runlinc AI Project 2: Simple AI Chatbot (E32W Version)

Contents

Introduction

Problem

How can we use microchips to turn on objects by asking an AI to do it? What if we're too hot and want to be cooled down?

Background

By learning E32W, you can learn to program microchips to tell them what to do. Microchips can monitor devices and warn people when your laptop battery is low, and 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 problems like turning on the green lights at the same time. By using runlinc, we can create an AI to do the things we ask it to do or even have conversations with it. By using AIs, we can help streamline what?

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 generally on an alarm system? What outputs can we use for our AI can use?

Plan

So, we want the runlinc AI to be able to talk to us and others. First, we are going to need a way to talk to our AI. We'll also need to give our AI information so it can talk and give responses. To do these, we'll have to create an input through our webpage, which will allow our AI to talk to us.

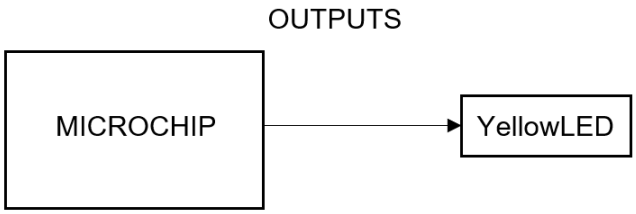

Figure 1: Block diagram of Microchip outputs

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 YellowLED and set it as DIGITAL_OUT.

In our circuit design, we will be using a 3-pin LED module. We happen to have these in our kits, so these can be used on our circuit design, as per the plan.

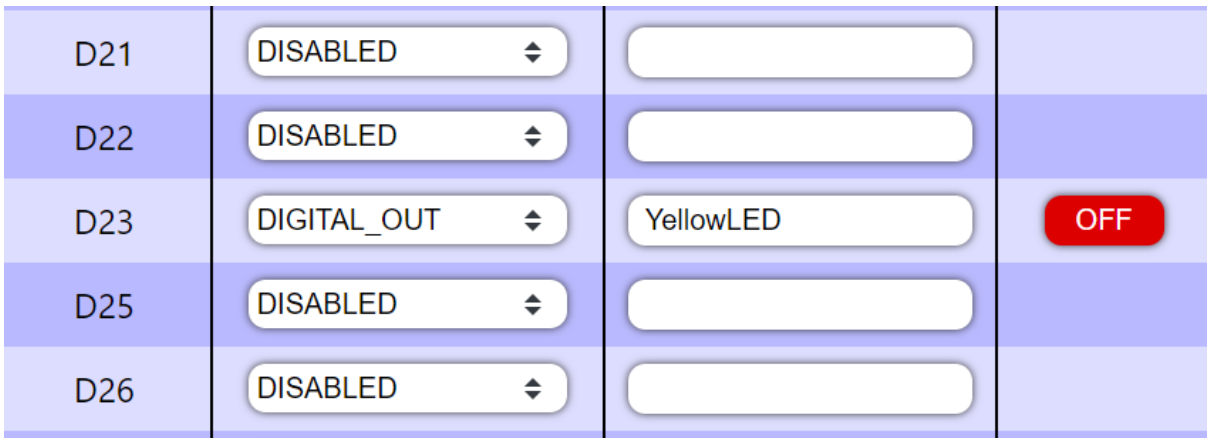

Figure 3: I/O configurations connections

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).

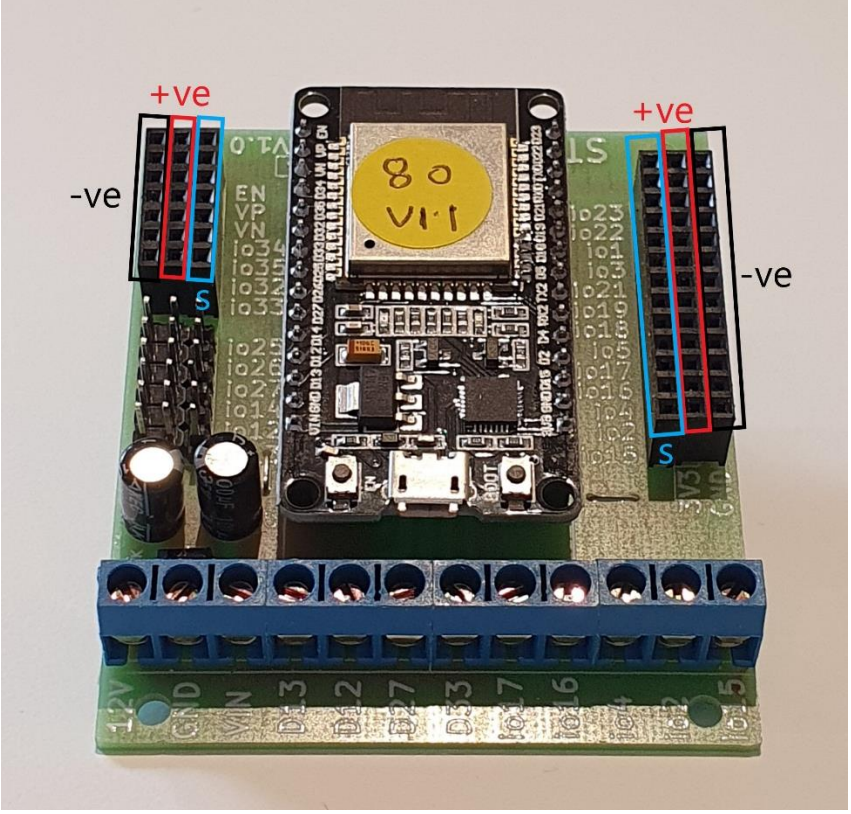

Figure 4: Negative, Positive and Signal port on the E32 board

There is only one I/O part we are using for this project, a 3-pin LED light, its respective pins are shown in the figure below.

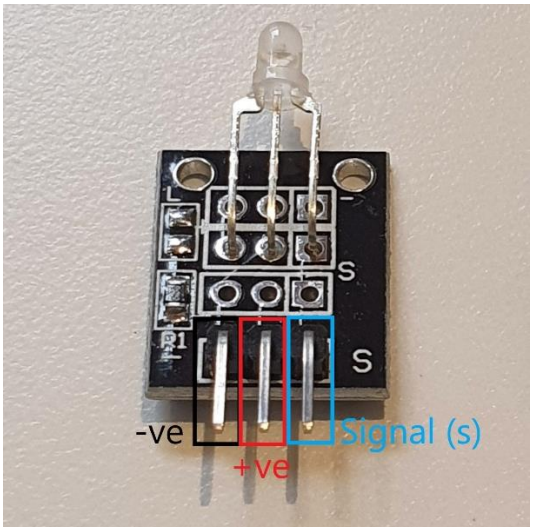

Figure 5: I/O parts with negative, positive and signal pins indicated

Wiring instructions

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

b.) Make sure the (-ve) pin is on the GND (outer) side of the I/O port.

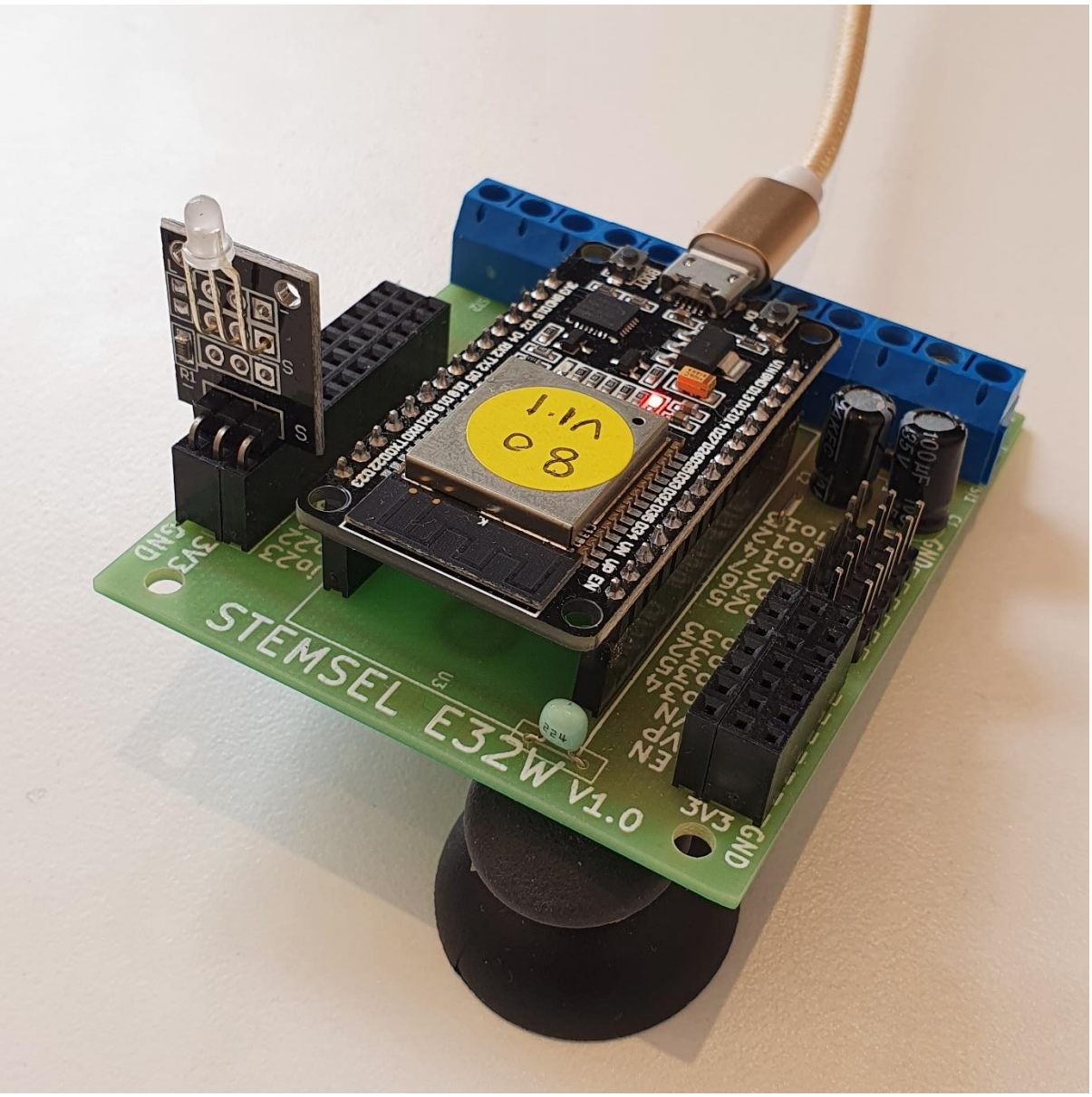

Figure 6: Circuit board connection with I/O parts (side view)

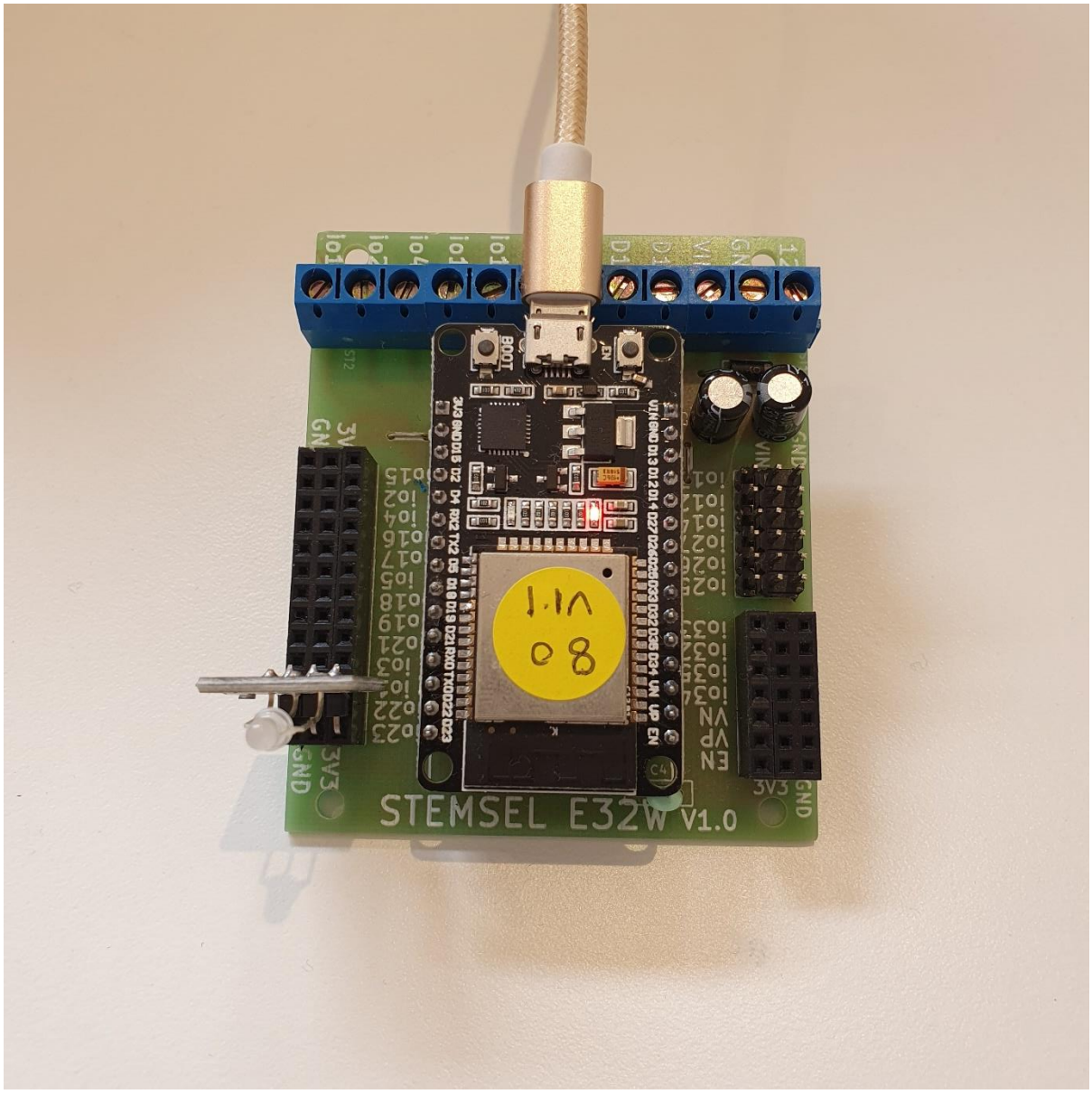

Figure 7: Circuit board connection with I/O parts (top view)

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 contents, CSS to add style to your taste and Javascript to program the microchip. For this case, we will be using HTML and JavaScript to program our AI chatbot.

After naming the port C4, we are going to program the circuit. First, we need **HTML block** to add in the graphics for what we are going to do.

In this step, we'll need to give our webpage a title first, let's call it, My runlinc AI ChatBot. To do this, we'll need to use the Heading tag <h3>.

```
<h3>My runlinc AI Chat Bot</h3>
```

After we have our title, we need to create the text box and a text line saying what it's for. To do this, we will use the paragraph tag <p>.

```
<p>Chat to runlinc AI BOT. Type your text...</p>
```

Now that we have set up the text sections, we need to add an input text field for a user to type in their conversations.

```
<input type="text" id="myText" value="type here...">
```

Next, we need a button to send what has been typed inside the AI. Thus, using the button tag:

```
<button onclick="myFunction()">send</button>
```

Make sure to put in a space between the different sections using the tag at least 3 times to give a good gap.

Now that we have created our input field, we need to create our output field so the AI can respond. First, we should make sure that it's clear that the AI is talking, to do this we'll type in:

```
"runlinc AI says..."
```

Make sure to exclude the " for what you just typed.

Now put in another small break by using the tag.

Then we can put in the output for the AI:

```
<p id="replyText"></p>
```

Now that we have set up all the inputs and outputs, we can now look at creating our AI. To do this, we'll need to use the **JavaScript block.**

To start with, we will create a function called myFunction and declare a variable called x:

```
function myFunction() {
 var x = document.getElementById("myText").value;
```

Now that we have set up the variable, we can give responses to our AI. To do this, it will need to receive what has been typed, then compare it with what it is expecting, which will allow it to give a response.

```
var n = x.includes("Hello");
if ( n == true ) {
x = "Hello to you too!";
turnOff( YellowLED );
}
```

In above, we have set up the AI to respond to somebody who said Hello to it. Now we can give some more functions. Let's say that the AI is in danger and that it will then light up the YellowLED when it is.

```
var n = x.includes("danger");
if ( n == true ) {
x = "Oh now I'm scared!";
turnOn( YellowLED );
}
```

```
var n = x.includes("relax");
if ( n == true ) {
x = "OK Thanks! That was close!";
turnOff( YellowLED );
}
```

Now we should tell it to relax, meaning it's no longer in danger.

Now that we have created a basic AI, we need to code the JavaScript to communicate with the HTML. To do this we need to use the innerHTML method:

```
document.getElementById("replyText").innerHTML = x;
}
```

For **HTML** box:

```
<h3>My runlinc AI Chat Bot</h3>
<p>Chat to runlinc AI BOT. Type your text...</p>
<input type="text" id="myText" value="type here...">
<button onclick="myFunction()">send</button>
<br> <br> <br>
runlinc AI says...
<br>
<p id="replyText"></p>
```

For **JavaScript** box:

```
function myFunction() {
  var x = document.getElementById("myText").value;
  var n = x.includes("Hello");
  if ( n == true ) {
  x = "Hello to you too!";
  turnOff( YellowLED );
 }
  var n = x.includes("danger");
  if ( n == true ) {
  x = "Oh now Im scared";
  turnOn( YellowLED );
 }
  var n = x.includes("relax");
  if ( n == true ) {
  x = "OK Thanks. That was close.";
  turnOff( YellowLED );
 }
  document.getElementById("replyText").innerHTML = x;
}
```

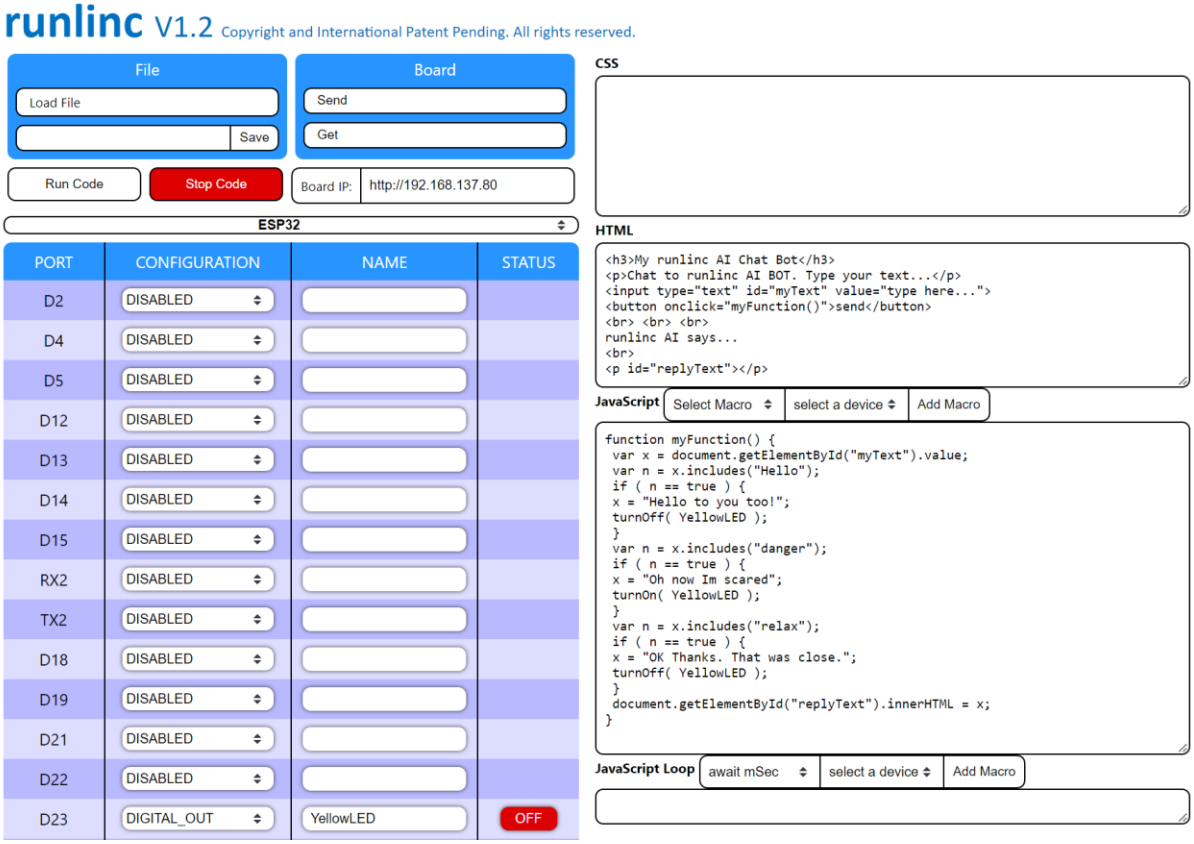

Figure 8: runlinc webpage screenshot

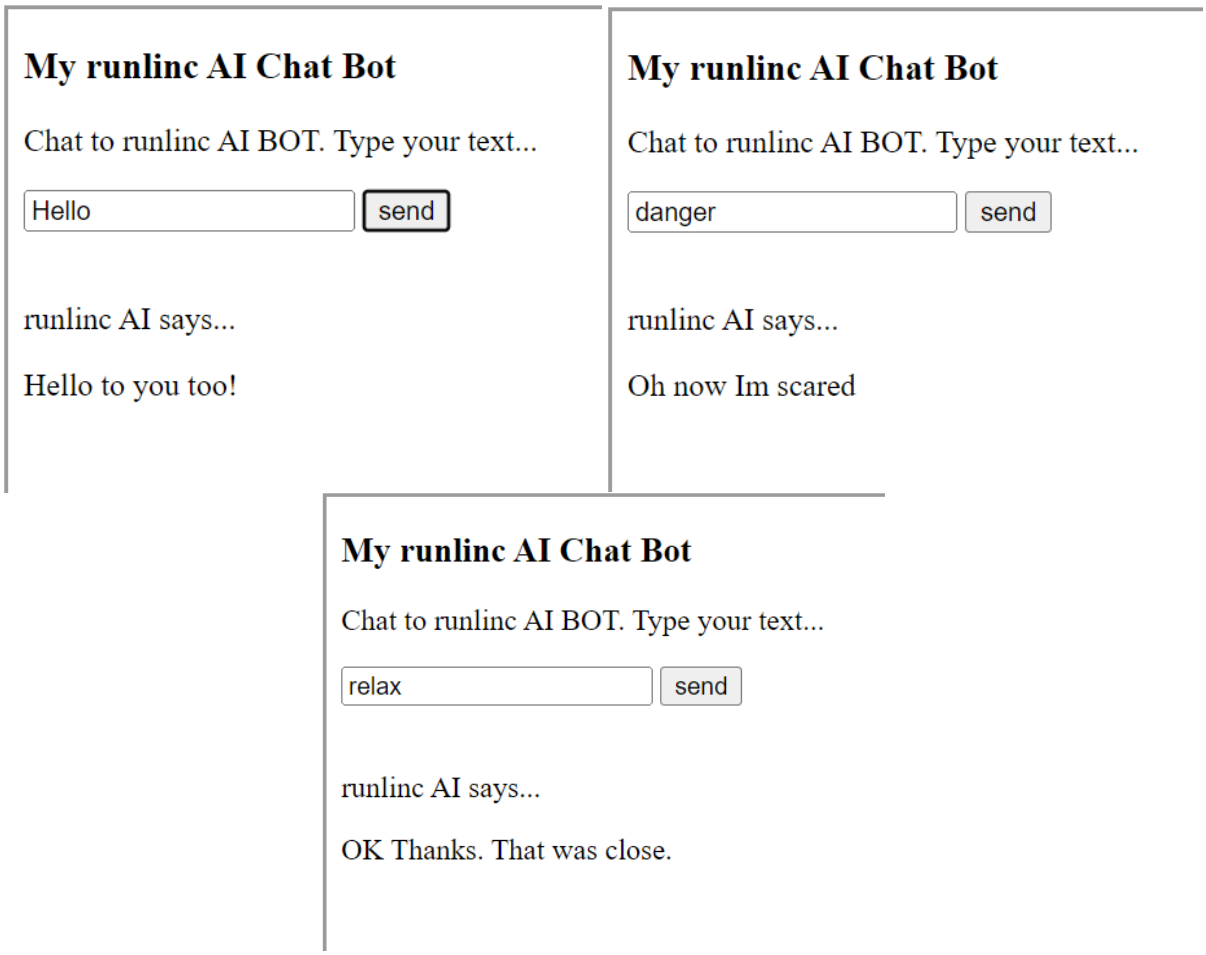

Figure 9: runlinc webpage expected output screenshot

Extension

Now that we have created the core code, we can now look at expanding it to do more than just talk.

The first challenge is to make it have a full conversation with you or somebody else, even turning on and off other devices like a fan. You can achieve this by copying and pasting the code from one of the var declarations and changing the expected input and outputs to whatever you want.

The Second Challenge is to get the AI to talk. Here is a segment of code that could help:

```
speech = new SpeechSynthesisUtterance("Here is a square.")
 window.speechSynthesis.speak(speech);
```

The Third challenge is to get it to draw shapes when it's asked to. To give you a hand, here is a small section of code that might just help (goes in the JavaScript Block and place this piece before document.getElementById("replyText").innerHTML = x;):

```
if (x.includes("square")) {
 speech = new SpeechSynthesisUtterance("Here is a square.")
 window.speechSynthesis.speak(speech);
 x = "Here is a square.";
 var canvas = document.getElementById('canvas');
 var ctx = canvas.getContext('2d');
 ctx.strokeStyle = 'black'
 ctx.beginPath();
 ctx.strokeRect(520,0,100,100);
 ctx.stroke();
}
```

And this line of code should go to HTML block:

```
<canvas id="canvas" height=1000 width=1000></canvas>
```

Summary

People can use programming to tell AI what to do. However, sometimes those AIs can be programmed to have conversations with people, so it is important to program them correctly. In this project, we learned that we can make an AI talk to people and turn on and off devices.