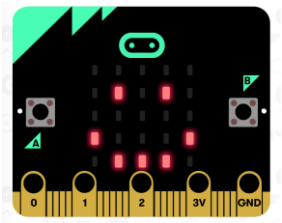
Micro:bits = Macro Fun

Peter Fox &

Introduction



The purpose of this workshop and handout is to provide accessible getting started resources, stimulating ideas (challenges) and experiences that highlight the opportunities afforded by the Micro:bit™. The first thing to consider when using Micro:bits is whether you want students to learn coding or simply engage in pre-prepared activities. This dilemma is similar to considerations pertaining to dynamic geometry: Do students learn by using the amazing array of available tools or through prepared interactive documents? In both cases, a wealth of resources exists to help you achieve success: 10 Minutes of Coding, Activity Downloads, YouTube, Google Groups and of course the wonderful Texas Instruments STEM team. A world of opportunities starts here. Are you ready?

Micro:bit Sensors and Functionality (inbuilt):

- 3 axis accelerometer
- Ambient light
- Temperature
- 2 x push buttons
- Microphone
- Compass
- Speaker
- LED Display
- General purpose input/output
- Low level radio communications
- Bluetooth
- Expansion board connectivity

You can attach the Micro:bit to an expansion board to introduce a world of new opportunities. Imagine flying a drone from your calculator and using ultrasonic sensors to access hand gestures to manoeuvre. What about attaching a GPS sensor to capture your location, create a treasure hunt or calculate speed.

Equipment

Calculator:

- TI-Nspire CX II CAS, TI-Nspire CX II non-CAS or TI-84Plus CE Python
- Micro:bit™ Python Module installed in Python Library (Other modules will be useful)
- Make sure your calculator has the latest operating system.

Micro:bit:

- Version 2.2 – With latest HEX file from Texas instruments. (Drag and drop set up)
- USB Mini to USB Micro – (Specific OTG cable required)

Micro Parabolas

Introduction

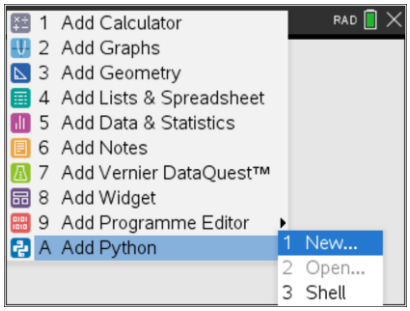
In this activity you will learn how to capture data from the inbuilt accelerometer and use it to manipulate a graph. The activity is a novel way of exploring transformations. From a educational neuroscience perspective, novelty stimulates the dopamine system which is responsible for associative learning. Dopamine release is part of the brain's reward system encouraging us to find out more. Put simply, the nature of the activity means that students are more likely to remember it. Caveat: The activity is not designed to help students *understand* transformations, other activities exist for this aspect. Once you have completed this activity, spend some time reflecting:

"Is the coding in this activity a necessary component of the learning experience?"

Getting Started

1. Start a new document and insert a Python program.

   Name: Micro



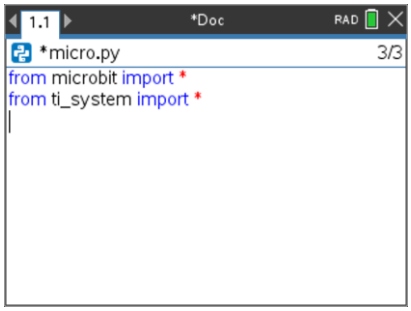
2. Import the Micro:bit and TI System modules.

   Menu > More Modules > BBC micro:bit > from microbit import*

   This instruction means that the Micro:bit commands will be accessible from your Python program.

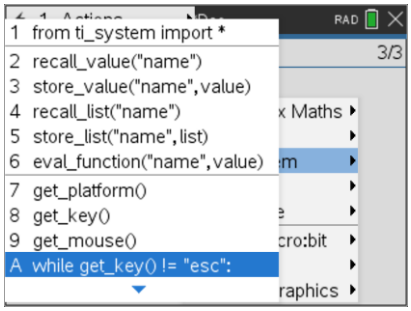
   Menu > More Modules > TI System > from ti_system import*

   Some of the variables collected in the Python program need to be made accessible to other calculator applications.



3. A while loop that can be halted by the "esc" (escape) key is useful when exploring and particularly when loop conditions are yet to be determined. A pre-prepared instruction exists:

   Menu > More Modules > TI System > while get_key()!="esc"



The Python programming tool on the calculator includes some handy navigation tools. The TAB key can be used to jump from one user input component to another within commands. If there are no more user input components, the TAB key will jump to the end of the line.

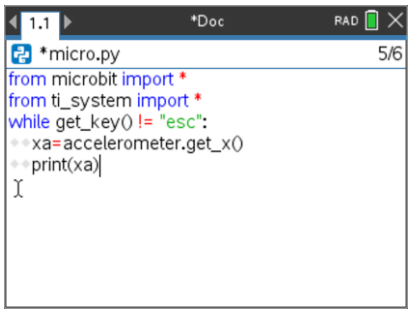
4. We're ready to start capturing data from the accelerometer and displaying the results on screen.

   Menu > More Modules > BBC micro:bit > Sensors > Accelerometer

   Select the x axis option: x=get_x() and store as: xa
   To see the values, use the print() command:

   Menu > Built-ins > I/O > print()

   Put xa in the print command and run the program.



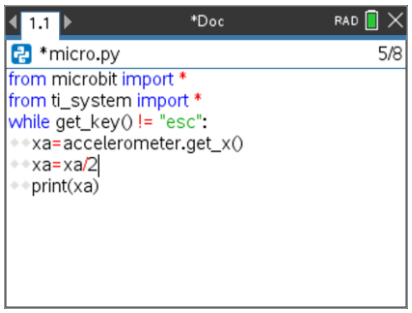
Run the program and explore what happens when you tilt the Micro:bit in different directions.
Note the range of values you can get through a full range of movement and through which axis.

Authors: Peter Fox

Acting on Data

1. The values for the transformations need to vary between -10 and 10 in the x direction so a conversion factor needs to be incorporated. The conversion factor needs to be included in the loop, and preferably prior to the print(xa) statement.

   xa = xa/*** [Replace the *** with **your** conversion factor.]



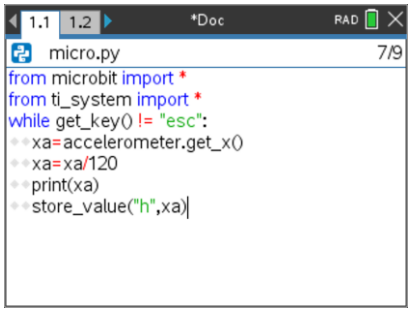
2. The 'xa' variable collected in Python needs to be transferred to the other applications in the calculator. This needs to occur within the "While" loop:

   Menu > More Modules > TI System > store_value("name",value)

   Name = Variable name in the calculator environments.

   Value = Python variable name.

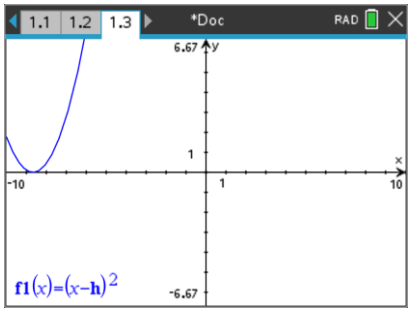
   In this example, 'xa' will be stored to 'h' and accessible from the calculator environments. **Run your program**. You won't notice any difference; however, 'h' will now have a value which is critical for the next step!



3. You should now be in the Python Shell (Page 1.2). Insert a Graphs Application, (Page 1.3) and graph the following function:

   $$y = (x - h)^2$$

   When you enter the function, you should notice that the 'h' is bold, signifying it has already been defined, courtesy of your "micro" program.



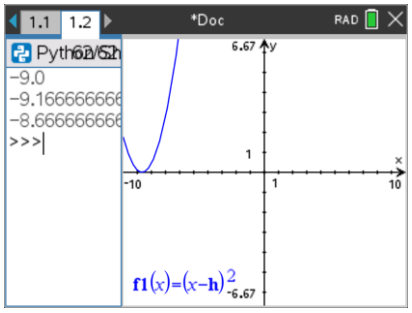
4. Navigate back to the Python Shell (Page 1.2) and press:

   Ctrl + 4

   This is a short-cut to combine or group pages 1.2 and 1.3 onto one page. The two applications: Python Shell and Graphs should now be visible. We want to focus on the Graph application, press:

   doc > Page Layout > Custom Split

   Move the divider to the left to make more of the Graph application visible.

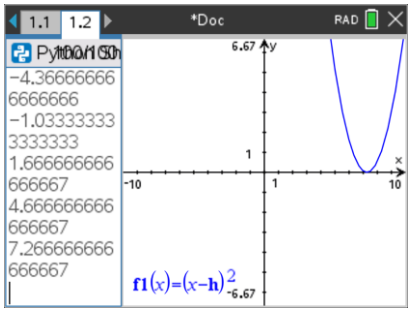


5. We are now ready to run the Python program again. The calculator focus may be on the Graphs application, to shift focus press: Ctrl + Tab, this is similar to Alt + Tab in the Microsoft Windows™ environment.

   To run the program press:

   Menu > Tools > Run

   Select the micro program and start moving the Micro:bit! When you're done exploring, press "esc" to exit the loop and therefore end the program.



Try it!

Edit the conversion line in your program: xa = xa / … so that it reads: xa = int(xa/ …) including your conversion factors and run the program again. What does the "int( )" command do to the numbers and graph movement?

Authors: Peter Fox

Challenge 1:

Edit your Python program. Use the same loop to capture data for acceleration in the Y direction. Apply a suitable scale factor and transfer this measurement to a calculator variable: 'k'. Run your program to ensure k has a stored value, then change your graph to:

$$y = (x - h)^2 + k$$

Challenge 2:

Another transformation needs to be created. The 'z' axis accelerometer is a little harder to control. The other transformation is referred to as a dilation, typically referenced as "a". Edit your Python program, and create a new loop so that the accelerometer can control the value of 'a' where 'a' is able to vary between -3 and +3 in increments of 0.25. Redefine your equation as follows:

$$y = a(x - h)^2 + k$$

Challenge 3:

Another option is to reflect a function in the x or y axis. Write a program to reflect a function using the respective tilting of the Micro:bit.

Challenge 4:

The micro:bit can be used as a 'plane', write a program to graph the corresponding plane in the 3D graphing window.

IDEAS!

Calculus:

Imagine the Micro:bit is attached to a ruler using a rubber band and calibrated so that the measurements reflect the slope (gradient). Set the program up to capture the 'slope'. The teacher moves their finger along a function and the students model the slope using their ruler. Students will now have a series of slope measurements they can graph, the derivative!

Imaginary Dice:

Imagine you are holding a cube/dice, each time you tilt the Mciro:bit the corresponding side of the cube or dice is displayed. Provide students with a net for the cube and get them to populate the net with the corresponding images. This challenge is made even harder if the images are no longer visible/accessible when the students attempt to populate the net!

Write your own:

Discuss and share ideas with a partner for other activities.

Authors: Peter Fox

Micro Data Collection

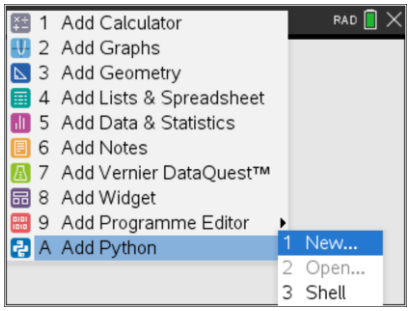
Introduction

The Micro:bit has an built in thermistor which means it can measure temperature. In this activity you will learn how to capture temperature and time data and store that data in a list.

Getting Started

1. Start a new document and insert a Python program.

   Name: DataCollect



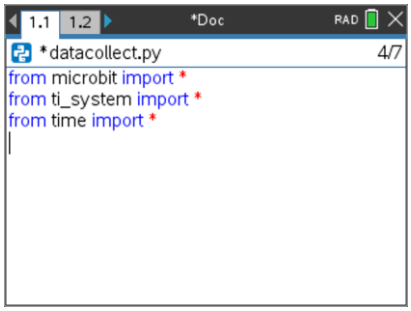
2. Three modules need to be imported for this program:

   Menu > More Modules > BBC micro:bit > from microbit import*

   Menu > More Modules > TI System > from ti_system import*

   Menu > More Modules > time > from time import*

   The Micro:bit module is required for the onboard sensor, the TI-System is required to transfer data from the Python program to the calculator applications and the time is useful for capturing time stamps.



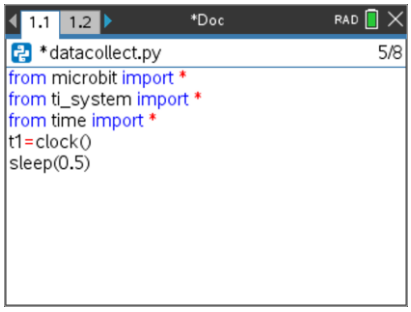
3. To access the calculator's internal clock:

   Menu > More Modules > Time > clock()

   The clock time needs to be stored in a variable: t1

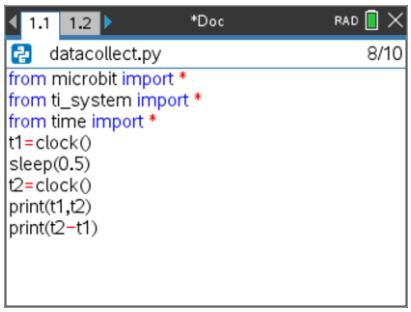
   The program can then be set to sleep for 0.5 seconds.

   Menu > More Modules > Time > sleep()



4. Collect another time and store the result in: t2

   The two times can be printed to the screen and the difference calculated.



Try it!

Run the program to see the values generated from the calculator's internal clock.

Try placing a print(t1) command before collecting t2 to gauge how long it takes to execute a print() command.

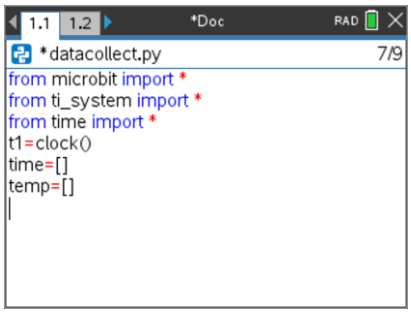
Authors: Peter Fox

5 The superfluous time commands and calculations can be removed leaving only the original time stamp: t1=clock()

Two empty lists need to be established to record the data, one for the time and another for temperature.

time = [ ]

temp = [ ]

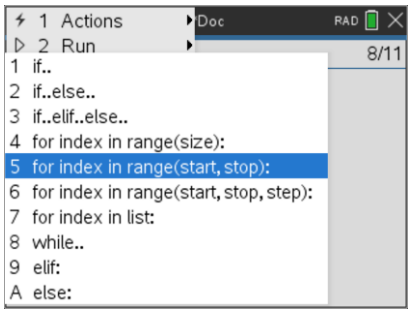


6 Ten data points will be collected.

As we know the quantity of points, a "For" loop is the most appropriate structure.

Menu > Built-ins > Control > for index in range (start, stop):

The loop counter will be 'n', starting at 0 and ending at 10.

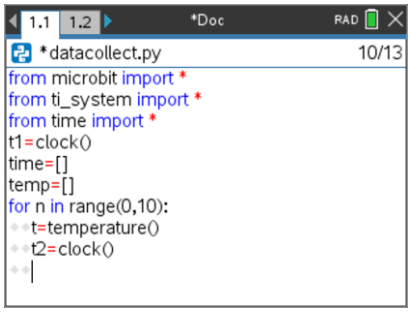


7 Within the loop the temperature will be collected from the Micro:bit, the calculator clock time will be recorded, and both of these values added (appended) to their respective lists.

Menu > More Modules > BBC micro:bit > Sensors > …Temperature

Store the temperature measurement at 't'.

Capture the calculator's clock time and store the result as t2.



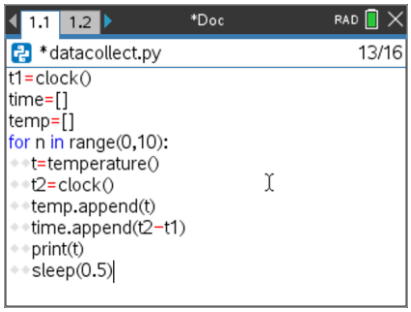
8 The lists can now be updated (appended). Start with the temperature: temp

Menu > Built-ins > Lists > .append

Add the temperature just collected (t) and repeat for the time list.

Print the temperature to the calculator screen and add a sleep time (0.5).

Run your program. To display the results at the end of the program type the variable names (or use the variable key) to access the values.



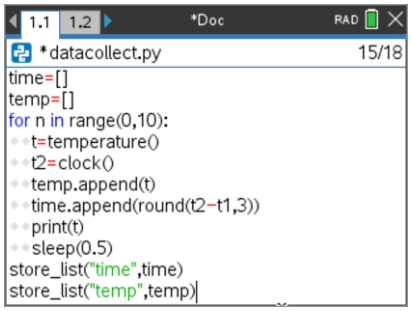
Run the program to see temperature values. You can hold onto the Micro:bit to increase the temperature.

Time readings can be 'simplified' by including the 'round' command: time.append(round(t2-t1,2))

9 The final step is to send the data across to the calculator. This step is done once the loop has finished.

Menu > More Modules > TI System > store_list("name",list)

Transfer both the time and temperature (temp) lists and run the program. Once the program has finished, graph your data in the Data & Statistics application.



Authors: Peter Fox

Micro-Communications

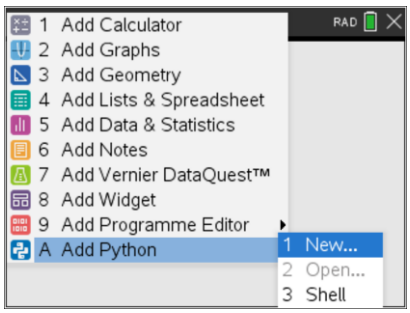
Introduction

The BBC Micro:bit™ has a built wireless functionality. In this activity you will learn how to send data from one calculator to another, wirelessly. Calculator 1 represents the data source (sender), calculator 2 is the receiver.

Sender – Calculator 1

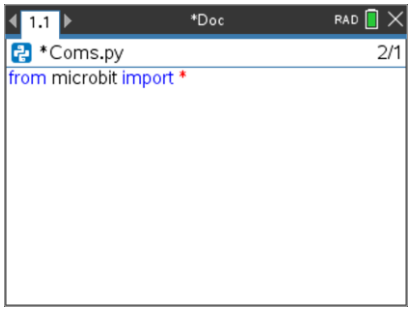
1. Start a new document and insert a Python program.

   Name: Coms



2. Import the Micro:bit and TI System modules.

   Menu > More Modules > BBC micro:bit > from microbit import*



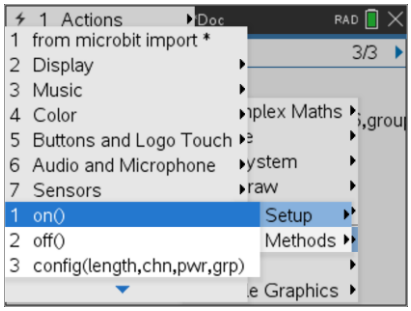
3. By default, radio communications are turned off to save power, so we need to turn them on:

   Menu > More Modules > BBC micro:bit > Radio > Setup > On()

   The next step is to configure the communication length, channel, power and group:

   Menu > More Modules > BBC micro:bit > Radio > Setup > config(…

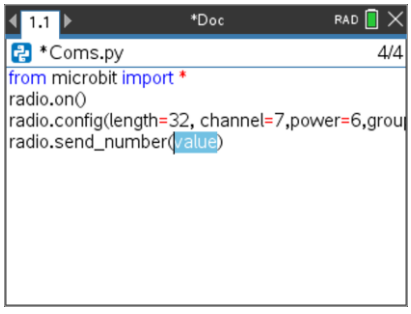
   For now, we can leave the default configuration, it can be changed later.



4. We're now ready to send some information to another device, that device can be another calculator or computer! In this example we will send a number:

   Menu > More Modules > BBC micro:bit > Radio > Methods > …

   Enter a number for the 'value'. The program is ready to run! Now we need a second device which will receive the data.



Authors: Peter Fox

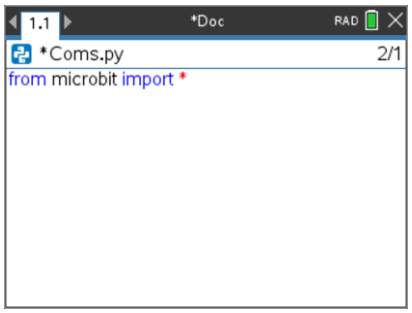
Receiver – Calculator 2

1. Start a new document and insert a Python program.

   Name: Coms

   Import the Micro:bit and TI System modules.

   Menu > More Modules > BBC micro:bit > from microbit import*



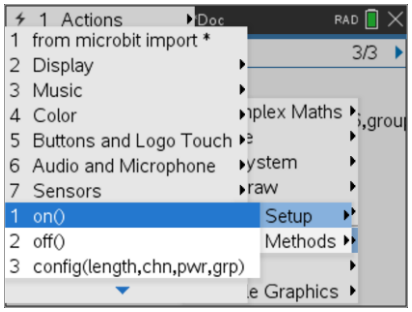
2. By default, radio communications are turned off to save power:

   Menu > More Modules > BBC micro:bit > Radio > Setup > On()

   Configure the communication length, channel, power and group:

   Menu > More Modules > BBC micro:bit > Radio > Setup > config(…

   These settings need to match the sender! Leave these as default for now.

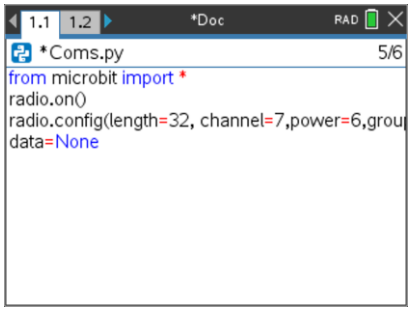


3. Receiving data is slightly more complicated than sending, because we don't know *when* the data will be pushed out into the world, therefore the receiving device needs to be constantly listening.

   We start by assigning a variable to store the information. The variable should not contain any information.

   data = None

   Make sure to capitalise the N, Python is case sEnSiTiVe!



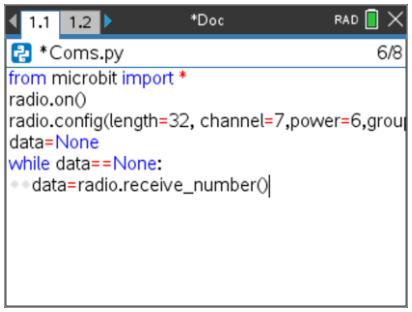
4. A **while** loop can be used to receive and monitor radio communications, no signal means the data variable equals "None".

   Menu > Built-ins > Control > While

   Set the condition: **data == None**

   The only command in the body of the loop is to 'receive' a value:

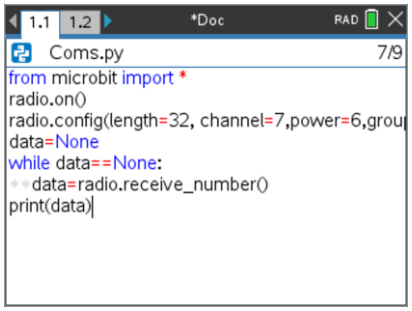
   Menu > More Modules > BBC micro:bit > Radio > Manage > receive



5. The final step is to display the data once it has been received, note that this only needs to be done once the data has actually been received.

   Menu > Built-ins > I/O > print()

   Insert the 'data' into the print command. The program is ready to run. The receiving program MUST be run before the sending program! Once the receiving program is running on Calculator 2, run then sending program on calculator 1.



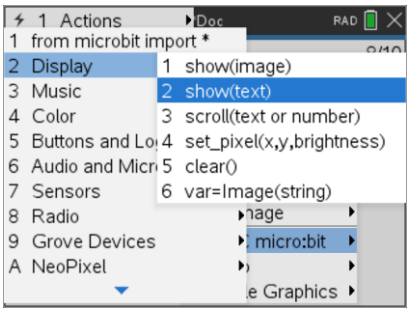
Authors: Peter Fox

Challenge 1:

Two-way communication means information moves backwards and forwards between devices. On calculator 1 add a data = … , while loop …, radio receive … and print instructions immediately after the send command. Once the calculator has sent its information (number), it will wait to receive a response.
On calculator 2, immediately after the print(data) command, add in some send instructions. Think about the timing of the send/receive communications.

Challenge 2:

Change the send/receive instructions on each device so that 'text' messages can be sent from one device to another. Note that the length of the text is set to "32" in the configuration command, so the message text cannot exceed 32 characters.
For an added bonus, try using the display text command:



Challenge 3:

Explore what happens when the signal strength (power) is reduced in the configuration command. How far does the signal travel for power = 6 compared with power = 1.

Challenge 4:

When several people are using the same 'channel', you may pick up someone else's messages. There are many ways this can be prevented, the simplest way is for each pair to select the same channel, but a different channel than everyone else. Another way is to 'encrypt' the message. From a simplistic perspective, working with numbers, the encryption could be as simple as the sender adding on the multiple of some number (n) to the value they want to send. The receiver can then use modular arithmetic (n) to remove the unwanted quantity.
Example:

- Sender and receiver agree on "1021" as the encryption value. (modular arithmetic).
- Sender wants to send the number 37, so they send: 1058.
- Receiver then uses: print(data%1021)
- Any other person receiving the information that is not privy to the encryption value will receive a 'meaningless' value. This concept can be extended with multiple communications.

IDEAS!

Treasure Hunt:

Imagine multiple calculators are set up across multiple spaces/rooms. Each of these calculators is constantly sending out a signal. Student calculators are in constant 'receive' mode. Each student is provided with a clue that helps them find a specific calculator. "IF" the student arrives at the correct location, the signal they receive will match the stored value for that clue and immediately display the next clue! Design a treasure hunt!

Authors: Peter Fox