Saturday Kids

Level 2 Python Applications

Lesson 4 - Intro to Radio

By the end of the lesson, students should be able to:

Intro to Radio

- describe how a radio works
- use radio functions to send and receive messages(string & tuple), turn the radio on, and set up a radio channel
- recall and import libraries
- create and display images on the micro:bit
- use the sleep() function to slow down the program
- use the display.clear() to clear the micro:bit display
- perform simple debugging via reading documentation
- recall and use conditionals (if, if elif else) to determine if a button is pressed, if the message is received and if the message received is a particular string
- recall and use comparison operators
- recall and use while loops to create a forever loop
- recall and use variables to store data
- identify tuple and access values in a tuple
- describe signal strength to be stronger/weaker depending on distance from the transmitter source
- use 'None' to check for a null value
- use 'is not' to check if an object is not the exact same object

Version
Date: January 2020
Format: 8 lessons x 2 hours
Important! View speaker notes for details

<8) =) ;D :)/>
CODE IN THE COMMUNITY

Things to note

- 🔌 **Unplugged** = Activities not involving technology (Videos, Kinaesthetic activities etc.)
- 💬 **Discussion** = Get the students to think and respond about a question
- 🎎 **Guided** = Demonstration → Instructor does the activity while the student mimics)
- 👐 **Unguided** = Instructor will give the students the task and show what the final result should look like and give the students a certain amount of time to do it by themselves before moving on to "Check for Understanding"
- 🎯 **Check for Understanding** = Instructor will go through the solution with them or get a student to share the solution
- 🚀 **Sandbox** = Free-Play (Students recap what they learnt from the entire day by creating a project)
- 🎁 **Bonus** = This is given to students who are fast-paced

Materials Needed

Per student:

- 1x microbit set
  - 1x microbit
  - 1x usb
  - 1x battery pack
  - 2x AAA batteries
- 1x Chromebook/Laptop

Per instructor:

- 1x microbit set
  - 1x microbit
  - 1x usb
  - 1x battery pack
  - 2x AAA batteries

Frequently Asked Questions

“What if I can't finish the activities for that particular day? ”

- In the event that you can't finish all of the activities in the given time, DO NOT rush to finish the concepts and just continue where you left off the next week.
- The bonus activities are for the faster students that have completed the general task that was given to the whole class. You do not need to cover this with everyone.

“How do you know you’ve been teaching the right way?”

- When students are able to create their sandbox with minimal to no help from you.

Frequently Asked Questions

“What is the purpose of this course?”

- For students to practice applying Python knowledge learnt previously on a micro-controller and build structures with Strawbees to present their creations to tasks given.
- Students also learn more basic coding concepts through Python and use them via computational thinking.

“What is computational thinking?”

- Computational thinking allows us to take a complex problem, understand what the problem is and develop possible solutions. We can then present these solutions in a way that a computer, a human, or both, can understand.

Frequently Asked Questions

“Why must I follow the speaker notes and teach in a certain way? I prefer to freestyle.”

- For follow-up purposes as there will be cases where you might be unable to teach your class on a particular day and another instructor will need to cover you.

Frequently Asked Questions

“ Can students bring home the Strawbees structure?”

- No, but they can take pictures of their structures before dismantling them.

“ Can students bring home the microbit set?”

- No, but they can take videos of their projects/ pictures of their structures before dismantling them.

“Can students buy the Strawbees or microbit set?”

- No.

PYTHON

LEVEL 2

CODE IN THE COMMUNITY

Attendance Taking

Please ensure that your attendance has been taken at the start of every lesson.

You will need to attain 80% attendance in order to graduate from this course.

Lesson 1	Lesson 2	Lesson 3	Lesson 4	Lesson 5	...
✓	✓	✓	✓	✓	...

15 min

Recap & Agenda

Today's Lesson

- Introduction to Radio
- Firefly
- Emoji Messaging
- Treasure Hunt

45 min

Radio

Unplugged

10 min

What is Radio?

Unplugged

How do messages get passed between walkie talkies?

Unplugged

Using radio waves!

<8) =) ;D :)/>
CODE IN THE COMMUNITY

Unplugged

Will the walkie talkies work if the channel is changed?

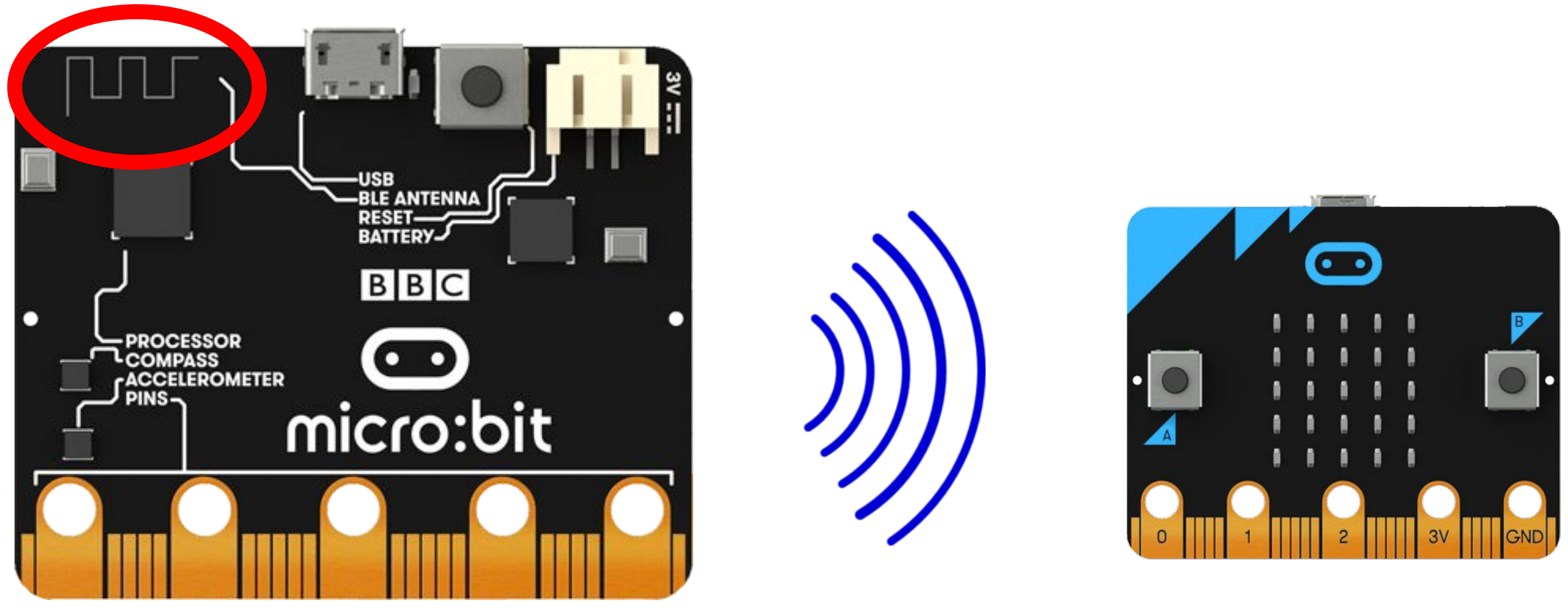
Unplugged

Both walkie talkies have to be on the same channel

Unplugged

Micro:bit Radio Feature

Micro:bit uses radio waves to send and receive messages, and has channels



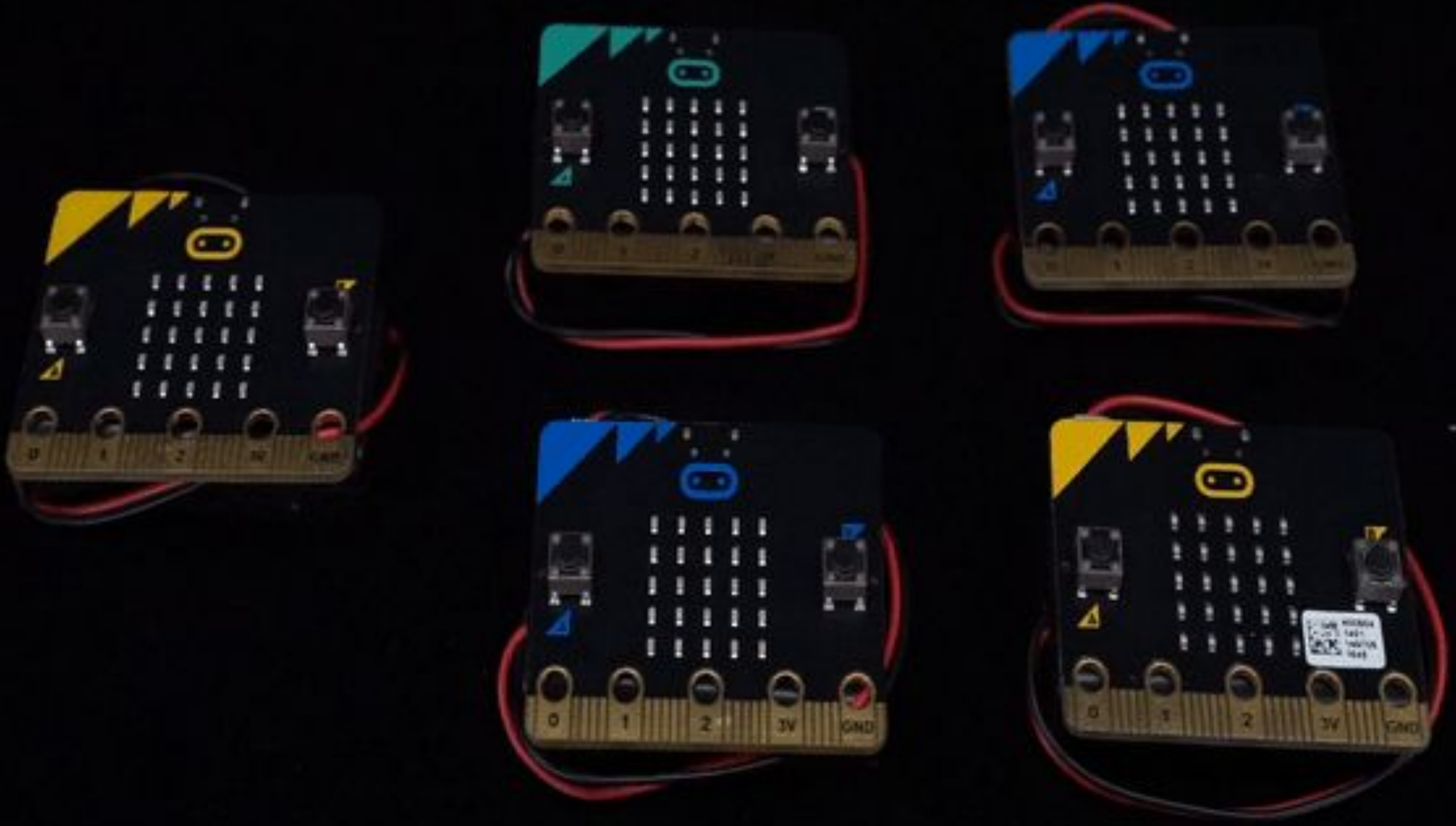
Transmitter

Receiver

Guided

15 min

Firefly


0
1
2
3V
GND

🎎 **Guided**

Programme the micro:bit

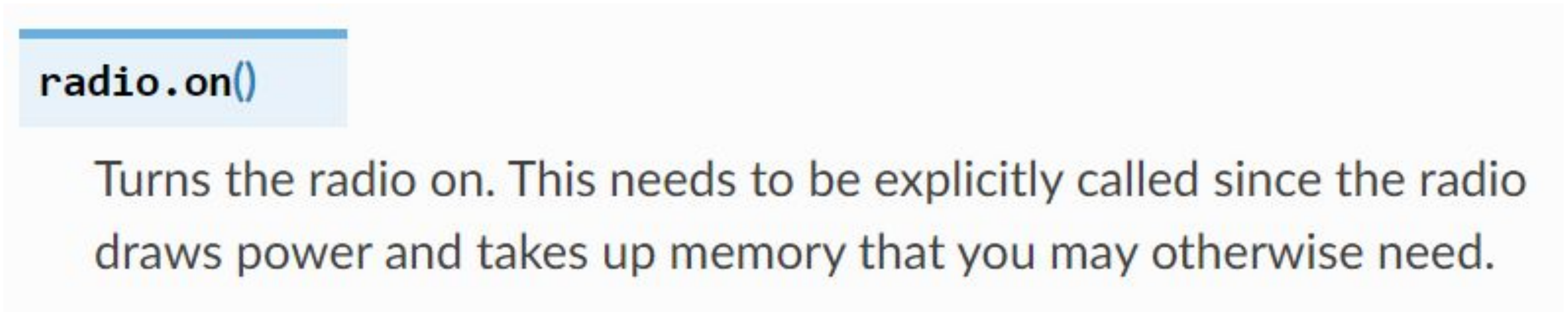
Send a message when button A is pressed. When a message is received, display a square image.

Guided

Find the Functions to turn the radio on and configure the radio channel

Guided

Turn Radio On

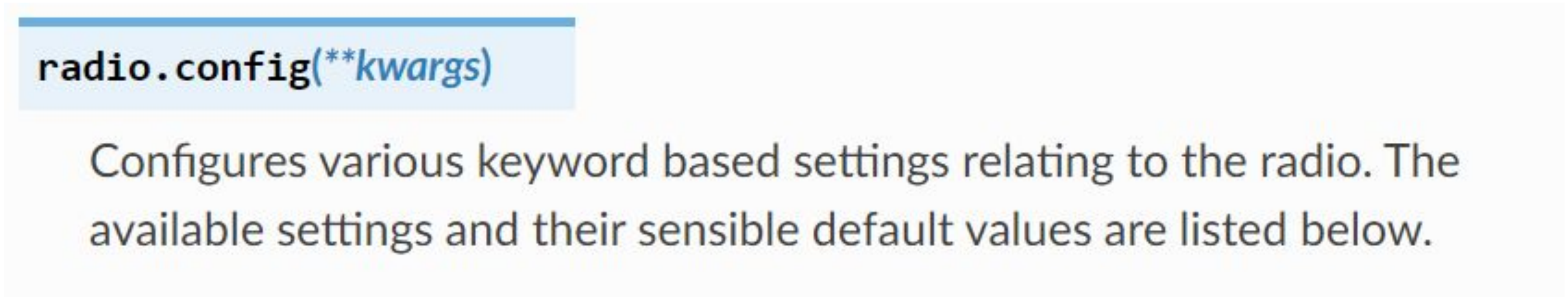


```
radio.on()
```

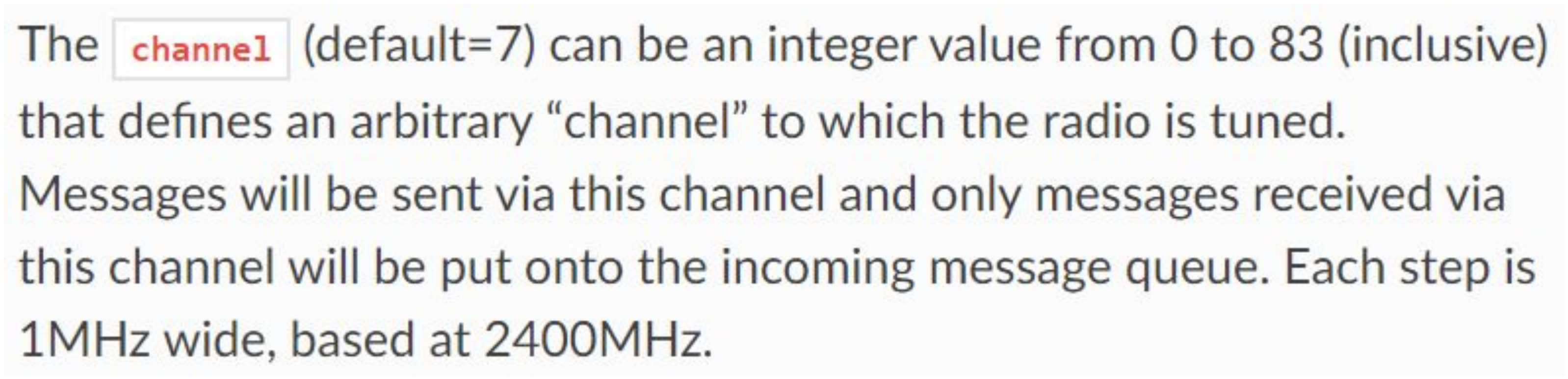
Turns the radio on. This needs to be explicitly called since the radio draws power and takes up memory that you may otherwise need.

Guided

Configure Radio Settings


`radio.config(**kwargs)`

Configures various keyword based settings relating to the radio. The available settings and their sensible default values are listed below.

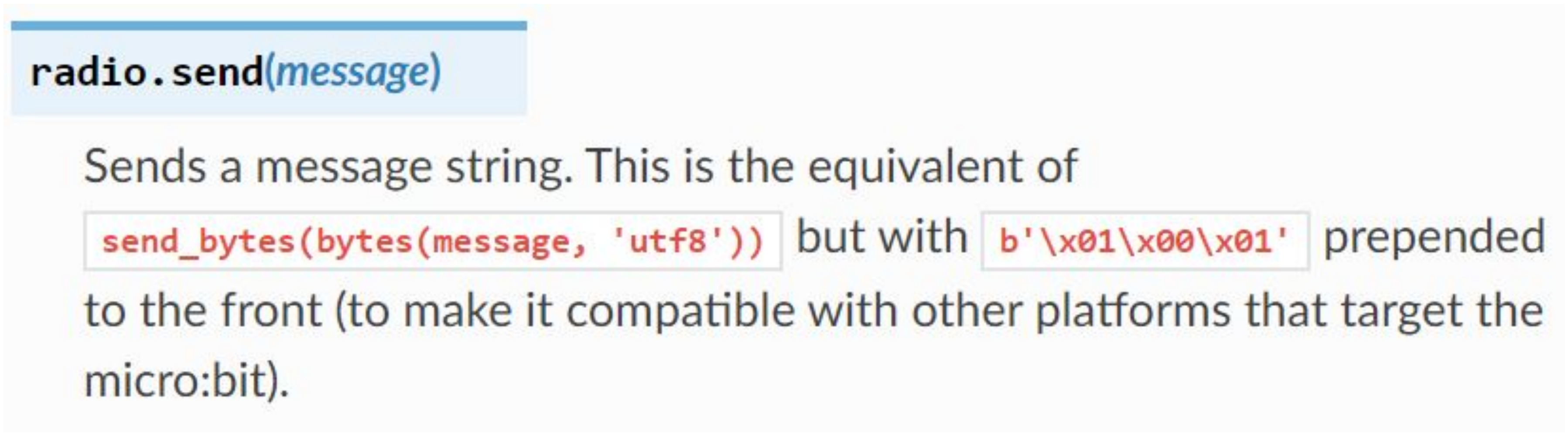

The `channel` (default=7) can be an integer value from 0 to 83 (inclusive) that defines an arbitrary “channel” to which the radio is tuned. Messages will be sent via this channel and only messages received via this channel will be put onto the incoming message queue. Each step is 1MHz wide, based at 2400MHz.

🎎 Guided

🔍 Find the Function to send a radio message

Guided

Send a Radio Message



radio.send(*message*)

Sends a message string. This is the equivalent of `send_bytes(bytes(message, 'utf8'))` but with `b'\x01\x00\x01'` prepended to the front (to make it compatible with other platforms that target the micro:bit).

<8) =) ;D :)/>
CODE IN THE COMMUNITY

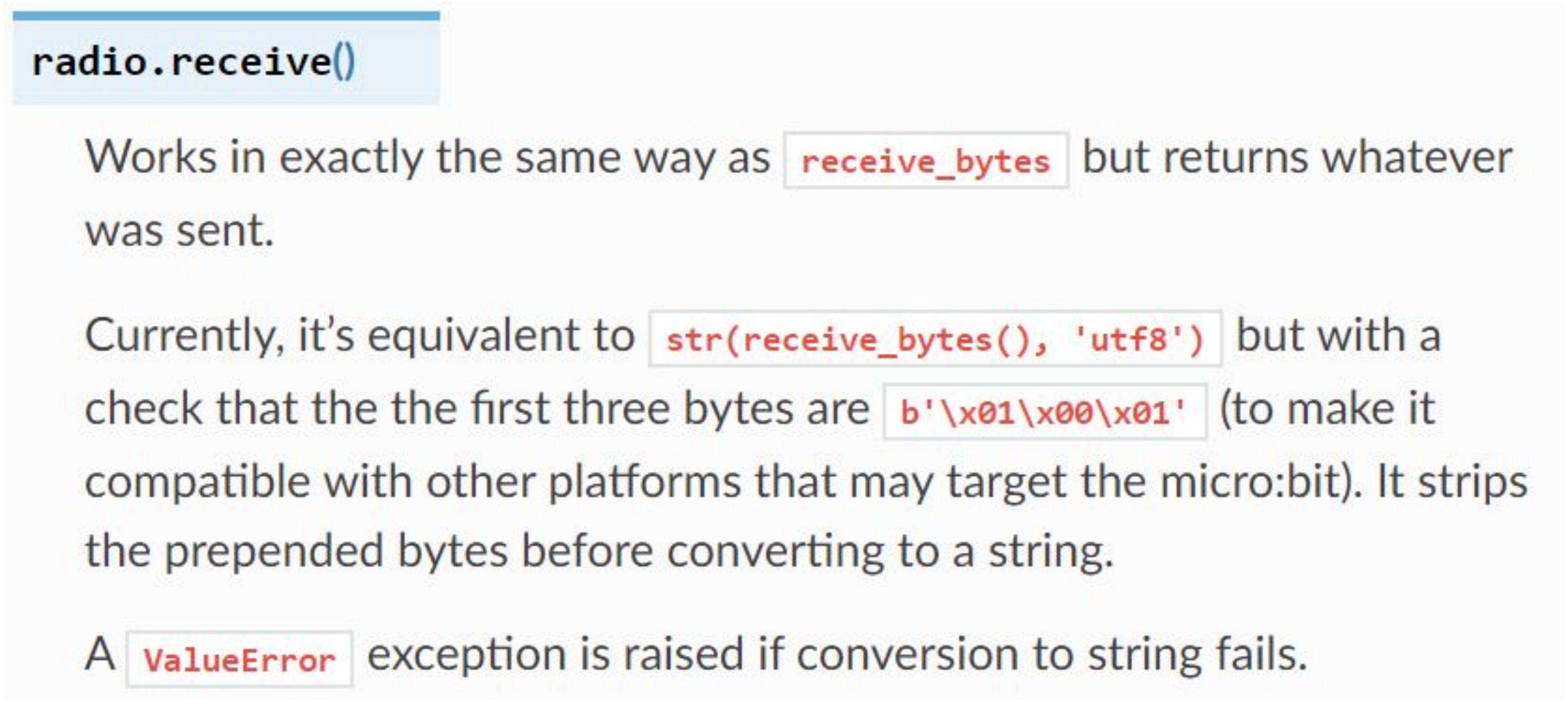
Guided

Find the Function to receive a radio message

<8) =) ;D :)/>
CODE IN THE COMMUNITY

Guided

Receive a Radio Message



`radio.receive()`

Works in exactly the same way as `receive_bytes` but returns whatever was sent.

Currently, it's equivalent to `str(receive_bytes(), 'utf8')` but with a check that the the first three bytes are `b'\x01\x00\x01'` (to make it compatible with other platforms that may target the micro:bit). It strips the prepended bytes before converting to a string.

A `ValueError` exception is raised if conversion to string fails.

📥 Download the Project onto the micro:bit

Guided

AttributeError

Line 9 AttributeError type object: MicroBitImage has no attribute: TICK

Guided

Find the image that displays a tick

Guided

How to Correct the Code

~~TICK~~

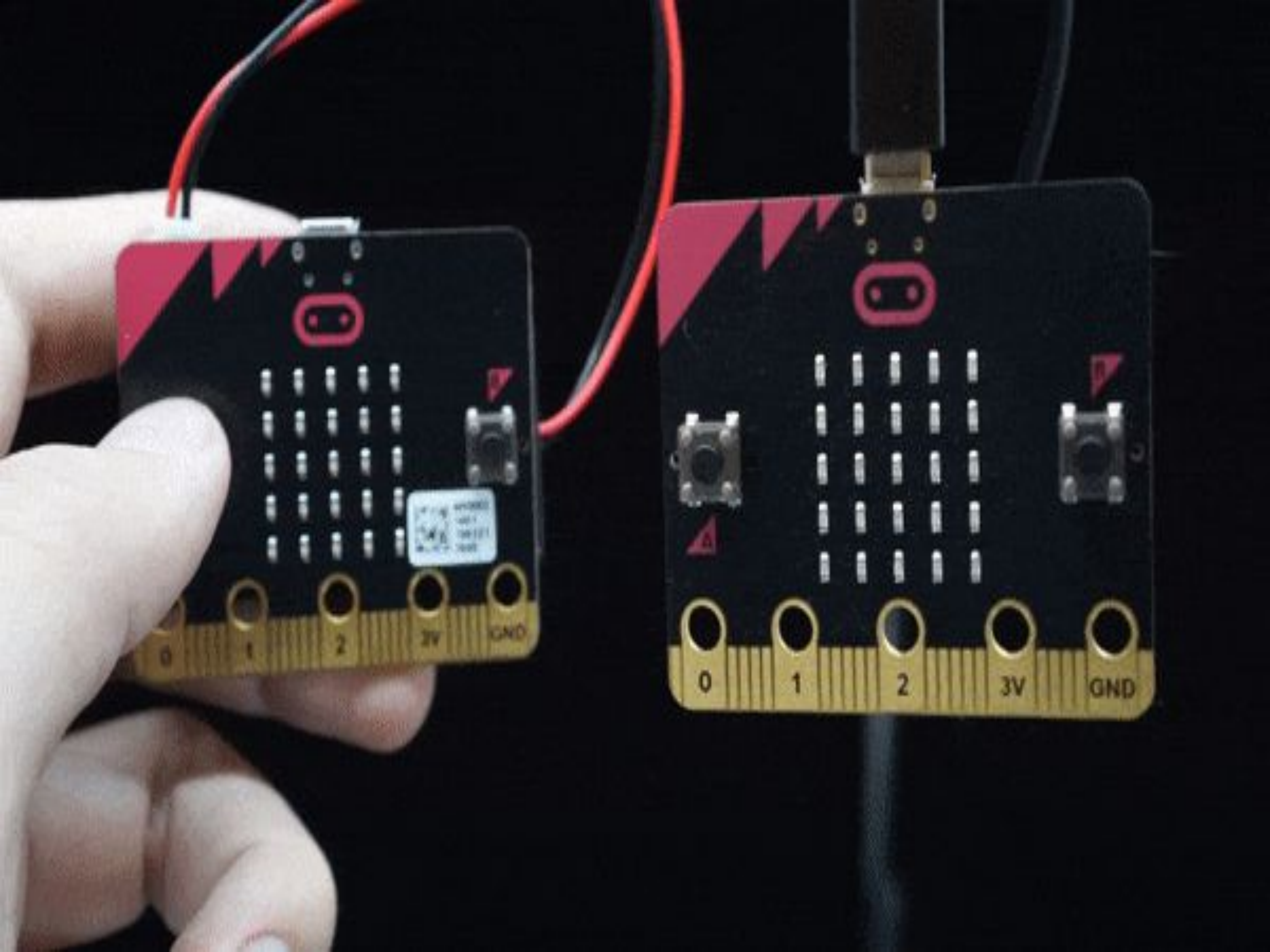
Image.**YES**

💾 Give your project a name and save it!

🙌 **Unguided**

20 min

Emoji Messaging


B
0
1
2
3V
GND
A
B
0
1
2
3V
GND

Unguided

10:00

Programme the micro:bit

When button A is pressed, send a happy face message. When button B is pressed, send a sad face message. When a message is received, display a happy/sad face.

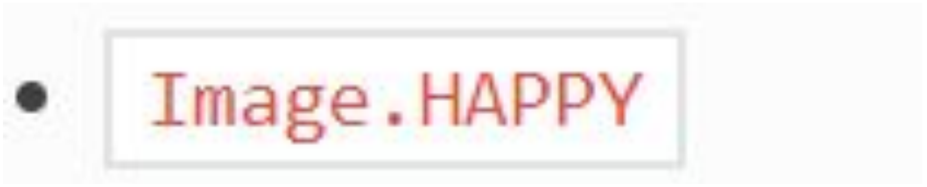
🙌 **Unguided**

~~👀~~ Hints

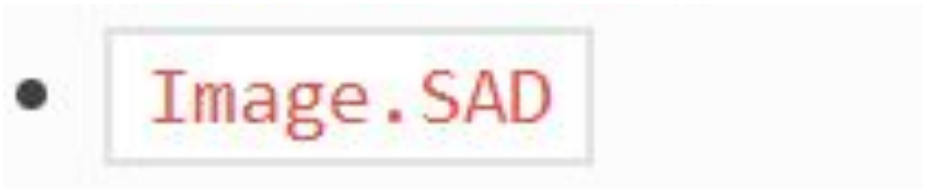
Find images in the microbit library

Guided

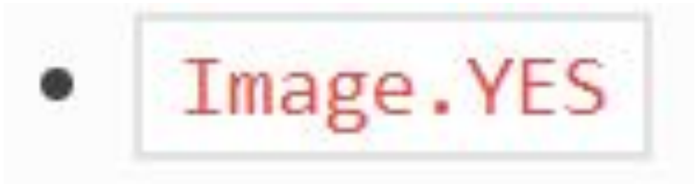
Sad, Happy, Tick Images



- `Image.HAPPY`



- `Image.SAD`



- `Image.YES`

💾 Give your project a name and save it!

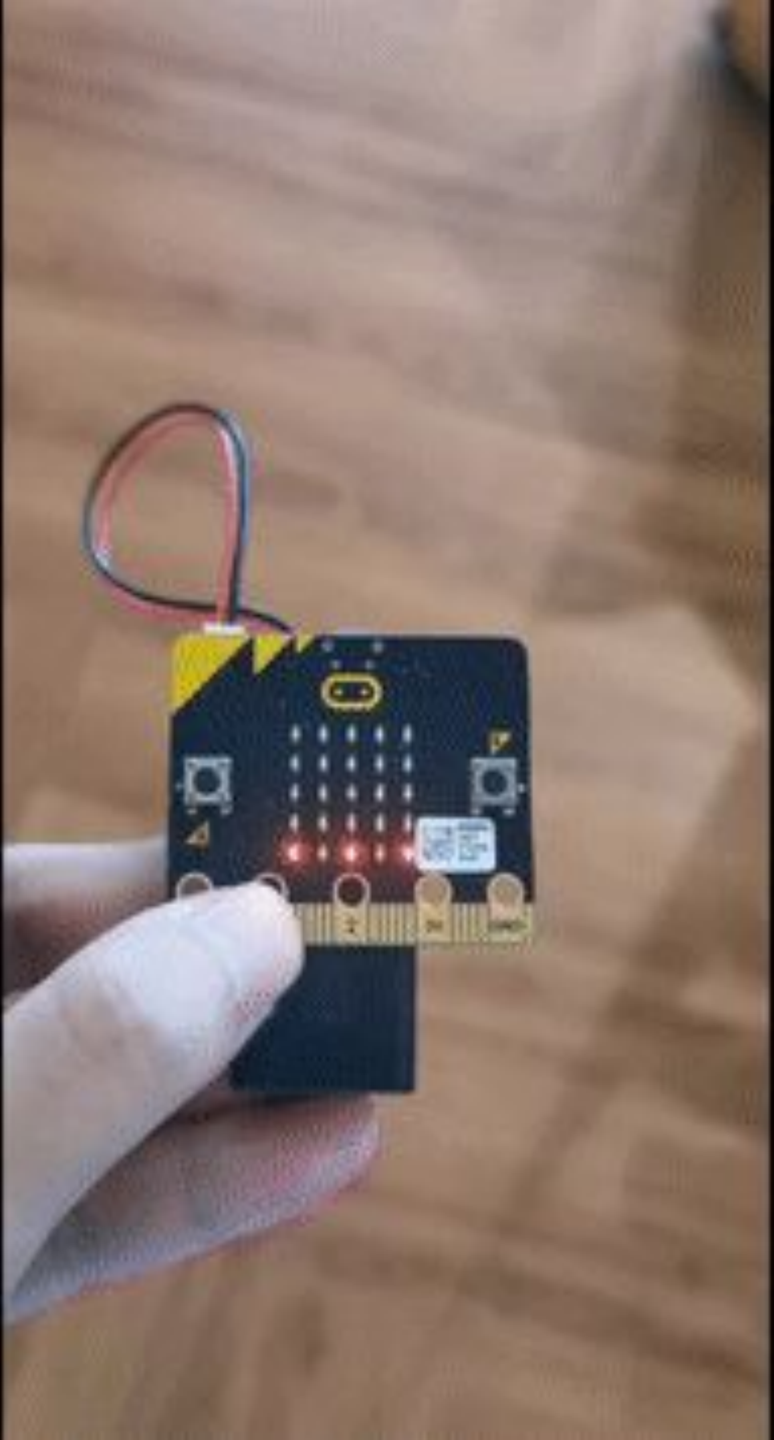
10 min

Break

Guided

30 min

Treasure Hunt



Guided

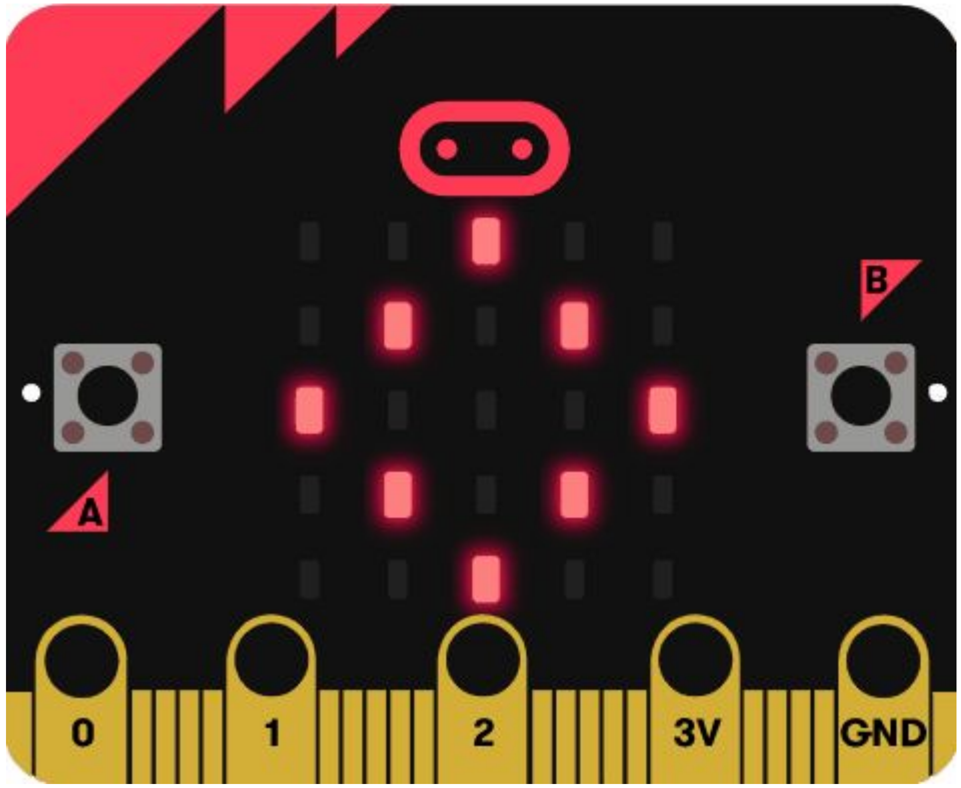
Programme the micro:bit

Hidden transmitter micro:bit (treasure) continuously transmit code.

The receiver microbits receives the code and displays the signal strength received. The closer the player is to the treasure, the more LEDs light up.

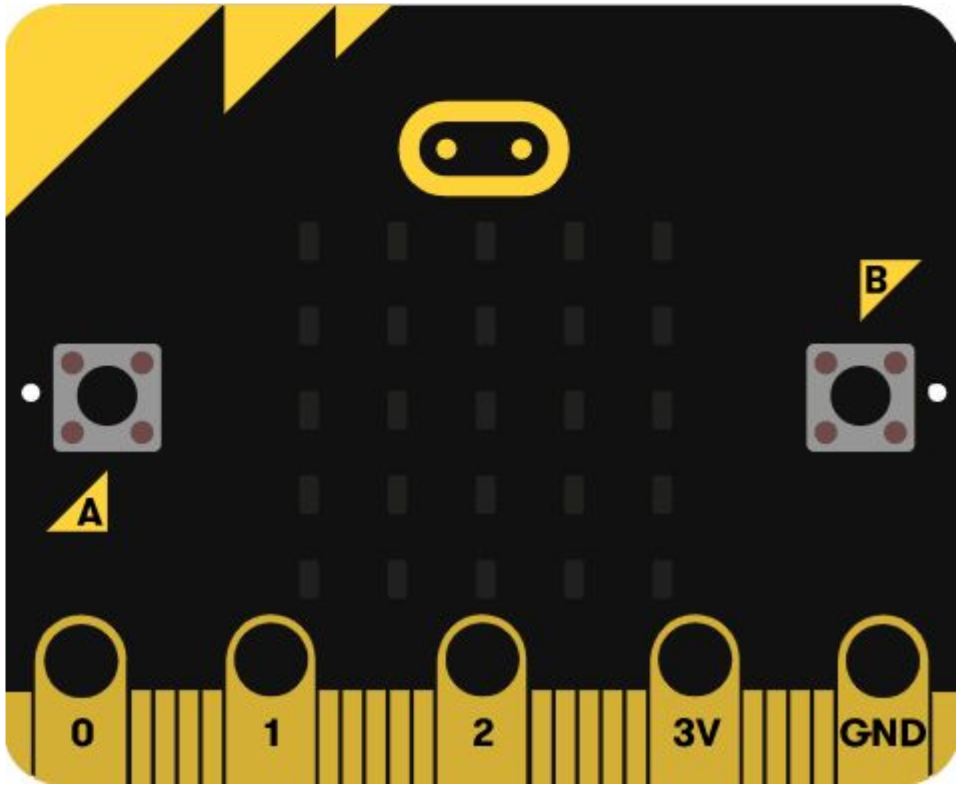
Guided

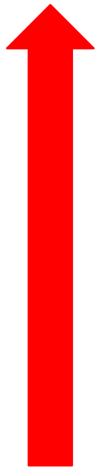
Transmitter



Guided

Receiver





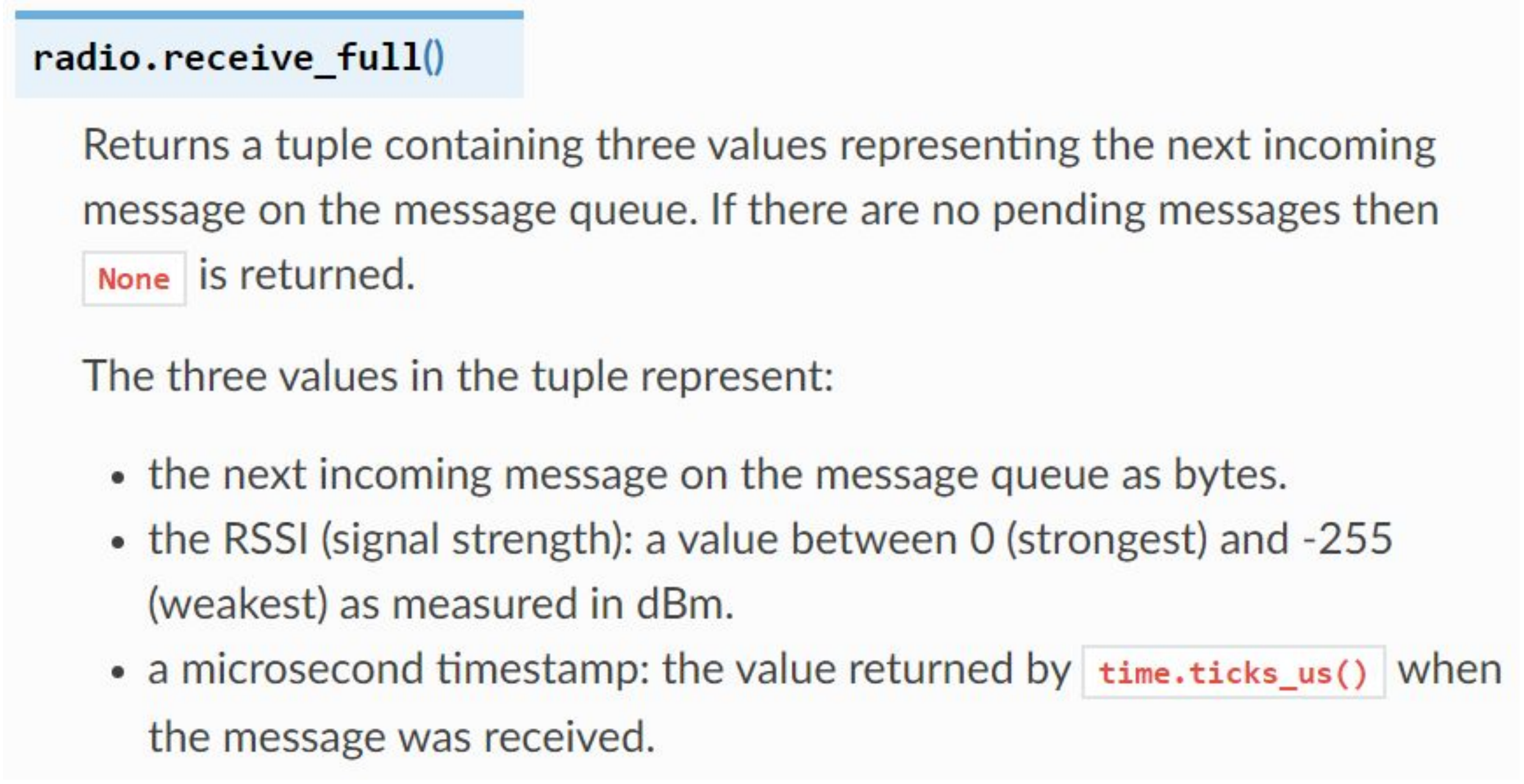
The nearer to the transmitter, the more LEDs light up

🎎 Guided

🔍 Find the Function that receives a message and returns a tuple

Guided

Receive a Tuple



`radio.receive_full()`

Returns a tuple containing three values representing the next incoming message on the message queue. If there are no pending messages then `None` is returned.

The three values in the tuple represent:

- the next incoming message on the message queue as bytes.
- the RSSI (signal strength): a value between 0 (strongest) and -255 (weakest) as measured in dBm.
- a microsecond timestamp: the value returned by `time.ticks_us()` when the message was received.

Guided

`radio.receive_full()` returns **(message, signal strength, timestamp)**

🎎 Guided

Tuple

(message, signal strength, timestamp)

🎎 Guided

Tuple

(**message**, signal strength, timestamp)

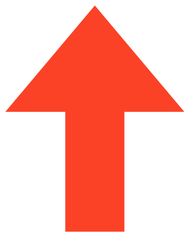
⬆

Index **[0]**

🎎 Guided

Tuple

(message, **signal strength**, timestamp)



Index **[1]**

<8) =) ;D :)/>
CODE IN THE COMMUNITY

Guided

Tuple

(message, signal strength, **timestamp**)

Index **[2]**

<8) =) ;D :)/>
CODE IN THE COMMUNITY

Guided

```
if msg is not None
```

Guided

```
if msg is not None
```

Means nothing or null value

Guided

```
if msg is not None
```

Checks if 2 things are not the same

Guided

```
if msg is not None
```

If no message is received, the condition is False.

Guided

```
if msg is not None
```

↓

If message is received, then ???

Guided

Which value of the tuple is the signal strength?

🎎 Guided

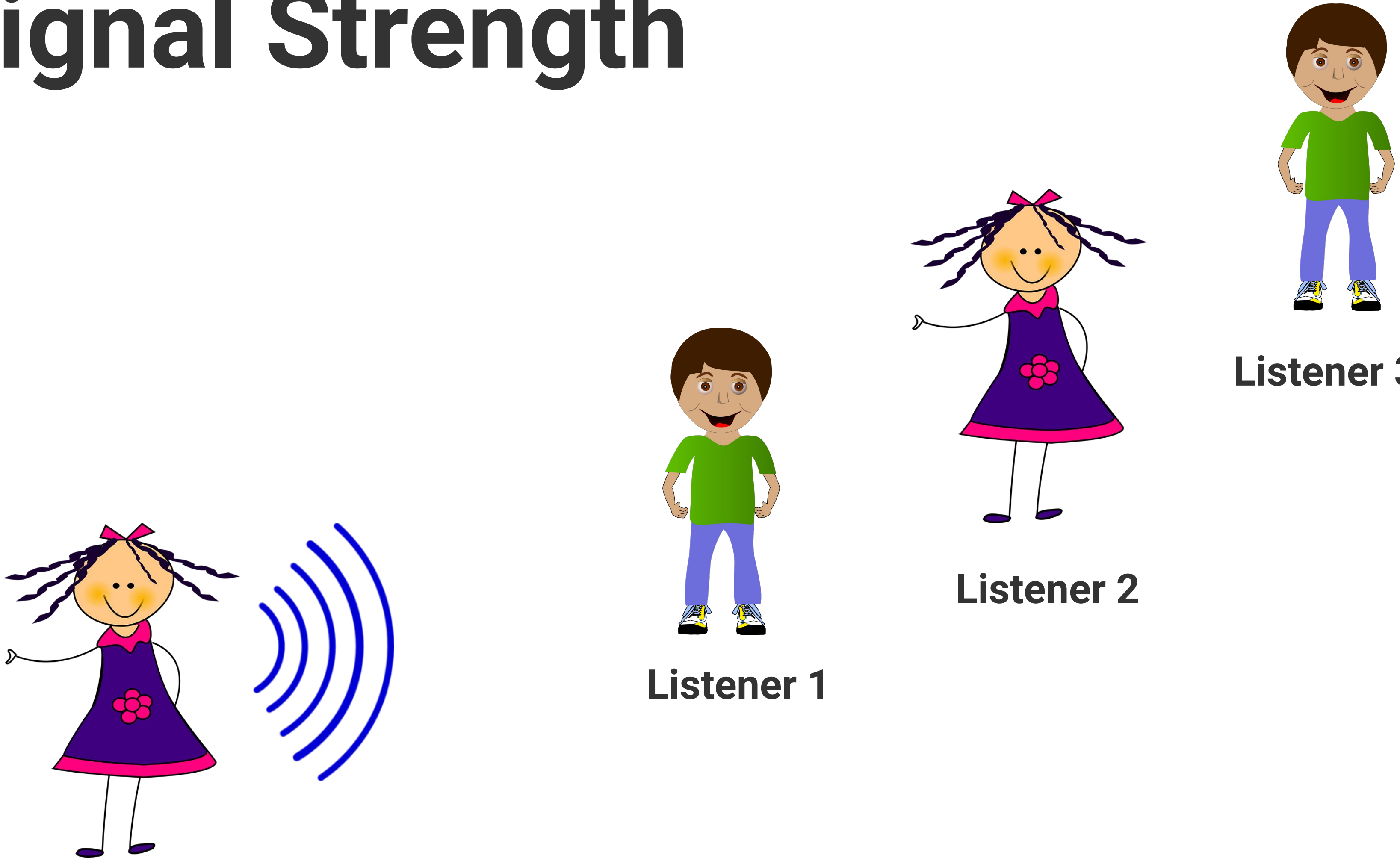
Tuple

(message, **signal strength**, timestamp)

Index **[1]**

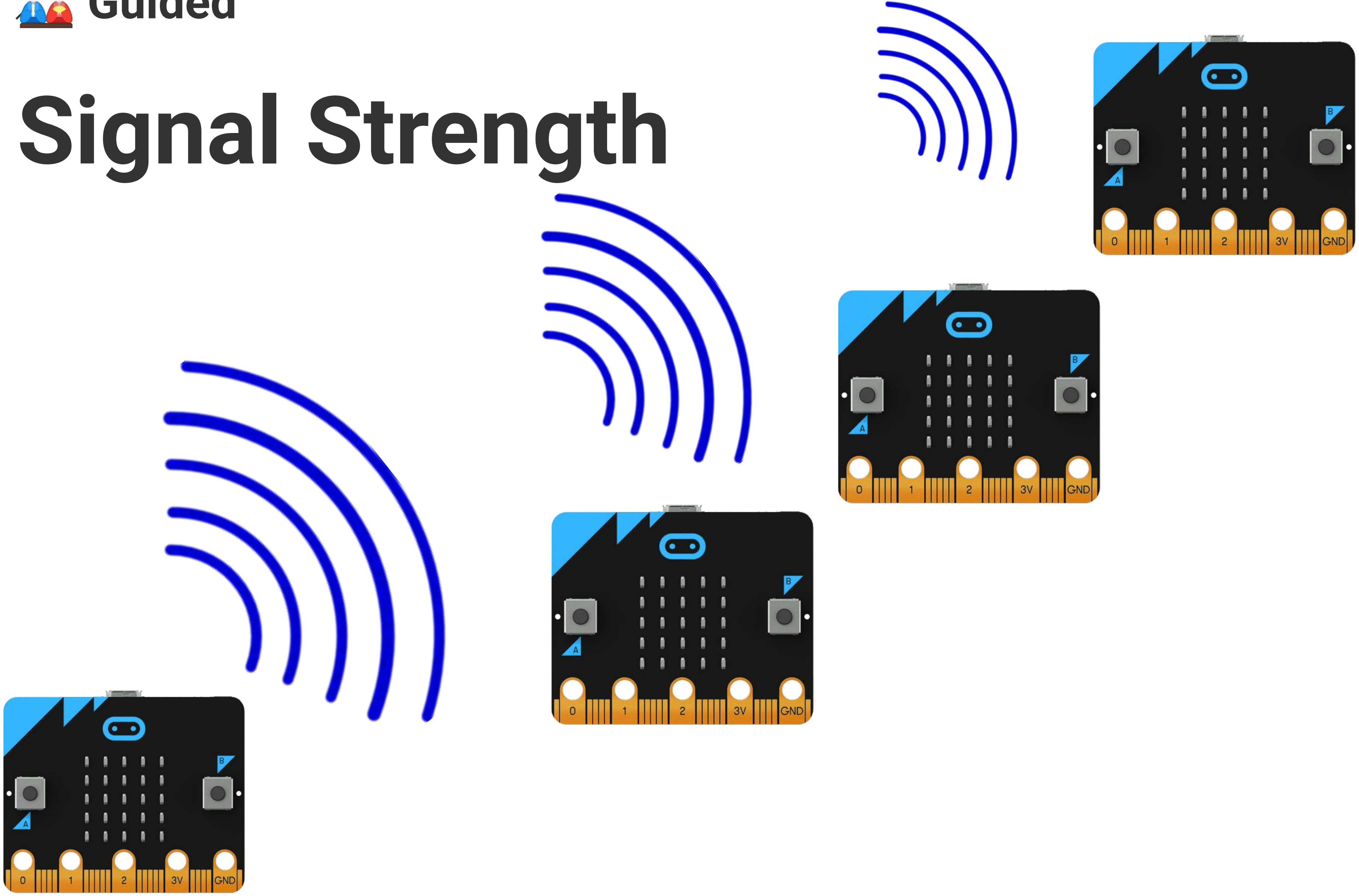
Guided

Signal Strength



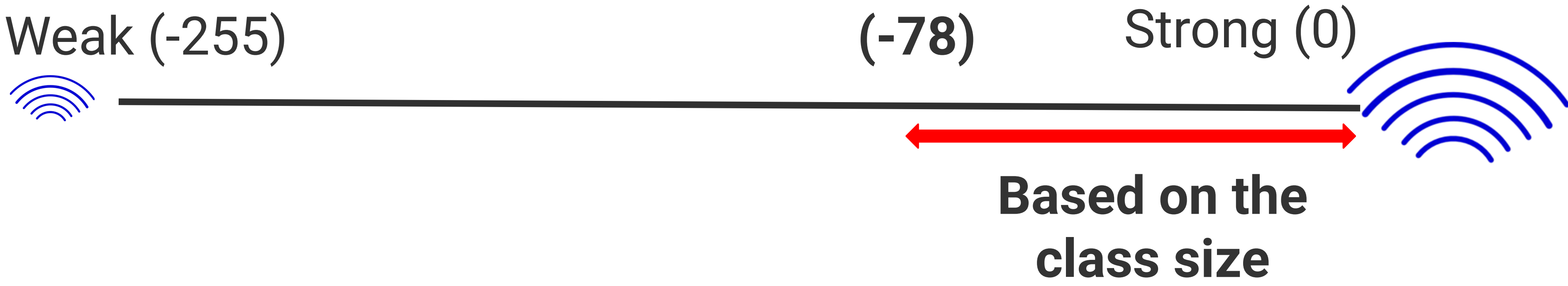
🎎 Guided

Signal Strength



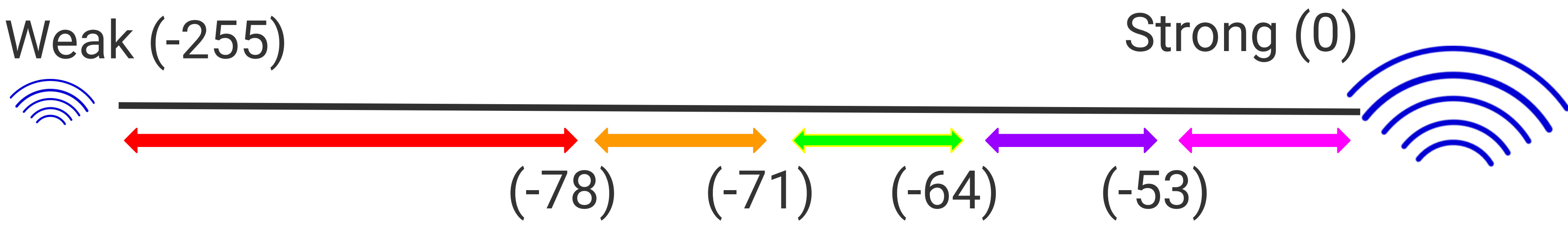
Guided

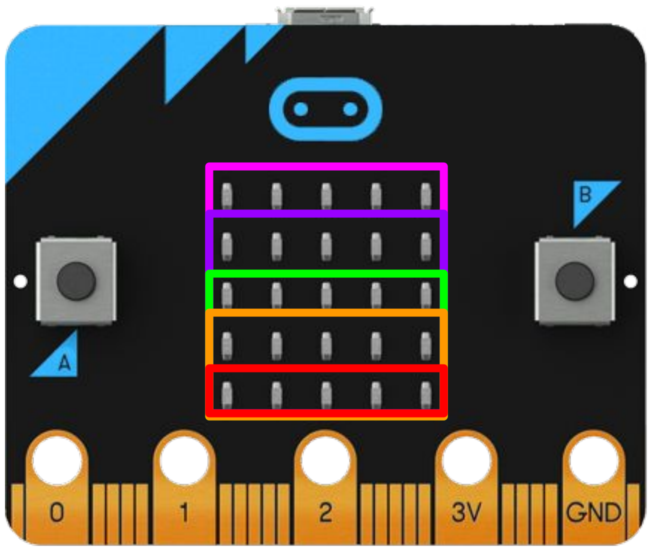
Signal Strength



🎎 Guided

Signal Strength



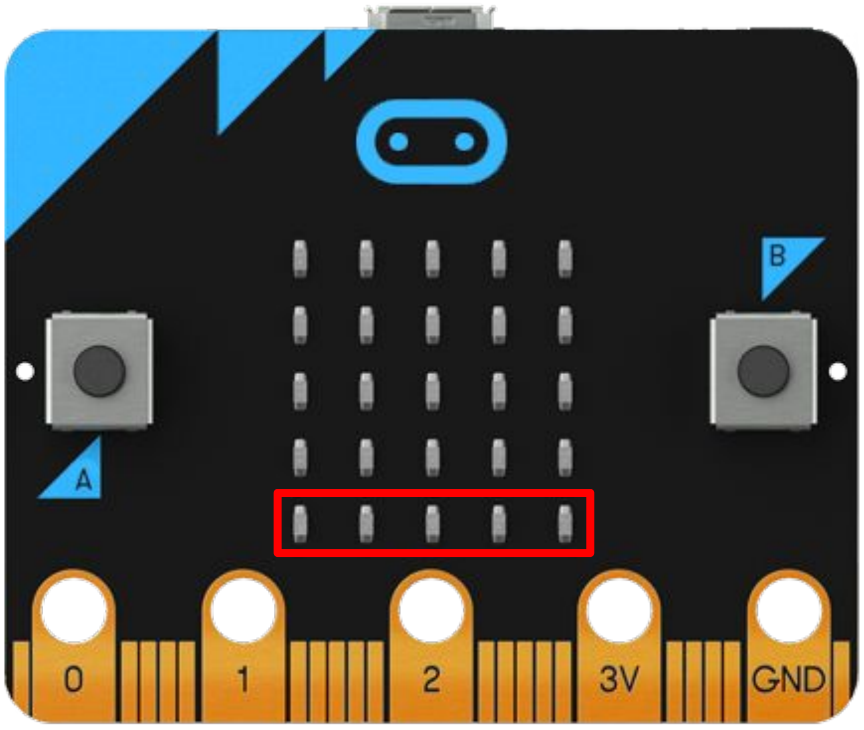


<8) =) ;D :)/>
CODE IN THE COMMUNITY

Guided

Signal Strength

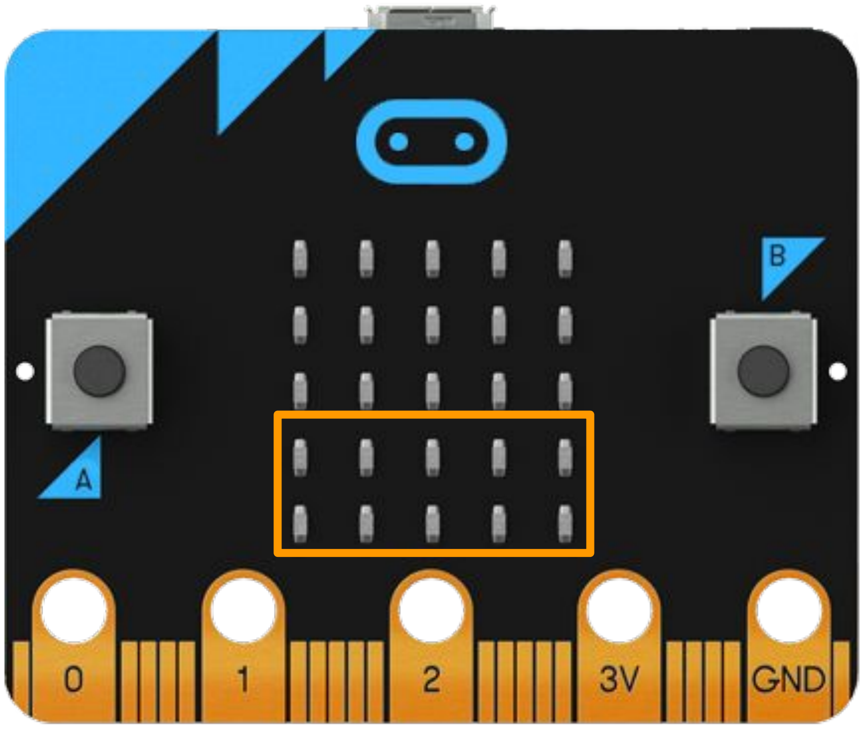
if strength is ≤-78:



Guided

Signal Strength

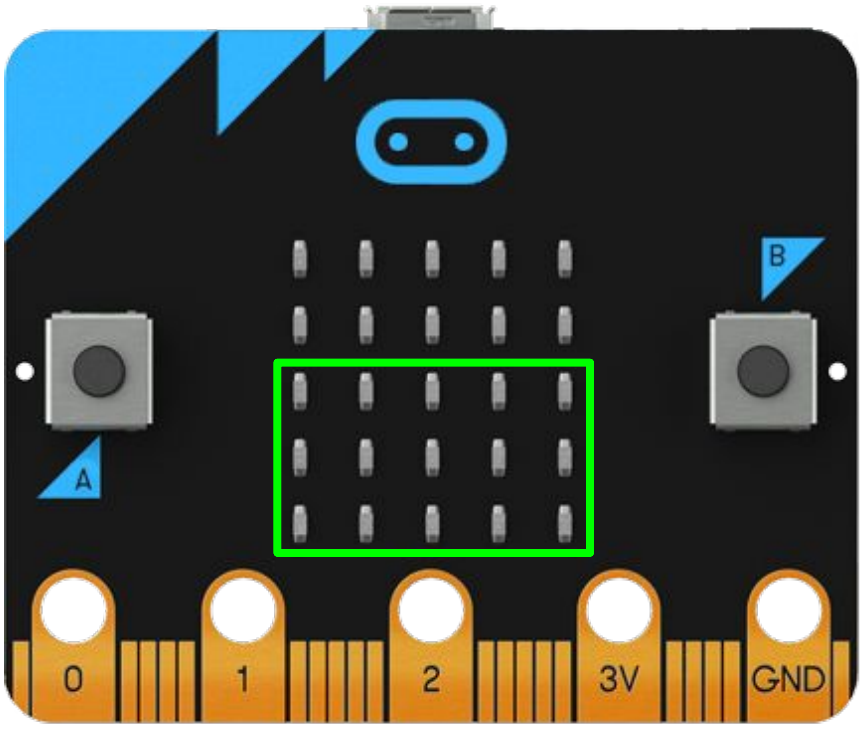
elif strength is
between -77 to -71:



Guided

Signal Strength

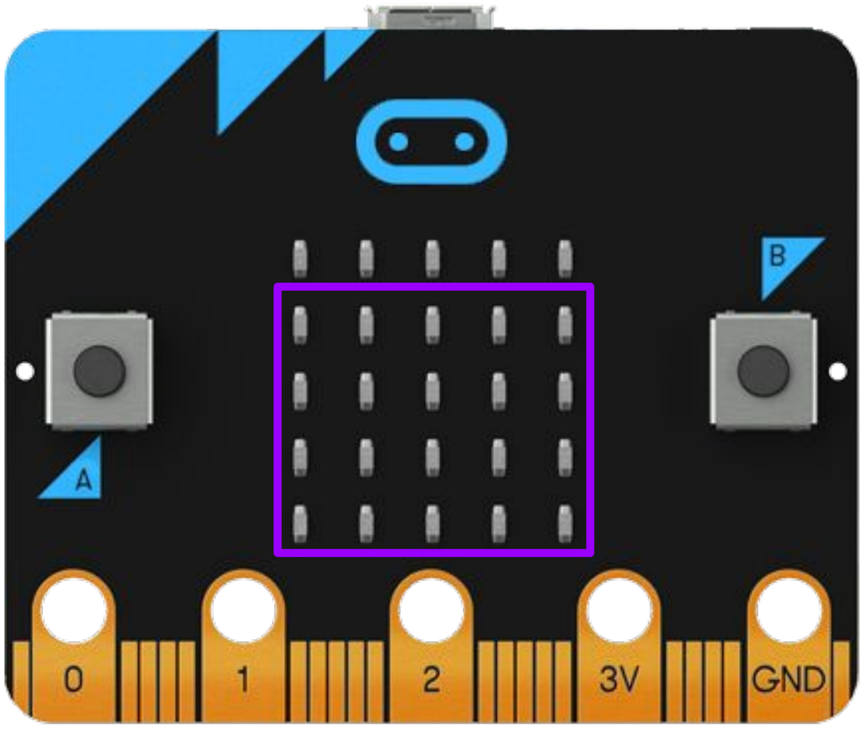
elif strength is
between -70 to -64:



Guided

Signal Strength

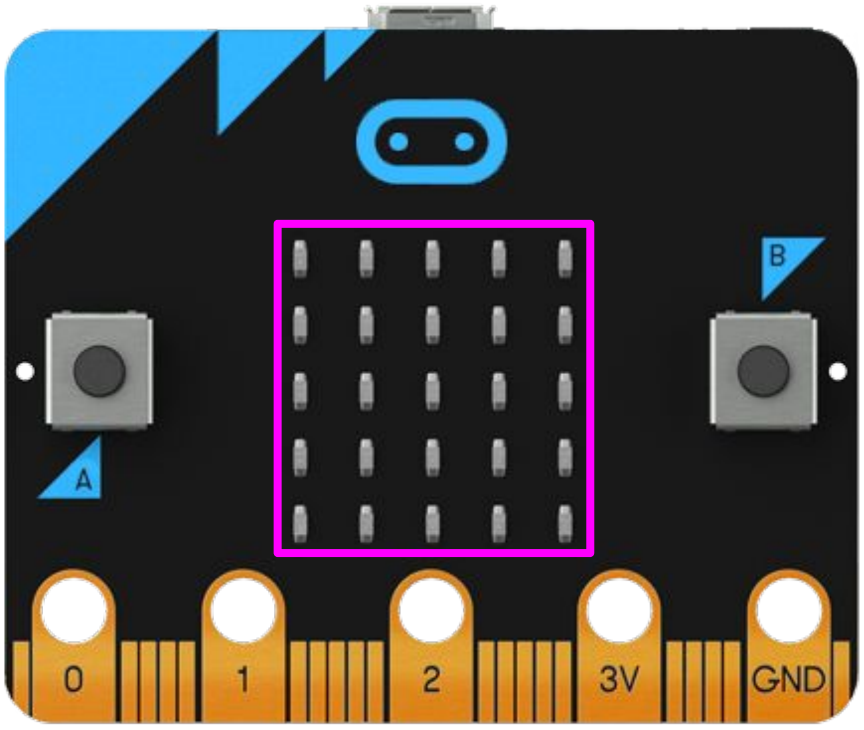
elif strength is
between -63 to -53:



🎎 Guided

Signal Strength

else:

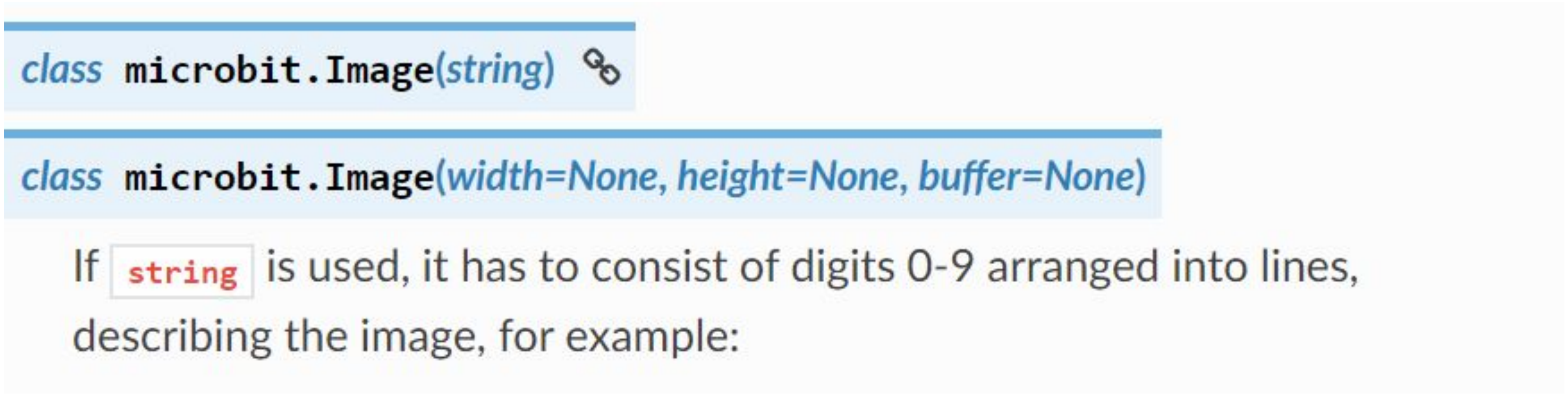


Guided

Find the function to make specific LEDs light up

Guided

Lighting specific LEDs



class **microbit.Image**(*string*)

class **microbit.Image**(*width=None, height=None, buffer=None*)

If `string` is used, it has to consist of digits 0-9 arranged into lines, describing the image, for example:

```
Image("00000:00000:00000:00000:99999")
```

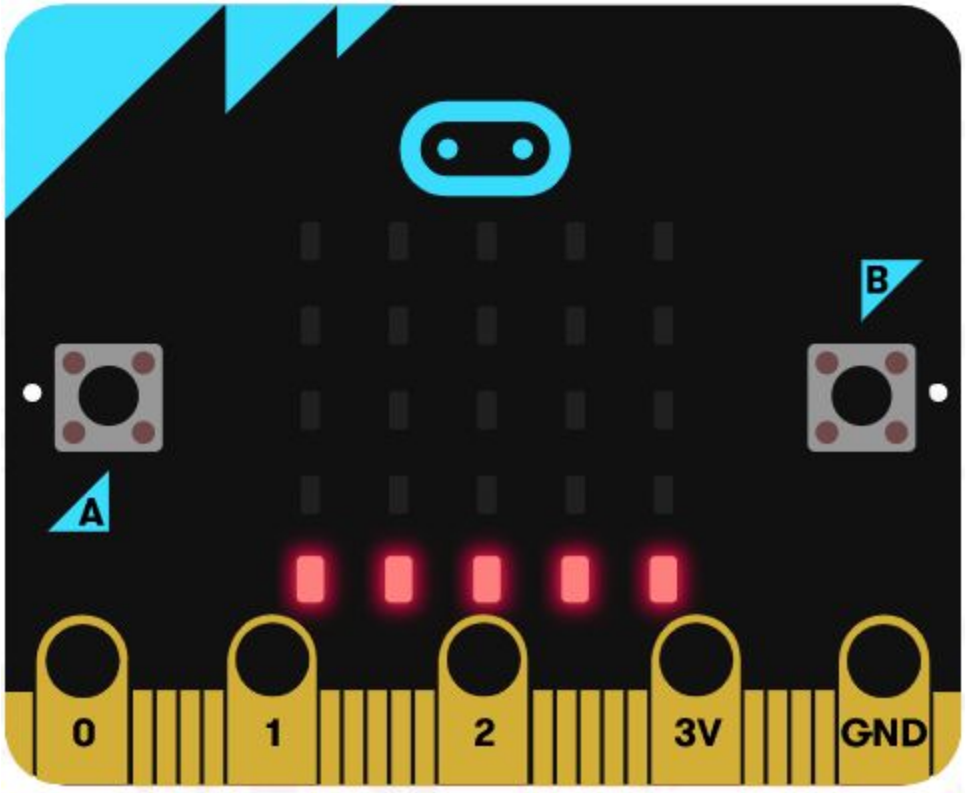
CODE IN THE COMMUNITY

🎎 Guided

```
Image("00000:00000:00000:00000:99999")
```

Row 5 Row 4 Row 3 Row 2 Row 1

Row 5
Row 4
Row 3
Row 2
Row 1

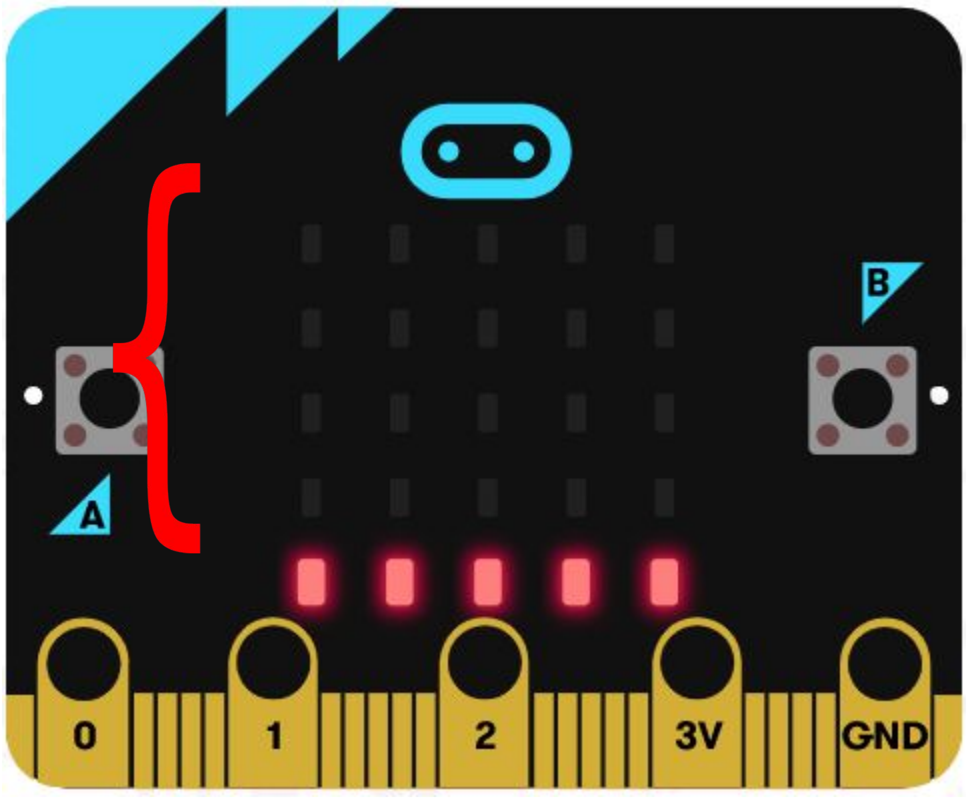


Guided

```
Image("00000:00000:00000:00000:99999")
```

$\underbrace{\hphantom{00000:00000:00000:00000:99999}}$

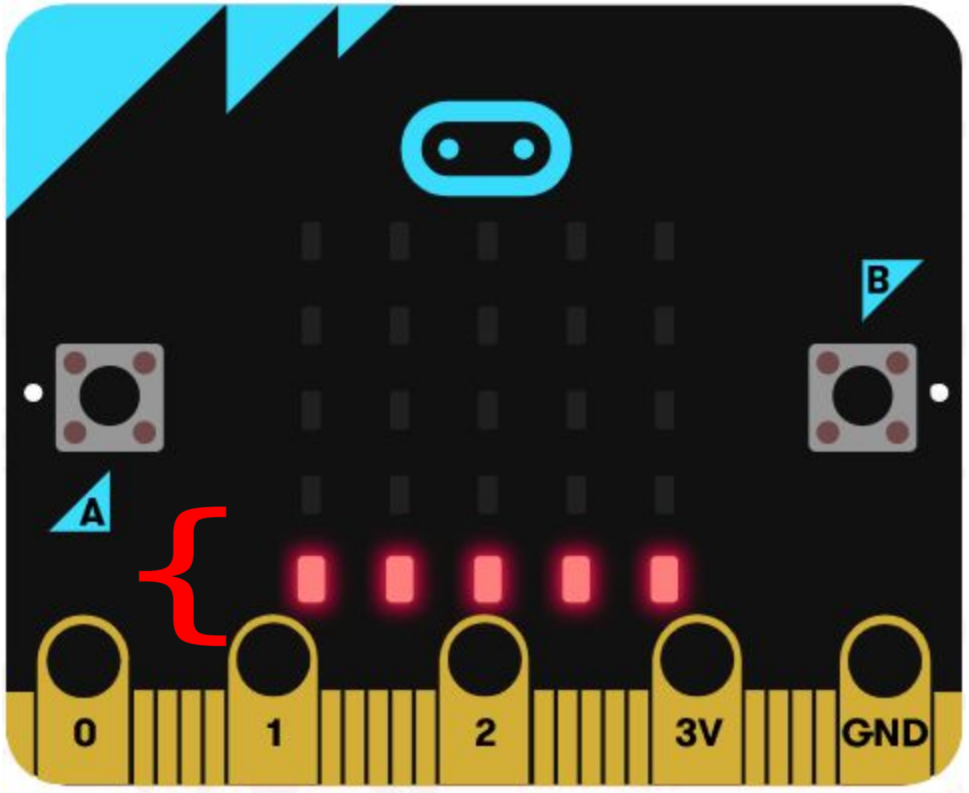
0: LEDS not lighted up



Guided

$$\texttt{Image("00000:00000:00000:00000:}\underbrace{\texttt{99999}}_{}\texttt{")}$$

9: Lighted LEDs



Guided

Which row of LEDs light up?

```
Image("00000:00000:99999:00000:00000")
```

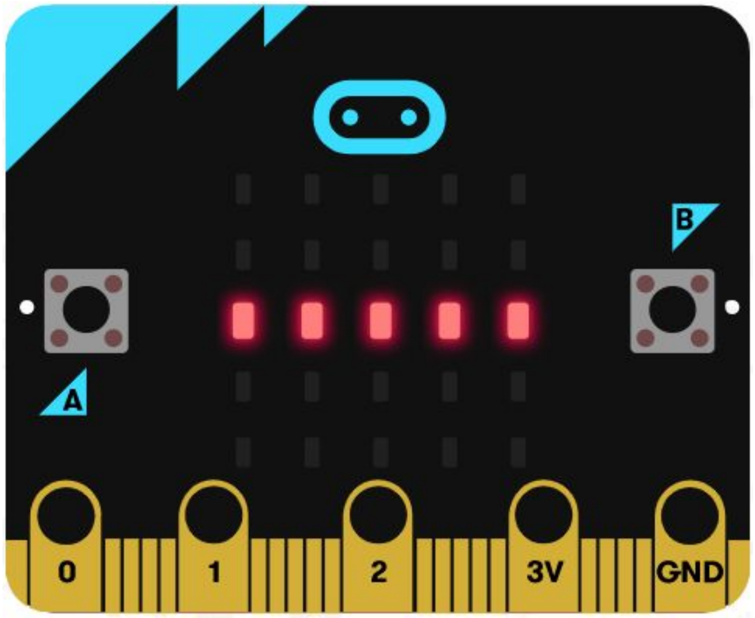
CODE IN THE COMMUNITY

Guided

Which row of LEDs light up?

```
Image("00000:00000:99999:00000:00000")
```

3rd Row!



📥 Download the Project onto the micro:bit

💾 Give your project a name and save it!

🎁 Bonus

Playing the Treasure Hunt Game