Nenner Signal Engine

Project Roadmap & Architecture

Vartanian Capital Management

Generated: February 19, 2026

Project Overview

The Nenner Signal Engine automates signal processing from Charles Nenner's cycle research service. It parses incoming email reports, extracts trading signals, cycle timing data, and price targets, then stores them in a structured database for real-time monitoring and alerting.

4-Layer Architecture

- Layer 1: Email Parser + Database - IMAP ingestion, regex parsing, SQLite storage
- Layer 2: Alert Engine - Proximity alerts, Windows toast notifications, SMS via Twilio
- Layer 3: Dashboard - Plotly Dash real-time signal monitoring
- Layer 4: Intelligence - Convergence analysis, fragility scoring, hit rate tracking

Technology Stack

- Python 3.12+, SQLite (WAL mode), Plotly Dash + dash-bootstrap-components
- xlwings for T1/Excel (LSEG) real-time bridge, yfinance for daily closes
- Windows desktop deployment (single-user RIA)
- GitHub: github.com/SevvyV/NennerEngine

Database Summary

- nenner_signals.db - 18MB SQLite with WAL journaling
- 2,251 parsed emails, 12,108 signals, 2,635 cycles, 617 price targets
- 52 instruments tracked across 9 asset classes
- Date range: November 14, 2018 to February 18, 2026

Package Structure

```
NennerEngine/
|-- nenner_engine.py              # Backward-compatible shim
|-- nenner_engine/                # Main package (8 modules)
|   |-- __init__.py               # Public API re-exports
|   |-- __main__.py               # python -m nenner_engine support
|   |-- instruments.py            # 55 instruments, 9 asset classes
|   |-- parser.py                 # 5 regex patterns, signal extraction
|   |-- db.py                     # Schema, migrations, state machine
|   |-- imap_client.py            # Gmail IMAP, processing pipeline
|   |-- reporting.py              # Status, history, CSV export
|   |-- cli.py                    # argparse with 7 CLI flags
|-- dashboard.py                  # Plotly Dash dashboard
|-- test_nenner_engine.py         # 68 unit tests
|-- nenner_signals.db             # SQLite database
|-- .env                          # Gmail credentials (not in git)
```

CLI Reference

```
python -m nenner_engine --status            # Show all signal states
python -m nenner_engine --backfill          # Pull all IMAP emails
python -m nenner_engine --import-folder .   # Import .eml files
python -m nenner_engine --history GC        # Signal history for ticker
python -m nenner_engine --export            # Export tables to CSV
python -m nenner_engine --rebuild-state     # Rebuild current_state
python dashboard.py [--port 8050] [--debug] # Launch dashboard
```

Key Domain Rules

- Signal cancellation implies reversal: BUY cancelled = effective SELL active
- Cancel level of old signal becomes origin price of the new implied signal
- Trigger level of old signal becomes cancel level of the implied signal
- Price targets are awareness markers, not actionable trade signals
- The state machine walks signal history per ticker using date+id ordering

Current State Table Schema

```
current_state (
    ticker TEXT PRIMARY KEY,
    instrument TEXT, asset_class TEXT,
    effective_signal TEXT,    -- BUY / SELL / NEUTRAL
    effective_status TEXT,    -- ACTIVE
    origin_price REAL,
    cancel_direction TEXT,    -- ABOVE / BELOW
    cancel_level REAL,
    trigger_direction TEXT, trigger_level REAL,
    implied_reversal INTEGER DEFAULT 0,
    source_signal_id INTEGER,
    last_updated TEXT, last_signal_date TEXT
)
```

Implementation Roadmap

COMPLETE

Step 1: Signal State Machine

- Created current_state table as a materialized view pattern
- Implemented compute_current_state() with cancellation = implied reversal logic
- Built migrate_db() for safe, idempotent schema migrations
- Added --rebuild-state CLI command for full state recomputation
- Validated: 10/10 match against Nenner's Feb 18, 2026 published signal state

COMPLETE

Step 2: Test Suite

- 68 unit tests across 8 test classes in test_nenner_engine.py
- Coverage: regex patterns, instrument attribution, state machine transitions
- Coverage: email classification, live database validation
- All 68 tests passing (0.23s execution)

COMPLETE

Step 3: Module Split

- Refactored 1,345-line monolith into 8-module nenner_engine package
- Thin backward-compatible shim (nenner_engine.py) preserved
- Both python nenner_engine.py and python -m nenner_engine work
- All 68 tests pass unchanged after refactor

COMPLETE

Step 4: Plotly Dash Dashboard

- Panel A: Active Signal Board - sortable/filterable DataTable, 35 displayed instruments
- Panel B: Change Log - 7-day recent signal changes with NTC row highlighting
- Panel E: Watchlist Focus - large cards for GC, SI, TSLA, MSFT, BAC
- Stats bar: instrument count, BUY/SELL breakdown
- Dark DARKLY theme, 30-second auto-refresh via dcc.Interval
- Priority sort: stocks first (TSLA, MSFT, NVDA, BAC, AAPL, GOOG), then SLV/GLD
- Hidden from display (still tracked): ag futures, forex, bond futures, UNK
- CLI flags: --port, --db, --debug

Implementation Roadmap (continued)

UPCOMING **Step 5: yFinance + xlwings Price Integration**

- yfinance daily close integration for all tracked instruments
- xlwings bridge to T1/Excel (LSEG Workspace) for real-time intraday quotes
- Price data caching layer (in-memory or lightweight DB table)
- Ticker mapping: Nenner tickers to yfinance symbols (futures rolling, ETFs)

UPCOMING **Step 6: Live Prices in Dashboard + Proximity**

- Wire live/delayed prices into signal board and watchlist cards
- Proximity indicators: how close current price is to cancel/trigger levels
- Visual alerts when price approaches critical levels (color intensity, badges)
- P&L column: current price vs. signal origin price

UPCOMING **Step 7: Alert Engine**

- Windows toast notifications for proximity and signal change events
- SMS alerts via Twilio for critical signals (new signals, level breaches)
- Configurable alert thresholds per instrument
- Alert history logging to database

UPCOMING **Step 8: Price Target Tracking**

- Monitor 617 stored price targets against live prices
- Mark targets as reached with timestamp when price crosses target level
- Dashboard panel showing pending vs. reached targets
- Historical hit rate analysis per instrument

UPCOMING **Step 9: Macro Heatmap + Cycle Calendar**

- Macro heatmap: all instruments at a glance with BUY/SELL color intensity
- Cycle calendar: upcoming cycle turn dates from 2,635 stored cycle records
- Cycle timeline visualization per instrument
- Cross-asset cycle convergence display

UPCOMING **Step 10: Intelligence Layer**

- Convergence scoring: when multiple signals/cycles align on same instrument
- Fragility analysis: signals near cancel levels, implied reversals
- Historical hit rate: how often Nenner signals played out correctly
- Risk-weighted signal strength ranking
- Portfolio-level exposure summary across asset classes

Primary Instruments

Active Trading (Watchlist)

- Gold (GC) - Precious Metals futures
- Silver (SI) - Precious Metals futures
- Tesla (TSLA) - Single Stock
- Microsoft (MSFT) - Single Stock
- Bank of America (BAC) - Single Stock

Context / Monitoring

- S&P 500 (ES), Nasdaq (NQ), Dow Jones (YM) - Equity Index futures
- VIX - Volatility Index
- Bitcoin (BTC), Ethereum (ETH) - Crypto
- Crude Oil (CL), Natural Gas (NG) - Energy futures
- All referenced ETFs: GLD, SLV, GDXJ, TLT, USO, UNG, FXE, BITO, GBTC, ETHE, etc.

Hidden from Dashboard (Still Tracked in DB)

- Agriculture futures: Corn (ZC), Soybeans (ZS), Wheat (ZW), Lumber (LBS)
- Currency pairs: EUR/USD, USD/JPY, GBP/USD, AUD/USD, USD/CAD, DXY, etc.
- Bond futures: 30-Year (ZB), 10-Year (ZN), Bunds (FGBL)