Runlinc Beginners Project 1: Alarm System (E32 Version)

Contents

Introduction

Problem

- How can we use microchips to protect our valuables?
- How could a chip know if there is a thief and what can it do when it detects one?
- How can runlinc communicate with the microchip to implement these commands?

Background

Have you heard of a microchip before? Do you know what microchip means? Micro means very small, and a chip is a small piece, like a potato chip or a wood chip. A microchip is actually a circuit that has been fitted into a tiny piece of silicon, and people can use that tiny silicon circuit to form a tiny computer which can store a list of instructions to make the microchip do many different things. Microchips are all around, in everything from computers to cars, credit cards to toys. One thing that microchips are very useful for is alarm systems, since unlike a person the microchip can't get bored of watching the same thing for hours on end.

Alarm Systems are used to protect a wide variety of valuables, from a car to household belongings. Alarms use sensors which can be a simple switch or more complex heat, light and motion sensors. A key component of an alarm system is a microchip to gather sensor information and decide when to activate the alarm.

Ideas

Look at the E32W controller board. Can you see any inputs, i.e. something that we can touch or change to tell the microchip something? What about an output, i.e. something the microchip can change to tell us something? What kind of inputs and outputs are normally on an alarm system? What inputs and outputs can we use on our alarm system?

Plan

We have a 3-pin button and a 3-pin buzzer in our kit, therefore we can use to create a situation where if the button is pressed, the buzzer will sound. If the button is released, the buzzer will turn off. This will simulate a force sensor, where if the button is hidden beneath the floor and an intruder step on it, therefore the alarm will be heard and scare away the intruder.

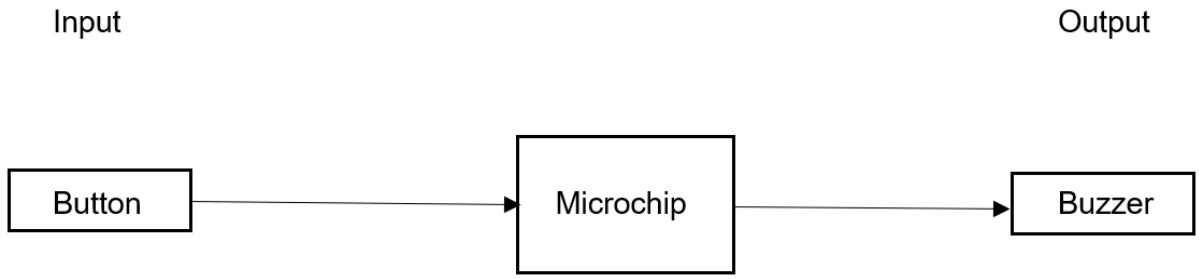


Figure 1: Block diagram of Microchip outputs

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 D23 name it Buzzer and set it as DIGITAL_OUT.
For port D33 name it Button and set it as DIGITAL_IN.

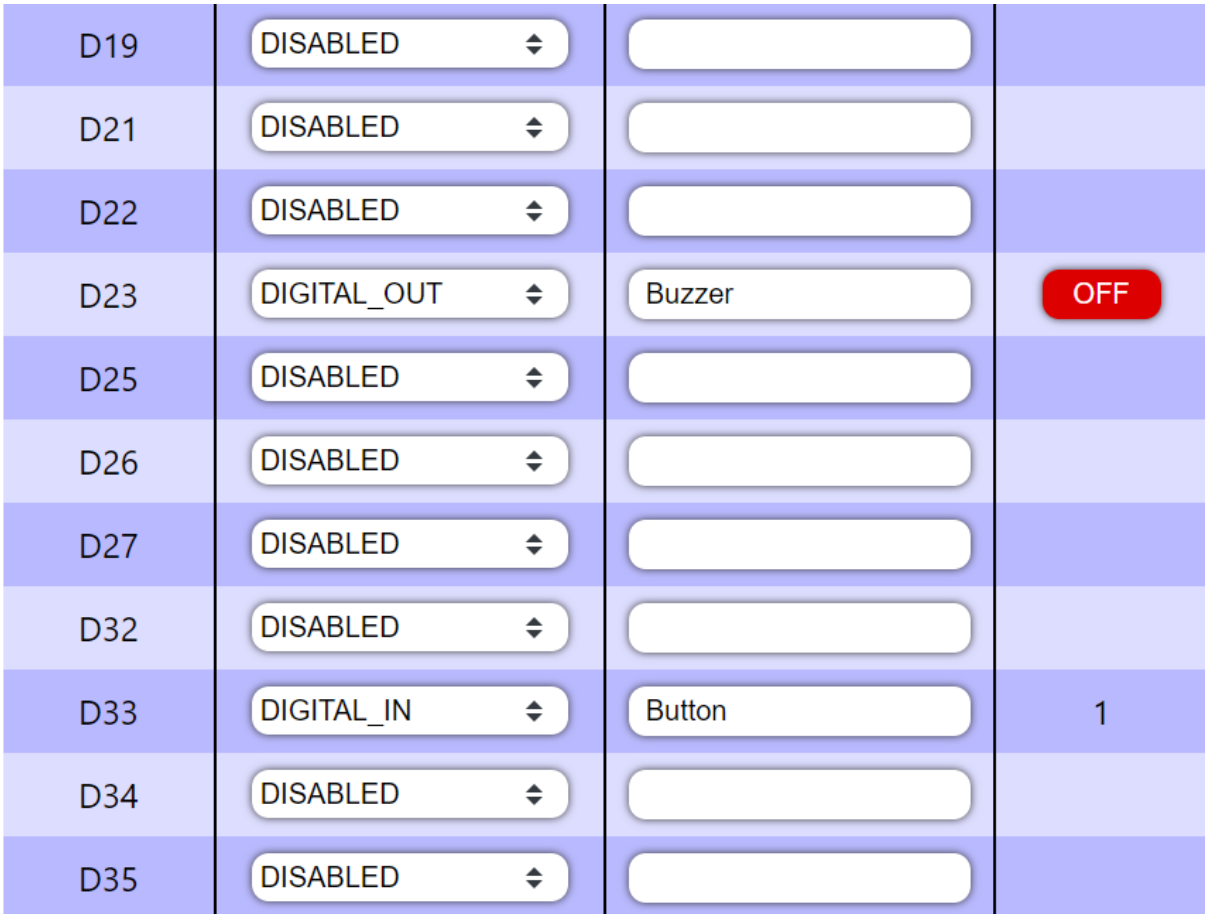


Figure 2: I/O configurations connections

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).

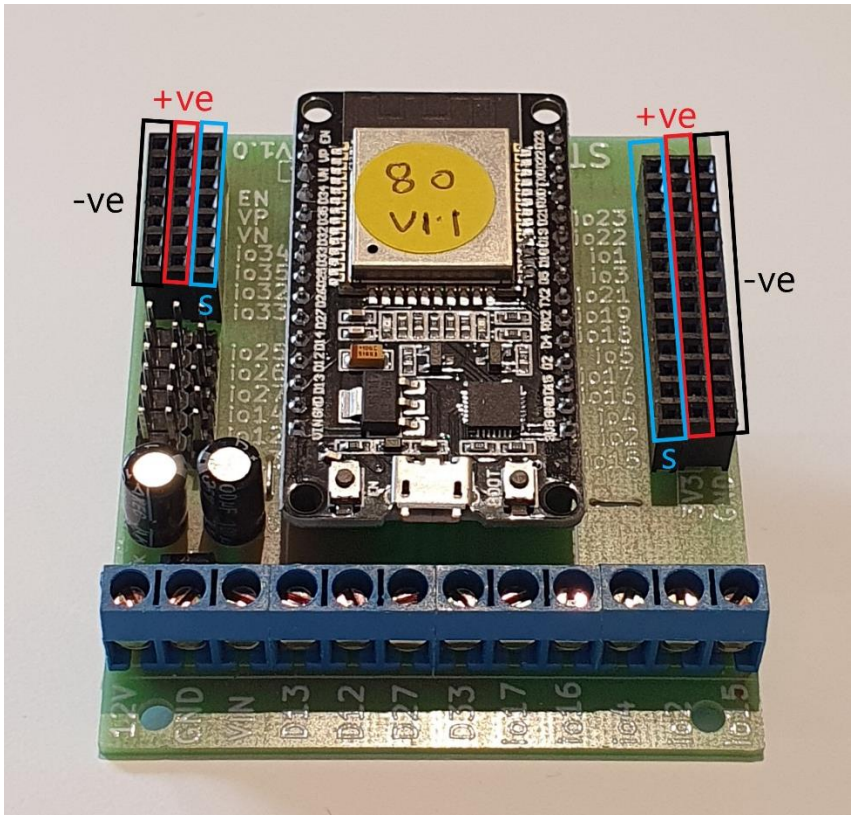


Figure 3: Negative, Positive and Signal port on the E32 board

There are two I/O parts we are using for this project, a 3-pin Buzzer and a Button, their respective pins are shown in the figure below.

Note: The correct Buzzer has a + sign on the top near the pins, the Buzzer should make continuous noise when turned on.

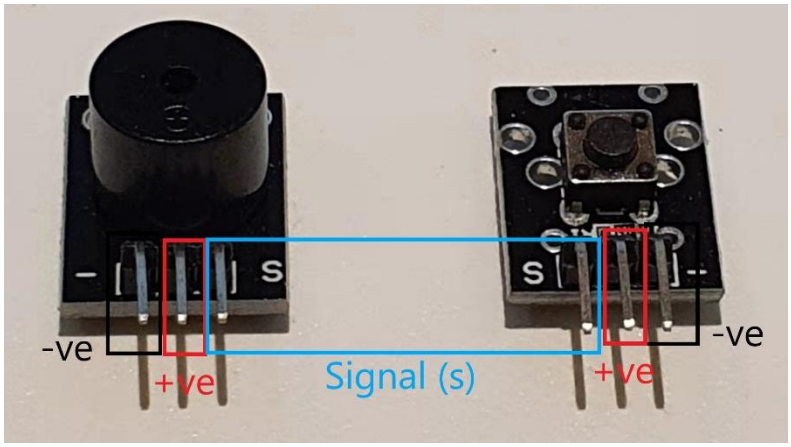


Figure 4: I/O parts with negative, positive and signal pins indicated

Wiring instructions

a.) Plug in the Buzzer to io23 on the E32 board.

b.) Plug in the Button to io33 on the E32 board.

c.) Make sure all (-ve) pins are on the GND (outer) side of the I/O ports.

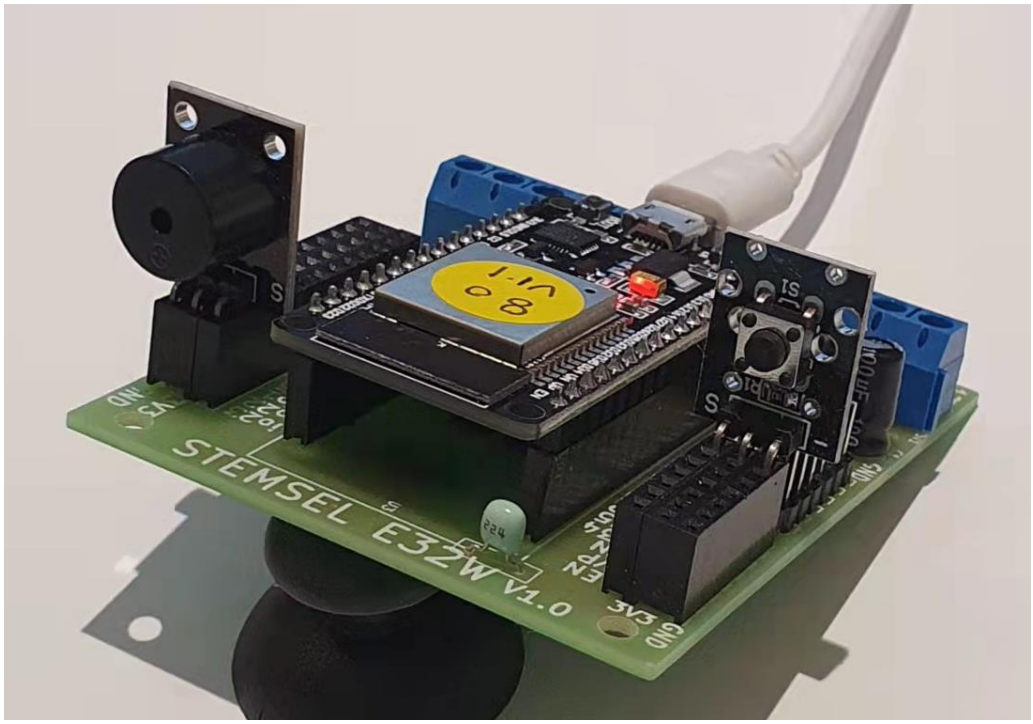


Figure 5: Circuit board connection with I/O parts (side view)

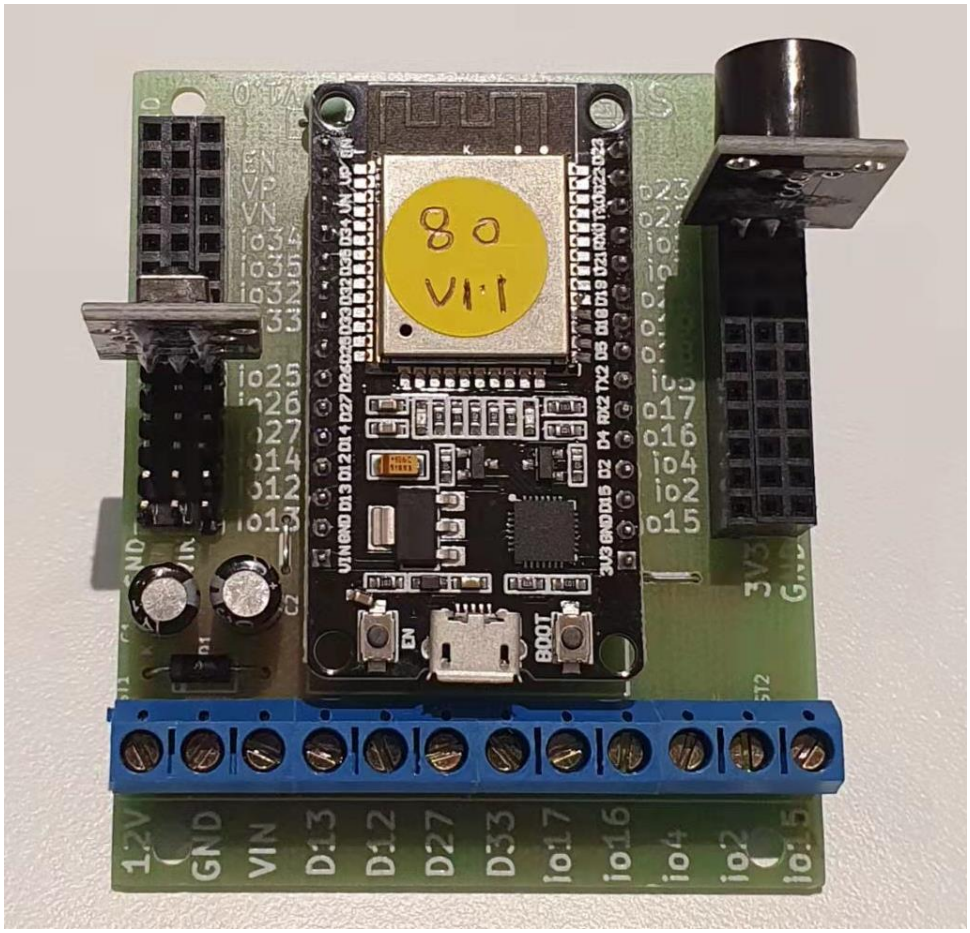
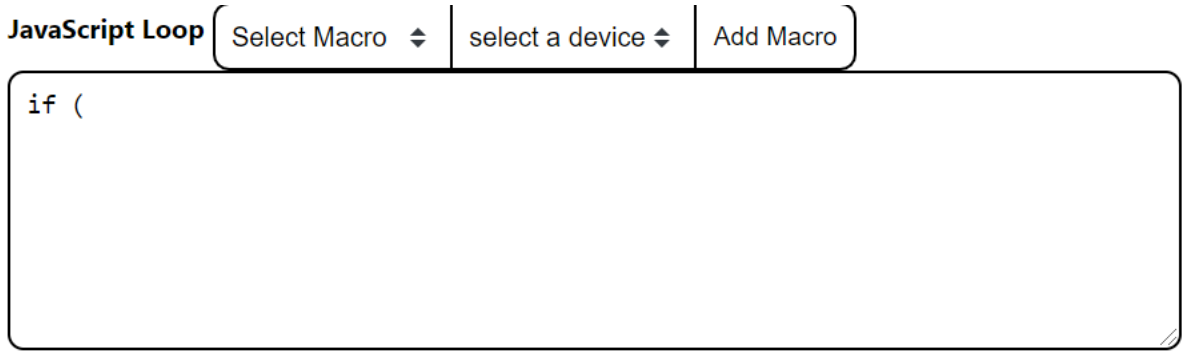


Figure 6: Circuit board connection with I/O parts (top view)

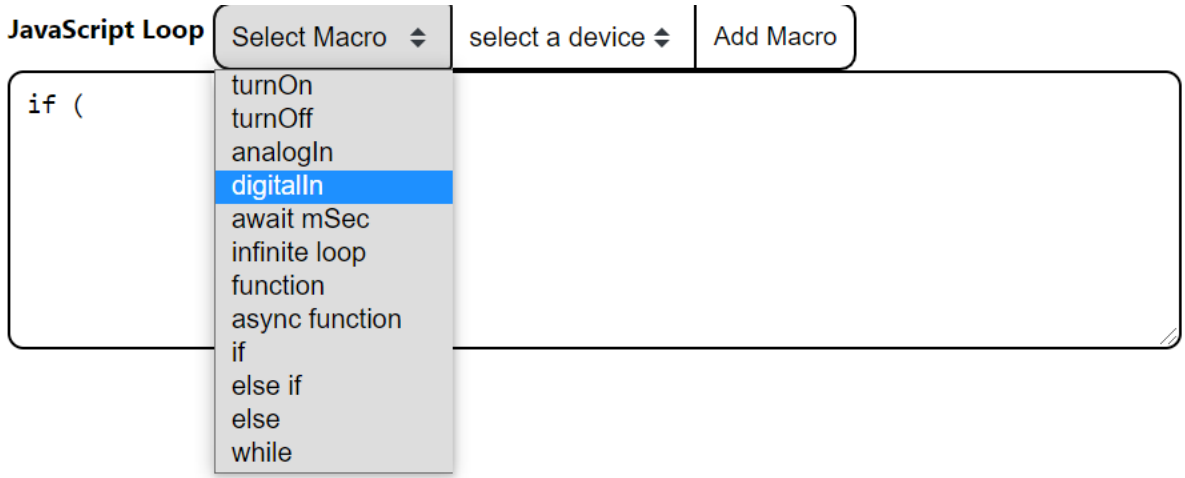
Part C: Program the Circuit

This project we only need to program in the JavaScript Loop box.

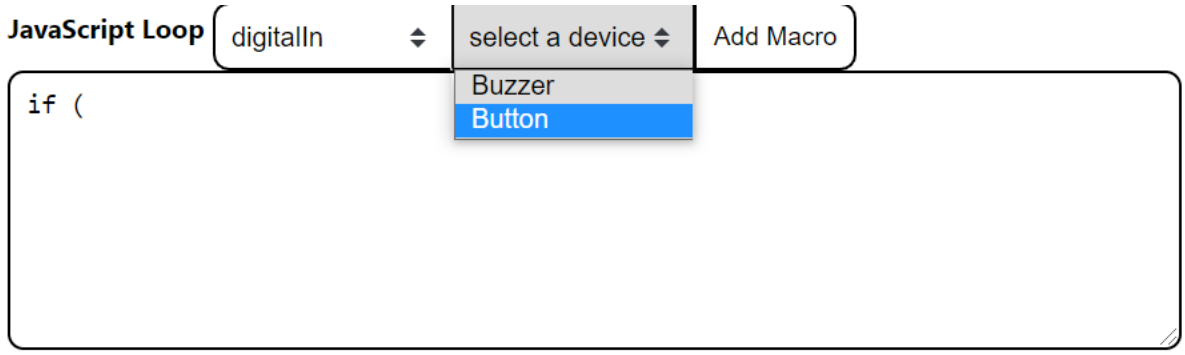
For **JavaScript Loop** box type the following code:
First, we need to declare an if statement. To do this, type in the following code in the first row:



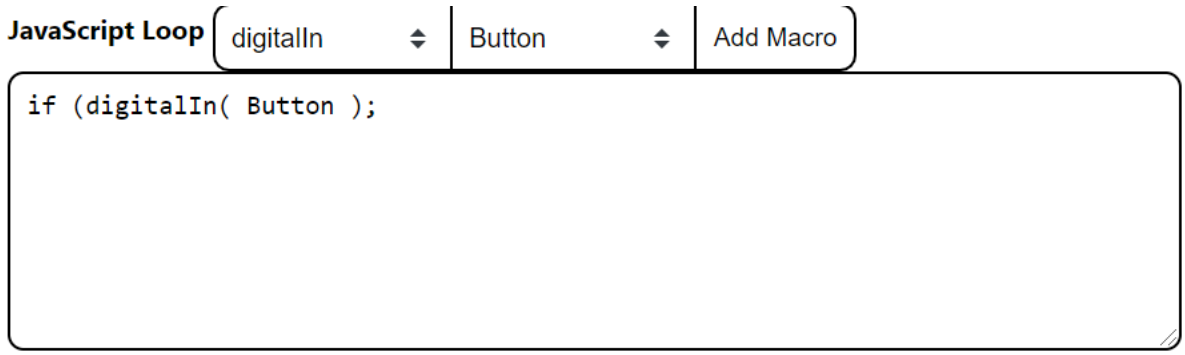
Next, we need to add a macro for the button. To do this, go to the select macro button and choose digitalIn.



Afterwards, go to the select device button and choose Button.

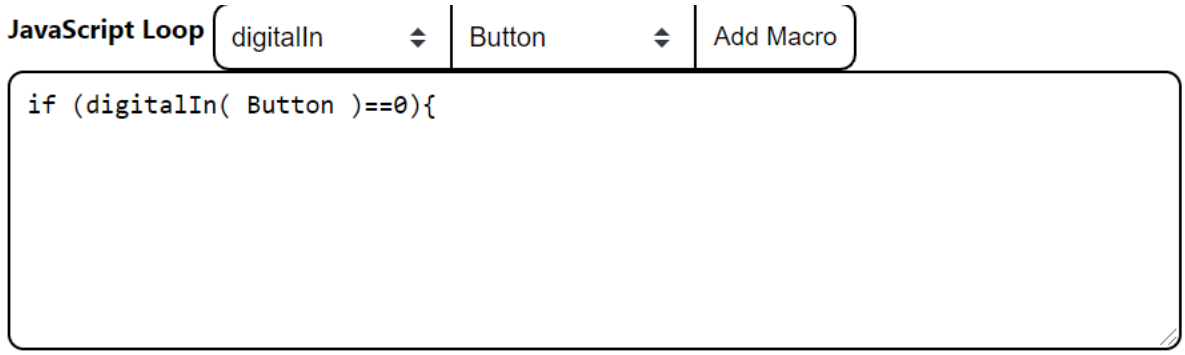


Click Add Macro to add the macro to the code.

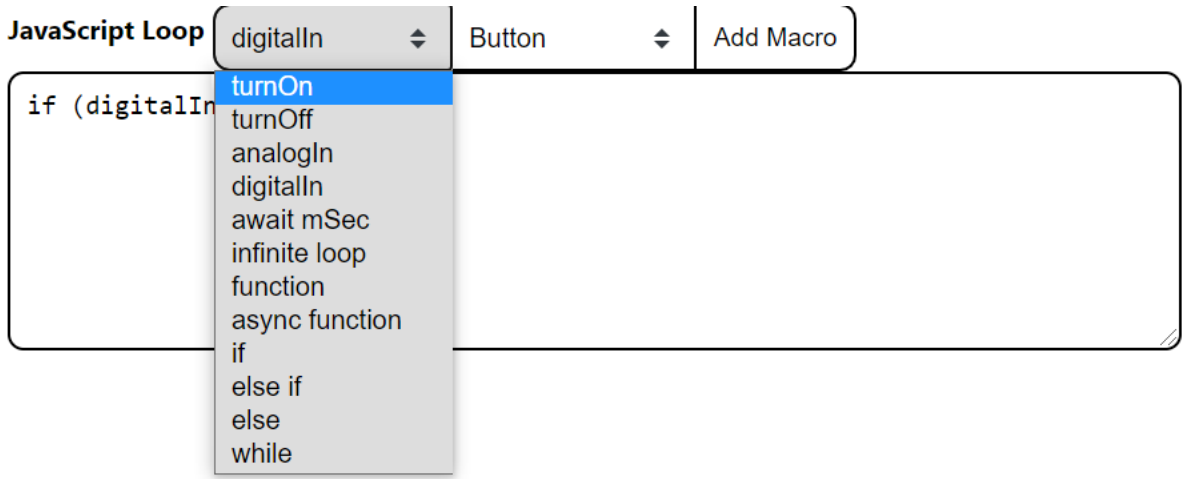


To complete the if statement, delete the semicolon and replace it with the following code snippet: ==0){

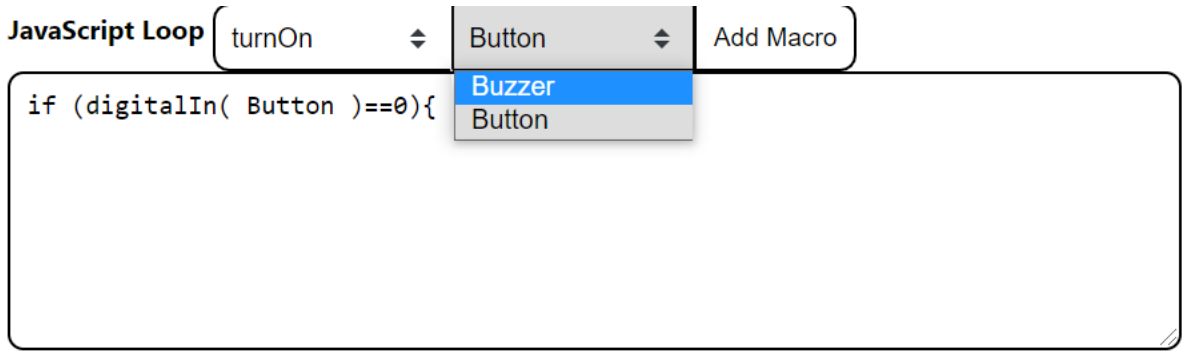
This makes sure that the microchip will determine if the button is pressed or not all the time. The **default value for the button is 1**, therefore by setting it to 0, the Buzzer will remain **turned off** when we run our program. And when the Button is pressed, the **value will turn to 0**, therefore **turning on our Buzzer**.



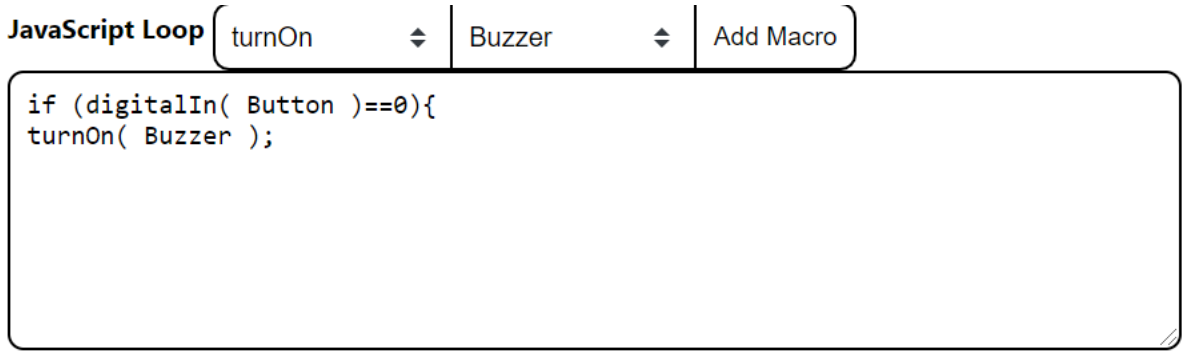
At this stage, we need to add an output which is the buzzer, by using a macro. Go to the select macro button and choose turnOn.



Now, select device and choose Buzzer.



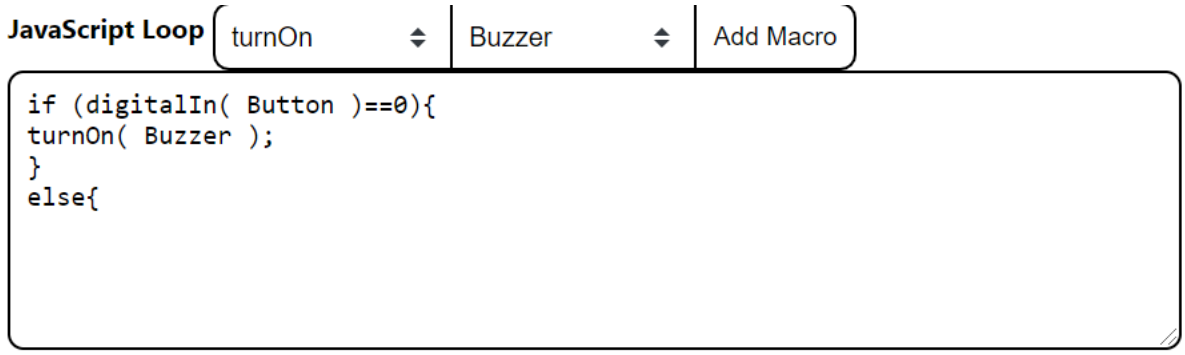
Now add the Macro.



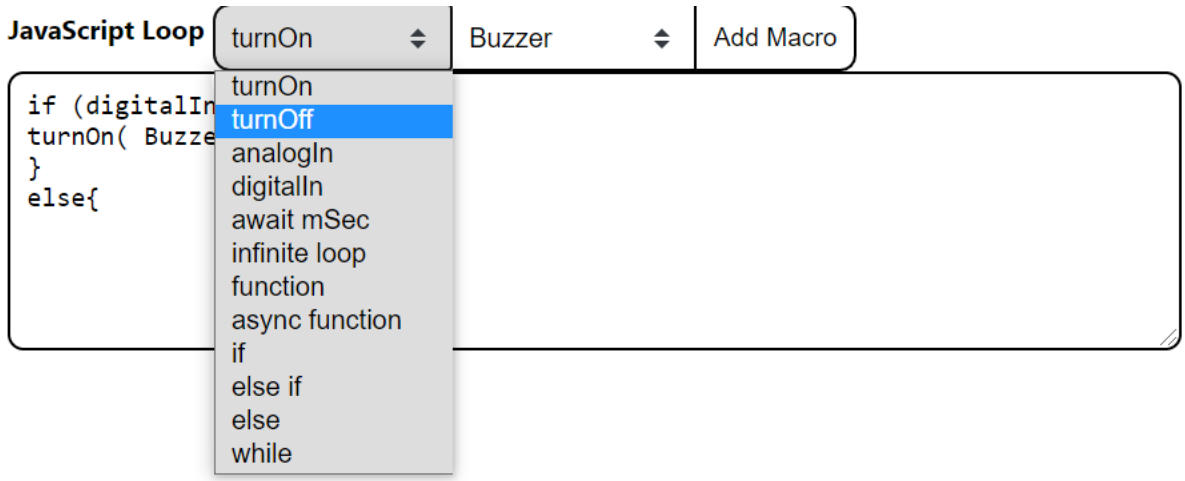
Type in the following code snippet in next line:

}

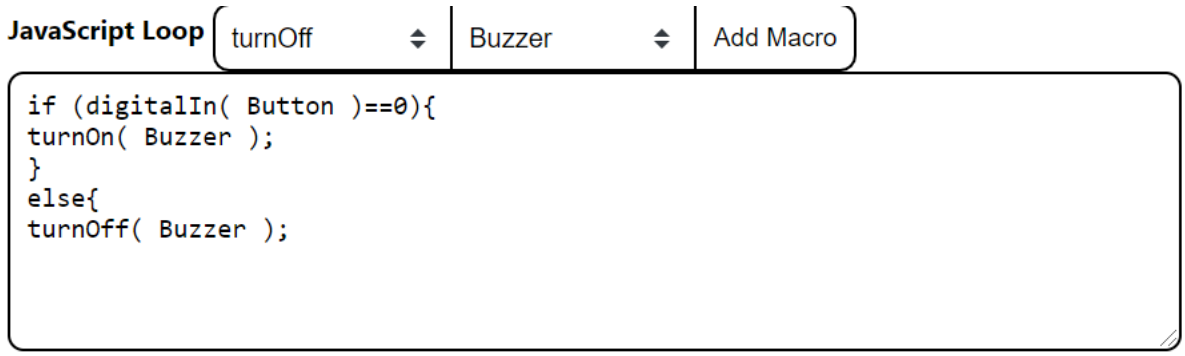
else{



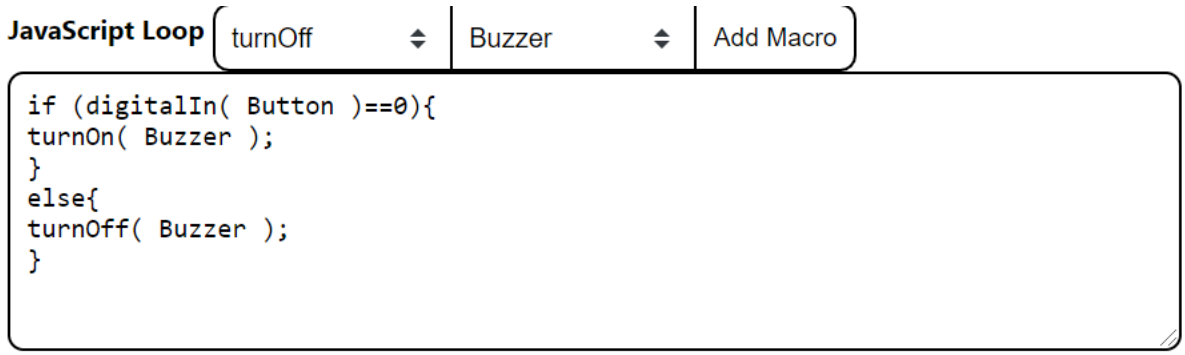
As we don't want the buzzer going off constantly we need to add a turnOff macro to turn off the buzzer. To do this select the turnOff macro from the select macro button.



Then, select the Buzzer from devices and click Add Macro.



At the very end of the code, the program needs a closing bracket "}".



The whole script should look like this:

```
if (digitalIn( Button )== 0){
turnOn( Buzzer );
}
else{
turnOff( Buzzer );
}
```

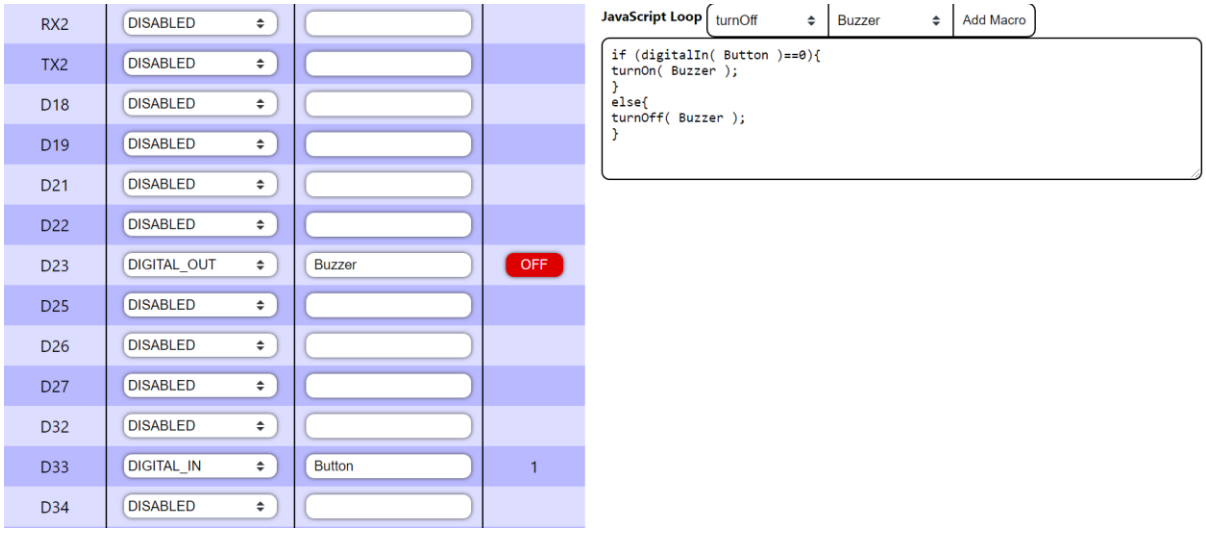


Figure 7: runlinc webpage screenshot **before** the button is pressed

After we click **Run Code,** there should be no sound at all. When we press the Button, the button value will turn from 0 to 1, therefore turning on our Buzzer. If we release the Button, the button value will turn back to 0, thus turning off the Buzzer.

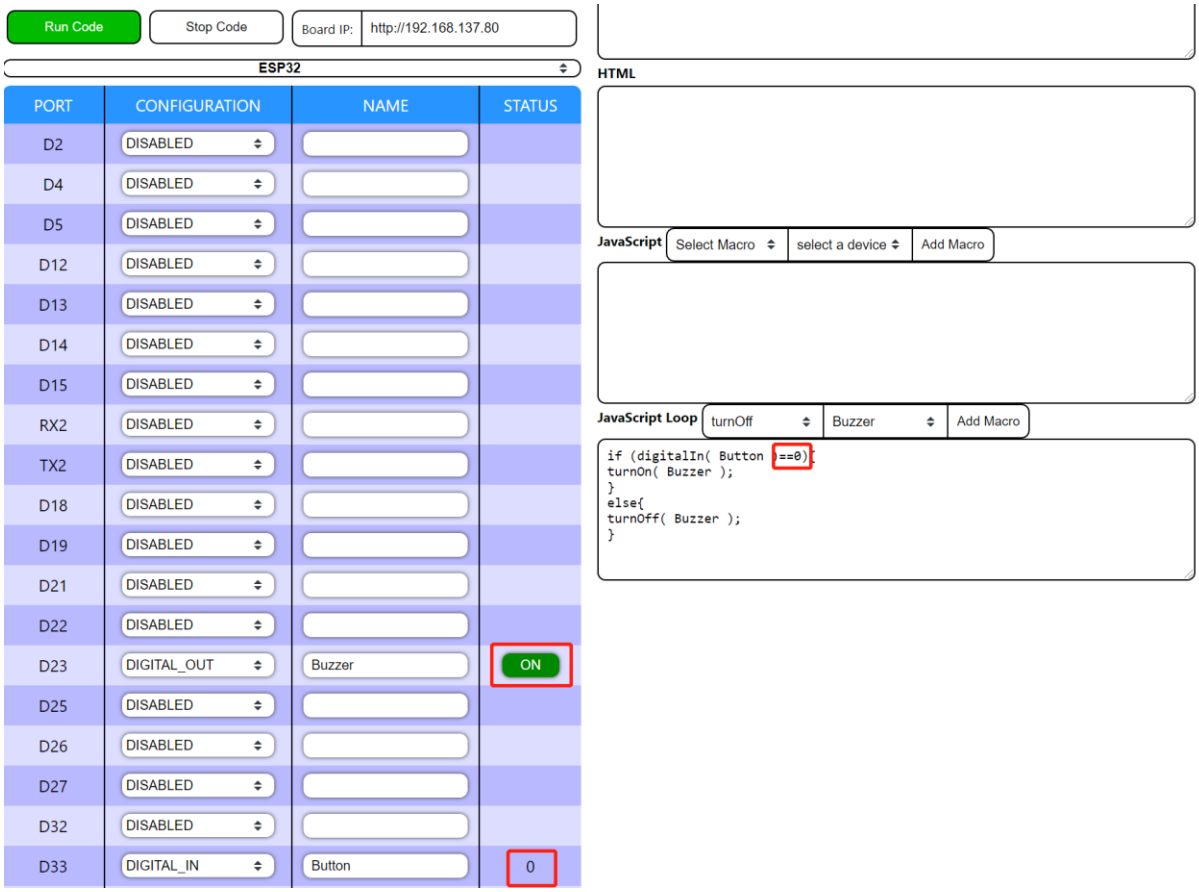


Figure 8: runlinc webpage screenshot **after** the Button is pressed

Summary

Microchips are basically small and simple computers that we can use to do many different things. We use microchips every day in many devices, such as alarm systems. In this project, we built an alarm system that used a pushbutton from the runlinc webpage as input and would activate a buzzer to let us know that the alarm had been triggered. This project should have given you a good background about how to program the microchip on the E32W controller board via runlinc.