```
// Import required libraries
#ifdef ESP32
  #include <WiFi.h>
  #include <AsyncTCP.h>
#else
  #include <ESP8266WiFi.h>
  #include <ESPAsyncTCP.h>
#endif
#include <ESPAsyncWebServer.h>

// Replace with your network credentials
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";

const char* PARAM_INPUT_1 = "state";
const char* PARAM_INPUT_2 = "value";

const int output = 2;

String timerSliderValue = "10";

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>ESP Web Server</title>
  <style>
    html {font-family: Arial; display: inline-block; text-align: center;}
    h2 {font-size: 2.4rem;}
    p {font-size: 2.2rem;}
    body {max-width: 600px; margin:0px auto; padding-bottom: 25px;}
    .switch {position: relative; display: inline-block; width: 120px;
height: 68px}
    .switch input {display: none}
    .slider {position: absolute; top: 0; left: 0; right: 0; bottom: 0;
background-color: #ccc; border-radius: 34px}
    .slider:before {position: absolute; content: ""; height: 52px; width:
52px; left: 8px; bottom: 8px; background-color: #fff; -webkit-transition:
.4s; transition: .4s; border-radius: 68px}
    input:checked+.slider {background-color: #2196F3}
    input:checked+.slider:before {-webkit-transform: translateX(52px); -ms-
transform: translateX(52px); transform: translateX(52px)}
    .slider2 { -webkit-appearance: none; margin: 14px; width: 300px;
height: 20px; background: #ccc;
      outline: none; -webkit-transition: .2s; transition: opacity .2s;}
    .slider2::-webkit-slider-thumb {-webkit-appearance: none; appearance:
none; width: 30px; height: 30px; background: #2f4468; cursor: pointer;}
    .slider2::-moz-range-thumb { width: 30px; height: 30px; background:
#2f4468; cursor: pointer; }
  </style>
</head>
<body>
  <h2>ESP Web Server</h2>
  <p><span id="timerValue">%TIMERVALUE%</span> s</p>
  <p><input type="range" onchange="updateSliderTimer(this)"
id="timerSlider" min="1" max="20" value="%TIMERVALUE%" step="1"
class="slider2"></p>
  %BUTTONPLACEHOLDER%
```

```
<script>
function toggleCheckbox(element) {
  var sliderValue = document.getElementById("timerSlider").value;
  var xhr = new XMLHttpRequest();
  if(element.checked){ xhr.open("GET", "/update?state=1", true);
xhr.send();
    var count = sliderValue, timer = setInterval(function() {
      count--; document.getElementById("timerValue").innerHTML = count;
      if(count == 0){ clearInterval(timer);
document.getElementById("timerValue").innerHTML =
document.getElementById("timerSlider").value; }
    }, 1000);
    sliderValue = sliderValue*1000;
    setTimeout(function(){ xhr.open("GET", "/update?state=0", true);
    document.getElementById(element.id).checked = false; xhr.send(); },
sliderValue);
  }
}
function updateSliderTimer(element) {
  var sliderValue = document.getElementById("timerSlider").value;
  document.getElementById("timerValue").innerHTML = sliderValue;
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "/slider?value="+sliderValue, true);
  xhr.send();
}
</script>
</body>
</html>
)rawliteral";

// Replaces placeholder with button section in your web page
String processor(const String& var){
  //Serial.println(var);
  if(var == "BUTTONPLACEHOLDER"){
    String buttons = "";
    String outputStateValue = outputState();
    buttons+= "<p><label class=\"switch\"><input type=\"checkbox\"
onchange=\"toggleCheckbox(this)\" id=\"output\" " + outputStateValue +
"><span class=\"slider\"></span></label></p>";
    return buttons;
  }
  else if(var == "TIMERVALUE"){
    return timerSliderValue;
  }
  return String();
}

String outputState(){
  if(digitalRead(output)){
    return "checked";
  }
  else {
    return "";
  }
  return "";
}

void setup(){
  // Serial port for debugging purposes
  Serial.begin(115200);
```

```
  pinMode(output, OUTPUT);
  digitalWrite(output, LOW);

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
  }

  // Print ESP Local IP Address
  Serial.println(WiFi.localIP());

  // Route for root / web page
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/html", index_html, processor);
  });

  // Send a GET request to <ESP_IP>/update?state=<inputMessage>
  server.on("/update", HTTP_GET, [] (AsyncWebServerRequest *request) {
    String inputMessage;
    // GET input1 value on <ESP_IP>/update?state=<inputMessage>
    if (request->hasParam(PARAM_INPUT_1)) {
      inputMessage = request->getParam(PARAM_INPUT_1)->value();
      digitalWrite(output, inputMessage.toInt());
    }
    else {
      inputMessage = "No message sent";
    }
    Serial.println(inputMessage);
    request->send(200, "text/plain", "OK");
  });

  // Send a GET request to <ESP_IP>/slider?value=<inputMessage>
  server.on("/slider", HTTP_GET, [] (AsyncWebServerRequest *request) {
    String inputMessage;
    // GET input1 value on <ESP_IP>/slider?value=<inputMessage>
    if (request->hasParam(PARAM_INPUT_2)) {
      inputMessage = request->getParam(PARAM_INPUT_2)->value();
      timerSliderValue = inputMessage;
    }
    else {
      inputMessage = "No message sent";
    }
    Serial.println(inputMessage);
    request->send(200, "text/plain", "OK");
  });

  // Start server
  server.begin();
}

void loop() {

}
```