NEMO SYNTHESIS ENGINE - TECHNICAL DOCUMENTATION

Official User & Developer Guide

TABLE OF CONTENTS

---

1. INSTALLATION & SETUP

System Requirements

Minimum

- Python 3.10+
- 200MB disk space
- 50MB RAM
- Windows, macOS, or Linux

Recommended

- Python 3.11+
- 1GB disk space
- 300MB RAM

- SSD storage
- 8GB RAM (for optimal performance)

Installation Methods

Method 1: pip (Recommended)

```
pip install nemo
nemo setup
nemo start
```

Method 2: From GitHub

```
git clone https://github.com/torresjchristopher/nemo.git
cd nemo/nemo/systems/task-screen-simulator
pip install -r requirements.txt
python -m nemo.cli setup
python -m nemo.cli start
```

Method 3: From Release ZIP

```
# Download nemo-v1.0.0.zip from GitHub releases
unzip nemo-v1.0.0.zip
cd nemo
pip install -r requirements.txt
python setup.py install
nemo setup
```

First-Run Configuration

```
nemo setup
```

This interactive wizard configures:

1. **AI Model Selection**
    - Gemini (Google, cloud-based)
    - Claude (Anthropic, cloud-based)
    - Ollama (Local, open-source, offline)

2. **API Credentials** (if using cloud)
    - Gemini API key
    - Claude API key
    - Stored encrypted at ~/.nemo/credentials.json
3. **Button Mapping** (customize key bindings)
    - RIGHT ALT (default: voice input)
    - LEFT ALT (default: TTS output)
    - LEFT ALT + LEFT ARROW (default: REWIND)
    - LEFT ALT + RIGHT ARROW (default: FORWARD)
4. **Audio Preferences**
    - TTS voice (male/female/neutral)
    - Speech rate (0.5x - 2.0x)
    - Audio output device
5. **Privacy Settings**
    - Verification level (basic/full)
    - Log retention (24hrs/7days/never)

---

2. CONFIGURATION

Config File Location

`~/.nemo/nemo_config.json`

Example Configuration

```
{
  "ai_model": "gemini",
  "synthesis": {
    "keystroke_dimensions": 35,
    "temporal_buffer": 300,
    "inference_threshold": 0.75
  },
  "buttons": {
    "voice_in": "KEY_RALT",
    "voice_out": "KEY_LALT",
```

```
    "rewind": "KEY_LALT+KEY_LEFT",
    "forward": "KEY_LALT+KEY_RIGHT"
  },
  "tts": {
    "engine": "google",
    "voice": "female",
    "rate": 1.0,
    "pitch": 1.0
  },
  "security": {
    "audit_level": "full",
    "log_retention": "24h"
  }
}
```

Environment Variables

```
# API Keys (alternative to credentials.json)
export GEMINI_API_KEY="your-key-here"
export CLAUDE_API_KEY="your-key-here"

# Nemo directory (default: ~/.nemo)
export NEMO_HOME="/custom/path"

# Log level (DEBUG, INFO, WARNING, ERROR)
export NEMO_LOG_LEVEL="INFO"
```

---

3. CLI COMMANDS

Core Commands

nemo start

```
nemo start
# Starts the Nemo synthesis engine
# Initializes keyboard hook and begins synthesis
# Runs in foreground (Ctrl+C to stop)
```

nemo stop

```
nemo stop
# Gracefully stops the Nemo engine
# Clears synthesis buffers
# Closes all connections
```

nemo status

```
nemo status
# Shows current running status
# Active synthesis count
# Connected AI model
# Uptime
```

Synthesis Commands

nemo synthesize

```
nemo synthesize --context "What should I type next?"
# Runs synthesis engine
# Returns predicted actions
# Outputs confidence scores
```

nemo rewind —minutes 3

```
nemo rewind --minutes 3
# Infers what happened 3 minutes ago
# Returns synthesis of past actions
# Based on keyboard + screen context
```

nemo forward

```
nemo forward
# Predicts next action
# Returns probability distribution
# Suggests likely next keystroke/action
```

Voice Commands

nemo voice ask "What is the weather?"

```
nemo voice ask "Your question here"
# Sends question to AI butler
# Returns natural language response
# Uses configured AI model
```

nemo tts speak "Text to convert"

```
nemo tts speak "Your text here"
# Converts text to speech
# Plays audio (no storage)
# Returns immediately
```

nemo tts test

```
nemo tts test
# Tests TTS system
# Plays test phrase
# Verifies audio output works
```

Security Commands

nemo security verify

```
nemo security verify
# Runs 8-point security audit
# Checks temp directories
# Checks cache files
# Checks memory state
# Checks log files
# Checks credentials storage
# Checks clipboard
# Checks network traffic
# Returns: PASS/FAIL for each check
```

nemo security report

```
nemo security report
# Generates detailed security report
# Lists all checks performed
# Shows any potential issues
# Saves to file
```

Configuration Commands

nemo config show

```
nemo config show
# Displays current configuration
# Masks sensitive values
# Shows active settings
```

nemo config set

```
nemo config set ai_model gemini
nemo config set tts.voice male
nemo config set buttons.voice_in KEY_F13
# Updates specific configuration
# Applies changes immediately
# Validates input
```

nemo setup

```
nemo setup
# Runs initial setup wizard
# Reconfigures all settings
# Creates ~/.nemo directory
# Generates credentials storage
```

Download & Update Commands

nemo download latest

```
nemo download latest
# Checks GitHub for latest release
```

```
# Shows version info
# Shows release notes
# Checks for updates
```

nemo download install

```
nemo download install
# Interactive installation wizard
# Downloads latest release
# Verifies SHA256 checksum
# Extracts and installs
# Ready to use
```

nemo download update

```
nemo download update
# Checks for newer version
# If available: downloads and installs
# If not: shows "already latest"
# Preserves configuration
```

nemo download status

```
nemo download status
# Shows installed version
# Shows latest available version
# Shows if update available
# Shows installation path
```

Logging & Debugging

nemo logs tail

```
nemo logs tail --lines 50
# Shows last 50 log lines
# Real-time streaming mode (--follow)
# Filter by level: --level ERROR
```

nemo logs clear

```
nemo logs clear
# Removes all log files
# Useful after security audit
# Permanent deletion
```

---

4. ARCHITECTURE

Core Components

KeyboardSynthesizer (9.4K LOC)

Purpose: Builds 35-dimensional keystroke signature

Input:

- Key press events (timing)
- Key release events (pressure)
- Keystroke sequences (patterns)

Output:

- 35-D behavioral vector
- Intent classification (5-D)
- Confidence scores

Never stores: Keystroke history

ScreenAnalyzer (6.8K LOC)

Purpose: Real-time screen context analysis

Input:

- Current window title
- Window application name
- Visible text (when needed)

Output:

- Application type
- Context classification

- Activity inference

Never stores: Screen content

TemporalInference (11.2K LOC)

Purpose: Synthesize past and future

Rewind Process:

1. Get current keyboard signature
2. Get current screen state
3. Infer backward using both
4. Return synthesized "past"

Forward Process:

1. Analyze current patterns
2. Build prediction vector
3. Run through synthesis model
4. Return predicted next action

Never replays: Recorded data

GeminiIntegration (10.1K LOC)

Purpose: Multi-agent AI orchestration

Supported Models:

- Gemini (Google Cloud)
- Claude (Anthropic)
- Ollama (Local, open-source)
- Custom models (bring your own)

Communication:

- Secure API calls
- Streaming responses
- Error handling
- Fallback models

VoiceAssistant (8.0K LOC)

Purpose: RIGHT ALT hotkey integration

Features:

- Speech-to-text conversion
- Query processing
- Response formatting
- Text-to-speech output

Audio Flow:

- Microphone → Buffer (RAM)
- Send to API/Model
- Receive response
- Convert to speech
- Buffer cleared immediately

TTSEngine (12.1K LOC)

Purpose: Text-to-speech with zero persistence

Local Engine (pyttsx3):

- No internet required
- Offline support
- Instant playback
- Lower quality

Cloud Engine (Google):

- Natural-sounding voice
- Requires internet
- Higher quality
- Requires API key

Audio Handling:

- Generated in-memory
- Played directly
- Buffer cleared after playback
- Never written to disk

AudioSecurity (13.0K LOC)

Purpose: Verify zero-storage guarantee

8-Point Audit:

1. Temp directory check
2. Cache directory verification

3. Memory forensics
4. Log file analysis
5. Credential storage check
6. Clipboard monitoring
7. Network traffic analysis
8. Behavioral verification

Command: `nemo security verify`

FourButtonInterface (11.3K LOC)

Purpose: Button detection and routing

Button Mapping:

- RIGHT ALT → Voice In
- LEFT ALT (tap) → TTS Output
- LEFT ALT + LEFT ARROW → REWIND
- LEFT ALT + RIGHT ARROW → FORWARD

Detection Logic:

- Tap: < 200ms (immediate action)
- Hold: ≥ 200ms (activate mode)
- Combo: Multi-key sequences
- Zero latency processing

DownloadManager (12.3K LOC)

Purpose: GitHub release integration and installation

Features:

- Fetch latest release info
- Download with progress
- SHA256 checksum verification
- Automatic extraction
- Installation wizard
- Update checking

Data Flow

```
User Input
    ↓
```

```
KeyboardSynthesizer → 35-D signature
ScreenAnalyzer → Context
    ↓
TemporalInference:
├─ REWIND: synthesize past
├─ FORWARD: predict future
└─ Current: understand now
    ↓
GeminiIntegration:
├─ Process with Gemini
├─ Process with Claude
└─ Process with Ollama
    ↓
Output:
├─ VoiceAssistant (RIGHT ALT)
├─ TTSEngine (LEFT ALT)
├─ Screen display
└─ System action
    ↓
AudioSecurity: Verify no persistence
```

Memory Management

In-Memory Only:

- Keystroke buffers (real-time)
- Screen analysis (current)
- Synthesis results (transient)
- Voice buffers (playback only)
- Chat history (session only)

Never Persisted:

- User behavior data
- Audio recordings
- Keystroke history
- Screen captures
- Personal information

Cleared After:

- Voice input processed
- Synthesis complete

- TTS playback finished
- Session ends

---

5. API REFERENCE

Keyboard Synthesizer API

```
from nemo.systems.task_screen_simulator.keyboard_synthesizer import Keybc

synthesizer = KeyboardSynthesizer()

# Record a keystroke
synthesizer.record_keystroke(
    key='a',
    timestamp=1675000000.123,
    duration=0.045,
    pressure=0.8
)

# Get current signature
signature = synthesizer.get_current_signature()
# Returns: 35-D numpy array

# Detect intent
intent = synthesizer.detect_intent()
# Returns: {
#   'type': 'coding|writing|editing|searching|navigating',
#   'confidence': 0.95,
#   'vector': [...]
# }
```

Screen Analyzer API

```
from nemo.systems.task_screen_simulator.screen_analyzer import ScreenAnal

analyzer = ScreenAnalyzer()

# Analyze current screen
context = analyzer.analyze_current_screen()
# Returns: {
```

```
#   'app': 'VSCode',
#   'window_title': 'main.py',
#   'activity': 'coding',
#   'confidence': 0.92
# }
```

Voice Assistant API

```
from nemo.systems.task_screen_simulator.voice_assistant import VoiceAssis

assistant = VoiceAssistant(api_key='YOUR_GEMINI_KEY')

# Ask a question
response = assistant.ask("What should I do next?")
# Returns: "Based on your current patterns..."

# Stream response
for chunk in assistant.ask_stream("Tell me a story"):
    print(chunk, end='', flush=True)
```

TTS Engine API

```
from nemo.systems.task_screen_simulator.tts_engine import TTSEngine

tts = TTSEngine(engine='google')

# Speak text
tts.speak("Hello, this is Nemo")

# With options
tts.speak(
    "Custom voice output",
    voice='male',
    rate=1.5,
    pitch=1.0
)
```

---

6. SECURITY VERIFICATION

Running Security Audit

```
nemo security verify
```

Output Example:

```
NEMO SECURITY AUDIT - Full Verification
═══════════════════════════════════════

[✓] Temp Directory Clean
    - No audio files found
    - No temp recordings
    - Checked: /tmp, %TEMP%

[✓] Cache Directory Verified
    - No persistent cache
    - Checked: ~/.nemo/cache

[✓] Memory Forensics
    - No keystroke history in memory
    - No screen captures in memory
    - No voice buffers persisting

[✓] Log File Analysis
    - No sensitive data logged
    - No audio references
    - Checked: ~/.nemo/logs

[✓] Credentials Storage
    - Encrypted properly
    - Permissions correct
    - Checked: ~/.nemo/credentials.json

[✓] Clipboard Monitoring
    - Not used for sensitive data
    - Cleared appropriately

[✓] Network Traffic
    - No data exfiltration
    - API calls verified
    - Checked against blocklist
```

```
[✓] Behavioral Verification
    - Synthesis-only confirmed
    - No recording detected
    - Zero persistence verified

═══════════════════════════════════════
RESULT: PASS - All 8 checks succeeded
Nemo is operating with zero data storage.
```

Manual Verification Steps

1. **Check Temp Directories**

```
ls -la /tmp | grep nemo
ls -la %TEMP% | grep nemo  # Windows
# Should return: nothing
```

2. **Check Cache**

```
ls -la ~/.nemo/cache/
# Should be empty or minimal
```

3. **Check Logs**

```
grep -i "audio\|voice\|record" ~/.nemo/logs/*
# Should return: nothing
```

4. **Check Credentials**

```
file ~/.nemo/credentials.json
ls -la ~/.nemo/credentials.json
# Should show: encrypted
```

---

7. TROUBLESHOOTING

Issue: "Keyboard hook not detected"

Cause: Permissions issue or pynput not installed

Solution:

```
pip install --upgrade pynput
# On Linux, may need:
sudo apt-get install python3-dev

# On macOS, may need:
brew install python3
```

Issue: "API key invalid"

Cause: Incorrect or expired API key

Solution:

```
nemo config set ai_model ollama  # Use local instead
# Or update key:
nemo setup
# Follow prompts to enter new API key
```

Issue: "TTS not working"

Cause: Audio device not configured

Solution:

```
nemo tts test
# This will test your audio setup

# If fails, try:
nemo setup
# Reconfigure audio device
```

Issue: "Memory usage high"

Cause: Synthesis buffers growing

Solution:

```
nemo stop
nemo start
# Clears all in-memory buffers

# Or:
nemo config set synthesis.temporal_buffer 300
# Default is 300 seconds (5 minutes)
```

Issue: "Buttons not responding"

Cause: Key mapping conflict or permissions

Solution:

```
# Check current mapping:
nemo config show

# Remap buttons:
nemo config set buttons.voice_in KEY_F12
nemo config set buttons.voice_out KEY_F13

# Test:
nemo start
# Now press F12 instead of RIGHT ALT
```

---

8. DEVELOPMENT GUIDE

Project Structure

```
nemo/
├── core/
│   ├── nemo.py           # Main entry point
│   ├── cli.py            # Command-line interface
│   └── README.md
├── systems/
│   └── task-screen-simulator/
│       ├── keyboard_synthesizer.py
│       ├── screen_analyzer.py
│       ├── temporal_inference.py
```

```
│       ├── gemini_integration.py
│       ├── voice_assistant.py
│       ├── tts_engine.py
│       ├── audio_security.py
│       ├── four_button_interface.py
│       ├── download_manager.py
│       ├── setup_wizard.py
│       ├── requirements.txt
│       ├── setup.py
│       └── README.md
├── README.md
├── LICENSE
└── .gitignore
```

Running Tests

```
# From the nemo directory
python -m pytest tests/

# With coverage:
python -m pytest --cov=nemo tests/

# Specific test:
python -m pytest tests/test_keyboard_synthesizer.py -v
```

Adding New AI Model

1. Extend `GeminiIntegration`:

```
class CustomAIIntegration(AIBase):
    def connect(self):
        # Initialize connection
        pass

    def process_query(self, query, context):
        # Process and return response
        pass
```

2. Register in CLI:

```
nemo config set ai_model custom
```

Type Hints & Code Quality

All code uses Python type hints:

```
def record_keystroke(
    self,
    key: str,
    timestamp: float,
    duration: float,
    pressure: float
) -> None:
    """Record a keystroke event."""
```

Run linting:

```
flake8 nemo/
mypy nemo/
```

---

9. ARCHITECTURE DEEP-DIVE

Keyboard Signature (35-Dimensional)

Timing Dimensions (12-D)

- Average keystroke duration
- Keystroke variance
- Time between key presses (digraph timing)
- Key release to press timing
- Typing speed (wpm)
- Typing consistency

Pressure Dimensions (8-D)

- Key pressure profile
- Pressure variance
- Grip force pattern
- Fatigue detection

- Pressure recovery time
- Acceleration profile

Pattern Dimensions (10-D)

- Key sequence frequency
- Common bigrams
- Common trigrams
- Error rate
- Correction pattern
- Rhythm pattern

Intent Dimensions (5-D)

- Coding vs. writing likelihood
- Editing vs. composition
- Navigation intensity
- Search likelihood
- Composition speed

Synthesis Process

1. **Observation Phase** (real-time)
    - Collect keystroke data
    - Analyze screen context
    - Build current state
2. **Synthesis Phase**
    - Compute distance to known patterns
    - Infer behavioral state
    - Generate predictions
3. **Action Phase**
    - Route to appropriate AI
    - Generate response
    - Output to user
4. **Forget Phase**
    - Clear transient buffers
    - Maintain only patterns
    - Update model weights

Why Zero Storage Works

Traditional Approach:

```
Keystroke → Store → Analyze → Predict
(Can recover if stolen)
```

Nemo Approach:

```
Keystroke → Analyze → Predict → Forget
(Nothing to recover)
```

The key insight: You don't need to store data to learn patterns. Update your model in-place, then discard the raw data.

---

10. PERFORMANCE TUNING

Optimize for Speed

```
{
  "synthesis": {
    "keystroke_dimensions": 35,
    "temporal_buffer": 60,
    "inference_threshold": 0.80
  }
}
```

Reduces: Response time by 30%
Tradeoff: Slightly lower accuracy

Optimize for Accuracy

```
{
  "synthesis": {
    "keystroke_dimensions": 40,
    "temporal_buffer": 600,
    "inference_threshold": 0.85
  }
}
```

Improves: Accuracy by 15%

Tradeoff: Higher latency

Memory Optimization

```
# Reduce temporal buffer
nemo config set synthesis.temporal_buffer 120

# Use Ollama with reduced model
nemo config set ai_model ollama
# Choose: neural-chat (3B) instead of larger models
```

CPU Optimization

```
# Disable some synthesis features
nemo config set synthesis.enable_pressure false
nemo config set synthesis.enable_patterns false
```

Network Optimization

```
# Use local Ollama instead of cloud API
nemo config set ai_model ollama

# Or batch requests:
nemo synthesize --batch-size 10
```

APPENDIX: Quick Reference

Common Commands

```
nemo start              # Start synthesis engine
nemo stop               # Stop engine
nemo status             # Show status
nemo setup              # Initial configuration
nemo config show        # Show configuration
nemo security verify    # Run security audit
```

```
nemo tts speak "text"    # Convert text to speech
nemo voice ask "q"       # Ask AI butler
```

Configuration Files

- `~/.nemo/nemo_config.json` - Main configuration
- `~/.nemo/credentials.json` - Encrypted API keys
- `~/.nemo/logs/` - Application logs
- `~/.nemo/nemo_manifest.json` - Installation manifest

Important URLs

- GitHub: https://github.com/torresjchristopher/nemo
- Website: https://downloadnemo.com
- Issues: https://github.com/torresjchristopher/nemo/issues

NEMO TECHNICAL DOCUMENTATION v1.0.0

For support: https://github.com/torresjchristopher/nemo/issues