Tom Schouten

Extended CV
PDF (http://zwizwa.be/tom/cv.pdf)
Email (mailto:tom@zwizwa.be)
Web (http://zwizwa.be)

I provide wide spectrum embedded software development services. I specialize in early stage R&D work setting up an initial architecture and implementation.

Skills

Fields of Expertise

- Embedded Software Development (Linux, bare metal uC, FPGA)
- Numerical Computing and Digital Signal Processing
- Test Automation
- Code Rescue

Programming and Software Design

- Full cycle software development: domain study, problem analysis, requirements gathering, prototyping, design evolution, testing and debugging, continuous deployment, refactoring in response to requirement changes, and long term maintenance
- Code rescue: simplify and refactor a "ball of mud" prototype into something that can be evolved over time
- Object Oriented Design
- Functional Programming
- Domain Specific Languages for modeling, validation and code generation
- Testing and debugging: strong intuitive feel of how things go wrong, and where to focus effort while testing
- Web-based user interface development with focus on embedded and real-time
- Build automation (make, redo) and source control (git, darcs, SVN, CVS)
- Broad understanding of algorithms and eager to learn new tricks
- Mapping a problem across the platform stack: FPGA (FSM + CPU design), assembly, C, OS layers, high level languages, code generation from high level models.
- Multithreading: shared resource & locking, dataflow, scheduling, RTOS implementation, message passing and actor systems.
- Reverse engineering
- Code review
- Development cycle optimization in complex embedded systems: how to improve productivty through immediate edit-compile-run feedback.
- Managing technical debt and "Duct Tape Programming"

Programming languages

- Active non-trivial use: C, Forth, Lisp/Scheme, Erlang, Rust, Haskell, Bash, SQL, Nix, JavaScript, GNU Make.
- Past non-trivial use: Lua, C++, OCaml, Java, Objective C, Perl, Python, Matlab, Pure Data, Latex, MyHDL, Assembler for PIC, x86, TMS320C40, ARM, PRU.
- Explored: Common Lisp, Prolog, Smalltalk, TCL, APL, J, Oz, Fortran, Fortress, Faust, Chuck, SuperCollider, Occam, VHDL, Assembler for AVR, MIPS, PPC, Z80, HC11, TMS320C6x, ...
- Created: PF, SC, Staapl, DSPM, RAI, Seq, PRU and the minimal Seq control CPU instruction set.

Language / Runtime / Compiler design

- Lifelong student of programming languages, runtimes and compiler design.
- Language implementation in C, Forth, Scheme (Racket), Haskell and OCaml.
- Implementation of minimalistic RTOS for bare-bones microcontrollers (Staapl on PIC, uc_tools on Cortex M),
- Virtual Machines for soft real-time Audio/Video systems (PDP,PF,libprim on PC,Android).
- Implementation of bare-bones state machine host based on GDBSTUB (uc_tools).

Operating systems and environments

- GNU/Linux, current focus on Debian, Nix, Buildroot and OpenWRT
- Linux APIs: Linux audio/video APIs, OpenGL, Posix
- Linux kernel and the Xenomai real-time extension
- Erlang/OTP, rebar, and cross-language interfaces (ports and ETF)
- Experience with Android, OSX, eCos RTOS (ARM), MS Windows
- The Web platform: server side HTTP, SQL and client side HTML, CSS, JavaScript, WebSockets.
- Haskell on Nix (cabal2nix)
- Rust and Cargo
- GNU binutils/GCC/GDB toolchain
- GNU Make, redo, and incremental build + test + deployment system design
- Soft and hard real-time systems
- Bare metal microcontrollers and DSPs, FPGAs
- Low-level bit twiddling, digital logic
- Device driver implementation
- Virtualization: VmWare, Qemu/KVM, LXC
- Linux system administration: Apache, MySQL, Asterisk, Exim, OpenVPN, SSH, Borg backup ...

Numerical and Signal Processing

- Continuous and discrete time signal processing theory, Laplace, Z-transform
- Linear algebra, FFT, SVD, Wavelets
- FIR and IIR filter design and implementation
- Noise reduction, echo canceling
- Control theory for linear and non-linear systems, Kalman filters
- Optimization theory, statistics, information theory, communication and digital codes
- 2D/3D computer graphics,
- Basics in software-defined radio, classical mechanics, computer vision and generic numerical algorithms
- Good understanding of audio effects and post-production processing, sound synthesis
- Fixed point and floating point numerical code implementation
- Bridge to implementation and testing: abstract interpretation, automatic differentiation, and code generation for C, assembly, Verilog targets.

Electronics

- Basic familiarity with analog and digital electronics, theory and practice
- Simple analog and digital circuit design and PCB layout (gEDA suite)
- Circuit debugging using oscilloscope, logic analyzer and test code
- FPGA digital logic design using MyHDL, Verilog, and in-house Seq Haskell eDSL

Team Integration

- I've had the honor of working with an diverse set of people from all over the globe. People with different ideas about how to design software, how to run a team, and how to balance code quality, feature set and delivery time.
- I care and I finish what I start.
- Worked in co-located teams, globally distributed teams, loosely knit open source communities and as a solo developer/researcher
- Wrote user manuals, code documentation, informal articles and scientific papers
- Gave presentations and played the role of tutor on development teams and in more formal workshops.
- My mother tongue is Dutch and I speak native-level English with a mixed Flemish / Midwestern accent. I can read and understand spoken French and German for technical content.

Project History

Humanetics ATD (November 2014 - present)

Software design, implementation, and assistance with hardware design for data acquisition systems in a small team. I gained experience with managing the bringup and evolution of the software and digital logic for a fairly complex heterogenous system: FPGA, microcontrollers, application processors, embedded Web server and client side software.

Zwizwa (November 2010 - present)

Solo R&D work. The open source projects are described elsewhere. This interleaves payed contract work.

- Study: Digital and Analog circuit design, Digital Signal Processing theory, Programming Language theory, exploration of various open source tools.
- DSL (Domain Specific Language) projects in Haskell (asm_tools, DSPM) and Racket (Staapl, RAI).
- Development of support libraries for embedded systems: uc_tools, erl_tools
- Development of exo, an IDE written in Emacs and Erlang, taylored to heterogenous distributed systems development (Linux, Cortex M, FPGA and many different languages: C, Rust, Erlang, Haskell, SQL, HTML, CSS, JavaScript).

Emweb / Fike (January 2014 - December 2015)

Design and implementation of a production test framework for a real-time industrial monitoring product. Full software responsibility. Host software in Python, test board dsPIC firmware in C. Tasks included debugging test board hardware, making circuit modifications, original analog/digital test circuit design, creating GUI and database.

Beep (July 2013 - November 2013)

Early stage consulting for an MVNO startup. Tasks involved: exploring telco-related technical specifications, developing ISO7816-4 firmware for the Sysmocom SIMtrace AT91SAM7, and building an early application prototype in Python.

Ubidata N.V. (July 2010 - March 2013)

I developed firmware for an application in the field of fleet management and remote data I/O running on AT91SAM7 and PIC16F. I worked on site for 3 months to get a redesign of an early prototype started, and continued remotely after moving to the U.S. I worked as part of a 3-4 people team of experienced embedded developers. The job spanned the full product life cycle, including the maintenance of a remote embedded system.

Zwizwa (April 2010 - July 2010)

- Development of DSPM, a Haskell eDSL.
- Reactive dataflow framework for sweb in Racket.
- Study: classical mechanics and DSP theory

Sony Techsoft Centre Europe (November 2009 - March 2010)

I was hired by Sony's software division in Brussels for a contracting job requiring Linux system programming experience and signal processing skills. The product was a test system for Sony-specific media framework modifications to the Android OS. This work involved coordination with development departments in Belgium, Japan and China.

Zwizwa (March 2009 - October 2009)

- Continuation of development of Staapl, sweb in Racket.
- Study of programming language theory and implementation.
- Development of libprim in C,Scheme.

Triphase N.V. (October 2008 - March 2009)

Triphase is a Belgium-based startup working in the area of rapid prototyping tools for energy conversion systems. I was hired for Linux/Xenomai driver development for EtherCat and CANbus systems. Aside from that I suggested and implemented some changes to their core product, adding configurability to the architecture. I gained experience with embedded PC hardware, Debian, Linux, Xenomai, field busses, and Matlab/Simulink Real-Time Workshop.

Zwizwa (December 2007 - October 2008)

- Refactoring of the Staapl system based on insights gained from its use in the Waag project, and new insights in the use of Racket as a system for designing domain specific languages.
- A short consulting project involving PF and a computer vision problem
- Development of sweb, weblog software with Latex support used in my development logs

Waag Society (October 2007 - December 2007)

Artist-in-residence program at Waag Society in Amsterdam, institute for art, science & technology. This was a collaboration with Metabiosis/GOTO10. The project consisted of building the platform for an art installation, consisting of a collection of glowing orbs, communicating amongst each other using digital data transmitted over sound waves (OOK, PSL). The system used Staapl and PF.

The Packets Project (October 2005 - October 2007)

A two year project aimed at building tools for media artists. I was at that time part of the GOTO10 media art collective. The project was funded by the School of Art, Design and Architecture at the University of Huddersfield. Work included development of Packet Forth (PF), Staapl, CatKit (a Staapl PIC18F dev board), workshops, and technical support for the Metabiosis media art installation, written in PF.

Zwizwa (April 2003 - October 2005)

Start of solo R&D project work, mostly open source libraries:

- PDP, a video processing extension for Pure Data.
- Creb, a library of audio processing modules.
- PF, a prototyping language for video and computer graphics.
- Staapl (then called Brood), a low-level microcontroller metaprogramming system.

Next to software development, I lead hands-on workshops, and gave invited talks.

Arboretum Systems (October 2001 - March 2003)

Part of a small software development team working on maintaining legacy audio processing software, integrating company IP in third party systems, and developing new audio effects.

SISTA lab, KULeuven (October 1998 - August 2001)

Research assistant in the field of signal processing for musical applications. Topics: Piecewise stationary signal models for audio signal modifications, pitch-synchronous dynamic wave table synthesis, scanned synthesis, subspace based frequency estimation techniques, evaluation of trade-offs involved in FFT window design for additive sinusoidal synthesis, and alias-free synthesis of discontinuous waveforms using discrete state updates to a bank of damped oscillators.

Open Source Projects

This is a chronological list of the projects I've put online, most recent first. The thread that runs through these is the synergy between embedded software, digital signal processing, and development tools focusing mostly on domain specific languages and interactive incremental development.

- asm_tools (http://github.com/zwizwa/asm_tools) Haskell eDSL library including two instrumented macro languages: Seq for sequential logic, and PRU for the AM335x deterministic embedded control processor. It uses a final tagless eDSL representation to implement validation and code generation from the same description. Basically, QuickCheck for machine code and digital logic.
- uc_tools (http://github.com/zwizwa/uc_tools) A collection of C macros for writing malloc-less, static-allocation state machines, and a minimalistic monitor OS based on GDBSTUB.
- erl_tools (http://github.com/zwizwa/erl_tools) Erlang support library for embedded systems applications, with some native C, Rust and JavaScript components.
- studio (http://github.com/zwizwa/studio) An Erlang MIDI studio. It supports incremental development of Audio sythesis and Audio/MIDI processing software.
- RAI (http://zwizwa.be/rai), short for Racket Abstract Interpretation, is a Domain Specific Language (DSL) for analysis and implementation of audio signal processing algorithms. Its main goal is to marry the process of mathematical modeling and the low-level implementation of algorithms.
- DSPM (http://zwizwa.be/darcs/meta/dspm/dspm.html). Precursor to RAI. This project explores the use of tagless-final embedding of a typed DSL into the Haskell programming language, exploring the idea of typed metaprogramming.
- SISO (http://zwizwa.be/darcs/meta/siso/DSPM_RAI.txt). Tagless-final embedding of state space models in Haskell, based on Applicative Functors. Based on ideas behind DSPM and RAI.
- Staapl (http://zwizwa.be/staapl) is a metaprogramming tool based on Forth and Scheme. It uses the Microchip PIC18 architecture. Staapl contains a tiny 8-bit sound synthesizer running in 2K of Flash which serves as test for the compiler, and a driver for generating PAL TV signals, used in the ForthTV workshop[1] (http://www.youtube.com/watch?v=M-UUbm_K9sw) [2] (http://www.youtube.com/watch?v=cNdic7pe5UU).
- libprim (http://zwizwa.be/-/libprim), a library for building small dynamically typed programming languages. Consists of a collection of OS abstractions, basic objects, a simple garbage collector, a Scheme interpreter and a rework of the PF core language and virtual machine.
- sweb (http://zwizwa.be/-/sweb), a Racket-based web application hosting my web logs with support for rendering posts of mathematical nature written in Latex. The design is based on a reactive programming engine.
- Packet Forth (http://zwizwa.be/packetforth) (PF), a stand-alone dynamically typed scripting language taking ideas from Forth and Scheme, aimed at video processing using C-based signal processing code plugins.
- Creb (http://zwizwa.be/pd/creb/), an extension for the computer music system Pure Data (http://http://puredata.info). Creb adds building blocks for music DSP that were missing in Pure Data, or are the result of my research on audio synthesis techniques.
- Pure Data Packet (http://zwizwa.be/pd/pdp/overview.html) (PDP), another extension for Pure Data, which adds building blocks for image and video processing.
- Pyla (http://github.com/zwizwa/pyla) real-time protocol analyzer with support for the Saleae Logic. Written in Python combined with C++/Swig for the analyzer modules, and LARS (http://github.com/zwizwa/lars), a Rust port of Pyla.

For a complete list, see the darcs (http://zwizwa.be/darcs) and git (http://zwizwa.be/git) sections of the web site, which also hosts the associated notes (http://zwizwa.be/-/topics). Some projects are mirrored on github (https://github.com/zwizwa).

Education

1998-2001

Graduate courses and seminars @ KULeuven, Belgium: optimization theory, statistical learning theory, neural networks, system theory, identification and control, numerical algorithms, computer architecture and advanced signal processing.

1993-1998

Master of Electronics Engineering (Burgerlijk Elektrotechnisch Ingenieur) at the KULeuven, Belgium, with a major in Digital Signal Processing.

1987-2017

Autodidacticism. I acquire most of my skills by following curiosity. This is a short summary of the path I followed.

- 1987-1989 MSX BASIC, Z80 assembler
- 1989-1991 MS-DOS, Pascal, DEBUG.COM, 8086 asm & arch
- 1991-1993 Borland Turbo C/C++ & Debugger, 80386 asm & arch, 2D/3D graphics, sound
- 1993-1997 Electronic music, hardware synthesizers, analog & digital electronics
- 1997-1998 Linux: RedHat, Suse, Debian, system administration
- 1998-2000 C++, Matlab, Perl, wrote an audio synthesizer engine
- 2000-2002 Pure Data, computer music & signal processing, GCC/GDB, Linux development, wrote creb
- 2002-2004 Python, emacs, video processing, wrote PDP
- 2004-2006 Forth, Scheme, language & VM design, wrote PF
- 2006-2008 OpenWRT and embedded Linux, Microchip PIC asm & arch, ColorForth, compiler design, wrote Staapl, sweb
- 2008-2010 Scheme (Racket) and programming language theory, wrote libprim
- 2010-2012 Haskell, type theory, and domain specific language design, wrote DSPM
- 2012-2013 back to DSP theory, wrote RAI: a language for DSP algorithm design
- 2013-2014 Python Qt GUI and MySQL, SQLite database, wrote Pyla, Staapl development
- 2014-2017 Erlang, MyHDL (+ VHDL/Verilog), Analog Electronics, RAI & Staapl tweaks, wrote SISO
- 2017-2019 Erlang, Rust, Haskell, Buildroot, digital logic, Web technology. Focus on integration and incremental development of distributed systems.

The most recent version of this document can be found here (http://zwizwa.be/tom/cv.html).