Resumen de los Cambios Realizados en la Screen y Envío de Datos desde la ESP8266

1. Cambios en el Archivo `screen.c`

Se realizaron modificaciones en la pantalla para procesar y mostrar mensajes de error basados en un byte de errores enviado por la ESP8266.

a. Decodificación de Errores

- Se agregó una función llamada `decode_bit`, que verifica si un bit específico de un byte está activado (`1`). Esto se utiliza para determinar qué errores están presentes.
- Ejemplo:

```
bool decode_bit(uint8_t byte, uint8_t bit_position) {
    return (byte & (1 << bit_position)) != 0;
}
```

b. Mostrar Mensajes de Error

- Se implementó una función `display_errors` que genera una lista de mensajes de error basados en los bits activados del byte recibido.
- Los mensajes se muestran en el área de alarmas de la pantalla (`label_alarm`).
- Ejemplo:

```
void display_errors(uint8_t error_byte) {
    char error_text[256] = "";
    for (uint8_t i = 0; i < NUM_ERROR_BITS; i++) {
        if (decode_bit(error_byte, i)) {
            strcat(error_text, error_messages[i]);
            strcat(error_text, "\n");
        }
    }
    if (strlen(error_text) == 0) {
        strcpy(error_text, "No se detectaron errores.");
    }
    lv_label_set_text(label_alarm, error_text);
}
```

c. Estructura de Datos Extendida

- La estructura `uart_data_t` se amplió para incluir un campo para almacenar el byte de errores (`uint8_t errores`).

d. Actualización de Pantalla

- En la función `update_labels_callback`, se actualizan los valores de temperatura, volumen y errores.
- Ejemplo:

```
static void update_labels_callback(void *param) {
    uart_data_t *data = (uart_data_t *)param;
    lv_label_set_text_fmt(label_temp1, "T1: %.2f °C", data->t1);
    lv_label_set_text_fmt(label_temp2, "T2: %.2f °C", data->t2);
    lv_label_set_text_fmt(label_volume, "Volumen: %d ml", data->vol);
    display_errors(data->errores);
}
```

e. Manejo de Datos Recibidos

- Se modificó `screen_data_handler` para parsear el byte de errores desde la trama enviada por la ESP8266.
- Formato esperado: `DATA:T1=<temp1>;T2=<temp2>;VOL=<volumen>;ERR=0x<error_byte>;`.
- Ejemplo:

```
static void screen_data_handler(const char *data) {
    float t1, t2;
    int vol;
    uint8_t errores = 0;
    if (sscanf(data, "DATA:T1=%f;T2=%f;VOL=%d;ERR=0x%hhX;", &t1, &t2, &vol, &errores) >= 3) {
        latest_data.t1 = t1;
        latest_data.t2 = t2;
        latest_data.vol = vol;
        latest_data.errores = errores;
        lv_async_call(update_labels_callback, &latest_data);
    }
}
```

2. Envío de Datos desde la ESP8266

Se añadió lógica para enviar datos simulados, incluyendo temperaturas, volumen y un byte de errores aleatorio.

Formato de la Trama

La ESP8266 envía los datos en el siguiente formato:

```
DATA:T1=<temp1>;T2=<temp2>;VOL=<volumen>;ERR=0x<error_byte>;
```

Código en la ESP8266

- El byte de errores se genera aleatoriamente para pruebas:

  ```
  uint8_t error_byte = random(0, 256); // Byte aleatorio entre 0x00 y 0xFF
  ```

- Ejemplo de generación y envío de la trama:

  ```
  void loop() {
      unsigned long currentMillis = millis();
      if (currentMillis - previousMillis >= interval) {
          previousMillis = currentMillis;

          // Generar valores simulados
          float t1 = random(200, 300) / 10.0; // 20.0°C a 30.0°C
          float t2 = random(200, 300) / 10.0;
          int volume = random(50, 500); // 50 ml a 500 ml
          uint8_t error_byte = random(0, 256);

          // Crear y enviar la trama
          String trama = "DATA:T1=" + String(t1, 2) + ";T2=" + String(t2, 2) +
                         ";VOL=" + String(volume) + ";ERR=0x" + String(error_byte, HEX) + ";\n";
          Serial.print(trama);
      }
  }
  ```

Interpretación del Byte de Errores

- Cada bit del byte representa un posible error:
  - Bit 0: Error 1
  - Bit 1: Error 2
  - ...
  - Bit 7: Error 8
- Ejemplo:
  - `0x05` (`00000101` en binario): Representa que están activos los errores 1 y 3.

---

3. Integración por el Compañero

Para integrar estos cambios:

1. **Desde el lado de la pantalla**:
   - Asegúrate de que el byte de errores (`ERR=0xXX`) esté incluido en la trama enviada.
   - Los datos deben estar en el formato especificado para que `screen.c` pueda procesarlos correctamente.
2. **Desde el lado de la ESP8266**:
   - Implementa el envío de los datos simulados con el formato correcto.
   - Si ya se tienen errores reales, ajusta el valor del byte de errores (`error_byte`) según los estados detectados por los sensores o lógicas del sistema.

Con estas modificaciones, ambos sistemas podrán interactuar correctamente para mostrar los errores detectados en tiempo real en la pantalla.