Traffic Lights Crossroads and Cars

Introduction

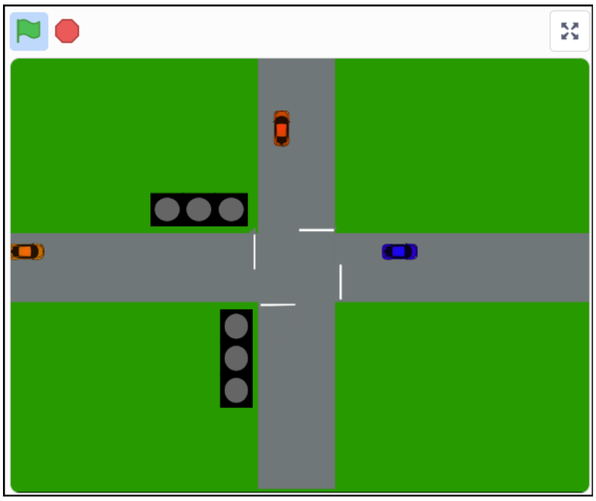
In this SCRATCH project you will program a set of traffic lights that will control the flow of cars at crossroads. This is a follow on from the Traffic Lights project completed by the Wokingham Library Online Code Club in November 2020. If you would like to look at the videos for that project please see:

https://youtu.be/O_glSaiaM9A (7th November 2020)
https://youtu.be/-PU2A6Q7RiY (21st November 2020)

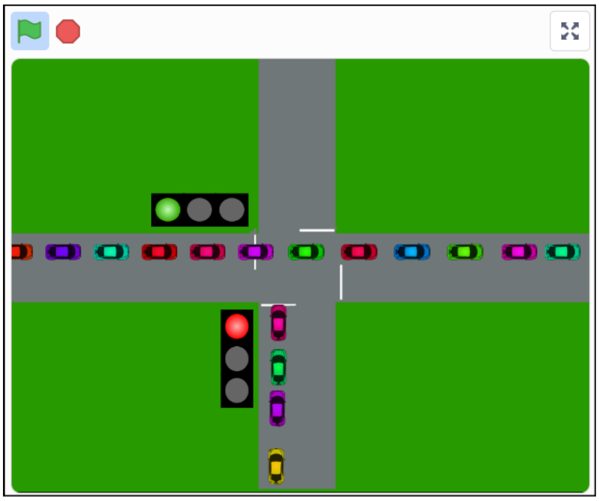
This project has a Starter project - https://scratch.mit.edu/projects/455279461/

When you open the Starter project you'll see the layout of a crossroads with two sets of traffic lights, which aren't yet working.

If you click the green flag ⚑ you'll see cars start 'driving' from left to right & from bottom to top. You might see some near misses!



The task of this project is to add code to the traffic lights in order that the traffic is controlled. i.e. cars will queue when the lights are at red, will slow down when the lights are at amber and will only 'go' when the lights are at green, as illustrated in this screen capture.



Contents of the Starter project

The starter project has some sprites with some pieces code already included.

Backdrop and Sprites:

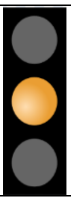
- a **Stage Backdrop** showing the green field and crossroads with white ‘stop’ lines
- a **Car X** sprite, which generates the cars moving horizontally; left to right
- a **Car Y** sprite, which generates the cars moving vertically; bottom to top
- a **Lights X** sprite, used to control the cars moving horizontally
- a **Lights Y** sprite, used to control the cars moving vertically
- a **Start Button** sprite, which will be used to turn the lights ‘on’, after they have been ‘installed’

Code:

- code on the **Stage** to initialise some variables and lists that are available to all sprites
- code on the **Car X** and **Car Y** sprites that:
    - generates multiple cars of differing colours using clones
    - controls the speed of the cars based on:
        - the current colour of the lights (off, red, red & amber, or green)
        - the distance from the white ‘stop’ line in front of the lights, and
        - the distance between the car itself and the car in front

Costumes:

Each light sprite (**Lights X** and **Lights Y**) has 5 costumes. The name of each costume shown in this table:

					
Lights X	Lights X-red	Lights X-red&amber	Lights X-green	Lights X-amber	Lights X-off
Lights Y	Lights Y-red	Lights Y-red&amber	Lights Y-green	Lights Y-amber	Lights Y-off

‘Lights X State’ and ‘Lights Y State’ global variables

The code for the movement of the cars relies on the values of variables that are available for all sprites:

- **Lights X State** variable showing the current state of the lights for the cars moving in the horizontal direction, and
- **Lights Y State** variable showing the current state of the lights for the cars moving in the vertical direction.

The possible values for the variables correspond to the names of the costumes. They are:

- red
- red&amber
- green
- amber
- off

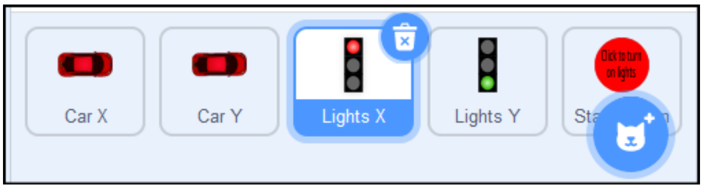
The following steps show how to create the code needed to generate the correct settings for **Lights X State** and **Lights Y State**, which will in turn cause the traffic to be controlled.

Open the Starter Project

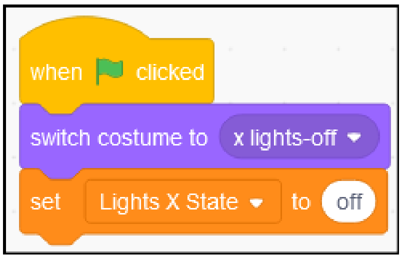
Login to your Scratch account. Open the starter project – see the link in the **Introduction** (*page 1*) **Remix** the project to save it to your own account.

Create code blocks to set the 'Lights X State' variable

a) Select the **Lights X** sprite



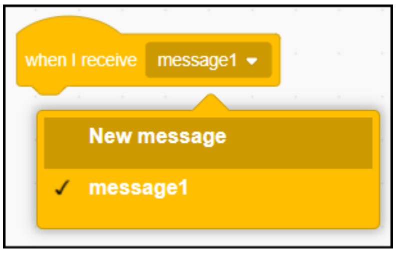
b) Using blocks in the **Looks** & **Variables** sections set the initial costume and **Lights X State** variable to 'off' when the Green flag is clicked



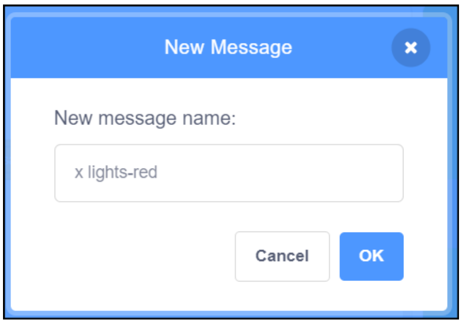
```
when 🏁 clicked
switch costume to x lights-off ▾
set Lights X State ▾ to off
```

c) Use a broadcast message to set the costume and **Lights X State** variable to 'red'

- Using the **when I receive** block in the **Events** section create a new message called 'Lights X-red':



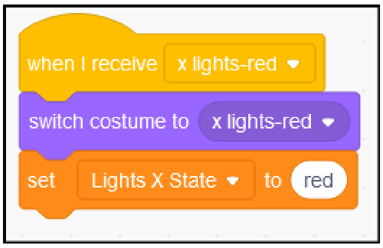
```
when I receive message1 ▾
```



New Message

New message name: x lights-red

- Now set the costume and **Lights X State** variable to 'red':



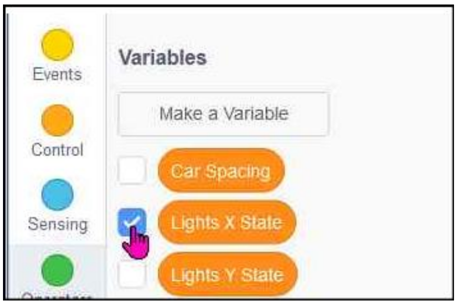
```
when I receive x lights-red ▾
switch costume to x lights-red ▾
set Lights X State ▾ to red
```

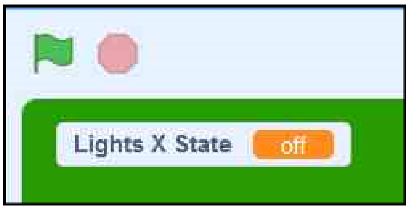
d) Repeat the step above to create similar broadcast message blocks for the other 3 costumes and states:

Broadcast message	Costume name	Lights X State setting
x lights-red&amber	x lights-red&amber	red&amber
x lights-green	x lights-green	green
x lights-amber	x lights-amber	amber

e) When you've finished this, check your work by taking a look at *page 12* for the full set of code for the **Lights X** sprite.

f) You can also test your code by clicking on each of the 5 code sections in turn. You should see the lights change to the relevant colour. If you show the **Lights X State** variable on the stage by ticking the box next to the variable you should also see the text change as well.

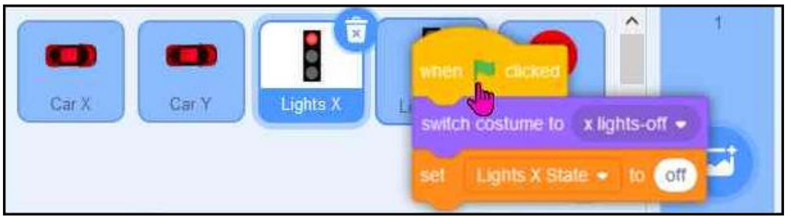



Lights X State off

Create code blocks to set the 'Lights Y State' variable

We could create code on the **Lights Y** sprite as we did above for the **Lights X** sprite. Alternatively we can use the facility in Scratch to drag and drop code from one sprite to another. We can then edit the new copied code as required.

a) Click on one of the code sections in the **Lights X** spite with the left mouse button and hold the button down. Drag the code section over the icon for the **Lights Y** sprite. When the **Lights Y** sprite 'wiggles' release the left mouse button:

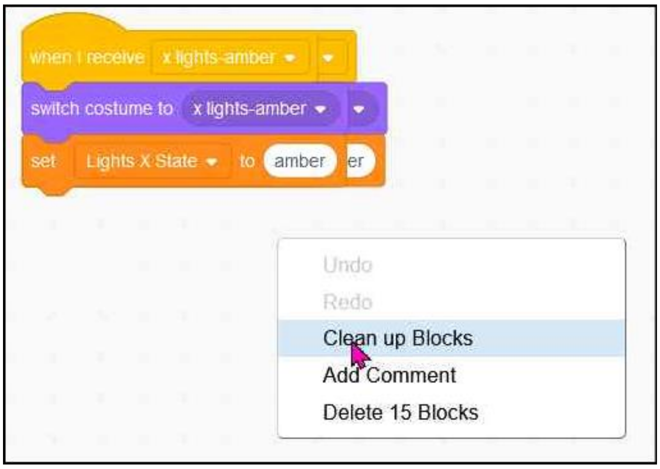


```
when 🏴 clicked
switch costume to x lights-off
set Lights X State to off
```

b) Repeat the 'drag and drop' for the other 4 code sections

c) Select the **Lights Y** sprite

d) The code sections may be sitting on top of each other. To sort them out, right-mouse click on the white background to show a pop-up menu, then choose **Clean up Blocks**



```
when I receive x lights-amber
switch costume to x lights-amber
set Lights X State to amber
```

e) Edit the code sections to:

- respond to new messages specific for the **Lights Y** sprite
- refer to the costumes for the **Lights Y** sprite
- set the **Lights Y State** variable

Broadcast message	Costume name	Lights Y State setting
Green flag	y lights-off	off
y lights-red	y lights-red	red
y lights-red&amber	y lights-red&amber	red&amber
y lights-green	y lights-green	green
y lights-amber	y lights-amber	amber

f) When you've finished all the editing, check your work by taking a look at *page 13* for the full set of code for the **Lights Y** sprite.

g) You can also test your code by clicking on each of the 5 code sections in turn. You should see the lights change to the relevant colour. If you show the **Lights Y State** variable on the stage by ticking the box next to the variable you should also see the text change as well.

Define the Traffic Light sequences

The sequence required for the two sets of traffic lights is shown by 8 states as in this table. The coloured circles indicate the light that should be displayed for the corresponding time as specified in the 3rd column.

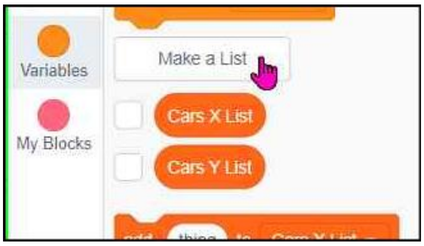
	Lights X	Lights Y	Duration (seconds)
1	🔴	🔴	1
2	🔴 🟡	🔴	2
3	🟢	🔴	5
4	🟡	🔴	2
5	🔴	🔴	1
6	🔴	🔴 🟡	2
7	🔴	🟢	5
8	🔴	🟡	2

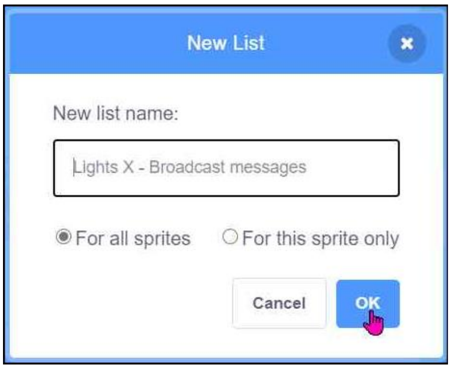
We will set up these sequences using 3 Scratch Lists:

- **Lights X - Broadcast messages**, contains the ordered list of broadcast messages for the cars moving horizontally
- **Lights Y - Broadcast messages**, contains the ordered list of broadcast messages for the cars moving vertically
- **Lights - Duration**, contains the ordered list of durations that the lights remain in each state

Lights X Sequence

a) Using the **Make a List** option in the **Variables** section to create a new list called **Lights X - Broadcast messages**. We want other sprites to able to read this list so make sure you choose the **For all sprites** option.



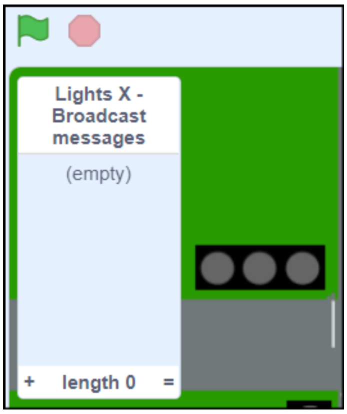

New List

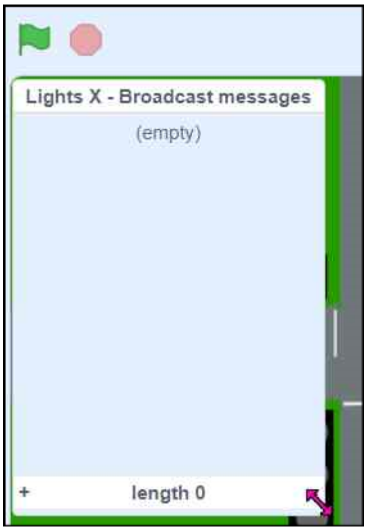
New list name: Lights X - Broadcast messages

● For all sprites ○ For this sprite only

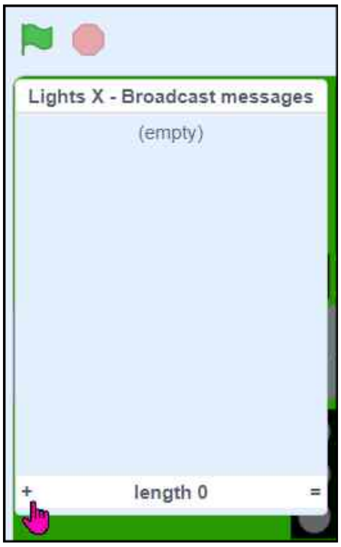
Cancel OK

b) You should see an empty list in the Stage area. The list can be made wider by using the mouse to drag the bottom right corner of the list down and to the right.



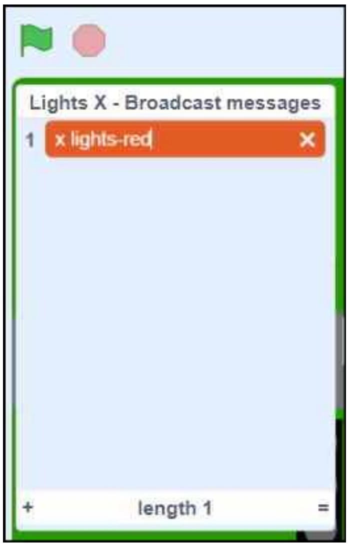


c) Add the name of the first broadcast message corresponding to red light on the **Lights X** sprite. (We created the broadcast messages on *page 3*.) To do this click on the little **+** in the bottom left of the list, then type the name of the first broadcast message into the box.


Lights X - Broadcast messages

(empty)

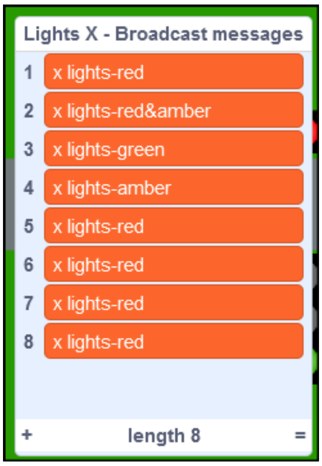
+ length 0 =


Lights X - Broadcast messages

1 x lights-red

+ length 1 =

d) Repeat the previous step to add the other 7 broadcast messages. Your list should now look like the screen shot below:


Lights X - Broadcast messages

1 x lights-red
2 x lights-red&amber
3 x lights-green
4 x lights-amber
5 x lights-red
6 x lights-red
7 x lights-red
8 x lights-red

+ length 8 =

Lights Y Sequence

Repeat the above steps to create a list for the other set of lights, **Lights Y - Broadcast messages**. It will be similar to the list you've just created but not the same. Use the chart on page 5 to help you decide which order to add the items to the list.

Lights - Duration Sequence

Using similar methods now create a **Lights - Duration** list using the values in the **Duration** column in the chart on page 5.

When you've finished check your work by taking a look at *page 14* for a screen shot showing all the populated lists.

Program the Light Sequence

We will create the code for making the lights run through their sequences on the **Start Button** sprite. The light sprites change according to **broadcast** messages that they receive. The messages to be sent are contained in **Lights X - Broadcast messages** and **Lights Y - Broadcast messages** lists that we created in the last section. The **Duration** list contains the amount of time we want to wait between each message being broadcast. The basic steps are:

1. Broadcast message from **1st** item in the **Lights X - Broadcast messages** list
2. Broadcast message from **1st** item in the **Lights Y- Broadcast messages** list
3. Wait for time of **1st** item in the **Duration** list
4. Broadcast message from **2nd** item in the **Lights X - Broadcast messages** list
5. Broadcast message from **2nd** item in the **Lights Y- Broadcast messages** list
6. Wait for time of **2nd** item in the **Duration** list
7. ….
8. ….
9. ….
10. Broadcast message from **8th** item in the **Lights X - Broadcast messages** list
11. Broadcast message from **8th** item in the **Lights Y- Broadcast messages** list
12. Wait for time of **8th** item in the **Duration** list

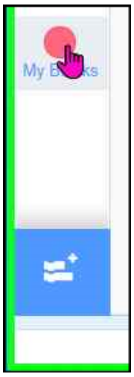
Initialise the Start Button

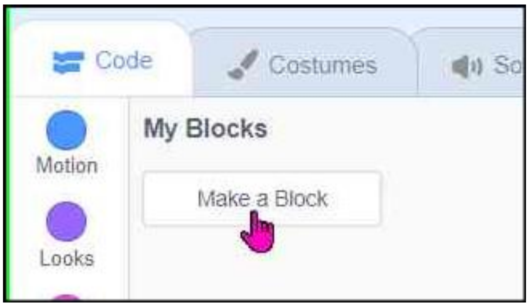
Using the **Show** block in the **Looks** section add the code to show the **Start Button** when the **Green Flag** is clicked

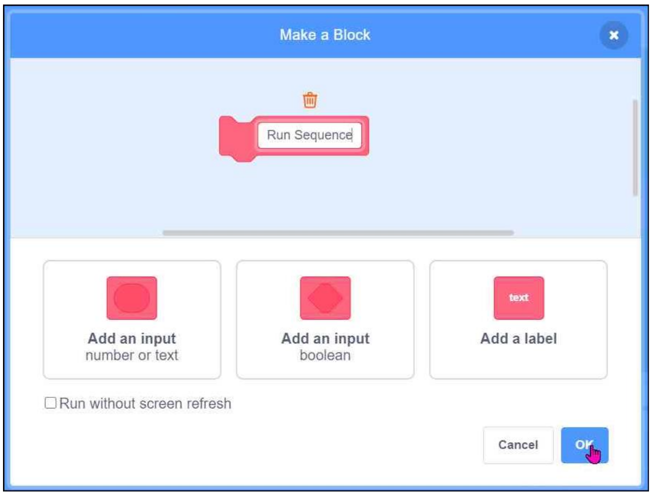
Add a Block for the sequence code

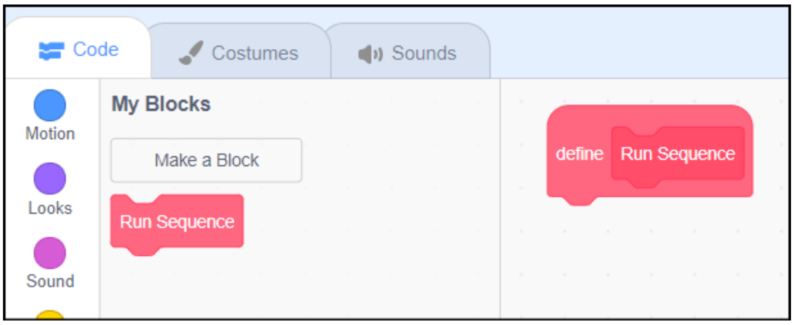
We'll put all the code for running the sequence into a Block. We will only call the Block once, so it isn't entirely necessary, but it does clearly show what this part of the code is doing.

a) Create new Block called **Run Sequence**

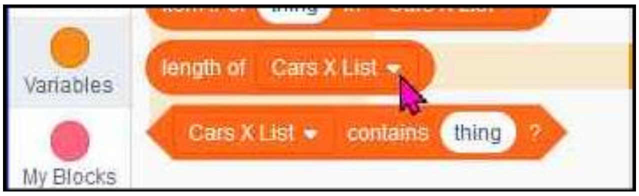




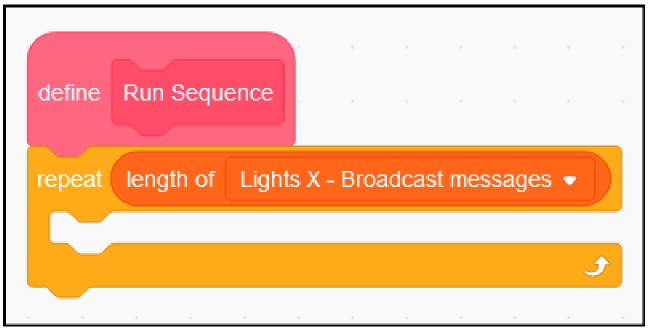




b) We need to create loop that repeats for the number of items in one of the **Broadcast messages** lists. We know that this is 8. Another way to get the number of items in a list will be to use the **length of** block in the **Variables** section:

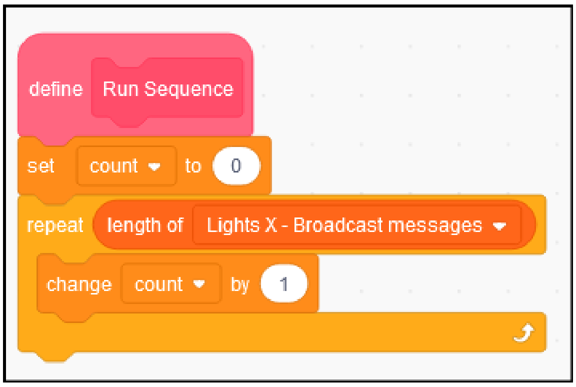


c) We have created the lists to be all the same length so it doesn't matter which one we use. The **repeat** loop should look like this:



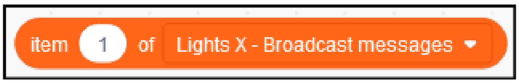
d) We need a way to reference the items in the lists. We'll do this using a variable.

- Create a new variable called **count**. It only needs to be available to this sprite
- Initialise it to zero outside the **repeat** loop and increment it by 1 inside the loop



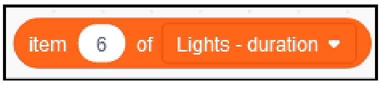
```
define Run Sequence
set count to 0
repeat length of Lights X - Broadcast messages
    change count by 1
```

e) Also included in the **Variables** blocks is a block to obtain the data from a specific row of a **List**. The following will obtain the 1st item of the **Lights X - Broadcast messages** list, i.e "*x lights-red*"



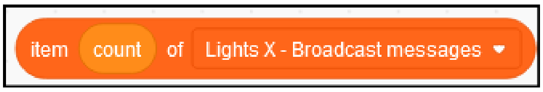
```
item 1 of Lights X - Broadcast messages
```

and this will obtain the 6th item of the **Lights - duration** list, i.e "*2*"

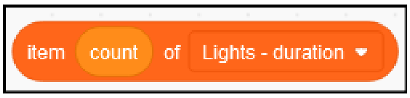


```
item 6 of Lights - duration
```

In place of actual numbers we need to use the **count** variable, so the above blocks will look like this

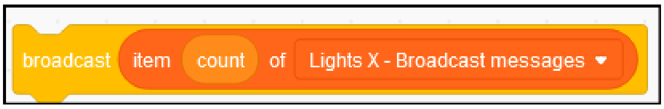


```
item count of Lights X - Broadcast messages
```



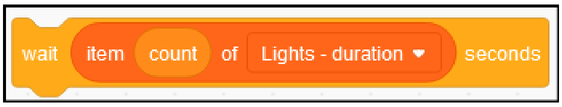
```
item count of Lights - duration
```

This will **broadcast** a message for the text contained in item **count** of the **Lights X - Broadcast messages** list



```
broadcast item count of Lights X - Broadcast messages
```

and this will **wait** for the number of seconds contained in item **count** of the **Lights - duration** list



```
wait item count of Lights - duration seconds
```

f) Based on the information above, now add three blocks of code underneath the **change count** block, but inside the **repeat** loop. These blocks should

- Broadcast a message for item **count** from the **Lights X - Broadcast messages** list
- Broadcast a message for item **count** from the **Lights Y - Broadcast messages** list
- Wait for item **count** from the **Lights - duration** list

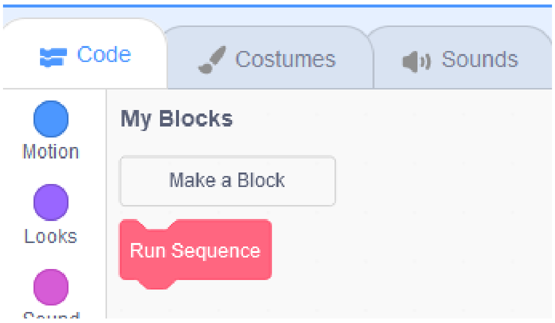
g) The above steps will run through the traffic light sequence once. To make it repeat the sequence continuously add a **forever** block from the **Control** section. All the code under the **define** block needs to be enclosed inside the **forever** loop.

h) When you've finished check your work by taking a look at page 15 for a screen shot showing all the code for the **Start Button** sprite.

Action when clicking the Start Button

a) Use the event **when this sprite is clicked** and add the code to hide the **Start Button**

b) Next, add the call to the **Run Sequence** block



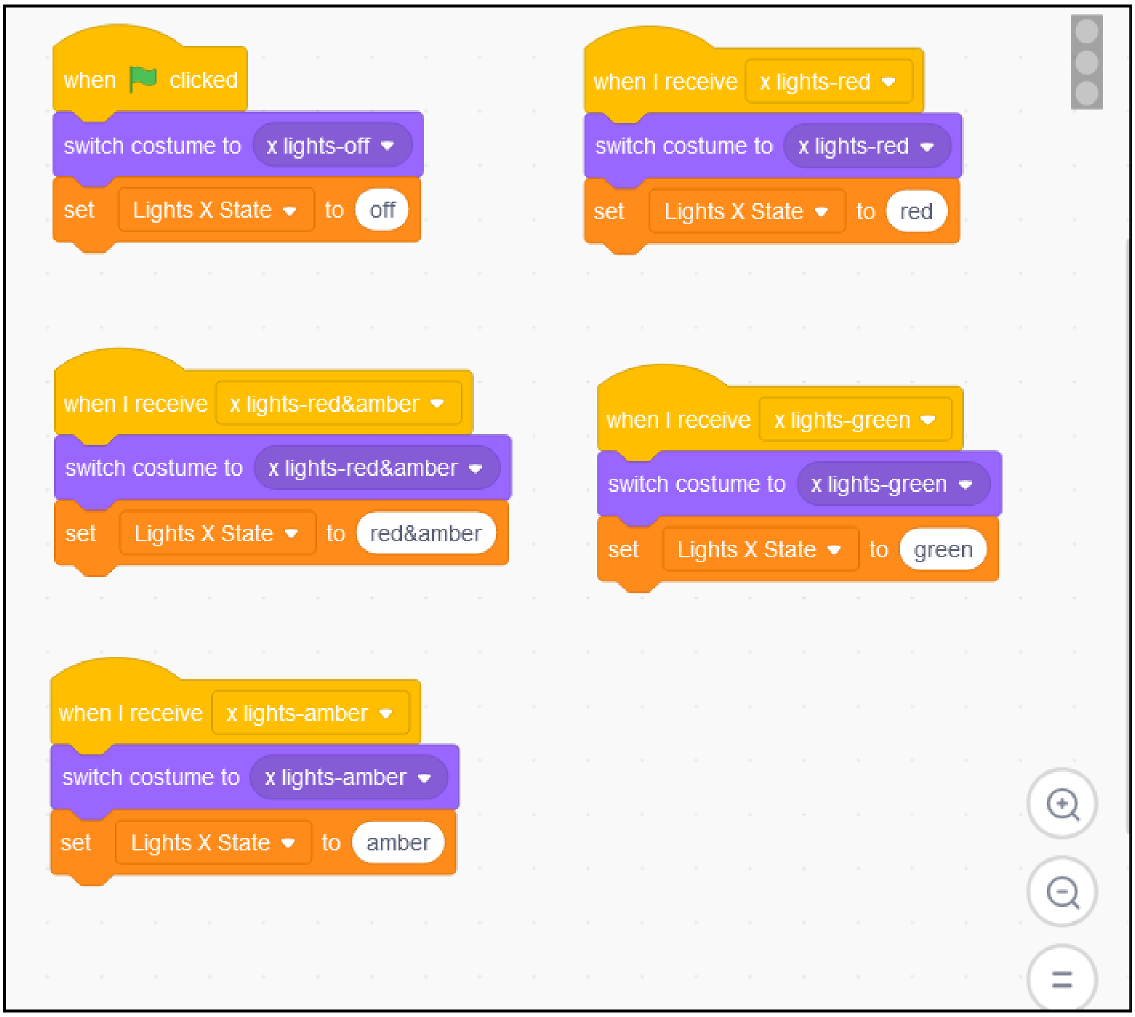
c) When you've finished check your work by taking a look at page 15 for a screen shot showing all the code for the **Start Button** sprite.

The program is complete – run the code

a) Click the **Green flag** – the cars should flow freely in both directions

b) Click the **Start Button** sprite – both sets of lights should turn 'on' and the cars should start responding to the lights.

Lights X sprite - code



```
when ⚑ clicked
switch costume to x lights-off ▾
set Lights X State ▾ to off
```

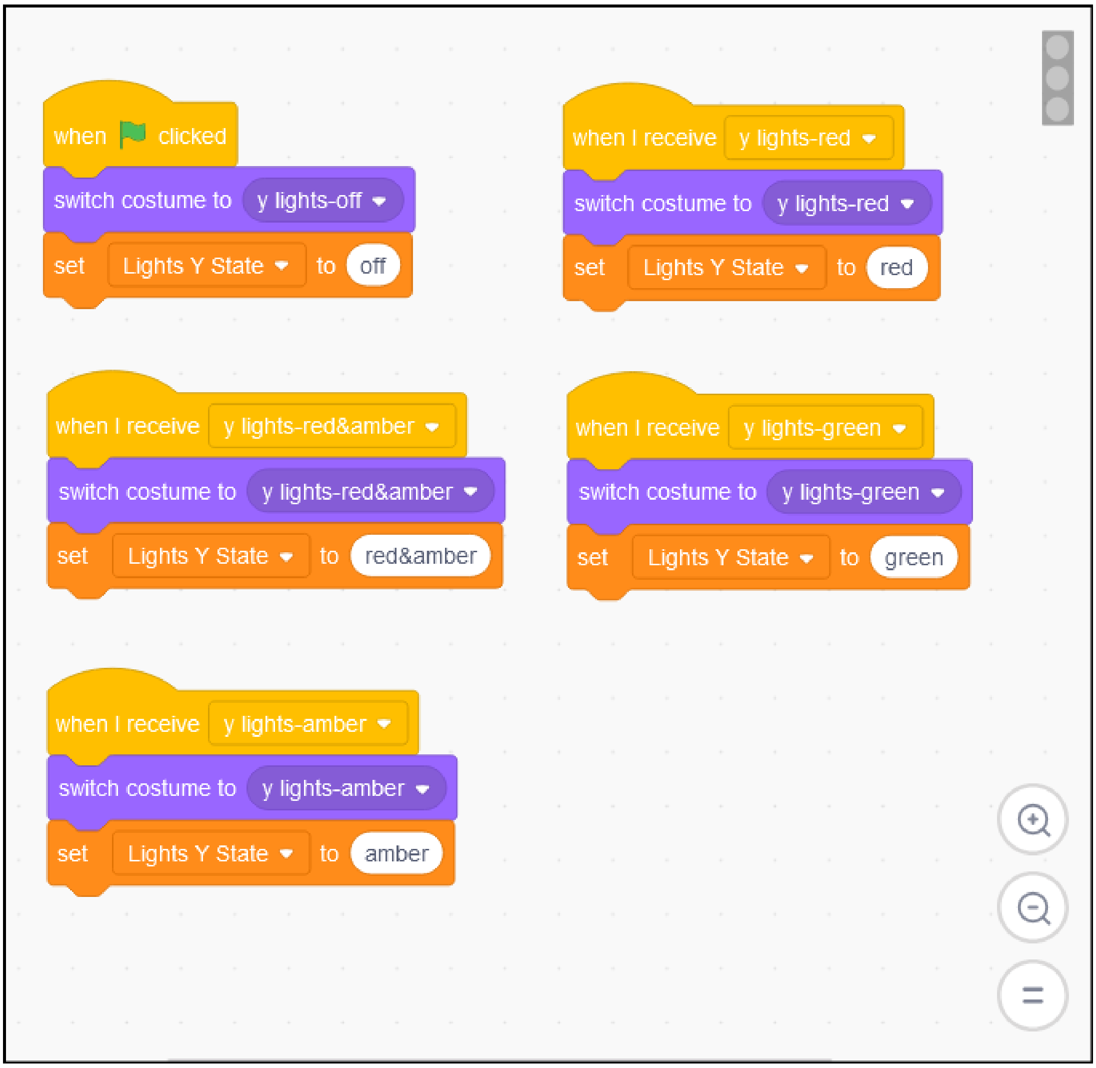
```
when I receive x lights-red ▾
switch costume to x lights-red ▾
set Lights X State ▾ to red
```

```
when I receive x lights-red&amber ▾
switch costume to x lights-red&amber ▾
set Lights X State ▾ to red&amber
```

```
when I receive x lights-green ▾
switch costume to x lights-green ▾
set Lights X State ▾ to green
```

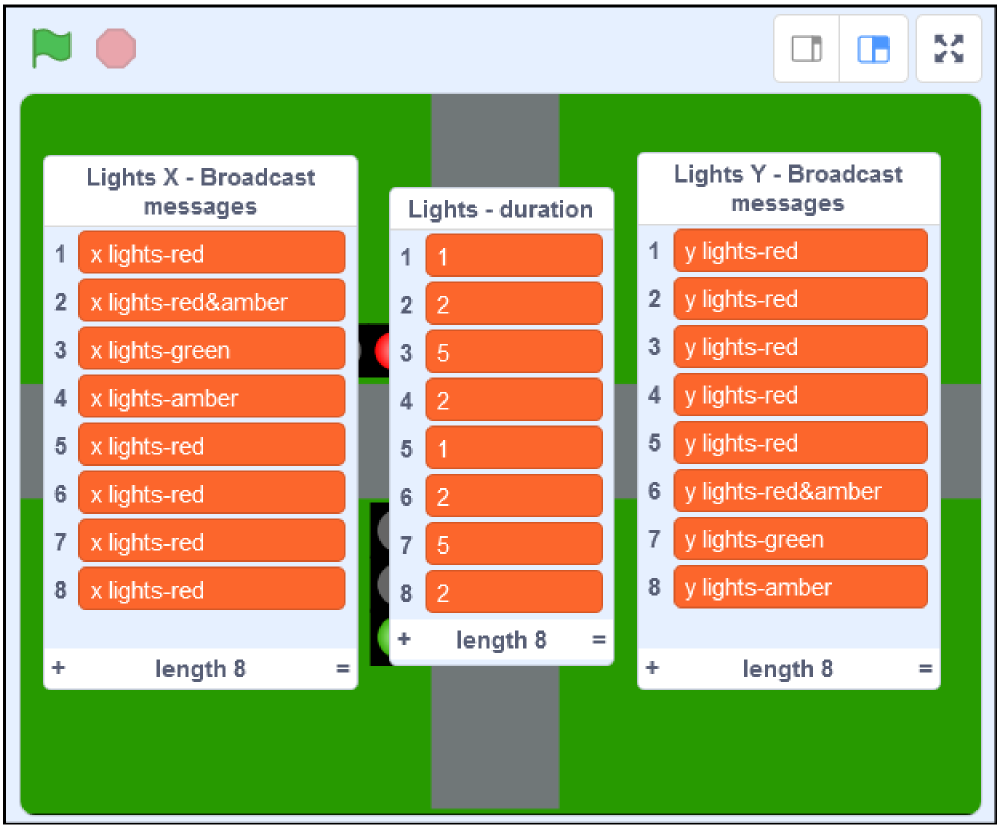
```
when I receive x lights-amber ▾
switch costume to x lights-amber ▾
set Lights X State ▾ to amber
```

Lights Y sprite – code



```
when ⚑ clicked
switch costume to y lights-off ▾
set Lights Y State ▾ to off

when I receive y lights-red ▾
switch costume to y lights-red ▾
set Lights Y State ▾ to red

when I receive y lights-red&amber ▾
switch costume to y lights-red&amber ▾
set Lights Y State ▾ to red&amber

when I receive y lights-green ▾
switch costume to y lights-green ▾
set Lights Y State ▾ to green

when I receive y lights-amber ▾
switch costume to y lights-amber ▾
set Lights Y State ▾ to amber
```

Broadcast message lists



Lights X - Broadcast messages

1	x lights-red
2	x lights-red&amber
3	x lights-green
4	x lights-amber
5	x lights-red
6	x lights-red
7	x lights-red
8	x lights-red

+ length 8 =

Lights - duration

1	1
2	2
3	5
4	2
5	1
6	2
7	5
8	2

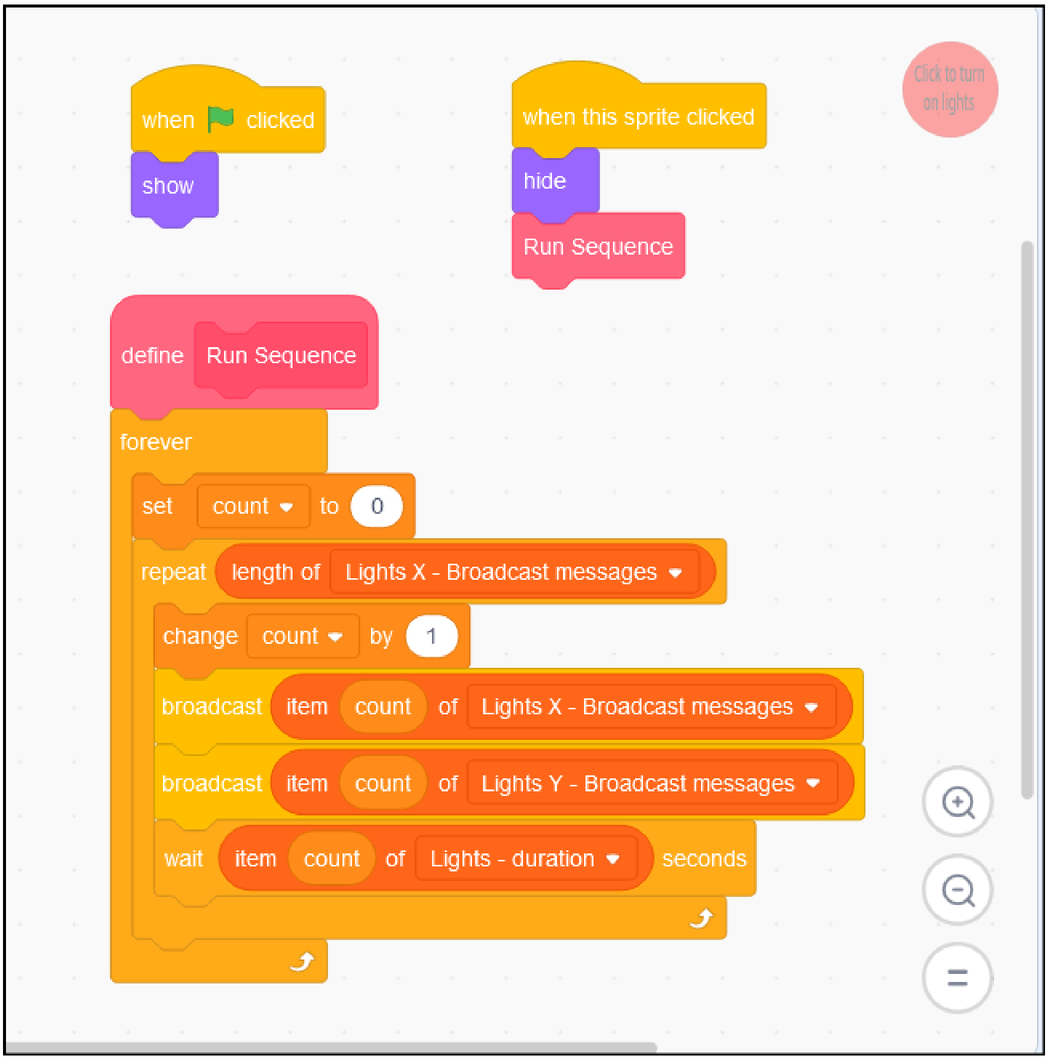
+ length 8 =

Lights Y - Broadcast messages

1	y lights-red
2	y lights-red
3	y lights-red
4	y lights-red
5	y lights-red
6	y lights-red&amber
7	y lights-green
8	y lights-amber

+ length 8 =

Start Button Code



```
when 🏁 clicked
show

when this sprite clicked
hide
Run Sequence

define Run Sequence
forever
    set count ▾ to 0
    repeat length of Lights X - Broadcast messages ▾
        change count ▾ by 1
        broadcast item count of Lights X - Broadcast messages ▾
        broadcast item count of Lights Y - Broadcast messages ▾
        wait item count of Lights - duration ▾ seconds
```