Guía de Despliegue nRF9151 - Firmware Sateliot NTN

ENTORNO DE DESARROLLO REQUERIDO

Ecosistema Nordic Semiconductor

- **Framework**: Zephyr RTOS + nRF Connect SDK
- **Hardware Target**: nRF9151 Development Kit
- **Toolchain**: ARM GCC + Nordic tools
- **IDE**: VS Code + nRF Connect Extension

INSTALACIÓN DEL ENTORNO DE DESARROLLO

Paso 1: Descargar e Instalar nRF Connect SDK

Opción A: Usando nRF Connect for Desktop (RECOMENDADO)

1. **Descargar nRF Connect for Desktop**:

   https://www.nordicsemi.com/Products/Development-tools/nRF-Connect-for-desktop

2. **Instalar Toolchain Manager**:
   - Abrir nRF Connect for Desktop
   - Instalar "Toolchain Manager"
   - Instalar nRF Connect SDK v2.6.1 o superior
3. **Instalar VS Code Extension**:
   - Instalar "nRF Connect for VS Code"
   - La extensión instalará automáticamente las dependencias

Opción B: Instalación Manual

Windows (usar PowerShell como administrador):

powershell

```
# 1. Instalar Python 3.8+
winget install Python.Python.3.11

# 2. Instalar Git
winget install Git.Git

# 3. Clonar nRF Connect SDK
git clone --recursive https://github.com/nrfconnect/sdk-nrf.git
cd sdk-nrf
git checkout v2.6.1

# 4. Instalar dependencias Python
pip install -r scripts/requirements.txt
pip install -r zephyr/scripts/requirements.txt
```

Linux/macOS:

bash

```
# 1. Instalar dependencias del sistema (Ubuntu/Debian)
sudo apt update
sudo apt install git cmake ninja-build gperf ccache dfu-util device-tree-compiler wget \
    python3-dev python3-pip python3-setuptools python3-tk python3-wheel \
    xz-utils file make gcc gcc-multilib g++-multilib libsdl2-dev

# 2. Clonar y configurar
git clone --recursive https://github.com/nrfconnect/sdk-nrf.git
cd sdk-nrf
git checkout v2.6.1
pip install -r scripts/requirements.txt
pip install -r zephyr/scripts/requirements.txt
```

Paso 2: Instalar nRF Command Line Tools

Descargar desde:

```
https://www.nordicsemi.com/Products/Development-tools/nrf-command-line-tools
```

Verificar instalación:

bash

```
nrfjprog --version
```

Paso 3: Configurar Variables de Entorno

Windows (PowerShell):

```powershell
$env:ZEPHYR_BASE = "C:\ncs\v2.6.1\zephyr"
$env:PATH += ";C:\ncs\v2.6.1\toolchain\bin"
```

Linux/macOS (bash):

```bash
export ZEPHYR_BASE=/opt/nordic/ncs/v2.6.1/zephyr
export PATH=/opt/nordic/ncs/v2.6.1/toolchain/bin:$PATH
```

ESTRUCTURA DEL PROYECTO

Crear Directorio del Proyecto

```
sateliot_ntn_firmware/
├── CMakeLists.txt
├── prj.conf
├── .gitignore
├── boards/
│   └── nrf9151dk_nrf9151.overlay
└── src/
    └── main.c
```

Archivo CMakeLists.txt (CREAR ESTE ARCHIVO)

```cmake
# CMakeLists.txt
cmake_minimum_required(VERSION 3.20.0)

find_package(Zephyr REQUIRED HINTS $ENV{ZEPHYR_BASE})
project(sateliot_ntn_firmware)

target_sources(app PRIVATE src/main.c)
```

Archivo prj.conf Mejorado

Basado en la revisión del código, se recomienda la siguiente configuración actualizada:

conf

```
# CONFIGURACIÓN nRF9151 - VERSIÓN MEJORADA

# --- Logging ---
CONFIG_LOG=y
CONFIG_LOG_DEFAULT_LEVEL=3
CONFIG_LOG_BACKEND_RTT=y
CONFIG_USE_SEGGER_RTT=y

# --- LTE y Módem ---
CONFIG_LTE_LINK_CONTROL=y
CONFIG_LTE_AUTO_INIT_AND_CONNECT=n
CONFIG_NRF_MODEM_LIB=y

# --- Comandos AT ---
CONFIG_AT_CMD_PARSER=y
CONFIG_AT_MONITOR=y
CONFIG_AT_MONITOR_HEAP_SIZE=1024
CONFIG_MODEM_KEY_MGMT=y

# --- Red y Sockets ---
CONFIG_NETWORKING=y
CONFIG_NET_NATIVE=y
CONFIG_NET_SOCKETS=y
CONFIG_NET_SOCKETS_POSIX_NAMES=y

# --- Localización / GNSS ---
CONFIG_LOCATION=y
CONFIG_LOCATION_METHOD_GNSS=y
CONFIG_NRF_MODEM_GNSS=y

# --- Gestión de Energía ---
CONFIG_PM=y
CONFIG_PM_DEVICE=y
CONFIG_LTE_PSM_REQ=y
CONFIG_LTE_EDRX_REQ=y

# --- Watchdog (CORREGIDO) ---
CONFIG_WDT=y
CONFIG_WDT_NRF=y

# --- Memoria ---
CONFIG_HEAP_MEM_POOL_SIZE=16384
CONFIG_MAIN_STACK_SIZE=8192
CONFIG_SYSTEM_WORKQUEUE_STACK_SIZE=4096
CONFIG_ISR_STACK_SIZE=2048
```

```
# --- Debug (deshabilitar en producción) ---
CONFIG_ASSERT=y
CONFIG_DEBUG=y
```

PREPARACIÓN DEL HARDWARE

Configuración del nRF9151DK

1. **Conectar el Kit**:
   - Conectar nRF9151DK via USB a la PC
   - Verificar que aparezca como dispositivo J-Link
2. **Verificar Conexión**:

   bash

   ```
   nrfjprog --ids
   # Debe mostrar el serial number del kit
   ```

3. **Antenas (CRÍTICO para Sateliot)**:
   - **GPS**: Conectar antena GPS al conector GPS
   - **Cellular**: Conectar antena LTE al conector NB-IoT/LTE-M
   - **Ubicación**: Colocar antenas con vista al cielo despejada
4. **SIM Card**:
   - Insertar SIM de Sateliot en el slot del nRF9151DK
   - Asegurar que esté habilitada para NTN

COMPILACIÓN Y FLASHEO

Método 1: Usando VS Code (RECOMENDADO)

1. **Abrir VS Code con nRF Connect Extension**
2. **Crear Nueva Aplicación**:
   - Ctrl+Shift+P → "nRF Connect: Create a new application"
   - Seleccionar "Copy a sample" → "Browse samples"
   - Crear proyecto base y reemplazar archivos
3. **Configurar Board**:
   - En nRF Connect panel: Add Build Configuration
   - Board: `nrf9151dk_nrf9151`
   - Configuration: Release o Debug
4. **Build**:

- Click en "Build" en nRF Connect panel
- O usar: Ctrl+Shift+P → "nRF Connect: Build"

5. **Flash**:
   - Click en "Flash" en nRF Connect panel
   - O usar: Ctrl+Shift+P → "nRF Connect: Flash"

Método 2: Línea de Comandos

```bash
# 1. Navegar al directorio del proyecto
cd sateliot_ntn_firmware

# 2. Configurar build
west build -b nrf9151dk_nrf9151

# 3. Flashear el firmware
west flash

# 4. Ver logs en tiempo real
west logs
```

Método 3: Usando nRF Connect Programmer

1. Abrir nRF Connect Programmer
2. Seleccionar Device (nRF9151DK)
3. Load HEX file: `build/zephyr/zephyr.hex`
4. Write para flashear

CONFIGURACIÓN ESPECÍFICA PARA SATELIOT

Configuraciones Obligatorias Antes del Primer Uso

1. **Configurar IP del Servidor VAS**: En `src/main.c`, línea aproximadamente 237, modificar:

```c
// CAMBIAR ESTA LÍNEA:
strncpy(config.server_ip, "your.vas.server.ip", sizeof(config.server_ip) - 1);

// POR:
strncpy(config.server_ip, "IP_DE_TU_SERVIDOR_VAS", sizeof(config.server_ip) - 1);
```

2. **Configurar Puerto del Servidor** (si es diferente a 17777):

c

```
config.server_port = 17777; // Cambiar si es necesario
```

3. **Configuración Dinámica del Servidor (Recomendado)**: Para hacer la configuración más flexible, agregar al inicio de `main.c`:

c

```
#ifndef SATELIOT_VAS_SERVER_IP
#define SATELIOT_VAS_SERVER_IP "10.0.0.1"  // IP por defecto
#endif

#ifndef SATELIOT_VAS_SERVER_PORT
#define SATELIOT_VAS_SERVER_PORT 17777      // Puerto por defecto
#endif
```

Y en `initialize_sateliot_config()`:

c

```
strncpy(config.server_ip, SATELIOT_VAS_SERVER_IP, sizeof(config.server_ip) - 1);
config.server_port = SATELIOT_VAS_SERVER_PORT;
```

4. **Compilar con Configuración Específica**:

bash

```
west build -b nrf9151dk_nrf9151 -- -DSATELIOT_VAS_SERVER_IP="192.168.1.100" -DSATELIOT_VAS_SERVER_PORT=
```

Configuraciones para Producción

Para entorno de producción, modificar `prj.conf`:

conf

```
# Deshabilitar debug para producción
CONFIG_ASSERT=n
CONFIG_DEBUG=n

# Mantener logging pero nivel más bajo
CONFIG_LOG_DEFAULT_LEVEL=2

# Habilitar optimizaciones
CONFIG_SIZE_OPTIMIZATIONS=y
```

DEBUGGING Y MONITOREO

Ver Logs en Tiempo Real

Opción 1: VS Code Terminal

```bash
west logs
```

Opción 2: Serial Terminal Manual

Encontrar Puerto:

- Windows: Buscar en Device Manager → Ports (COM & LPT)
- Linux: `ls /dev/ttyACM*` o `dmesg | grep tty`
- macOS: `ls /dev/tty.usbmodem*`

Configuración Serial: 115200 baudios, 8N1

Windows (PowerShell):

```powershell
mode COM3: baud=115200 parity=n data=8 stop=1
```

Linux:

```bash
sudo minicom -D /dev/ttyACM0 -b 115200
```

Opción 3: nRF Connect Serial Terminal

- Abrir nRF Connect for Desktop
- Instalar "Serial Terminal"
- Conectar al puerto del nRF9151DK

Comandos AT de Debugging

Para verificar el estado del sistema, puedes usar estos comandos:

bash

```
# Verificar estado del módem
AT+CFUN?

# Ver información de red
AT+COPS?

# Estado de GNSS
AT%XGPS=1,1,1,60

# Ver configuración de banda
AT%XBANDLOCK?

# Ver coordenadas GPS configuradas
AT%XSETGPSPOS?

# Estado de PSM
AT+CPSMS?

# Estado de eDRX
AT+CEDRXS?
```

TROUBLESHOOTING COMÚN

Error: "Board not found"

bash

```
# Verificar que el kit esté conectado
nrfjprog --ids

# Si no aparece, revisar driver J-Link
# Reinstalar nRF Command Line Tools
```

Error: "Build failed"

bash

```
# Limpiar build anterior
west build -t clean

# Rebuild completo
west build -b nrf9151dk_nrf9151 --pristine

# Si persiste, verificar variables de entorno
echo $ZEPHYR_BASE
```

Error: "Flash failed"

bash

```
# Reset completo del chip
nrfjprog --eraseall

# Verificar conexión
nrfjprog --pinreset

# Flashear aplicación
west flash
```

Error: "No hay salida de logs"

bash

```
# Verificar configuración de logging en prj.conf
CONFIG_LOG=y
CONFIG_LOG_DEFAULT_LEVEL=3

# Verificar puerto serial correcto
west logs

# Probar terminal manual con diferentes velocidades
```

Error: "GPS no obtiene fix"

- Verificar antena GPS conectada correctamente
- Asegurar vista despejada al cielo
- Esperar hasta 5 minutos para primer fix
- Verificar logs: `GNSS: Fix válido obtenido!`

Error: "Attachment Sateliot falla"

- Verificar SIM Sateliot instalada
- Verificar antena cellular conectada
- Verificar configuración de banda 64
- Revisar logs de comandos AT

VERIFICACIÓN DEL DESPLIEGUE

Checklist Pre-Deployment

Entorno de Desarrollo:

- [ ] nRF Connect SDK instalado y funcionando
- [ ] nRF9151DK conectado y detectado (`nrfjprog --ids`)
- [ ] Variables de entorno configuradas

Hardware:

- [ ] Antenas GPS y Cellular conectadas
- [ ] SIM Sateliot instalada
- [ ] Kit alimentado via USB

Software:

- [ ] IP del servidor VAS configurada en código
- [ ] Puerto del servidor configurado
- [ ] Firmware compila sin errores (`west build`)
- [ ] Flash exitoso (`west flash`)
- [ ] Logs visibles en terminal (`west logs`)

Checklist Durante las Pruebas

Inicialización:

- [ ] "Iniciando firmware Sateliot NTN v3.2..." visible en logs
- [ ] "Configuración Sateliot inicializada" aparece
- [ ] Watchdog configurado correctamente

GPS:

- [ ] GNSS iniciado sin errores
- [ ] GPS fix obtenido: "GNSS: Fix válido obtenido!"
- [ ] Coordenadas GPS actualizadas en logs

Conexión Sateliot:

- [ ] Configuración Nordic exitosa: "Nordic configurado exitosamente para Sateliot"

- [ ] Attachment Step 1 ejecutado
- [ ] Attachment Step 2 completado
- [ ] Red registrada: "Red registrada exitosamente!"

Transmisión de Datos:

- [ ] Datos formateados correctamente
- [ ] Envío UDP exitoso: "Datos enviados exitosamente a Sateliot"
- [ ] Ciclo completado: "Ciclo Sateliot completado"

COMANDOS RÁPIDOS DE REFERENCIA

Desarrollo

```bash
# Build y flash rápido
west build -b nrf9151dk_nrf9151 && west flash

# Ver logs en tiempo real
west logs

# Clean y rebuild
west build -t clean && west build -b nrf9151dk_nrf9151 --pristine
```

Hardware

```bash
# Reset completo del hardware
nrfjprog --pinreset

# Verificar ID del dispositivo
nrfjprog --ids

# Backup del firmware actual
nrfjprog --readcode --out backup_firmware.hex

# Erase completo
nrfjprog --eraseall
```

Debugging

```bash
# Verificar versión del SDK
west --version

# Lista de boards disponibles
west boards | grep nrf91

# Información del build
west build -t menuconfig
```

ESTRUCTURA DE LOGS ESPERADA

Una vez flasheado correctamente, deberías ver una secuencia similar en los logs:

```
*** Booting Zephyr OS build v3.4.0 ***
[00:00:00.000,000] <inf> ntn_app: Iniciando firmware Sateliot NTN v3.2...
[00:00:00.100,000] <inf> ntn_app: Configuración Sateliot inicializada
[00:00:00.200,000] <inf> ntn_app: State transition: 0 -> 2
[00:00:01.000,000] <inf> ntn_app: Esperando fix de GNSS...
[00:00:05.000,000] <inf> ntn_app: GNSS: Fix válido obtenido!
[00:00:05.100,000] <inf> ntn_app: Coordenadas GPS actualizadas: lat=XX.XXXXXX, lon=XX.XXXXXX, alt=XXX.X
[00:00:05.200,000] <inf> ntn_app: State transition: 1 -> 3
[00:00:06.000,000] <inf> ntn_app: Sateliot Attachment Step 1: Esperando Attach Reject...
[00:00:06.100,000] <inf> ntn_app: Configurando Nordic nRF9151 para red Sateliot...
[00:00:07.000,000] <inf> ntn_app: Nordic configurado exitosamente para Sateliot
[00:00:10.000,000] <inf> ntn_app: Step 1 completado (Attach Reject recibido) - procediendo a Step 2
[00:00:10.100,000] <inf> ntn_app: State transition: 3 -> 4
[00:00:11.000,000] <inf> ntn_app: Sateliot Attachment Step 2: Esperando Attach Accept...
[00:00:45.000,000] <inf> ntn_app: Red registrada exitosamente!
[00:00:45.100,000] <inf> ntn_app: State transition: 4 -> 5
[00:00:46.000,000] <inf> ntn_app: Telemetry formatted successfully: XXX bytes
[00:00:46.100,000] <inf> ntn_app: Enviando datos via UDP a servidor VAS: XXX.XXX.XXX.XXX:17777
[00:00:47.000,000] <inf> ntn_app: Datos enviados exitosamente a Sateliot en intento 1.
[00:00:47.100,000] <inf> ntn_app: Ciclo Sateliot completado.
[00:00:47.200,000] <inf> ntn_app: State transition: 5 -> 2
```

DOCUMENTACIÓN ADICIONAL

Referencias Oficiales

- nRF Connect SDK Documentation
- nRF9151 Product Page
- Zephyr RTOS Documentation

Recursos de Sateliot

- Consultar documentación específica de Sateliot para configuraciones de red
- Verificar especificaciones de la SIM card NTN
- Confirmar direcciones IP y puertos del servidor VAS

El firmware está ahora listo para su despliegue en el nRF9151 Development Kit con la red Sateliot NTN.