runlinc Project 1 B0: Light Timing Control (E32W Version)

Contents

Introduction

Problem

How do we control the devices that are connected to the STEMSEL board with code? How do we connect the devices and how do we write the code? How do you use the code to control a light to make it blink? In this project, we will go through a process that tells you about a simple code that will make an LED strobe light, to let you familiar with the runlinc STEMSEL board, and get to know about a STEMSEL microcontroller-controlled system's connection and coding.

Background

To work with the microcontroller, we need to know how to let it control first.

Microcontrollers can be used for many applications, by executing pre-loaded codes and using the inputs and outputs that are connected to the board, to sense the signals and output signals.

By executing the code, the microcontroller will deal with the input and give a certain output. So, the code is what the microcontroller will follow, and what it will do.

Codes are the things we tell the microcontroller, to let it do what we want it to do, by using a special language the microcontroller will understand. The language is the code. We need to load the code onto the microcontroller by typing it and loading it onto the microcontroller on the control surface.

Ideas

So how can we use the STEMSEL board? We can start with some simple but meaningful projects, like making a light connected to the board behave in a certain way. We can start

with making a light that will turn on for 1 second and then stay off for 1 second and keep doing this while the board is on and the program is running, an LED strobe light.

Plan

We have a STMSEL board and some LED lights (a light built on the board and a light in the package with the board), we can first make a program to let the board control the on-board LED for a test, then connect the other LED to the board and let the board turn the LED on and off. Below is the input/output block diagram for the connection.

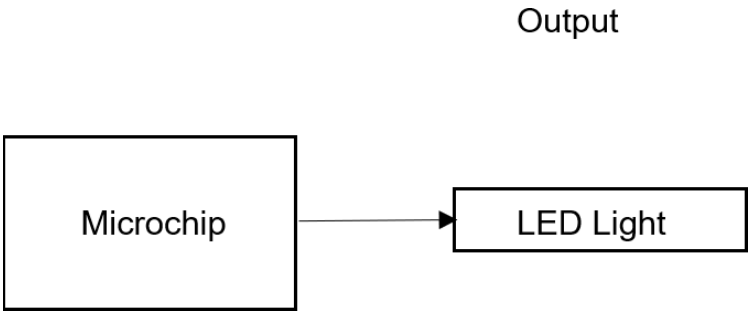


Figure 1: Block diagram of Microchip outputs

runlinc Background

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

Part A: Design the Test 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 D2 name it Board_LED_Light and set it as DIGITAL_OUT.

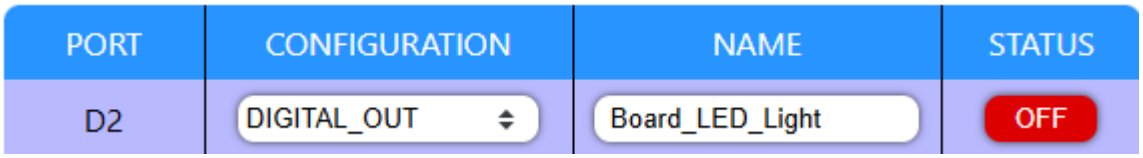


Figure 2: I/O configurations connections

Maybe you've already noticed that the **name for the ports is case-sensitive and cannot include space**. Case-sensitive here means the program will save the names as variables and recognise the Uppercase letters as uppercase letters and not treat every letter as lowercase letters.
e.g. if Jack writes "board_LED_Light" in the code to try to turn the light on D2 on and uses other codes right, the light will not be turned on as "board_LED_Light" is different from "Board_LED_Light", and the program will never turn the "Board_LED_Light" on. One simple different case can cause the entire program not to run correctly! So be careful about the letters' case.

Also, these ports correspond to the ports on the physical board with the same number but different letters. e.g. **D5 on the web page corresponds to io5 on the board, D19 on the web page corresponds to io19 on the board, etc**.

Here, you can click on the red "OFF" button to turn the port on to check it. Keep in mind to always do this to check connections before the programming and when problems happen. After you finish, click on the green "ON" button to turn it off.

Part B: Program the Test Circuit

Now we can start to program the code of the blinking LED. To do this, JavaScript Loop is needed. The code in the JavaScript Loop window will be running from its beginning, again and again, when the program is running.

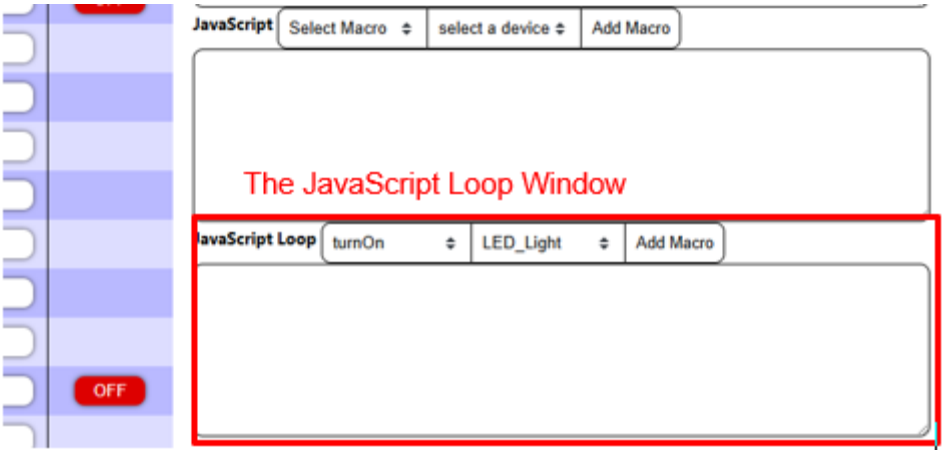


Figure 3: JavaScript Loop Window webpage screenshot

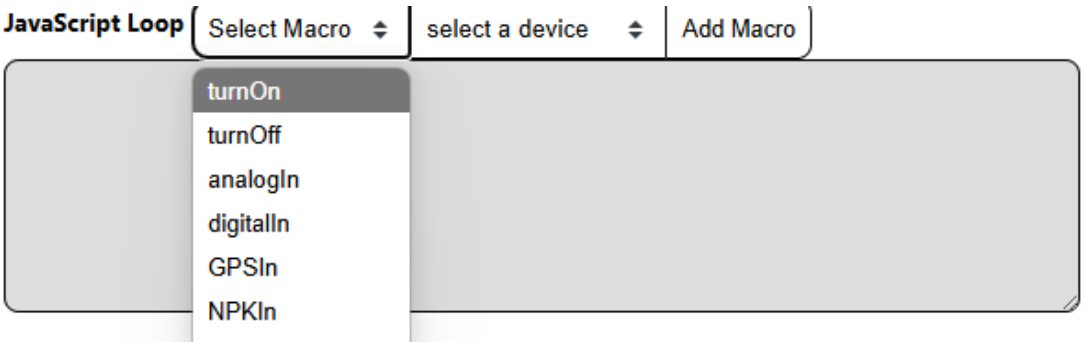
We want the board to act 4 steps. First, turn on the LED; second, hold for 1 second; third, turn off the LED; at last, hold for 1 second. Then it will repeat the 4 steps.

The first step is turning the LED on. To do this we need to write a line of code. We have Macro buttons to help you build the code, which makes things easier.

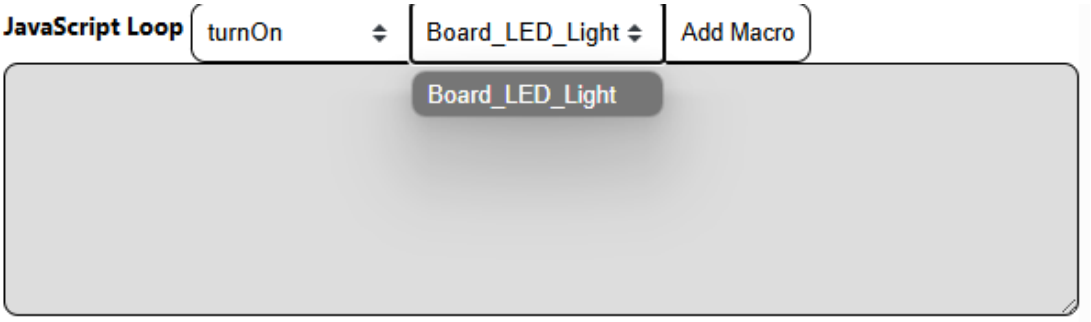
Use the add macro buttons on top of the JavaScript Loop window.

Figure 4: drop-down menus and the “Add Macro” button

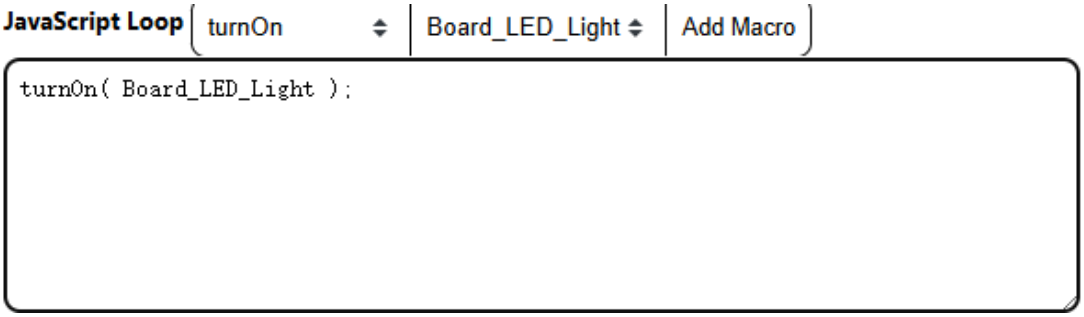
First, select “turnOn” from the “Select Macro”,



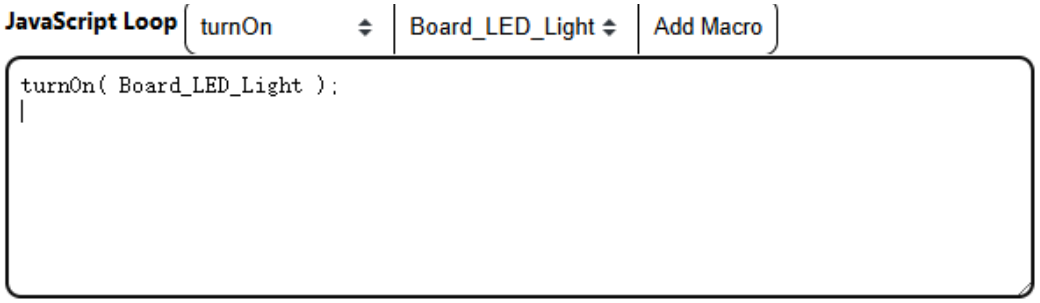
Then select the port name we named from the device,



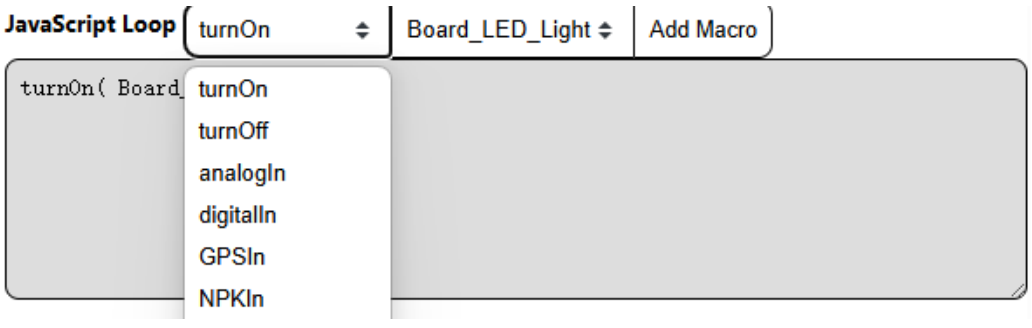
Then click on the “Add Macro” button to add the macro to the code.



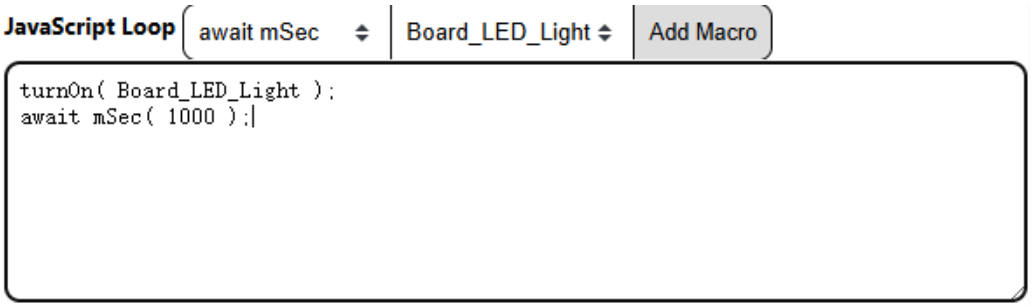
Then, we need to make it wait for 1 second. To do this, we need another line of code. First, in the code window, move the cursor to the place you want to add this macro. We better add the macro in a new line, to give it an overall organized look. Click in the code window to show the cursor, if it's on the right of the first line of code, press enter to move the cursor to the next line.



Then, select “await mSec” in the select macro button,



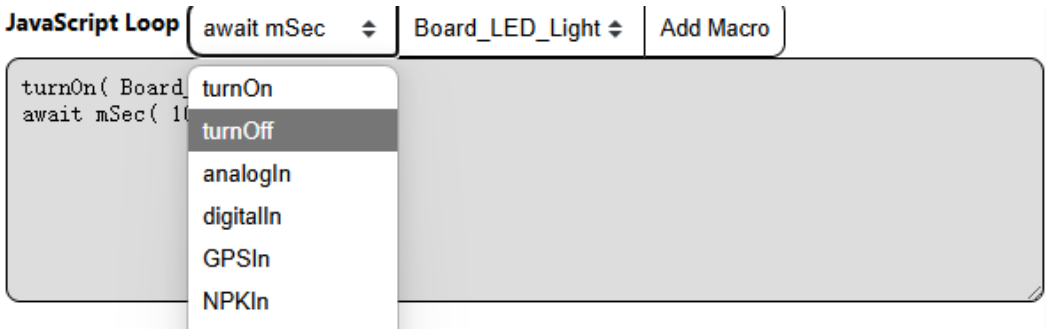
Then click on “Add Macro” button to add the macro.



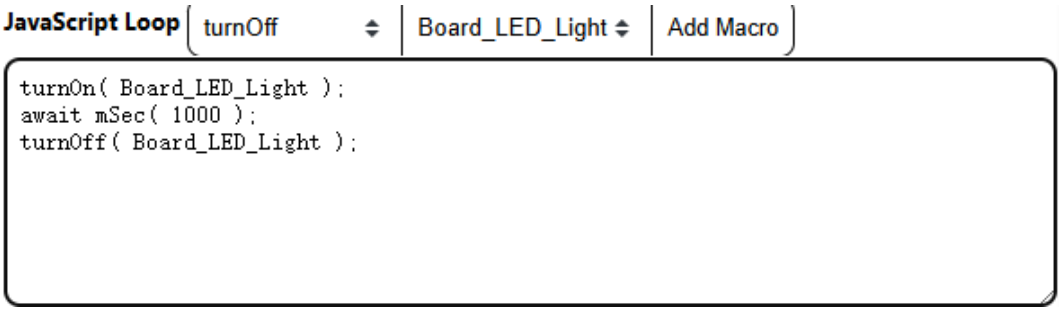
Now you’ve added another line of code, this will make the microcontroller wait for 1 second. Be aware that the unit here is in milliseconds, and 1 second is equal to 1000 milliseconds. You can enter other numbers to change the time it waits. e.g. 500 makes it to wait for 0.5 second, etc.

After the wait, we turn the light off.

First, move the cursor to the right place. Then, select “turnOff” in the select macro button,



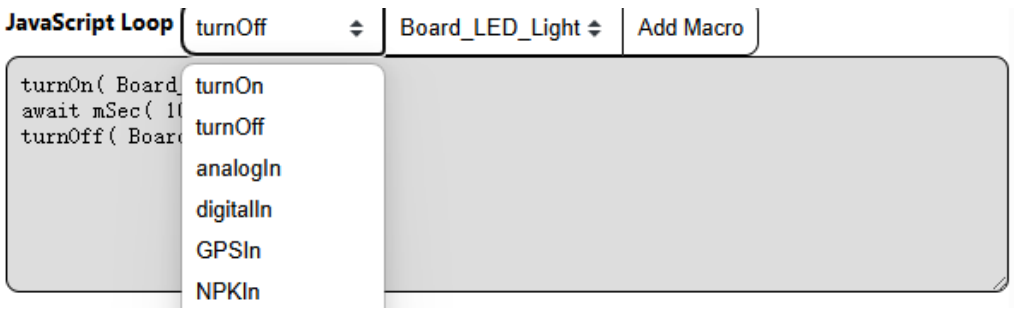
Then click on the “Add Macro” button to add the macro to the code.



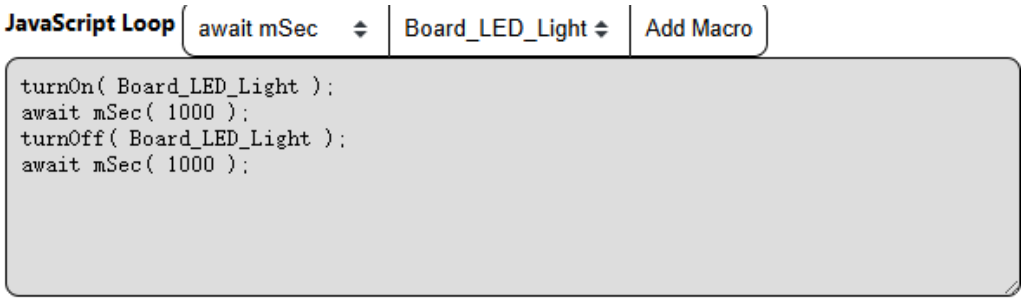
This will stop turning on to the corresponding port, thus turning the LED light off when the program is running.

After this, we will let it wait for 1 second again then start from the beginning to turn the light on. We need to let it wait for 1 second, because if not, the light will be turned on immediately after being turned off, and we cannot notice the turn-off phase.

First, move the cursor to the right place, then select “await mSec” in the select macro,



Then click on the “Add Macro” button to add this macro.



The final code should look like this:

```
turnOn( Board_LED_Light );
await mSec( 1000 );
turnOff( Board_LED_Light );
await mSec( 1000 );
```

The final runlinc webpage should look like this:

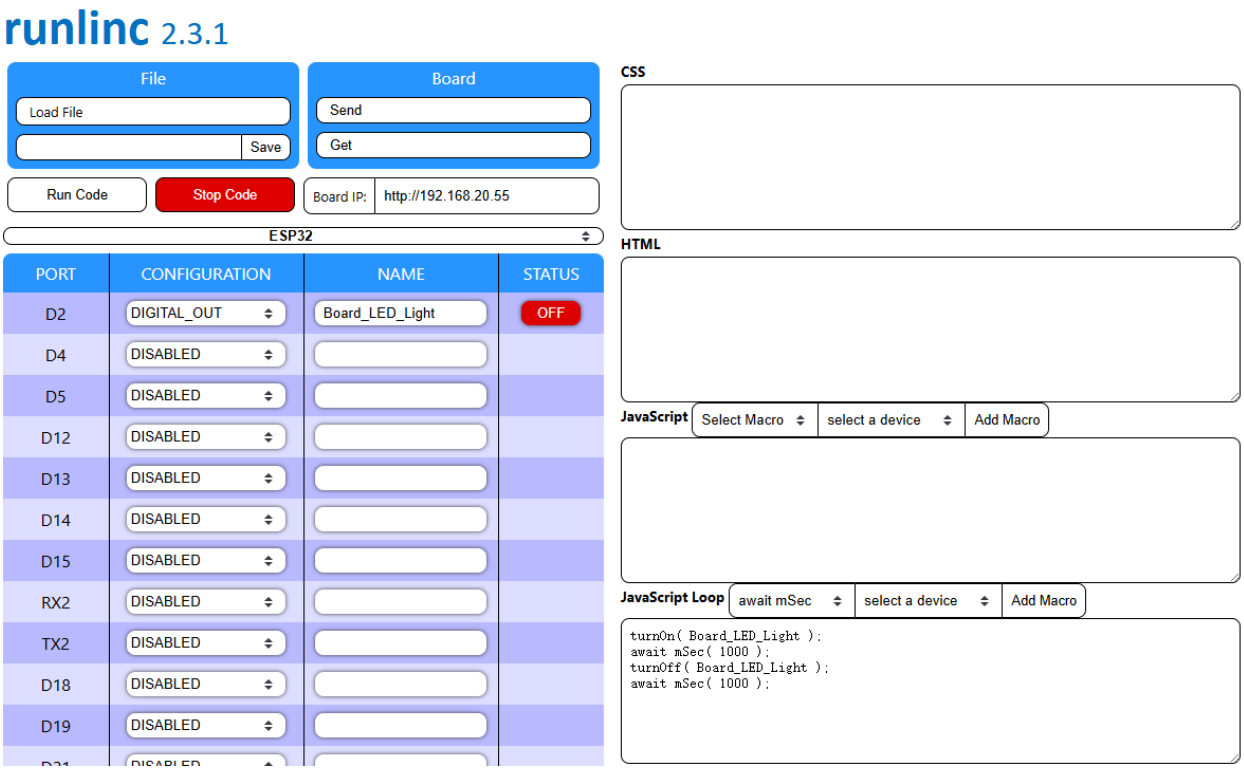


Figure 5: runlinc webpage screenshot

You can change the code, by manually typing code or deleting them. You can also enter the code by typing the exact same codes in, it will work, we will do this in the next section.

Now we can go on and test the system we built.

Part C: The Working test Project

Run the program by pressing “Run Code” button on the webpage;

Figure 6: runlinc webpage “Run Code”

When the project is running, there will be a blue light on the board blinking at 1-second intervals:

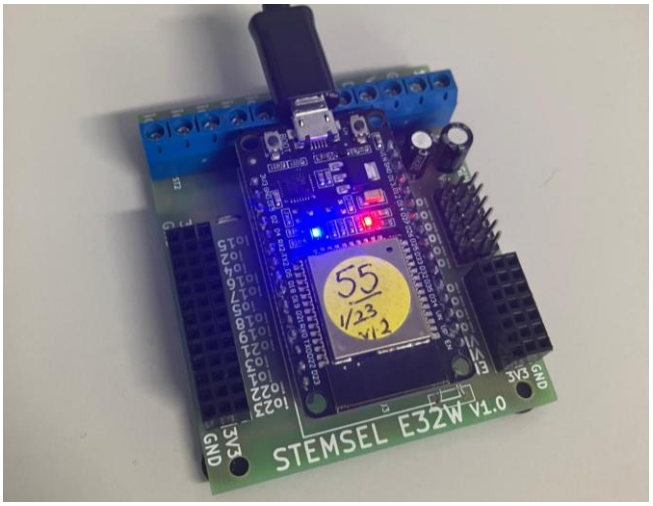


Figure 7: blinking blue LED

If your test program is working, we can go to the next part to build an LED circuit with the I/O ports. If it’s not working, you can go back to parts A to C to make sure that you’ve followed each step correctly.

Part D: Design the LED Circuit on runlinc

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

For port D5 name it LED_Light and set it as DIGITAL_OUT.

For port D19 set it as DIGITAL_OUT (used as the negative pin of LED – no name needed).

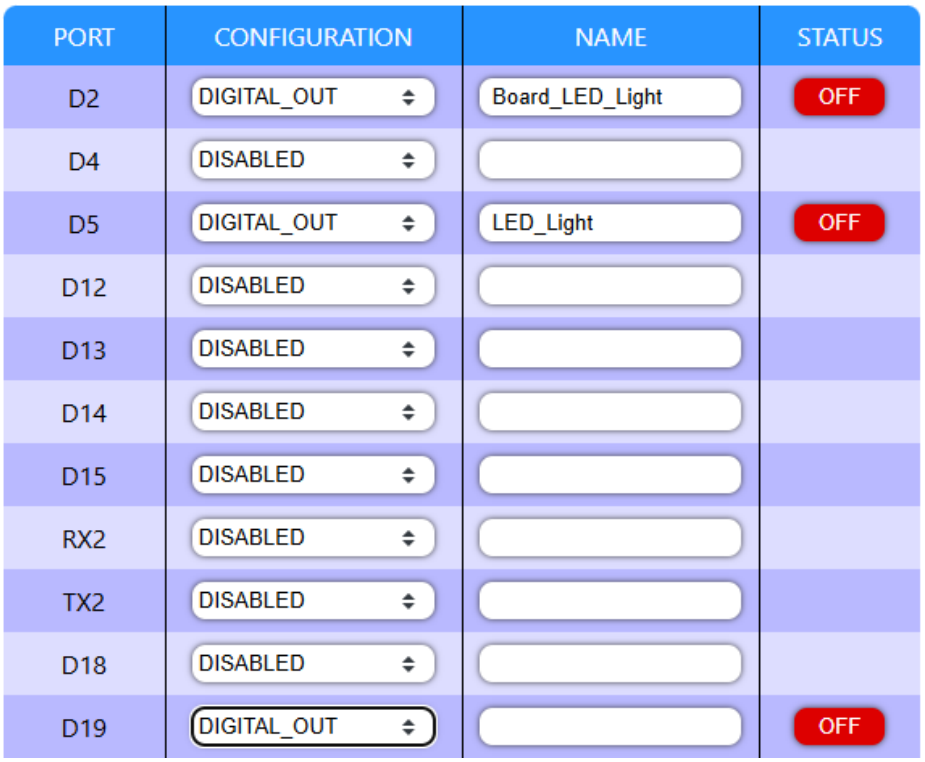


Figure 8: I/O configurations connections

Part E: Build the Circuit

Use the STEMSEL E32W 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).

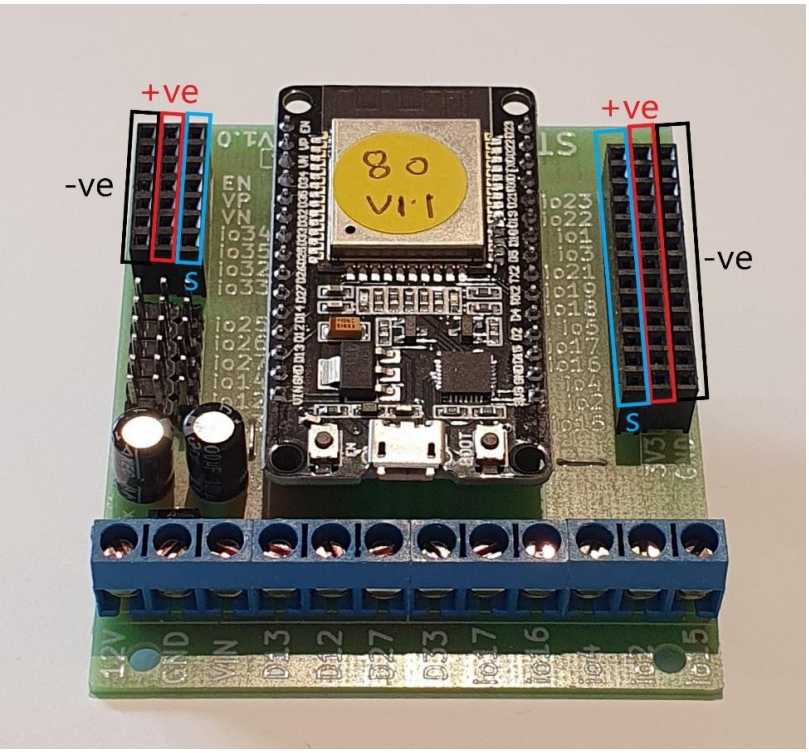


Figure 9: Negative, Positive and Signal port on the E32W board

There is one I/O part we are using for this project, a 3-pin LED light, its respective pins are shown in the figures below. In some kits, due to different manufacturers, the green pin and the red pin can be inverted. But no worries, we will test it after we build the circuit.

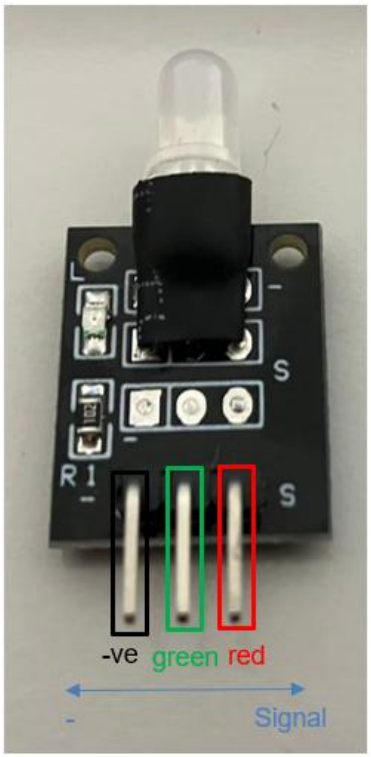


Figure 10: I/O parts with negative and signal pins indicated

Wiring instructions

a.) Plug in the LED to signal ports io5, io18 and io19 on the E32W board with the “Red” pin on the light that goes into io5, corresponding to the port D5 we’ve set as Digital_Out and named as “LED_Light”. We did not configure port D18 on the webpage for this connection because for this 2-color LED light, the middle pin which is connected to io18 is for lighting up the green colour, and we don’t need it for this project, so we don’t configure it and thus keep it off.

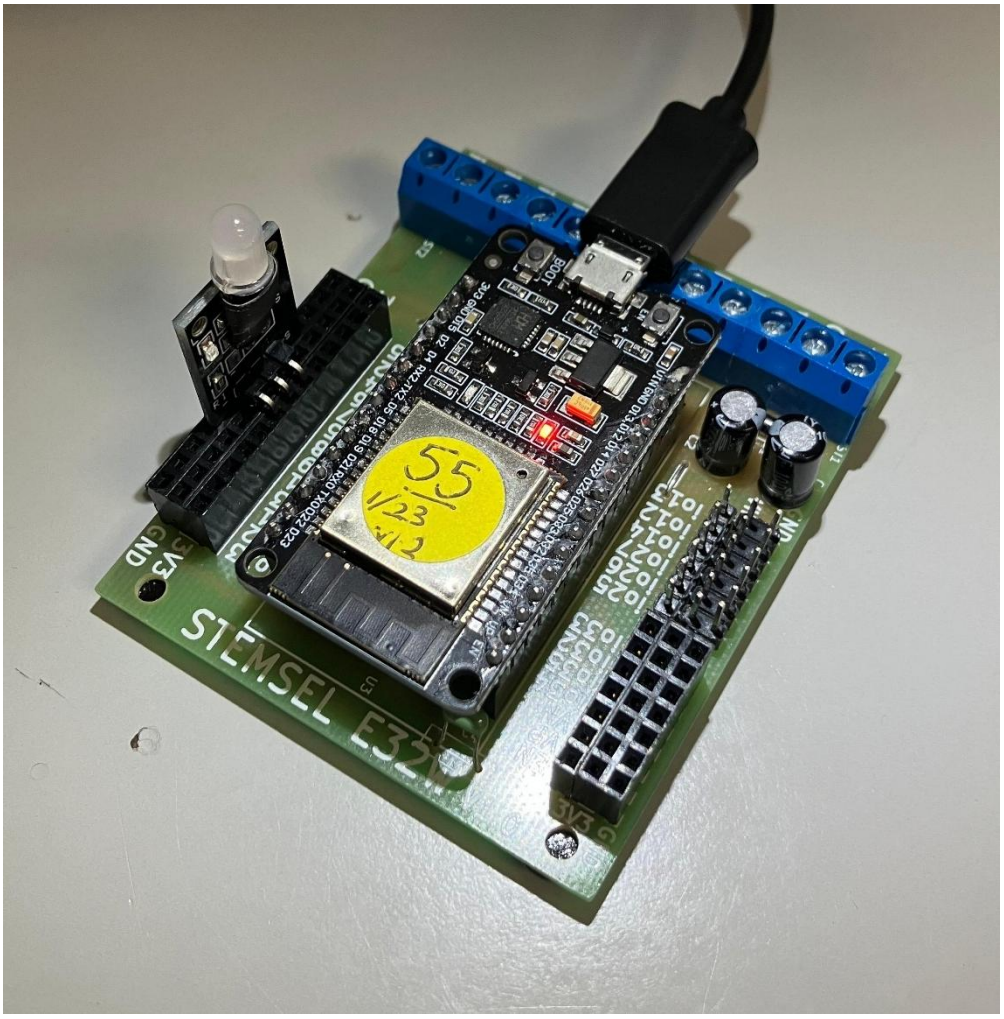


Figure 11: Circuit board connection with I/O parts (side view)

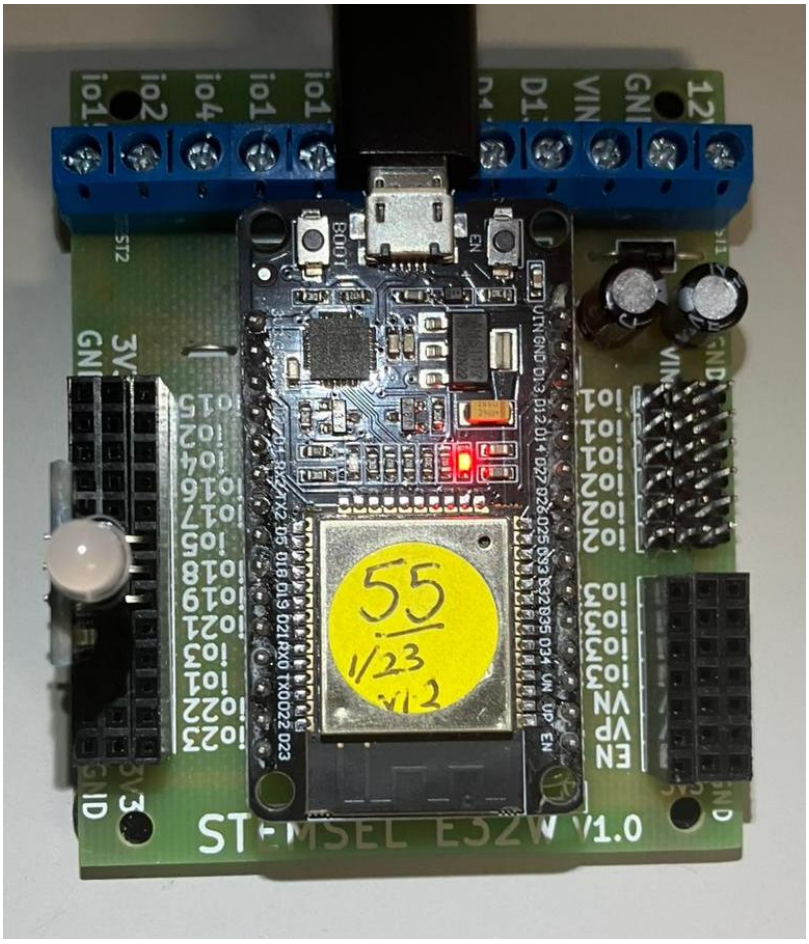


Figure 12: Circuit board connection with I/O parts (top view)

Part F: Program the Circuit

Now, we can start programming the functions of the blinking light. To do this, we can simply change some words in the previous code.

We are turning on the port named “LED_Light”, instead of “Board_LED_Light”. So, we just change all the “Board_LED_Light” in the code into “LED_Light”.

The final code should look like this:

```
turnOn( LED_Light );
await mSec( 1000 );
turnOff( LED_Light );
await mSec( 1000 );
```

The final runlinc webpage should look like this:

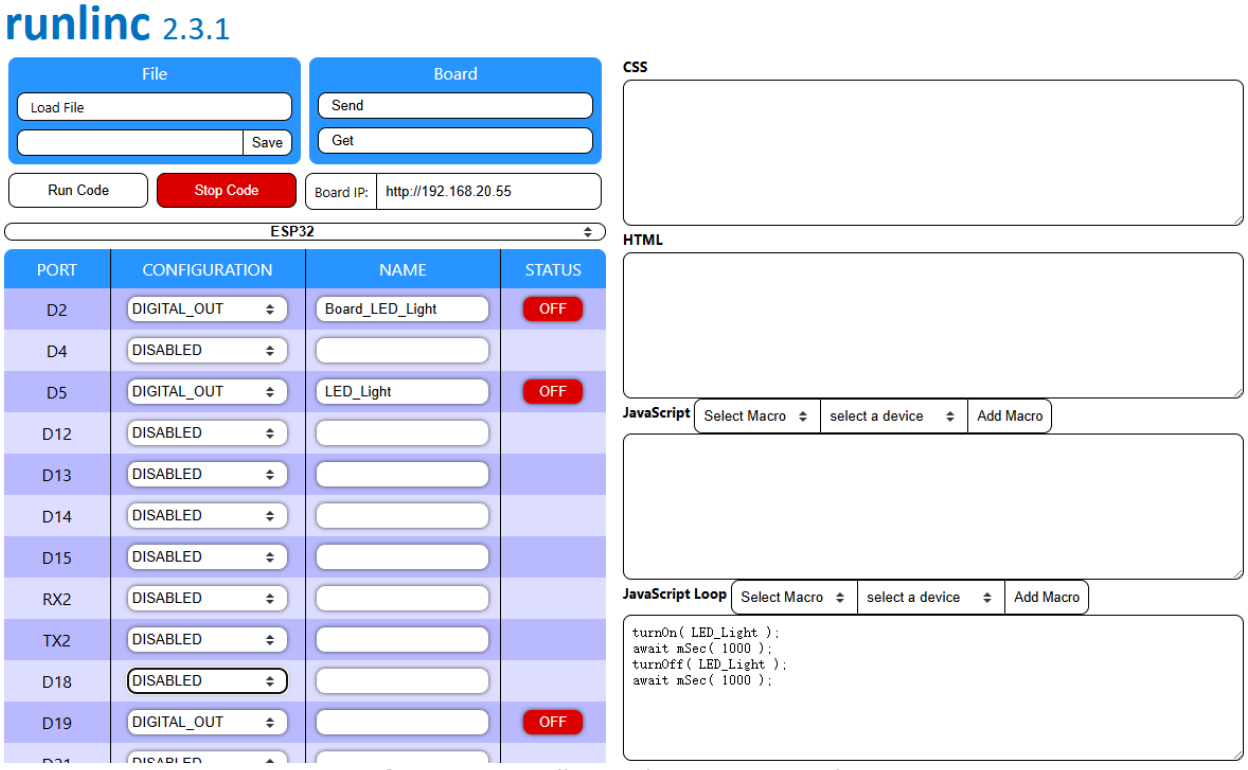


Figure 13: runlinc webpage screenshot

Part G: The Working Project

If you successfully built the circuit and the program correctly, it should act as what this part describes:

Run the program by pressing the “Run Code” button on the webpage;

Figure 14: runlinc webpage “Run Code”

you will see the LED light blinking red with a 1-second interval.

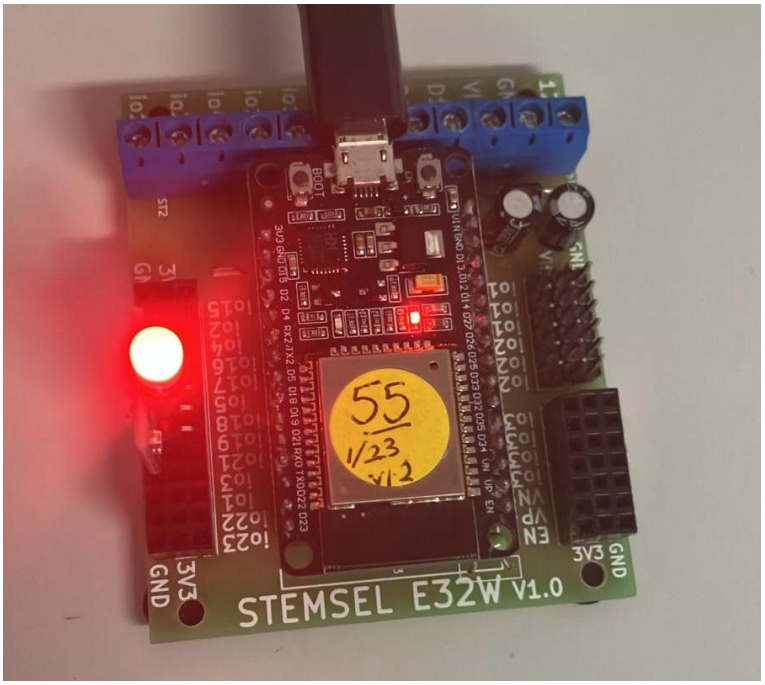


Figure 15: Turning on the red LED

Note: If you see a light blinking green, change the name and setting from port D5 to port D18:

Set the port D18 on the control web page as “DIGITAL_OUT”
Copy the name “LED_Light” on port D5
Paste it to port D18
Delete the port name on D5
Set D5 as “DISABLED”

and you will correct it.

Summary

If we want to turn on a device that is connected to the port of the STEMSEL board, we can do it by connecting it to a port and turn the port to “Digital_Out” and give it a name, then use the turnOn(port_name) code. The code for turning off the device is similar. We can use await mSec (time_in_millisecond) to let the program wait for a certain time in milliseconds.

We can let the light have different behaviour by changing these codes, and if we learn more code, we can use the STEMSEL board to do even more things. Code is what the control board will follow, what it will do.

Congratulations! Now you have entered the world of runlinc programming. It uses simple codes and it helps to make your coding way simpler and faster.