PROYECTO INTEGRADOR I –TST 2024

Sprint Backlog - Semana 21: Implementación de Almacenamiento Local Temporal en ESP32

Descripción:

El objetivo es almacenar temporalmente los datos en el **ESP32** si la conexión a la **API RESTful** falla. Cuando la conexión se restablece, el **ESP32** envía automáticamente los datos.

PAGINA

1. Configurar Almacenamiento

Utilizar la **EEPROM** o el **SPIFFS (SPI Flash File System)** para almacenar los datos localmente. Para este ejemplo, se usa **SPIFFS**, que permite un almacenamiento de los datos localmente.

Configuración inicial de SPIFFS:

Asegurar de tener instalada la biblioteca **SPIFFS.** Si no está incluido en tu entorno de desarrollo, se puede agregarla fácilmente en el **IDE** de Arduino.

En el código, inicializa SPIFFS en el **setup ()**

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <SPIFFS.h>



cons
const char* ssid = "TU_SSID";

c
const char* password = "TU_PASSWORD";
const char* apiUrl = "http://tuservidor.com/api";

void setup() {
  Serial.
  Seria



begin(115200);

  // Inicializar SPIFFS
  if (!SPIFFS.begin(true)) {
    Serial.
    Seria
println("Error al inicializar SPIFFS");

    r
return;
  }

  // Conectar a Wi-Fi
  WiFi.
  WiFi.begin
begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Conectando a WiFi...");
  }
  Serial.
  }
  Serial.pr
println("Conectado a WiFi");
}
```

2. Implementar

Código para almacenar datos localmente y reenvío:

```
getString();
    Serial.
    Seri
println("Respuesta del servidor: " + response);
  }

else {
    Serial.
    Se
println("Error en la conexión: " + String(httpResponseCode));

    almacenarDatosLoc

    alma
almacenarDatosLocalmente(data); // Almacenar datos si hay un error
  }

  http.
  }

void enviarDatos(String data) {
  HTTPClient http;
  http.
  HTTPClient http;
  http.begi

  HT
begin(String(apiUrl) + "/crear");
  http.addHeader("Content-Type", "application/json");

  int httpResponseCode = http.POST(data);

  if (httpResponseCode > 0) {
    Serial.println("Datos enviados: " + data);
    String response = http.
    St
```

//..

..//

```
    almacenarDatosLoc

    alma
almacenarDatosLocalmente(data); // Almacenar datos si hay un error
  }

  http.
  }

  http.end
end();
}
getString();
    Serial.
    Seri
println("Respuesta del servidor: " + response);
  }

else {
    Serial.
    Se
println("Error en la conexión: " + String(httpResponseCode));

    almacenarDatosLoc

    alma
almacenarDatosLocalmente(data); // Almacenar datos si hay un error
  }

  http.
  }

  http.end
end();
}
```

//**

**//

```
void almacenarDatosLocalmente(String data) {
  File file = SPIFFS.
  File file = SPIFFS.o
open("/datos.txt", "a"); // Abrir el archivo en modo append
  if (!file) {
    Serial.
    Serial.pri
println("Error al abrir el archivo para escribir");
    return;
  }
  file.
  }
println(data); // Escribir datos en el archivo
  file.close(); // Cerrar el archivo
}

void reenviarDatosAlmacenados() {
  File file = SPIFFS.
  File fi
open("/datos.txt", "r"); // Abrir el archivo en modo read
  if (!file) {
    Serial.
    Serial.print
println("No hay datos almacenados");
    return;
  }

  String data;

  }
  Stri
while (file.available()) {
    data = file.
    data = file

    da
readStringUntil('\n'); // Leer línea por línea

    en
enviarDatos(data); // Intentar enviar datos
  }
  file.
  }
  file.c
close(); // Cerrar el archivo


// Limpiar el archivo después de reenviar
  SPIFFS.remove("/datos.txt");
}
```

3. Probar el Reenvío de Datos

En el **loop()**, verificable

Código para el loop():

```
void loop() {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.

println("Desconectado, intentando reconectar...");
    WiFi.
    WiFi.
reconnect();

    del
delay(5000); // Esperar antes de intentar reconectar
  }

else {

    /
    /
// Simular datos a enviar
    String jsonData = "{\"temperatura\": 25, \"humedad\": 60}";

    enviarD
enviarDatos(jsonData);
  }

  // Reenviar datos almacenados si se reconecta
  if (WiFi.status() == WL_CONNECTED) {

    reenviarDatosAlmacenad

    reenviar
reenviarDatosAlmacenados();
  }


  }



delay(10000); // Espera antes de la próxima acción
}
```

Reconectar

1. **Configuración de almacenamiento:**

   Inicializar **SPIFFS** en el **setup().**

2. **Función de Envío de Datos:**

   Usar **enviarDatos()** para intentar enviar datos

   Si falla, llama a **almacenarDatosLocalmente(data)** para almacenar datos localmente

3. **Reenvío de datos:**

   En **reenviarDatosAlmacenados ()**, leer los datos datos.

4. Evitar que se eliminan los Datos 1. **Error en el Envío:** Si el ESP32 intenta enviar datos a la API y ocurre un error (por ejemplo, la conexión a Internet se interrumpe o el servidor no responde), los datos que intentabas enviar no deben ser eliminados descartados

5. **Reconexión:**

   En el **loop()**, revisar la reconexión.

Conclusión

Este enfoque permite almacenar los datos temporalmente en el **ESP32** y reenviarlos a la **API RESTful** una vez que la conexión se restablece.

Si el intento de enviar datos a la **API RESTful falla**, por ejemplo, si hay un problema de conexión o un error en la respuesta del servidor, el código ejecuta la función **almacenarDatosLocalmente (data)**, cuyo propósito de **`almacenarDatosLocalmente (data)`**
La función **`almacenarDatosLocalmente (data)`** se encarga de:

1. **Almacenar Datos Localmente**, guardar los datos en un archivo dentro del sistema de archivos del **ESP32**, usando **SPIFFS.** Esto asegura que no se perderán datos importantes si hay un problema de conexión.

2.Evitar que se pierdan datos importantes cuando el **ESP32** no puede enviar información a la **API RESTful.**
Una explicación más clara de lo que significa y su propósito: es evitar Que se Pierdan los Datos
Situación de Fallo: Cuando el **ESP32** intenta enviar datos a la **API** y se encuentra con un error, como un problema de conexión a Internet o una respuesta no válida del servidor, en lugar de simplemente ignorar esos datos o perderlos, se utiliza la función **`almacenarDatosLocalmente(data)`.**