An AustSTEM Teaching Resource

Blink a LED to send a distress signal

Context

- The electric telegraph system

School Years

- Stage 3

Curriculum:

- Technologies (Australian)

- Science & Technology (NSW)

Learning Outcomes

Blink a LED to send a distress signal

- Learn about the invention, spread and significance of the electric telegraph. *[Electric circuits, history of communications and commerce]*
- Learn about digital systems *[integration of peripherals with the Kookaberry]*
- Observe how data can be represented by numbers and symbols *[Morse Code]*
- Write and edit a programme in a visual programming language *[KookaBlockly]*
- Recognise that steps in algorithms need to be accurate and precise *[Length of sleep times are critical]*
- Learn about, use, and control different output devices *[LED and Buzzer]*
- Learn how to trouble shoot through trial and error whilst programming in real time *[Vary sleep times]*
- Learn how to use branching statements (if/do) in programmes *[If button pressed]*
- Learn about repeat loops *[Repeat a dot or a dash three times for the letters S and O]*
- Learn how functions can reduce the number of coding steps *[Set up functions for each of S and O]*
- Learn how to position and write text on a screen *[Write screen prompts for saved KookaBlockly file]*
- Learn how to input and change user data in a programme *[Pin numbers and sleep duration]*
- Save and rename files across multiple locations *[Add a .kby suffix]*

Blink an LED to send a distress signal

Blink an LED to send a distress signal

Overview

Students will learn about the invention of the electric telegraph and how to send a distress signal in Morse Code using the Kookaberry microcontroller STEM platform programmed using the KookaBlockly visual programming editor

An online version of this resource can be found on the AustSTEM Digital Learning Hub website.

Context

In the middle of the 18th Century, Samuel Morse co-invented the electric telegraph and the Morse Code. His code consisted of just two data states - a dot and a dash - to represent the letters of the alphabet; the numbers 0-9; and a few punctuation marks and symbols.

The electric telegraph is basically an electrical circuit between two destinations. The earliest systems consisted of single wires strung on poles.

An electrical voltage was applied to the overhead wire by batteries at a transmitting station, and the circuit was completed by the electric current travelling back to the battery through the earth. A telegraph key was held down briefly to make a short signal, a dot, and slightly longer for a dash.

The electric telegraph spread around the world transforming commerce and influencing society with its instant global news. When Marconi invented long distance radio in 1901, morse code was used to transmit text messages.

The British Empire used telegraph cables, both overland and under the sea, to connect its Colonies together in what was called the All Red Line. The last link was the amazing 3,200km overland telegraph between Port Augusta in South Australia, and Darwin completed in 1902.

After the Titanic sank in 1912, telegraph operators and their radios were required to be on all ships so that they could call for help and transmit their location if they were in distress. Although Samuel's code is no longer in commercial use, the universal name for his distress signal - SOS - is still well known today.

FunFact: SOS does not mean Save Our Souls or any other message. It is just a very recognisable and easy to remember sequence of morse code letters - dot, dot, dot, dash, dash, dash, dot, dot, dot.

***[Question:** What do think is the reason why the letter E is just a single dot in Morse code?]*

Blink an LED to send a distress signal

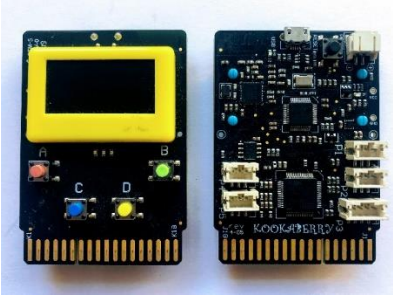
Prior Knowledge

Before commencing this set of resources, teachers and students should be familiar with the following concepts and technology fundamentals.

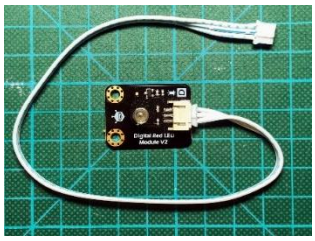
- Copying, moving, and pasting files
- Digital Systems: Introduction to digital systems
- Using the Kookaberry: Getting started with the Kookaberry
- Using KookaBlockly: KookaBlockly

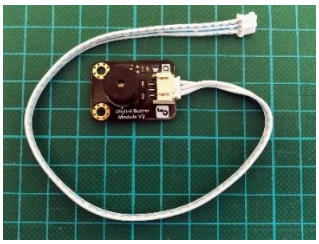
Resources

Kookaberry & USB lead

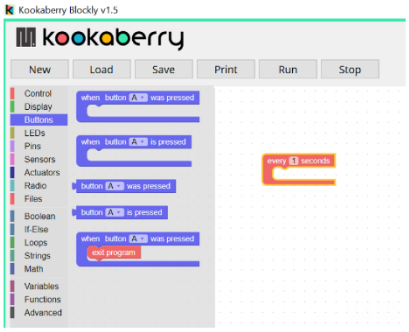


LED Module or Buzzer Module plus 3pin JST peripheral lead





KookaBlockly Visual Programming Editor



Blink an LED to send a distress signal

Connecting up

Connect the Kookaberry to your PC or Mac, and plug the LED Module into socket P1 on your Kookaberry. Plug the Buzzer Module into P! to hear a sound rather than blink a light. This is what the telegraph operators heard when they were receiving messages.

The Morse Code Distress Signal

In Morse Code. "S" is three dots and "O" is three dashes. The timing is as follows. Pick a duration time of 0.2 secs for a dot

- a dash is 3x the duration of a dot
- the silence between dots and dashes is the same as a dot.
- the silence between letters is equivalent to 3x dots or a dash
- the silence between words is a minimum of 7x dots

SOS Teaching Resources

In these resources, students will programme the Kookaberry using KookaBlockly to send the international distress signal (SOS) using several different coding statements and commands.

Downloads of the KookaBlockly code for these resources can be downloaded from a zip folder found in the sidebar to the right of this online resource page on the AustSTEM Digital Learning Hub

Blink an LED to send a distress signal

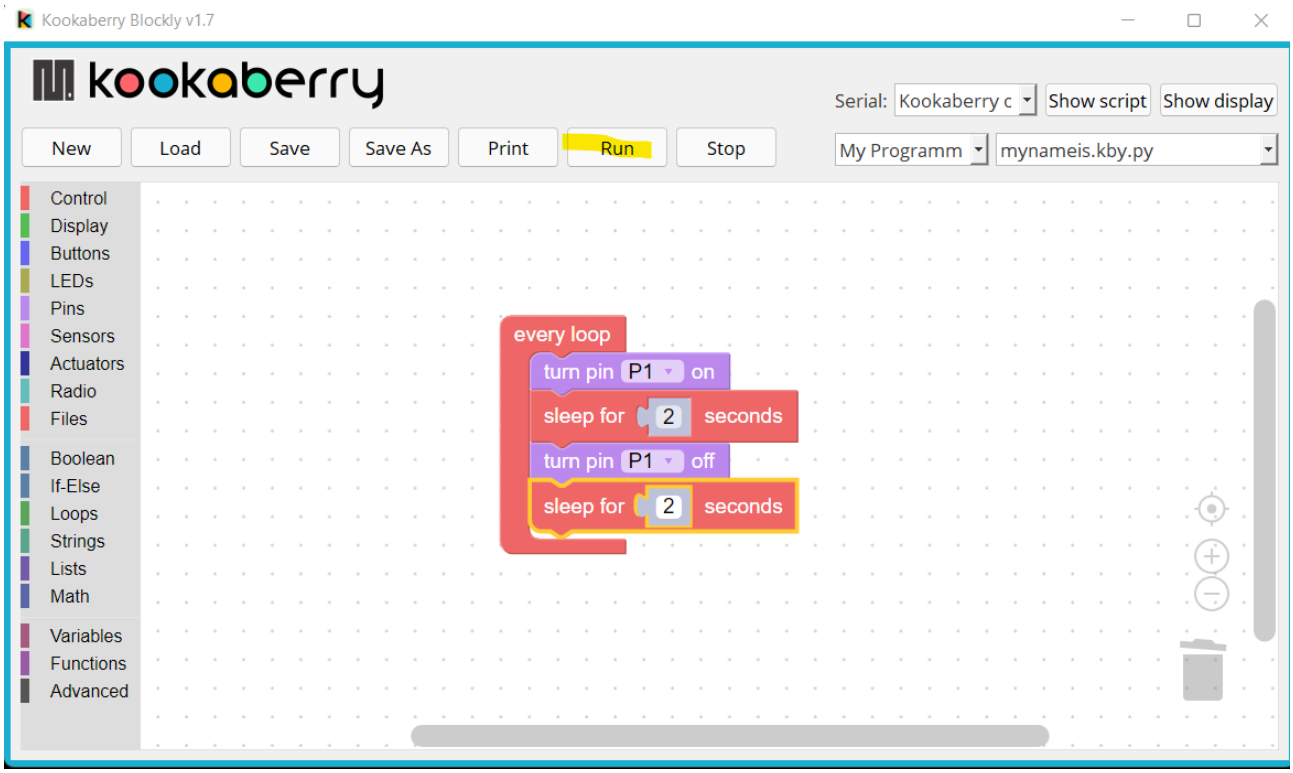
SOS Resource 1: Blinking a LED

Open KookaBlockly; open the Control menu; and drag the "every loop" Control block onto the canvas. Then go the Pins menu and drag the "turn pin....on" into the control loop. The default Pin number is P1, but this can be set to other Pins from the drop down menu within the block.

Now go back to the Control menu and drag the "sleep for ....seconds" into the control loop. Set the sleep time for the amount of delay required (2 secs in the example shown)

Now turn P1 off for the required number of seconds by dragging the blocks into the control loop.

The KookaBlockly code in the downloaded zip folder for this resource is called ***SOSblink.kby.py***. **Run** the programme.

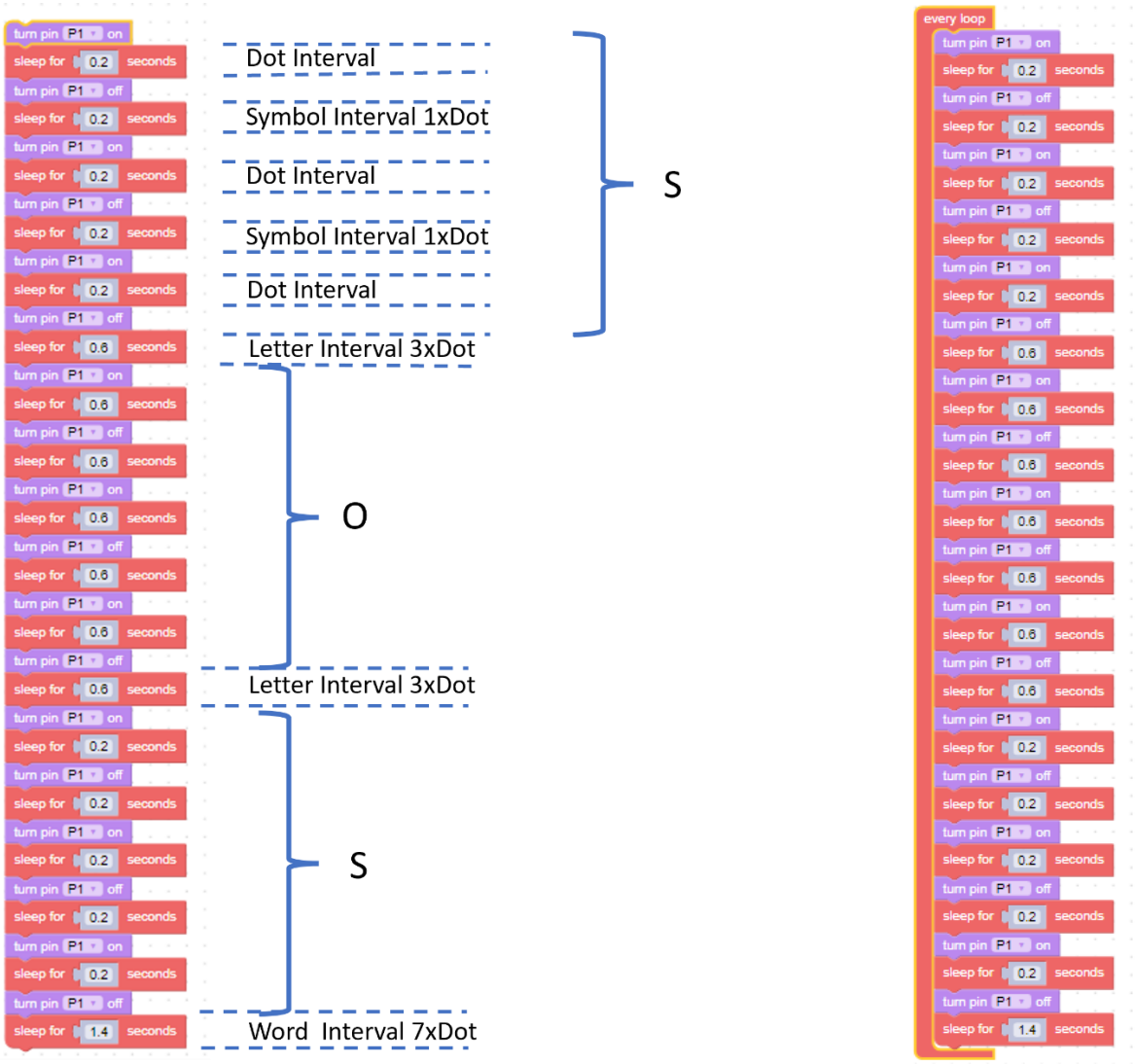


Blink an LED to send a distress signal

SOS Resource 2: Simple code repetition

To send the SOS message, take the blink code created above and create a dot and dash by varying the sleep interval between Pin ON and Pin OFF. The letters S and O are created by repeating the code for dots and dashes, taking care to use the correct spacing (sleep interval) between symbols (a dot or a dash); letters (S or O) and words (SOS). The resulting code will look like this in KookaBlockly.

Run the programme and see the distress signal sent once only. Adding a control loop will repeat the message over and over.



*The KookaBlockly code in the downloaded zip folder for this resource is called **SOSsimple**.kby. It shows the control loop programme blocks greyed out. This means that the instructions will be ignored when the programme is Run. Only one SOS message will therefore be sent*

The blocks within the loop can be enabled or disabled by right clicking on the block to reveal the block's dropdown menu. The same menu can be used to duplicate a block or set of blocks.]

Blink an LED to send a distress signal

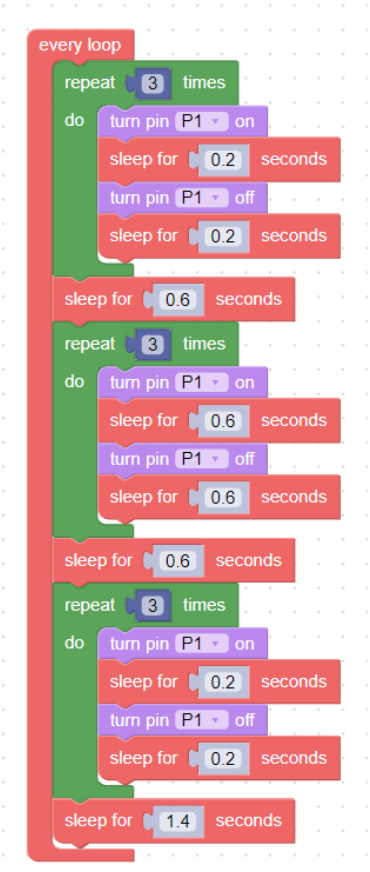
SOS Resource 3: Using the repeat loop

Computers are great at repeating instructions, or sets of instructions, at blinding speed. Programmers take advantage of this by setting up one instruction and then telling the computer how many times to repeat it. These instructions are called Loops and can be found in the Loops menu.

Drag the repeat 10 times loop onto the canvas and place the Dot sequence of steps inside it. Change the repeat times to 3. Repeat for the Dash sequence.

Now create just one dot and one dash sequence and drag them into the repeat loops as shown below.

The KookaBlockly code for this resource in the downloaded zip folder is called ***SOSrepeat.kby.py***. **Run** the programme.



.....Now hear it

Press the reset button on the back of the Kookaberry and swap the LED module for a Buzzer Module to hear the distress signal the same way as radio operators did in the past. If you have a problem, disconnect and reconnect the Kookaberry to your computer

Blink an LED to send a distress signal

SOS Resource 4: Using repeat loops and functions

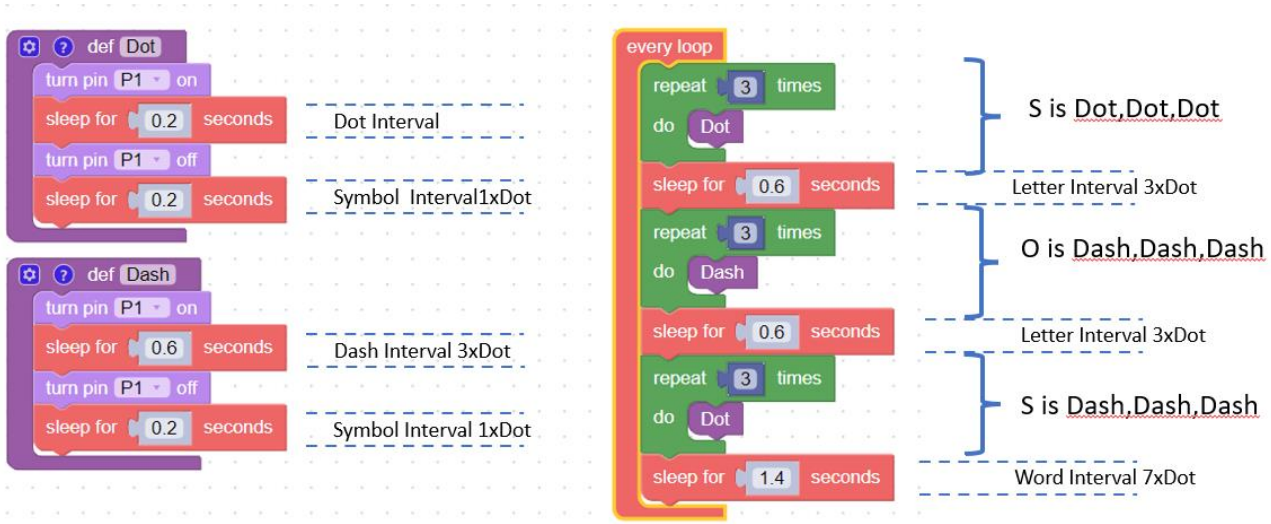
One aim of good programming is to use as few lines of code as possible. This reduces programming errors (fewer instructions to get wrong) and takes up less chip memory.

The number of lines of code in the SOS programme can be reduced further by using Functions. Functions are specific tasks that you want a computer to do - such as "Move 10 steps" or look for a name in a table and return it to the programme. They can also be a specific thing requiring a number of programming actions - in this case a Dash or a Dot.

Drag the def "my function" block from the Functions menu onto the canvas and name it Dot by overwriting "my function" text in the block. This will create a separate "Dot" function in the Functions menu. Create the code for a Dot and bring it into the function. Repeat for a Dash function

Now all you have to do is to drag the Dot and Dash functions from the Functions menu into the repeat loops as shown below.

The KookaBlockly code for this resource in the downloaded zip folder is called ***SOSfunct.kby.py***. **Run** the programme and then run it again with the Buzzer connected.



Blink an LED to send a distress signal

SOS Resource 5: Using the IF statement and saving your programmes on your Kookaberry

In this lesson Plan students will learn how to save their code creations on their Kookaberries so that they can be run independently from a computer. They will also learn how to use the "if-Do" command

This will require the addition of the following instructions to any one of the programmes created in the previous resources. The example shows the additional code being used with the programme from Resource 3

Exit Programme

Drag the "when button A is pressed" block from the Buttons menu onto the canvas. Drag "display clear" and "display show" blocks from the Display menu and place them inside the Button block. Then drag the "exit program" block from the Control menus underneath "display show".

Command Instruction

Drag the "when button A is pressed" block from the Buttons menu onto the canvas and insert it between the "every loop" block and the rest of the instructions. Change the button to C in the block's drop-down menu.

Screen Prompts

Select the "display text value=" block from Display menu and drag it onto the canvas. Overtype the "Hello" text with "C to send SOS" and start the text at the coordinates shown. Repeat for the "A to Exit" text display.

*[**Note:** the screen size is 128 pixels across and 64 pixels down]*

*[**Tip**: Don't be afraid to use trial and error to position the text on the screen correctly. Just keep changing the "x' or "y" values until you get it right. Remember that "x=64" and "y= 32" will start the text in the middle of the screen]*

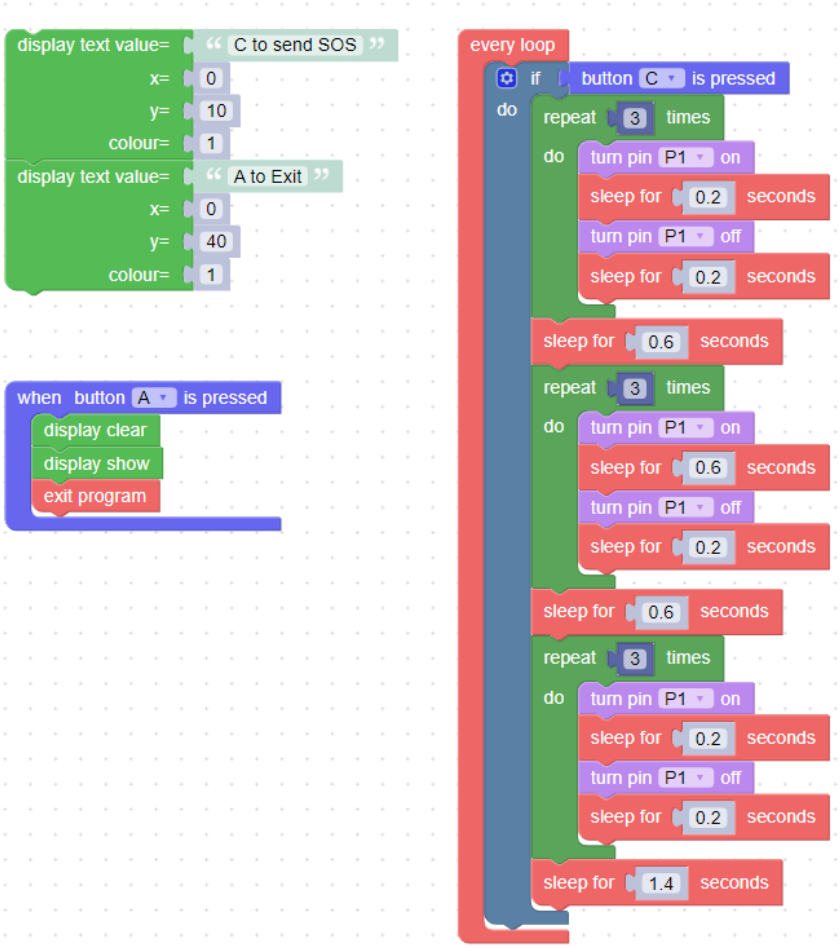
Blink an LED to send a distress signal

Save the file

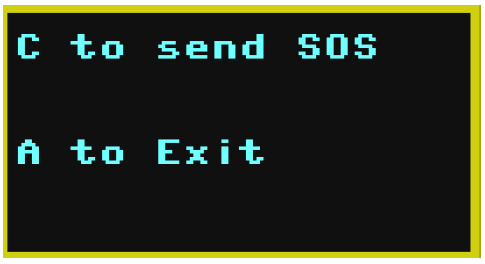
Save your app with the .kby suffix. Go to the KookaBlockly Support page for information on saving it in Kookaberry Scripts and on your Kookaberry.

Code Example

The KookaBlockly code for this resource in the downloaded zip folder is called ***SOSButton.kby.py***. **Run** the programme.



Kookaberry Screenshots

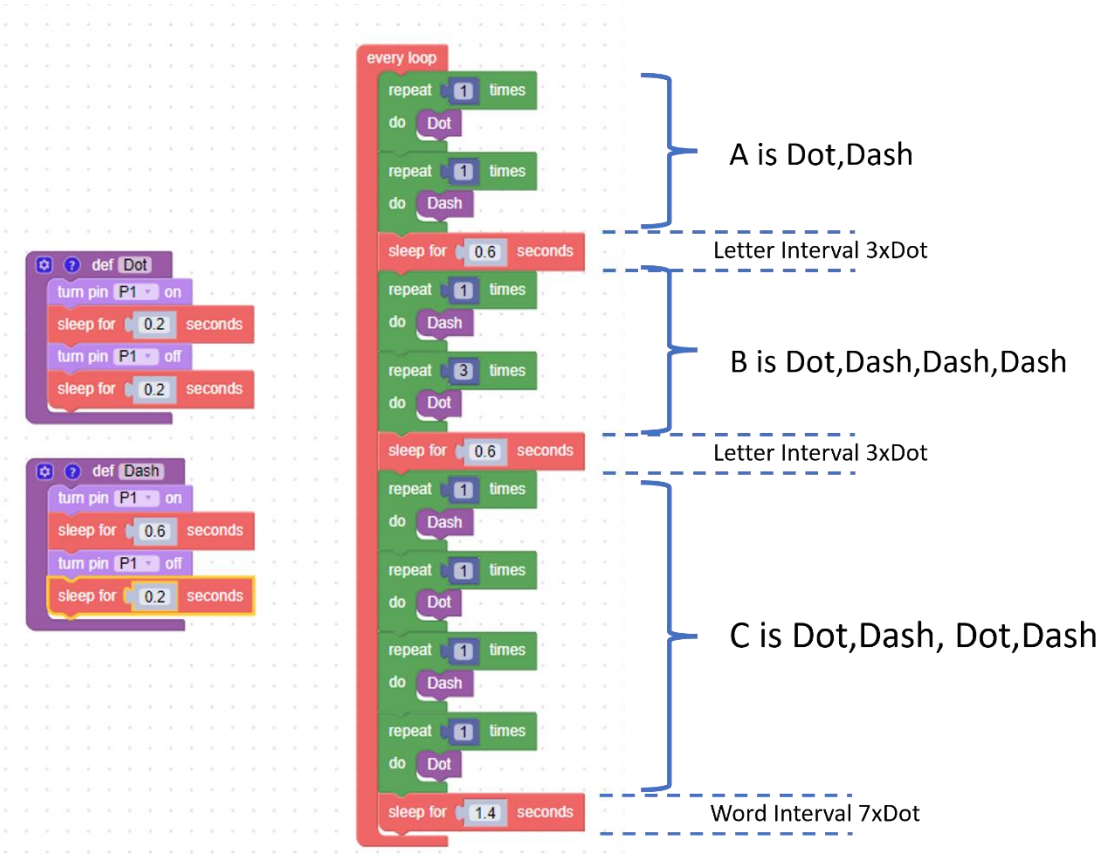


Blink an LED to send a distress signal

Learning Extensions

Programme your own morse code signals

Create other combinations of signals eg ABC as shown below



Blink an LED AND sound a buzzer

Modify your code to add the buzzer at P2 and have both operate at the same time. Or blink one of the embedded LED's on the Kookaberry board as well as sound the buzzer.. *[**Hint**: Look in the LEDs menu in KookaBlockly]*

Experiment with the MorseCode App

Learn more about Morse Code by running the MorseCode app. Watch the dots and dashes being created and transmitted between Kookaberries.