers between 1 and 6 and add them together. • Try both methods and tally how often each score occurs. Does it make a difference? Do some numbers come up more often than others?
Notes	

Topic	Sensors: Accelerometer
Project Name	**Digital Dice**
Description	A dice project that looks like a real die with patterns of dots instead of numbers.
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional) • Squared paper for designing your own dice faces (optional)
How it Works	• Like the Dice project this uses the accelerometer input to trigger the creation of a random number between 1 and 6 and show it on the LED display output when you shake the micro:bit. • Instead of showing a number, this program uses **selection** to show dots on the display to **represent** the numbers, looking like the dots on each face of real dice, depending on which random number was generated.

Code	
	`on shake`
	`clear screen`
	`set number to pick random 1 to 6`
	`if number = 1 then`
	`show leds`
	`. . . . .`
	`. . . . .`
	`. . # . .`
	`. . . . .`
	`. . . . .`
	`else if number = 2 then`
	`show leds`
	`. . . . .`
	`. . . . .`
	`# . . . #`
	`. . . . .`
	`. . . . .`
	`else if number = 3 then`
	`show leds`
	`. . . . #`
	`. . . . .`
	`. . # . .`
	`. . . . .`
	`# . . . .`
	`else if number = 4 then`
	`show leds`
	`# . . . #`
	`. . . . .`
	`. . . . .`
	`. . . . .`
	`# . . . #`
	`else if number = 5 then`
	`show leds`
	`# . . . #`
	`. . . . .`
	`. . # . .`
	`. . . . .`
	`# . . . #`
	`else`
	`show leds`
	`# . . . #`
	`. . . . .`
	`# . . . #`
	`. . . . .`
	`# . . . #`

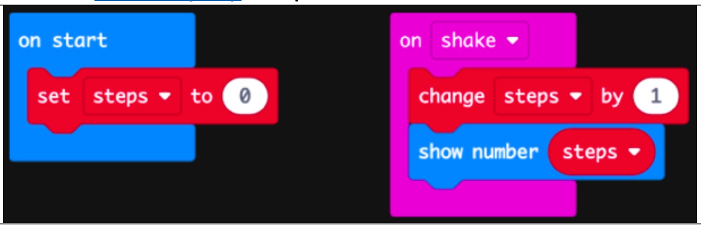
Options	• Make the display clear after a few seconds to make the batteries last longer and to make it clear when you have rolled two numbers the same. • Draw your own dot patterns to represent each number. • Make it roll higher numbers. How would you represent them on the 5x5 LED grid display output?
Notes	

Topic	Sensors: Accelerometer
Project Name	**Magic 8 Ball**
Description	Recreate a classic toy from the 1950s with your micro:bit and customise it to make it your own.
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional) • Some questions to ask your micro:bit
How it Works	• A Magic 8-ball is a toy invented in the USA in the 1950s. Shaped like an oversized pool ball, you ask it a question like 'will I be rich and famous one day?', shake the ball and one of 20 different answers randomly appear in a window. Answers can be positive, negative - or somewhere in between. • This program recreates a Magic 8-ball using the micro:bit's accelerometer, its ability to make random numbers and its LED display output to show a tick for yes, a cross for no or a 'meh' face for 'not sure'. • The program generates a random number between 1 and 3 then uses if… then… else… if… statements to make different symbols appear depending on the number. This is known as **selection**. • If the number is 3, it shows a tick for 'yes'. If the number is 2, it shows a cross for 'no'. • The program doesn't need to check if the number is 1, because if it's not 3 or 2, it must be 1, in which case it shows a 'meh' face for 'not sure.
Code	 on shake clear screen set number to pick random 1 to 3 if number = 3 then show icon else if number = 2 then show icon else show icon
Options	• Make the image vanish after a few seconds.

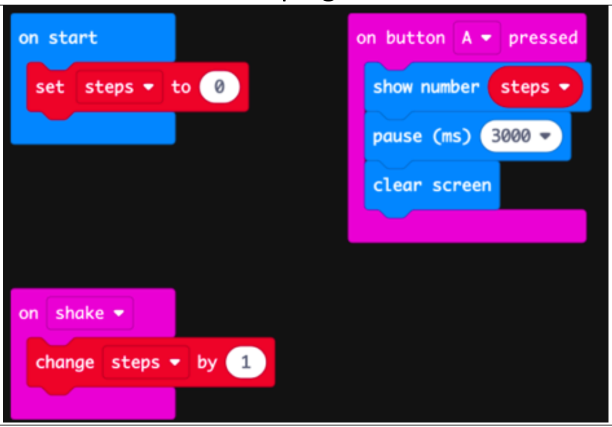
	• Make the micro:bit show different cryptic answers when you shake it, instead of pictures. It could say ‘I am not sure’ or ‘That remains to be seen’. • *Advanced: Here’s another way of making a Magic 8-ball using Python.*

Notes

Topic	Sensors: Accelerometer
Project Name	**The Long-Lasting Step Counter**
Description	Turn your micro:bit into a step counter (or pedometer) to help you track how active you are - and learn some coding at the same time!
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional) • Something to attach the micro:bit to your shoe or leg – string, tape or Velcro
How it Works	• Like the Dice project this program uses the micro:bit's accelerometer to make something happen. • It counts how many times the micro:bit has been shaken. It stores this number in a **variable** called 'steps'. • Variables are used by computers to store information that may change, such as the number of steps you've taken. • Every time the micro:bit accelerometer input senses a shake, the program increases the variable by 1, and shows the new number on the LED display output.

Code	
	on start
	  set steps to 0
	
	on shake
	  change steps by 1
	  show number steps

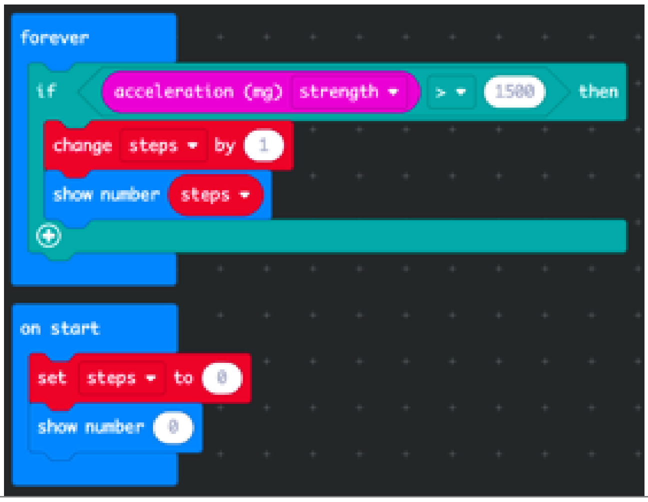
Options	• Add a button to reset the steps to 0. • Make the micro:bit beep each time a step is counted. • *Advanced: Add a graphical representation of how many steps you've taken.*
Notes	

Topic	Sensors: Accelerometer
Project Name	**The Long-Lasting Step Counter**
Description	Make a step counter with longer-lasting batteries.
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional) • Something to attach the micro:bit to your shoe or leg – string, tape or Velcro
How it Works	• Like the Step counter project this program uses the accelerometer to count a step every time the micro:bit is shaken and stores the total number in a **variable** called **steps**. • Keeping the LED lights on the micro:bit turned on requires more power. This program saves energy by only showing the step count when you press button A. • This means that the batteries will last longer, saving money, creating less waste and helping the environment.
Code	
Options	• Modify the program so button B sets the counter back to 0. • Think of ways of adapting other projects to make the batteries last longer.
Notes	

```
on start
    set steps ▾ to 0

on button A ▾ pressed
    show number steps ▾
    pause (ms) 3000 ▾
    clear screen

on shake ▾
    change steps ▾ by 1
```

Topic	Sensors: Accelerometer
Project Name	**A Custom Step Counter**
Description	A step-counter you can make more accurate by tailoring it to your own walking style.
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional) • Something to attach the micro:bit to your shoe or leg – string, tape or Velcro
How it Works	• The Step counter and Low energy step counter projects use the 'shake' gesture to count steps. The 'shake' gesture uses several accelerometer sensor readings to decide if the micro:bit has been shaken. • You may find that the 'shake' gesture isn't triggered every time you take a step, or that it's triggered too easily, leading to inaccurate counting of steps. • To make a more accurate step counter, instead of using the 'shake' gesture, this program uses numerical data from the accelerometer to decide whether you've taken a step and, if you have, increase the **steps** variable by 1. • If the acceleration is greater than (>) 1500, the **steps** variable is increased by one and show the step count on the LED display output. 1500 is the **threshold** – the point at which a movement will trigger a step to be counted. • You may need to change the 1500 number to make the step counter more accurate – but **you** can decide what threshold to use, whereas with the 'shake' gesture the threshold has been decided for you by the people who designed the micro:bit. • Modifying the threshold to work for you is called **calibration**. • Note that when micro:bit is not moving, the accelerometer gives a strength reading of about 1000. This is caused by the Earth's gravity pulling down on the micro:bit. **(WOW!!)**

Code

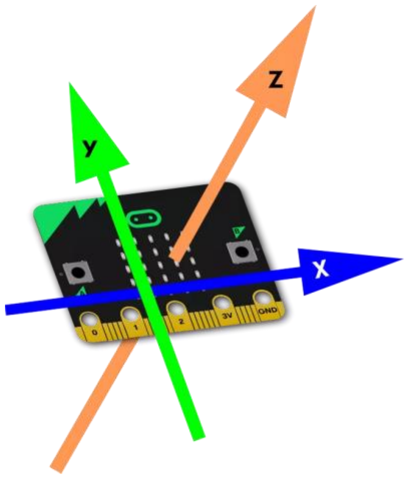


```
forever
  if acceleration (mg) strength > 1500 then
    change steps by 1
    show number steps
on start
  set steps to 0
  show number 0
```

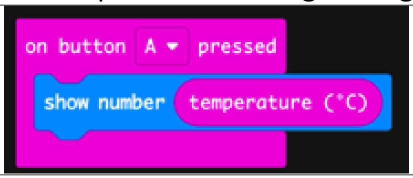
Options

- Make your batteries last longer by changing the program so it only shows the number of steps when you press button A.
- Modify the program so button B sets the counter back to 0.
- *Advanced: The accelerometer can measure forces in 3 dimensions, called the X, Y and Z axes. You can modify the code to choose which axis to measure, depending on which way up you fix your micro:bit to your leg or shoe.*

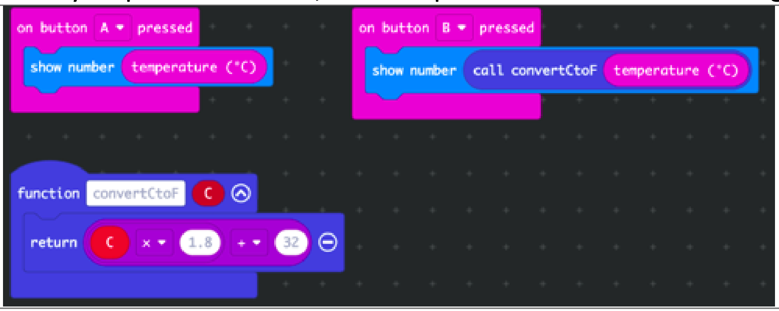
Notes



Topic	Sensors: Temperature
Project Name	**A Digital Thermometer**
Description	Show how hot or cold your micro:bit is using the built-in temperature sensor.
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional) • A source of heat or cooling, like a fan, if you want to see the temperature change quickly (optional).
How it Works	• This program shows how hot or cold your micro:bit is by taking a reading from the temperature sensor in its processor or CPU (central processing unit). • The processor's temperature is a fairly good approximation of the temperature around you in °C (Celsius). • In this program, when you press input button A, the micro:bit displays the processor's current temperature on its LED display output. • Take the micro:bit into warmer and colder places and see how the temperature readings change.

Code	
	`on button A pressed`
	`show number temperature (°C)`

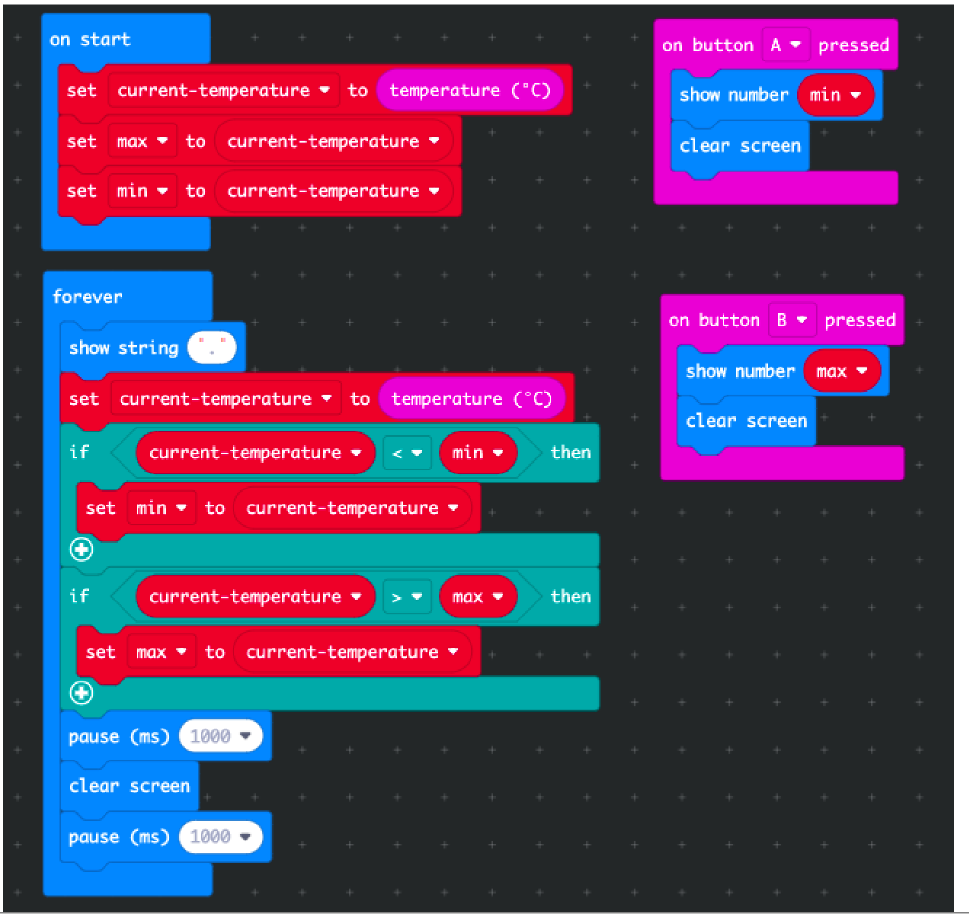
Options	• Compare the reading with another thermometer. How accurate is the micro:bit? Do you need to modify the micro:bit reading to get the air temperature?
Notes	

Topic	Sensors: Temperature
Project Name	**Celsius to Fahrenheit Converter**
Description	Use a simple **function** to convert centigrade readings from the micro:bit's temperature sensor to Fahrenheit.
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional)
How it Works	• The micro:bit's processor has a built-in temperature sensor input which gives readings in centigrade. • Using functions allows you easily to convert the temperature to Fahrenheit. • The **convertCtoF** function means you can re-use the conversion code easily, for example in a maximum-minimum thermometer. • The function is called by using **convertCtoF** in place of a variable or number when you press button B on your micro:bit. • We pass to the function the temperature in centigrade. • The function then takes the number passed to it, stored in a variable called **C**, and converts it to Fahrenheit by multiplying it by 1.8 and adding 32. • The function then returns the converted number so when you press button B the temperature is shown in Fahrenheit on the LED display output. • If you press button A, the temperature is shown in centigrade.
Code	
Options	• Improve the display by showing 'C' or 'F' after the temperature in centigrade or Fahrenheit. • *Advanced: Create your own function to add a conversion to degrees Kelvin when you press buttons A and B together.*
Notes	

```
on button A ▾ pressed
    show number temperature (°C)

on button B ▾ pressed
    show number call convertCtoF temperature (°C)

function convertCtoF C
    return C × ▾ 1.8 + ▾ 32
```

Topic	Sensors: Temperature
Project Name	**Max and Min Temperature**
Description	Track highest and lowest temperatures by leaving this program running on a micro:bit.
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional) • A source of heat or cooling, like a fan, if you want to see the temperature change quickly – or take the micro:bit outside • Graph paper if you want to keep a chart of temperatures over time.
How it Works	• Like the Thermometer project, this uses the temperature sensor inside the micro:bit's CPU (central processing unit) to measure the temperature in °C (Celsius). • This program keeps track of the lowest and highest temperatures recorded by using 3 **variables**: **currentTemp** is the current temperature reading, **max** is the maximum and **min** is the minimum. • At the start of the program they are all set to the same value; an infinite (forever) **loop** ensures that every two seconds it takes a reading, and the program compares the current temperature with the **max** and **min** variables. • If the current temperature is **less than (<)** than the value stored in the **min** variable, it changes the **min** variable to be the same as the current temperature. • If the current temperature is **greater than (>)** the **max** variable's value, it changes the **max** variable to be the same as the current temperature. • The program also flashes a dot on the LED display every time the infinite loop runs so that you know it's working. • Press button A to show the minimum and button B to show the maximum temperatures recorded. • You could leave the micro:bit running for 24 hours, record the maximum and minimum temperatures and plot on a chart at the same time every day and then reset.

Code

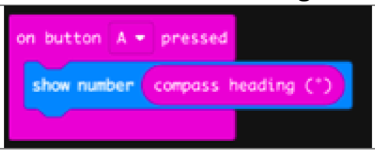
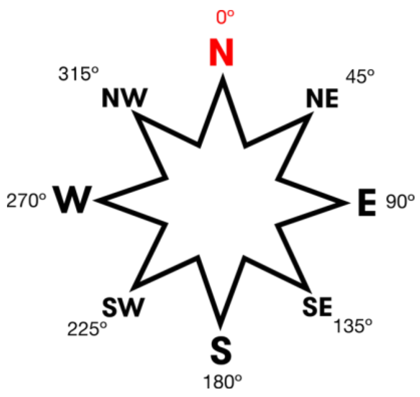


```
on start
    set current-temperature ▾ to temperature (°C)
    set max ▾ to current-temperature ▾
    set min ▾ to current-temperature ▾

forever
    show string ""
    set current-temperature ▾ to temperature (°C)
    if current-temperature ▾ < ▾ min ▾ then
        set min ▾ to current-temperature ▾
    ⊕
    if current-temperature ▾ > ▾ max ▾ then
        set max ▾ to current-temperature ▾
    ⊕
    pause (ms) 1000 ▾
    clear screen
    pause (ms) 1000 ▾

on button A ▾ pressed
    show number min ▾
    clear screen

on button B ▾ pressed
    show number max ▾
    clear screen
```

Options

- Convert the temperature to Fahrenheit using another input, like two buttons or the accelerometer.

Notes

Topic	Sensors: Magnetometer
Project Name	**The Navigator**
Description	Turn your micro:bit into a simple compass which shows its bearing from magnetic North in degrees.
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional) • A planet with magnetic poles to stand on – for example, Earth!
How it Works	• Your micro:bit has a built-in compass sensor called a magnetometer. You can use it to measure the Earth's magnetic field and use it as a compass. • When you first use the micro:bit compass you have to calibrate it – a little game appears on the screen where you have to tilt the micro:bit to light up every LED, then you're ready to go. • When you press the button A input, the micro:bit takes a reading from the compass sensor and shows the device's numerical compass bearing on the LED display output. Point the micro:bit North and you should see a reading of 0 degrees.
Code	 on button A ▾ pressed / show number compass heading (°)
Options	• Add another button to recalibrate the compass. • Have the micro:bit make a sound when it's pointing in a particular direction – this could be useful to help navigation when you can't look at a display or for people with impaired vision. • Make the micro:bit display letters or arrows to show if it's pointing North, South, East or West.
Notes	

Topic Sensors: Magnetometer

Project Name **Which Way is North?**

Description This simple compass will show you which way is North.

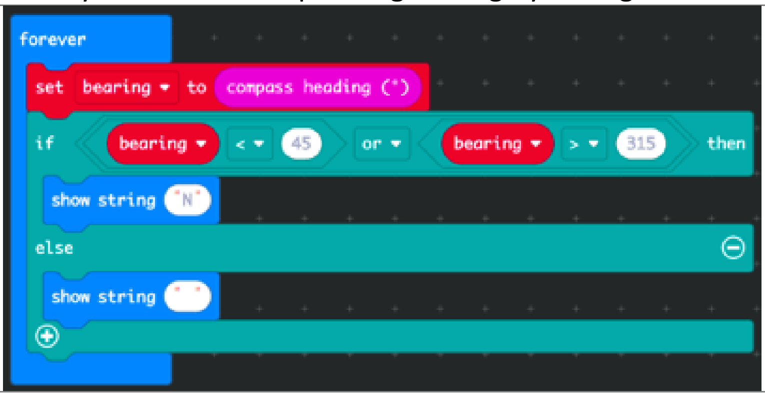
Requirements

- micro:bit (or MakeCode simulator) with USB
- MakeCode or Python editor
- Battery pack (optional)
- A planet with magnetic poles to stand on – for example, Earth!

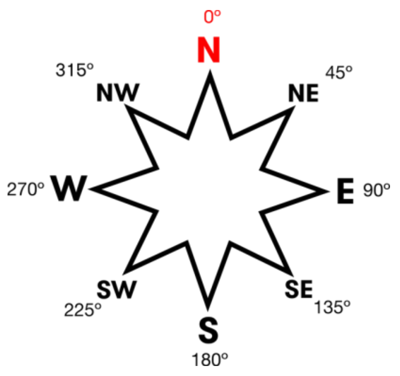
How it Works

- Your micro:bit has a compass sensor called a magnetometer that measures magnetic fields. It can sense the Earth's magnetic field and so you can use it as a compass.
- When you first use the micro:bit compass you have to calibrate it. A little game appears on the screen where you have to tilt the micro:bit to light up every LED, then you're ready to go.
- The program uses an infinite (forever) **loop** to keep taking compass readings and it stores them in a **variable** called 'bearing'. It then uses **selection**: an **if… else** statement to show N for North on the LED display **if** the bearing is greater (>) than 315 degrees **or** less than (<) 45. This means that it will show you where North is as long as your micro:bit is pointing in roughly the right direction.

Code



```
forever
  set bearing ▾ to compass heading (°)
  if bearing ▾ < ▾ 45 or ▾ bearing ▾ > ▾ 315 then
    show string "N"
  else
    show string " "
```



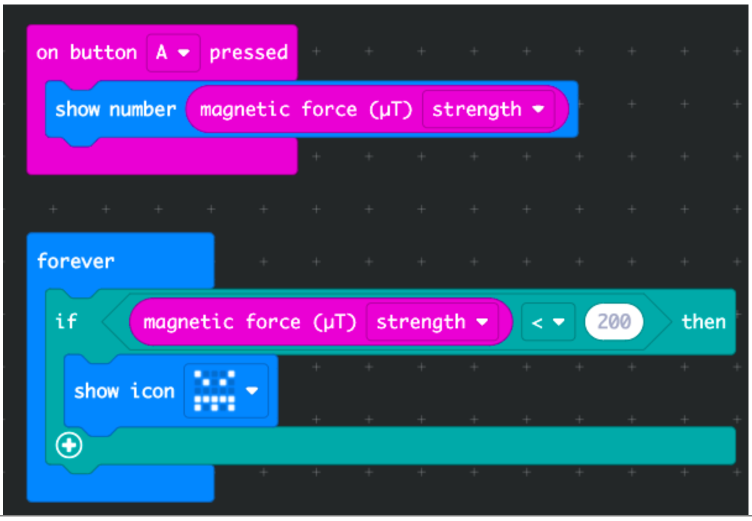
Options

- Make the compass more accurate by reducing the range of bearings: make the number 45 smaller and 315 bigger.
- Add other points of the compass to show when the micro:bit is pointing East, West and South.
- Add sound so it makes a noise when pointing North so someone who is visually impaired can use the compass.

Notes

Topic	Sensors: Magnetometer
Project Name	**Snoop Detector**
Description	Has anyone been in your room? With a micro:bit, a battery pack and a magnet you can make an alarm to alert you to sneaky snoopers...
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional) • Magnet • Some way of fixing the magnet, micro:bit and battery pack to the door and door frame
How it Works	• In this project you'll make a door alarm that works just like a real door sensor in a home security system. • Your micro:bit has a built-in compass sensor called a magnetometer. You can use it to measure the Earth's magnetic field as a compass - or to sense how strong magnetic fields are much closer to home! • Fix a magnet in the corner of a door and a micro:bit with the door alarm program close to it on the door frame like in the video. • The program uses the micro:bit's compass (magnetometer) **input sensor** and a **forever loop** to keep measuring the strength of the magnetic field. • It uses **selection** so when it falls below a certain level (the threshold), it shows an angry face on the LED display. This means the magnet moved away from the micro:bit - when the door was opened - so someone may have been in your room! • Pressing button A shows the current magnetic force reading. Use this to decide what **threshold** number to use by taking readings with the door open and closed. We used 200 in our example, but this depends on how strong your magnet is and if there are any other magnetic fields nearby. The coding video above shows you how to do this. • Press button A to clear the angry face and **reset** the alarm. • Note that when you first use the compass sensor you may have to **calibrate** it by playing a game to fill the screen with dots, like calibrating a mobile phone's compass.

Code

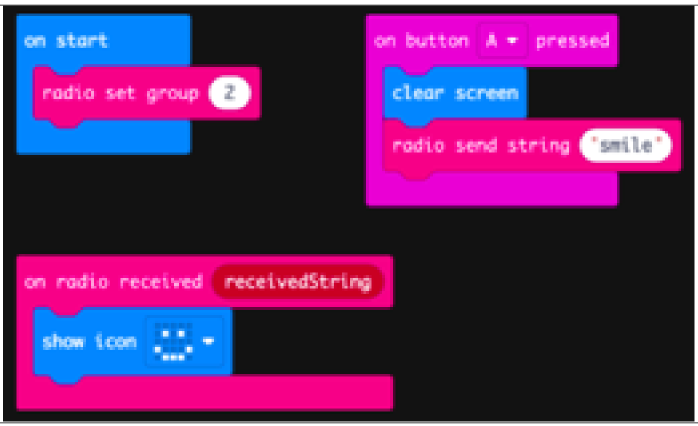


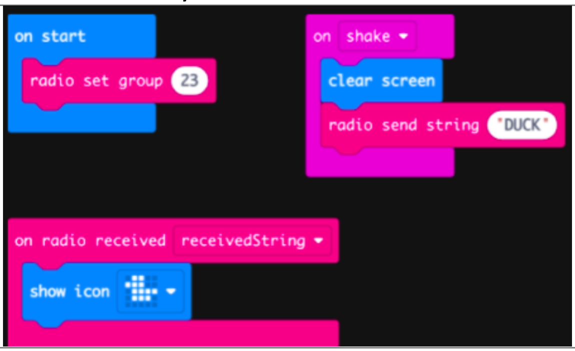
```
on button A ▾ pressed
    show number magnetic force (µT) strength ▾

forever
    if magnetic force (µT) strength ▾ < ▾ 200 then
        show icon ▾
    ⊕
```

Options

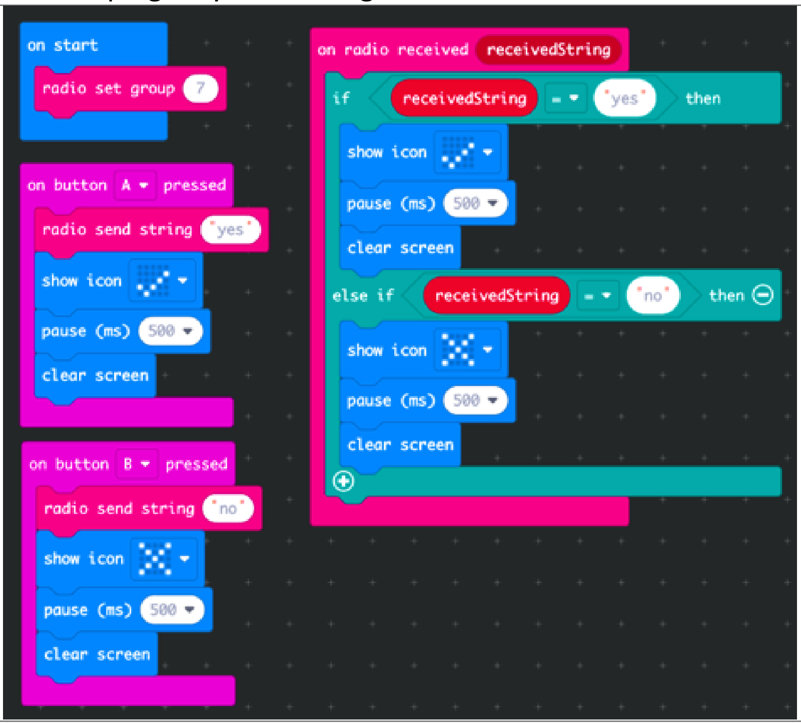
- Use a variable to count the number of times your door has been opened - you'll need to add code to sense when it's been opened **and** closed
- *Advanced: Create a timer to measure how long a door has been left open*

Notes

Topic	Radio
Project Name	**Send a Smile**
Description	Sharing and receiving kindness is a good way to support your well-being and that of your friends. Create a program using radio to send a smile from one micro:bit to another to support a friend
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional) • A friend to play with
How it Works	• This program uses the micro:bit's radio feature to share a smile. You can use it in the MakeCode simulator or flash the code on to 2 or more micro:bits. • First, it sets the radio group to 2. Groups are like channels, so any micro:bit using the same group will get the smile. You can use any group number you like from 0-255. • When you press button A, it sends a radio text message 'smile'. It also clears the screen so you can send another smile. • When it receives a radio message, it shows a smile emoji on the LED display. • The combination of radio group and the text of the radio message sent make up a protocol: a set of rules for how two devices communicate.
Code	 on start / radio set group 2 / on button A pressed / clear screen / radio send string "smile" / on radio received receivedString / show icon
Options	• Customize the smile emoji to your own happy face. • If you're working in pairs in a class, select your own unique radio group numbers for each pair of students so you can send messages to your partner but not anyone else. • How might you send a different emoji if you press button B? • *Advanced: You could also do this by keeping the same radio group but modifying the code, so the text message sent is unique to your pair. Modify the code so it only shows a smile if the correct message is received.*
Notes	

Topic	Radio
Project Name	**Teleporting Duck**
Description	Make a duck fly invisibly through the air from one micro:bit to another.
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional) • A friend to play with
How it Works	• Flash this program onto two micro:bits, shake one and a duck appears to travel magically through the air from one to the other. Shake the other to send it back. • It's not really magic. It uses the micro:bit's radio function to send data from one micro:bit to another when the accelerometer detects a shake gesture. • The program first sets the radio group to 23. Groups are like channels on walkie-talkie radios; they can be number between 0 and 255. It doesn't matter what number you pick as long as your friend's micro:bit is using the same group number, and no-one else nearby is using the same group. • When you shake it, it sends the word 'DUCK' on that radio group and clears the screen. If either micro:bit receives a radio message (**any** radio message), a duck icon appears on its display, so you should only ever have 1 duck visible at any time.
Code	
Options	• Find out how far apart you can you go for this still to work. • Teleport other animals. Would you need to change the picture, the message – or both? • *Advanced: What happens if more than 2 of you use the same radio group? How can you fix this*
Notes	

```
on start
    radio set group 23

on shake
    clear screen
    radio send string "DUCK"

on radio received receivedString
    show icon
```

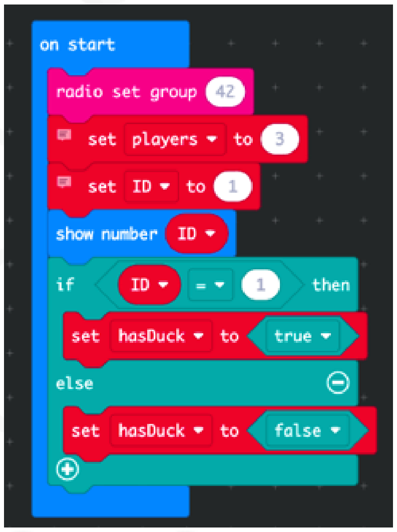
Topic	Radio
Project Name	**Tell Me a Secret**
Description	Use micro:bit's radio feature to answer questions in secret.
Requirements	• micro:bit (or MakeCode simulator) with USB • MakeCode or Python editor • Battery pack (optional) • A friend to play with
How it Works	• Flash this program on to two micro:bits. You and a friend ask each other questions that have 'yes' or 'no' as the answer. • Press input button A to send 'yes' and button B to send 'no' messages. A tick or cross will flash on both micro:bits' LED display outputs for half a second. • The program uses radio to send your answer secretly - no-one (except your partner) can hear the radio signal. • When a radio message is received, the program uses **selection** to test the message: **if** the received message **is equal to** 'yes', **then** it shows a tick on the LED display, but **if** the message **is equal to** 'no' **then** it shows a cross. • Make sure the radio group number is the same on both micro:bits – you can use any number between 0 and 255. • If lots of you are using this program in the same place, you'll want to make sure each pair of people has their own radio group number. • Keep your radio group number a secret if you don't want anyone snooping on your messages!
Code	 on start radio set group 7 on button A pressed radio send string "yes" show icon pause (ms) 500 clear screen on button B pressed radio send string "no" show icon pause (ms) 500 clear screen on radio received receivedString if receivedString = "yes" then show icon pause (ms) 500 clear screen else if receivedString = "no" then show icon pause (ms) 500 clear screen
Options	• Show different icons or messages for 'yes' and 'no'.

- Use shake, tilt or buttons A and B together to send different answers such as ‘maybe.’
- *Advanced: Change ‘yes’ and ‘no’ to ‘dot’ and ‘dash’ and send Morse code messages.*

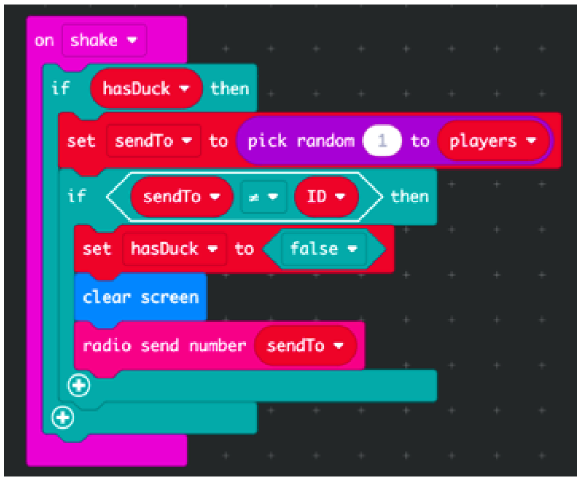
Notes

Topic	Radio
Project Name	**Pass the Duck**
Description	A group game of ‘hot potato’ using radio – or hot duck!
Requirements	• 3+ micro:bits (or MakeCode simulator) with USBs • MakeCode or Python editor • Battery pack (optional) • A group of people to play with
How it Works	• Like the Teleporting duck game, this uses radio to send a ‘duck’ through the air between micro:bits. That game only works with 2 players as it sends the same message to everyone – soon you would find almost everyone had a duck and anyone could throw one. • If you have more than 2 players each micro:bit needs to have a way of choosing which player will get the duck so each player’s program has a unique ID number, starting with 1. • We store this in a **variable** called **ID**, and you’ll need to change this to 2, 3, 4 etc for each player before flashing the program onto their micro:bits. • Set the **players** variable to the number of people to make sure everyone has a chance of getting the duck. The program shows the player ID number on the LED display at the start so you know who’s got which number. • To make sure that only the player that has the duck can chuck it, the program only sends a message when you shake it **if** you have the duck. It keeps track of this using a **Boolean variable** called **hasDuck**. Boolean variables can only have two values: True or False. At the start only player 1 has the duck, no-one else can throw it. • When player 1 shakes her micro:bit, the program generates a random number between 1 and the number of players. If the random number is **not equal to** her own ID number, it sends the new ID number by radio, clears her screen and sets her **hasDuck** variable to False. • If the random number **is equal to** her own ID, she’ll need to throw again, but that’s better than the duck being sent to herself – and getting lost in hyperspace! • If your micro:bit receives a number, it checks to see if it is equal to (matches) your ID number. If it does, congratulations, you now have the duck! A duck appears on your LED display, and your **hasDuck** variable gets set to True, meaning you can now chuck the duck to someone else. • **Please chuck ducks responsibly: make sure you don’t drop your micro:bit or hit a friend in your enjoyment of this game**.

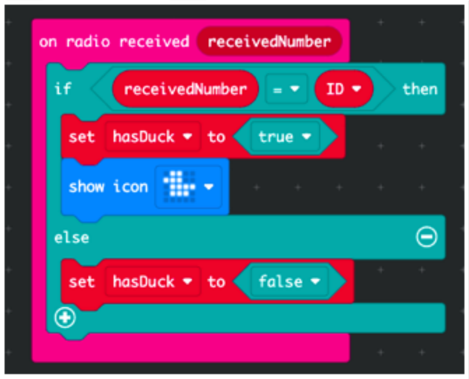
Code



```
on start
    radio set group 42
    set players to 3
    set ID to 1
    show number ID
    if ID = 1 then
        set hasDuck to true
    else
        set hasDuck to false
```



```
on shake
    if hasDuck then
        set sendTo to pick random 1 to players
        if sendTo ≠ ID then
            set hasDuck to false
            clear screen
            radio send number sendTo
```



```
on radio received receivedNumber
    if receivedNumber = ID then
        set hasDuck to true
        show icon
    else
        set hasDuck to false
```

Options

- Change the program to send other things instead of ducks.
- At the moment, if it picks a random number that's the same as your own ID number, you have to shake again. Modify the program so this never happens. There may be more than one way to do this.
- Player 1 **always** has the duck at the start of the game. Could you improve the program so the first player with the duck is chosen at random? How would you communicate this to every player's micro:bit?

Notes

Lab Notes

Summary of Key Concepts

Topic	Concept	Notes
Computers	Inputs	
Computers	Outputs	
Computers	Processor	
Computers	Program	
Coding	If-Then-Else	
Coding	Variable	
Coding	Loops	
Coding	Threshold	
Computers	Accelerometer	
Computers	Magnetometer	
Coding	Functions	
Coding	Iteration	

Topic	Concept	Description

Date

New Terms and Concepts

Project Notes

Ideas

Date

New Terms and Concepts

Project Notes

Ideas

Date

New Terms and Concepts

Project Notes

Ideas

Date

New Terms and Concepts

Project Notes

Ideas

Date

New Terms and Concepts

Project Notes

Ideas