Wilobu Firmware

Firmware ESP32 para dispositivos Wilobu de seguridad IoT con soporte multi-hardware.

Hardware Soportado

Hardware A (Producción)

- **Placa**: LILYGO T-PCIE
- **Módem**: SIM7080G (Cat-M)
- **Conexión**: HTTPS directo con TLS 1.2
- **APN**: "hologram"

Hardware B (Prototipo con Batería)

- **Placa**: ESP32 DevKit
- **Módem**: SIMCOM A7670SA (Cat-1)
- **Conexión**: HTTP vía Cloudflare Worker (proxy)
- **APN**: Prepago local

Hardware C (Demo USB)

- Igual a Hardware B pero alimentado por USB (sin batería)

Arquitectura del Código

```
src/
├── main.cpp           # Lógica principal y loop()
├── ModemHTTPS.cpp     # Implementación para SIM7080G
└── ModemProxy.cpp     # Implementación para A7670SA

include/
├── IModem.h           # Interfaz abstracta
├── ModemHTTPS.h       # Headers SIM7080G
└── ModemProxy.h       # Headers A7670SA
```

Patrón de Diseño

Usa **polimorfismo** con clase base abstracta `IModem` e implementaciones concretas según el hardware.

Mapa de Pines

```
// Botones SOS
#define BTN_SOS         15    // Botón SOS General
#define BTN_MEDICA      5     // Alerta Médica
```

```
#define BTN_SEGURIDAD   13    // Alerta Seguridad

// Control
#define SWITCH_PWR      27    // Switch de encendido

// LEDs
#define LED_ESTADO      23    // LED Azul (estado)
#define LED_AUX         19    // LED Verde (solo Hardware B)

// UART Módem
#define MODEM_TX        21
#define MODEM_RX        22
```

Compilación

Seleccionar Hardware

Edita `platformio.ini` y comenta/descomenta las líneas:

```
; Hardware A (Producción)
build_flags = -D HARDWARE_A

; Hardware B (Prototipo)
; build_flags = -D HARDWARE_B

; Hardware C (Demo)
; build_flags = -D HARDWARE_C
```

Compilar y Flashear

```
# Compilar
pio run

# Flashear
pio run -t upload

# Monitor serial
pio device monitor
```

Funcionalidades

1. Provisioning BLE

- UUID Service: `0000ffaa-0000-1000-8000-00805f9b34fb`
- Recibe credenciales WiFi y userID
- Kill Switch automático post-vinculación

2. Detección de Botones

- Debounce de 50ms
- Presión larga (5 seg) para provisioning
- Diferencia entre 3 tipos de SOS

3. Comunicación Celular

- Envío de status a Firestore
- Envío de coordenadas GPS
- Reintentos automáticos

4. GPS

- Integración con TinyGPSPlus
- Actualización continua de ubicación
- Fallback a coordenadas hardcoded si falla

Dependencias (platformio.ini)

```
lib_deps =
    bblanchon/ArduinoJson @ ^7.2.0
    mikalhart/TinyGPSPlus @ ^1.1.0
    h2zero/NimBLE-Arduino @ ^1.4.1
```

Firestore REST API

El firmware actualiza documentos en:

```
users/{userId}/devices/{deviceId}
```

Campos actualizados:

- `status`: "online" | "sos_general" | "sos_medica" | "sos_seguridad"
- `lastLocation`: { geopoint, timestamp }
- `otaProgress`: 0-100

Gestión de Energía

- **Deep Sleep** cuando no hay alertas activas
- **Secuencia PWRKEY** para arranque del módem
- Monitoreo de batería (Hardware B/C)

Troubleshooting

El módem no responde

- Verifica conexión UART (TX/RX cruzados)
- Revisa voltaje de alimentación (mínimo 3.7V)
- Confirma que el módem tiene SIM insertada

No conecta a red celular

- Verifica APN correcto para tu operador
- Asegúrate que hay cobertura LTE/Cat-M
- Revisa configuración de bandas en código

BLE no aparece

- Presiona botón 1 por 5 segundos
- Verifica que no se ejecutó el Kill Switch previamente
- Reinicia el ESP32