NVX Sensor Integration - Development Time Estimation

Project Overview

High-speed data acquisition system integrating NVX EEG/biosignal sensor (up to 136+ channels at 10-100kHz) with PC backend and GUI. Parses sensor data and streams via MQTT/Redis for real-time visualization.

Key Assumptions:

- Experienced C/C++ developer with data acquisition background
- NVX DLL and driver provided (Windows)
- Backend API endpoints defined
- Available capacity: ~30h/week
- **Target: 120 hours (single device only)**
- **Multi-device support: +20h**
- **Local time-series database: +20h**

Task Breakdown

1. Requirements & Architecture (12-15h)

- NVX API deep-dive (5h)
- System architecture design (4h)
- Interface definitions (3h)

2. NVX Sensor Integration (25-30h)

- Device management and configuration (10h)
- Continuous data acquisition engine (15h)

3. Data Parsing & Protocol (30-35h)

- Raw data parser (EEG, AUX, status, triggers) (18h)
- Data transformation (voltage conversion, calibration) (10h)
- Protocol serialization (JSON/Protobuf) (7h)

4. Queue Integration (20-25h)

- MQTT/Redis client implementation (12h)
- High-throughput streaming pipeline (10h)
- Topic/channel architecture (3h)

5. Control & Command Interface (15-18h)

- Sensor control API (start/stop, modes, channels) (10h)
- Bidirectional communication (8h)

6. Monitoring & Diagnostics (10-12h)

- Performance monitoring (throughput, latency) (5h)
- Error reporting and logging (5h)
- Health checks (2h)

7. Testing & Integration (15-18h)

- Unit testing (6h)
- Integration and stress testing (7h)
- Backend integration (5h)

8. Documentation & Deployment (8-10h)

- Technical documentation (4h)
- Deployment guides (3h)
- Code review and refactoring (3h)

Summary

Task Category	Hours
Requirements & Architecture	12-15
NVX Integration	25-30
Data Parsing & Protocol	30-35
Queue Integration	20-25
Control Interface	15-18
Monitoring	10-12
Testing	15-18
Documentation	8-10
Base Total	**135-163**
Target (Single Device)	**~120**

Add-Ons

Feature	Hours

Feature	Hours
Multi-Device Support	+20
Local Time-Series Database	+20
Both Add-Ons	**+40**

Multi-Device Support (+20h)

- Device enumeration and management (5h)
- Synchronized data acquisition (6h)
- Device-specific routing (3h)
- Multi-device state management (3h)
- Testing and validation (3h)

Time-Series Database (+20h)

- Database setup & schema design (5h)
- Batch writer implementation (8h)
- Data retention & management (3h)
- Query interface (2h)
- Testing & validation (2h)

Database Options: InfluxDB, TimescaleDB, QuestDB **Expected Storage:** ~50-100 GB/day raw, ~5-10 GB/day downsampled

Timeline Recommendations

Base System (120h / 4 weeks)

- **Week 1** (30h): Architecture, NVX integration, basic acquisition
- **Week 2** (30h): Data parsing, validation, serialization
- **Week 3** (30h): Queue integration, streaming, control interface
- **Week 4** (30h): Testing, integration, monitoring, docs

With Database (+20h / 5 weeks)

Add Week 5 for database integration

With All Add-Ons (+40h / 6 weeks)

Add Weeks 5-6 for database and multi-device

Performance Targets

- **10kHz mode**: 5-10 MB/s (primary focus)
- **Queue latency**: 20-50ms end-to-end
- **Database writes**: 100K-1M points/sec
- **Batching**: 50-100ms per queue message

Risk Factors (+8-20h each)

- Data throughput challenges at 100kHz
- NVX DLL bugs or limitations
- Queue infrastructure problems
- Backend API changes during development
- Database performance optimization

What's Included in Base (120h)

 Single device integration

 Multi-channel parsing (EEG + AUX)

 MQTT/Redis streaming

 Bidirectional control

 Error handling & monitoring

 Core testing & documentation

 10kHz mode optimized

What's NOT Included

 Multi-device support (add-on)

 Local database (add-on)

 GUI development

 Backend API development

 Advanced signal processing

 100kHz mode optimization

Tech Stack

Base System

- **Language**: C++17
- **MQTT**: Paho MQTT C++
- **Redis**: Hiredis or redis-plus-plus
- **JSON**: nlohmann/json
- **Build**: CMake

- **Testing**: Google Test

Database Add-On

- **DB**: InfluxDB 2.x or TimescaleDB
- **Client**: influxdb-cxx or libpqxx
- **Deploy**: Docker container

Key Notes

- Use NVX emulation mode for development without hardware
- Implement circular buffers for memory safety
- Monitor `NVXGetDataStatus()` and `NVXGetErrorStatus()` continuously
- Start with JSON; consider Protobuf if needed
- For database: Use SSD storage, implement write-ahead logging
- Regular integration testing with backend team