

# Morse Code Workshop

## A micro:bit makecode project (mentor guide)

### Base character types:

`morse_decoder`  
`morse_encoder`  
`morse_transmitter`  
`morse_receiver`  
`morse_transceiver`

### Mentor support characters:

`morse_channel_admin`  
`morse_echo_challenge`  
`morse_messenger_challenge`  
`morse_translator`

# morse\_decoder

decodes user input morse signals into alphanumeric characters

button A == display a dot, add "0" to morse character being built

button B == display a dash, add "1" to morse character being built

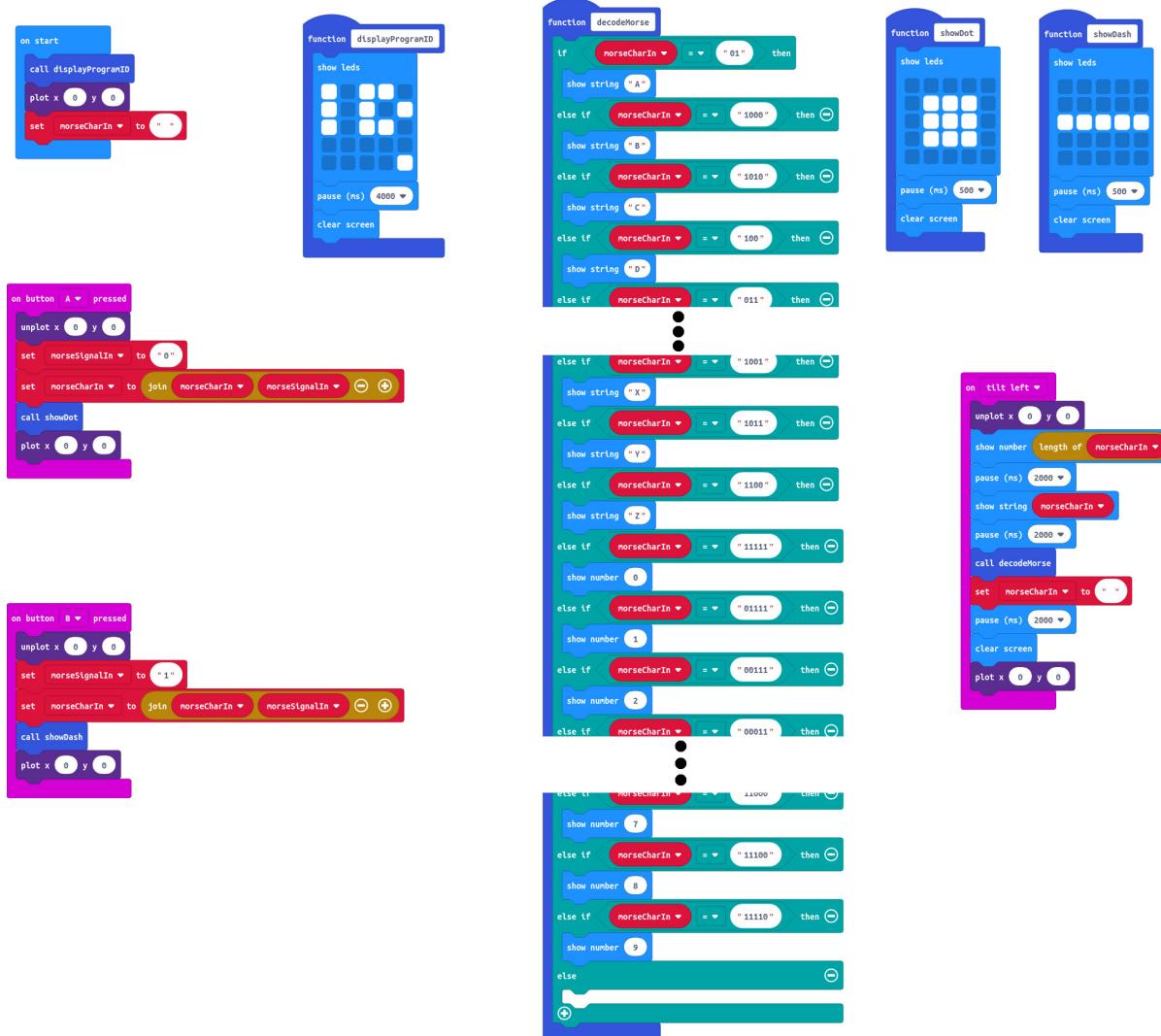
onTiltLeft == confirm inputs, decode the user-built morse character and reset

Program plots and unplots led 0,0 on start to indicate program running.

Program replots 0,0 after show... functions to indicate ready for next input

debugging data: confirmation of length of morseCharIn string and it's contents, left in for now. remove later to clean up, or leave in as useful feature?

The program can only decode single alphanumeric characters at a time at the moment because there is no way to recognise an inter-character pause. Therefore characters 'BE' ( -... .) would not be distinguishable from the number '6' (.... )



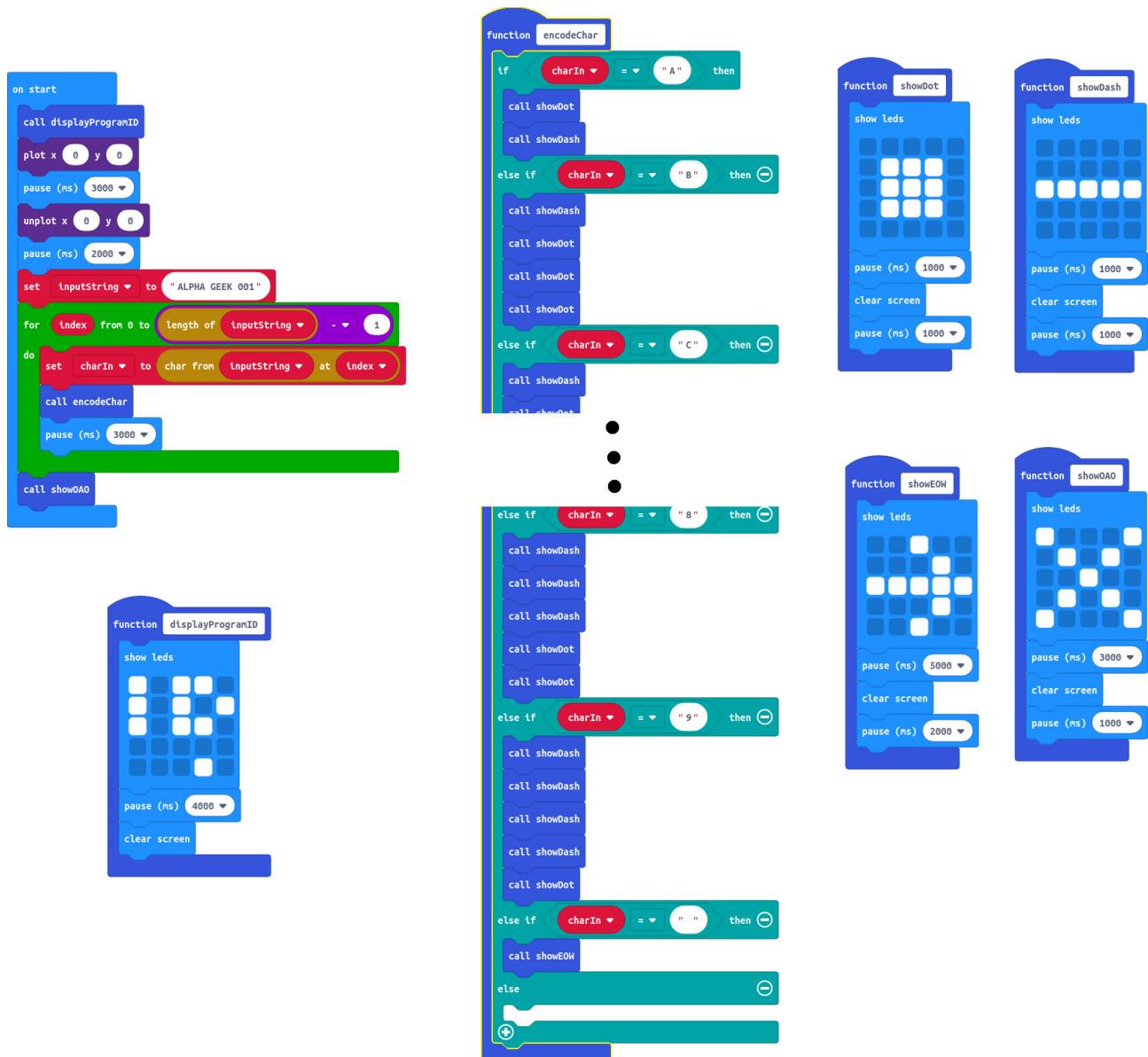
## morse\_encoder

iterates through each character of a pre-defined input string, encoding each into morse code and outputting the morse signals on led display.

Program plots and unplots led 0,0 on start to indicate program running.

timing of dots and dashes, intra and inter character spaces, inter word spaces roughly adopted:

char 1, intra-char 1, inter-char 3, inter-word 7



## morse\_transmitter

simply echoes morse signals corresponding to user input to the LED display, then transmits that morse signal.

channel must be configured.

To avoid that annoying transmission delay, we must send packets BEFORE we show LEDs!

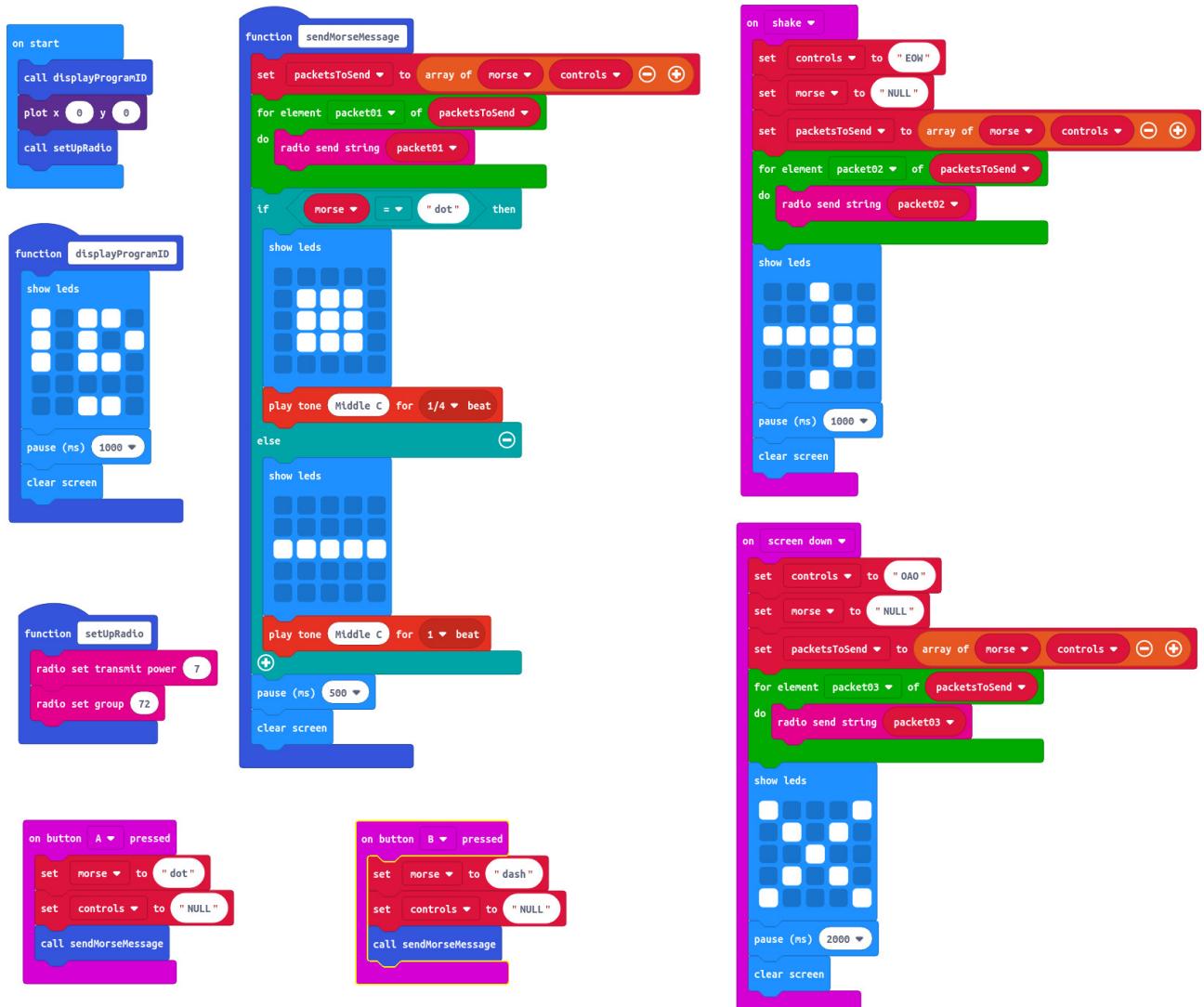
Program plots led 0,0 on start to indicate program running.

button A == send "dot" and display a dot

button B == send "dash" and display a dash

onShake == send "EOW" and display end of word symbol "->"

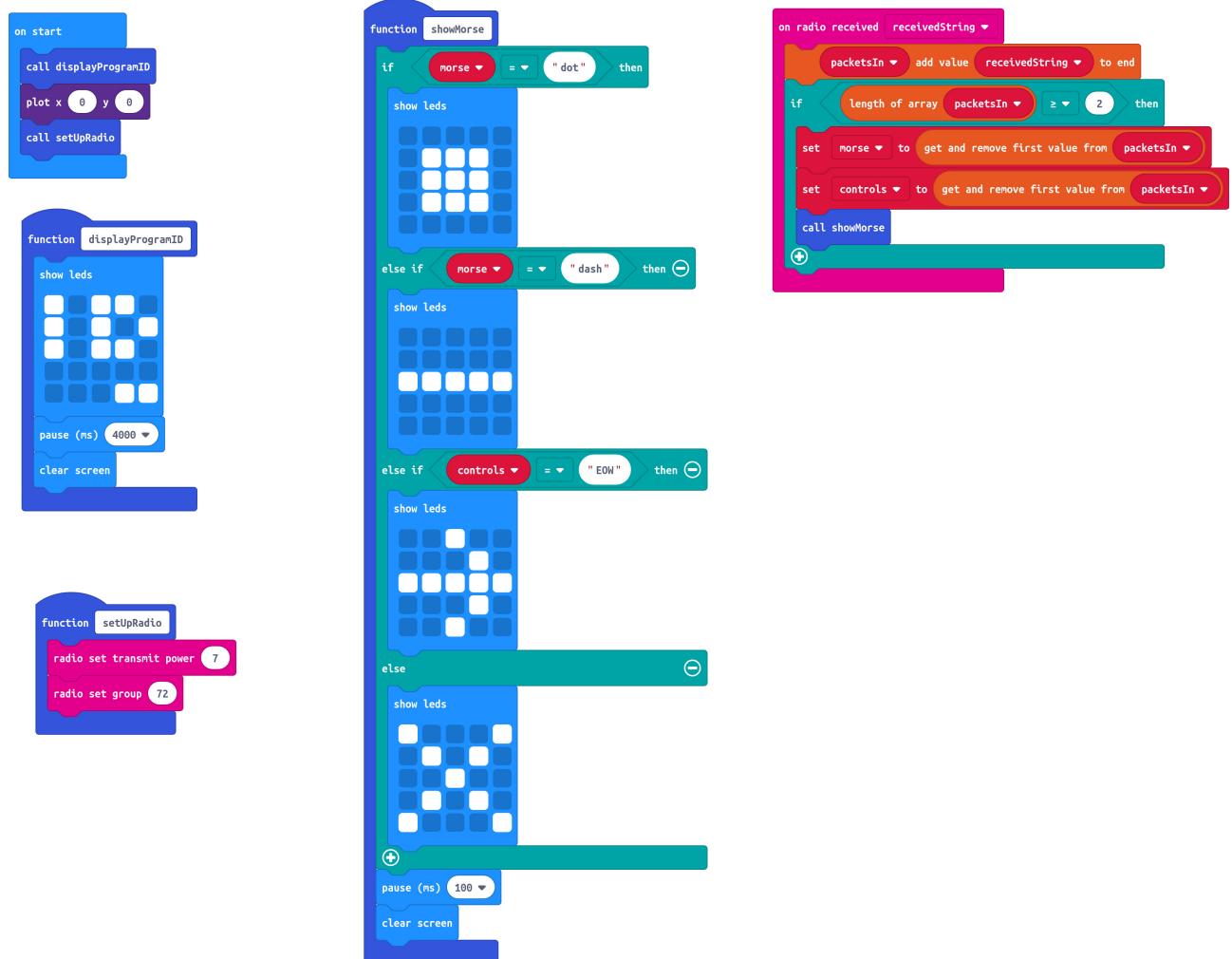
screenDown == send "OAO" and display over and out symbol "X"



## morse\_receiver

simply receives morse signals and echoes each signal received to the LED display  
channel must be configured.

Program plots led 0,0 on start to indicate program running.



## morse\_transceiver

“dot”, “dash”, “EOW” and “OAO” are preferred to something like 0,1,2,3 as values as they keep programs more readable and maintainable.

showing led output on both transmitter and receivers seem good as feedback to users.

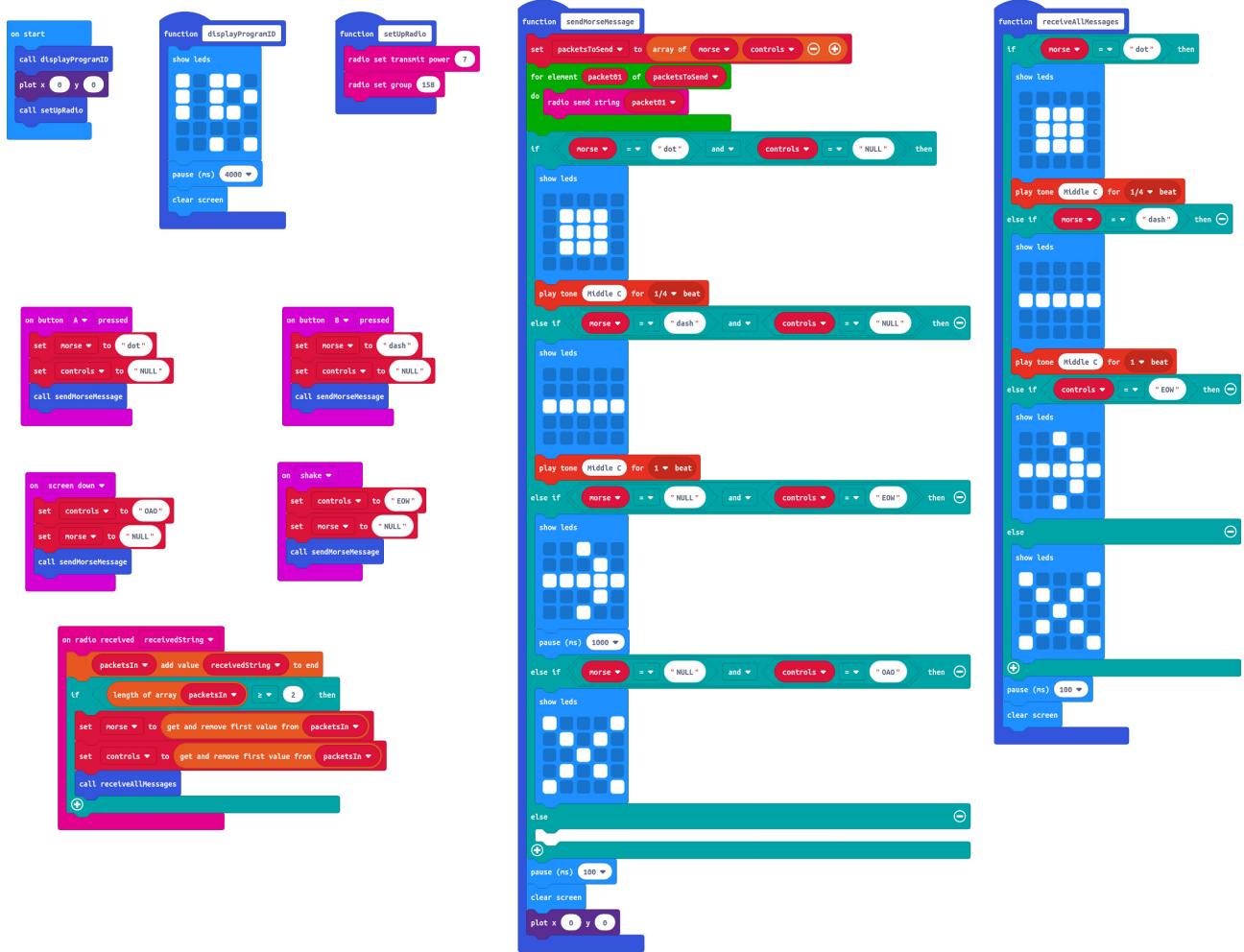
All transmitters must send packets BEFORE showing LEDs!

button A == send “dot” and display a dot

button B == send “dash” and display a dash

onShake == send “EOW” and display end of word symbol “->”

screenDown == send “OAO” and display over and out symbol “X”



# morse\_channel\_admin

## a transmitter and number decoder

used to change the channel on which the echo\_challenge character communicates. when a young coder wants to communicate with echo\_challenge, we use the channel administrator to switch echo\_challenge temporarily to their unique channel, switching back to the default when they've finished.

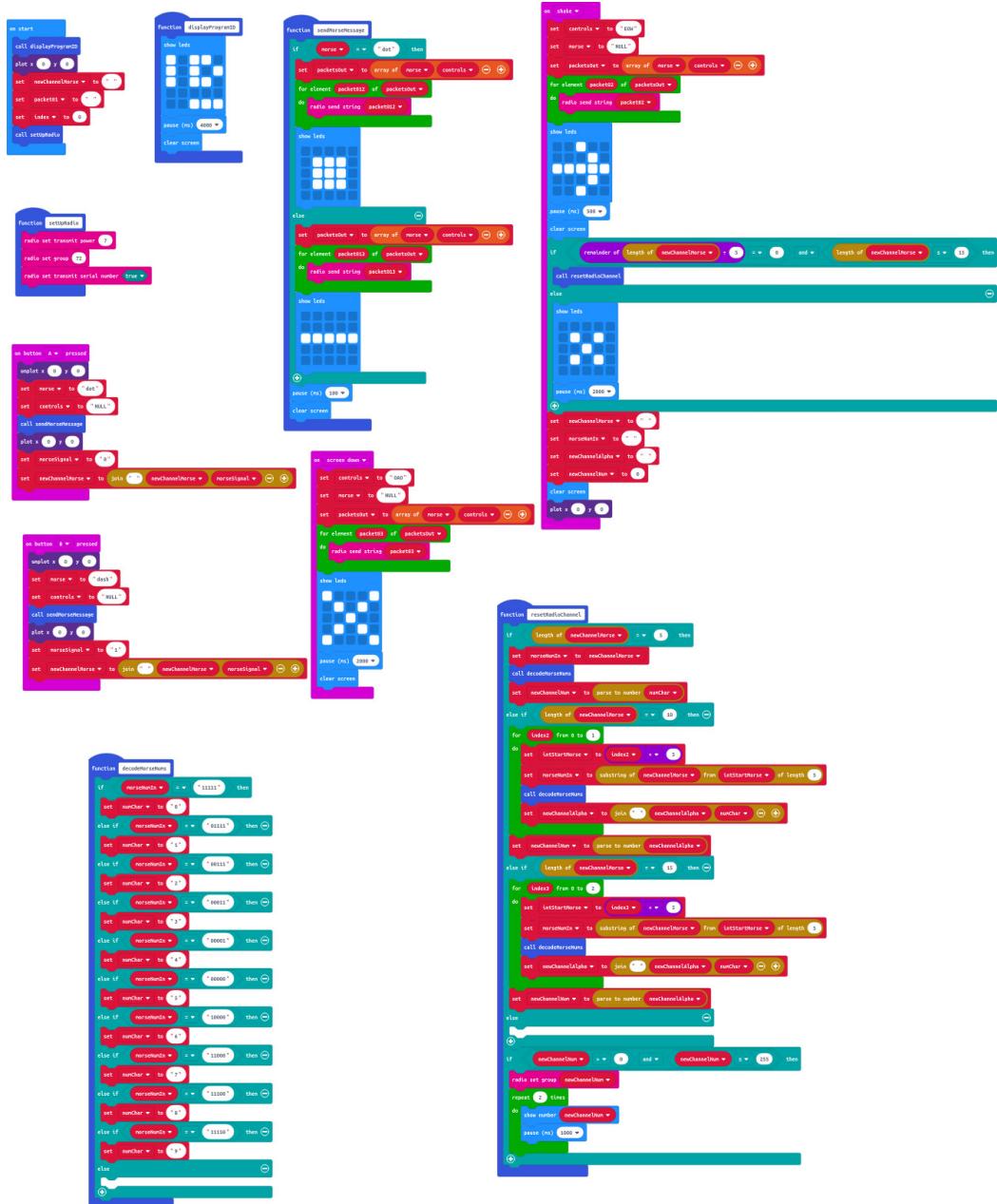
button A == send "dot" and display a dot

button B == send "dash" and display a dash

onShake == send "EOW" and display a "->" and reset radio channel to the number the which the morse input decoded

screenDown == send "OAO" and display "X"

defaults to  
channel 72 on  
start. all  
transmitters  
must send  
packets  
BEFORE  
showing LEDs!



## morse\_echo\_challenge

## a **decoder** and **receiver**

on radio received (dot or dash) == build a morse string of zeros and ones

on radio received (EOW) == either reset channel (if talking to channel administrator) or decode morse to alphanumeric character (if not talking to channel administrator).

displays random alphanumeric characters to LED.

receives Morse signals from a transmitter-type device, which it decodes and compares with the displayed alphanumeric character

accepts channel switch instruction from channel\_administrator device (based on packet serial number field)



the serial number of the channel administrator is hard coded and checked against received\_packet.serial\_number, so echo challenge knows when it's being asked to switch radio channels rather than decode morse.

# morse\_messenger\_challenge

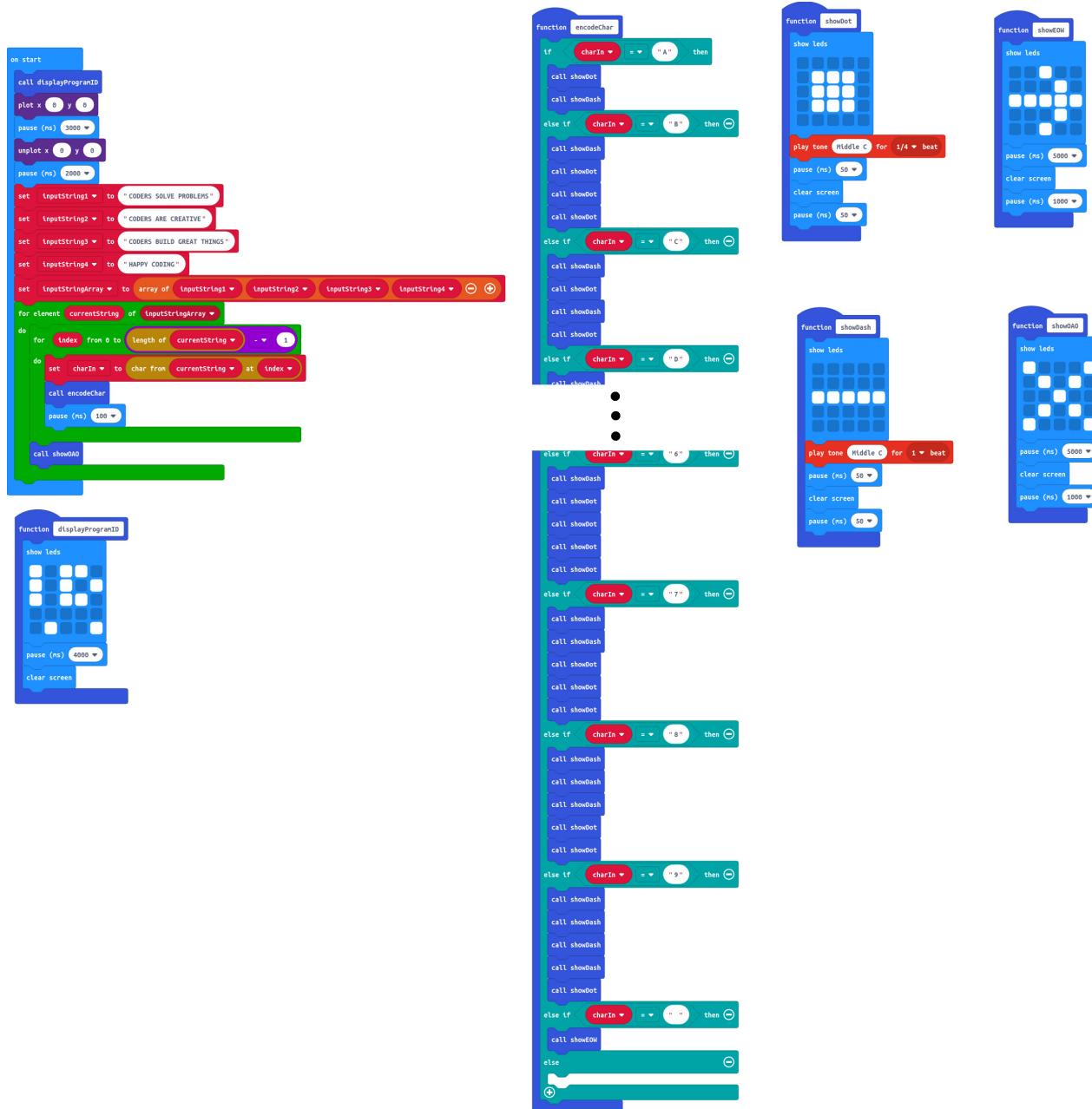
## an encoder

iterates through each character of a pre-defined (hard coded) input string, encoding each into morse code and outputting the morse signals on LED display.

iterates through each string of a pre-defined array of strings, sending the OAO signal at the end of each one.

timing of dots and dashes, intra and inter character spaces, inter word spaces roughly adopted:

char 1, intra-char 1, inter-char 3, inter-word 7



## morse\_translator

### a type of **decoder**

decodes user input Morse signals into alphanumeric characters. same as `morse_decoder`, but without debugging outputs

button A == display a dot, add "0" to Morse character being built

button B == display a dash, add "1" to Morse character being built

onShake == decode the user-built Morse character and reset

Program plots and unplots led 0,0 on start to indicate program running. Program replots 0,0 after show... functions to indicate ready for next input

