/* REVERSE ENGINEERING */

Reverse Engineering Engineer

Anton Nikolaevich Veretenenko

SUMMARY

Reverse engineer specialising in embedded systems and undocumented hardware. When there is no datasheet, build the instrument first: QEMU machine models, firmware dumpers, Ghidra + MCP analysis, and logic-analyser / oscilloscope evidence. 18+ years in embedded, firmware and PCB design; recovered ~3,900 functions from a vendor mask-ROM monolith and made an entire stock firmware observable under emulation, using an AI agent as the analysis driver while reviewing every claim myself.

EXPERIENCE

Total engineering experience — 18+ years.

Reverse Engineering Engineer (AI-first) 2026 – Present

NDA

- Target: Smartlink SL6801 SoC (ARM Cortex-M4F class, Thumb-2/VFP) in Jointbees / SMTLINK / YP3 MP3 players — 128×160 RGB565 display over SPI/DBI (GC9106/ST7735S class), CST726/CST816D touch on I²C, microSD over SDIO, internal KADC/DAC codec, 5-key ADC ladder plus PMU power key, on-die Bluetooth. No public datasheet, SDK or register map: the ~0x5C000 mask ROM is the only authoritative code.
- Built the instrument before the analysis: a QEMU 8.2 machine model (emulator/qemu/smartlink_sl6801.c) covering CPU/SRAM windows, mask ROM and 2 MiB SPI NOR XIP (Puya P25Q16 class), LCD controller with DCS parser (CASET, RASET, RAMWR), touch, SD/MMC with FatFs, audio DMA/codec with a PCM tap to the host, key ADC, PMU/CPLD, RTC and backlight; plus a Python GUI with live framebuffer, touch/key injection, hardware switches, audio recording and screenshots.
- Recovered the boot contract: mask ROM → HLKJ bootloader (RAM 0x00804C00, entry 0x00820001) → FIRM image with header, partition table, CRC16 and a 4 KiB RAM block → XIP application. The stock image is a single monolith with UI, media and Bluetooth interleaved and no separate SDK partition; it is never modified on disk — only volatile in-RAM patches (ViewTask / context-2 wakeups, DSP write stubs) let the single-core model run it.
- Solved the hard paths: context-2 / CPU1 wakeups the single-core model does not provide; the audio path (DMA descriptors, codec registers, DSP stubs) that otherwise stalls at 75 % or returns silence; the key-ADC ladder plus PMU power key whose missing "space available" wakeup deadlocks the page manager; and the vendor Bluetooth host stack talking to an on-die controller through a proprietary register window.
- AI-first analysis loop: I direct an LLM agent that drives Ghidra 12.2 through a forked OGhidraMCP HTTP server — /disassemble, /decompile_function, /create_function, /create_data, /undefine, /create_label, /variables, /data_types, /set_comment, /basic_blocks, /call_graph, /search_bytes, /bookmark, /batch, /undo, /redo, /save_program (18 endpoints added). Destructive operations require confirm=true, a written reason and concrete evidence; every claim is verified against the dump, the emulator or the device.
- Fixed the tooling where it failed: forked smartlink_flash (USB timeouts/retries, dump_chunks.py with 64 KiB chunks, resume and --compare, flash_ranges.py, verify_flash.py) because the bootloader stops answering after ~30 commands while a 2 MiB read needs 512+; and forked OGhidraMCP for function/data creation, undo, batch edits and save_program.
- Result: the stock firmware fully boots and is observable under emulation (UI, media, Bluetooth), ~3,900 functions mapped in the Ghidra database, 22 step-* analysis phases and emulator stages M0–M7 documented, two upstream tool forks published, and a one-click desktop AppImage (Electron + PyInstaller). Stack: QEMU, Ghidra, MCP/LLM, ARM Cortex-M4F, Embedded C, Python, FreeRTOS/LVGL/FatFs (observed), Docker, GNU Make.

Embedded Developer (Freelance) Jan 2014 – Dec 2019 (6 years)

NDA

- Reverse engineering of devices and firmware, communication protocol recovery
- Node emulators/simulators based on originals; various bus converters
- Custom electronics development and PCB design
- Microcontroller programming, mostly MSP430, AVR, Cortex-M
- Wireless sensors with Bluetooth 4 / LE / BLE / Smart / 5 interfaces; MBED framework

Lead Engineer Jan 2008 – Jan 2014 (6 years 1 month)

NDA

- Trading robot automated testing system, Windows and Linux scripting; test-result visualization and reporting
- MetaTrader 4 plugin development in a C-like language
- PHP/Symfony backend and HTML5/Canvas/JavaScript frontend

SKILLS

Reverse Engineering

Ghidra, Decompilation, Disassembly, Xrefs, Call Graph, Function Recovery, Data Types, Patch Analysis, Static Analysis, Dynamic Analysis, Firmware Dumping, Bootloader Analysis, Protocol Recovery

Emulation & Instrumentation

QEMU, Custom Machine Model, GDB, Framebuffer Capture, PCM Audio Tap, SDIO/MMC, I²C, SPI, DMA, Logic Analyzer, Oscilloscope

Hardware & Firmware

ARM Cortex-M4F, Thumb-2/VFP, Mask ROM, SPI NOR XIP, USB Boot ROM, FreeRTOS, LVGL, FatFs, Embedded C, Python, GNU Make, Docker

AI-assisted Workflow

MCP, LLM Agents, Prompt Engineering, Tool Forks & Extensions, Evidence-based Verification, Batch Automation, Git

CERTIFICATES

2012

MITx 6.002x: Circuits and Electronics

MIT (edX)

Verify ↗ (https://verify.edx.org/cert/317037ee95ed49ebb6142ef20f527706) PDF ↗ (https://veretenenko.ru/anton/mitx-6002x-certificate.pdf)

2014

MITx 6.00.1x: Introduction to Computer Science and Programming Using Python

MIT (edX)

Verify ↗ (https://verify.edx.org/cert/07e1b2377c044bb8aec8b55c19418eb2)

2014

MITx 6.01x: Introduction to EECS I — "Shape the World"

MIT (edX)

Verify

LANGUAGES

Russian — Native, English — B2 (Upper-Intermediate)

Contact

EMAIL
anton@veretenenko.ru

TELEGRAM
@egaistek

GITHUB
github.com/anton-veretenenko