Coder Workbook for the Morse Code micro:bit project

Project 1

Introduction

The best thing about learning to communicate using morse code is that we don't even need a keyboard to send any message we want – perfect for our micro:bits!

A **receiver** is a device that can listen and receive communication.

A **transmitter** is a device that can send communication.

A **transceiver** is a device that can both send and receive communication.

Encoding is turning alphabet and number characters into morse code dots and dashes.

Decoding is turning is turning morse code back into alphabet and number characters.

Project 1:

Programming the micro:bit into a morse code transmitter

Apart from the two **morse** signals (dot and dashes), we'll also need to transmit two **control** signals to let the receiver know when we've finished each word and when we've finished 'speaking'...

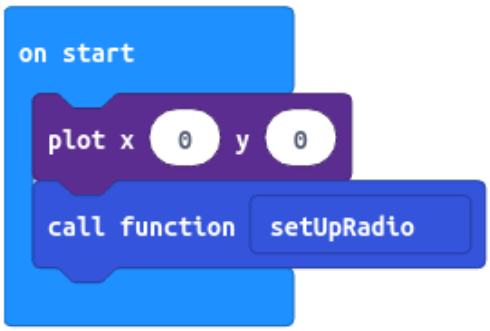
button A	=	dot
button B	=	dash
shake	=	end of word (EOW)
screen down	=	over and out (OAO)

..plus we'll be creating two **functions** that know how to do the work of:

1. **setting up** the micro:bit radio
2. **sending** the morse and control signals

…. and we'll be naming them appropriately.

<u>Coding the 'on start' block:</u>



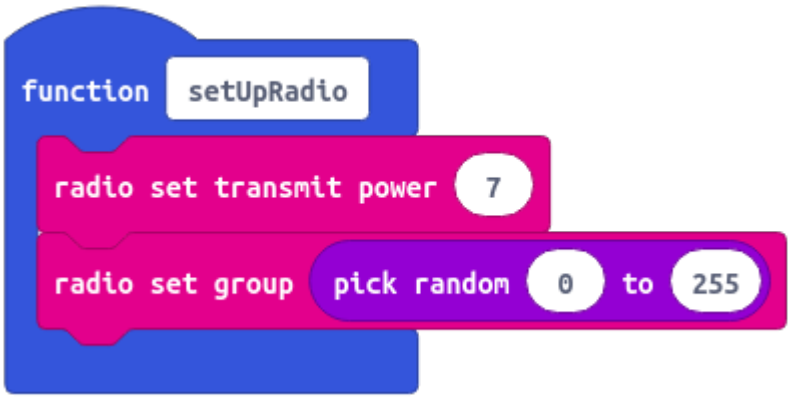
Now to create the function that's being '**called'** to run by the 'on start' block.

Coding tip: Writing **functions** is a good idea, as they make it easier to see what your code is doing! You should give them names that describe what they're doing and that will be understandable by anyone else reading your code.

<u>coding the setUpRadio function:</u>

Whenever the 'setupRadio' function is **called**, it will do just two things:

1. It sets the radio transmitter power, the higher the power, the further away you can be from the receiver.
2. It sets the radio group (communication channel) that your transmitter will be 'speaking' on. Only others who are 'listening' on that channel will be able to hear you. For now we'll let this be a random number.



So the micro: bit, has started up and set up the radio. It hasn't been told to do anything else. So let's tell it what to do when the buttons are pressed. Remember, we want a button A press to transmit a 'dot' and a button B press to transmit a 'dash'.

VARIABLES AND VALUES:

variables and their **values** are always used together. So, for example,

we could create a variable called ageNow and give it a value of 12:

ageNow = 12

or another variable called meetingPlace and give it a value of “Halton Library”:

meetingPlace = “Halton Library”

When we **press the micro:bit button A**, we want to **send a dot**, so we want two variables, **morse** and **controls** to be set like this:

morse = “dot”
controls = “NULL”

The NULL value is just another way of saying “EMPTY” or “NO VALUE SET”.

	button A pressed	button B pressed	shake	face down
morse	“dot”	“dash”	“NULL”	“NULL”
controls	“NULL”	“NULL”	“EOW”	“OAO”

See if you can complete the follow statements:

When we **press the micro:bit button B**, we want to **send a dash**, so we want the two variables, morse and controls to be set like this:

morse = “ ”
controls = “ ”

When we **shake the micro:bit**, we want to **send an EOW**, so we want the two variables, morse and controls to be set like this:

morse = “ ”
controls = “ ”

When we **turn the micro:bit face down**, we want to **send an OAO**, so we want the two variables, morse and controls to be set like this:

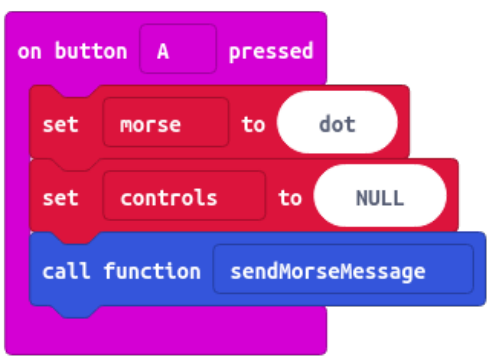
morse = “ ”
controls = “ ”

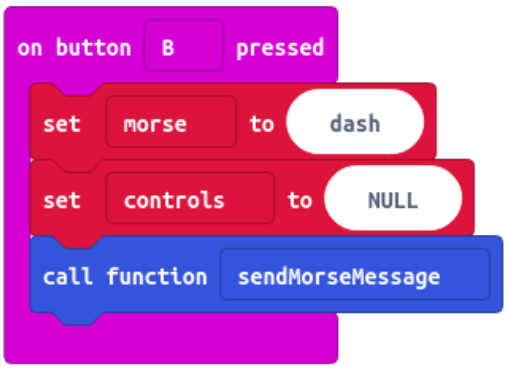
<u>Coding the button presses:</u>

So, just as we decided, whenever the buttons are pressed, the variables are **set**.

Once we’ve set the variables we still need to send them across to whoever is listening so they can receive either a dot, dash, EOW or OAO.

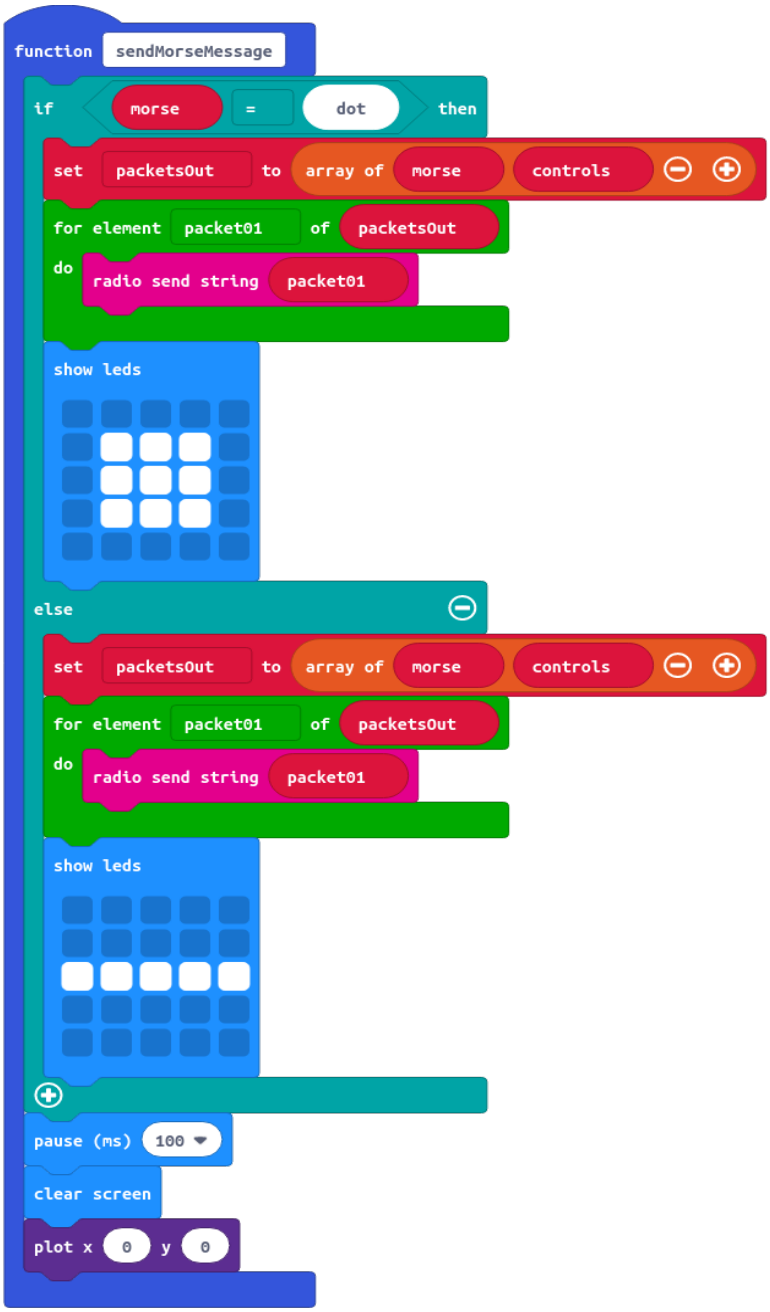
So next we’ll just call a function called sendMorseMessage to do that work for us.





Coding the sendMorseMessage function:

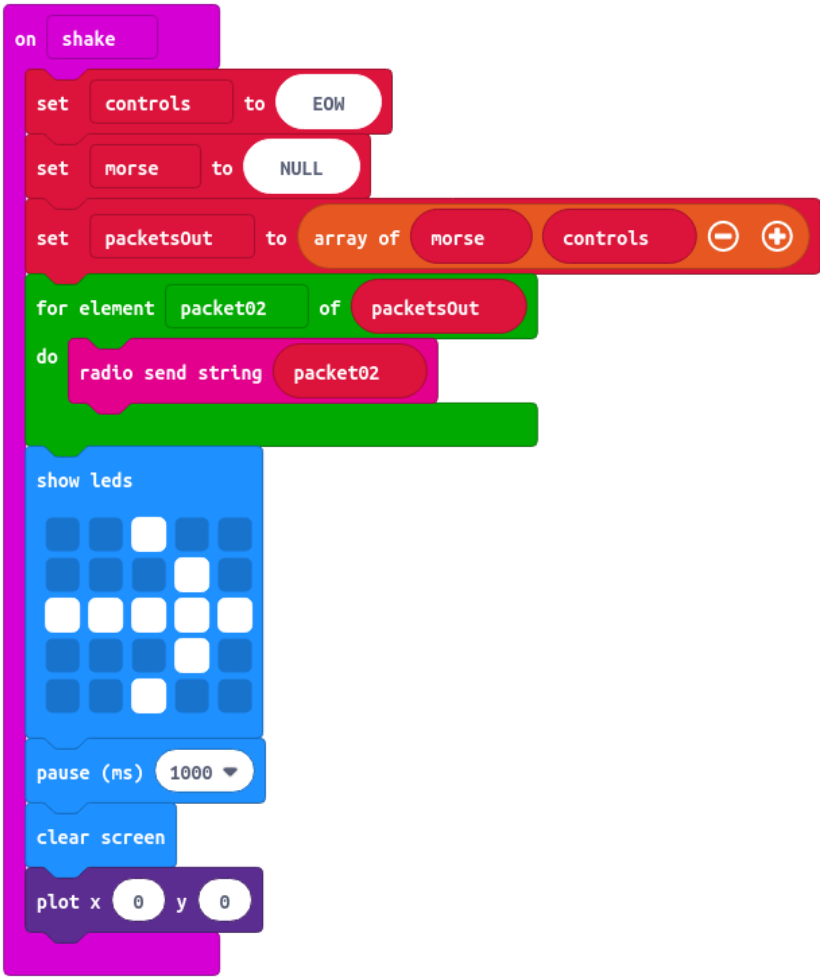
Whenever it's called, the job of this function is to decide whether to send a dot or a dash.
It will then transmit two values over the radio - the value of our morse variable and the value of our controls variable.



Notice how we store our variables before looping through that store and transmitting them one by one.

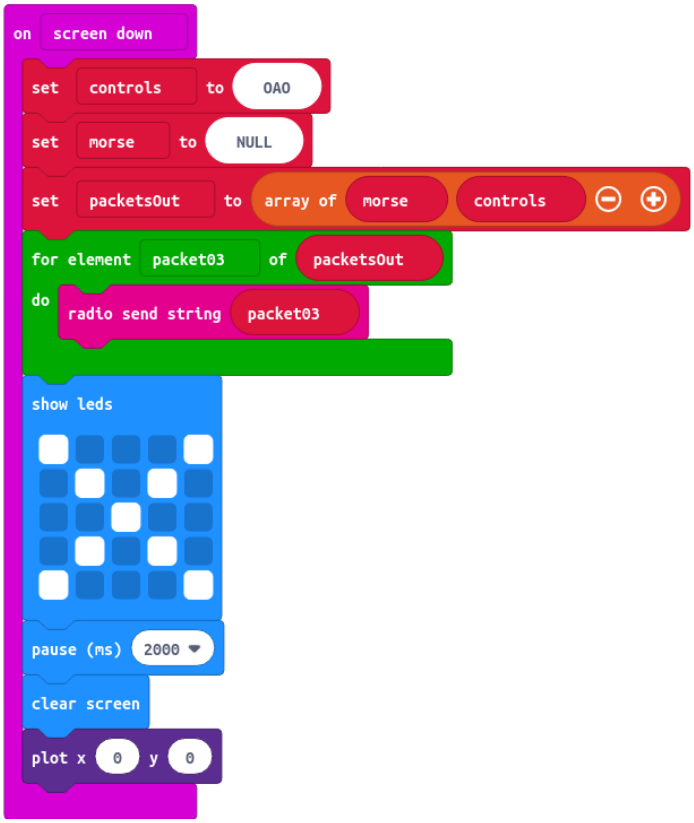
Coding the ‘on shake’ input gesture:

Remember, we’ll be shaking the micro:bit to tell the receiver that “we’ve just finished the the word we were sending, so get ready for either the next word or an over and out”.



Coding the ‘on screen down’ input gesture:

We want the micro:bit to send a message to say we finished talking, or “Over and Out”.
Just as before, all we need to do is to send the appropriate values for the variables **morse** and **controls**.



Done!

You should now have turned your micro:bit into a morse code transmitter!

You can test that you’ve successfully programmed your micro:bit as a morse code transmitter by getting it to talk to echo_challenge (a micro:bit programmed to be a decoder and receiver).

You’ll need to tell the channel administrator which channel you’re using and then ask *very nicely* whether they could please reset echo_challenge to your channel so that you can do the echo test.

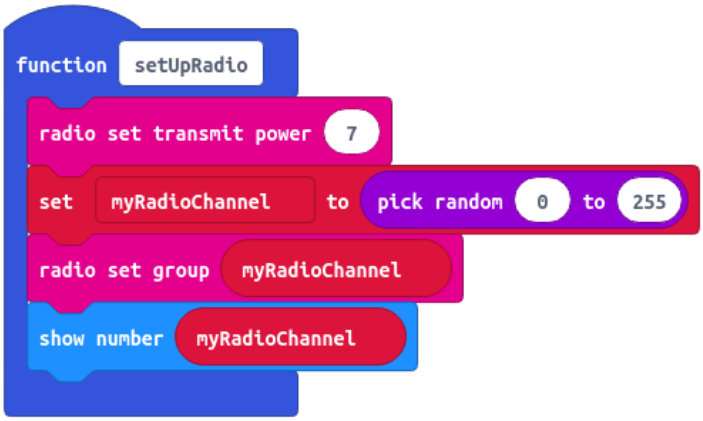
The channel_administrator:

Whoever possesses the channel_administrator (a micro:bit programmed as a transmitter and number decoder), will *become* the Channel Administrator and will have great power!

They'll be in control of which channel any other micro:bit receivers can listen to.

But first, you'll have to update your setUpRadio function to make it tell *you* which random radio channel it has picked.

When your program starts (at **runtime**), write down or remember your 'hopefully unique' communication channel. Remember, this will be a different channel every time you restart your micro:bit.



Hopefully testing was successful, however if your code didn't work as expected don't worry. You're not alone!

Debugging your code:

Remember to change just one part of your program at a time.
If you're struggling to find why something isn't working correctly, get someone else to look at it with you. That usually helps.

The very last thing we need to is to add comment to the top of your JavaScript **source code**, with a short description of what your program does, the name of the coder any other information you think might be useful to anyone else reading your code.

Start a comment with /* and end with */

For example:

/**** my Morse code transmitter*****/

Source code vs compiled code:

Source code is the code we write, using makecode, JavaScript or any other programming language. Source code is understandable by other human programmers.

Compiled code (like the .hex files we flash over to our micro:bits) have been converted (compiled) from source code into a form that computers can understand (basically zeros and ones). Because only computers can understand it, it's also called machine code.