runlinc Project 05 B5:
Traffic Lights (E32W Version)

Contents

Introduction 1
Part A: Design the Circuit on runlinc 3
Part B: Build the Circuit 4
Part C: Program the Circuit 7
Extension – Traffic Light Animation 12
Summary 14

Introduction

Problem

The correct order and timing of instructions are critical. To investigate this, we want to build a set of traffic lights by using our E32W board and run it on runlinc.

Background

By learning STEMSEL, you can learn how to program microchips and tell them what to do. Microchips can monitor devices and warn people like when your laptop battery is low, it will tell you that you need to plug the cable in. However, they can also control people's behaviour more directly, such as electronic road signs and traffic lights. But the microchips must send the right message, otherwise there can be some problems, like turning on the green lights of both directions at the same time. It is not only important to consider the correct sequence, but also it is important to think about how long each colour should be on for. For example, if the green light was only on for one second, the cars would not have enough time to get across the intersection. Microchips should be programmed so that the function of the traffic lights are correct in order to avoid accidents

Ideas

What kind of things will we need for our traffic light? What are the inputs/outputs of traffic lights? What if the pedestrians want to cross? How many traffic lights are there when two roads intersect? What is the correct sequence of the lights? What if the green light was only on for 1 second, would that be long enough for the cars to cross? How long should each light be on for?

Plan

As we all know, all traffic lights have a red light on the top, yellow light on the middle and green light on the bottom. About visibility: Why are the lights safer if the red light is at the top? The correct sequence of lights is green -> yellow -> red, then back to green again. For timing in Australia, it is the law that the yellow light should be on for at least 3 seconds. If we turn on the green light for 7 seconds to let the cars go across the intersection, how long should the red light be on for? Well, while the traffic light facing one way is red, the traffic light facing the other way will be green and yellow. Therefore, our traffic lights will need to be red for 7+3 = 10 seconds.

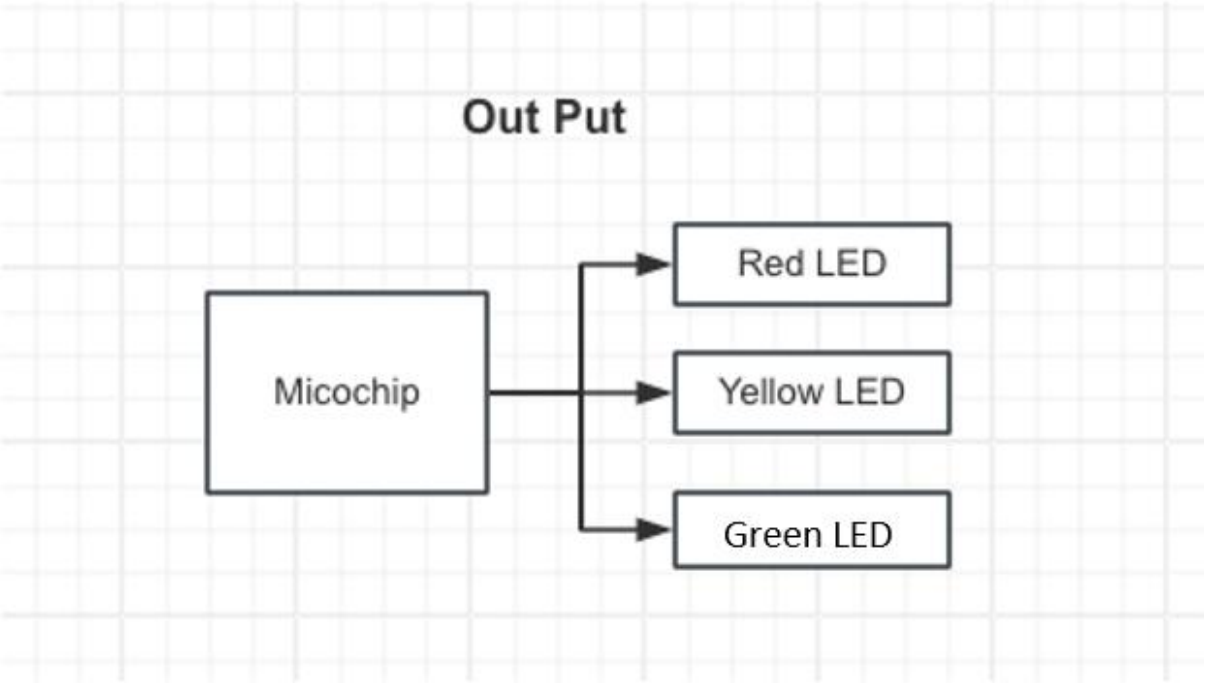


Figure 1: Block diagram of Microchip outputs

runlinc Background

runlinc is a web page inside a Wi-Fi chip. The programming is done inside the browsers compare to programming inside a chip. The runlinc web page inside the Wi-Fi chip will command the microchips to do sensing, control, data logging Internet of Things (IoT). It can predict and command.

Part A: Design the Circuit on runlinc

Note: Refer to runlinc Wi-Fi Setup Guide document to connect to runlinc

Use the left side of the runlinc web page to construct an input/output (I/O).

For port D05 name it RedLED set it as DIGITAL_OUT.
For port D18 name it GreenLED and set it as DIGITAL_OUT. (Since RGB LEDs needs 'PWM mixing' to achieve the colour yellow, we'll use Red+Green.)
For port D19 name it negative and set it as DIGITAL_OUT.

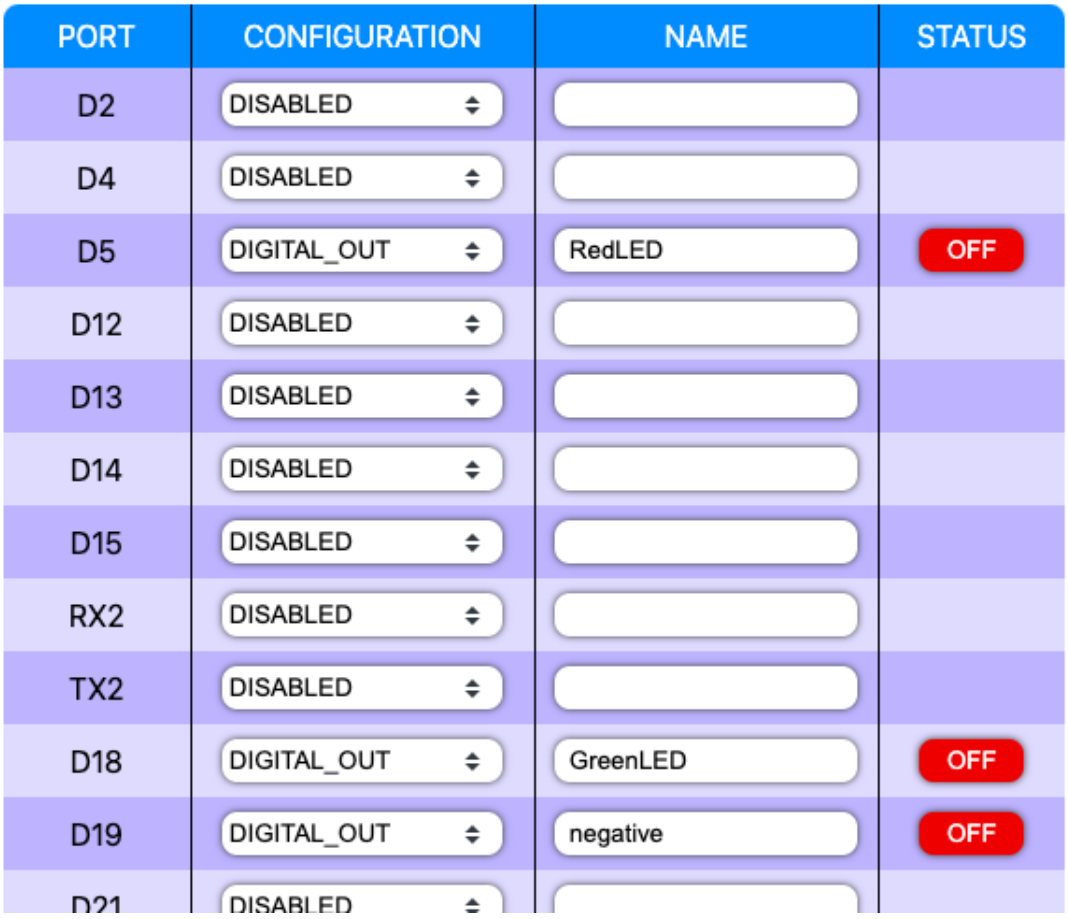


Figure 2: I/O configurations connections

Part B: Build the Circuit

Use the STEMSEL E32 board to connect the hardware. For this project we are using both the left and right I/O ports, with **negative port (-ve)** on the outer side, **positive port (+ve)** on the middle and **signal port (s)** on the inner side (as shown below).

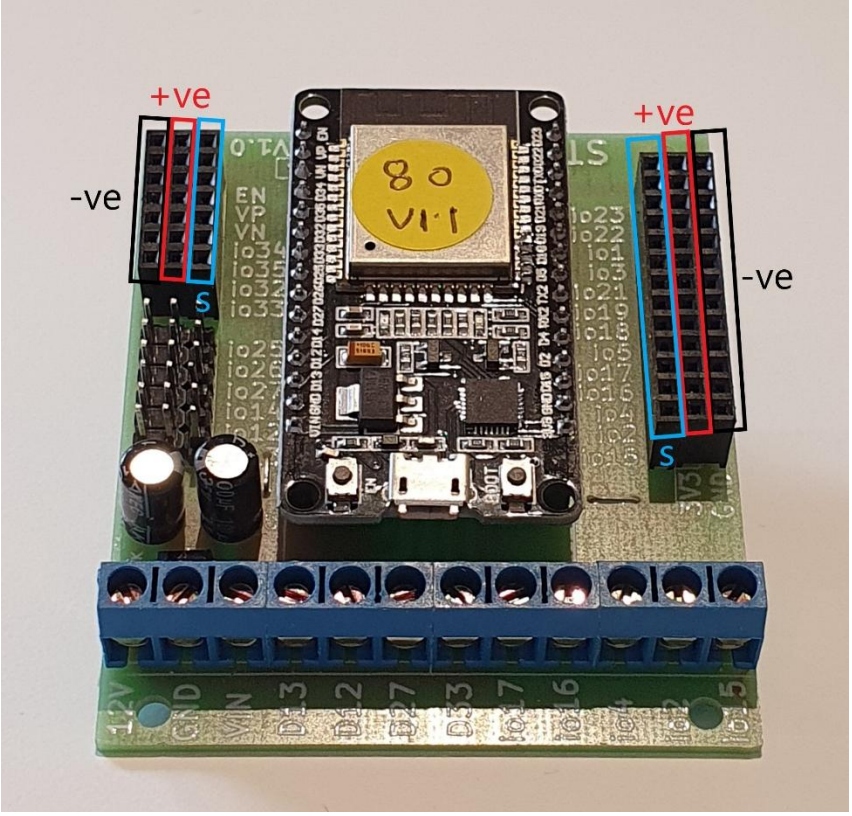


Figure 3: Negative, Positive and Signal port on the E32W board

There is only one I/O parts we are using for this project, a Dual Color LED, its respective pins are shown in the figure below.

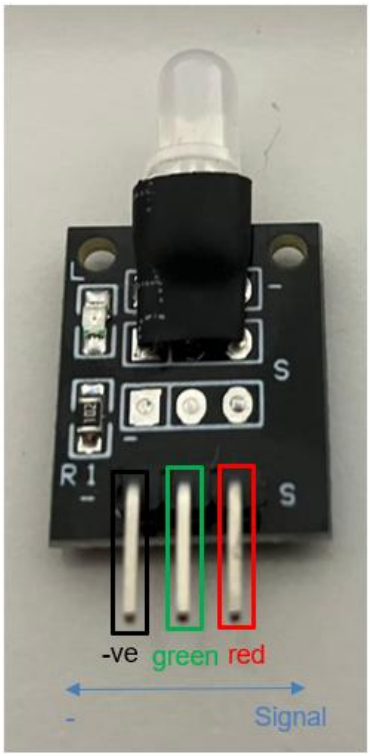


Figure 4: I/O parts with negative, positive and signal pins indicated

Wiring instructions

Since we only have 3-pins port slots, we need to plug in all 3 pins on the Dual Color LED into the signal ports.

a.) Plug in the negative pin on the Dual Color LED into port io21, the rest should be in horizontal order and plug into port io19, io18, io5. (Signal "S" on io5)

b.) Please refer to **Figure 5** for more detail.

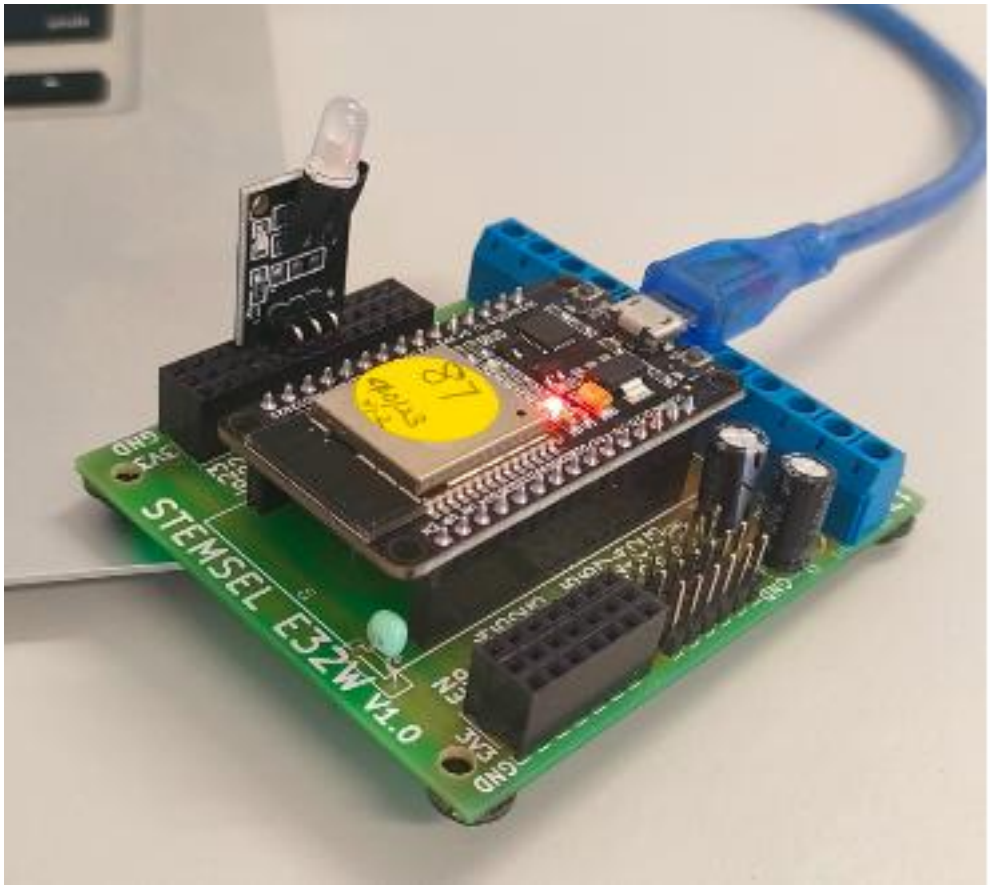


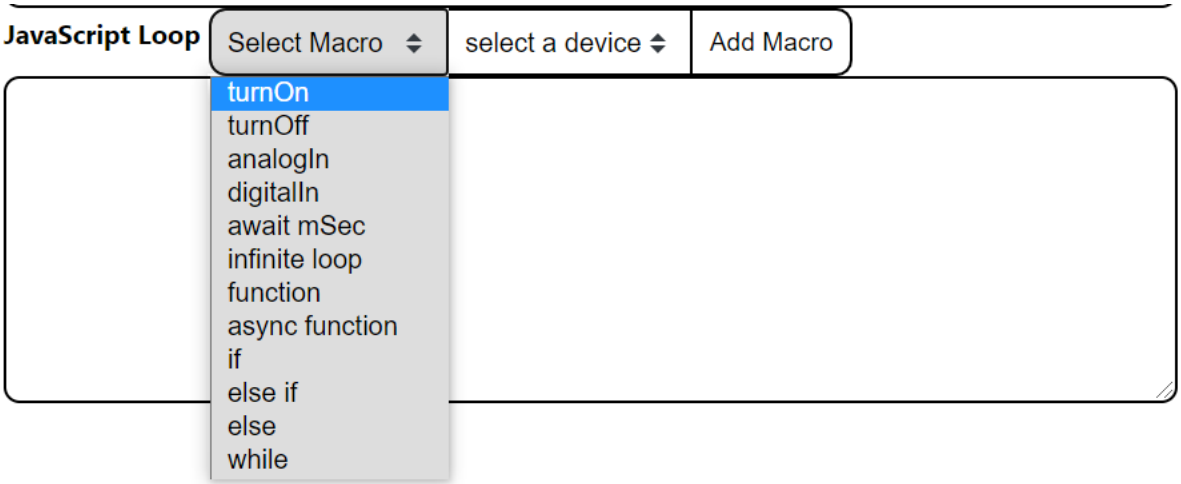
Figure 5: Circuit board connection with I/O part (side view)

Figure 6: Circuit board connection with I/O part (top view)

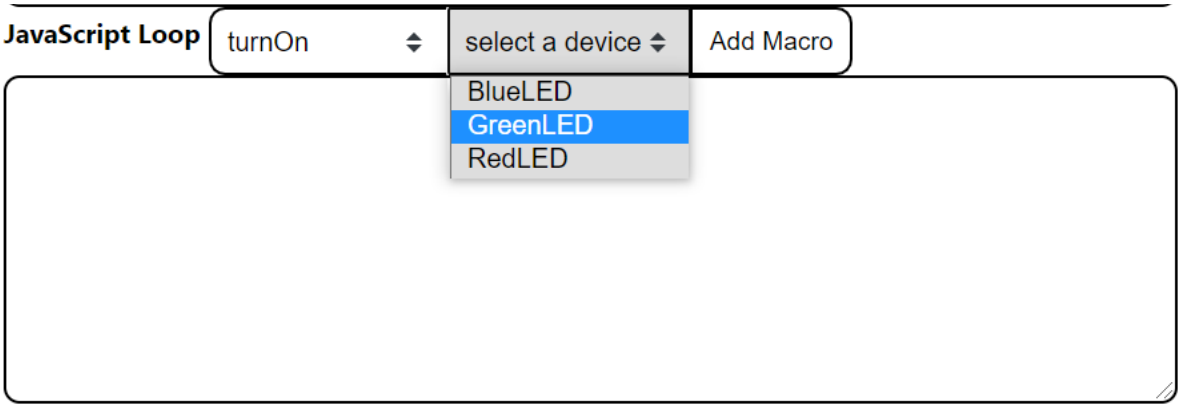
Part C: Program the Circuit

Use the blocks on the right side of the runlinc webpage to program the functions of the traffic light. Use the HTML to add content, CSS to add style to your taste and JavaScript to program the microchip. In this case, only JavaScript Loop is needed to program it to act as a traffic light.

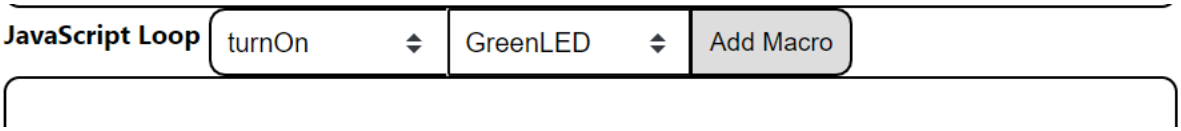
This project we need to program in the JavaScript Loop and HTML box. After naming the port D18, D19 and D21, we are going to program the circuit. First of all, we want to turn on the Green LED. Go to JavaScript Loop, there is a Select Macro button, select the turnOn:



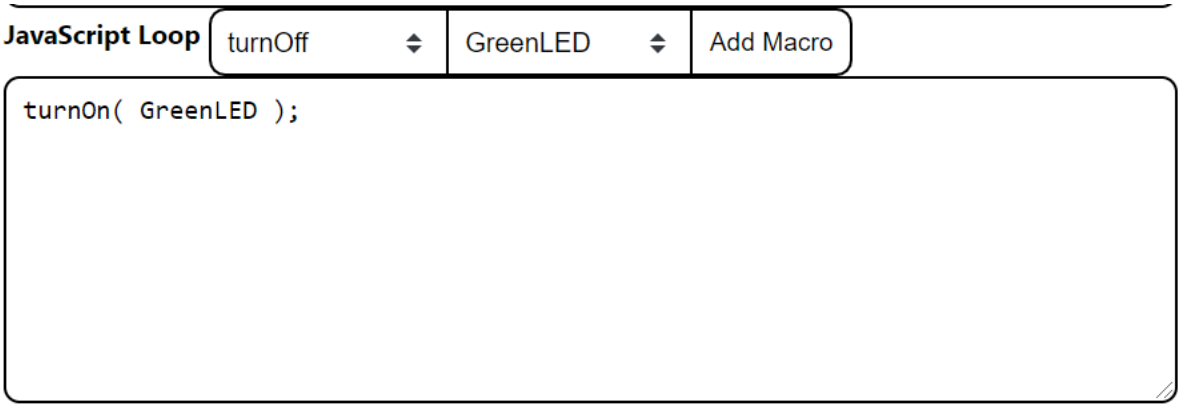
And then go to the next button, select a device. Select GreenLED.



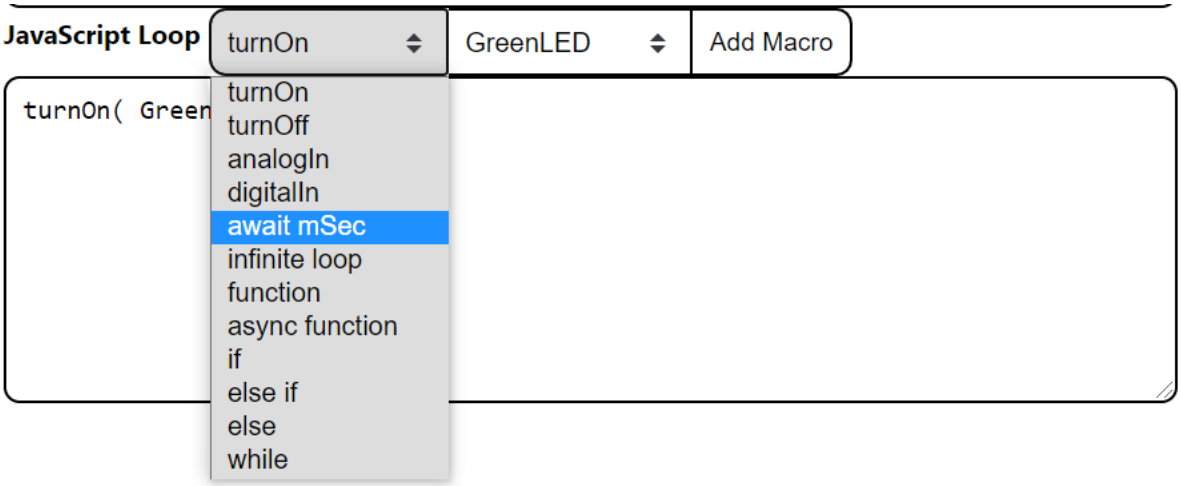
And then click the Add Macro button.



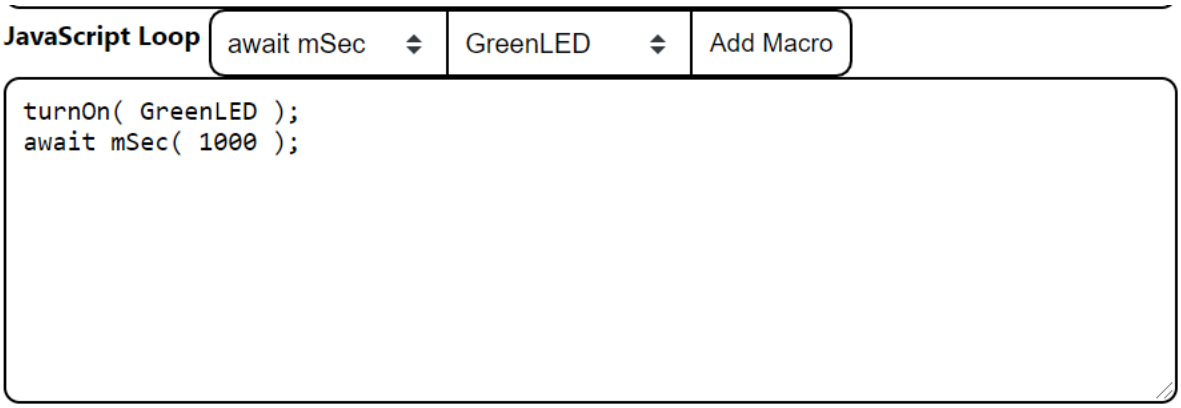
You will have the turnOn(GreenLED) program.

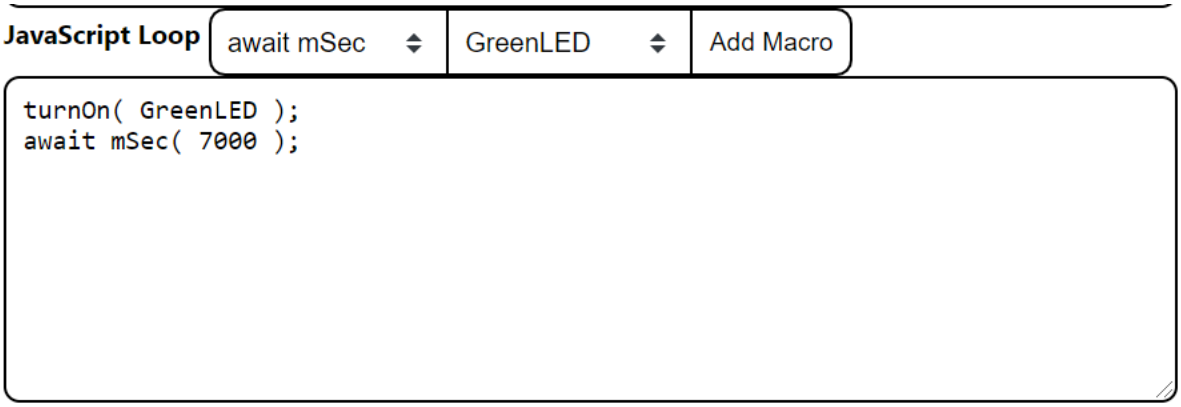


Then, we need to keep the Green LED turned on for 7 sec. Go to the Select Macro button, choose the await mSec.

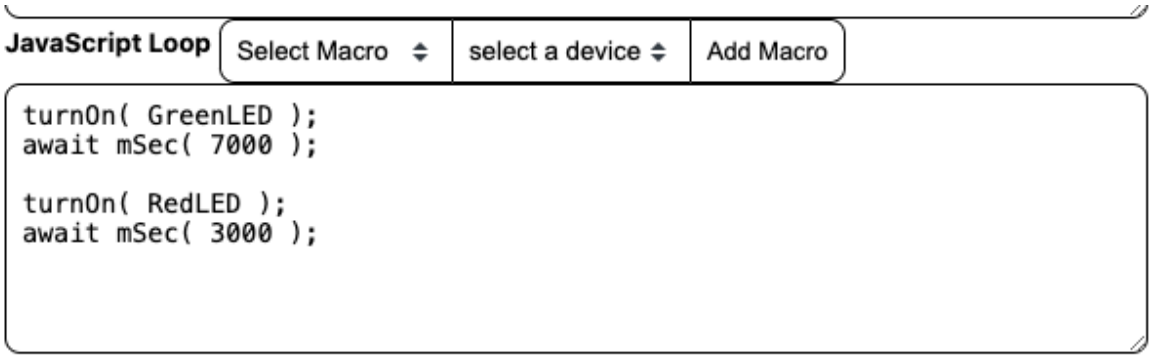


After click the **Add Macro**, you will have a time delay of 1000. 1000 in the brackets means 1-sec delay. But we need the Green LED to be turned on for 7 sec. Therefore, we change 1000 in the brackets to 7000.



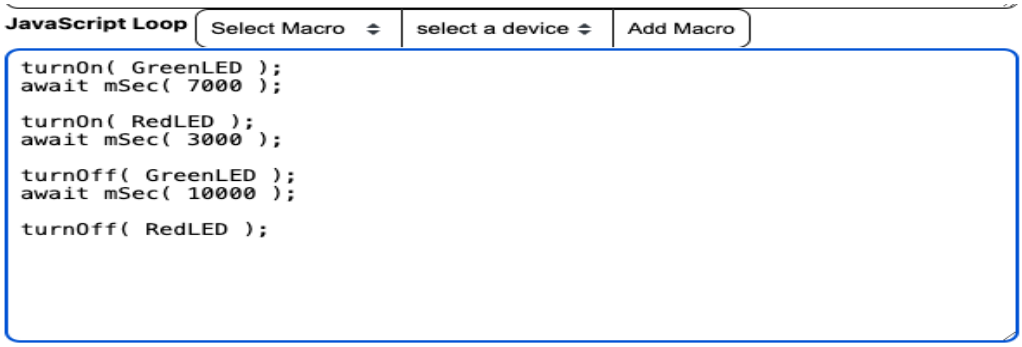


For the next Yellow Light delay 3 sec, you need to use Green + red = Yellow, which is turn On RedLED for 3 sec. At this point the green LED will not turn off, as we are not writing “turnOff” to turn it off.



For the next Red Light delay 10 sec, you need to turnoff Green light for 10 sec.

At last turn off red Led for reset.



For **JavaScript Loop** the final code should be:

```
turnOn( GreenLED );  \\ Green
await mSec( 7000 );  \\ 7 Seconds

turnOn( RedLED );     \\ Green + Red = Yellow
await mSec( 10000 );  \\ 10 Seconds

turnOff( GreenLED );   \\ Yellow - Green = Red
await mSec( 3000 );    \\ 3 Seconds

turnOff( RedLED );     \\ Reset All Off
```

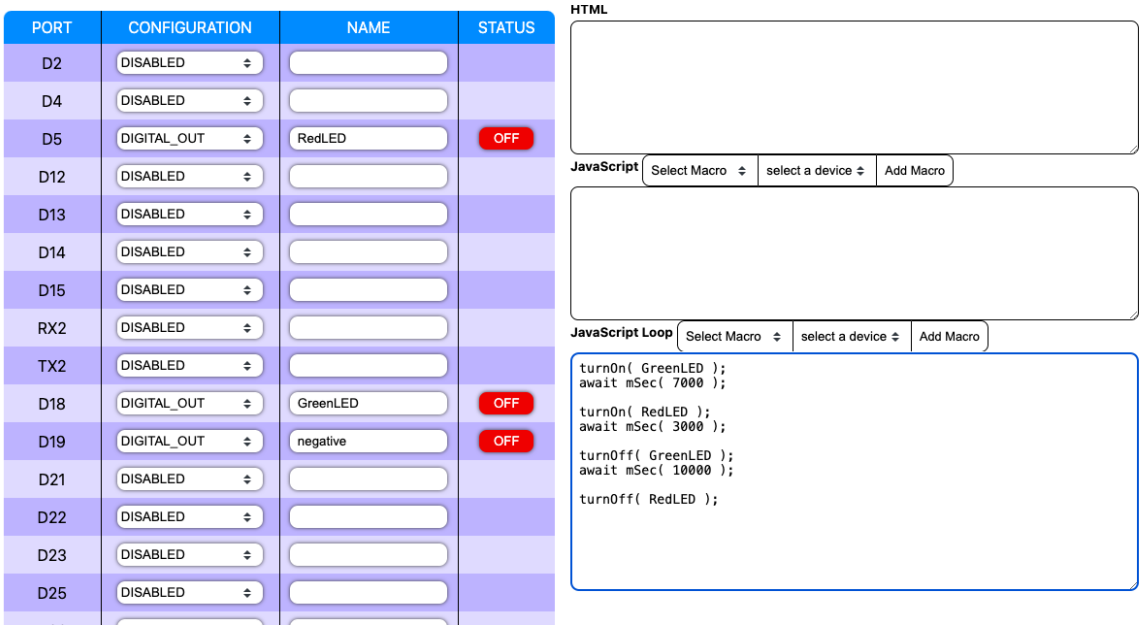


Figure 7: runlinc webpage screenshot (JavaScript Loop)

From this code, the green light will turn on for 7 seconds for “go” and turn off for the next light colour.

Next, the yellow light will turn on for 3 seconds for the “slow down” signal and turn off for the next light colour.

Then the red light will turn on for 10 seconds for the “stop” signal and turn off for the next light colour.

Since this is inside the loop, it will go back to the green LED line and start from there.

HTML

Since we have the java loop working, we can now work on creating an HTML traffic light image. Go to the HTML text box. To start making the image we first must create the board to do it on. To do this, we need to use the tag and declare how big the board will be. Use the following code to do this:

```
<svg height= "1000" width= "100">
```

Now that we have created the board, we can put the first circle on it.

```
<circle cx="50" cy="50" r="40" stroke="black" storke-width="3" fill="red"/>
```

Let’s take a closer look at the code we just wrote. is how we declare what shape we want in HTML. The “cx” and “cy” represent the x and y coordinates used to position the circle. To get how big the circle is we use the “r” declaration which stands for radius. The stroke declarations give us a border around the circle so it is easier to see while “fill” will fill the circle with whatever colour we declare.

As the first circle has been created, we can move on to the second one. Start by copying the line of code that we just did and paste it on a new line. Now if you run the code what do you see? Do you see a new circle? Maybe not?

The issue comes from the x and y positions. We created a new circle but we put it over the top of the first one. To fix this, change the “cy” coordinates from “50” to “150” and change the fill colour to “yellow” like this:

```
<circle cx="50" cy="150" r="40" stroke="black" storke-width="3" fill="yellow"/>
```

Now if you run the code, it will now have the yellow light. Let's add the green light now by pasting the code again and changing the “cy” and fill the colour appropriately:

```
<circle cx="50" cy="250" r="40" stroke="black" storke-width="3" fill="green"/>
```

After we have done the exterior of the traffic lights, let’s make it actually look like a traffic light by adding a pole and a background. To do this we need to go to the top of the code, adding a line of code shown below to where is under the svg line:

```
<rect height="300" width="100"
```

Now this will give us our background, but the corners are going to be rigid, to fix this we can add this extra piece of code to the end of the last line we wrote:

```
<rect height="300" width="100" rx="20" ry="20"/>
```

This will now add rounded corners to the background. Now we can add a pole with this line of code:

```
<rect height="500" width="25" fill="grey" y="300" x="35"/>
```

Finally, we will need to close the tag that we used at the start, using the following tag.

```
</svg>
```

Final HTML Code:

```
<svg height="1000" width="100">
 <rect height="300" width="100" rx="20" ry="20"/>
 <circle cx="50" cy="50" r="40" stroke="black" storke-width="3" fill="red"/>
 <circle cx="50" cy="150" r="40" stroke="black" storke-width="3" fill="yellow"/>
 <circle cx="50" cy="250" r="40" stroke="black" storke-width="3" fill="green"/>
 <rect height="500" width="25" fill="grey" y="300" x="35"/>
</svg>
```

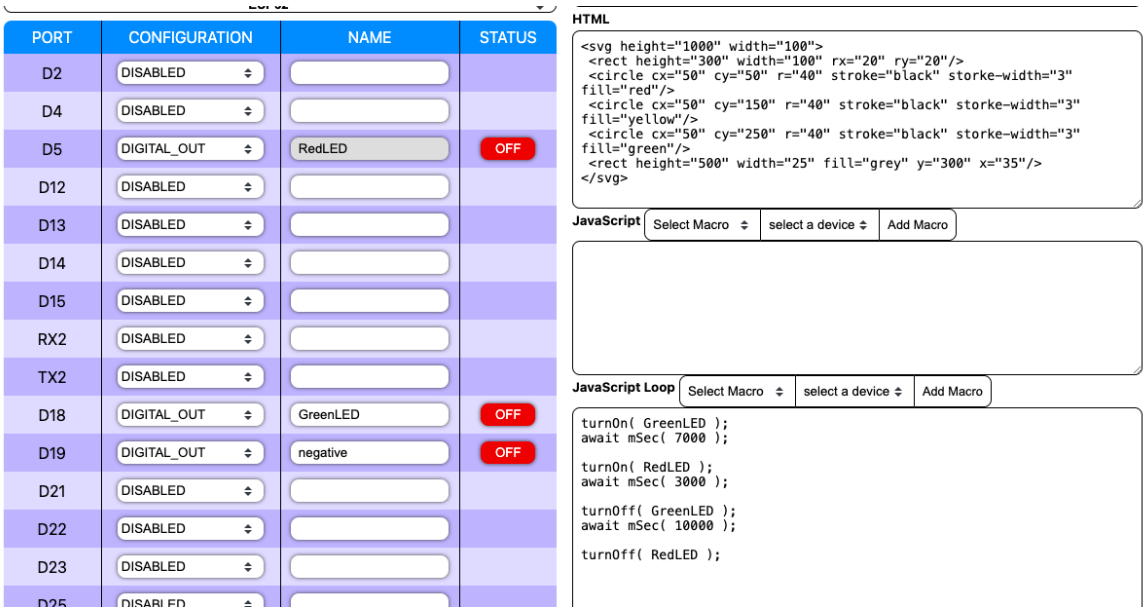


Figure 8: runlinc webpage screenshot (JavaScript Loop and HTML)

Extension – Traffic Light Animation

You may find that the HTML code that you've written is just a static image of a traffic light. It can be boring, right? However! We can add some additional code that will animate the traffic light to light up synchronously with the LED lights. We will use javascript for our animation.

Note: Javascript is a popular coding language that can be used to spice up an HTML page.

When we look at the HTML code, we will find that each circle tag has no way to be uniquely identified. But we can fix this by using the HTML attribute, id. For example, <circle id="Circle1">.

Therefore, edit the HTML code to the same as the following:

```
<svg height="1000" width="100">
<rect height="300" width="100" rx="20" ry="20"/>
<circle id="R" cx="50" cy="50" r="40" stroke="black" stroke-width="3"
fill="FireBrick"/>
<circle id="Y" cx="50" cy="150" r="40" stroke="black" stroke-width="3"
fill="orange"/>
<circle id="G" cx="50" cy="250" r="40" stroke="black" stroke-width="3"
fill="darkgreen"/>
<rect height="300" width="25" fill="grey" y="300" x="35"/>
</svg>
```

The new fill colours are darker colours of the previous colours.

Now, we will add some of the Javascript code that will edit the HTML tags which in turn will animate traffic light.

In JavaScript, we can edit a tag with a unique ID using the following syntax:

```
document.getElementById("<ID>");
```

We will then elongate the syntax which will change the fill colour attribute of the element deciphered by its ID like the following:

```
document.getElementById("G").style.fill="lightgreen";
```

```
turnOn( GreenLED );
document.getElementById("G").style.fill="lightgreen";
await mSec( 7000 );
document.getElementById("G").style.fill="darkgreen";


turnOn( RedLED );
document.getElementById("Y").style.fill="yellow";
await mSec( 3000 );
document.getElementById("Y").style.fill="orange";

turnOff( GreenLED );
document.getElementById("R").style.fill="red";
await mSec( 10000 );
turnOff( RedLED );
document.getElementById("R").style.fill="FireBrick";
```

As a result, by coding the **new JavaScript Loop** to the following:

You will see that the new HTML traffic light image will animate. It will light up the same light as the LED light currently shining.

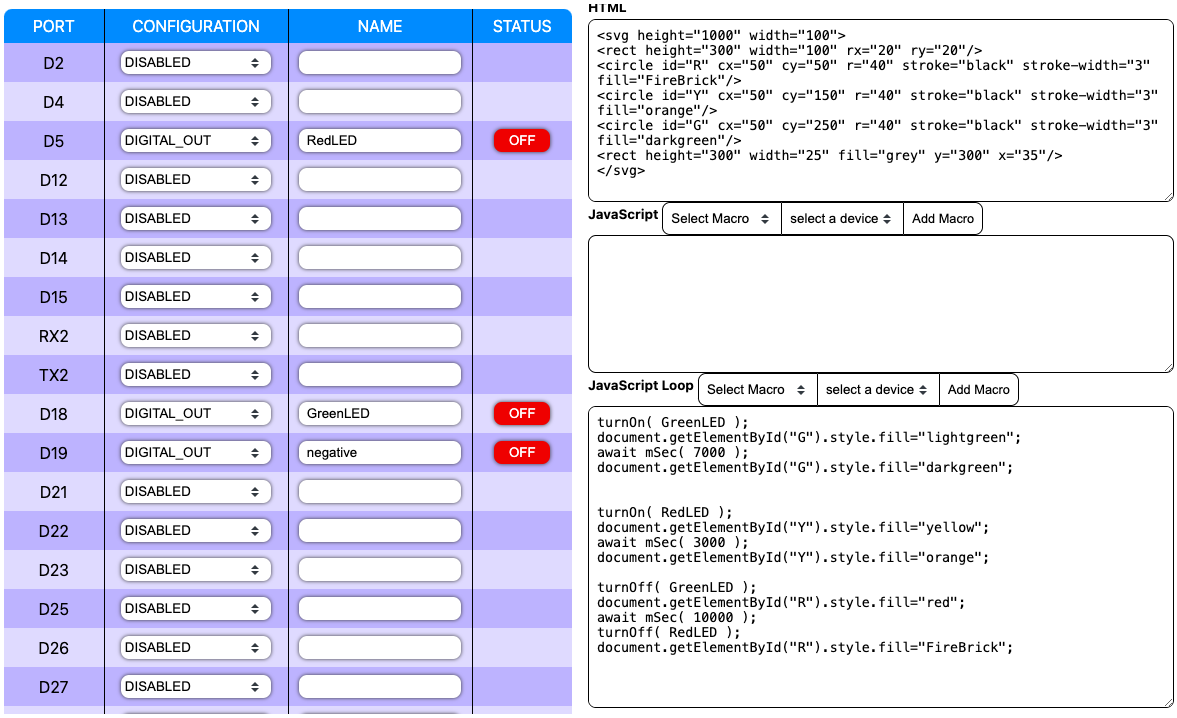


Figure 9: runlinc webpage screenshot (JavaScript Loop and HTML)

Summary

People can use programming to tell microchips what to do. However, sometimes those microchips, in turn, tell people what to do, so it is important to program them correctly. In this project, we learned that both the correct sequence and correct timing is important not only for traffic lights but for making eye-catching displays.