Clase SIM800SmsManager

1. Propósito

La clase `SIM800SmsManager` abstrae la comunicación UART con el módem **SIM800L**.

Está diseñada para:

- Inicializar el puerto serie con parámetros configurables.
- Enviar y recibir **comandos AT** con control de tiempo de espera.
- Limpiar el buffer del módem para evitar respuestas contaminadas.
- Enviar SMS de manera **confiable** en modo texto (`AT+CMGF=1`), con confirmación de entrega.

---

2. Definición en Header (`SIM800SmsManager.h`)

```
#pragma once
#include <Arduino.h>

class SIM800SmsManager {
public:
  SIM800SmsManager(HardwareSerial& modem) : modem_(modem) {}

  bool begin(uint32_t baud = 9600, int rxPin = -1, int txPin = -1);
  String at(const String& cmd, uint16_t waitMs = 1500, bool echo = true
  // Envío confiable de SMS en modo texto
  bool sendSmsReliable(const String& number, const String& text);

private:
  void flushModem();
  String readFor(unsigned long ms);

  HardwareSerial& modem_;
};

```

---

3. Implementación en `.cpp` (`SIM800SmsManager.cpp`)

3.1 Inicialización del puerto serie

```
bool SIM800SmsManager::begin(uint32_t baud, int rxPin, int txPin) {
  if (rxPin >= 0 && txPin >= 0) modem_.begin(baud, SERIAL_8N1, rxPin, tx
  else modem_.begin(baud);
  return true;
}

```

- Permite iniciar la comunicación con el módem.
- Si se especifican `rxPin` y `txPin` , se usan explícitamente.
- Si no, se usan los pines por defecto.

3.2 Limpieza de buffer

```
void SIM800SmsManager::flushModem() {
  while (modem_.available()) modem_.read();
```

```
}

```

- Vacia el buffer del puerto serie antes de enviar un nuevo comando.
- Evita que datos residuales contaminen la siguiente respuesta.

3.3 Lectura con timeout

```
String SIM800SmsManager::readFor(unsigned long ms) {
  unsigned long t0 = millis();
  String r;
  while (millis() - t0 < ms) {
    while (modem_.available()) r += (char)modem_.read();
    delay(1);
  }
  return r;
}

```

- Lee datos del módem durante un tiempo máximo `ms`.
- Se utiliza internamente en `at()` y `sendSmsReliable()`.
- Devuelve la cadena recibida, incluso si llega incompleta.

3.4 Envío de comandos AT

```
String SIM800SmsManager::at(const String& cmd, uint16_t waitMs, bool ec
  flushModem();
  if (echo) { Serial.print("[AT] "); Serial.println(cmd); }
  modem_.print(cmd); modem_.print("\r\n");
  String r = readFor(waitMs);
  r.trim();
  if (echo) { Serial.println(r); Serial.println("----------------------
  return r;
}

```

- Envía un comando AT y espera la respuesta.
- Parámetros:
  - `cmd`: el comando AT.
  - `waitMs`: tiempo máximo de espera en ms.
  - `echo`: imprime en Serial el comando y la respuesta.
- Retorna la respuesta como `String`.

3.5 Envío confiable de SMS

```
bool SIM800SmsManager::sendSmsReliable(const String& number, const Stri
  at("AT+CMGF=1", 1000, false);   // modo texto
  flushModem();

  modem_.print("AT+CMGS=\""); modem_.print(number); modem_.print("\"\r\
  String prompt = readFor(5000);
  if (prompt.indexOf('>') == -1) {
    Serial.println("[SMS] No prompt '>'");
    return false;
  }

  modem_.print(text);

```

```
  modem_.write((uint8_t)0x1A); // Ctrl+Z

  String resp = readFor(15000);
  bool ok = (resp.indexOf("+CMGS:") != -1 && resp.indexOf("OK") != -1);
  Serial.println(ok ? "[SMS] Enviado OK" : "[SMS] Fallo envío");

  at("AT+CMGF=0");  // volver a PDU
  return ok;
}

```

- Cambia al **modo texto** (`AT+CMGF=1`).
- Envía el comando `AT+CMGS="número"` .
- Espera el prompt `>` (permiso para escribir el mensaje).
- Envía el texto + **Ctrl+Z (0x1A)** para finalizar.
- Espera confirmación (`+CMGS` y `OK`).
- Retorna `true` si el SMS se envió correctamente.
- Regresa al **modo PDU** (`AT+CMGF=0`) para compatibilidad con la recepción.

---

4. Tabla de Métodos Públicos

Método	Retorno	Descripción	Ejemplo
`begin(baud, rxPin, txPin)`	`bool`	Inicializa la comunicación UART con el SIM800L.	`smsMgr.begin(9600, 26, 27);`
`at(cmd, waitMs, echo)`	`String`	Envía un comando AT y devuelve la respuesta.	`smsMgr.at("AT+CSQ", 1000, true);`
`sendSmsReliable(number, text)`	`bool`	Envía un SMS con confirmación de entrega.	`smsMgr.sendSmsReliable("+521234567890", "Hola!");`

---

5. Ejemplo de Uso

```
#include "SIM800SmsManager.h"

HardwareSerial MODEM(2);
SIM800SmsManager smsMgr(MODEM);

void setup() {
  Serial.begin(115200);

```

```
  // Iniciar SIM800L en UART2
  smsMgr.begin(9600, 26, 27);

  // Verificar comunicación
  String resp = smsMgr.at("AT");
  Serial.println("Resp AT: " + resp);

  // Enviar SMS de prueba
  if (smsMgr.sendSmsReliable("+521234567890", "Sistema iniciado")) {
    Serial.println("SMS enviado correctamente");
  } else {
    Serial.println("Error al enviar SMS");
  }
}

void loop() {
  // Aquí se integraría recepción de SMS y lógica adicional
}

```

---

6. Posibles Problemas y Soluciones

Problema	Causa	Solución
No aparece prompt `>`	Red no lista o SIM sin señal	Revisar con `AT+CSQ` y `AT+CREG?`
`sendSmsReliable` falla siempre	SIM sin saldo o SMS deshabilitados	Revisar plan de datos/SMS
Respuestas AT vacías	Baudrate incorrecto	Ajustar `SIM_BAUD` en `config.h`
Respuesta con retraso	Timeout muy corto	Incrementar `waitMs` en `at()`