BBC Micro: bits – Radio Communication

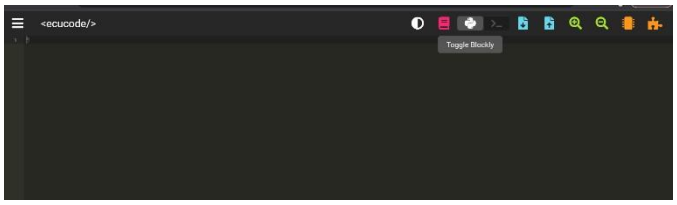
ECU Makerspace Group

Written by Julia Szymanski

Getting started

Let's do a practice run just using the Micro: bit LED's. Once you have access to the computers, follow through this guide. If your computer is not on the makerspace website then use this link to gain access. **Please use Google Chrome to access this website (makerspace.ecu.edu.au/code)**

Click on **'Toggle Blockly'** to switch to Python code.



Task 1 – Print your name!:

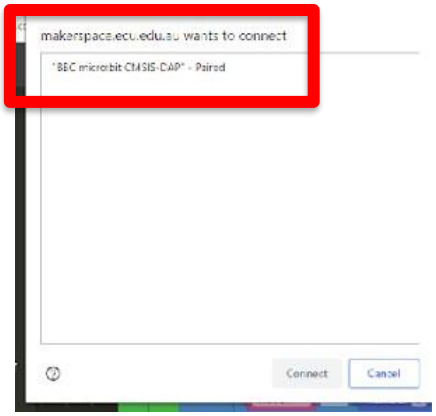
Step One – Print your name:

Click on to the first line and import 'display' from 'microbit', then display a message using display.scroll(" ").

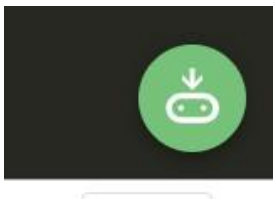
```
1 from microbit import display
2
3
4 display.scroll('Hello, World!')
5
```

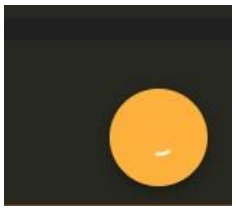
Step Two - Now that you have the scroll message on the page, write your name and upload it to the Micro: Bit. Do this by following these instructions:

Click on the red button then a box comes up inviting you to pair the micro: bit. Click on the "BBC Micro: bit CMSIS-DAP"



Once this has been completed click on the green button to download. An orange button will appear as it is downloading.





Congratulations! Remember to delete this scroll message before starting the next section.

Troubleshooting:

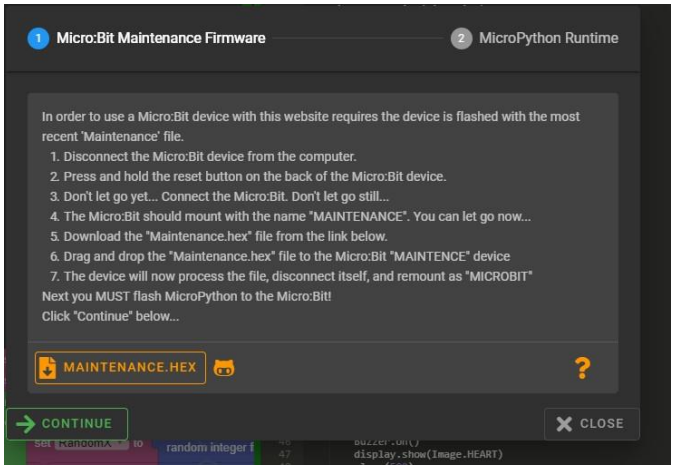
Updating the firmware on the Micro: bit

Click on the orange shape which appears 'update micro: bit'. Follow set instructions on the **Maintenance Firmware** guide.

BBC Micro: bits – Radio Communication

ECU Makerspace Group

Written by Julia Szymanski



If the micro: bit appears not to be responding. Unplug the micro: bit, press the reset button at the back of the micro: bit and repeat this process. If the micro: bit continues to fail, swap it for another one!

Now let's learn how to write a program that enables our Microbits to communicate with each other!

Part 1

First, we need to import a library for the display on the Microbit, so the Microbit can display a text. We also need to import a library for the buttons, so we can use them.

Our Microbits will use radiocommunication to talk to each other, so we need to import the radio library as well.

```
from microbit import display, button_a, button_b
import radio
```

Before the Microbits can communicate, it is important that we turn on the radio, otherwise there will be silence.

```
# Turn on the radio
radio.on()
```

We want the Microbit to send a message with "Hi!", whenever button A is pressed. First, we need to check whether button A is pressed. As we want the program to keep running and not only execute once, we will use a While True loop. This will check whether Button A has been pressed and will keep running continuously.

If button A is pressed, the microbit will send a radiosignal with the message "Hi".

```
while True:
    # Check if button A is pre
    if button_a.is_pressed():
        radio.send("Hi!")
```

So that we can be sure that the message was sent, we will also get the Microbit to show us a confirmation that the message was sent by displaying "Sent!".

```
        display.scroll("Sent!")
```

BBC Micro: bits – Radio Communication

ECU Makerspace Group

Written by Julia Szymanski

This is our code so far:

```
from microbit import display, button_a, button_b
import radio

#Turn on the radio
radio.on()


while True:
    #Check if button A is pressed to send a message
    if button_a.is_pressed():
        radio.send("Hi!")
        display.scroll("Sent!")
```

We want our Microbit to be both a sender and receiver, so it can receive a radio signal and can also receive a signal.

To achieve this, we need to create a variable that holds any messages that our Microbit receives.

```
    #Check for incoming messages
    message = radio.receive()
```

If a message has been received, we then want the Microbit to show the received message. For this, we use an if statement to check whether a message has been received – if so, the Microbit displays the message.

```
    if message:
        display.scroll(message)
```

This is what our code looks like now:

```
from microbit import display, button_a, button_b
import radio

#Turn on the radio
radio.on()


while True:
    #Check if button A is pressed to send a message
    if button_a.is_pressed():
        radio.send("Hi!")
        display.scroll("Sent!")

    #Check for incoming messages
    message = radio.receive()
    if message:
        display.scroll(message)
```

Well done!

You can download the code onto two Microbits and send messages around.

Challenge:
Can you change up the text message that is send on one of the Microbits?

Part 2

At the moment, our Microbits send radio signals (messages) around to any other Microbits around – it is broadcasting a message.

The problem is, the Microbit also receives messages from any other Microbits around, which we don't want We only want our two Microbits to communicate with each other and we don't want anyone else to access our Network unless we allow them to.

We can protect our network by setting up radio communication channels. This

means that only Microbits within the same group and communication channel will be able to talk to each other.

For this, we need to set up a group and a communication channel **after we switch on the radio.**

```
# Configure the radio with a unique group and channel
radio.config(group=1, channel=7) # group and channel must be the same on both micro:bits
```

We can change the channel to any that we want, we just need to make sure the same channel is set up on both Microbits.

You might want to change the communication channel to a different number, as there are other people setting up theirs, too. By choosing a higher number, you can avoid accidentally assigning the same channel as someone else.

Good job! No one else is now able to access your network unless you give them the communication channel number.