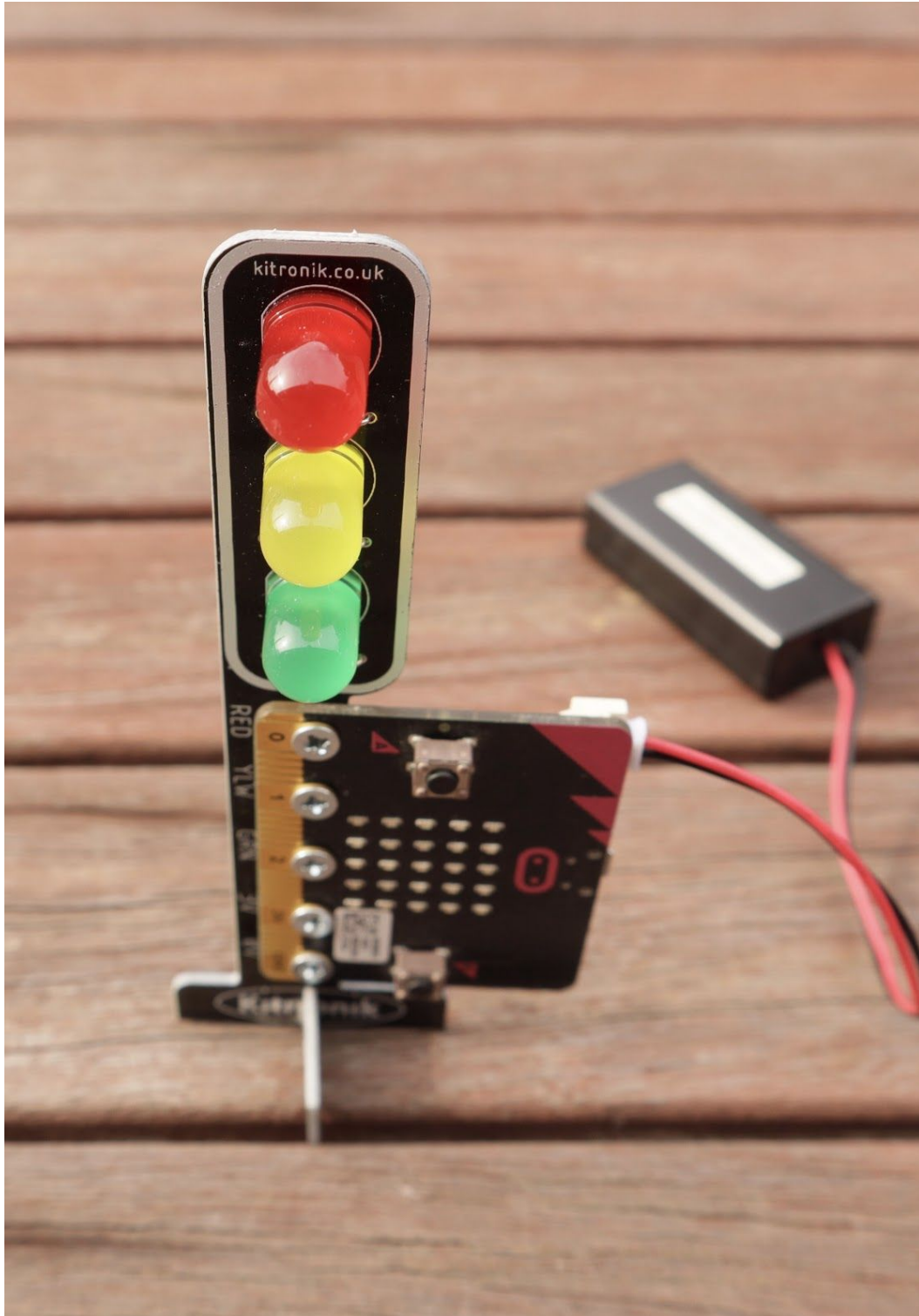


Project 1: Single Traffic Light





In this project, you will learn to:

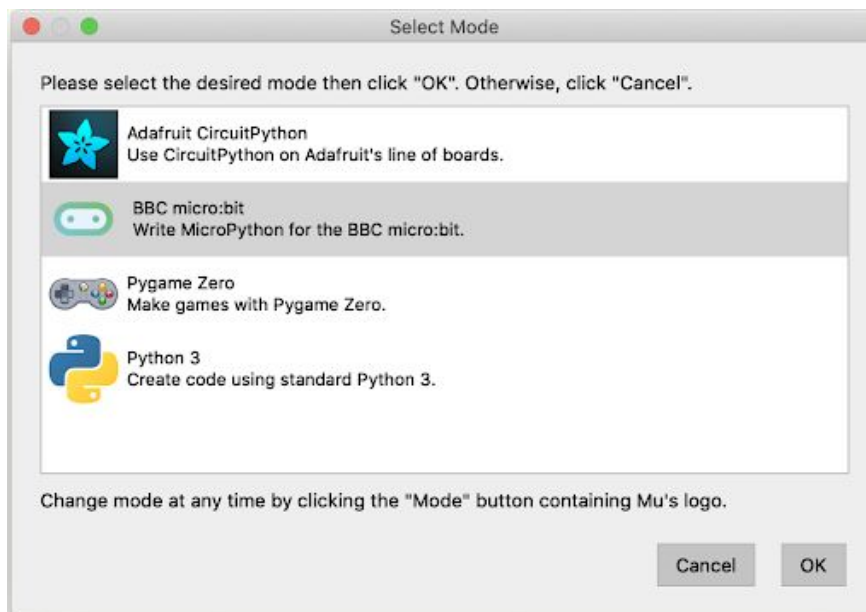
- Write a function to turn on individual Traffic Light LEDs
- Call the function in your main loop

You will need:

- One Kitronik STOP:bit, assembled with a micro:bit
- Micro USB cable for programming the micro:bit
- Battery pack (2x AAA batteries) for the micro:bit
- Mu editor¹ installed on your laptop

Step 1: Launch mu editor, and enable micro:bit mode


- Go to **Applications** in mac, or the **Programs** menu in Windows, and launch **mu editor** by clicking the icon that looks like this: 
- Once the editor is running, click on the Mode button 
- Click on **BBC micro:bit** so it is highlighted, then click **OK**.




¹ <https://codewith.mu/>

Step 2: Write your code

Each traffic light is made up of three coloured Light Emitting Diodes (LEDs). Each LED has two wires; one connects to ground, and the other to pin 0, 1, or 2, on the micro:bit.

- ❑ Type the code below into **mu editor**, then connect your micro:bit USB cable, and click the **Flash**  button to download the code onto the micro:bit.

```
from microbit import *  
  
pin2.write_digital(1)
```

- ❑ If all went well, the green LED connected to pin2 should be lit! Try changing pin2 to pin1 and the amber LED connected to pin1 should now be lit.
- ❑ Instead of a `.write_digital(1)` to pin2, write 0 to the pin, and it should turn the LED off.
- ❑ Now click the **save**  button on **mu editor**, and save a copy of your code as “stopbit.py” in your `mu_code` directory.

What our code does:

- The LEDs each have one wire connected to ground, and one connected to a pin: 0, 1, or 2. We switch the power from low(0) to high(1) with our `digital_write(1)`
- By sending to a different pin, we turn on a different LED.
- Writing zero(0) to the pin will set it to low(0), and turn off the LED.

Step 3: Write a function to turn one LED on, and the others off

We can reduce the amount of code we need to write, by creating a [function in Python](https://www.w3schools.com/python/python_functions.asp)². We call the function by typing its name, and customise what it does by giving it information in a parameter, inside the brackets `()`.

- ❑ First, delete the line, `pin2.write_digital(1)` from your code in **mu editor** (leave the import line there). Now that we can turn LEDs on and off, let's make a function to turn on an individual LED, and turn the others off.

² https://www.w3schools.com/python/python_functions.asp

- ❑ To start defining our `stopBit()` function, we add the following line, and add a parameter called `colour`. The parameter will tell our function which colour LED to turn on, when we later call our function.

```
def stopBit(colour):
```

- ❑ To put code in our function, we'll **indent**³ it four spaces (**mu editor** will do this for us automatically). We indent it again to put in inside an **if** statement. So now our code should look like this (**Note:** the new lines you need to add are always in **bold**):

```
from microbit import *

def stopBit(colour):
    if colour == "green":
        pin0.write_digital(0)
        pin1.write_digital(0)
        pin2.write_digital(1)
```

- ❑ Now that our function is defined, we can call it, and tell it what colour we want lit. Add the next line under your function, and make sure it isn't indented, so it's **outside** of the function definition.

```
stopBit("green")
```

Now save your updated code in your `mu_code` directory again, then click on the **flash** button to send it to your connected micro:bit. Only the green LED should turn on.

- ❑ We still need to add two more colours to our `if` statement to let us turn on the amber and green LEDs - can you extend your code with `elif` to add **amber** and **red** colours to your function?
- ❑ Once your function can handle "green", "amber", and "red" colours, make them loop forever in a `while` loop. In your loop, make sure tell the traffic lights to wait for some time between changes!
- ❑ Test your code by **flashing** it to the micro:bit. It should cycle through green, amber, then red, waiting about 5000 milliseconds between each. **Hint:** use `sleep(5000)`.

³ "Indent" means to move text four spaces to the right, tab will work as well in mu (although there's huge arguments between using tabs vs. space)

What our code does:

- Defines a function that accepts a `colour` parameter, which can accept 3 different **arguments**; “green”, “amber”, and “red”.
- An endless loop calls our function to change the traffic light between 3 different **states**, shown in the **state table** below.

State	Red LED	Amber LED	Green LED
green	off	off	on
amber	off	on	off
red	on	off	off

Congratulations! You’ve successfully programmed a single Australian traffic light!



Project 2: Two Traffic Lights



In this project, you will learn:

- Write a state table to represent the different changes between two traffic lights.
- How to reuse our function and save time.
- Send our light changes to another STOP:bit over radio, and write code to match our truth table.

You will need:

- To have completed *Project1: Single Traffic Light* first.
- 2 x STOP:bits and micro:bits with USB cables.
- 2 x battery packs for our micro:bits.
- Mu editor installed on your laptop.

Step 1: Extend our truth table to use 2 traffic lights

Here is our previous state table that shows 3 states with 1 traffic light

State	Red LED	Amber LED	Green LED
green	off	off	on
amber	off	on	off
red	on	off	off


We can add another traffic light to our state table - T1 is the first traffic light and T2 is the second traffic light. These traffic lights are arranged on different streets of an intersection (they need to work together to make sure traffic runs smoothly).

State	T1 Red LED	T1 Amber LED	T1 Green LED	T2 Red LED	T2 Amber LED	T3 Green LED
1	off	off	on	on	off	off
2	off	on	off	on	off	off
3	on	off	off	off	off	on
4	on	off	off	off	on	off

Note: When one traffic light changes to amber, the other traffic light stays red - this gives the traffic enough time and warning to stop, before the traffic from the other street gets the green light to go.

Step 2: Reuse our code

We need to remove the main `while` loop from our previous code, so that we can make it into a **module** that we can `import`.


- ❑ Launch **mu editor** as we did in Step 1 of the **Single Traffic Light** project. Click on the **load**  button in mu, to load our `stopbit.py` script. Delete the while loop from the bottom of the script so it looks like this:

```
from microbit import *

def stopBit(colour):
    if colour == "green":
        pin0.write_digital(0)
        pin1.write_digital(0)
        pin2.write_digital(1)
    elif colour == "amber":
        pin0.write_digital(0)
        pin1.write_digital(1)
        pin2.write_digital(0)
    elif colour == "red":
        pin0.write_digital(1)
        pin1.write_digital(0)
        pin2.write_digital(0)
```

- ❑ Click the save button in mu, to save your script.
Congratulations, you just made a module that can be imported into other scripts!

Step 3: Copy our module onto the micro:bit

- ❑ Connect the first micro:bit and click the **Files**  button in mu. You'll see two boxes open up at the bottom of mu. Find your `stopbit.py` module file in the right box, and drag it to the box labelled `Files on your micro:bit`. If you've already got some code **flashed** on your micro:bit, you'll see a `main.py` file there, too.

Files on your micro:bit:

```
stopbit.py  
main.py
```


- ❑ Repeat the process by plugging in the second micro:bit and clicking **Files** again, to drag the `stopbit.py` module to that micro:bit too.

Step 4: Write code for the first micro:bit

Now we can convert the new state table into our main loop on the first micro:bit.

State	T1 Red LED	T1 Amber LED	T1 Green LED	T2 Red LED	T2 Amber LED	T3 Green LED
1	off	off	on	on	off	off
2	off	on	off	on	off	off
3	on	off	off	off	off	on
4	on	off	off	off	on	off

It looks like a lot of work, but don't worry because the `stopBit()` function in our module, does most of the work for us!

- ❑ Click on **New**  in **mu editor** to open up a new tab and add the following code at the top:

```
from microbit import *  
from stopbit import stopBit
```

- ❑ Because we just imported our `stopBit()` function, we don't have to rewrite it!

Add the main `while` loop below that - this time we have the red and amber function calls too (**Remember:** the new lines you need to add are the bolded ones)

```
from microbit import *
from stopbit import stopBit

while True:
    stopBit("green") # state 1
    sleep(5000)
    stopBit("amber") # state 2
    sleep(2500)
    stopBit("red") # state 3
    sleep(2500)
    stopBit("red") # state 4
    sleep(2500)
```

Flash this to the first micro:bit and check that it cycles through the different colours. The comments help us remember that each `stopBit()` call corresponds to each of our states.

Step 5: Use radio to send signals (first micro:bit)

- ❑ We need the radio module to be able to send our signals from the first traffic light. Under that, add `import radio` and code to setup the radio. If you're near other micro:bits using radio, make sure to set a different channel.

```
from microbit import *
from stopbit import stopBit
import radio

radio.on()
radio.config(channel=7)
```

- ❑ To send a message to the second micro:bit, use the `radio.send()` function. For example, to send the message "green": `radio.send("green")`.

In our truth table, when Traffic Light 1 has "red" lit, Traffic light 2 has "green" lit.

Looking at the state table, send the corresponding message for Traffic Light 2, under each `stopBit()` function call.

The code for the first micro:bit should now look like this:

```
from microbit import *
from stopbit import stopBit

import radio
radio.on()
radio.config(channel=7)

while True:
    stopBit("green") # state 1
    radio.send("red")
    sleep(5000)
    stopBit("amber") # state 2
    radio.send("red")
    sleep(2500)
    stopBit("red") # state 3
    radio.send("green")
    sleep(2500)
    stopBit("red") # state 4
    radio.send("amber")
    sleep(2500)
```

Save this code in your `mu_code` directory as `t1.py` and then flash it to the first micro:bit.

Step 6: Write code to receive radio (second micro:bit)

The only difference in the **second micro:bit** is in the main `while` loop. This time, we just need to call `stopBit()` whenever we receive a radio message.

- ❑ Connect the second micro:bit to your computer, and click the **New** button in mu editor, to open a new tab, and copy over the code from `t1.py`. Now, remove everything after the `while True:` line in this new tab.

The code in the new tab should look like this:

```
from microbit import *
from stopbit import stopBit

import radio
radio.on()
radio.config(channel=7)
```

```
while True:
```

- ❑ Now just add three lines to receive and set the LEDs on the second micro:bit.

```
from microbit import *
from stopbit import stopBit

import radio
radio.on()
radio.config(channel=7)

while True:
    incoming = radio.receive()
    if incoming:
        stopBit(incoming)
```

- ❑ Save the tab with the above code in the `mu_code` directory, as `t2.py`. Flash the code onto the second micro:bit and use the **Files** button to drag the `stopbit.py` module on.

With the STOP:bits connected to each micro:bit, you can now attach a battery pack to each and draw a mini intersection on a piece of paper and watch the lights work together!

Congratulations, you've just made a working set of Australian traffic lights! 