INSTITUTO TECNOLÓGICO DE SONORA
Dirección de Ingeniería y Tecnología

Departamento de Computación y Diseño

Sistemas Empotrados

Práctica 10 - Comunicación con el Microcontrolador ESP32 mediante la Aplicación Web WebSerial

En la **Práctica 05 - Comunicación entre el IDE 2.0 de Arduino y el Microcontrolador ESP32 mediante el Puerto USB** se vio que podemos usar el monitor serial del IDE 2.0 de Arduino para recibir y enviar mensajes desde y hacia el microcontrolador ESP32. Esto nos permite recibir información del microcontrolador ESP32 y enviarle comandos. También podemos usar los mensajes enviados por el microcontrolador ESP32 con propósitos de depuración.

Sin embargo, en el caso de tener una aplicación en el que el microcontrolador esté conectado a una red Wifi y no esté conectado directamente a una computadora mediante el puerto USB, podemos aprovechar la conexión a la red Wifi para el envío y recepción de mensajes desde y hacia el microcontrolador ESP32.

La biblioteca **WebSerial** permite que nuestra aplicación genere una aplicación Web con una interfaz de usuario muy sencilla que permite una comunicación serial con el microprocesador ESP32. La interfaz de la aplicación web se implementa usando em marco de trabajo **Vue.js**. El servidor de la aplicación Web es el mismo microcontrolador ESP32.

La biblioteca WebSerial depende de la biblioteca **ESPAsyncWebServer** que nos permite establecer un servidor HTTP asíncrono y de websockets, los que nos permite tener más de una conexión a la vez. La biblioteca **ESPAsyncWebServer** a su vez depende de la biblioteca **AsyncTCP**, así que las tres deben de estar instaladas en el IDE 2.0 de Arduino.

Algunas de las funciones de la clase WiFi

```
void begin(AsyncWebServer *server, const char* url = "/webserial");
```

Inicializa la instancia de la clase WebSerial. El primer parámetro es la instancia de la clase ESPAsyncWebServer. El segundo parámetro es opcional y es el URL de la aplicación Web,

```
msgCallback(RecvMsgHandler _recv);
```

Esta función permite establecer el nombre de la función que será invocada cada vez el microcontrolador reciba un mensaje del navegador. El nombre de la función invocada se le pasa como parámetro. La sintaxis de la función invocada es:

```
void recvMsg(uint8_t *data, size_t len);
```

Esta función procesa los mensajes recibidos del navegador. El mensaje está en el arreglo `data` y tiene una longitud `len`.

```
void print(String m = "");
void print(const char *m);
void print(char *m);
void print(int m);
void print(uint8_t m);
void print(uint16_t m);
void print(uint32_t m);
void print(double m);
```

```
void print(float m);
```

Despliega el dato del parámetro sin hacer un salto de línea.

```
void println(String m = "");
void println(const char *m);
void println(char *m);
void println(int m);
void println(uint8_t m);
void println(uint16_t m);
void println(uint32_t m);
void println(double m);
void println(float m);
```

Despliega el dato del parámetro sin hacer un salto de línea.

Objetivo

Enviar y recibir mensajes al Microcontrolador ESP32 DEVKIT DOIT mediante una aplicación Web usando la biblioteca WebSerial.

Equipo y Materiales

1 Microcontrolador ESP32 DEVKIT DOIT de 30 pines
1 Base para el Microcontrolador ESP32 DEVKIT DOIT de 30 pines
1 Cable USB a micro USB
1 Protoboard
1 Resistencia de 220 Ω
1 LED
Cables Dupont macho – hembra
Cables Dupont macho – macho

Procedimiento

Instalación de las bibliotecas AsyncTCP, ESPAsyncWebServer y WebSerial

1. Descargue la biblioteca **AsyncTCP** de su repositorio github:

   https://github.com/me-no-dev/AsyncTCP

   No descomprima el archivo.

2. Usando el procedimiento: **Instalar una Biblioteca Arduino Importando el Archivo Comprimido con la Biblioteca** visto en la **Práctica 02 - Instalación de una Biblioteca para las placas ESP32 en el IDE 2.0 de Arduino**, instale la biblioteca **AsyncTCP**.

3. Renombre la carpeta AsyncTCP-master a AsyncTCP.

4. Descargue la biblioteca **ESPAsyncWebServer** de su repositorio github:

   https://github.com/me-no-dev/ESPAsyncWebServer

   No descomprima el archivo.

5. Usando el procedimiento: **Instalar una Biblioteca Arduino Importando el Archivo Comprimido con la Biblioteca** visto en la **Práctica 02 - Instalación de una Biblioteca para las placas ESP32 en el IDE 2.0 de Arduino**, instale la biblioteca **ESPAsyncWebServer**.

6. Renombre la carpeta ESPAsyncWebServer-master a ESPAsyncWebServer.

7. Usando el procedimiento: **Instalar una Biblioteca para el microcontrolador ESP32 usando el Administrador de Bibliotecas del IDE 2.0 de Arduino** visto en la **Práctica 02 - Instalación de una Biblioteca para las placas ESP32 en el IDE 2.0 de Arduino**, instale la biblioteca **WebSerial**.

Recepción de mensajes del Microcontrolador ESP32 mediante la aplicación Web usando la biblioteca WebSerial

1. Arme el circuito de las figuras 1 y 2:

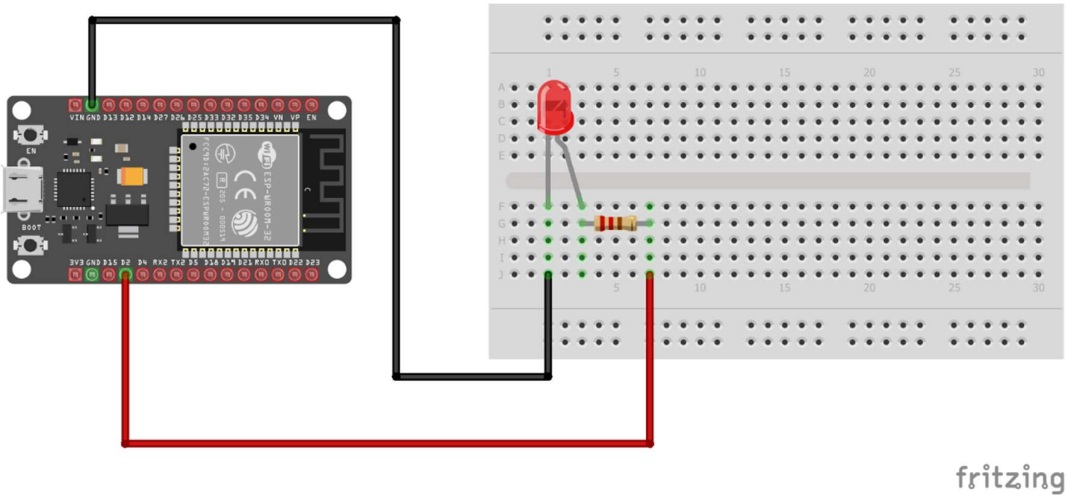

Figura 1

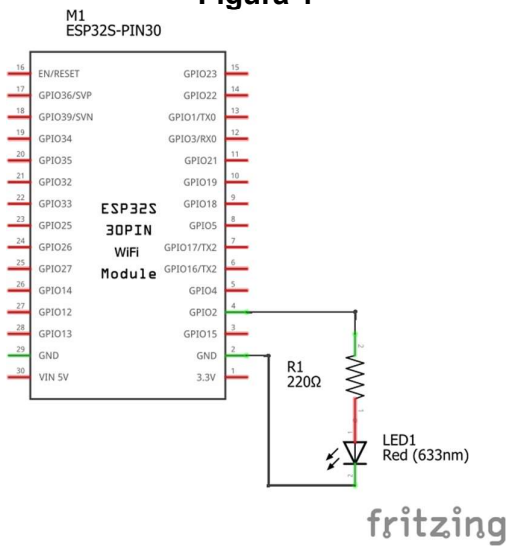

Figura 2

2. Usando el IDE 2.0 de Arduino, edite el siguiente programa. Guárdelo con el nombre de **blinkWebSerial** y compílelo.

blinkWebSerial.ino

```
/*
 * blinkWebSerial.ino
 *
 * Este programa hace que el LED de status del microcontrolador
 * ESP32 DEVKIT DOIT de 30 pines o un LED conectado al pin 2,
 * parpadee. Los periodos de estar encendido y apagado del led
```

```
 * son iguales.
 *
 * Ademas el programa muestra mediante la aplicacion web WebSerial,
 * si el led esta prendido o apagado.
 *
 * Para acceder a la aplicación web, una vez que el programa
 * esta ejecutando, acceder desde un navegador a la direccion:
 *
 *    direccionIP_ESP32/webserial
 *
 * No usa la funcion delay(). En lugar de ello utiliza la
 * biblioteca NoDelay. Esto permite que otro codigo ejecute al
 * mismo tiempo que se encuentra en el periodo de espera.
 */

#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <WebSerial.h>
#include <NoDelay.h>

// Pin a la que esta conectado el LED, GPIO2
const unsigned int PIN_LED = 2;
// Periodo en ms que dura encendido o apagado el LED
const long PERIODO = 500;
// Inicialmente el LED está apagado
int estadoLed = LOW;

// Substituya "SSID" por el nombre del SSID de la red Wifi a
// conectarse
const char* ssid = "SSID";

// Substituya "contraseña" por la contraseña de la red Wifi a
// conectarse
const char* password = "contraseña";

// Velocidad de transmisión del puerto serie
const unsigned int BAUD_RATE = 115200;

// Crea una instancia de la clase noDelay
// que determina si han transcurrido PERIODO ms
noDelay pausa(PERIODO);

// Crea una instancia de un servidor web asincrono que
// escucha en el puerto 80
AsyncWebServer server(80);
```

```
void conectaRedWiFi(const char* ssid, const char* password);

void setup() {
  // Establece el pin PIN_LED (GPIO2) como de salida.
  pinMode(PIN_LED, OUTPUT);

  // Establece la velocidad de transmisión del puerto
  // serie al valor BAUD_RATE
  Serial.begin(BAUD_RATE);

  // Conecta a una red WiFi
  conectaRedWiFi(ssid, password);

  // Inicializa la aplicacion web webSerial
  WebSerial.begin(&server);

  // Inicializa el servidor
  server.begin();
}

void loop() {
  // Verifica si es tiempo de prender/apagar el LED
  if (pausa.update()) {
      // Prender/apagar el LED
      if (estadoLed == LOW) {
          estadoLed = HIGH;
          // Envia mensaje al navegador
          WebSerial.println("LED encendido");
                } else {
          estadoLed = LOW;
          // Envia mensaje al navegador
          WebSerial.println("LED apagado");
      }
      digitalWrite(PIN_LED, estadoLed);
  }
}

/*
 * Conecta el microcontrolador ESP32 a una red WiFi
 */
void conectaRedWiFi(const char* ssid, const char* password) {
  // Conexion a la red
  WiFi.begin(ssid, password);
  Serial.print("Conectandose a la red ");
  Serial.print(ssid);
  Serial.println(" ...");

  // Mientras no se ha conectado a la red WiFi
  while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(1000);
    Serial.print(".");
  }

  Serial.println('\n');
  Serial.println("Connexion establecida");

  // Obten la direccion IP del microcontrolador ESP32
  Serial.print("Direccion IP del servidor web: ");
  Serial.println(WiFi.localIP());
}
```

3. Conecte el Microcontrolador ESP32 a la computadora mediante el cable USB a micro USB.

4. Cargue el programa al Microcontrolador ESP32.

5. Abra el monitor serie.

6. Después de unos segundos, deberá aparecer la dirección IP asignada por la red al microcontrolador ESP32.

7. Abra un navegador Web y acceda a la siguiente URL:

   http://IP_ESP32/webserial

   donde IP_ESP32 es la dirección IP que se le asignó al microcontrolador ESP32 y que aparece en el monitor serie.

   En la ventana del navegador deberán aparecer los mensajes de encendido y apagado del LED, como se muestra en la figura 3.

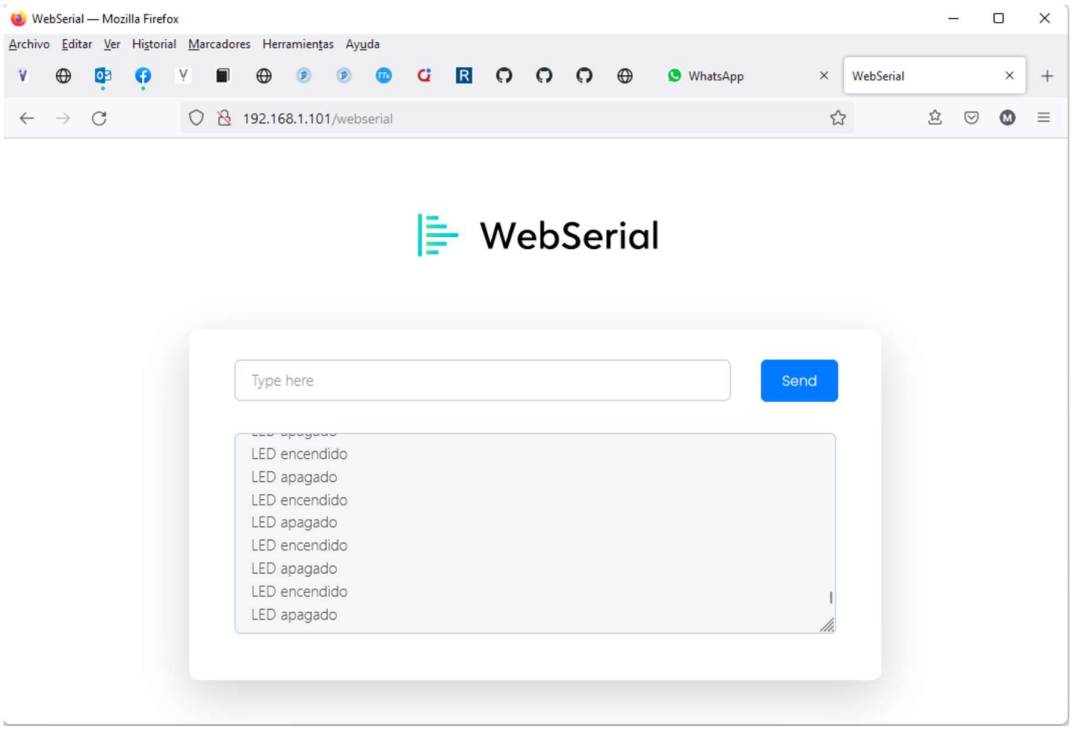

Figura 3

8. Verifique que el programa funciona correctamente.

9. Cierre el monitor serie.

10. Desconecte el Microcontrolador ESP32 de la computadora.

Envío de Mensajes al Microcontrolador ESP32 mediante la aplicación Web usando la biblioteca WebSerial

11. Usando el IDE 2.0 de Arduino, edite el siguiente programa. Guárdelo con el nombre de **blinkWebSerial2** y compílelo.

blinkWebSerial2.ino

```
/*
 * blinkWebSerial2.ino
 *
 * Este programa hace que el LED de status del microcontrolador
 * ESP32 DEVKIT DOIT de 30 pines o un LED conectado al pin 2,
 * se encienda o apague controlado por comandos recibidos por
 * la aplicacion web WebSerial
 *
 * Para acceder a la aplicación web, una vez que el programa
 * esta ejecutando, acceder desde un navegador a la direccion:
 *
 *    direccionIP_ESP32/webserial
 */

#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <WebSerial.h>
#include <string.h>

// Pin a la que esta conectado el LED, GPIO2
const unsigned int PIN_LED = 2;
// Velocidad de transmisión del puerto serie
const unsigned int BAUD_RATE = 115200;

// Substituya "SSID" por el nombre del SSID de la red Wifi a
// conectarse
const char* ssid = "SSID";

// Substituya "contraseña" por la contraseña de la red Wifi a
// conectarse
const char* password = "contraseña";

// Crea una instancia de un servidor web asincrono que
// escucha en el puerto 80
```

```
AsyncWebServer server(80);

void conectaRedWiFi(const char* ssid, const char* password);
void procesaComando(uint8_t *data, size_t len);

void setup() {
  // Establece el pin PIN_LED (GPIO2) como de salida.
  pinMode(PIN_LED, OUTPUT);

  // Establece la velocidad de transmisión del puerto
  // serie al valor BAUD_RATE
  Serial.begin(BAUD_RATE);

  // Conecta a una red WiFi
  conectaRedWiFi(ssid, password);

  // Inicializa la aplicacion web webSerial
  WebSerial.begin(&server);

  // Establece la funcion que sera llamada cada vez que
  // el microcontrolador reciba un comando desde el navegador
  WebSerial.msgCallback(procesaComando);

  // Inicializa el servidor
  server.begin();
}

void loop() {
}

/*
 * Conecta el microcontrolador ESP32 a una red WiFi
 */
void conectaRedWiFi(const char* ssid, const char* password) {
  // Conexion a la red
  WiFi.begin(ssid, password);
  Serial.print("Conectandose a la red ");
  Serial.print(ssid);
  Serial.println(" ...");

  // Mientras no se ha conectado a la red WiFi
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }

  Serial.println('\n');
  Serial.println("Connexion establecida");
```

```
  // Obten la direccion IP del microcontrolador ESP32
  Serial.print("Direccion IP del servidor web: ");
  Serial.println(WiFi.localIP());
}

/*
 * Esta función es llamada cada vez que el servidor
 * recibe un comando. El comando se recibe en el
 * arreglo apuntado por el apuntador data y la
 * longitud del mensaje está dada por len
 *
 */
 void procesaComando(uint8_t *data, size_t len){
  // Envía mensajes al navegador
  WebSerial.println("Comando recibido...");
  WebSerial.print("Longitud del comando: ");
  WebSerial.println(len);

  String comando = "";
  for(int i=0; i < len; i++){
    comando += char(data[i]);
  }
  // Hace eco del comando al navegador
  WebSerial.print("Comando: ");
  WebSerial.println(comando);

  // Si se lee el comando "on"
  if (comando == "on") {
      // Enciende el led conectado al pin PIN_LED
      digitalWrite(PIN_LED, HIGH);
      // Envía mensaje al navegador
      WebSerial.println("LED encendido");
  }
  else
  // Si se lee el comando "off"

  if (comando == "off") {
      // Apaga el led conectado al pin PIN_LED
      digitalWrite(PIN_LED, LOW);
      // Envía mensaje al navegador
      WebSerial.println("LED apagado");
  }
  else {
      // Envía mensajes al navegador
      WebSerial.print("Comando desconocido: ");
      WebSerial.println(comando);
  }
}
```

12. Conecte el Microcontrolador ESP32 a la computadora mediante el cable USB a micro USB.

13. Cargue el programa al Microcontrolador ESP32.

14. Después de unos segundos, deberá aparecer la dirección IP asignada por la red al microcontrolador ESP32.

15. Abra un navegador Web y acceda a la siguiente URL:

    http://IP_ESP32/webserial

    donde IP_ESP32 es la dirección IP que se le asignó al microcontrolador ESP32 y que aparece en el monitor serie.

    En la ventana del navegador aparece la interfaz de la aplicación Web

16. Para enviarle un mensaje al Microcontrolador ESP32 desde el navegador de Arduino utilice el campo de texto que se encuentra en la parte superior de la interfaz de la aplicación Web, como se muestra en la figura 4.

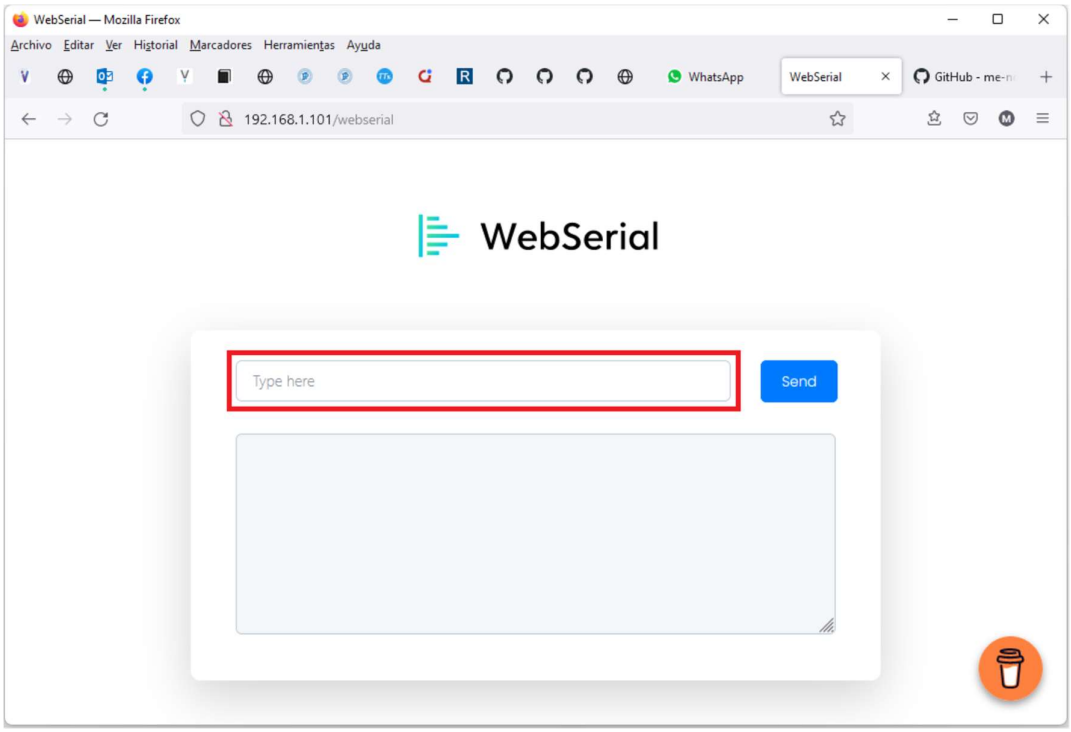

Figura 4

17. Tecleé los mensajes “**on**” o “**off**” para encender o apagar el led y presione el botón **Enviar**, figura 5.

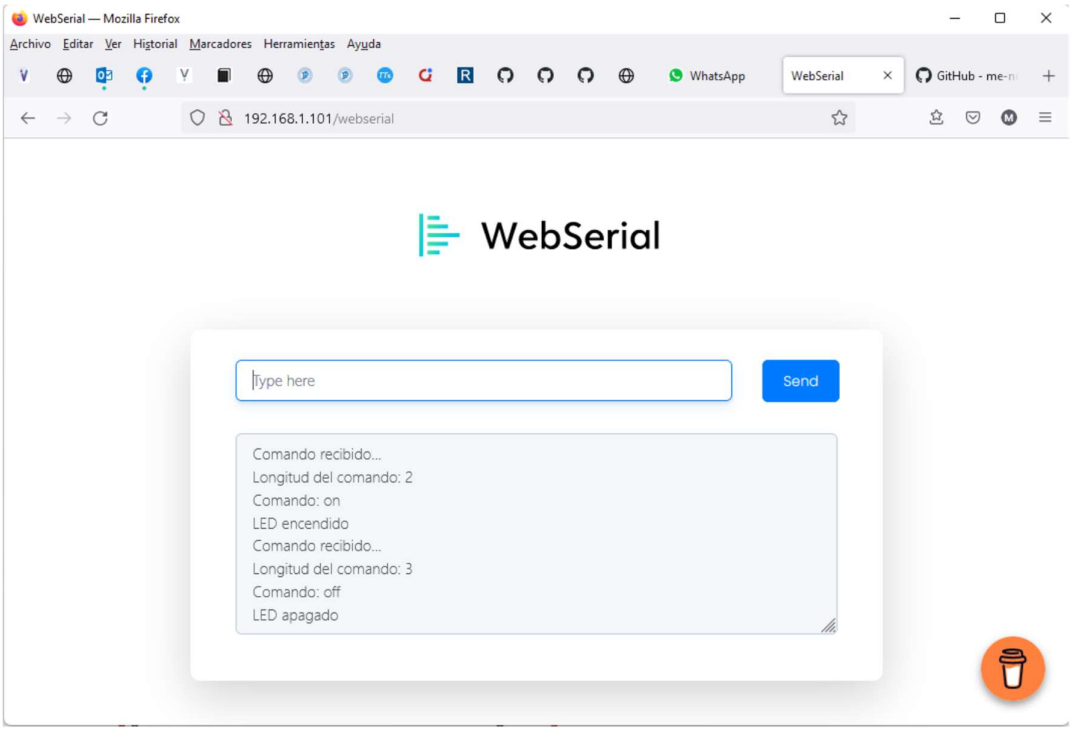

Figura 7

18. Verifique que el programa funciona correctamente.
19. Cierre el monitor serie.
20. Desconecte el Microcontrolador ESP32 de la computadora.