£9.95 | €10.95 (DE) | €13.50 (other EU countries) | $12.95

elektor MAG
design > share > earn

536
* SINCE 1961 *
MARCH & APRIL 2025
ELEKTORMAGAZINE.COM

RISC-V: IT'S IN YOUR FUTURE

Including 16 Bonus Pages by eeNews Europe!

Embedded Industry Trends, R&D and More

FOCUS ON

Embedded & AI

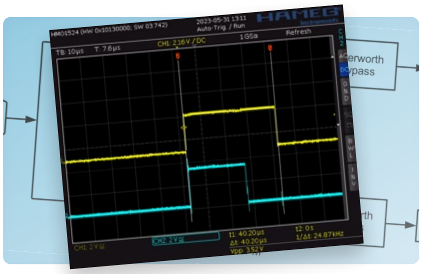
16 Boards & MCUs You Should Know

AI Prevents Damage

Predictive Maintenance in Practice

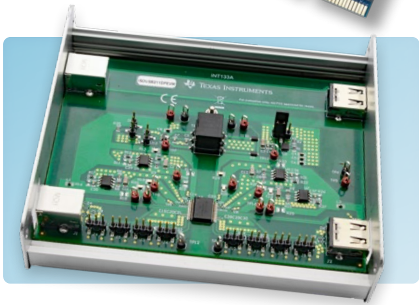
An FPGA-Based Audio Player with Equalizer

Mixing Digital Audio with an Arduino MKR Vidor 4000



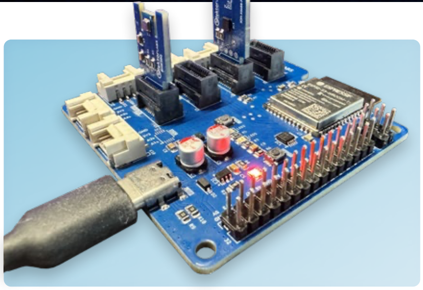
Dual-Core Programming

Parallel Programming with a Raspberry Pi Pico



USB 2.0 Isolator

Electrically Isolated Connections for USB Devices



Modular Sensor Testing

The ESP32-S3-Based Sensor Evaluation Board

04
9 771757 087002

Volume 51, No. 536
March & April 2025
ISSN 1757-0875

Elektor Magazine is published 8 times a year by
Elektor International Media b.v.
PO Box 11, 6114 ZG Susteren, The Netherlands
Phone: +31 46 4389444
www.elektor.com | www.elektormagazine.com

Content Director: C. J. Abate
Editor-in-Chief: Jens Nickel

For all your questions
service@elektor.com

Become a Member
www.elektormagazine.com/membership

Advertising & Sponsoring
Büsra Kas
Tel. +49 (0)241 95509178
busra.kas@elektor.com
www.elektormagazine.com/advertising

Print
Senefelder Misset, Mercuriusstraat 35,
7006 RK Doetinchem, The Netherlands

Distribution
IPS Group, Carl-Zeiss-Straße 5
53340 Meckenheim, Germany
Phone: +49 2225 88010

PEFC Certified
This product is from sustainably managed forests and controlled sources
PEFC
PEFC/30-31-151
www.pefc.org

Jens Nickel

International Editor-in-Chief, Elektor Magazine

RISC-V in Practice

When my former colleague, Mathias Claussen, suggested years ago that I should report on “RISC-V,” I was immediately fascinated. It had been about 25 years since I had last dealt with processor instructions and registers. But, I could instantly see how valuable it was to have a modern, standardized, and freely usable instruction set, developed by the brightest minds from many companies and universities.

Since then, I’ve had the opportunity to learn more firsthand: I met an expert at a symposium who is himself part of a standardization committee, and I also spoke with Calista Redmond, at that time CEO of the RISC-V Foundation, at a trade fair. Today, RISC-V has become an everyday tool for all of us in development. That’s precisely why we decided to make the open-source ISA our cover story (written by Elektor engineers Saad Imtiaz and Jean-François Simon). As a hands-on magazine, we wanted to show that there are now high-quality boards available for every need and performance class, which can be used for developing IoT, audio/video, AI, and many other projects (page 6).

Personally, I’m not particularly fascinated by the high-end boards costing several hundred euros, but rather by the other end of the spectrum. RISC-V makes it possible to use a microcontroller for just 10 cents in your own projects. Here’s a small preview of the next issue: Frequent contributor Tam Hanna will bring such a controller to life in a hands-on article and test how well it works with the associated IDE.

Our colleagues from news platform eeNews Europe also regularly report on RISC-V. In this issue, you’ll find about 16 extra pages from their news portal, including articles created exclusively for us. On page 120, for example, we cover a RISC-V processor that can handle GPU, CPU, and even FPGA functions within a single architecture.

What else is in this Embedded and AI issue? As an audio enthusiast, I highly recommend our FPGA-based audio mixer (page 14). It’s amazing what can be achieved in terms of quality and features with an Arduino MKR board for well under €100. You should also check out the tool we introduce on page 50, which provides developers with valuable insights into what’s happening inside their microcontrollers during program execution. Both articles impressively demonstrate how far you can go with hobby projects.

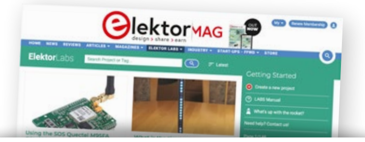
Get involved in development!

elektor MAG
Submit to Elektor!
Articles | Book Proposals | Courses | Projects | Product Ideas | Videos

Submit to Elektor!

Your electronics expertise is welcome! Want to submit an article proposal, an electronics tutorial on video, or an idea for a book? Check out Elektor’s Author’s Guide and Submissions page:

www.elektormagazine.com/submissions



Elektor Labs Ideas & Projects

The Elektor Labs platform is open to everyone. Post electronics ideas and projects, discuss technical challenges and collaborate with others.

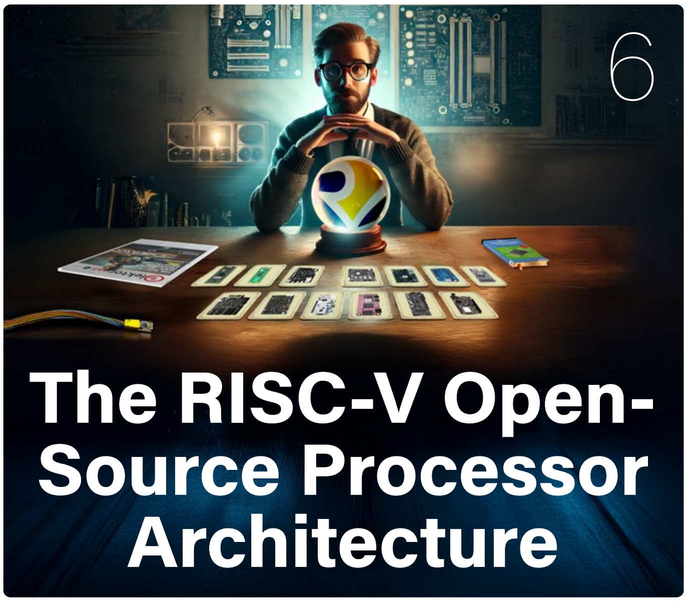
www.elektormagazine.com/labs

The Team

International Editor-in-Chief: Jens Nickel | **Content Director:** C. J. Abate | **International Editorial Staff:** Asma Adhimi, Roberto Armani, Eric Bogers, Jan Buiting, Rolf Gerstendorf (RG), Ton Giesberts, Saad Imtiaz, Alina Neacsu, Dr. Thomas Scherer, Jean-Francois Simon, Clemens Valens, Brian Tristam Williams | **Regular Contributors:** David Ashton, Stuart Cording, Tam Hanna, Ilse Joostens, Prof. Dr. Martin Ossmann, Alfred Rosenkränzer | **Graphic Design & Prepress:** Harmen Heida, Sylvia Sopamena, Patrick Wielders | **Publisher:** Erik Jansen | **Technical Questions:** editor@elektor.com

THIS EDITION

Volume 51, No. 536
March & April 2025



6

The RISC-V Open-Source Processor Architecture

Audio Player with Equalizer

14

Mixing Digital Audio with an Arduino MKR Vidor 4000

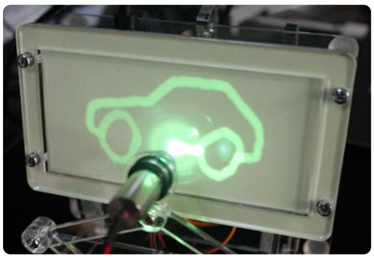
Regulars

Features

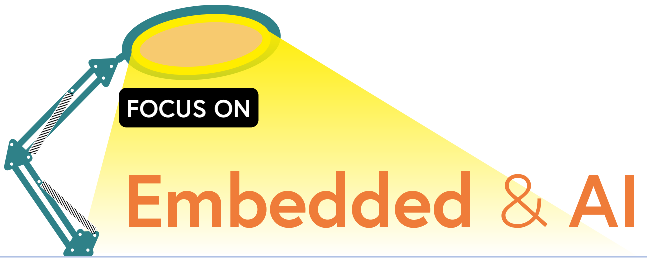
Projects

20

Laser Head for Pico-Based Sand Clock
Drawing with Light



Industry



FOCUS ON

Embedded & AI

Next Editions

Elektor Magazine May & June 2025
As usual, we'll have an exciting mix of projects, circuits, fundamentals, and tips and tricks for electronics engineers and makers. Our focus will be on Test & Measurement.

- MPP Tracker and Charge Controller
- Sine Wave Generator with Adjustable Frequency
- Battery Monitoring System
- GUI for Optimizing PID Parameters
- Frequency Counter
- 10-Cent Controller in Practice
- Stand-Alone Crystal Tester
- FPGA-Based Audio Player: Mixing and Filters

Elektor Magazine's May & June 2025 edition will be published around **May 14, 2025**.

Arrival of printed copies for Elektor Gold members is subject to transport.

BONUS CONTENT

Check out the free Embedded & AI bonus edition of Elektor Mag!

- Virtual Assistant with ChatGPT and Raspberry Pi
- Review: Makerfabs SenseLoRa
- The Connected Autonomous Vehicle and Its Environment
- Peculiar Parts: The Intel 8279 Keyboard/Display Interface
- Infographics: Embedded and AI

www.elektormagazine.com/embedded-ai

The RISC-V Open-Source Processor Architecture

16 Boards and MCUs You Should Know

By Saad Imtiaz (Elektor) and Jean-François Simon (Elektor)

Discover how RISC-V, the open-source newcomer in the microcontroller world, is benefiting many engineers and innovators in the industry with its simplicity and other advantages. Join us as we explore a selection of RISC-V-based boards to experiment with!

Microcontrollers are everywhere, powering everything from your dishwasher to powerful computing systems and millions of wearable and IoT devices. For the processor cores and architectures, names such as ARM, AVR, MIPS, Xtensa, 8051, etc., dominate the landscape, each with unique strengths and areas of application. These platforms, used in many very popular microcontrollers such as the STM32, ESP32-S3, ATmega and so on, share a common trait: They are proprietary.

Enter RISC-V (**Figure 1**), a relatively new player in the world of microcontrollers and processors. Some of our readers may recall that we published a few articles on this topic a few years ago, when hardware choices were much more limited [1][2]. Developed at the University of California, Berkeley, starting in 2010, RISC-V was envisioned as a forward-looking architecture unburdened by legacy compatibility. Unlike traditional processor architectures, RISC-V is open-source, modular, and designed in a modern way. More precisely, it is an open-standard instruction set architecture (ISA), i.e. a standardized definition of the instructions a processor can execute. It is designed to overcome the proprietary restrictions of traditional ISAs, such as those from Intel, AMD or ARM. Unlike proprietary ISAs, RISC-V allows anyone to implement its specifications without legal restrictions, fostering innovation and collaboration between companies and leading experts.

With an open ISA, many companies can develop and sell ready-to-use RISC-V cores, in the form of intellectual property (IP) blocks. A microcontroller manufacturer can buy a RISC-V core and use it in a microcontroller, adding the manufacturer's own peripherals. This encourages competition between IP providers, stimulating innovation and driving down costs. Manufacturers can switch to higher-performance ICs without being locked into proprietary ecosystems.

RISC-V: A Simple Concept

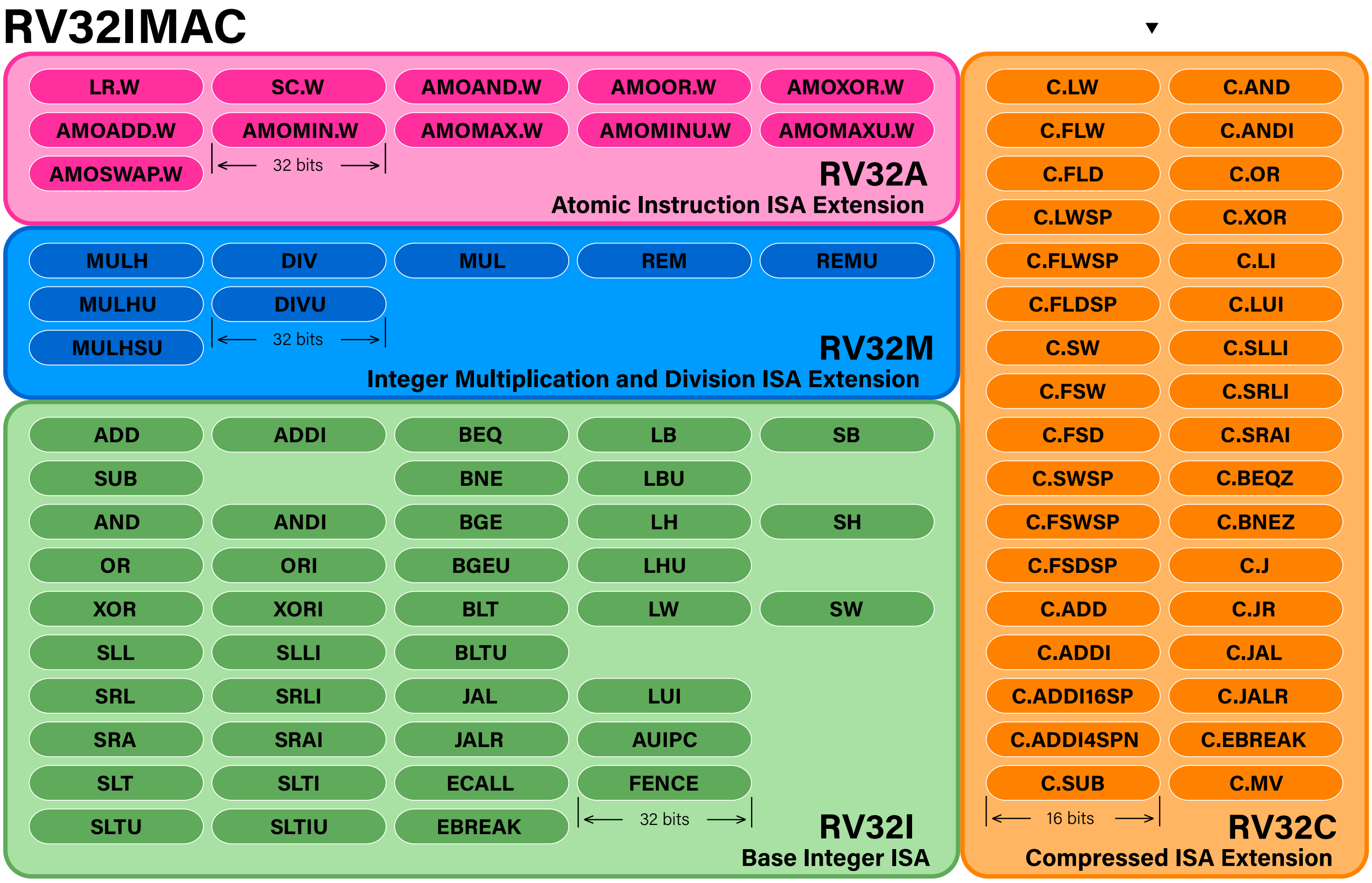
RISC-V, as its name implies, adheres to the Reduced Instruction Set Computer (RISC) principles, which emphasize a small, optimized set of instructions. This reduces the complexity of hardware design and facilitates faster development cycles. Unlike legacy architectures such as x86, which carry decades of backward-compatibility baggage, RISC-V starts with a clean slate, incorporating only what's necessary for modern applications. For example, RISC-V's base consists of just 47 instructions, compared to hundreds on the x86. You can find details about the instruction set here [3], summarized by GitHub user msyksphinz-self. This lean design makes it easier to implement and verify, resulting in lower costs and fewer bugs. While this base instruction set is indeed quite minimalist, there are optional extensions that can be added as needed.

Figure 1: RISC-V logo. Source: © RISC-V Foundation, CC BY-SA 4.0

Modularity

The extensions enable processors to be customized to specific needs. There are about 30 of them, including multiplication and division (M) for arithmetic operations, atomic instructions (A) for multithreaded programming, single and double-precision floating-point (F and D) for scientific computing and signal processing, vector processing (V) for parallel data operations, compressed instructions (C), etc. The full list can be found at [4] along with more detail. This modularity optimizes silicon utilization and energy efficiency: Chip manufacturers can produce microcontrollers that contain just what is needed for a given application, without wasting resources, thus reducing costs. For instance, a microcontroller for IoT devices

Figure 2: Instruction Set for the RV32I base core with M, A, and C extensions. Source: github.com/kuashio/risc-v-diagrams CC BY-SA 4.0



RV32IMAC

RV32A
Atomic Instruction ISA Extension

LR.W	SC.W	AMOAND.W	AMOOR.W	AMOXOR.W
AMOADD.W	AMOMIN.W	AMOMAX.W	AMOMINU.W	AMOMAXU.W
AMOSWAP.W				

RV32M
Integer Multiplication and Division ISA Extension

MULH	DIV	MUL	REM	REMU
MULHU	DIVU			
MULHSU				

RV32I
Base Integer ISA

ADD	ADDI	BEQ	LB	SB
SUB		BNE	LBU	
AND	ANDI	BGE	LH	SH
OR	ORI	BGEU	LHU	
XOR	XORI	BLT	LW	SW
SLL	SLLI	BLTU		
SRL	SRLI	JAL	LUI	
SRA	SRAI	JALR	AUIPC	
SLT	SLTI	ECALL	FENCE	
SLTU	SLTIU	EBREAK		

RV32C
Compressed ISA Extension

C.LW	C.AND
C.FLW	C.ANDI
C.FLD	C.OR
C.LWSP	C.XOR
C.FLWSP	C.LI
C.FLDSP	C.LUI
C.SW	C.SLLI
C.FSW	C.SRLI
C.FSD	C.SRAI
C.SWSP	C.BEQZ
C.FSWSP	C.BNEZ
C.FSDSP	C.J
C.ADD	C.JR
C.ADDI	C.JAL
C.ADDI16SP	C.JALR
C.ADDI4SPN	C.EBREAK
C.SUB	C.MV

might exclude floating-point units to save power and silicon, while a processor for AI workloads would include vector extensions for accelerated computation. As an example, a nice diagram showing the base integer instruction set for a 32-bit core (RV32I) together with the M, A, and C extensions has been put together by Github user kuashio (**Figure 2**).

More Security?

RISC-V's openness has spurred innovation in processor security. If you have an application for which security is important, then the open-source nature of RISC-V is a great feature: It's then easier to inspect. It's for the same reason that many crypto wallets are open-source! Extensions such as CHERI (Capability Hardware Enhanced RISC Instructions) enable fine-grained memory protection, reducing vulnerabilities to attacks such as buffer overflows. Unlike proprietary architectures, RISC-V allows researchers to experiment and implement security features without licensing restrictions.

Reducing Costs and Sharing Results

An open ISA eliminates licensing fees associated with proprietary ISAs. Microcontroller manufacturers can develop their own RISC-V cores or purchase ready-to-use intellectual property (IP) blocks from vendors. This competitive ecosystem drives down costs, making advanced microcontrollers and processors accessible to a wider audience. Further cost reduction is achieved by sharing the development of software ecosystems (compilers, OS support, etc.) between several companies. RISC-V's open model encourages the pooling of resources and expertise, akin to how Linux revolutionized operating systems or Ethernet transformed networking. Companies can focus on unique differentiators rather than duplicating foundational work, accelerating innovation and improving the overall ecosystem.

> *RISC-V's openess has spurred innovation in processor security.*

Legal Peace of Mind for Everyone

For a university, how can you legally teach processor design to computer engineering students when x86 and ARM cores are not open-source? Beyond the legal constraints, there is also a technical challenge: These are not modular, which requires students to implement a massive set of instructions before achieving a potentially functional processor. Semiconductor multinationals also value this legal piece of mind. Have you heard about the legal dispute between ARM and Qualcomm [5]? RISC-V, on the other hand, offers companies a different approach, with no licensing fees.

Adoption by Major Players

RISC-V's open approach has garnered significant attention, with major companies integrating it into their products. By 2015, the RISC-V Foundation was formed, attracting major players such as Google, NVIDIA, Western Digital, and NXP. Over the years, AMD, Qualcomm, IBM, and others have joined, further solidifying its presence in the market. NVIDIA uses RISC-V for specific cores in its GPUs, while Western Digital leverages it for storage devices. SiFive, a pioneer in RISC-V development, offers a range of processors for embedded and high-performance applications. The main IP suppliers are Nuclei, SiFive, and T-Head, while some manufacturers, such as Espressif and WCH, are developing their own IPs to differentiate their products. Without knowing it, you may already have used RISC-V hardware, such as the ESP32-C3, ESP32-C6, and ESP32-P4. Even Raspberry Pi incorporated RISC-V cores into its latest microcontroller, the RP2350, used on the Raspberry Pi Pico 2.

RISC-V in Practice

Despite the hype surrounding it, is RISC-V truly "revolutionary" for the average user? Most engineers and hobbyists program in C/C++ or other high-level languages, meaning you won't need to learn this reduced instruction set. For developers and engineers, transitioning to RISC-V only requires modest changes to established workflows and habits. Tools such as compilers and development environments are already available and getting better every day. If you're interested in embedded development, using RISC-V microcontrollers is a very relevant skill to acquire and add to your toolbox. For those who enjoy getting hands-on and programming in assembly, one of our authors has published a short article [6] on our website about programming the RISC-V core on an ESP32-C3, with a companion Elektor book. For those who prefer programming in C on ultra-low-cost microcontrollers such as the CH32V003 from WCH, we've discovered an excellent educational site [7] created by Vincent Defert. The site is in French, but we encourage you to use a browser extension for real-time translation to take full advantage of this outstanding content! Our bet is that the RISC-V standard is here to stay, and these skills will be easily reusable in the future. In the second part of this article, we will present some of the exciting RISC-V-based development boards available today that you can use for your next project. Have fun!

Notable RISC-V Development Boards

RISC-V development boards have been gaining traction in recent years as the RISC-V ecosystem continues to expand. These boards cater to hobbyists, researchers, and professionals looking to leverage the flexibility and open-source nature of the RISC-V architecture. Below is a detailed look at some of the most notable RISC-V development boards available today, their uses, and potential advantages.

Right now, these are just the beginning of a small selection of RISC-V-based MCUs and CPUs. They range from Arduino Nano-grade MCUs to desktop and laptop-grade CPUs, and many more are expected to emerge in the coming years, reflecting the rapid growth and potential of the RISC-V ecosystem.

240736-01

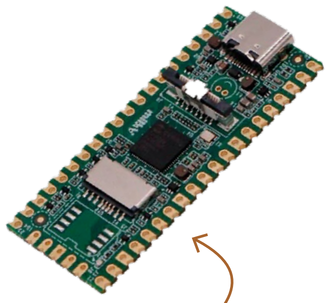
High-Performance RISC-V Processors

Besides single board computers and microcontrollers for embedded electronics, RISC-V processors are already making significant inroads into high-performance computing (HPC) and AI applications. The SiFive Intelligence X280, for example, is optimized for AI and machine learning workloads. It features scalable multi-core configurations, integrated vector extensions, and support for advanced AI operations such as tensor computations. Last year, SiFive also announced their Performance P870-D processor for use in servers and datacenters [8]. For more info on the deployment of Large Language Models (LLMs) on their top-of-the-range hardware with performance benchmarks, read this interesting blog post [9]. Similarly, Alibaba's Xuantie C910 processor, which incorporates AI-specific instructions and supports high data throughput, powers cloud-based AI systems, including real-time inference platforms.

The future of RISC-V in high-performance computing is promising, with several ambitious projects underway. Ventana Micro Systems [10] is developing a server-class RISC-V chip with up to 128 cores per processor, targeting enterprise-level applications such as cloud computing, database management, and large-scale AI training. Their designs prioritize parallelism and energy efficiency, targeting resource-intensive workloads. Another player, Esperanto Technologies, is building the ET-SoC-1, a processor that integrates over 1,000 RISC-V cores onto a single chip [11]. Designed for AI inference tasks, it follows the path of extreme parallelism. Let's see what comes next!

Questions or Comments?

Do you have questions or comments about this article? Email the authors at saad.imtiaz@elektor.com and jean-francois.simon@elektor.com, or contact Elektor at editor@elektor.com.



Related Products

- **Raspberry Pi Pico 2**
  www.elektor.com/20950
- **Milk-V Duo 256M SBC**
  www.elektor.com/20973
- **WCH CH32V307V-EVT-R1 Dev Board**
  www.elektor.com/20448
- **LilyGo TTGO T-Display-GD32**
  www.elektor.com/19510
- **Seeed Studio XIAO ESP32C3**
  www.elektor.com/20265
- **Luckfox Pico Mini B**
  www.elektor.com/21011

WEB LINKS

[1] Stuart Cording, "What Is RISC-V?," elektormagazine.com, April 2021: https://elektormagazine.com/articles/what-is-risc-v

[2] Mathias Claussen, "BL808 and Cohorts: A Look at New RISC-V MCUs," elektormagazine.com, June 2023: https://elektormagazine.com/articles/bl808-and-cohorts-new-riscv-mcus

[3] RISC-V Instruction Set, RISC-V ISA pages, GitHub: https://msyksphinz-self.github.io/riscv-isadoc

[4] RISC-V on Wikipedia: https://en.wikipedia.org/wiki/RISC-V

[5] Arm vs. Qualcomm Dispute, Capacity Media: https://capacitymedia.com/article/arm-pulls-qualcomms-architecture-licence

[6] Warren Gay, RISC-V Assembly Language Programming (Elektor 2022): https://elektormagazine.com/articles/why-risc-v-assembly-language

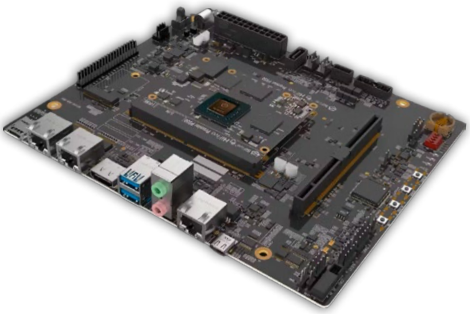
[7] Embedded Development with RISC-V [French]: https://riscv-mcu.defert.com

[8] High-performance RISC-V Datacenter Processor from SiFive: https://sifive.com/press/sifive-announces-high-performance-risc-v-datacenter-processor-for-ai-workloads

[9] LLM Optimization and Deployment on SiFive RISC-V: https://sifive.com/blog/llm-optimization-and-deployment-on-sifive-intellig

[10] Ventana Micro Systems: https://ventanamicro.com

[11] 1,000 Cores on a Chip, VLSIFacts: https://tinyurl.com/et-soc-1-chip

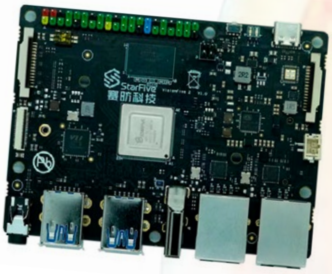


HiFive Premier P550

The HiFive Premier P550 is a high-performance development board designed to push the boundaries of RISC-V development. Powered by the Eswin EIC7700X SoC with a quad-core SiFive P550 CPU, it provides a robust platform for developing and optimizing RISC-V operating systems and applications across diverse markets. It starts at $399 for the 16 GB RAM variant, and can support up to 32 GB of LPDDR5-6400 memory, 128 GB of eMMC storage, and HDMI 2.0 display support, enabling intensive computational tasks. Pre-installed with Ubuntu Linux 24.04, this board is perfect for advanced development in AI, operating system design, and high-performance application.
https://sifive.com/boards/hifive-premier-p550

HiFive1 Rev B

The HiFive1 Rev B is an entry-level board designed for IoT and edge computing, powered by the FE310-G002 processor, which includes a 32-bit RV32IMAC core. Costing around $65, its 16 KB L1 instruction cache, 16 KB data SRAM, and support for flexible clock generation make it efficient for lightweight applications. With a USB debugger upgraded to SEGGER J-Link-OB and compatibility with SiFive Freedom Studio, developers benefit from seamless drag-and-drop flash programming and robust debugging tools. This board is ideal for prototyping IoT devices, developing low-power applications, and exploring the fundamentals of RISC-V development.
https://sifive.com/boards/hifive1-rev-b

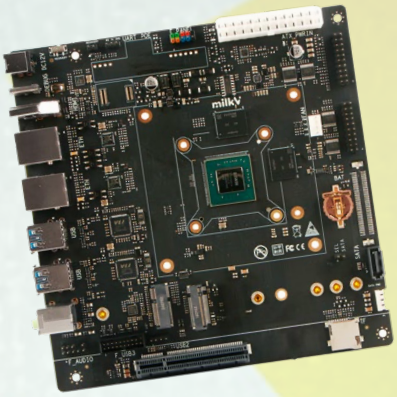


VisionFive 2 SBC

The VisionFive 2 is the world's first high-performance RISC-V SBC with an integrated GPU, powered by the StarFive JH7110 SoC. With a quad-core CPU running up to 1.5 GHz and support for up to 8 GB LPDDR4 memory, it excels in multimedia processing and dual-display output via HDMI and MIPI DSI interfaces. Features such as three USB 3.0 ports, Gigabit Ethernet with PoE, and GPIO headers make it a strong contender for IoT, lightweight servers, and edge computing. Its robust multimedia capabilities, including 4K video decoding and encoding, make it ideal for developers exploring high-performance RISC-V applications in cost-effective projects, which is currently sold on Amazon's website for $99.
https://starfivetech.com/en/site/boards

Milk-V Megrez

The Milk-V Megrez is a Mini-ITX RISC-V device powered by the Eswin EIC7700X SoC, featuring a quad-core SiFive P550 CPU at 1.8 GHz. Its built-in GPU supports advanced graphics standards such as OpenGL ES 3.2 and Vulkan 1.2, while the 19.95 TOPS NPU enables local AI processing for applications in machine learning and robotics. With support for up to 32 GB LPDDR5 memory, multiple storage options, including SATA SSDs and eMMC, and a range of connectivity options like HDMI, USB 3.0 and dual Gigabit Ethernet, this board is ideal for AI development, high-performance computing, and multimedia tasks. Its compatibility with Linux and versatile hardware interfaces make it a significant step forward in RISC-V desktop computing. You can grab this powerful board for $200.
https://milkv.io/megrez



Milk-V Duo 256M

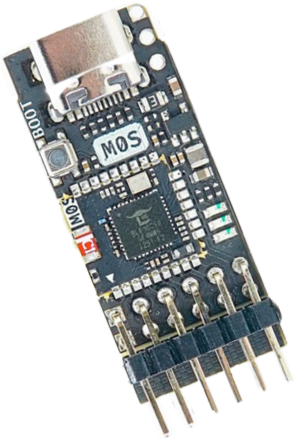
The compact Milk-V Duo 256M is a versatile embedded development platform powered by the SOPHGO SG2002 chip. With a memory boost to 256-MB DRAM, it caters to applications requiring larger memory capacities. The platform features a dual-core RISC-V CPU (C906 at 1 GHz and 700 MHz) alongside a Cortex-A53 Arm CPU, enabling seamless switching between RISC-V and Arm architectures. Its TPU delivers 1.0 TOPS of AI computing power, making it ideal for edge intelligence in smart cameras, visual doorbells, and IoT devices. Rich GPIO interfaces (SPI, UART) and multimedia capabilities like H.265 video encoding, HDR, and noise reduction further enhance its suitability for industrial and smart home applications. The Duo also supports Linux and RTOS, offering developers a powerful and flexible platform for diverse projects. The boards are available for about €30.

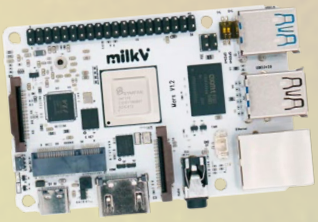
https://milkv.io/docs/duo/getting-started/duo256m

Bouffalo Lab BL616/BL618 and Sipeed M0S

The Bouffalo Lab BL616 and BL618 are 32-bit RISC-V wireless MCUs built for IoT applications. They support Wi-Fi 6, Bluetooth 5.2, and Zigbee, making them ideal for smart home devices and Matter-based automation. Running at up to 320 MHz with an integrated FPU and DSP, they bawlance performance and efficiency. With 480 KB SRAM, embedded flash, and multiple communication interfaces (USB 2.0, SDIO, SPI, I^2S), they are versatile for embedded projects. Their ultra-low-power modes and secure boot features make them well-suited for battery-powered devices requiring reliable connectivity and security. Additionally, Sipeed has launched the compact M0S module based on the BL616. With 4 MB flash, 512 KB SRAM, and USB 2.0 support, this tiny (11×10 mm) module is designed for ultra-low-cost IoT applications. With all these features, a board is available for a modest $4.

https://openbouffalo.org/index.php/BL616
https://wiki.sipeed.com/hardware/en/maixzero/m0s/m0s.html





Milk-V Mars

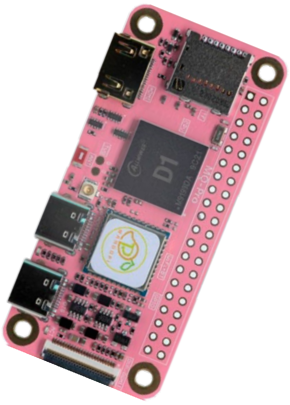
The Milk-V Mars is a compact and high-performance RISC-V SBC powered by the StarFive JH7110 SoC, featuring a quad-core CPU clocked up to 1.5 GHz. It supports up to 8 GB of LPDDR4 memory, an eMMC slot, and SPI flash for bootloader storage, making it highly adaptable for development tasks. With three USB 3.0 ports, one USB 2.0 port, and an HDMI 2.0 output supporting 4K resolution, it is well-suited for multimedia projects, lightweight servers, and general-purpose Linux development. Additional features such as a 40-pin GPIO, PoE-enabled Ethernet, and MIPI interfaces for cameras further enhance its versatility, enabling use in IoT, edge computing, and embedded systems. At around $70 for the 8 GB variant, it's a steal for the performance it delivers.

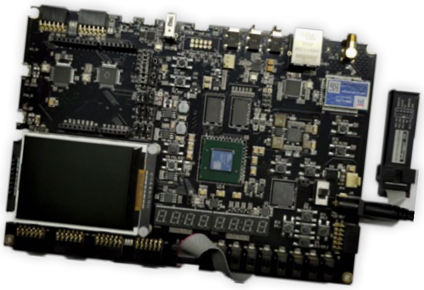
https://milkv.io/mars

MangoPi MQ-Pro SBC

Compact and efficient, this board serves as a viable alternative to the Raspberry Pi Zero, tailored for IoT and lightweight embedded systems. Equipped with the D1 RISC-V core, it supports Tina-Linux/Debian and runs complete Python applications. Its peripheral-rich design includes GPIO, I^2C, SPI, and HDMI, making it ideal for small-scale automation, portable gadgets, and educational projects requiring minimal space and power. Its community-driven ecosystem ensures flexibility and ease of use in diverse lightweight applications. Surprisingly, you can get all these features for as low as $35.

https://mangopi.org/mqpro



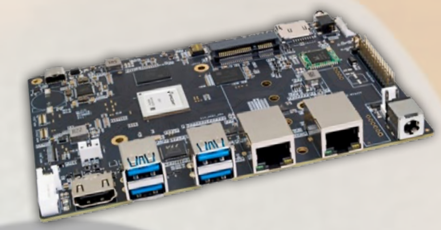


Nuclei DDR200T Development Board

This board by Nuclei System integrates a Xilinx XC7A200T-2 FPGA for hardware acceleration, prototyping, and custom logic development, along with abundant storage and extended interfaces for versatile connectivity. The RISC-V microcontroller, the GD32VF103, enhances programmability, making it ideal for control tasks and interfacing with the FPGA. Its combination of FPGA flexibility and MCU integration supports industrial automation and embedded development. While priced at $770, the board justifies its cost with its advanced features and exceptional performance for demanding applications.
https://nucleisys.com/developboard.php

Banana Pi BPI-F3

The Banana Pi BPI-F3 is an industrial-grade RISC-V development board powered by the SpaceMiT K1 8-core RISC-V processor, which integrates 2.0 TOPS of AI computing power. It offers flexible configurations with 2/4/8/16 GB DDR and up to 128 GB eMMC storage. With dual Gigabit Ethernet ports, four USB 3.0 ports, PCIe for M.2 expansion, and support for HDMI and dual MIPI-CSI cameras, this board excels in advanced prototyping, industrial applications, and AI-driven tasks. Its compatibility with Linux distributions and diverse hardware interfaces makes it ideal for high-performance computing and robust development environments. Available at $70, it strikes a perfect balance between cost and capability.
https://banana-pi.org/en/banana-pi-sbcs/175.html

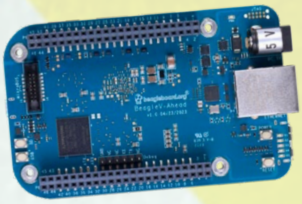


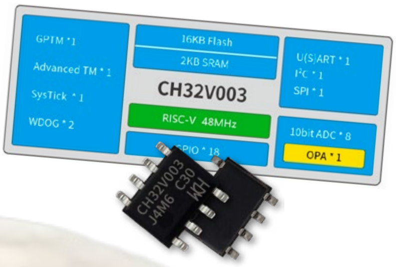
Espressif ESP32 boards

Espressif's RISC-V-based MCUs, including the ESP32-P4, ESP32-C3, and ESP32-C6, are among our favorites and highly favored by the community for their versatility and robust software ecosystem. The latest and greatest ESP32-P4 features a dual-core CPU running at up to 400 MHz, an auxiliary low-power core, and 768 KB of on-chip SRAM with external PSRAM support. It excels in AI, IoT, and HMI applications, boasting 55 programmable GPIOs and extensive peripheral support, including USB OTG 2.0 HS, Ethernet, and MIPI-CSI for high-resolution cameras. With hardware accelerators and media encoding for H.264 at 1080p, it is a top choice for multimedia-rich projects. The wider Espressif RISC-V family offers excellent framework compatibility, making firmware development seamless across multiple platforms. These boards are also budget-friendly, with prices ranging from as low as $3 to $50, depending on the variant and features. One of the best options is the Seeed Studio XIAO ESP32C3, equipped with the ESP32-C3 SoC, combining 400-KB SRAM and 4-MB Flash in a compact thumb-sized design. It is ideal for the IoT, wearables, and low-power networking.
https://espressif.com/en/products/socs/esp32-p4
https://wiki.seeedstudio.com/XIAO_ESP32C3_Getting_Started

BeagleV Ahead

The BeagleV Ahead is an open-source RISC-V SBC powered by the T-Head TH1520 SoC, featuring a 2 GHz quad-core XuanTie C910 processor with advanced GPU and NPU capabilities. Its compatibility with BeagleBone Black cape headers allows for hardware expansion, making it suitable for robotics, AI, and multimedia applications. With support for Linux and open-source frameworks, it is designed to enable developers to explore the potential of RISC-V architecture in complex AI and machine learning projects. For only $150, this SBC punches well above its weight.
https://beagleboard.org/boards/beaglev-ahead





WCH CH32V003 boards

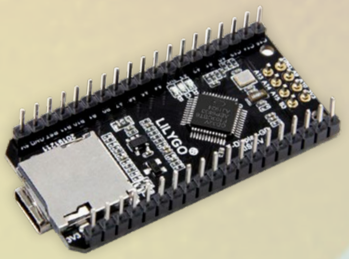
The CH32V003 by WCH is the most cost-effective of this bunch, a 32-bit RISC-V MCU designed for industrial and general-purpose applications. It features a QingKe V2A core running at up to 48 MHz, 16 KB flash, and 2 KB SRAM. With support for multiple low-power modes, it is optimized for energy-efficient operations. It includes a 10-bit ADC, op-amp comparator, and standard interfaces such as USART, I^{2}C, and SPI. The ultra-small package and 1-wire serial debug interface make it ideal for compact embedded systems, automation, and low-power IoT devices. The chip itself costs less than $0.20, and I was even able to spot a CH32V003 development board on AliExpress, being sold for less than $1 — a deal that's hard to resist. By the way: In one of the next editions, author Tam Hanna will try out the CH32V003 and the corresponding IDE.

https://wch-ic.com/products/CH32V003.html

WCH CH32V307V-EVT-R1

The WCH CH32V307 on the CH32V307V-EVT-R1 board is a feature-rich RISC-V microcontroller designed for interconnected applications. It runs at up to 144 MHz, with a single-precision FPU and hardware stack area for improved performance. The controller includes 64-KB SRAM, 256-KB Flash, and a wide range of peripherals, such as eight UART ports, USB 2.0 HS, Ethernet with built-in PHY, and multiple timers. Its GPIOs can be mapped to external interrupts, and it supports ADC, DAC, SPI, and I²C interfaces, making it versatile for industrial automation, real-time data processing, and communication-centric tasks. Its efficient low-power modes and robust connectivity make it a solid choice for advanced embedded systems. You can find the dev board at different suppliers (including the Elektor Store) for around €20.

https://github.com/openwch/ch32v307



GigaDevice GD32VF103CBT6 boards

The GD32VF103CBT6 microcontroller by GigaDevice can be found on development boards like the Sipeed Longan Nano and the LilyGo TTGO T-Display-GD32 RISC-V Development Board (available in the Elektor Store for a discounted price of just €12.95). Both boards are equipped with a small LCD and SD card socket, making all kinds of stand-alone devices possible. The 32-bit RISC-V CPU integrates a Bumblebee Core by Nuclei System, 128-K Flash and 32-K SRAM, an RTC, 3× USART and many other interfaces like USB, I²C, SPI, I²S and CAN.

www.gigadevice.com/product/mcu/main-stream-mcus/gd32vf103-series

Raspberry Pi Pico 2

Raspberry Pi surprised everyone by adding two Hazard3 RISC-V cores to the recent RP2350 powering the Raspberry Pi Pico 2! It offers 520 KB of SRAM, 4 MB of flash storage, 26 multi-purpose GPIO pins, including 4 that can be used for ADC, and a comprehensive set of peripherals including two UART interfaces for serial communication, two SPI plus two I^{2}C controllers and 24 PWM channels. Additionally, the board includes 12 PIO (programmable I/O) state machines and a USB 1.1 controller with PHY supporting both host and device modes. Priced at only $5, the Pico 2 is perfect for learning and experimenting with RISC-V.

https://raspberrypi.com/products/raspberry-pi-pico-2

An FPGA-Based Audio Player with Equalizer (1)

Mixing Digital Audio with an Arduino MKR Vidor 4000

By Dr. Christian Nöding (Germany)

Have you ever wanted to know what is inside a digital audio mixing console? Or have you already programmed some MP3-player based on an ESP32 and the great Arduino libraries, but without the option for processing these audio signals? In this two-part series, we will create a multichannel digital mixing console with multiband equalizers, an audio file player and an analog input. Rock your next party with an expandable DIY system!

This is the first part of a two-part series exploring the design and implementation of an FPGA-based audio player with equalizer functionality, using the Arduino MKR Vidor 4000. In this part, I'll explain the foundational aspects of the project, including the required toolchain, the interplay between FPGA logic and microcontrollers, and an introduction to the basics of audio signal processing. By the end, you'll have a clear understanding of the project and its goals. The full documentation and source files are available online, with links provided at the end of the article.

The Starting Point

For over 10 years, I've been a fascinated user of digital audio mixing consoles without questioning their inner workings. However, when I found a defective digital Behringer P16-I at a flea market, I saw an opportunity to dig deeper and understand how such a device operates.

The P16-I features 16 analog inputs and six Ethernet-like jacks for digital audio output via Cat5e cables, using a protocol called "Ultranet." Ultranet is a slightly modified AES/EBU signal, which we'll explore later in detail.

The Challenge: Building a Versatile Audio Mixer

Studying the P16-I fascinated me and gave me the idea of designing my own audio mixer. My goal was to design an audio system capable of inputting up to 20 channels from various analog and digital sources, processing these signals in real time, and mixing them with advanced audio features like equalization and dynamic control. This project replicates a mixing console's core functionality, and we'll also keep it flexible and see how to expand its functionality.

Choosing the Right Platform

Since the advent of digital audio, signals have been processed at specific bit depths and sample rates. Modern audio, often 24-bit with sample rates starting at 48 kHz, demands high processing power, which

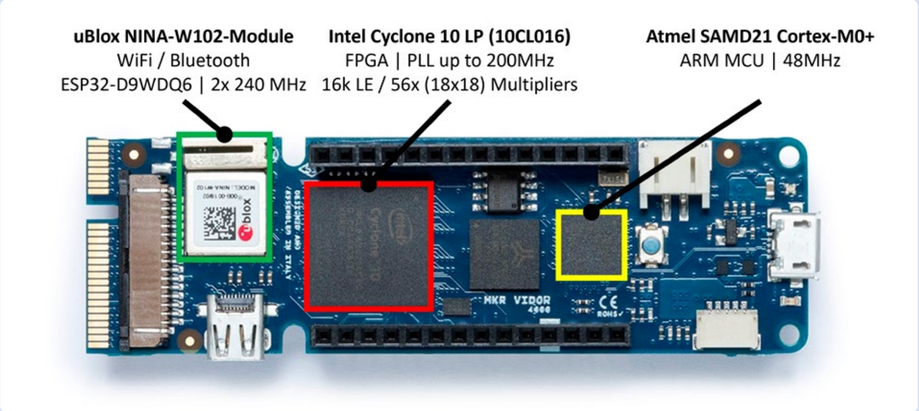


Figure 1: Arduino MKR Vidor 4000.

smaller microcontrollers struggle to manage in real time. This is where digital logic, and particularly FPGAs, excel. So, how to find the right FPGA board for our needs, one that's both user-friendly and appealing in terms of hardware?

While researching an FPGA solution for this project, I came across the Arduino MKR Vidor 4000. This board from Arduino's MKR family combines an ideal set of components for digital audio processing: two microcontrollers, built-in Wi-Fi and Bluetooth, and of course an FPGA — making it a perfect starting point for this project.

Figure 1 shows this Arduino board. Next to the USB port, a SAMD21 microcontroller from Microchip connects to a host computer and acts like a bridge to the onboard FPGA at the same time, as both share pins of the MKR pin header. The NINA W102 module made by U-blox, containing a full-featured dual-core ESP32, connects to the FPGA as well. In its original condition the NINA module contains a special software from Arduino, to communicate via Wi-Fi and Bluetooth. An onboard Flash chip contains a predefined logic from Arduino for the FPGA. But for our audio application we will change the setup of all three devices for our own purpose.

Setting Up the Toolchain

First, we must prepare our toolchain to support all desired devices. For the NINA/ESP32, follow the tutorial at [2] to install the ESP32 package to the Arduino IDE. For the SAMD21 you find the official Arduino sources by searching for "SAMD" in the board manager library. As we are planning to upload our own bitstream to the FPGA, we must download the design software Quartus Prime Lite from Intel [3]. The usual way to upload a new bitstream to these kinds of FPGAs is using a special JTAG adapter. Thankfully, there is an Arduino library under [4] to upload own bitstreams by embedding it into a regular header file of the Arduino sketch directly.

Quartus will output the created logic in several output formats. One of them is the tabular text file, containing the configuration data for use outside of the Quartus software. To feed our JTAG emulation with these data, we must flip the bit order of the file to generate an Arduino-compatible header file. We can use the small program *vidorcvt* from [5] on the command line:

```
vidorcvt.exe < FPGA.ttf > bitstream.h
```

At each boot of the SAMD21, when the function `setup_fpga()` is called within our Arduino sketch, the FPGA is configured with the content of *bitstream.h* copied to the onboard SPI-Flash during flashing the SAMD21. This occurs at every power-up, as FPGAs are losing their logic on a voltage loss.

Mixing C Code and Digital Logic

A digital audio system consists of both time-critical real-time processing, as well as less critical, but more complex calculations based on user inputs. The former use the fast and parallel processing properties of the FPGA; for the latter we benefit from the advantages of microcontrollers to handle complex calculations. **Figure 2** shows the connection of all three devices on the MKR board: the USB port is used for software updates of the USB controller (SAMD21) piggy-backing the bitstream of the FPGA and programming the onboard SPI Flash during the upload-process. Once we have a working FPGA, we can route some communication lines through the FPGA to the NINA/ESP32 to program it via the built-in bootloader. Furthermore, the SAMD21 is used for calculating some nice VU-meter effects and has a lot of free program space and calculation time remaining for your own ideas.

The FPGA will be able to receive digital audio data from a variety of sources: either from a coaxial or optical S/PDIF or TOSLINK jack (stereo), an analog-to-digital converter with stereo I^{2}S output, the ESP32 (with playback from MP3 files on an SD Card) via I^{2}S and finally, the already mentioned 16-channel input from the Behringer Ultranet signal. In this article, I will focus on the digital signal processing aspect of the project. For testing, I used jumper wires for all connections. To read data from a micro SD card, you can use an extension board like the Arduino MKR SD Proto Shield or a similar module. To implement all these audio inputs more neatly in a more advanced version, it would be necessary to create a small motherboard with two rows of pins matching the headers of the MKR Vidor 4000 and equipped with all the appropriate connectors.

Besides processing all the input signals, the FPGA will be equipped with five full-featured parametric equalizers followed by a Linkwitz-Riley Crossover filter to feed a 2.1 stereo set with connected subwoofer — we want to hear and "feel" the sound, right? As mentioned, some calculations are better for a microcontroller. Calculating the filter coefficients for the used IIR filters is certainly a task for a CPU rather than FPGA, but we get back to this in a moment.

Webserver

Another task for the ESP32 is the communication with the environment – specifically with

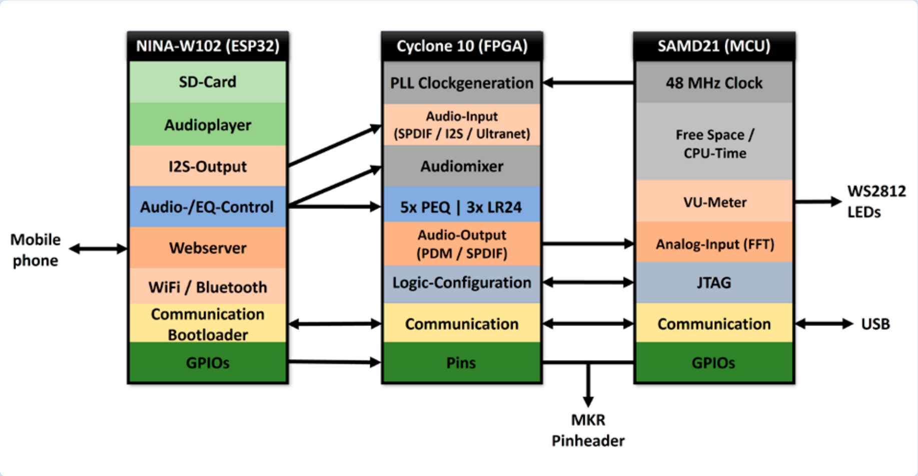


Figure 2: Overview of the multi-controller-system.

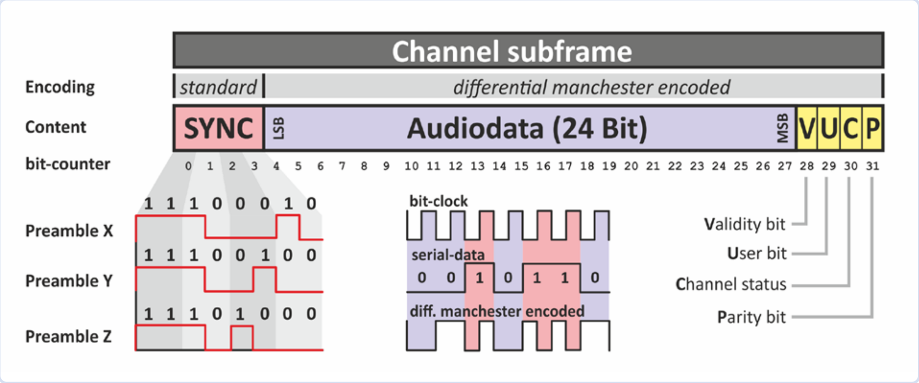


Figure 3: AES/EBU signal containing up to 24 bits of digital audio data.

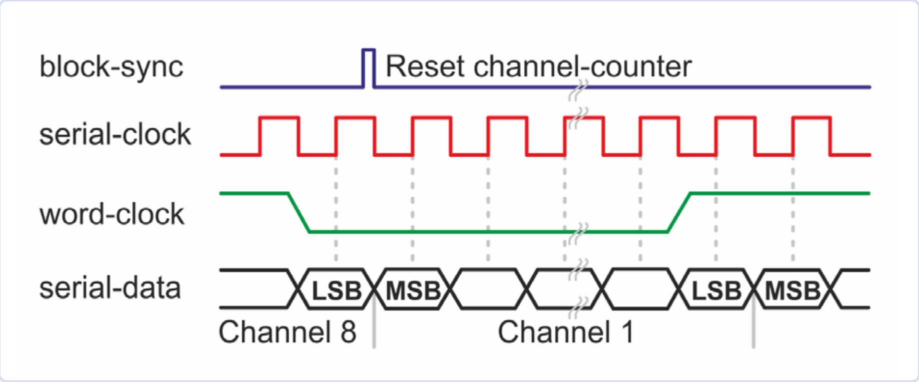


Figure 4: Inter-IC-Sound (I2S) signal.

our mobile phone. As shown in many projects in Elektor, we are using the SD card to load the webpages and we are using the well-known Bootstrap framework to style our interface. With some basic HTML knowledge, it is possible to create a fancy and responsive user interface. A small Javascript within the website is polling some update information using AJAX over a REST-like API fed by the webserver of the ESP32, like current title, progress, volumes, equalizer settings, and more. User inputs are passed to the file *cmd.php*. Even though we are not implementing a full-blown PHP server, the browser will transmit the parameters which our webserver will parse and call the desired functions. For example the call `/cmd.php?mixer:volume:main=-6` will be translated to the C function `executeCommand("mixer:volume:main@-6")` and set the volume to -6 dB. The same command can be sent via USB as well. With this ASCII-based command system, it would also be very easy to add MQTT commands to the firmware, like `fbape/mixer/volume/main -6`, for example.

From Stereo to Multi-Channel

But enough said about web interfaces – our audio data waits to be processed. As FPGAs are great to handle parallel tasks, we can implement the inputs for all desired signals as parallel logic. The idea is, to convert the serial data (mainly S/PDIF, Ultranet and I²S) into individual 24-bit logic vectors that we can handle quite easily. Let's start with S/PDIF and Ultranet. Both use AES/EBU, an interface for digital audio data specified in 1985. It transmits between 16 and 24 bits of stereo audio data with sample rates between 32 kHz and 192 kHz. According to the App Note [6], AES/EBU contains a data structure shown in **Figure 3**, representing a single subframe. Ultranet transmits four stereo channels on two separate lines with 48 kHz, to maintain the maximum sample rate of 192 kHz.

Most of the S/PDIF data is encoded using a differential Manchester encoding, so that the bit clock is transmitted multiplexed with the serial data (see **Figure 3** again). An exception is the sync-block, that contains conventional bits forming the so-called "preamble", an eight-bit pattern. For the preamble three different bit patterns using invalid Manchester codes are specified. That allows us to identify the beginning of channel 1 on receiving the preamble Z which is sent every 192 subframes. The four status bits can be ignored when receiving the pure audio samples.

Now, the first task for the FPGA is to decode the Manchester-coded S/PDIF serial data. Therefore, it must collect the individual bits into a shift-register and search for the three high bits of the preamble. Within a state machine, the synchronization to the signal is monitored and the inherent bit clock of the Manchester code is restored to an internal clock signal. As the space for this article is limited, I'm not going into details of this specific block as I'm using a slightly modified VHDL block from OpenCores.org, a nice page with lots of VHDL snippets [7]. The receiver block outputs a conventional I²S signal, that is much easier to handle.

As Ultranet is using AES/EBU, as S/PDIF, we can convert our individual subframes into an I²S signal with the same receiver block. The only additional task is a channel counter, counting from channel 1 to channel 8 instead of the stereo signals of S/PDIF, synchronized to the Z preamble again.

"Inter-IC Sound," or I²S (see **Figure 4**), uses a fixed serial clock (bit clock) as well as a word-select signal (also called word clock or LR clock) to transmit audio data. So at least three connection lines are necessary. Here we must create a logic, that searches for the falling or rising edges of the word-select as synchronization. So, we can read individual bits into a shift register on each rising edge of the serial clock. One pitfall is, that the word-select is shifted by one clock cycle.

The Digital Logic

So, let's dive into the digital logic (see **Listing 1**). First, we define our input and output signals. Here we need the main clock, which must be much faster as the signal to be read, followed by the other signals, each defined as logic signal as we are reading serial data bits. The output then is defined with two logic vectors with a width of 24 bit as we are processing all audio data with this bit depth within the FPGA. 16-bit signals will be scaled to 24-bit by bit-shifting by 8 bits to the left to keep the 0 dB Fullscale (dBfs) level.

Now we are ready to detect the edges of our three signals. The way to do this is shown in **Listing 2**. In this case, it is the bit clock that is considered, but for the word clock it is comparable. Here we are only reacting on the positive edge and will create a process for this. Within this process, we will use a shift register to detect a change within the serial clock `sclk` by comparing it to the previously sampled `zsclk`. If a positive edge is detected, we set `b_pos_edge` to 1, otherwise to 0.

On each positive edge of `b_pos_edge` we will then store the read bit into a shift register. As in the I²S signal the first received bit is the most significant bit, we must add the current serial bit `sdata` from the right into the register while removing the left most bit (see **Listing 3**).

We can now take the word clock into account to copy the correct bits to the output logic vectors. As the word clock is shifted by one bit clock, the shift register has more bits than the designated payload data. Finally, we remove the least significant 8 bits from the 2× 32-bit buffer and output the desired 24-bits (**Listing 4**).

FPGA Audio Processing

All serial audio data is now within our FPGA represented in handy logic vectors with a bit depth of 24 bits. These bits contain signed audio-samples and will be updated with a sample rate of 48 kHz (or whatever the original signal will use). This represents a fairly large number of signals: assuming an external stereo ADC connected via I²S, the I²S stream from the ESP32's SD card, and the AES/EBU input (which represents between 2 channels in case of S/PDIF and 2 × 8 channels in case of Ultranet), there are between 6 and 20 input signals now. Great!

Going Further

With the toolchain prepared and the foundational concepts of FPGA and microcontroller interaction established, we've also begun our journey into audio signal processing by

Listing 1: Defining the inputs and outputs.

```
entity i2s_32bit_rx is
  port (
    clk      : in std_logic := '0'; -- Mainclock
    sclk     : in std_logic := '0'; -- I2S serial-clock
    sdata    : in std_logic := '0'; -- I2S serial-data
    wordclk  : in std_logic := '0'; -- I2S word-clock

    out_l : out std_logic_vector(23 downto 0) := (others=>'0');
    out_r : out std_logic_vector(23 downto 0) := (others=>'0');
    sync_out : out std_logic := '0' -- high for 1 clock if successful
  );
end i2s_32bit_rx;
```

Listing 2: Syncing with the clock.

```
detect_edge : process(clk)
begin
  if rising_edge(clk) then
    zsclk <= sclk; -- save current serial-clock
    if zsclk = '1' and bclk = '0' then
      b_pos_edge <= '1';
    else
      b_pos_edge <= '0';
    end if;
  end if;
end process;
```

Listing 3: Bit-shifting the received signal.

```
get_data : process(clk)
begin
  if rising_edge(clk) then
    if b_pos_edge = '1' then
      s_shift <= s_shift(s_shift'high - 1 downto 0) & sdata;
    end if;
  end if;
end process;
```

Listing 4: Processing the output data.

```
detect_sample : process(clk)
begin
  if rising_edge(clk) then
    if wclk_neg_edge = '1' then
      signal_l <= s_shift(62 downto 31);
    elsif wclk_pos_edge = '1' then
      signal_r <= s_shift(62 downto 31);
      sync_out <= '1';
    else
      sync_out <= '0';
    end if;
  end if;
end process;
-- take only 24 bits beginning at MSB (signed-bit)
output_l <= signal_l(31 downto 8);
output_r <= signal_r(31 downto 8);
```

converting stereo data into a manageable format for FPGA-based manipulation. In the next part of this series, we'll build on this groundwork to implement advanced features like volume control, parametric equalizers, dynamic compressors, and crossover filters, bringing the project to life as a full-fledged digital audio system. The documentation is available on my GitHub repository [8]. Stay tuned for the second part in the next edition of *Elektor*!

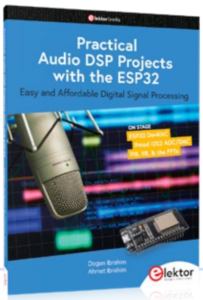
230632-01

Questions or Comments?

Do you have questions or comments about this article? Feel free to contact the author at christian@noeding-online.de, or contact Elektor at editor@elektor.com.

About the Author

Dr.-Ing. Christian Nöding studied electrical engineering at the University of Kassel from 2003 to 2009. He then began working at the department of Power Electronics at the University of Kassel and completed his PhD in 2016. His main interests lie in the design and control of electronic power converters for decentralized energy supply, stage and lighting as well as music and digital audio technology.


Practical Audio DSP Projects with the ESP32
Easy and Affordable Digital Signal Processing
Elektor

Related Products

- **Dogan and Ahmet Ibrahim, *Practical Audio DSP Projects with the ESP32* (Elektor 2023)**
  www.elektor.com/20558
- **Elektor Audio Collection (USB Stick)**
  www.elektor.com/19892

WEB LINKS

[1] The Arduino MKR Vidor 4000: https://docs.arduino.cc/hardware/mkr-vidor-4000/
[2] ESP32 Package for the Arduino IDE: https://docs.espressif.com/projects/arduino-esp32/en/latest/installing.html
[3] Quartus Prime Lite from Intel: https://www.intel.com/quartus
[4] An Arduino library that uploads your custom FPGA bitstream: https://github.com/HerrNamenlos123/JTAG_Interface
[5] Software for Arduino Vidor boards: https://github.com/wd5gnr/VidorFPGA/tree/master/C
[6] Application Note from NTI Audio: https://www.nti-audio.com/Portals/0/data/en/NTi-Audio-AppNote-AES3-AES-EBU.pdf
[7] Simple AES3 or SP/DIF receiver on OpenCores: https://opencores.org/projects/aes3rx
[8] Author's GitHub repository: https://www.github.com/xn--nding-jua

DaVinci Resolve
File
Edit
Trim
Timeline
Clip
Mark
View
Playback
Fusion
Color
Fairlight
Workspace
Help
Media Pool
Sync Bin
Transitions
Titles
Effects
Aerospace Propulsion Project
Edited
Quick Export
Full Screen
Inspector
Main Edit
00:00:19:14
Master / Clips
01:00:12:17
DaVinci Resolve 19
Media
Cut
Edit
Fusion
Color
Fairlight
Deliver

Laser Head for Pico-Based Sand Clock

Drawing with Light

By Clemens Valens (Elektor)

The laser head kit presented in 2018 turned the Elektor Sand Clock of 2017 into a clock that writes the time on glow-in-the-dark film instead of in sand. A few years later, the Sand Clock was redesigned to use the Raspberry Pi Pico instead of the Arduino UNO — this version is currently available in the Elektor Store. Therefore, it is only logical to also reissue the laser head modification for the current Sand Clock. And why not use the clock to make drawings? In this article, we explain how you can achieve this.

In January 2017, *Elektor* published the Sand Clock, a novel clock that wrote the time in a sand bed [1]. It was based on an Arduino UNO. A year later, a modification was published [2] that replaced the sand bed by a sheet of glow-in-the-dark film and the pen by an ultraviolet laser. The laser clock was more discreet than the Sand Clock, as it didn't need to shake the sand bed every time to clear it before writing the new time.

A second update [3] added a stand to allow the laser clock to be placed vertically on a horizontal surface instead of hanging it on a wall. A PIR sensor was added too, so the clock could be made to write the time only if there was someone around to see it.

A few years later, the Sand Clock with the PIR sensor option was redesigned to use the Raspberry Pi Pico instead of the Arduino UNO.

This version of the Sand Clock is currently available in the Elektor Store. Therefore, it is only logical to also reissue the laser head modification for the current Sand Clock, which is what we did.

Hardware Update

The new laser head kit is almost identical to the kit from 2018, except for the laser holder arm that was made a bit more robust. Instead of an open, clamping grip, the holder is now a closed ring that cannot break, at least not easily (**Figure 1**). The laser is kept in place with two rubber rings with the arm sandwiched in between (**Figure 2**). Because the rest of the pantograph remained the same, assembling it is identical to the way described in 2018. Note that the stand-offs that carry the sand bed must be turned upside down for use with the laser head (**Figure 3**).

The UV laser replaces the two vibration motors used by the Sand Clock to level the sand. These motors are powered with a voltage of about 3 V. Diodes D2 and D3 take care of lowering the 5-V power supply to a

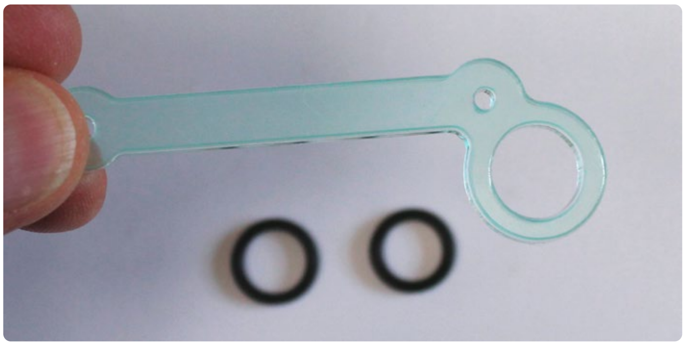


Figure 1: The new laser head mount has a ring rather than a grip to hold the laser.

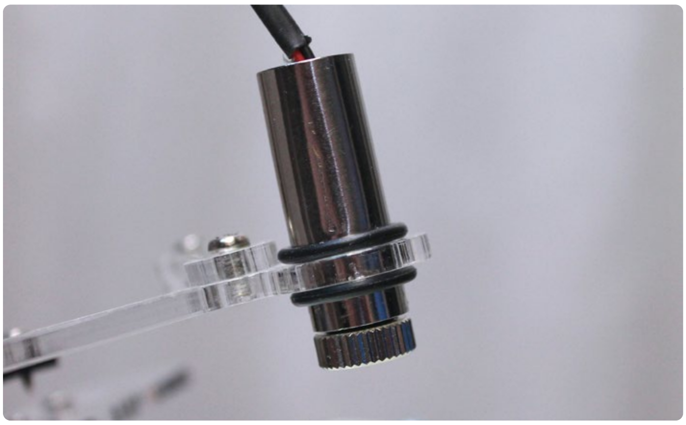


Figure 2: The laser is fixed with two rubber rings.

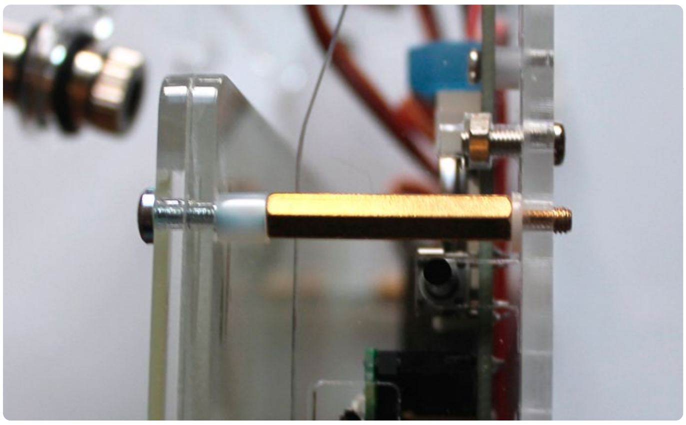


Figure 3: The stand-offs must be faced pointy threaded-side down.

suitable level. The laser, on the other hand, needs a 5 V supply, and so diodes D2 and D3 must be bypassed, see **Figure 4**. This is easily done by running a length of wire from the 5-V pin of the PIR connector to the M+ pin of the vibration motor's screw terminal (**Figure 5**). That's the only modification required for the circuit board.

Note that the new laser head is compatible with the original Arduino-based Sand Clock from 2017. It requires a similar modification for the laser's power supply as described above. Refer to [2] and [3] for the details.

Software Update

The software of the Raspberry Pi Pico-Based Sand Clock has been updated to drive the laser instead of the vibration motors. The main changes are the dimensions of the new pantograph arms that differ slightly from the Sand Clock and, of course, the On/Off control of the laser that replaces the vibration motor control.

While we were working on the new software, it was thought interesting to add a vector-drawing option to the clock. The functions required to draw lines, arcs, and circles were already available in the clock's software, and the observant user might have noticed the comments describing serial commands for them in the source code. Therefore, we added the possibility to receive a drawing script and execute it. Instead of just writing the time, the clock can now also be used to create ephemeral works of art (**Figure 6**).

Furthermore, as the Raspberry Pi Pico has plenty of program memory, scripts can also be placed in memory. This allows you to create animations, turning the clock in a fun decorative object for, e.g., Christmas or other special events. Making a script is a bit laborious, though, as it requires you to convert a drawing into line and arc segments with millimeter coordinates. However, a PCB design program like KiCad or a mechanical 2D-design program capable of drawing in millimeters can help you here. We've left the implementation of a g-code interpreter as used by CNC machines as an exercise to the reader.

Drawing Commands

The commands available for drawing are simple text-based commands that can be sent over a serial link. Make sure that the serial port speed of your terminal program is set to 115,200 baud (with eight databits, no parity, one stopbit, i.e., 115200n81). Also, serial commands must be terminated with both CR (carriage return) and LF (line feed). In Tera Term this is configured in *Setup* → *Terminal* → *New-Line* → *Transmit: CR+LF*. To send a script file using Tera Term, use the *File* → *Send File* command.

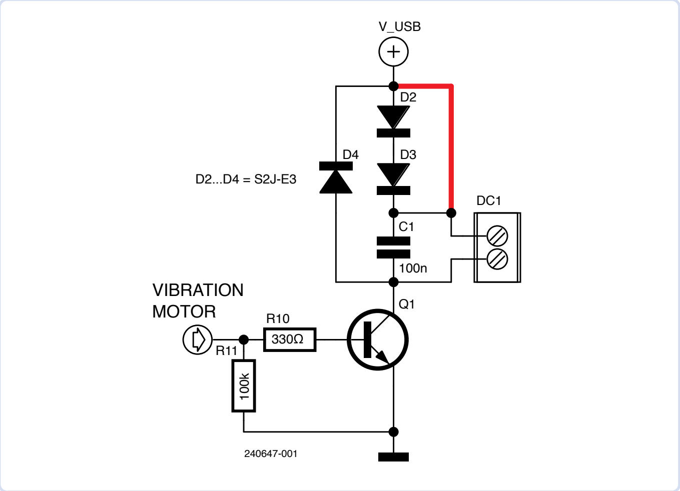


Figure 4: The red line shows the electrical modification of the main board.

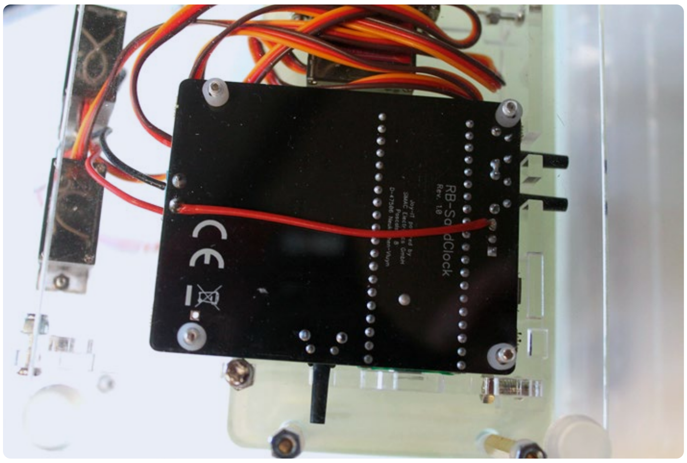


Figure 5: The electrical modification in practice.

The following commands can be used:

`f`: Script start, not required when a script is stored in internal memory.
`@`: End of script, not required when a script is stored in internal memory.

`pld`: Laser head down.
`plu`: Laser head up.
`le`: Laser enable (on).
`ld`: Laser disable (off).

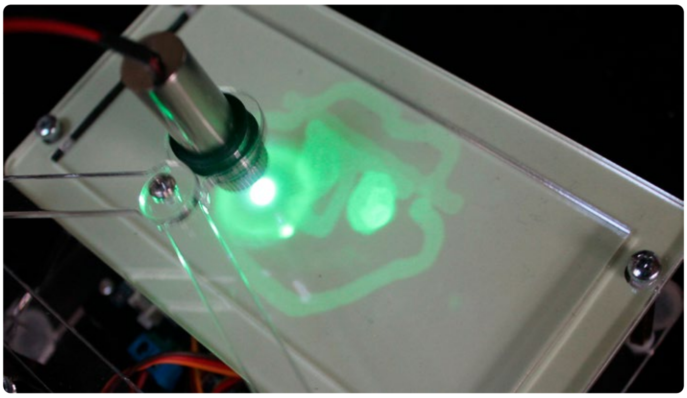


Figure 6: A script received over the serial link is being processed.

Moving

`pm <x> <y>` and `ps <x> <y>`

Move to position (x,y). Both commands `pm` and `ps` produce the same result. The values of `<x>` and `<y>` are in millimeters. The horizontal origin (x = 0) of the canvas is in the horizontal middle of the bed, therefore x-values may be negative. The vertical origin (y = 0) is 10 mm below the bed; therefore, when drawing, the y-value should not be smaller than about 10 mm. The exact limits depend on your construction and calibration. Example:

Move to x = 25 mm, y = 48 mm (i.e., up and to the right):

```
pm 25 48
```

Arcs and Circles

`pa <rx> <ry> <ab> <ae>`

Starting at the current position, draw an arc with a horizontal radius of `<rx>`, a vertical radius of `<ry>`, and from angle `<ab>` to angle `<ae>`. If the end angle `<ae>` is greater than the begin angle `<ab>`, the laser will move counterclockwise, else the laser moves clockwise. The angles are in radians, unless they are postfixed with the letter `d`, in which case they are interpreted as degrees. Example:

Draw a counterclockwise arc from 73° to 253° with a horizontal and vertical radius of 5.2 mm:

```
pa 5.2 5.2 73d 253d
```

Draw an ellipse with a horizontal radius of 5 mm and a vertical radius of 10 mm:

```
pa 5 10 0d 360d
```

Numerical values may have decimals and signs, which in theory allows for very precise positioning of the laser. In practice, however, this precision is not respected at all due to play in the pantograph joints and the wideness of the laser beam that is highly dependent on the writing height.

Note that the real origin of the pantograph (0,0) is right between the two servos that drive the arms. Therefore, it is unattainable by the laser.

The file *scripts.h* included in the source code available from [4] contains a few example drawings. To execute a built-in drawing script from the command line, use the command `z` with the number of the script as parameter (starting at 0). For example, execute built-in drawing script no. 0.:

```
z 0
```

You can, of course, add your own scripts to this file. Enjoy!

240647-01

Firmware Update for Raspberry Pi-Pico-Based Sand Clock

A byproduct of adapting the Raspberry Pi-based Sand Clock software for the laser head was a correction of an issue sometimes observed with the Sand Clock. When connected to a computer, everything worked fine, but when in stand-alone mode (i.e., only connected to a power supply), it would write the time just once. This issue has been fixed in V1.5. You can download it from [1].

Questions or Comments?

Do you have technical questions or comments about his article? Email the author at clemens.valens@elektor.com or contact Elektor at editor@elektor.com.

WEB LINKS

[1] Ilse Joostens & Peter S'heeren, "Sand Clock - A Real Eye-Catcher," Elektor 1-2/2017: https://elektormagazine.com/magazine/elektor-201701/40130

[2] Ilse Joostens & Peter S'heeren, "Laser Time Writer - Writing with light," Elektor 1-2/2018: https://elektormagazine.com/magazine/elektor-201801/41254

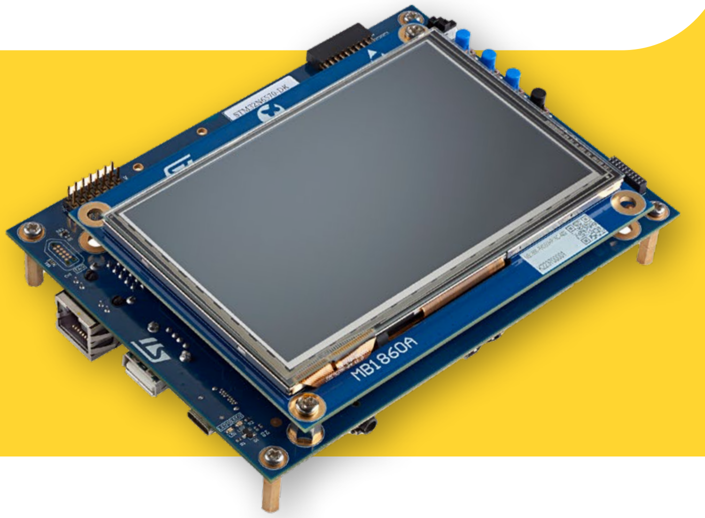
[3] Ilse Joostens & Peter S'heeren, "Laser Time Writer - Revisited!," Elektor 7-8/2018: https://elektormagazine.com/magazine/elektor-201807/41746

[4] Downloads for this article: https://elektormagazine.com/labs/laser-head-for-raspberry-pi-pico-based-sand-clock

ENTER THE STM32 EDGE AI CONTEST

By the Elektor Content Team

Unleash your creativity in the STM32 Edge AI Contest! Turn bold ideas into reality with the STM32N6 and showcase your Edge AI skills. Submit a project idea by April 30, 2025, to have a chance to get a free well-equipped STM32N6570-DK Development Kit!



Explore the cutting edge of artificial intelligence and edge computing by participating in the STM32 Edge AI Contest, powered by STMicroelectronics. This is your chance to bring your most innovative Edge AI ideas to life using the STM32N6570-DK Discovery Kit. Whether you're an experienced developer or a passionate maker, this competition is your platform to demonstrate your expertise, creativity, and problem-solving skills.

Timeline

- Deadline for Project Ideas: 30 April 2025
- Deadline for Final Project Submissions: 1 September 2025
- Nominees Revealed: 30 September 2025
- Grand Winner Announcement: November 2025 — Stay Tuned!

Prizes

- 1st Prize: €2,500
- 2nd Prize: €1,500
- 3rd Prize: €1,000

Visit ***www.elektormagazine.com/stm32ai*** for details and to submit a project. With €5,000 in cash prizes up for grabs, the judges are looking for Edge AI-driven projects that inspire and innovate. Your journey to cutting-edge innovation starts now!

250006-01

The STM32N6570-DK Dev Kit

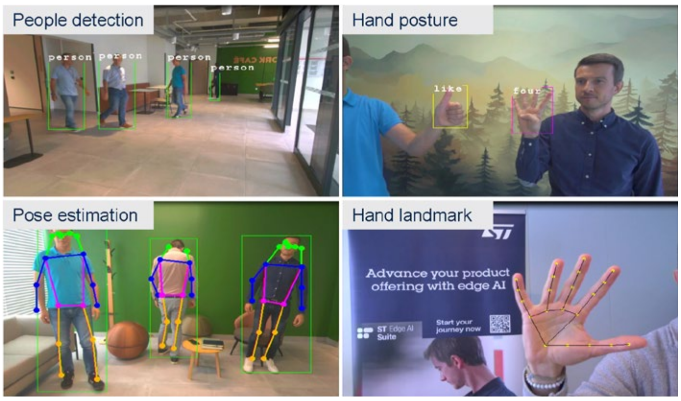
The first high-performance STM32 MCU with AI acceleration is your ultimate tool for advanced prototyping and development. With the STM32N6570-DK Discovery Kit, you can bring your AI vision projects to life with unparalleled ease and efficiency. Whether you're developing next-gen applications or exploring innovative prototypes, this kit has everything you need to succeed. Elevate your AI projects today!

Key Features

- STM32N657 microcontroller — with Neural-ART accelerator and 4.2 MB SRAM
- STLINK-V3 debugger — Effortless debugging and programming
- STMod+ & Arduino UNO connectors — Expand with unmatched flexibility
- MIPI connector — High-speed camera interface for seamless AI vision
- 2 USB ports — Lightning-fast data transfer
- 1 Gbit Ethernet — Reliable, high-speed networking
- 32 MB Hexadeca-SPI PSRAM — Power your most complex tasks
- Audio Jack and Codec— Built-in audio for versatile applications
- MicroSD Card Slot — Simplify storage expansion
- 5" capacitive multi-touch display
- 2 User LEDs and 3 Buttons
- ST AI Camera Module included
- Fan-Out Board with mikroBUS socket and Grove connectors included

Find out more

https://st.com/en/evaluation-tools/stm32n6570-dk.html



◄

The ST Neural-ART Accelerator enables the execution of advanced embedded AI applications on an MCU.
** Significantly reduces CPU load.*
** Increases frames per second.*

Source: Adobe Stock

A Multi-Sensor Environmental Monitoring System for Plants

Wireless Measurement of Water Supply and Light Conditions

By Alain Romaszewski (France)

This project was designed to care for indoor plants or a greenhouse during times when you may be away, while also storing data to analyze and optimize watering cycles. It uses the STM32WB5MM-DK board with multiple sensors and Zigbee communication, alongside software like Node-RED and Zigbee2MQTT for automation and remote control. The project was entered in the ST Wireless Innovation competition organized by Elektor and ST — and won the second prize.

My goal was to create a multi-sensor environmental monitoring system for plants and greenhouses, using a wireless development board from STMicroelectronics and various sensors to measure soil moisture, temperature, humidity, air quality, and light conditions. It wirelessly transmits data via the Zigbee protocol, with the ability to remotely control irrigation and lighting, and stores the collected data in an SQL database. Automation is managed through Node-RED, allowing real-time adjustments and remote monitoring through a web interface.

An overview of the project is shown in **Figure 1**. The STM32WB5MM-DK development board is coupled to a custom-made extension shield, all integrated in a 3D-printed case. This main module integrates several internal sensors to measure environmental conditions. External sensors are used to monitor soil moisture and soil temperature for three different plants. For water supply management, the board controls the irrigation pumps for the three plants and checks that enough water is left in the reservoir using the sensor. When it comes to lighting, the system measures luminosity and controls the lighting based on these readings.

The data gathered by these various sensors are transmitted as attributes over the Zigbee protocol. (See more details in the text frame **More about Zigbee**.) A USB dongle is used as a wireless interface and the ZigBee2MQTT application, installed on a Windows 11 machine, creates a Zigbee Coordinator, which additionally bridges between ZigBee events and a remote MQTT server. Each attribute defined in the measurement system updates on the MQTT server, and any changes made on the server can be sent back to the measurement system.

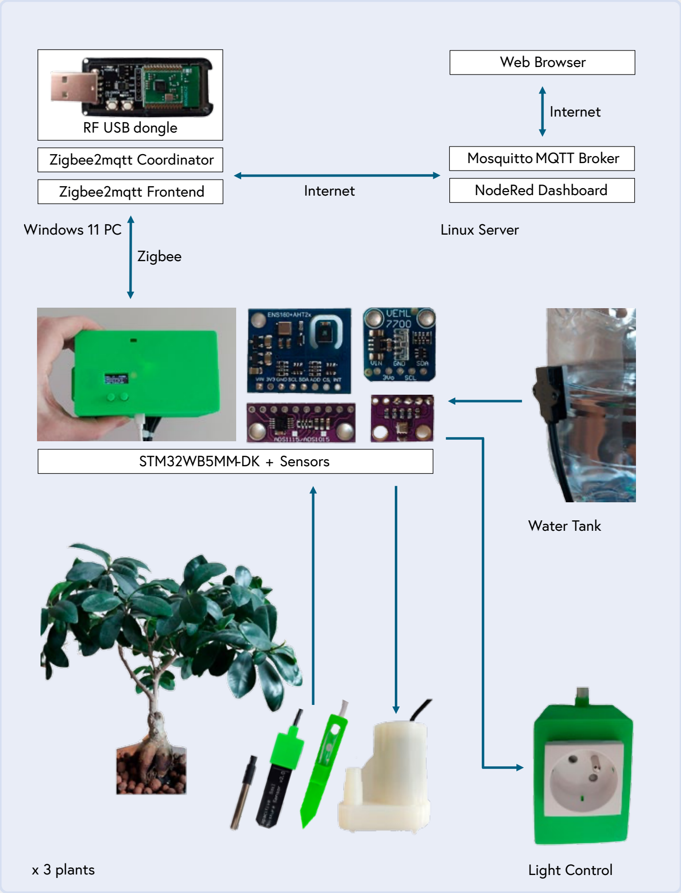


Figure 1: General architecture of the project.

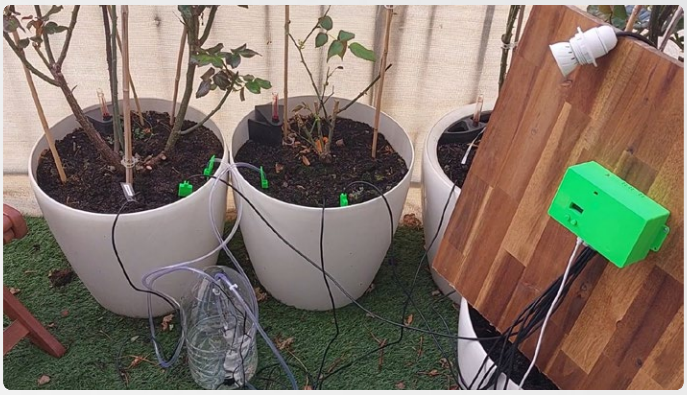


Figure 2: The system in use.

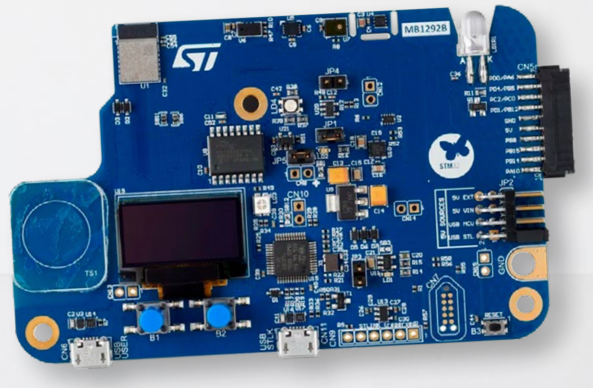


Figure 3: The STM32WB55MM-DK development board.

Since the data is accessible via the MQTT server, I chose to use Node-RED to subscribe to the server and track events, enabling automation. In my case, Node-RED is running on a Virtual Private Server (VPS) hosted by OVH and running Debian. The Dashboard feature of Node-RED provides a user interface accessible through a web browser from all over the world. Furthermore, all data is stored in an SQL database for further analysis. The system can be seen being used in **Figure 2**.

Main Board

The STM32WB5MM-DK board (**Figure 3**) is built around the STM32WB5MMG microcontroller from STMicroelectronics, which includes a Cortex M4 core and a Cortex M0+ communication coprocessor. To operate, the coprocessor needs to be flashed with the appropriate firmware based on the selected protocol and settings. The board provides 1 MB of flash memory and 256 KB of RAM.

I use the integrated temperature sensor on the board. This board has two I^{2}C interfaces named I2C1 and I2C3; the temperature sensor connects to the latter. Measurements are displayed on the onboard LCD screen, and navigation through the various display screens is controlled via two buttons. Other I/Os (including the I2C1 bus and six other GPIOs) available through the Arduino-compatible set of headers are used for managing pump and light controls, water level readings, and interfacing with the soil temperature sensors.

The board is powered through USB; I also installed a battery backup to supply three low-power water pumps (operating at 3 V), used to water each plant individually from a shared reservoir. A three-part enclosure, designed for easy mounting of the PCB and the additional daughterboards (see below) while allowing button access, was 3D printed with ASA material.

Sensors

For each plant, a soil moisture sensor is used. Like all the sensors in this project, these are easy to find at various places on the internet. They are capacitive, with two electrodes forming a capacitor. A 1-MHz signal is applied to the electrodes. The capacitance between the electrodes changes with the soil's moisture content, which changes the frequency. That frequency change is finally converted into an output voltage. To protect the electronic components of the sensor, I 3D-printed an enclosure in ASA material and filled it with two-part resin. The result is shown in **Figure 4**.

An ADS1115 (16-bit, four-channel analog-to-digital converter) from TI connected via the I2C1 bus is used to measure the voltage from the soil moisture sensors. I found this converter and sensor arrangement to be fairly robust against interference caused by noise and long wire lengths. The soil moisture sensors have been modified to operate on

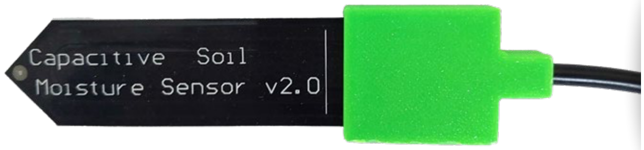


Figure 4: Soil moisture sensor.

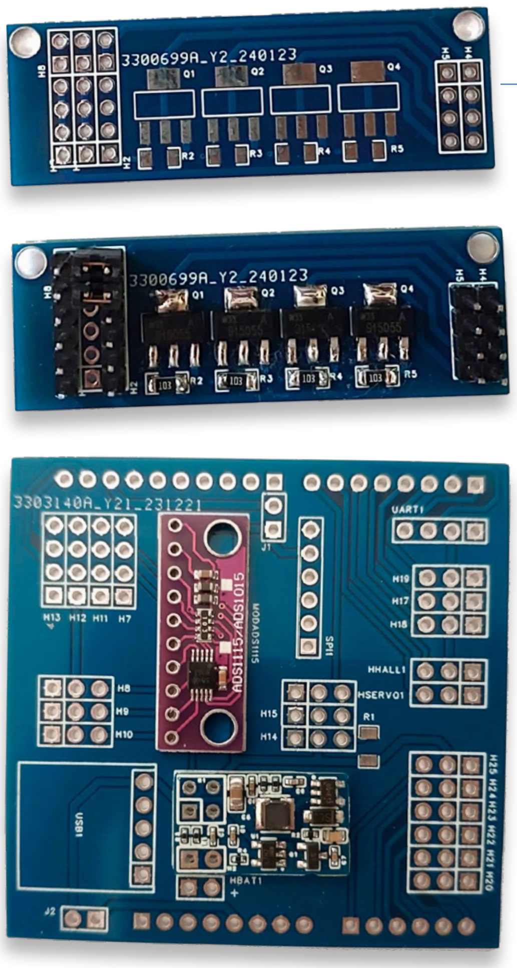


Figure 5: Custom PCBs to connect sensors.

3.3 V by removing the voltage regulator and are connected to three inputs of the ADS1115 analog-to-digital converter. One input of the converter is used to measure the battery voltage, which is divided by two to respect the ADS1115's maximum input voltage.

To monitor soil temperature, three DS18B20 sensors from Analog Devices are used, each housed in a stainless steel tube. These sensors operate over a shared 1-wire bus. Other environmental measurements, such as temperature, humidity, and pressure, are provided by a BME280 sensor from Bosch, while CO_2, total volatile organic compounds (TVOC) and indoor air quality (IAQ) levels are measured using an ENS160 sensor from ScioSense. An AHT25 sensor (Aosong) is also used for temperature and humidity compensation. Finally, luminosity is measured by a VEML7700 lux meter from Vishay. All these sensors operate on the I2C1 bus. The water level in the tank is monitored using a capacitive sensor mounted at the low level of the tank.

External Interfaces

An Arduino-compatible interface card was developed to simplify sensor connectivity. This card includes a switching power supply module to power the sensors and recharge the battery, which provides additional power for the pumps. It also houses the ADS1115 converter on a PCB module. Additionally, a custom interface card was made to control the pumps. This PCB has several N-channel power MOSFETs connected to the microcontroller's ports. The interface boards are shown on **Figure 5**; see the connected sensors in **Figure 6**. Another external module, equipped with a solid-state relay in a custom 3D-printed enclosure, is used to control a 230 V / 200 W light source.

A Short Review of the STM32WB5MM-DK

The STM32WB5MM-DK is a good development board offering many options for those working on IoT projects. On the plus side, the board is equipped with a rich set of sensors, an integrated debugger, and a well-sized flash memory and RAM. The processor also integrates the entire RF part, which is a notable advantage, although its implementation on a PCB can be tricky and adds to production costs. It's very easy to use, and the abundance of provided examples allows for quick learning and implementation.

However, there are some limitations to be aware of. The Zigbee firmware is somewhat obscure, though ST can provide the sources under certain conditions. Additionally, custom endpoints and clusters do not behave as expected, and only 8 attributes can be edited or read per request, which can be restrictive. Managing two processors with two applications can also feel cumbersome. In this respect, using the NUCLEO-WBA52CG board (also available for this ST contest) would have been more flexible, as it only has a single microcontroller.

While the board offers great functionality, the lack of examples using FreeRTOS, no external antenna option, and no Arduino support for Zigbee are drawbacks. Furthermore, the examples don't use STM32CubeMX, which complicates adding peripherals or tweaking settings. Overall, it's a solid development board, you just need to keep its limitations in mind.

Programming

Firmware (version 1.18 in this case) from the manufacturer must be downloaded into the memory area reserved for the M0+ radio processor. There are several firmware options for BLE, Thread, Zigbee, and BLE-Zigbee combinations, which are selected based on the specific use case. These firmware versions (the current version is 1.20) can be found online [1].

The programming for this project was carried out using the C language and the STM32CubeIDE 1.14 environment. The project is based on an example from the STM32WB library [2]. The example program can be found online [3]. The system uses a pseudo-preemptive multitasking model specific to STMicroelectronics and includes advanced functions for processor communication. However, the provided example does not utilize a known real-time operating system (RTOS).

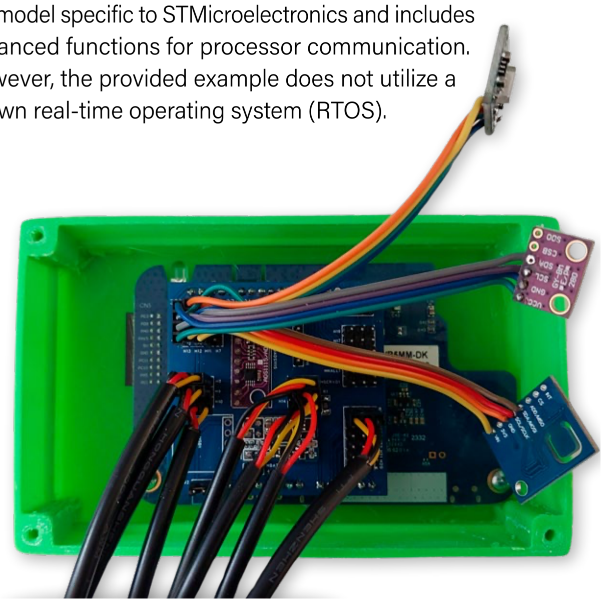


Figure 6: Connected sensors.

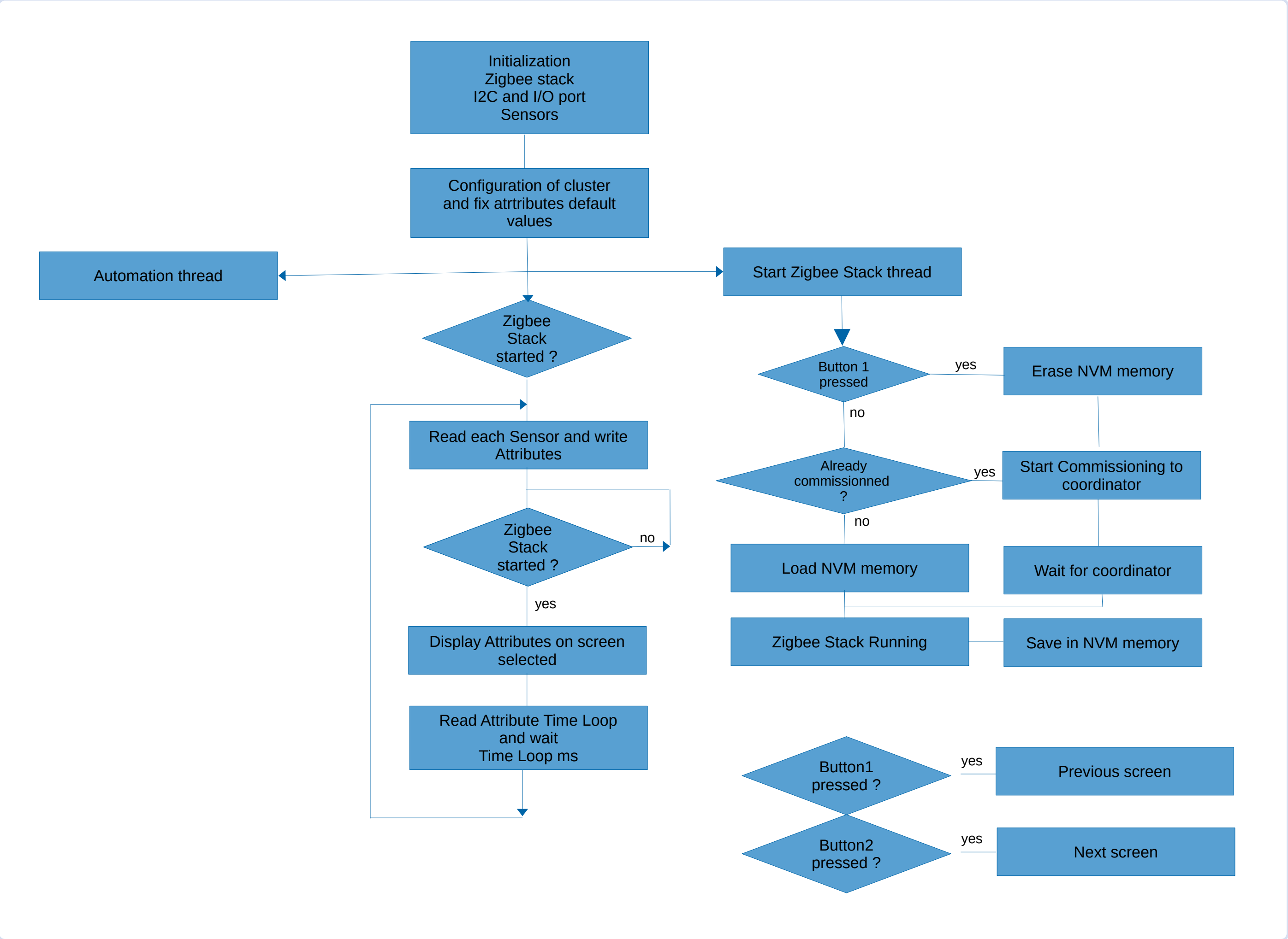


Figure 7: Flowchart of the program.

In this case, for physical measurements, a client-server architecture is used, where the client is the PC while the server is the STM32WB-5MM-DK. The Zigbee coordinator/client is responsible for regularly querying the server's attributes in order to retrieve and store the information. To perform actions, the coordinator can modify the values of the attributes on the endpoint, the latter upon receipt will execute the requested tasks, indicating its status by modifying the attributes.

To help with this, STMicroelectronics provides a pre-compiled library for cluster management. A Zigbee cluster is a group of related attributes and commands within a Zigbee device that defines a specific functionality, such as temperature measurement or lighting control, standardized across Zigbee devices for interoperability. I chose to use a standard temperature measurement cluster. The application was configured on Zigbee channel 15, with a single server endpoint. A basic cluster is automatically added by the stack. Proprietary attributes are then added to this standardized cluster.

All sensors are queried cyclically, and the results are stored in the respective cluster attributes. By default, this occurs every 20 s, though the time interval can be modified through the `measuring time loop` cluster attribute.

Automation of the pumps is handled by setting a pump activation time, specified in milliseconds, via the corresponding Zigbee attribute `running time pump`. If this attribute is non-zero, the pump starts and the value is decremented until it reaches zero, at which point the pump stops. Each pump is controlled by a separate I/O pin, and the ON/OFF status of each pump is reflected in the `state pump` attribute. Lighting control is similarly managed through the `light state` attribute.

A flowchart of the program is shown in **Figure 7**, with the automation thread detailed in **Figure 8**. The function for reading sensor data and transmitting it to the cluster attributes is too lengthy to show in full, but key portions are displayed in **Listing 1**.

Zigbee Processing

The Zigbee stack is initialized in the file *Application/User/STM32_WPAN/App/app_zigbee.c*, with the the functions `APP_ZIGBEE_Init`, `APP_ZIGBEE_StackLayersInit`, `APP_ZIGBEE_ConfigEndpoints`, as well as some other functions managing commissioning and persistence with non-volatile memory (NVM).

Network formation is carried out by the `APP_ZIGBEE_NwkForm` function. A standardized temperature measurement cluster was created, and additional attributes necessary for the project were added. This was done in the modules *app_cluster.c* and *app_cluster.h*, with the `AddProjectCluster(void)` function, which adds a cluster to the application. The implementation of this function is shown in **Listing 2**.

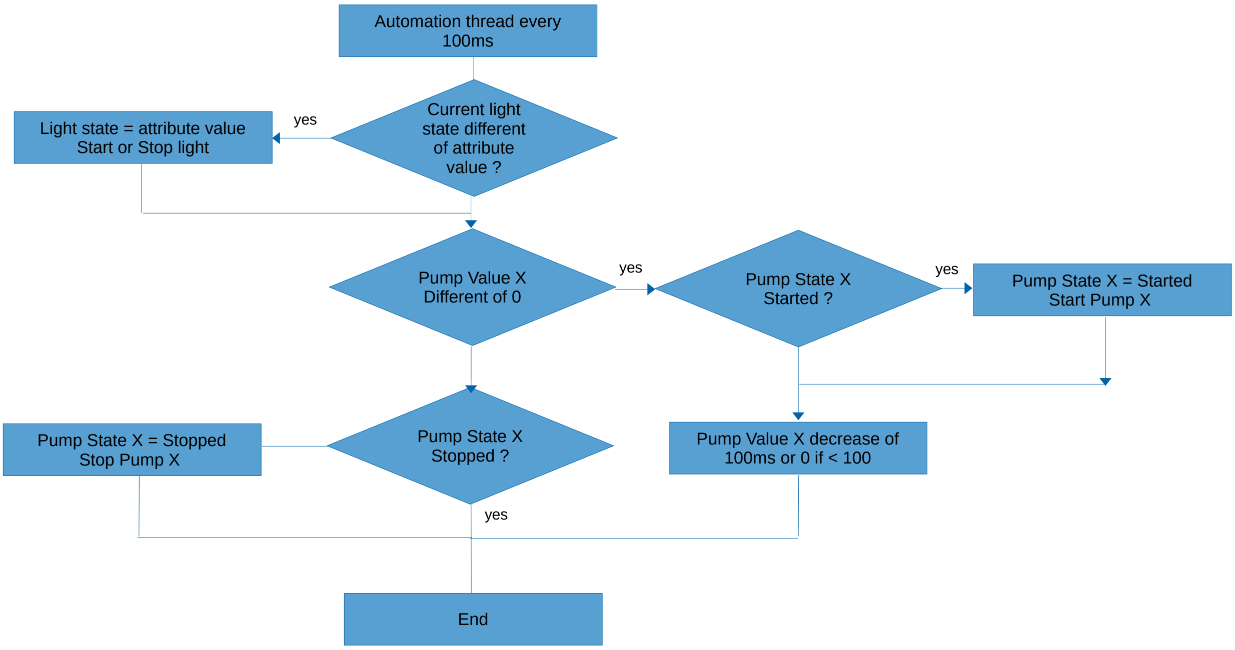

Automation thread every 100ms
Current light state different of attribute value ?
yes
Light state = attribute value Start or Stop light
Pump Value X Different of 0
yes
Pump State X Started ?
yes
Pump State X = Started Start Pump X
Pump State X = Stopped Stop Pump X
Pump State X Stopped ?
Pump Value X decrease of 100ms or 0 if < 100
yes
End

Figure 8: Details of the automation thread.

Listing 1: Reading sensors

```
void Measuring(void) {
     enum ZclStatusCodeT status;
     float temp, mes;
     int16_t vint16t;
     uint16_t vuint16t;
     uint16_t als, white;
     // Read temperature value of the main board
     STTS22H_getTemperatureValue(&temp);
     vint16t = (int16_t)(temp * 100);
     status = ZbZclAttrIntegerWrite(zigbee_app_info.clusters[IDCLUSTER_CARD],
      ZCL_TEMP_MEAS_ATTR_MEAS_VAL, (int16_t)vint16t);

     if (status != ZCL_STATUS_SUCCESS) {
      APP_DBG("Error Writing Att. temperature in cluster %d", status);
     }
     APP_DBG("T:%.1f C", temp);
     // Read temperature, pressure and ambient humidity with BME280
     read_bme280();
     if (dev.Error != 0) {
      APP_DBG("Error reading BME280: %d", dev.Error);
     }
     vuint16t = (uint16_t)(dev.pressure * 10);
     status = ZbZclAttrIntegerWrite(zigbee_app_info.clusters[IDCLUSTER_CARD],
      ATTENVPRESS, (uint16_t)vuint16t);

     if (status != ZCL_STATUS_SUCCESS) {
      APP_DBG("Error Writing Att. Pressure in Cluster");
     }
     vint16t = (int16_t)(dev.temperature * 100);
     status = ZbZclAttrIntegerWrite(zigbee_app_info.clusters[IDCLUSTER_CARD],
      ATTENVTEMP, (int16_t)vint16t);

     if (status != ZCL_STATUS_SUCCESS) {
      APP_DBG("Error Writing Att. temperature in Cluster");
     }
     vuint16t = (uint16_t)(dev.humidity * 100);
     status = ZbZclAttrIntegerWrite(zigbee_app_info.clusters[IDCLUSTER_CARD],
      ATTENVHUM, (uint16_t)vuint16t);

     if (status != ZCL_STATUS_SUCCESS) {
      APP_DBG("Error Writing Humidity in Cluster");
     }
     APP_DBG("Temp: %.2f - Hum: %.2f - Press: %.2f", dev.temperature, dev.humidity, dev.pressure);
     return;
 }
```

Listing 2: The AddProjectCluster(void) function.

```
void AddProjectCluster(void)
 {
     struct ZbApsmeAddEndpointReqT req;
     struct ZbApsmeAddEndpointConfT conf;
     enum ZclStatusCodeT status;
     uint8_t i = 0;

     // Create Endpoint Gen
     memset(&req, 0, sizeof(req));
     req.profileId = ZCL_PROFILE_HOME_AUTOMATION;
     // profile >0x7FFF if proprietary
     req.deviceId = ZCL_DEVICE_ENVIRONMENTAL_SENSOR;
     // Normalized Endpoint else >0x2FFF if proprietary
     req.endpoint = ENDPOINT_GEN;
     ZbZclAddEndpoint(zigbee_app_info.zb, &req, &conf);
     assert(conf.status == ZB_STATUS_SUCCESS);
     zigbee_app_info.clusters[IDCLUSTER_CARD] =
       ZbZclTempMeasServerAlloc(zigbee_app_info.zb, ENDPOINT_GEN, -4000, 14500, 10);
     assert(zigbee_app_info.clusters[IDCLUSTER_CARD] != NULL);
     ZbZclClusterEndpointRegister(zigbee_app_info.clusters[IDCLUSTER_CARD]);
     status = ZbZclAttrAppendList(zigbee_app_info.clusters[IDCLUSTER_CARD],
      attr_list_Main, ZCL_ATTR_LIST_LEN(attr_list_Main));

     // Initializing default Pump alimentation and state
     for (i = 0; i <= 2; i++) {
       status = ZbZclAttrWrite(zigbee_app_info.clusters[IDCLUSTER_CARD], NULL,
        ListAttrPumpAlim[i], (uint8_t *)&Automat.PumpValue[i],
        2, ZCL_ATTR_WRITE_FLAG_NORMAL);
       if (status != ZCL_STATUS_SUCCESS) {
        APP_DBG("Error Initialising Att Water Alimentation in Cluster Plant %d", i + 1);
       }
       status = ZbZclAttrWrite(zigbee_app_info.clusters[IDCLUSTER_CARD],
        NULL, ListAttrPumpState[i], (uint8_t *)&Automat.PumpState[i],
        1, ZCL_ATTR_WRITE_FLAG_NORMAL);
       if (status != ZCL_STATUS_SUCCESS) {
        APP_DBG("Error Initialising Att Water Alimentation
        State in Cluster Plant %d", i + 1);
       }
     }
 }
```

More About Zigbee

The Zigbee communication in this system is based on the IEEE 802.15.4 physical protocol, which shares the 2.4 GHz frequency band with Wi-Fi. The band is divided into 16 channels (11 to 26), with some channels not used by Wi-Fi and thus more suitable for Zigbee use (e.g., channels 15, 20, and 25). All devices connected to the main coordinator must operate on the same channel.

Zigbee operates on a mesh network architecture and follows a master-slave communication model. A brief explanation of how Zigbee functions is provided below. Zigbee uses a mesh topology, meaning each device or node can communicate directly with other nearby nodes. There are three primary types of Zigbee nodes:

- Zigbee Coordinator: The main node responsible for network initialization and organization.
- Zigbee Router: Participates in routing information and can function as an end device within the network.
- Zigbee End Device: A simple device that connects directly to a coordinator or via a router.

Each router or end device must first connect to the coordinator to establish a secure connection, exchanging a key to create an association. This association is stored between the router/end device and the coordinator, each device having a unique 64-bit MAC address. The coordinator authorizes associations for a limited time and generates a 16-bit short address to identify each Zigbee node.

In the Zigbee network, master-slave communication is implemented where the coordinator functions as the master, and end devices act as slaves. End devices report data to the coordinator when the data is modified or at regular intervals, a process referred to as reporting and binding.

Each Zigbee node can consist of one or more endpoints (ranging from 1 to 240). Each endpoint contains one or more clusters, which represent specific functionalities. Clusters can be standard Zigbee protocol functions or proprietary ones, with default attributes defined. Custom attributes can be added to any cluster.

Each attribute is identified by a standardized 16-bit identifier and contains a value, which can be of varying types (e.g., 8, 16, 32, or 64-bit, strings, enumerations, tables, etc.). Attributes can be read, written, or reported to the coordinator or other nodes at set intervals or persistently (its value is then memorized by the endpoint). It's important to note that Zigbee nodes regularly report to routers and coordinators, which must remain active at all times. This is one of the protocol's main drawbacks.

The Zigbee stack handles the protocol for reporting and binding attributes to other nodes; in this example the only other node is the Zigbee coordinator on the PC. The program mainly involves updating attributes and managing automation when attributes change. A portion of flash memory is reserved as a Non Volatile Memory (NVM) to store commissioning, reporting, binding, and attribute values. When this memory is blank, the Zigbee thread attempts to commission with a coordinator. Once the process is successful, the information is saved in NVM memory. On the next startup, this information is reloaded from memory, preventing the need for re-commissioning.

At power-up, pressing button erases the NVM memory and forces a re-commissioning process. The list of attributes added within `Endpoint 1`, to the base cluster (of type `TempMeasServer`), is shown in **Table 1**.

Program Modules

In the directory *Drivers/BSP/Component/*, you'll find drivers for various sensors. The `ADS1115` is for a 16-bit AD converter, the `BME280` handles temperature, humidity, and atmospheric pressure, and the `ENS160` supports CO$_2$, air quality index and TVOC sensing. The `SSDL1315` driver is for OLED displays. Other drivers in the directory include `AHTxx`, `DS18B20`, `Dwt`, `OneWire`, `STS22h`, and `VEML7700`. On the other hand, the *STM32WB5MM-DK* directory includes drivers for the buttons, RGB LED and OLED display.

Finally, the directory *Application/User/Core* contains the files used to provide a high level control interface to the various sensors and manage system functions. The `app_ads1115`, `app_bme280`, `app_ahtxx`, `app_ds18b20` etc. control the corresponding sensors listed above.

Table 1: Zigbee Attributes.

Address	Attribute	Size (bits)	Unit or Range
0x000	Temperature Card	16	°C * 100
0x101	Battery Voltage	16	0 to 5000 mV
0x303	Temperature	16	°C * 100
0x300	Luminosity	16	0 to 65535
0x30B	White Luminosity	16	0 to 65535
0x301	Pressure	16	Hp * 10
0x302	Humidity	16	% * 100
0x304	AQI	8	0 to 4
0x305	CO$_2$	16	ppm
0x306	TVOC	16	ppb
0x307	Resistance 1	32	Ω
0x308	Resistance 2	32	Ω
0x309	Resistance 3	32	Ω
0x30A	Resistance 4	32	Ω
0x30C	Tank Level	8	0 or 1
0x340	Light State	8	0 or 1
0x102	Measuring Time loop	16	ms
0x3X0	Temperature Plant X (X from 1 to 3)	16	°C * 100
0x3X1	ID Adress DS18B20 X (X from 1 to 3)	64	Address
0x3X2	Soil Moisture X (X from 1 to 3)	16	% * 100
0x3X3	Running time pump X (X from 1 to 3)	16	ms
0x3X4	State Pump X (X from 1 to 3)	8	0 to 2

Listing 3: JSON Payloads.

```
JSON payload sent for writing attribute PlantWaterAlimentation1:

 {
     "write":
     {
     "cluster": "msTemperatureMeasurement", "options": {},
     "payload": {
         "PlantWaterAlimentation1": 1000
     }
     }
 }

 JSON payload sent for reading here eight attributes:

 {
     "read":{
     "attributes" :[
         "Luminosity",
         "WhiteLuminosity",
         "PlantTemperature1",
         "PlantSoilMoisture1",
         "PlantMACDS1",
         "PlantWaterAlimentation1",
         "PlantWaterAlimentationState1"
     ],
     "cluster":"msTemperatureMeasurement", "options":{}
     }
 }
```

The driver app_automat controls automations like pumps and lights, while app_screen handles the display of measurement results and app_ScheduleMeasure manages cyclical measurements and Zigbee attribute updates. Other files include ee, flash_driver, hw_flash to emulate an EEPROM on a Flash memory area of the processor. Finally, hw_timerserver, hw_uart, main, and stm_logging are for timers, serial interfaces and debug display.

Zigbee2MQTT – The Zigbee Coordinator

To operate, the system requires access to a Zigbee coordinator. In this case, an open-source application, *Zigbee2MQTT*, is used, running on either Windows or Linux. The application supports a radio coprocessor on a USB stick, based on the Silicon Labs EFR32MG21 chip with EZSP v8 firmware, and it manages the Zigbee protocol through a web interface.

Zigbee2MQTT allows commissioning of Zigbee devices, viewing and modifying attributes, and interfacing with an MQTT server. When a Zigbee device is commissioned, the coordinator receives its description and distinguishes it by name and manufacturer. While many existing devices are already supported by Zigbee2MQTT, custom configuration file is needed for the attributes specific to this project. This file specifies the type of attributes and the names that are transmitted to the MQTT server under a designated topic. As a reminder, a topic is a path to a value in the MQTT messaging system.

For example, a typical MQTT topic path would follow the <path>/<device>/<endpoint>/set pattern and could be:

```
zigbee2mqtt/STM32WB5MM-DK/1/set
```

This allows MQTT to modify one or more attributes via a specific topic. Zigbee2MQTT subscribes to this topic and transmits read or write requests to the Zigbee network. The ZigBee device responds with the attribute values, and Zigbee2MQTT publishes these results with separate MQTT topics. Examples of JSON payloads used for reading or writing attributes are shown in **Listing 3**.

Once the attributes are read, their values are published with the corresponding MQTT topics. Applications can subscribe to topics such as Luminosity, PlantTemperature1, etc., to receive updates accordingly. Commands and attribute modifications can also be tested directly through the Zigbee2MQTT graphical user interface (GUI). The complete configuration for Zigbee2MQTT is available online [4], along with detailed installation instructions for Windows.

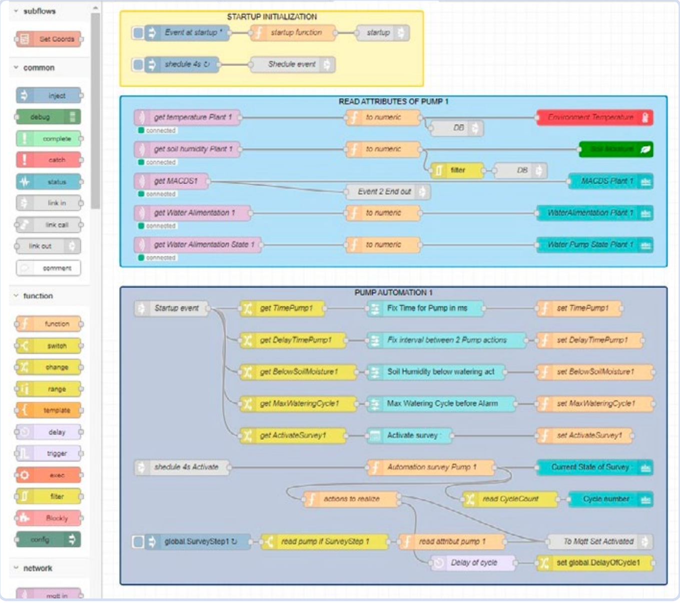


Figure 9: Node-RED flow.

Node-RED Interface and Automation

Various home automation tools can be used to manage MQTT topics and create automation workflows, such as Node-RED, Home Assistant, OpenHab, etc. For this project, I chose to use Node-RED. It is built on Node.js and allows users to create flows by connecting programmable graphical elements (see **Figure 9**) to manage MQTT publications and subscriptions. Node-RED integrates with databases like MariaDB and can store attribute values for historical analysis. A timer system and a dashboard facilitate interaction, configuration, and automation generation.

In this implementation, general sensor values from three plants are displayed, and automation controls the lighting based on luminosity levels and sunset times. Another automation cycle controls watering, activating the pumps based on soil moisture levels. Node-RED is highly customizable, allowing users to adjust the system's behavior to their needs. The resulting dashboard is shown in **Figure 10**.

Online Resources and Software

The complete documentation for this project is available in my GitHub repository [5], including the STM32Cube project in ZIP format, PCB files, and 3D models for the enclosure. Detailed information regarding the Zigbee2MQTT configuration can be found in the file *ZigBee2MQTT_on_Windows_install.pdf*. Additional details about Node-RED configuration, relevant datasheets, and project images are also included in the repository. As far as software is concerned, apart from those already mentioned, I've also used and can recommend DesignSpark [6] for 3D modeling, while PrusaSlicer [7] was used for preparing 3D prints. The PCB was designed with EasyEDA [8].

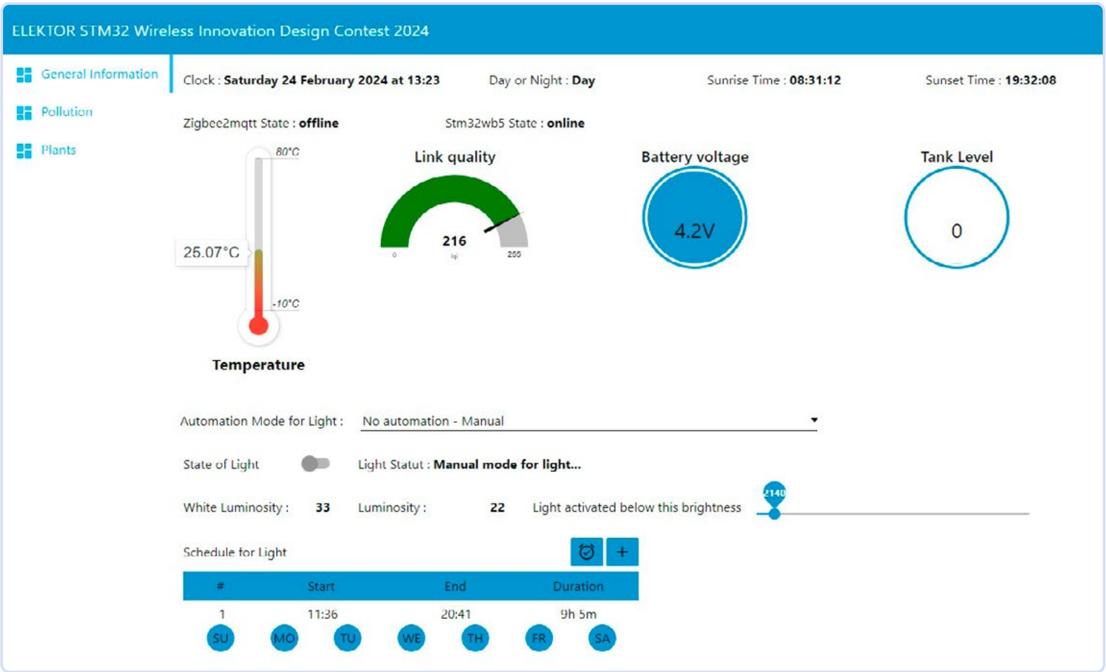
Enriching and Motivating

This project provided an opportunity to evaluate STMicroelectronics' communication processor and establish a complete measurement system. Future plans involve further development with new products, such as the STM32WBA5x, which will be integrated into each sensor and actuator. This will allow for battery operation, reducing wiring, and the ability to work with multiple communication protocols. Last but not least, this contest allowed me to discover and learn from the work of other participants, which has been very enriching and motivating! ◀

240578-01

Questions or Comments?

Do you have questions or comments about this article? Email the author at alain@romaszewski.fr, or contact Elektor at editor@elektor.com.



Clock : **Saturday 24 February 2024 at 13:23** Day or Night : **Day** Sunrise Time : **08:31:12** Sunset Time : **19:32:08**

Zigbee2mqtt State : **offline** Stm32wb5 State : **online**

Automation Mode for Light : No automation - Manual

State of Light Light Statut : **Manual mode for light...**

White Luminosity : **33** Luminosity : **22** Light activated below this brightness

Schedule for Light

#	Start	End	Duration
1	11:36	20:41	9h 5m

Figure 10: The resulting dashboard.

About the Author

Alain Romaszewski is a specialist in IT and digital electronics based in Amélie-les-Bains, Pyrénées-Orientales, France. He teaches IT, with a focus on Unix systems and networks, at the high school level. Alain is a founding member of the Physical Measurements IUT in Maubeuge, France, and is an expert in the Windev-Webdev programming environment and Typo3 CMS. He is also interested in research and development for natural resource management.

Related Products

- **Majid Pakdel, *Advanced Programming with STM32 Microcontrollers* (E-book, Elektor, 2020)**
  www.elektor.com/19527
- **Node-Red Development Bundle**
  www.elektor.com/20849

WEB LINKS

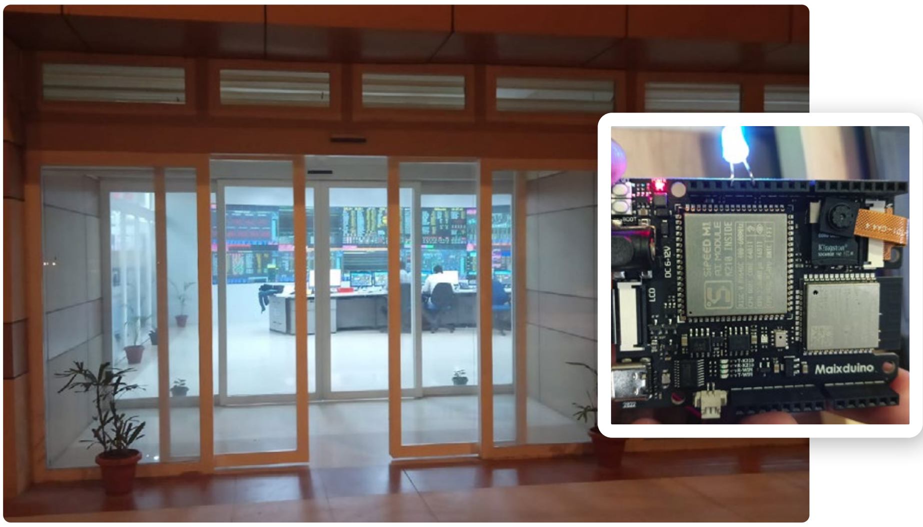
[1] Firmware Downloads: https://tinyurl.com/mw8xw3d4
[2] The STM32WB library: https://github.com/STMicroelectronics/STM32CubeWB
[3] Basic examples: https://tinyurl.com/yeypkdh3
[4] Zigbee2MQTT package and instructions: https://tinyurl.com/3uam3d83
[5] This project on GitHub (Online Resources): https://tinyurl.com/5yhmajkv
[6] DesignSpark Mechanical: https://www.rs-online.com/designspark/mechanical-software
[7] Prusa Slicer: https://www.prusa3d.com/en/page/prusaslicer_424/
[8] EasyEDA: https://easyeda.com/

Maixduino AI-Powered Automatic Doorman

Face Detection with a Camera

By Somnath Bera (India)

Tired of infrared proximity sensors opening a door every time someone passes by? With this project, you can build an AI-powered automatic door system that will help you minimize outside noise disturbances and improve energy efficiency. The system uses a Maixduino development board with a Kendryte K210 chip, combining AI-powered face detection with a camera and a TFT screen to identify when a person approaches the door directly. The software is written in MicroPython, leveraging the KPU neural network processor for real-time facial recognition, and controls the door via GPIO-triggered relays.



The double doors of the Central Control Room at the NTPC power station insulate the engineers inside from the high noise levels from the turbine floor.

Automatic doors usually do a great job of protecting office workers from the outside world. They help keep the office at an optimal temperature while still providing easy access to anyone wanting to enter or exit. In most cases, the conventional infrared (IR) proximity sensor controls doors very effectively.

However, in noisy industrial environments, a little more sophisticated technology may be needed to offer more control. At one of the power stations in India, there are three massive 660-MW turbines situated on a large 100-meter-long turbine floor. These steam-driven turbines, coupled with a generator at one end, generate around 900 MW; enough electricity to power a city twice the size of San Francisco.

The Common Control Room (CCR) for this important power plant has one main entrance door, located on the turbine floor side. The only entrance is through a pair of swing glass doors — an outer door and an inner door. This two-door system protects the air-conditioned atmosphere inside the CCR. When someone approaches the door, whether from inside or outside, the doors open one after the other, allowing the visitor to enter or exit.

This double door setup helps keep the control room cool and quiet, as the air trapped between the doors acts as both insulation and a sound barrier, shielding the CCR as much as possible from the 140-dB noise of the turbine generators.

However, there is one drawback common to automatic doors everywhere; people walking alongside the doors can cause the motion

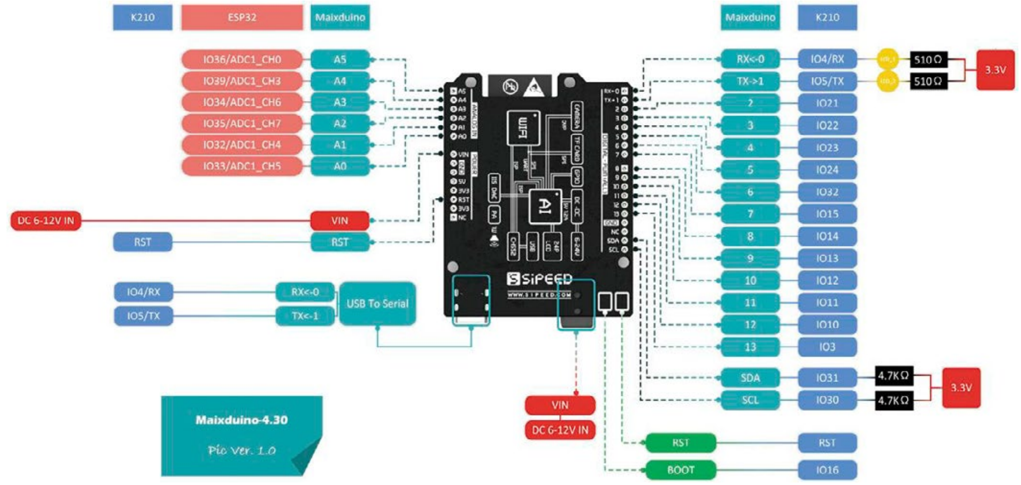


Figure 1: Maixduino pinout diagram. (Source: Sipeed)

sensors to open them, even if they have no intention of entering or exiting. In noisy environments such as these, it can cause the engineers inside to be constantly disturbed by the outside noise, triggered by the doors opening unnecessarily as the turbine floor's loud noise pours in.

A Smarter Solution

The proximity sensor's function is the root of this unwanted issue because these IR sensors have very limited functional capabilities. We have therefore replaced the traditional IR proximity sensors with AI-powered facial detection hardware and proximity calculation capabilities to ensure the doors only open when a human approaches.

A small Maixduino-powered development board (**Figure 1**) — assisted by a camera, a TFT screen, and AI code — determines whether the approaching entity is a human and how close their face is to the door. If it crosses a set threshold, the door will open to allow entry. If the distance criteria are not met, the system continues scanning for faces, preventing the doors from opening unnecessarily when people pass by.

The logic flow is shown in **Figure 2**. Note the "start" and "end" milestones to understand the built-in logic. Since this is an AI-based face detection, if a person passes by without looking at the door, the doors won't open, even if the distance logic is satisfied.

Gate-Opening Mechanism

When the conditions are met, two GPIO-powered relays trigger the gate opening mechanisms. Typically, when one gate opens, the other remains fully closed. Once the first gate opens and the person enters the chamber, the gate fully closes, isolating the chamber from both sides, and only then does the second gate open. This is usually done by sequentially firing the GPIOs. More GPIOs can be deployed if needed, depending on the gate mechanism. The gate opening can be customized to field requirements.

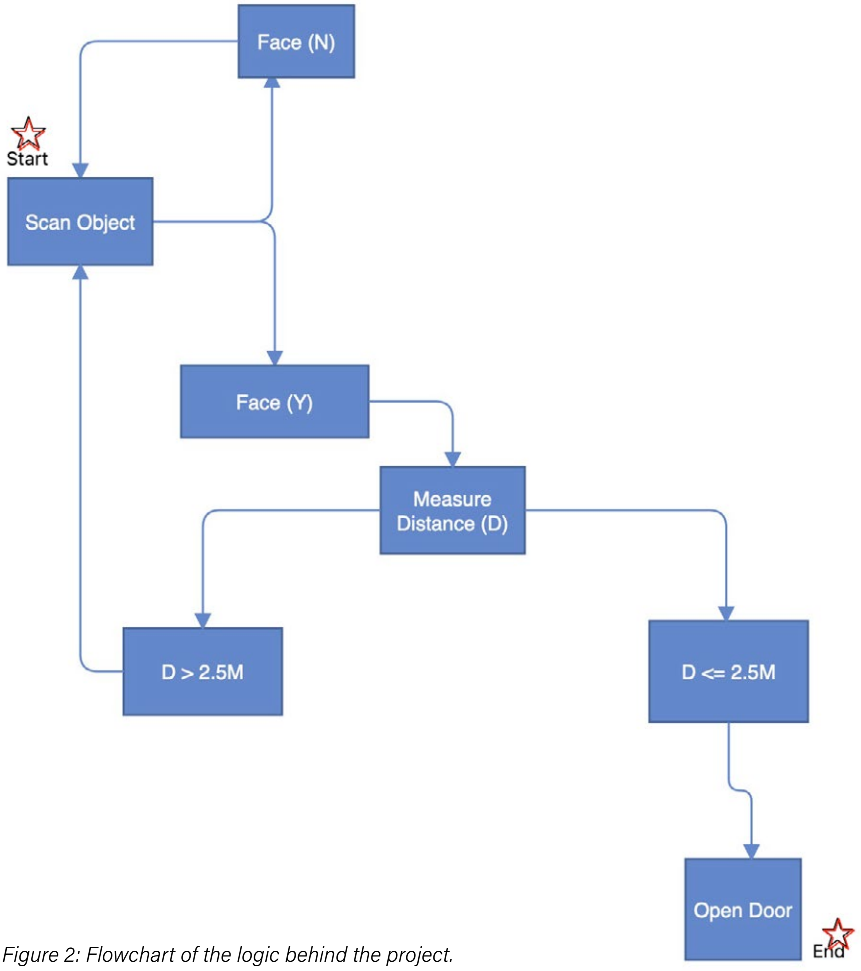


Figure 2: Flowchart of the logic behind the project.

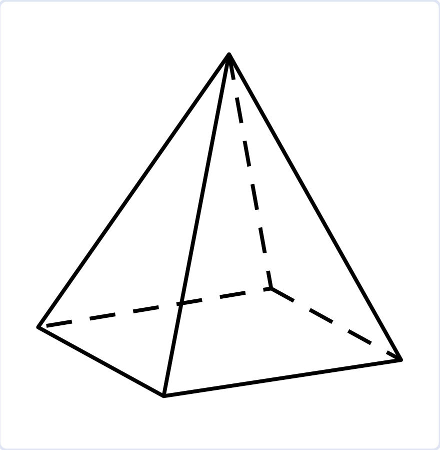


Figure 3: Perspective view of the base of a pyramid.

Calculating the Distance

To determine when to trigger the gates, the AI algorithm detects a person's face and draws a bounding box around it in the captured image, which can be seen in real-time on the TFT screen. The width of this box is used to estimate the distance between the face and the camera by applying a mathematical equation, with the parameters fine-tuned through experiments to ensure precision.

The camera's focal length is instrumental for determining the relationship between the distance of the face and the size of the facial box in the image. The change in size follows a linear relationship within the focal range, but beyond this range, the relationship becomes nonlinear, requiring a more complex calculation to maintain accuracy.

Two types of distance logic have been implemented: linear for distances up to 175 cm and quadratic beyond that, up to 250 cm. This was done to enhance the AI-powered door system's accuracy and efficiency. The linear model provides fast and reliable detection for close-range proximity, making the door responsive when someone is nearby. The quadratic model ensures more accurate distance calculations at longer ranges, accounting for perspective distortion as a person approaches from further away. This combination optimizes performance across varying distances, preventing premature triggers and enhancing overall user experience.

Using the first distance calculation of up to 175 cm, the width and height of the encompassing box are then compared linearly against various distances from the camera using a linear model. The distance is extrapolated based on the width of the object box. This is similar to how the base rectangle of a pyramid appears smaller as it moves upwards, eventually reaching a point at the pinnacle (**Figure 3**). Likewise, the picture rectangle becomes smaller as it moves away from the camera.

Up to 175 cm, a simple linear formula is used:

$$Y = M * X + C$$

X is the Pyramid Base Rectangle (Face Width), and Y is the Height of the Pyramid (distance).

However, beyond 175 cm, a quadratic equation is used:

$$Y = A * X^2 + B * X + C$$

which provides more accurate results up to 250 cm. With a better or longer focal-length camera, this computation can be further improved.

Once the distance is within 250 cm, the GPIO fires, and the gate opens or closes.

Hardware

The chip that sits as the heart of the Maixduino board, a Kendryte K210, is a highly integrated system-on-chip (SoC) designed for AI and IoT applications, incorporating machine vision and hearing capabilities. Manufactured using TSMC's 28-nm ultra-low-power process, it features dual-core 64-bit RISC-V processors for optimized power efficiency, stability and reliability. The K210 is designed for rapid deployment, enabling "zero threshold" AI development for fast integration into products.

A key component is the Knowledge Processing Unit (KPU), a general-purpose neural network processor. The KPU supports convolution, batch normalization, activation, and pooling layers, making it capable of real-time object and facial detection.

We are using MicroPython as the programming language to develop the code, with MaixPy IDE as the development environment, tailored specifically for the Maixduino board. The accompanying MicroPython code is concise, less than 50 lines, leveraging the KPU for most of the computation. Facial detection involves comparing rectangle dimensions (width and height) against predefined linear models to determine proximity. When the threshold distance is met, the General-Purpose Input/Output (GPIO) pins are triggered.

In this application, two GPIO pins (GPIO 13 and 12, as shown in **Figure 4**) control the gate mechanism via a 5 V relay. The GPIO pins are connected through a BC547 transistor, with the emitter grounded, and the relay connected to +5 V. For a 12-V relay, a separate power supply can be introduced via the transistor's collector, ensuring flexible power configurations for gate control.

MaixPy FaceDetect Classifier

The MaixPy FaceDetect classifier uses a pre-trained object detection AI model, which simplifies the implementation of face recognition. This ready-made model can be downloaded here [1].

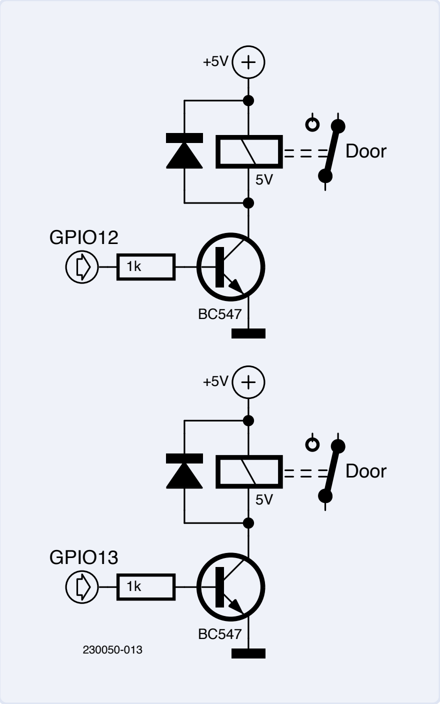


Figure 4: Connections for triggering the gate.

A key component is the Knowledge Processing Unit (KPU), a general-purpose neural network processor. The KPU supports convolution, batch normalization, activation, and pooling layers, making it capable of real-time object and facial detection.

The MaixPy FaceDetect classifier is available in a **.kfpkg* file, where * represents the model name. These models are highly efficient and fast-acting YOLO (You Only Look Once) Version 2 classifiers. Each scene is processed through the model, and the output is checked against its defined classifiers. YOLO v2's speed allows it to quickly identify a human face on the screen and draw a box around it. The classifier model used here is *facedetect.kfpkg*.

Aside from simpler face detection, it is also possible to use the MaixPy facial recognition classifier, but it should be noted that this program currently only supports facial recognition for a maximum of 10 faces. A more powerful development board would be needed to run a more robust facial recognition model.

Getting Started: Uploading a Project with MaixPy

By following these steps in order, you can set up the software and install the necessary files before running the project code.

Install MicroPython on the MaixPy Device

- First, connect your MaixPy device to your laptop via USB.
- Download *kflash_gui*, the flashing tool required to install software onto the device, from this link [2].
- Open *kflash_gui* and select the correct serial port for your device (**Figure 5**).
- If MicroPython is not installed, download the firmware file (e.g., *maixpy_v0.6.2_75_g973361c0d.bin*) from the MaixPy GitHub repository [3].
- Load the firmware file in *kflash_gui* and click the *Download* button to install MicroPython on the MaixPy device.

Install the Face Detection Model

- After installing MicroPython, you'll need to select the pre-trained face detection model [1] (*facedetect.kfpkg*) and upload it.
- In *kflash_gui*, select the *facedetect.kfpkg* file and press Download to install the AI model onto the device. You can also modify the *flash-list.json* file to change the registered model number (0x300000).

Upload a Project with MaixPy IDE:

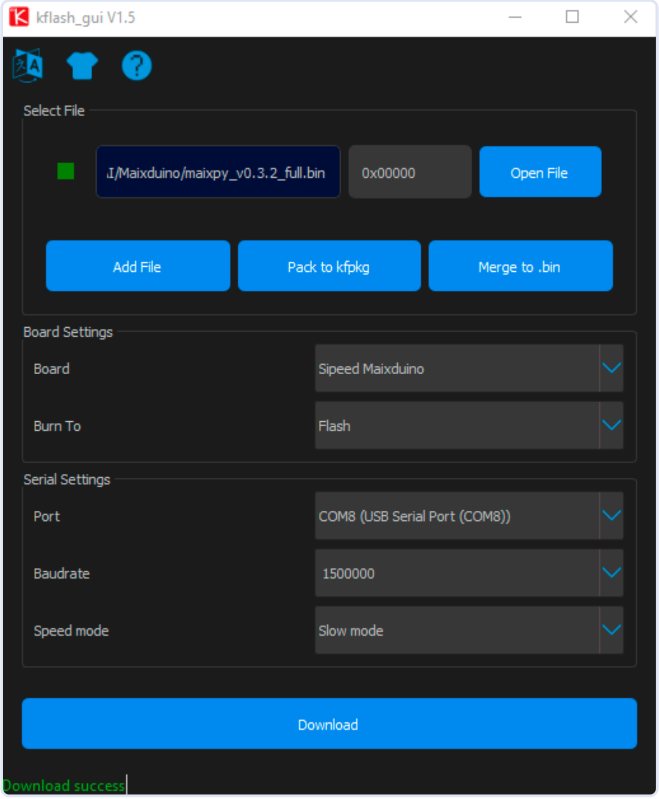
- Next, download and install the MaixPy IDE from here [4]. For a complete guide to MaixPy IDE, visit the MaixPy wiki [5].
- Open MaixPy IDE and connect your MaixPy device by pressing the chain symbol at the bottom left of the screen.
- Load the Python program (e.g., *doorman_mod3.py* as shown in **Figure 6**) by selecting File and opening it in the IDE window.
- To run the program, press the *play* button at the bottom left.

Set Auto-Play on Boot:

- To make the program auto-run every time the MaixPy device is powered on, go to *Tools* in the MaixPy IDE and select *Transfer File* to Board. This will copy the Python script to the device as *boot.py*, ensuring it automatically starts on boot.
- Alternatively, you can copy the Python file to the root directory of an SD card and rename it *boot.py*. Inserting the card into the MaixPy will make it auto-play as well.

Results

AI-powered face detection is nothing new today. It can easily be accomplished using a powerful computer and camera. However, implementing this on a standalone, low-powered (5 V) MaixPy microcontroller demonstrates that AI can make life easier, not just



kflash_gui V1.5

Select File

I/Maixduino/maixpy_v0.3.2_full.bin 0x00000

Board Settings

Board: Sipeed Maixduino

Burn To: Flash

Serial Settings

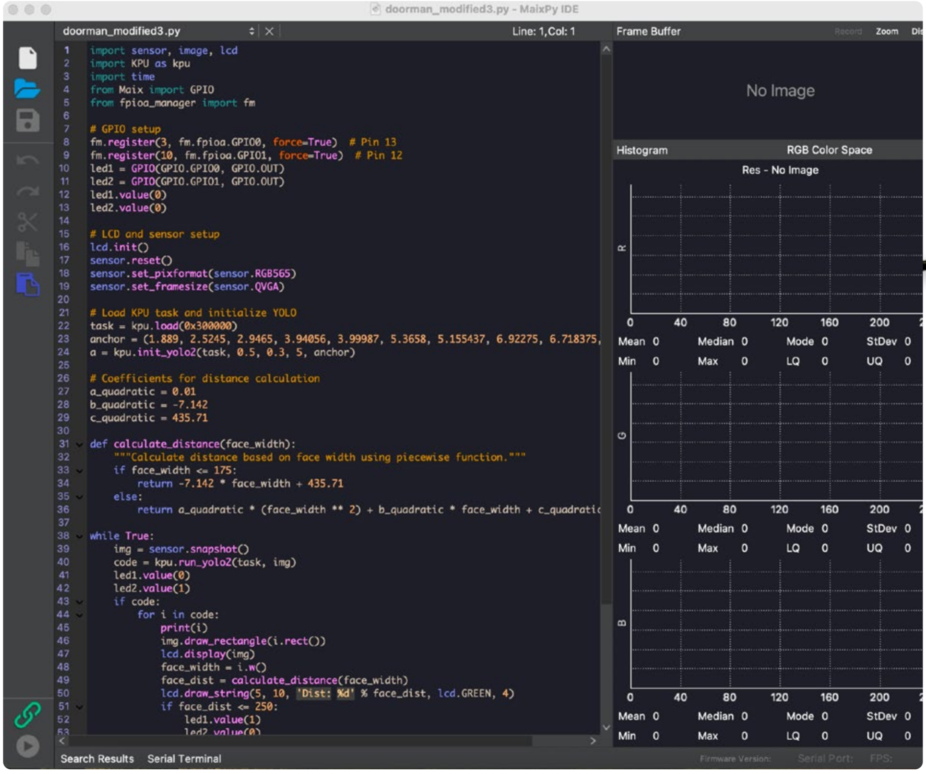
Port: COM8 (USB Serial Port (COM8))

Baudrate: 1500000

Speed mode: Slow mode

Download success

Figure 5: Steps to connect the MaixPy device using kflash_gui.



```python
import sensor, image, lcd
import KPU as kpu
import time
from Maix import GPIO
from fpioa_manager import fm

# GPIO setup
fm.register(3, fm.fpioa.GPIO0, force=True)  # Pin 13
fm.register(10, fm.fpioa.GPIO1, force=True)  # Pin 12
led1 = GPIO(GPIO.GPIO0, GPIO.OUT)
led2 = GPIO(GPIO.GPIO1, GPIO.OUT)
led1.value(0)
led2.value(0)

# LCD and sensor setup
lcd.init()
sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)

# Load KPU task and initialize YOLO
task = kpu.load(0x300000)
anchor = (1.889, 2.5245, 2.9465, 3.94056, 3.99987, 5.3658, 5.155437, 6.92275, 6.718375,
a = kpu.init_yolo2(task, 0.5, 0.3, 5, anchor)

# Coefficients for distance calculation
a_quadratic = 0.01
b_quadratic = -7.142
c_quadratic = 435.71

def calculate_distance(face_width):
    """Calculate distance based on face width using piecewise function."""
    if face_width <= 175:
        return -7.142 * face_width + 435.71
    else:
        return a_quadratic * (face_width ** 2) + b_quadratic * face_width + c_quadratic

while True:
    img = sensor.snapshot()
    code = kpu.run_yolo2(task, img)
    led1.value(0)
    led2.value(1)
    if code:
        for i in code:
            print(i)
            img.draw_rectangle(i.rect())
            lcd.display(img)
            face_width = i.w()
            face_dist = calculate_distance(face_width)
            lcd.draw_string(5, 10, 'Dist: %d' % face_dist, lcd.GREEN, 4)
            if face_dist <= 250:
                led1.value(1)
                led2.value(0)
```

Figure 6: IDE window showing the Python program.

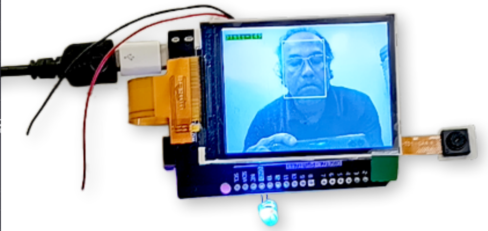


Figure 7: The prototype in action. Works flawlessly!

on powerful computers but also at the microcontroller level. The cost-effective Maixduino offers several advantages, including having a small yet fast TFT screen, the ability to run small AI projects and Micropython codes, and it also offers the sipeed.com repository support.

This AI-powered automatic doorman was installed on the CCR doors at the power station where I worked once (**Figure 7**). Initially, I used a linear proximity trigger with a 175 cm range [6], but I later discovered that a quadratic solution [7] provides greater precision beyond a certain distance, which has been incorporated into the latest software version which can be downloaded from [8]. After making this improvement, it operated flawlessly the entire time I was there.

230050-01

Questions or Comments?

If you have technical questions or comments about this article, feel free to contact the author by email at berasomnath@gmail.com or the Elektor editorial team at editor@elektor.com.

WEB LINKS

[1] MaixPy FaceDetect classifier download: https://dl.sipeed.com/MAIX/MaixPy/model

[2] kflash_gui: https://github.com/sipeed/kflash_gui

[3] MaixPy MicroPython firmware: https://github.com/sipeed/MaixPy/releases

[4] MaixPy IDE: https://dl.sipeed.com/MAIX/MaixPy/ide

[5] MaixPy IDE wiki: https://wiki.sipeed.com/soft/maixpy/en/get_started/env_maixpyide.html

[6] Video demonstration for a range of 1.5 meters: https://youtu.be/-2GsDlSfZVo

[7] Video demonstration for a range of 2.5 meters: https://youtu.be/2NoueDvC3KM

[8] Software download: https://www.elektormagazine.com/230050-01

FORTEC
POWER

Your expert for reliable power supplies

Our solutions:

- AC/DC power supplies
- DC/DC converters
- DC/AC inverters
- Customized solutions

One partner, unlimited technology solutions

Design-In Support:
support@fortec-power.de
+49 61 58 82 85-0
www.fortec-power.de

Embedded Electronics 2024

AI Is Set to Redefine the Industry

By Jean-Pierre Joosting (eeNews Europe)

Source: Alif Semiconductor

There were many trends in embedded electronics in 2024 ranging from AI at the edge, on-chip machine learning, the use of Generative AI for programming, collaborative robotics, wearable electronics, the automotive transition to software-defined vehicles (SDVs), rising interest in IoT cybersecurity, and multi-protocol wireless devices to mention a few. Some of these key trends are highlighted below.

On-Chip Artificial Intelligence

AI is the major trend in technology at the moment but for many things embedded, power consumption is a key issue. As a result, GPUs are typically integrated into CPUs, microcontrollers, and SoCs that are aimed at applications that can support high power requirements. However, for many embedded applications, a GPU consumes too much power and is often overkill.

A major trend that has come to the fore is the need for AI at the edge. Here power consumption is critical as these devices typically operate on small batteries or implement energy harvesting technology. Often, these applications also rely on smaller AI models that are more tailored to specific requirements.

To address power consumption in edge devices, chip developers and manufacturers have designed microcontrollers and SoCs with built-in machine-learning capabilities to enhance performance and efficiency using dedicated low-power neural processing units (NPUs) that can execute machine learning algorithms natively. Another technology that has come to the commercial market is event-based neuromorphic computing for very low-power and wake-up applications.

An emerging leader in microcontrollers with integrated NPUs, Alif Semiconductor [1] has pioneered a different architecture with its Ensemble Arm-based microcontroller family (**Figure 1**), which can scale from a single core to a new class of multicore devices. The Ensemble can combine up to two Cortex-M55 cores, up to two Cortex-A32 cores capable of running high-level operating systems, and up to two Ethos-U55 microNPUs for AI and machine learning acceleration. These power-efficient microcontrollers can handle heavy AI and ML workloads for battery-operated IoT devices up to 250+ GOPs. By optimizing the microcontroller architecture to tightly integrate the processing cores, low-power NPUs, and on-chip memory, the microcontroller family delivers at least two orders of magnitude higher edge AI performance than traditional microcontrollers with the lowest power consumption levels possible for vision, voice, and vibration analysis.

At electronica 2024, Alif Semiconductor also showed its latest Balletto B1 Bluetooth microcontroller family along with a corresponding development kit. The Balleto B1 integrates a 160 MHz Arm Cortex-M55 CPU core with Helium vector processing extension for fast and power efficient signal processing, an Arm Ethos-U55 NPU, a Bluetooth Low Energy and 802.15.4 radio, and a wide variety of digital and analog capabilities in tiny BGA or CSP package options.

In another example, BrainChip [2] has leveraged neuromorphic computing to solve AI problems. Cconsuming significantly less power consumption than a GPU, Akida technology is a brain-inspired cognitive event-based compute platform ideal for early detection, low-latency applications without massive compute resources for robotics, drones, automotive, and traditional sense-detect-classify-track systems. For example, built on the Akida event-based computing platform configuration engine, the Akida Pico low-power acceleration coprocessor accelerates limited use case-specific neural network models and can execute using battery-powered operation of less than a single milliwatt. Akida Pico enables the creation of very compact, ultra-low power, portable and intelligent devices for wearable and sensor-integrated AI into consumer, healthcare, IoT, defense, and wake-up applications.

A deep-tech Canadian startup, Blumind [3] has developed a disruptive all-analog neural signal processor technology — Blumind AMPL. This AI compute fabric is ideal for micropower AI applications and delivers deterministic and precise inferencing performance at up to ×1,000 lower power than legacy digital approaches. AMPL claims to be the first all-analog AI on advanced standard CMOS architected to fundamentally mitigate process, voltage, temperature and drift variations. Applications include always-on keyword detection, image classification, and always-on visual image wake to enable rapid identification for access control.

Another startup, Mythic [4] has pioneered analog compute-in-memory that is purpose-built for AI inference. This analog computing technology stores AI parameters directly in the processor, which eliminates memory bottlenecks and claims to enable AI applications that are impossible or unviable today. Since the traditional digital computing architecture needs to deal with transferring billions of AI parameters between memory and processing, it requires a lot of energy. Mythic removes this problem, claiming to be 18,000 times faster, 1,500 times more energy efficient, and 1,000 times more parallel.

Figure 1: The new E4, E6, and E8 Ensemble MCUs and fusion processors will integrate the latest neural processing unit (NPU) from Arm, the Ethos-U85, with a unique high bandwidth memory and intelligent power management architecture to support NPU operators required by transformer networks to execute generative AI workloads efficiently. (Source: Alif Semiconductor)

In 2025, expect to see microcontroller and low-power AI architectures continue to evolve. On-chip Generative AI is set to follow on-edge devices as chip manufacturers look to make AI more efficient. It is also likely, that these new technologies move up the food chain and replace traditional methods as the power consumption of AI is currently a global problem. Other technologies such as neuromorphic computing are expected to bring new levels of tailored performance to low-power edge applications as well as event-based detection for very low-standby power requirements.

Programming and Generative AI

A fast-growing technology, Generative AI has found use in generating code, debugging, and rapid prototyping in embedded electronics. For example, developers can use ChatGPT and GitHub Copilot [5] to provide instant code suggestions as well as help in debugging and optimization to resolve errors quickly. Generative AI has also become a useful tool in rapid prototyping as it can quickly generate bits of code to test functionality or explore different configurations.

While generative AI tools like ChatGPT and GitHub Copilot offer increased efficiency and faster prototyping, they also present challenges regarding code accuracy, confidentiality, and intellectual property. When using Generative AI, human oversight is important to ensure code meets its functional and performance metrics, as well as compiler and hardware requirements. Confidentiality and adherence to IP laws also need to be considered.

In 2025 and beyond expect to see more tools addressing the use of Generative AI to automate coding and programming.

Zephyr Open Source RTOS

Taking on FreeRTOS [6] and ThreadX [7], Zephyr [8] has become an important RTOS with significant growth in use in embedded electronics in 2024. This highly modular RTOS can scale to meet the requirements of various applications from tiny space-constrained IoT devices to complex embedded systems. It supports a wide range of hardware architectures and communication protocols and is designed to easily port to new platforms for reuse. Zephyr also offers a very secure kernel with support for multiple scheduling algorithms, memory protection, and fault handling. Security features include cryptography, secure boot, and firmware updates and a comprehensive set of tools are available for managing dependencies, toolchains, and builds.

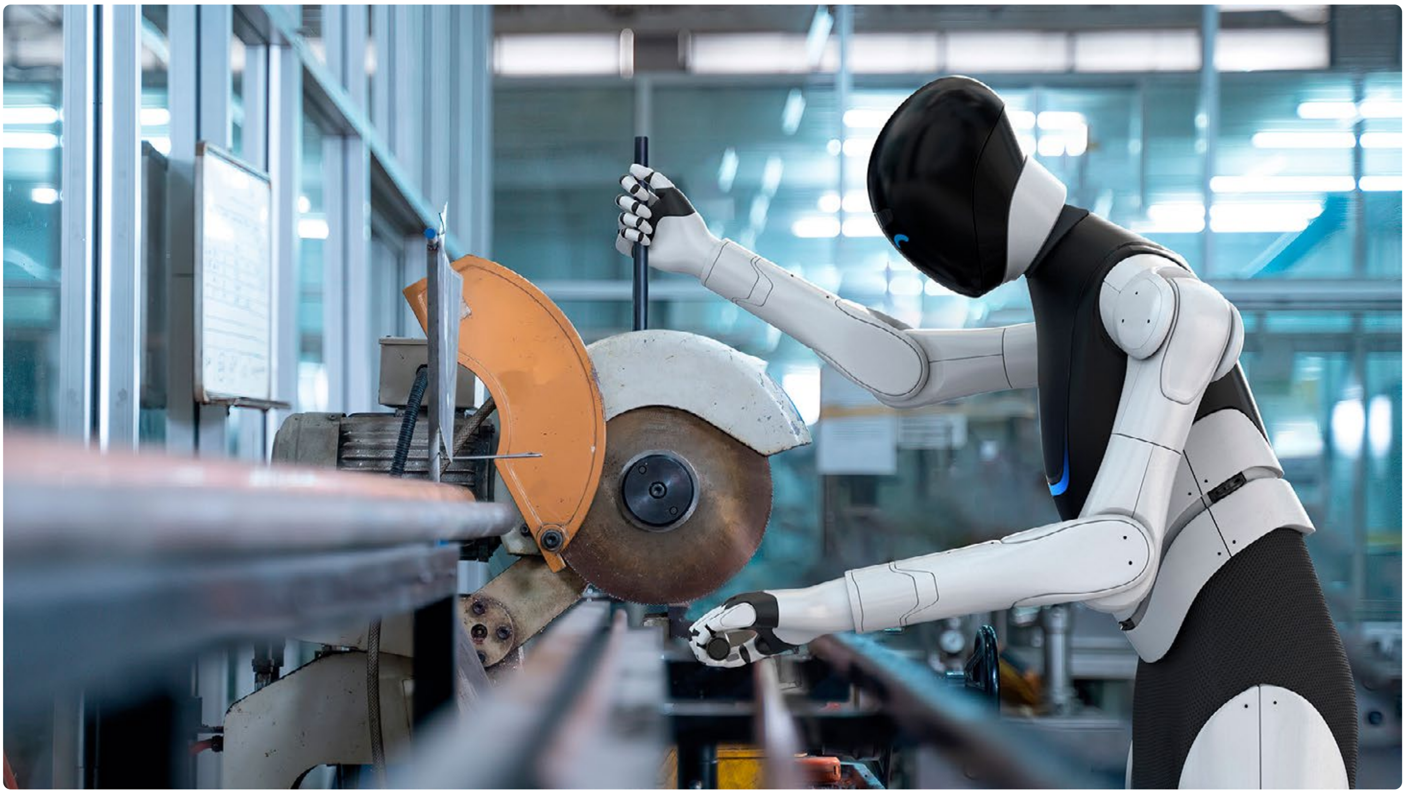


Figure 2: Humanoid robots are designed to work seamlessly together with humans in industries such as manufacturing, logistics, and healthcare. (Source: NEURA Robotics)

While FreeRTOS is backed by Amazon, ThreadX is part of the open-source Eclipse Foundation [9], and Zephyr is managed by the Linux Foundation—an important consideration for developers who prefer an RTOS not backed by a corporation. Zephyr offers broad hardware support, extensive connectivity options, and robust networking, making it particularly useful for developers transitioning from embedded Linux.

Highlighting its success, the Zephyr Project added several members in 2024, including CARIAD [10], Ac6[11], Alif Semiconductor, Arm [12], Honda [13], Inovex [14], MicroEJ [15], Qt Group [16], and STMicroelectronics [17]. According to the Zephyr Project, Nordic Semiconductor [18], Intel [19], and NXP [20] contributed the most to the ecosystem in 2024.

Several technical milestones were achieved in 2024, including over 100,000 commits and added support for 150 new boards (bringing the total to over 750), including development platforms such as Raspberry Pi Pico 2 [21] and the WCH CH32V003EVT [22] — a board powered by a 10 cent RISC-V MCU.

Going forward, Zephyr is expected to keep gaining traction in embedded systems, IoT, and safety-critical domains as more developers look to use open-source to cut costs, benefit from increased security, portability, wide support, and flexibility as well as to future-proof development.

Cybersecurity

With the increasing connectivity of embedded systems, 2024 saw advancements in cybersecurity that included hardware-based, full lifecycle, and zero-trust security, secure boot processes, and AI-driven threat detection. However, it is legislation in the form of the Cyber Resilience Act (CRA) [23] that became a key issue in 2024 that will affect markets in the years ahead.

The CRA sets a minimum level of cybersecurity for all connected products available within the EU market. It aims to ensure that there is an adequate level of cybersecurity in hardware and software products with a digital component as well as timely lifetime security updates, whether they are embedded systems or standalone software. It came into force on the 10th December 2024, starting a 36-month implementation period to full compliance.

This means that organizations need to design hardware and software for security to minimize vulnerabilities from the start and provide timely risk assessments and updates. In addition, clear information on security updates and known vulnerabilities must be provided to users throughout the product's lifetime. Expect organizations to use automation and risk management to meet compliance from 2025 onwards. On the 11th of September 2026, reporting obligations for manufacturers are expected to become applicable.

Robotics

According to the International Federation of Robotics (IFR) [24] the top five trends in robotics in 2024 are AI and machine learning, Cobots moving to new applications, digital twins, mobile manipulators, and humanoids. For example, generative AI-driven interfaces are being developed to enable users to program robots by using natural language instead of code. Machine learning is also enabling predictive maintenance in robotics to minimize downtime and cut costs.

Humanoid robots have caught the attention of the world with the Tesla Optimus humanoid slated for 2026. However, mass market adoption of humanoid robots remains a difficult challenge.

Founded in 2019, German startup NEURA Robotics [25] has become a global leader in cognitive and humanoid robotics, creating robots designed to work seamlessly with humans in industries such as manufacturing, logistics, and healthcare (**Figure 2**). Using unique sensor technology and AI integration, NEURA Robotics has developed the first cognitive cobot on the market and is now leading the way in the development of market-ready humanoid robots.

Linearly programmed and single-purpose robots are being superseded by versatile machines that are capable of reasoning. Moreover, large language models (LLMs), vision language models (VLMs), and robotics foundation models will provide robots with ever greater autonomy and awareness in the physical world. Foundation models are neural networks "pre-trained" on massive amounts of data without specific use cases in mind. These generalized models then train to get better at specific tasks through "transfer learning." Robots, in effect, learn on the job.

Automotive — SDVs

Apart from the EV trend of the last few years, software-defined vehicles (SDVs) have seen a jump in activity in 2024, driven by infotainment, cybersecurity, and LTE/5G connectivity. V2X and the ability of electric vehicles to support bidirectional power transfers to support the grid are key considerations. Typically, an SDV is built on what is known as a vehicle skateboard, which is similar to a common hardware platform — chassis, wheels, battery, and so on. On top of this, manufacturers add an operating system; SoCs for computing; a cloud platform such as Google Cloud, Azure, or AWS; user interfaces, and end-user applications. SDVs can then be updated via software over the lifetime of the vehicle to add new applications or adjust to new regulations and technologies. As a result, car companies are investing increasingly into embedded software.

For example, at CES 2025, Honda introduced its original vehicle operating system, the Asimo OS, which will be installed in future Honda 0 Series models. From 2025 onwards, expect to see more vehicle manufacturers redefine automotive design and lean towards developing their own software to differentiate their brands through infotainment, connectivity, end-user applications, voice recognition, ADAS, and in-cabin monitoring.

AI Is Dominating

Of all the trends in embedded electronics, AI is dominating the industry and is evolving quickly. In embedded applications, low-power AI at the edge looks set to completely transform the automotive, industrial, and consumer industries. Moving forward, edge AI will benefit from more powerful AI chips that bring LLMs onto microcontrollers and very low-power processors. Analog and neuromorphic computing will enable applications such as keyword and image detection at extremely low power levels.

Future vehicles using AI, for example, will be customized to the driver's needs, boost the effectiveness of driver aids, and enable predictive maintenance to address problems before they occur. In the smart home, expect software and AI to dominate, enabling interoperability (hardware-agnostic) and LLMs to provide natural language interaction for the whole home. In industry, machine learning, digital twins, robotics, and AI will continue to automate everything from setup to operation of smart factories. Predictive maintenance will also play a large role in cutting costs. However, a lot of these applications will require low-power AI chips. ◀

250072-01

WEB LINKS

[1] Alif Semiconductor: https://alifsemi.com
[2] BrainChip: https://brainchip.com
[3] Blumind: https://blumind.ai
[4] Mythic: https://mythic.ai
[5] GitHub Copilot: https://github.com/features/copilot
[6] FreeRTOS: https://freertos.org
[7] ThreadX: https://threadx.io
[8] Zephyr: https://zephyrproject.org
[9] Eclipse Foundation: https://eclipse.org
[10] CARIAD: https://cariad.technology
[11] Ac6: https://ac6-training.com
[12] Arm: https://arm.com
[13] Honda: https://global.honda/en
[14] Inovex: https://inovex.de/en
[15] MicroEJ: https://microej.com
[16] Qt Group: https://qt.io
[17] STMicroelectronics: https://st.com
[18] Nordic Semiconductor: https://nordicsemi.com
[19] Intel: https://intel.com
[20] NXP: https://nxp.com
[21] Raspberry Pi: https://raspberrypi.org
[22] WCH CH32V003EVT: https://wch-ic.com
[23] Cyber Resilience Act:
https://eur-lex.europa.eu/eli/reg/2024/2847/oj/eng
[24] International Federation of Robotics: https://ifr.org
[25] NEURA Robotics: https://neura-robotics.com

Charge-Based In-Memory Compute at EnCharge AI

Naveen Verma, CEO of EnCharge AI

By Nick Flaherty (eeNews Europe)

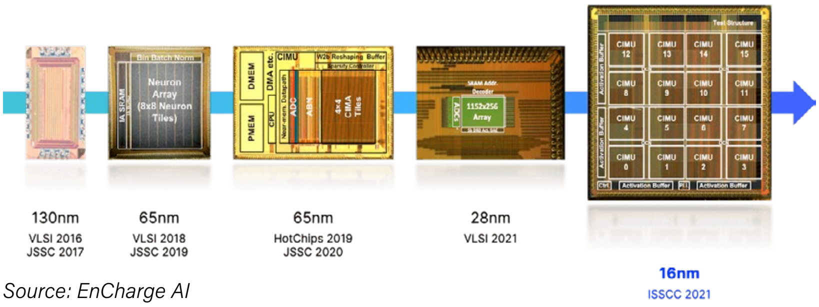
Naveen Verma, CEO of EnCharge AI, talks to *eeNews Europe* about the company's analog in-memory compute technology for edge AI.

US startup EnCharge AI [1][2] is reaching the next stage of its development of in-memory compute (IMC) for low-power AI. The company has raised $45 m for its capacitor-based analog in-memory computing technology and is actively raising funds for the commercialisation of AI chips for laptops.

"There has been a lot of interest in analog computing as it can be orders of magnitude energy efficient but it is sensitive to noise so the big problem that we have solved is managing analog IMC in a very robust and scalable way," said Naveen Verma, CEO of EnCharge AI and professor of electrical and computer engineering at Princeton University since 2009.

"AI is moving out of the data centre to scale up, moving into laptops and desktops, edge servers. These are often battery-powered or space-constrained, and that's where we have found a lot of traction."

"My research group at Princeton has been looking at this for many years, with all sorts of optical and electrical computing. The big fundamental difference is rather than using the current through a semiconductor device we are using charge," he said.



Source: EnCharge AI

"We don't use semiconductor devices, we use capacitors formed from the metal wires, they only depend on the geometry and that's something we can control very well."

"This solves the analog accuracy and scalability problems from the geometry control as we scale to more advanced nodes so the energy efficiency and density improve. The precision of patterning the wires does scale, and what geometry you need is an interesting question. What we need is more of an order of magnitude less (than approaches), and that allows you to use the upper metal layers."

EnCharge AI has built a macroblock that does the multiply-accumulate (MAC) functions very efficiently for AI, using SRAM to provide the addressability for the capacitors in the metal layers that can store the weights for the frameworks. This approach has been demonstrated in chips ranging from 130 nm in 2017 to 16 nm in 2021.

Building a Chip Is Not Enough

"The trick is to re-architect the chip to get the advantages," he said. "The fundamental technology was a breakthrough in 2017 and the next five years was to build an architecture around it with a software stack and a compiler to map applications to it."

"Because the fundamental capacitor technology does not require any additional technology, that allows us to be in the most advanced nodes and that gives us flexibility to build programmability into this with all the AI models."

"We defined an architectural unit, a tile, that is complete, highly efficient MAC engines with all the other compute and control, localised vector units and L2 memory cache, and you can build an array of these tiles."

Re-architecting chips with the EnCharge analog AI IP is key to running AI frameworks with billions of parameters.

"Then the problem is to map the software to this architecture to use the units efficiently. This gives high levels of density and power efficiency. Now you can very flexibly choose the parallelism of the AI graph."

"So we had the full stack in a baseline from where we understood the architecture, the compiler structure, we spun that out in 2022 for customers that needed the higher efficiency and performance."

"What we found was the first couple of models broke everything as there were small features in the first couple of AI workloads that were missing. Working with customers, we identified the features for the next spin of the hardware and the software. That's the journey that we were on," he said.

"We have been working with partners to prove out the workloads that need to run. The systems need to be optimised from top to bottom. We are building the chips and the software for a couple of reasons. Really getting the full performance advantage requires a top-to-bottom optimisation and being able to control that allows us to get the most out of the technology."

The initial form factor for this is simple discrete boards on PCIe cards, and he sees M.2 as well aligned for laptops. The existing technology can provide performance of over 150 8-bit TOPS/W, but the chip under development for 2025 has an efficiency of 375 TOPS/W and can also handle AI frameworks with billions of parameters.

"Our target is to be 3 to 4× the capability of laptop processors have 40 TOPS with an NPU in Snapdragon or Intel's Lunar Lake," he said. "We have a significant advantage at 16 nm over 5 nm digital so our preference is to stay in those nodes where we have proven our IP. The preference is to stay in very mainstream technologies and the technologies that will allow us to scale and these work very well for us."

Re-architecting chips with the EnCharge analog AI IP is key to running AI frameworks with billions of parameters. This uses well-established principles of memory virtualisation.

"Virtualised memory takes a capacity-limited L1 memory and couples that with the L2 cache all the way out to DRAM and we have used the same principles to IMC, orchestrating the way memory moves across the system so that we can scalably execute models with billions of parameters," he said.

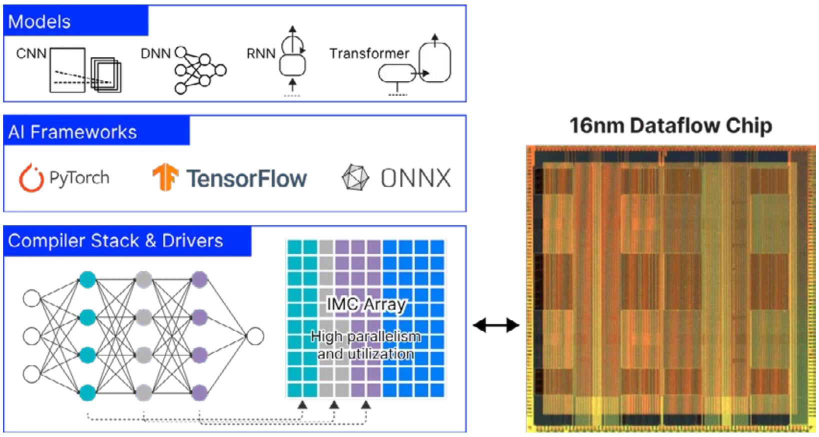
"What you have to have is some IMC hardware and work out how it works with the L2 and L3 and off-chip memory for the right amount of data reuse so that the data movement doesn't become the limiting factor. That drives the architectural design, and our systems are optimised for multiple multi-billion parameter models."

While the latest laptop devices such as Meteor Lake use chiplets, this is not a focus for the in-memory compute architecture, says Verma.

"Chiplets are a very interesting technology but there are constraints, whether using UCIe as an interface or not. I do see opportunities around chiplets."

240728-01

Editor's Note: Nick Flaherty first reported on this in eeNews Europe, a publication in the Elektor network. www.eenewseurope.com/en/domain/eenews-embedded/



Source: EnCharge AI

WEB LINKS

[1] EnCharge AI: https://www.enchargeai.com
[2] R. Pell, "In-memory computing startup launches to enable edge AI at scale," eeNews, 2022: https://www.eenewseurope.com/en/in-memory-computing-startup-launches-to-enable-edge-ai-at-scale/

Source: Adobe Stock

AI Inferencing at 10 Times Lower Power and 20-Fold Lower Cost

By Jean-Pierre Joosting (eeNews Europe)

Sagence AI has emerged from stealth to unveil a groundbreaking advanced analog in-memory compute architecture that directly addresses the untenable power/performance/price and environmental sustainability conundrum facing AI inferencing.

Based on industry-first architectural innovations using analog technology, Sagence AI [1] makes possible multiple orders of magnitude improvement in energy efficiency and cost reductions for AI inferencing, while sustaining performance equivalent to high performance GPU/CPU based systems.

Compared to the leading volume GPU processing the Llama2-70B large language model with performance normalized to 666 k tokens/s, Sagence technology performs with 10 times lower power, 20 times lower price, and 20 times smaller rack space. Using a modular chiplet architecture for maximum integration, the technology enables a highly efficient inference machine that scales from data center generative AI to edge computer vision applications across multiple industries. The technology balances high performance and low power at an affordable cost — addressing the growing ROI problem for generative AI applications at scale, as AI computing in the data center shifts from training models to the deployment of models to inference tasks.

"A fundamental advancement in AI inference hardware is vital to the future of AI. Use of large language models (LLMs) and Generative AI drives demand for rapid and massive change at the nucleus of computing, requiring an unprecedented combination of highest performance at lowest power and economics that match costs to the value created," said Vishal Sarin, CEO & Founder, Sagence AI. "The legacy computing devices today that are capable of extreme high-performance AI inferencing cost too much to be economically viable and consume too much energy to be environmentally sustainable. Our mission is to break those performance and economic limitations in an environmentally responsible way."

"The demands of the new generation of AI models have resulted in accelerators with massive on-package memory and consequently extremely high-power consumption. Between 2018 and today, the most powerful GPUs have gone from 300 W to 1200 W, while top-tier server CPUs have caught up to the power consumption levels of NVIDIA's A100 GPU from 2020," said Alexander Harrowell, Principal Analyst, Advanced Computing, Omdia. "This has knock-on effects for data center cooling, electrical distribution, AI applications' unit economics, and much else. One way out of the bind is to rediscover analog computing, which offers much lower power consumption, very low latency, and permits working with mature process nodes."

Sagence AI leads the industry on the frontier of in-memory computing innovation. It is the first to do deep subthreshold computing inside multi-level memory cells, an unprecedented combination that opens doors to the orders of magnitude improvements necessary to deliver inference at scale. As digital technology reaches limits in ability to scale power and cost, "run" a neural network on CPU/GPU digital machines requires extremely complicated hardware reuse and hardware scheduling. The natural hardware solution is not a general-purpose computing machine, but rather an architecture that more closely mirrors how biological neural networks operate.

> *In-memory computing aligns closely with the essential elements of efficiency in AI inference applications.*

Sagence innovated a new analog path forward leveraging the inherent benefits of analog in energy efficiency and costs to make possible mass adoption of AI that is both economically viable and environmentally sustainable.

In-memory computing aligns closely with the essential elements of efficiency in AI inference applications. Merging storage and computing inside memory cells eliminates single-purpose memory storage and complex scheduled multiply-accumulate circuits that run the vector-matrix multiplication integral to AI computing. The resulting chips and systems are much simpler, lower cost, lower power and with vastly more compute capability.

Sagence views the AI inferencing challenge not as a general-purpose computing problem, but as a mathematically intensive data processing problem. Managing the massive amount of arithmetic processing needed to

The statically scheduled deep subthreshold in-memory compute architecture employed by Sagence chips is much simpler and eliminates the variabilities and complexities of the dynamic scheduling required of CPUs and GPUs. Dynamic scheduling places extreme demands on the SDK to generate the runtime code and contributes to cost and power inefficiencies. The Sagence AI design flow imports a trained neural network using standards-based interfaces like PyTorch, ONNX and TensorFlow, and automatically converts it into Sagence format. The Sagence system receives the neural network long after GPU software created it, negating the further need for the GPU software. ◄

240730-01

Editor's Note: Jean-Pierre Joosting first reported on this in eeNews Europe, a publication in the Elektor network.
www.eenewseurope.com/en/domain/eenews-embedded/

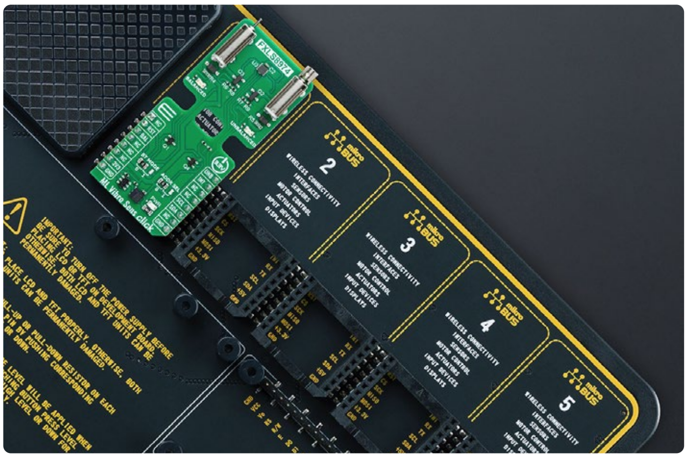
WEB LINK

[1] Sagence AI: https://sagence-ai.com

Click Board Helps Develop and Train ML Models for Vibration Analysis

By Jean-Pierre Joosting (eeNews Europe)

MIKROE has launched the compact add-on ML Vibro Sens Click board, which is designed for precise motion and vibration sensing and analysis.



Based on the FXLS8974CF, a 3-axis low-g accelerometer from NXP, this Click board offers high performance and versatility ideal for developing and training machine learning (ML) models for vibration analysis.

Comments Nebojsa Matic, CEO of MIKROE [1]: "A new member of our company's 1750-strong mikroBUS-enabled Click board family of compact add-on boards, ML Vibro Sens Click can be used to collect data for training ML models to recognize different types of vibrations, and to monitor the health of machines and industrial equipment based on vibration patterns. It can also be used to track motion and activity in wearable devices, and to detect vibrations caused by earthquakes or other seismic events."

The FXLS8974CF offers the versatility of ultra-low-power operation alongside high-performance modes, ensuring efficient use in diverse scenarios. Its integrated digital features simplify data collection and reduce system power consumption, while its robust performance over extended temperature ranges enhances reliability in demanding applications, including industrial diagnostics, wearable technology, and environmental monitoring.

This Click board incorporates two DC motors to simulate vibration stimuli for machine learning. The *balanced motor* generates steady "nominal" vibrations, serving as a baseline signal for training ML models in a "healthy" state. The *unbalanced motor* is designed to provide customizable vibration signals, ranging from low-intensity to specific frequency-based vibrations.

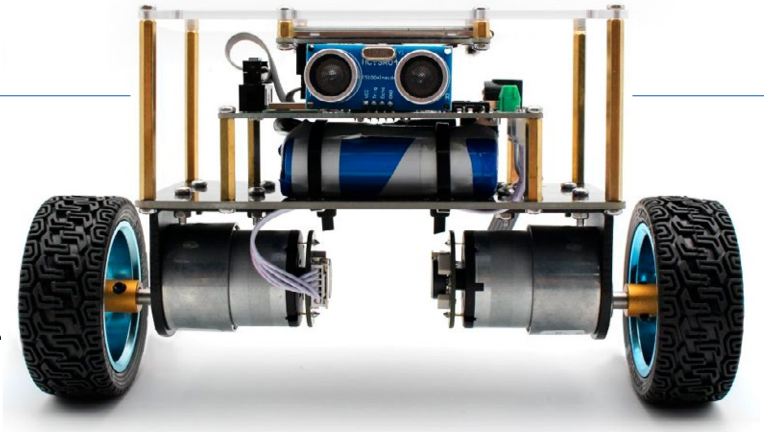
The FXLS8974CF accelerometer from NXP captures detailed data from the balanced and unbalanced motors, enabling the differentiation between healthy baseline states and anomalous conditions. It communicates with the host MCU via a standard 2-wire I²C interface.

240733-01

Editor's Note: Jean-Pierre Joosting first reported on this in eeNews Europe, a publication in the Elektor network. www.eenewseurope.com/en/domain/eenews-embedded/

WEB LINK

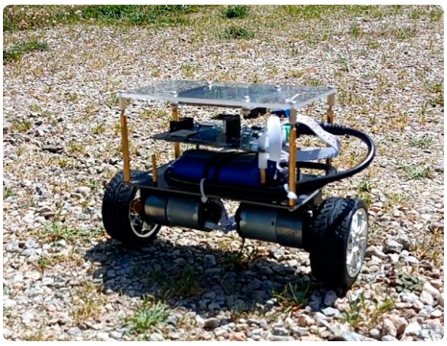
[1] MIKROE: https://mikroe.com



The Elektor Mini-Wheelie

A Self-Balancing Robot Kit

By The Elektor Team



The prototype of the self-balancing Mini-Wheelie robot during one of its test runs.

In 2009, Elektor published the Elektor-Wheelie, a DIY, two-wheeled, self-balancing, battery-powered vehicle inspired by the Segway PT, which was then hailed as the future of personal transportation. The Elektor-Wheelie brought this exciting new self-balancing technology to within makers' reach.

Now, some 15 years later, we introduce what we might call the Mini-Wheelie. While it operates on the same principles as its larger predecessor, its purpose has changed. Instead of transporting you from point A to B (riding it is strongly discouraged), the Mini-Wheelie serves as an experimental, autonomous, self-balancing robot platform. Powered by an ESP32-S3 microcontroller, the robot is fully programmable using the Arduino environment and open-source libraries. Its wireless capabilities enable remote control via Wi-Fi, Bluetooth, or ESP-NOW. They also allow for communication with a user or even another robot. An ultrasonic sensor facilitates obstacle detection, while its color display can show cute facial expressions or — for the more practically inclined — cryptic debug messages.

The robot comes as a neatly packaged kit that requires self-assembly. Everything you need, even a screwdriver, is included.

250112-01

The robot comes as a kit of parts that you must assemble yourself. Everything is included — even a screwdriver.

Specifications

- ESP32-S3 microcontroller with Wi-Fi and Bluetooth
- MPU6050 inertial measurement unit (IMU)
- Two independently controlled 12 V electric motors with tachometer
- Ultrasonic transducer
- 320×240-pixel TFT color display
- microSD card slot
- Battery power monitor
- 3S (11.1 V, 2200 mAh) rechargeable Li-Po battery (including charger)
- Arduino-based open-source software
- Dimensions (w/l/h): 23×8×13 cm

WEB LINK

[1] This project on Elektor Labs: https://elektormagazine.com/labs/self-balancing-robot-with-maker-fabs

MCU, I See You

MCUViewer Open-Source Multiplatform Debugging Tool

By Piotr Wasilewski (Poland)

MCUViewer is an open-source graphical debugging tool for microcontrollers, featuring two core modules that allow developers to visualize and interact with microcontroller data in a non-intrusive way. The Variable Viewer module enables real-time monitoring, logging, and manipulation of variables through a debug interface (SWDIO, SWCLK, GND). Trace Viewer graphically represents real-time SWO trace outputs, offering deeper insights into high-speed data. The multiplatform tool requires either an STLink or a J-Link programmer for operation and will work for ARM-based and other types of microcontrollers that are compatible with either type of probe.

The MCUViewer project began as a straightforward attempt to create a workaround alternative to STMicroelectronics's STM Studio, but it soon grew into a fully independent, open-source GPLv3 [1] tool for debugging a wide variety of embedded systems. The idea emerged from my own experiences as an embedded systems engineer. Over a year ago, while working intensively on motor control algorithms, I was using STM Studio to visualize system behaviors and debug controllers. Although STM Studio has served its purpose over the years for many developers, it has certain limitations, as it does not support Unix platforms or the import of C++ objects. There are some other similar tools available, such as CubeMonitor from STMicroelectronics or FreeMASTER from NXP, but they are limited to specific vendors and also have varying capabilities and features.

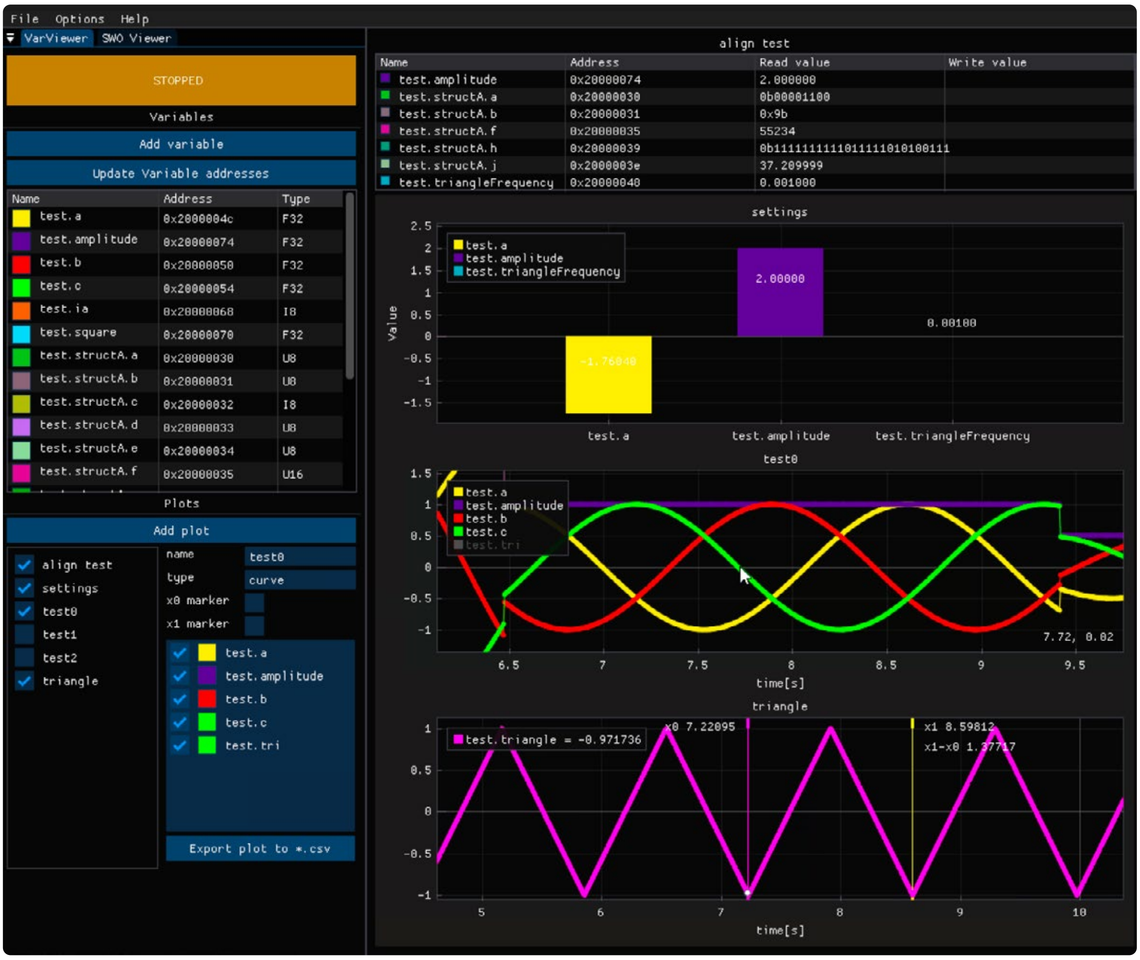
That's when the spark struck: why not create a custom tool that could address these gaps, not just for me but for the broader developer community? Initially, I was hesitant to start such a large project, especially without the resources of a full software team. However, as I made progress, I found the process rewarding. It allowed me to guide the development in a direction that made MCUViewer unique, particularly in the hobbyist community.

I likely would not have taken on this task without the positive feedback from others during development. It soon became clear that I was not developing this tool just for my own benefit — many other developers have found a tool like MCUViewer useful and have expressed their encouragement and appreciation. SEGGER also provided me with valuable assistance to integrate J-Link support into MCUViewer, for which I am grateful.

How Does It Work?

The *Variable Viewer* module of MCUViewer can sample microcontroller memory at constant time intervals, making it useful for monitoring real-time data. By parsing the provided ELF file, the software identifies variables and their corresponding memory addresses, which are accessed via a debug probe to read the values.

As **Figure 1** shows, this memory data can be visualized in one of three formats: a curve view for time-based graphs, a bar view for visual insights, or a table view for reading and modifying values. The table view also allows developers to modify variable values



Name	Address	Type
🟨 test.a	0x2000004c	F32
🟪 test.amplitude	0x20000074	F32
🟥 test.b	0x20000050	F32
🟩 test.c	0x20000054	F32
🟧 test.ia	0x20000068	I8
🟦 test.square	0x20000070	F32
🟩 test.structA.a	0x20000030	U8
■ test.structA.b	0x20000031	U8
■ test.structA.c	0x20000032	I8
■ test.structA.d	0x20000033	U8
■ test.structA.e	0x20000034	U8
■ test.structA.f	0x20000035	U16

☑ align test
☑ settings
☑ test0
□ test1
□ test2
☑ triangle

name: test0
type: curve
x0 marker: □
x1 marker: □

☑ 🟨 test.a
☑ 🟪 test.amplitude
☑ 🟥 test.b
☑ 🟩 test.c
☑ 🟩 test.tri

align test

Name	Address	Read value	Write value
🟪 test.amplitude	0x20000074	2.000000	
🟩 test.structA.a	0x20000030	0b00001100	
■ test.structA.b	0x20000031	0x9b	
■ test.structA.f	0x20000035	55234	
■ test.structA.h	0x20000039	0b11111111110111110101001111	
■ test.structA.j	0x2000003e	37.209999	
🟦 test.triangleFrequency	0x20000040	0.001000	

Figure 1: In Variable Viewer mode, users can visualize memory data as time-based graphs, bar charts, or tables, and edit variables directly for dynamic testing and development.

directly, providing a way to adjust program flow and trigger specific actions during development or testing.

In addition, MCUViewer offers basic statistical analysis of the gathered signals and enables exporting the data to CSV format for further analysis. Variable Viewer is especially useful for visualizing many signals at once; for example when tuning a PID controller you could display all three P, I and D components together with the setpoint and output value.

However, Variable Viewer is not sufficient in all cases — especially for very high-frequency signals such as inductor current responses. Depending on the number of logged variables and the type of probe used, Variable Viewer can reach up to a few tens of kilohertz. While this frequency may be adequate in many situations, there are times when it falls short.

To address this, the *Trace Viewer* module was developed. Unlike Variable Viewer, which samples memory, Trace Viewer processes SWO trace data from the embedded firmware. Variables are written to the ITM peripheral with hardware-generated timestamp packets, ensuring synchronization with code execution. Thanks to the optimized SWO protocol, Trace Viewer can achieve sub-microsecond timestamp resolutions, making it suitable for high-frequency signal analysis, and much more.

How To Use Variable Viewer

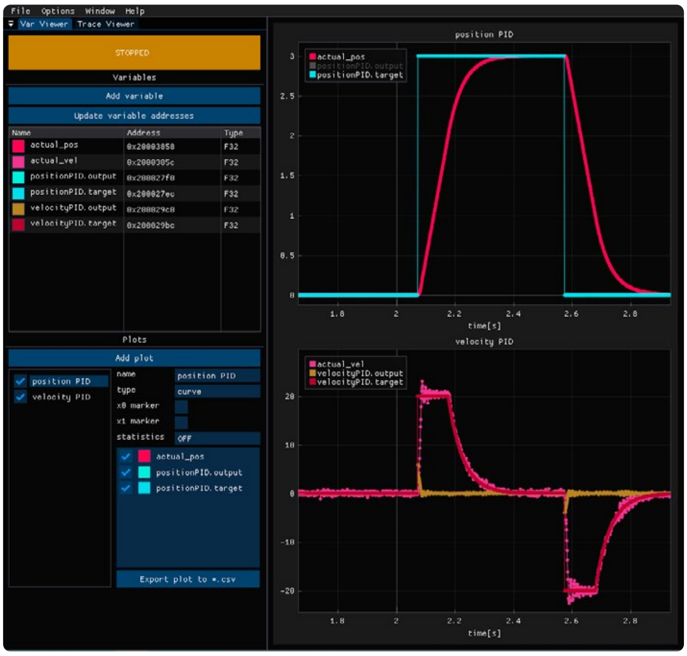
Initially designed as the core feature of MCUViewer, Variable Viewer collects the values of variables using a selected debug probe. Currently, ST-Link and J-Link probes are supported and they can work in different modes: ST-Link probes support only normal mode which samples variables' values one by one, whereas J-Link can work in both normal and HSS (High-Speed Sampling) mode in which it's capable of sampling at rates of tens of kilohertz. Depending on the debugging scenario, you can easily switch between the two modes.

After picking the debug probe we can set up the tracked variables and create some plots. The main rule is that all tracked variables have to be global; they have to persist throughout the program lifetime in order to have a constant address and be detected by the parser. At the moment all basic types up to 32-bits are supported.

After the import, plots can be added by selecting their type, then drag and drop the variables onto the canvas. Clicking the large button in the top left window's corner will start the acquisition. When stopped, the user can zoom in on particular parts of the collected data and analyze it. Additional tools are cursors which can be used to quickly measure time difference and check signal values, and simple statistics windows that measure and display some basic properties of the signal.

For example, Variable Viewer can be used with a cascaded PID of a servo drive. In **Figure 2**, I have created two plots, one for the outer position control loop and the inner one for the velocity control loop.

At first glance, it seems the step response of the position controller looks good — a nice critically damped response. However, when



Name	Address	Type
🟥 actual_pos	0x20003050	F32
🟥 actual_vel	0x2000305c	F32
🟩 positionPID.output	0x200027f0	F32
🟦 positionPID.target	0x200027ec	F32
🟧 velocityPID.output	0x200029c0	F32
🟥 velocityPID.target	0x200029bc	F32

☑ position PID
☑ velocity PID

name: position PID
type: curve
x0 marker:
x1 marker:
statistics: OFF

☑ actual_pos
☑ positionPID.output
☑ positionPID.target

Figure 2: Variable Viewer is used here in a cascaded servo drive PID with separate plots created for the position control loop and for the velocity control loop.

looking at the velocity PID response you can easily spot some oscillations. These could be not noticed when simply tuning the controllers without this type of visual feedback.

Besides only reading out the response, it is possible to command values by using a table view with setpoint values. This can be used to quickly test different setpoints or adjust the PID gains on the fly.

How To Use Trace Viewer

Trace Viewer works a bit differently compared to the Variable Viewer, and this difference brings a range of useful features, particularly for dealing with high-frequency signals. One notable feature enabled by its synchronous nature is execution time profiling. Typically, when developers want to measure how long a specific function takes to execute, they rely on a logic analyzer or an oscilloscope, using a GPIO pin that toggles on function entry and exit. While this approach is largely non-intrusive, it requires extra setup, such as connecting additional pins to external measurement devices. Moreover, if there is a need to measure relative timings between multiple functions or events, more GPIO pins are needed, which can quickly become impractical, if not impossible — especially on low-pin-count microcontrollers.

Trace Viewer addresses these challenges by using a single SWO pin connected to the debug probe, which is connected to the target during the development anyway. Thanks to a highly optimized serial protocol, it can manage up to 32 channels. This means that depending on the frequency of each channel and total SWO bandwidth we can use 32 channels on a single GPIO pin. The main limitation is that SWO needs additional peripherals that are present only in Cortex M3/M4/M7/M33 cores.

The SWO pin can be used to track the execution times of multiple functions and the real-time values of several variables. However, if these monitoring operations produce more data than the SWO pin can handle, then the Trace Viewer module can debug the probe SWO pin bandwidth — in other words, to help you identify where the bottleneck is and modify how the data is logged or transmitted to fit within the available bandwidth.

This is how you can measure a `memcopy` function execution time with different sizes of the copied buffer:

1. On entering a function we place an entrance marker — `ITM->PORT[X].u8 = 0xaa` — when plotted on the MCUViewer plot it will be shown as a transition from low to high state.

2. On the function's exit we place the exit marker — `ITM->PORT[X].u8 = 0xbb` — when plotted it will be shown as a transition from high to low state.

As **Figure 2** shows, both markers are single register writes to the ITM peripheral. In TraceViewer each channel has its own plot, and the X stands for the number of channels used. After the write is done the frame is automatically composed by the hardware and sent using the SWO pin.

Below you can see a simple code example for copying a buffer using the `memcopy` function:

```
volatile uint8_t dest[1024];
volatile uint8_t src[1024];
volatile uint16_t size = 1;
for (size_t i = 0; i < 10; i++)
{
  ITM->PORT[1].u16 = size;
  ITM->PORT[0].u8 = 0xaa;
  memcpy(dest, src, size);
  ITM->PORT[0].u8 = 0xbb;
  size += 64;
  for (size_t a = 0; a < 300; a++)
    __asm__ volatile("nop");
}
```

We're measuring the execution time on channel 0 and plotting the size on channel 1.

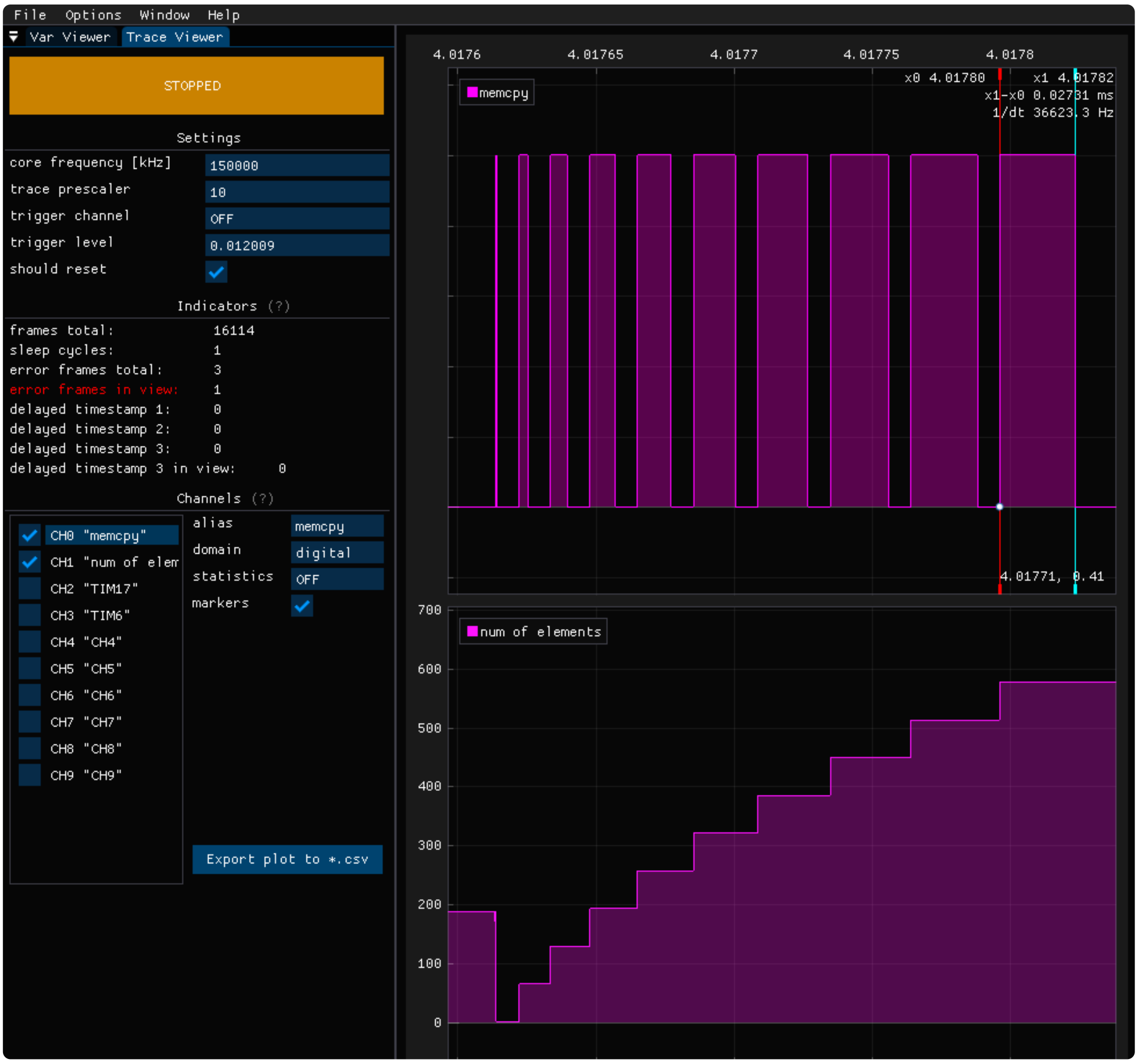
In **Figure 3** you can see the result in the Trace Viewer module window.

The execution times can be easily measured with cursors — in this case, copying 577 bytes took approximately 27,31 µs.

Analog Signal Example

In order to plot an analog float-type signal, a slightly different approach is taken:

```
float a = sin(10.0f * i);
//some high speed signal to trace
ITM->PORT[0].u32 = *(uint32_t*)&a;
// type-pun to desired size
```



```
Settings
core frequency [kHz]      150000
trace prescaler           10
trigger channel           OFF
trigger level             0.012009
should reset              ✓

Indicators (?)
frames total:             16114
sleep cycles:             1
error frames total:       3
error frames in view:     1
delayed timestamp 1:      0
delayed timestamp 2:      0
delayed timestamp 3:      0
delayed timestamp 3 in view:      0

Channels (?)
✓ CH0 "memcpy"          alias        memcpy
✓ CH1 "num of elem      domain       digital
  CH2 "TIM17"           statistics   OFF
  CH3 "TIM6"            markers      ✓
  CH4 "CH4"
  CH5 "CH5"
  CH6 "CH6"
  CH7 "CH7"
  CH8 "CH8"
  CH9 "CH9"

x0 4.01780     x1 4.01782
            x1-x0 0.02731 ms
            1/dt 36623.3 Hz

4.01771, 0.41
```

Figure 3: Trace Viewer can identify where bottlenecks are in the SWO pin bandwidth. This is shown here by measuring a memcpy function execution time with different sizes of the copied buffer.

```
// sizeof(float) = sizeof(uint32_t)
```

Basically, we need to type-pun the value to a suitable type, and select the proper type in MCUViewer plot settings. First, set the plot domain to analog, and then select the type from the drop-down list.

In **Figure 4**, a response of an inductor current for a given voltage command is plotted.

When registered, we can use cursors to measure the time constant and estimate the inductance. Registering analog signals can be very useful in the power electronics field, where controllers usually operate at tens of kilohertz, which makes them more challenging to analyze and debug.

Interrupt Profiling Example

The last demonstration example uses Trace Viewer to analyze the execution order and priorities of interrupts. By placing markers on the interrupt enter and exit we can easily visualize the relative interrupt timings as well as execution times and preemption events.

Figure 5 shows a TIM7 interrupt with the highest priority, TIM17 with medium priority, and TIM6 with lowest priority. The preemption can be easily spotted in places where the frequency of TIM6 and TIM17 is affected by TIM7 interrupt execution. This use case gives a lot of additional info about the system as CPU-intensive tasks can be easily spotted and fixed if needed.

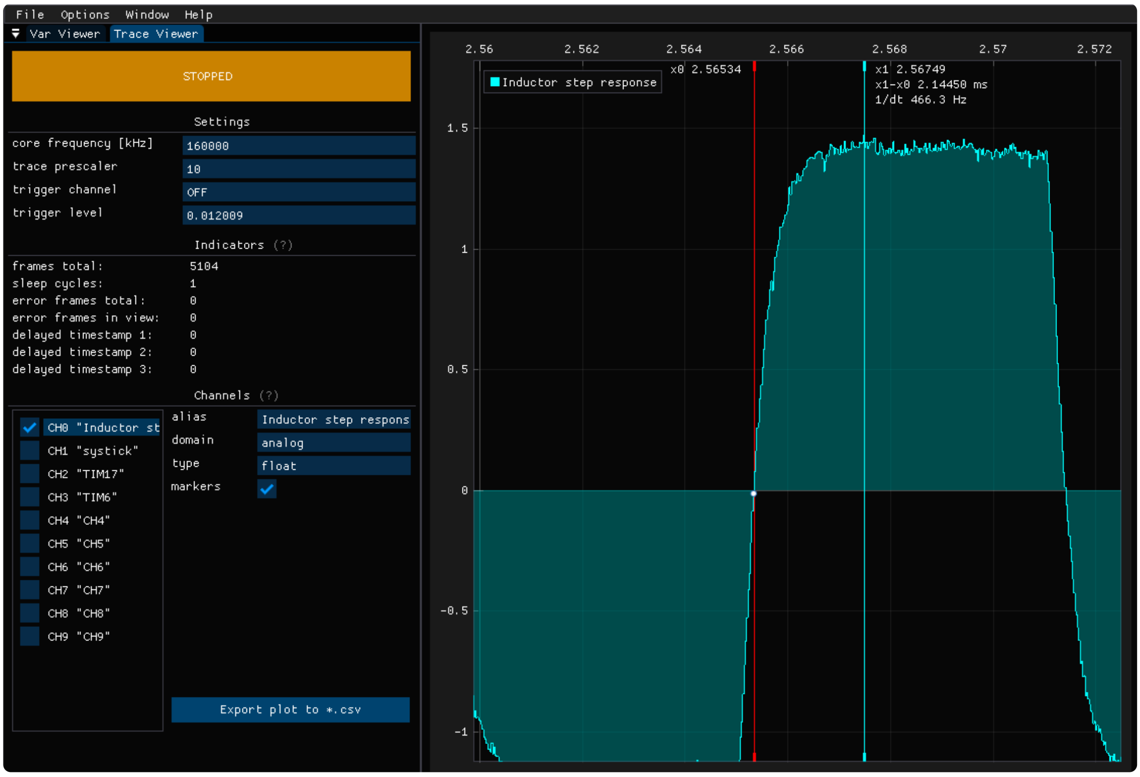
Results and Future Improvements

MCUViewer has evolved into a robust, open-source solution designed to make embedded systems debugging more efficient and accessible. Total downloads over all released versions have reached over 2500, and it now offers features that cater to a variety of sensor- and actuator-related applications, such as debugging power electronics, robotics and embedded control systems in general. My hope is that MCUViewer can empower developers across a range of fields, providing a reliable and adaptable tool for solving their most pressing challenges.

MCUViewer is under constant development. I am currently working on some new features that will make the debugging process even better. One of the most important of these is plot groups, which will allow users to switch quickly between plot collections.

Other important features to come are:

- XY plot
- fixed point interpretation of variables
- the ability to read arbitrary memory addresses
- the ability to perform basic preprocessing on collected variables



Settings

core frequency [kHz] 160000
trace prescaler 10
trigger channel OFF
trigger level 0.012009

Indicators (?)

frames total: 5104
sleep cycles: 1
error frames total: 0
error frames in view: 0
delayed timestamp 1: 0
delayed timestamp 2: 0
delayed timestamp 3: 0

Channels (?)

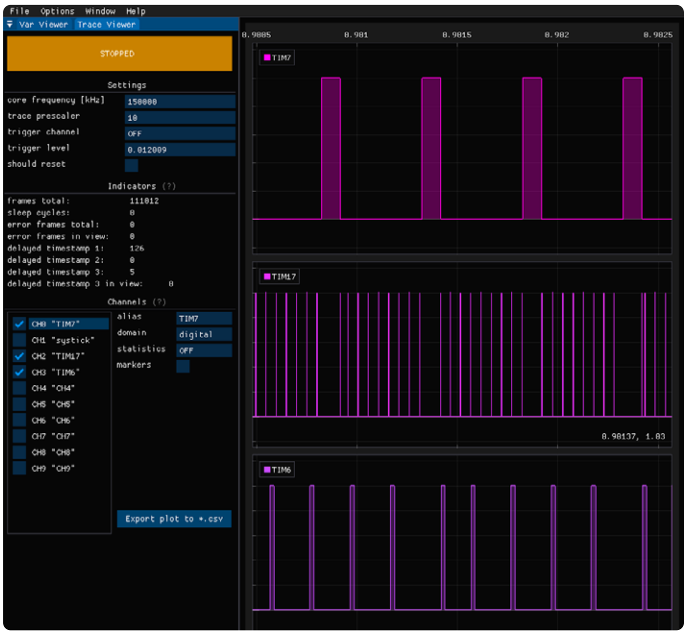
☑ CH0 "Inductor st
□ CH1 "systick"
□ CH2 "TIM17"
□ CH3 "TIM6"
□ CH4 "CH4"
□ CH5 "CH5"
□ CH6 "CH6"
□ CH7 "CH7"
□ CH8 "CH8"
□ CH9 "CH9"

alias Inductor step respons
domain analog
type float
markers ☑

x0 2.56534

x1 2.56749
x1-x0 2.14450 ms
1/dt 466.3 Hz

Figure 4: Trace Viewer can also visualize analog signals; here measuring the response of an inductor current for a given voltage command.



Settings

core frequency [kHz] 150000
trace prescaler 10
trigger channel OFF
trigger level 0.012009
should reset □

Indicators (?)

frames total: 111012
sleep cycles: 0
error frames total: 0
error frames in view: 0
delayed timestamp 1: 126
delayed timestamp 2: 0
delayed timestamp 3: 5
delayed timestamp 3 in view: 0

Channels (?)

☑ CH0 "TIM7"
□ CH1 "systick"
☑ CH2 "TIM17"
☑ CH3 "TIM6"
□ CH4 "CH4"
□ CH5 "CH5"
□ CH6 "CH6"
□ CH7 "CH7"
□ CH8 "CH8"
□ CH9 "CH9"

alias TIM7
domain digital
statistics OFF
markers □

Figure 5: Trace Viewer visualizes interrupt execution order, timings, durations, and preemption events by using enter and exit markers for detailed analysis.

If you would like to try out MCUViewer in your future projects, the project is hosted on GitHub [2], and contributions from the community are always welcome. If you have ideas for improvements, need to report a bug, or are interested in contributing to a specific feature, feel free to start a new issue or contact me directly. Even small contributions can make a big difference in improving the tool and expanding its capabilities. Engaging with the community has been one of the most rewarding parts of developing MCUViewer, and I look forward to collaborating with others who share an interest in embedded systems development. ◀

230602-01

About the Author

Piotrek Wasilewski is an embedded systems enthusiast with a deep passion for software development in power electronics and brushless motor controllers. His experience includes working on projects involving power electronics, quadruped and sumo robots, and the development of a milling machine. Through these varied ventures, he has developed a strong background in designing and implementing advanced embedded solutions.

Questions or Comments?

Do you have questions or comments about this article? Email the author at piwasilewski@interia.pl or contact Elektor at editor@elektor.com.

WEB LINKS

[1] GPLv3 Public License:
https://www.gnu.org/licenses/gpl-3.0.en.html

[2] MCUViewer project on GitHub:
https://github.com/klonyyy/MCUViewer?tab=readme-ov-file

Getting Started

Installation

First, you'll need to download MCUViewer installer on GitHub [2].

Linux

Ensure you have GDB version 12.1 or higher installed. Download the _*.deb_ package and install it with `sudo apt install ./MCUViewer-x.y.z-Linux.deb` All dependencies will be installed automatically.

For ST-LINK Users:
If your ST-LINK is not detected, copy the files from the _/launch/udevrules/_ folder to _/etc/udev/rules.d/_.

Windows

Download and run the MCUViewer installer from the _Releases_ section found in the right-hand menu of the repository's main page.

For ST-LINK users:
Ensure that ST-LINK is in _STM32 Debug + Mass Storage + VCP_ mode, as _STM32 Debug + VCP_ may cause libusb errors.

To improve performance, assign the external GPU to MCUViewer.

Quick Start

Variable Viewer

1. Setup variables:
- Open _Options → Acquisition Settings_ in the top menu.
- Select your project's ELF file (compiled in debug mode) and click _Done_.
- Click Import Variables form _*.elf_ to select variables, or manually add variables if needed.
- Report issues with undetected variables by attaching the ELF file.
- Click Update variable addresses and ensure types/addresses are valid.

2. Visualize variables:
- Drag and drop variables to the plot area.
- Connect the debug probe (ST-LINK/J-Link), download the executable, and press the _STOPPED_ button.

3. Troubleshooting:
- Try the _example/MCUViewer_test CubeIDE_ project with the corresponding _MCUViewer_test.cfg_ project file. Remember to build the project and update the ELF file path in _Options → Acquisition_ settings.

Trace Viewer

1. Enable SWO Pin by configuring _CubeMX: System Core → SYS Mode_ and in _Configuration_ choose the _Trace Asynchronous Sw_ option.

2. Add enter and exit markers:
For example, you can profile digital data as follows:

```
ITM->PORT[x].u8 = 0xaa;
//enter tag 0xaa - plot state high
foo();
ITM->PORT[x].u8 = 0xbb;
//exit tag 0xbb - plot state low
```

For profiling analog signals:

```
float a = sin(10.0f * i);
// some high frequency signal to trace
ITM->PORT[x].u32 = *(uint32_t*)&a;
// type-pun to desired size: sizeof(float) =
// sizeof(uint32_t)
```

Or

```
uint16_t a = getAdcSample();
// some high frequency signal to trace
ITM->PORT[x].u16 = a;
```

3. Run the program:
Compile and download to STM32 and set the correct _System Core Clock_ value in _Settings_ in kHz — this is very important as it affects the timebase.

4. Set prescaler:
Adjust to stay within the max trace speed of your programmer (e.g., ST-Link V2: 2 MHz, V3: 24 MHz). Example: for 160 MHz clock, ST-Link V2 prescaler should be ≥ 80.

5. Start recording:
Configure analog channel types and press the _STOPPED_ button to begin.

USB 2.0 Isolator

Electrically Isolated Connections for USB Devices

By Alfred Rosenkränzer (Germany)

In most cases, connecting peripherals to a PC is not a problem. The fact that a USB stick, for example, has the same ground potential as a laptop is hardly a problem. But when it comes to measuring devices connected via USB, the situation is different. Here, electrical isolation protects the PC from potentially harmful voltages. Isolation also helps to avoid ground loops. Specific chips are available for this purpose. The circuit presented here uses such a chip to provide electrical isolation for a USB connection.

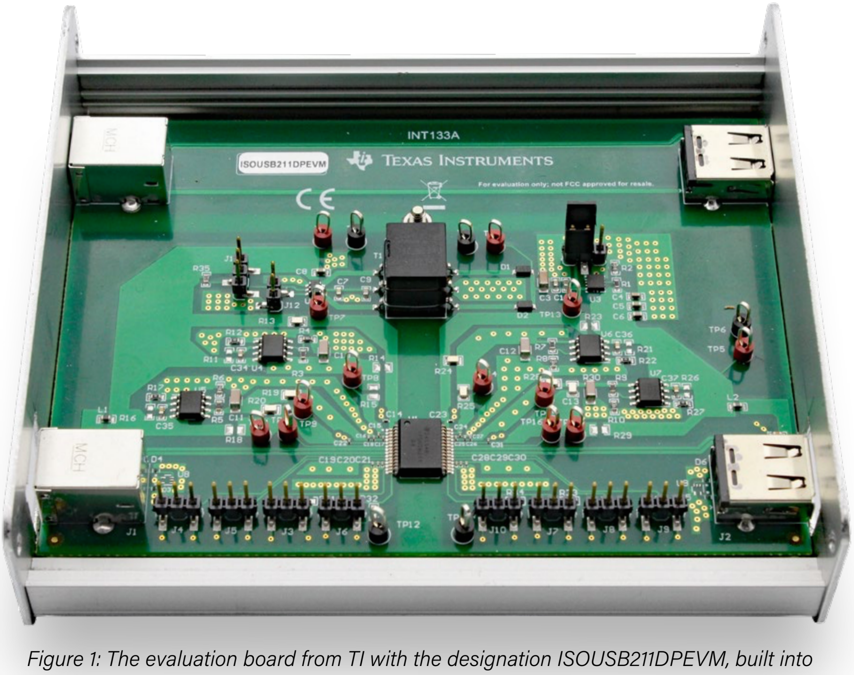


Figure 1: The evaluation board from TI with the designation ISOUSB211DPEVM, built into a metal casing.

Unlike in the past with GPIB, these days the individual devices in automatic test setups are usually controlled via USB. Since ground loops have always been a problem, GPIB had special devices for electrically isolating the control unit (usually a PC) from the measuring instruments and generators connected to it. Unfortunately, the transition to USB has not changed this challenge. This is because there is also a ground connection between the devices, and it is not uncommon to have interference from the PC affecting the peripherals.

Isolator Chips

A USB isolator can be used to prevent a ground connection and only transmit the digital signals. For the slow USB 1.1 standard, there are, for example, ICs from Analog Devices and boards equipped with them that can be purchased for a low price on eBay. However, reasonable data transfer rates today require at least a USB 2.0 connection. Devices for this speed are also available on the market. Their main disadvantage, however, is that they are quite expensive. Instead of just €10 to €20 for USB 1.1, you often have to pay three-digit sums for USB 2.0. In this case, building your own device can save you a lot of money.

Suitable chips for USB 2.0 have been developed by Analog Devices (ADUM4165 and ADUM4166) and TI (ISOUSB211). These can be used to implement low-cost isolators, as these ICs are available for less than €10. Since I found these ICs interesting, I first bought the evaluation board for the ADUM4166 and successfully tested this chip with a memory stick and a USB hard disk. Unfortunately, controlling an older Audio Precision 2422 audio analyzer with an external USB interface did not work at all. With the QA403 analyzer from QuantAsylum, the control only worked sporadically. Analog Devices' support confirmed the problem and also pointed out that there seem to be problems with other devices as well. Tests with an ADUM4165 evaluation board yielded the same result. Unfortunately, ADI was unable to provide any information on the cause of this problematic behavior.

Fortunately, an ISOUSB211 evaluation board from TI (**Figure 1**) was already available after

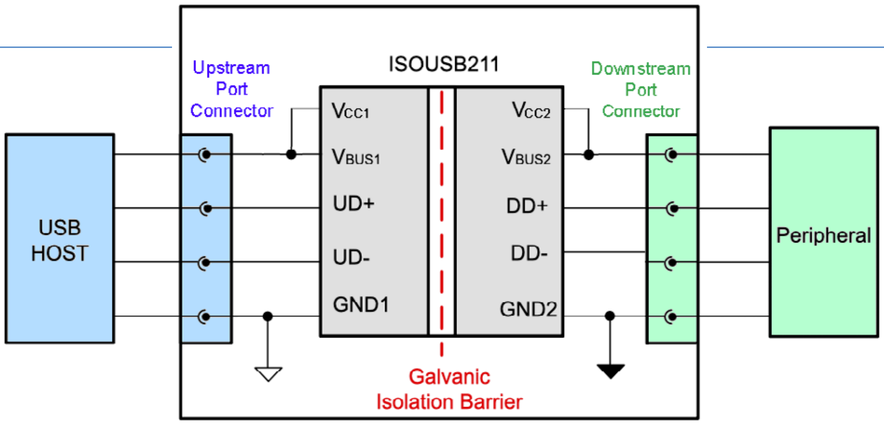


Figure 2: Block diagram of the ISOUSB211 isolator IC.

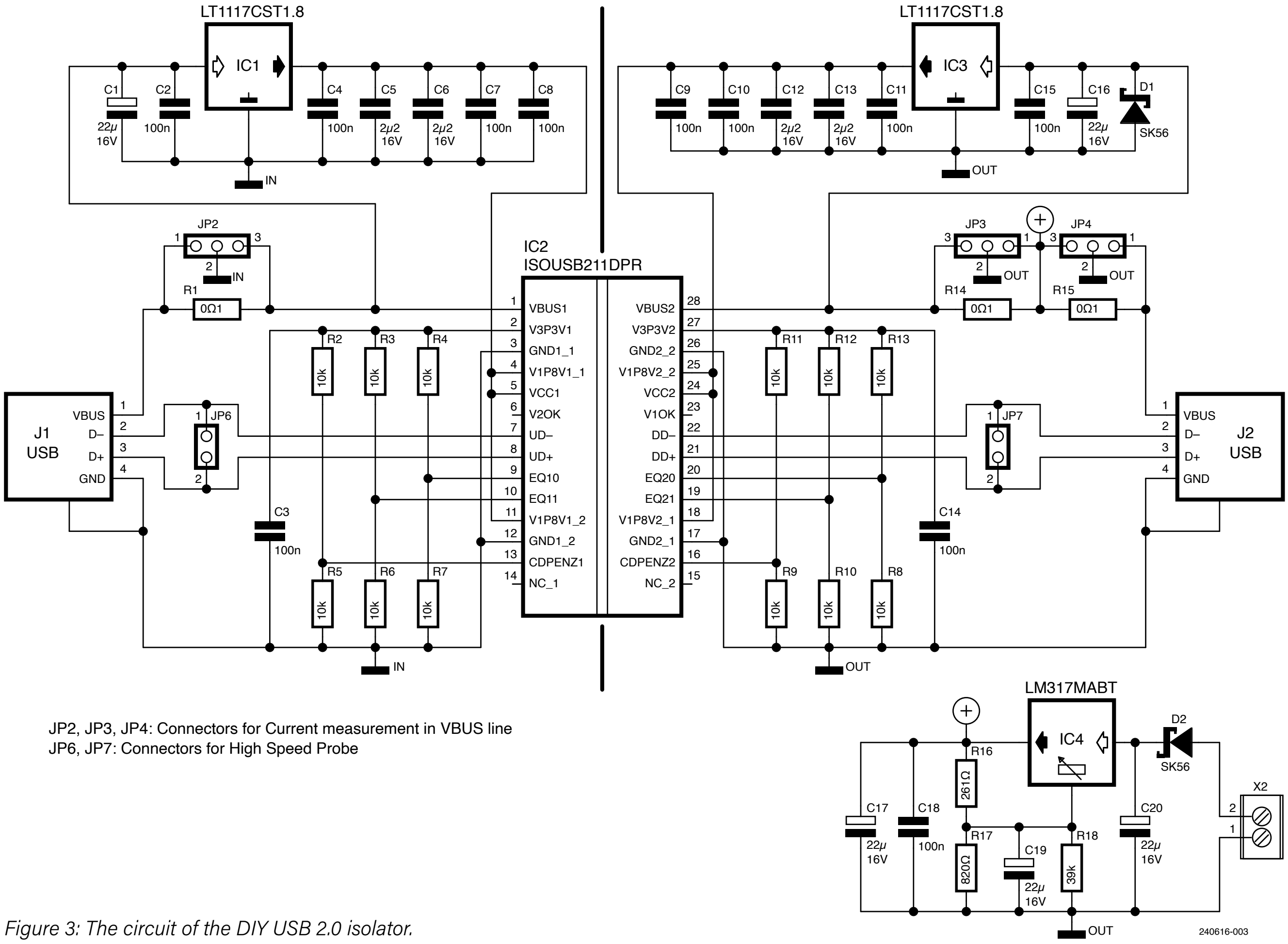
these sobering first attempts. And hooray: It works with all the devices I was able to try out. In addition to the USB connection, it has a DC/DC converter with electrical isolation on board, which generates the 5 V sometimes required for the device side from the 5 V on the input side. If the power provided is insufficient or the switching converter causes interference, an external supply can also be used.

The board offers a wide range of settings and has a fairly sophisticated power supply. The TI website [1] provides a data sheet and technical specifications. **Figure 2** shows the block diagram of the ISOUSB211.

The Circuit

After the first encouraging tests with the ISOUSB211, I decided to design my own circuit board for this IC. It should only contain the most essential components; in most cases, a DC/DC converter can be dispensed with. **Figure 3** shows the resulting circuit and **Figure 4** the layout. The layout files in Eagle format can be downloaded for free from the Elektor website for this article [2].

IC2 in the center separates the left side, where the host or PC is connected, from the right side, where the peripheral is connected. Internally, the integrated circuit has not only 3.3 V regulators on both sides that supply the logic, but also integrated 1.8 V regulators to supply other logic parts. However, since the use of the integrated 1.8 V regulators significantly heats up IC2, the use of external 1.8 V regulators (IC1 and IC3) is recommended instead. In addition, a 5 V regulator is provided around IC4 on the right, which can also be used to



JP2, JP3, JP4: Connectors for Current measurement in VBUS line
JP6, JP7: Connectors for High Speed Probe

Figure 3: The circuit of the DIY USB 2.0 isolator.

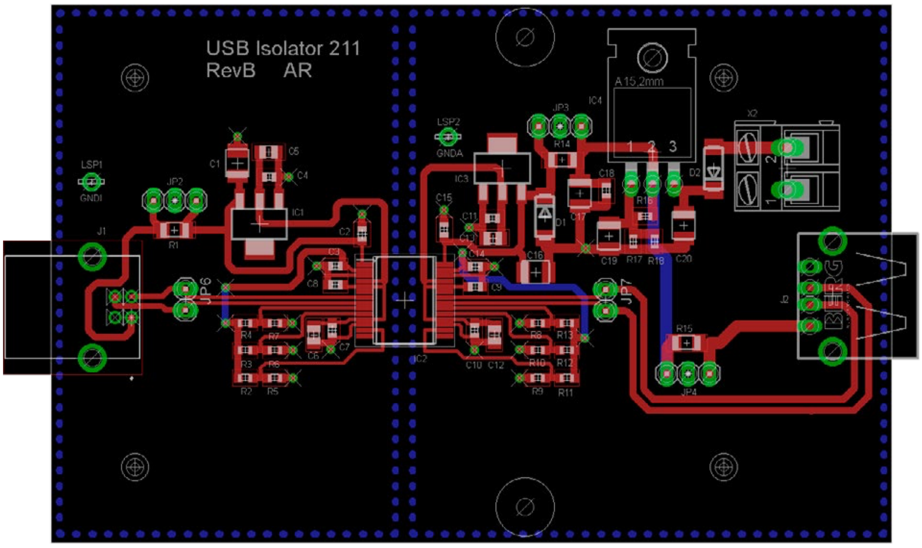


Figure 4: The PCB layout for the circuit.

supply the circuit from an external DC power supply with 8...15 V. IC4 is powerful enough to supply USB 2.0 devices connected to J2 with up to 400 mA.

There are six resistors on each side (R2... R7 and R8...R13) that can optionally be used to configure certain functions. The levels at the inputs EQ10, EQ11, and CDPENZ1 as well as EQ20, EQ21, and CDPENZ2 determine the configuration of an equalizer (the EQ pins) and the use as a charging port (CDP = Charging Downstream Port, active low). The equalizer is a switchable compensation of the electrical properties of the conductor tracks for signals, which of course depend on the specific PCB layout and can be quite relevant at the data rates occurring with USB 2.0. Since the EQXX pins can process three levels (high, open, and low), $3^2 = 9$ different compensations can be set per side. The details are described in the data sheet [3]. In general, the circuit works well with open inputs, which is why I did not fit R3, R4, R6, and R7, nor R8, R10, R12, and R13. The CDP function is not needed for the purposes described here, so it can be disabled by fitting R2 and R11. All settings are queried each time the IC is switched on and stored internally until the next time it is switched off.

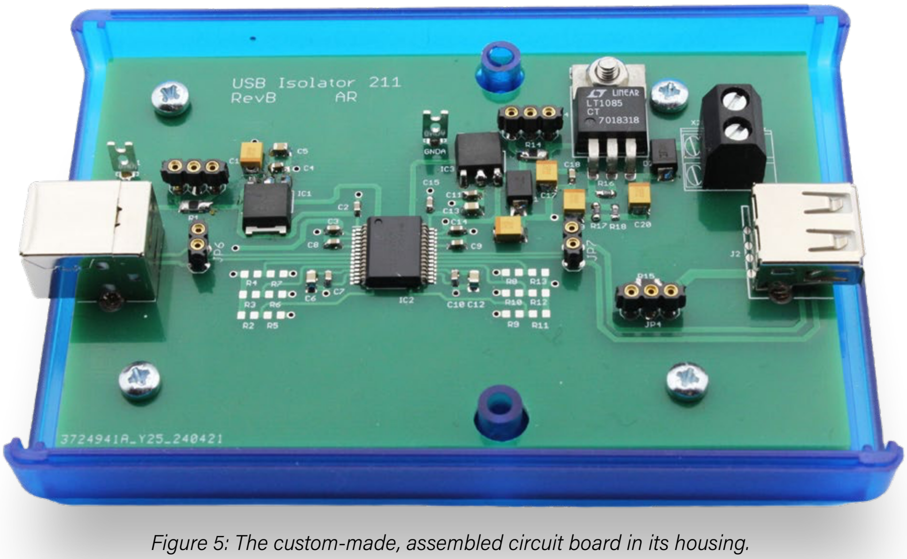


Figure 5: The custom-made, assembled circuit board in its housing.

Component List

Resistors:

Unless otherwise specified:
Thin film SMD 0603, 1%
R1, R14, R15 = 0Ω1, 1206
R2...R13 = 10 kΩ *
R16 = 261 Ω
R17 = 820 Ω
R18 = 39 kΩ

Capacitors:

C1, C16, C17, C19, C20 = 22 µF / 16 V, electrolytic, tantalum, SMC-B
C2...C4, C7...C11, C14, C15, C18 = 100 nF, X7R, SMD 0603
C5, C6, C12, C13 = 2.2 µF / 16 V, X7R, SMD 0805

Semiconductors:

D1, D2 = SK56, Schottky diode, DO214AA
IC1, IC3 = LT1117CST-1.8, SOT223
IC2 = ISOUSB211DPR, DP0028A
IC4 = LM317MABT, TO220

Miscellaneous:

JP2...JP4 = 3-pin header, 1/10″ pitch
JP6, JP7 = 2-pin header, 1/10″ pitch
X2 = 2-pin screw terminal, 5 mm pitch, AK300/2,
J1 = USB socket for PCB mounting, type B
J2 = USB socket for PCB mounting, type A
Enclosure Hammond 1593NTBU
Circuit board 240616-01

* see text

Measurement Options

Since I was interested in the currents flowing in the circuit, especially when something was not working properly, I added 0.1-Ω resistors R1, R14, and R15. The voltage drop can be conveniently measured at JP2, JP3, and JP4.

The quality of the signals can be assessed at JP6 and JP7 using a differential high-speed oscilloscope probe, for example. The circuit is largely symmetrical. On the right side, there is an additional voltage regulator configured for a voltage of 5 V, which can be supplied externally via X2 and is sufficient to provide enough power to supply a USB device connected to J2.

Miscellaneous

Figure 5 shows the assembled board in the plastic casing for which it was designed. My homemade USB isolator works well, and I am satisfied. Barely had the board been installed in the casing and tested when support from Analog Devices contacted me with a workaround: It was suggested that an ordinary USB hub be connected between the PC and the ADUM4165 evaluation board. I tried this suggestion, and it actually worked: The Audio Precision analyzer can now be controlled. The same trick also works for the ADUM4166. However, the cause of the problem (and why the USB hub fixes it) remains a mystery.

By the way: I still have a few empty circuit boards left over for the ISOUSB211 solution. ◀

Translated by Jörg Starkmuth — 240616-01

About the Author

Alfred Rosenkränzer worked for many years as a development engineer, initially in the field of professional television technology. Since the late 1990s, he has been developing digital high-speed and analog circuits for IC testers. Audio is his private passion.

Questions or Comments?

Do you have questions or comments about this article? Email the author at alfred_rosenkraenzer@gmx.de, or contact Elektor at editor@elektor.com.

Related Products

- **LabNation SmartScope USB Oscilloscope**
  www.elektor.com/17169
- **Aoyue Int 866 (3-in-1) SMD Hot Air Rework Station**
  www.elektor.com/20783

WEB LINKS

[1] ISOUSB211DPEVM evaluation board: https://www.ti.com/tool/ISOUSB211DPEVM
[2] Elektor web page for this article: https://www.elektormagazine.com/240616-01
[3] ISOUSB211 data sheet from TI: https://www.ti.com/lit/ds/symlink/isousb211.pdf

Intervention Before Damage

Predictive Maintenance in Practice

By Tam Hanna (Hungary)

One of the most valuable applications of AI is certainly predictive maintenance – the forward-looking servicing of machines. The neural analysis of data streams, usually from multiple sensors, makes it possible to detect critical system states at an early stage and to intervene before any damage actually occurs. In this article, we will provide not only background information but also a practical application of the method. We will use an inexpensive development board and example software from NXP, as well as an acceleration sensor, to detect anomalies in the operation of a fan.

With the right parameterization, self-learning algorithms enable predictive maintenance systems to detect almost any measurable problem state. In principle, the systems are represented as in the flowchart shown in **Figure 1**.

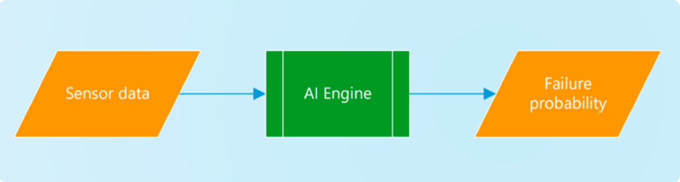


Figure 1: Where there are input and output parameters, predictive maintenance or anomaly detection is possible.

The term "anomaly detection" is based on a highly important theory: The AI learning process shown in the middle of the figure usually has no direct knowledge of what is going on inside the machine. Instead, the system examines measurement data from various sensors and compares it with training data in order to detect deviations and report them.

Controller Zoo

The MCX family, which was introduced at the beginning of 2024, is NXP's path to the future. **Figure 2** shows the options that developers will be faced with in this now very extensive microcontroller portfolio. The MCX N series, designed primarily for the needs of AI applications, is presented as shown in **Figure 3**. It should be noted here that the smallest variant, 2X, does not come with a Neural Processing Unit (NPU) — this is only available from the MCX 5X upwards.

For the sake of convenience, we will use the high-end version MCX N9X in the following steps (see box **Start with Large Controller Types**). The controller is available at a relatively low price in the form of the FRDM-MCXN947 evaluation board (see also the **Related Products** box).

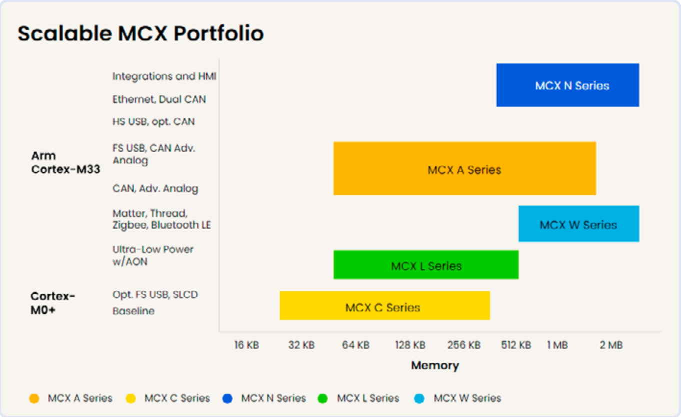
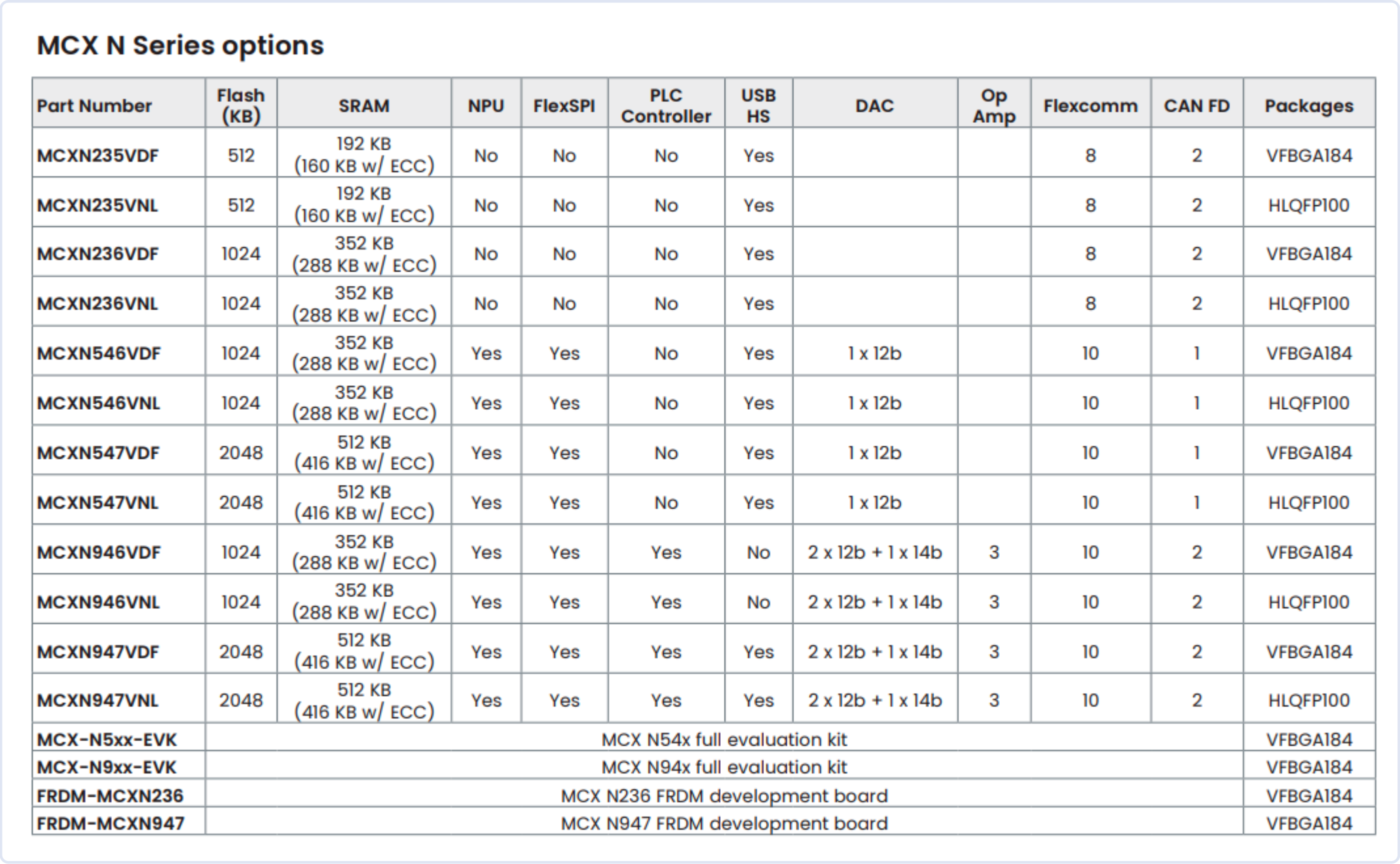


Figure 2: NXP's portfolio is teeming with choice (Source: [9]).



MCX N Series options

Part Number	Flash (KB)	SRAM	NPU	FlexSPI	PLC Controller	USB HS	DAC	Op Amp	Flexcomm	CAN FD	Packages
MCXN235VDF	512	192 KB (160 KB w/ ECC)	No	No	No	Yes			8	2	VFBGA184
MCXN235VNL	512	192 KB (160 KB w/ ECC)	No	No	No	Yes			8	2	HLQFP100
MCXN236VDF	1024	352 KB (288 KB w/ ECC)	No	No	No	Yes			8	2	VFBGA184
MCXN236VNL	1024	352 KB (288 KB w/ ECC)	No	No	No	Yes			8	2	HLQFP100
MCXN546VDF	1024	352 KB (288 KB w/ ECC)	Yes	Yes	No	Yes	1 x 12b		10	1	VFBGA184
MCXN546VNL	1024	352 KB (288 KB w/ ECC)	Yes	Yes	No	Yes	1 x 12b		10	1	HLQFP100
MCXN547VDF	2048	512 KB (416 KB w/ ECC)	Yes	Yes	No	Yes	1 x 12b		10	1	VFBGA184
MCXN547VNL	2048	512 KB (416 KB w/ ECC)	Yes	Yes	No	Yes	1 x 12b		10	1	HLQFP100
MCXN946VDF	1024	352 KB (288 KB w/ ECC)	Yes	Yes	Yes	No	2 x 12b + 1 x 14b	3	10	2	VFBGA184
MCXN946VNL	1024	352 KB (288 KB w/ ECC)	Yes	Yes	Yes	No	2 x 12b + 1 x 14b	3	10	2	HLQFP100
MCXN947VDF	2048	512 KB (416 KB w/ ECC)	Yes	Yes	Yes	Yes	2 x 12b + 1 x 14b	3	10	2	VFBGA184
MCXN947VNL	2048	512 KB (416 KB w/ ECC)	Yes	Yes	Yes	Yes	2 x 12b + 1 x 14b	3	10	2	HLQFP100
MCX-N5xx-EVK	MCX N54x full evaluation kit										VFBGA184
MCX-N9xx-EVK	MCX N94x full evaluation kit										VFBGA184
FRDM-MCXN236	MCX N236 FRDM development board										VFBGA184
FRDM-MCXN947	MCX N947 FRDM development board										VFBGA184

Figure 3: The AI accelerator is not available in all variants (Source: [10]).

Start with Large Controller Types

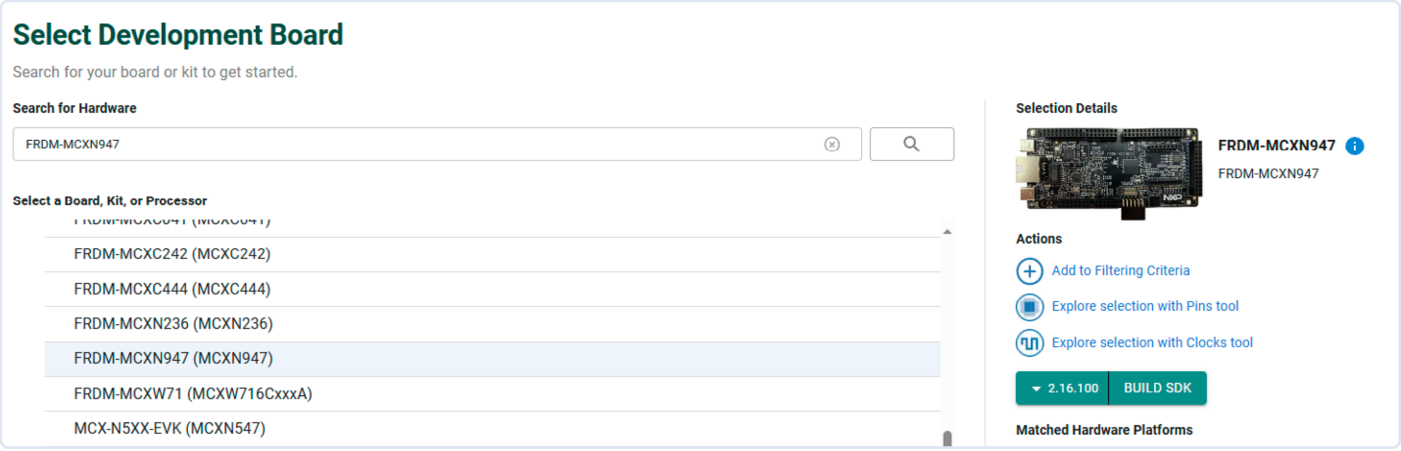
Particularly for small series, the price differences between the various memory configurations are marginal. The author therefore likes to start with the largest possible variant of a chip in his consulting business — if the system works, you can scale down later. However, it should be noted that a small reserve of resources can be helpful — you never know what additional functions the customer might want in the field.

A USB-C cable is required to communicate with the computer. In addition, of course, various (sensor) hardware is needed to enable the ingestion of the required information as shown in Figure 1.

The following practical example requires an accelerometer (in addition to an NXP-specific TFT display for showing the calculated results). This sensor is connected to the fan we want to monitor, in a way that couples as much vibration as possible. The accelerometer and evaluation board are connected via I²C, while the display is connected to the designated port.

Software Setup

Similar to the compulsory registration for Cube introduced by STMicroelectronics some time ago, NXP also only allows people with an NXP account to download the documentation and toolchain. The download process (probably inspired by Android) is also interesting: The SDK and the integrated development environment have to be obtained in two separate transactions.



Select Development Board

Search for your board or kit to get started.

Search for Hardware

FRDM-MCXN947

Select a Board, Kit, or Processor

- FRDM-MCXC242 (MCXC242)
- FRDM-MCXC444 (MCXC444)
- FRDM-MCXN236 (MCXN236)
- FRDM-MCXN947 (MCXN947)
- FRDM-MCXW71 (MCXW716CxxxA)
- MCX-N5XX-EVK (MCXN547)

Selection Details

FRDM-MCXN947

FRDM-MCXN947

Actions

Add to Filtering Criteria
Explore selection with Pins tool
Explore selection with Clocks tool
2.16.100
BUILD SDK

Matched Hardware Platforms

Figure 4: The board assistant asks you to select the appropriate evaluation kit.

The first step is to visit the SDK downloader available at [1], where we select the option *Select Development Board*. After clicking the button, the website prompts us to log in. The next step is to use the SDK search and download assistant, something we are familiar with from other providers.

The first thing to do is to click on the text field *Search for hardware*, where the string *FRDM-MCXN947* must be entered. The Boards section then shows the board, which can be selected as shown in **Figure 4**. Of particular note is the turquoise selection button on the right, which allows the compilation and assembly of a specific SDK version. Since we want to experiment with a skeleton project provided by NXP in the following steps, the correct version is *2.14.0*. Then click the *Build SDK* button to start the assembly wizard.

In the next step, the wizard asks for various settings. In addition to the *Host OS* (Windows 10 will be used in the following steps), the toolchain must be selected. Here we want to opt for the *MCUXpresso IDE*.

The list that appears below then allows us to select the additional components we want (see **Figure 5**). For convenience, it is recommended that you download everything that NXP provides for this microcontroller family by clicking *Select All*.

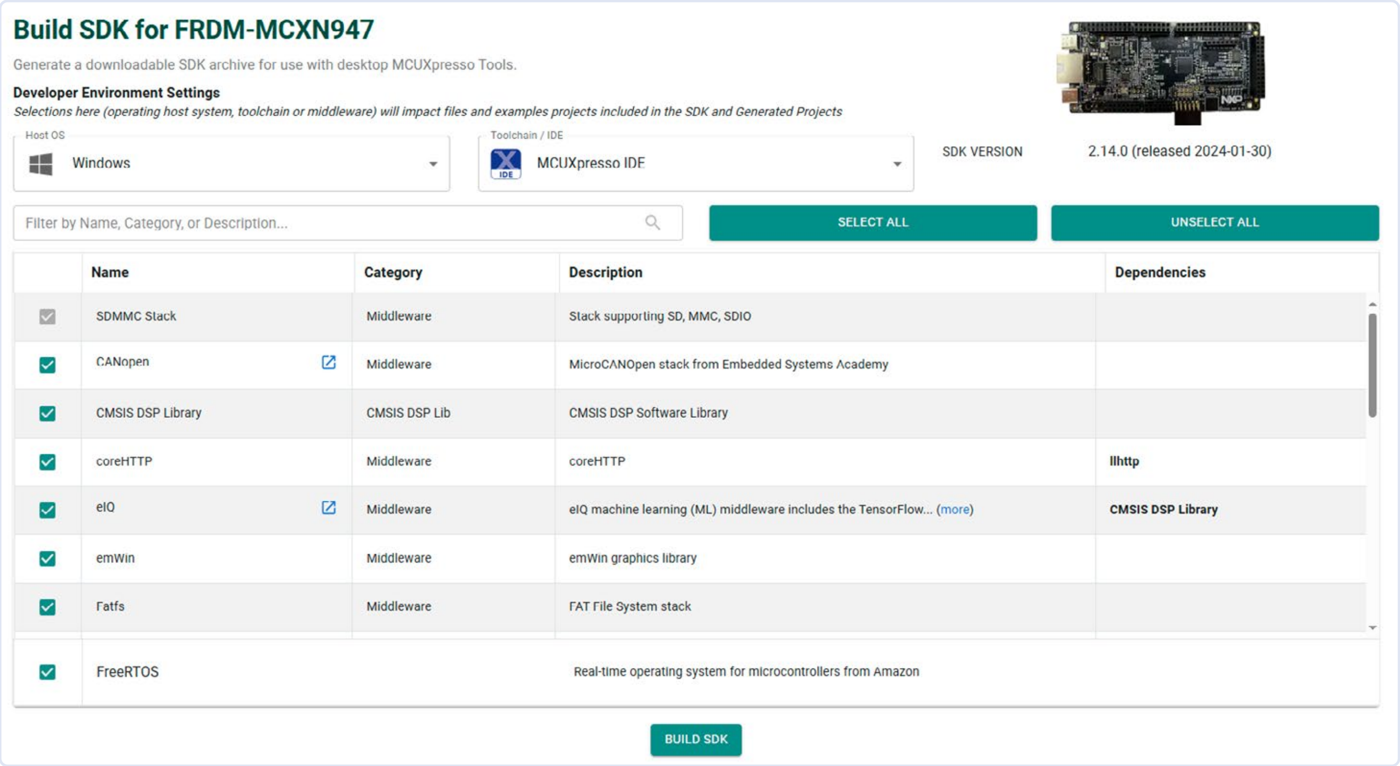
Once the work is done, click the *Build* button again. After a few seconds, the SDK will appear in the view labeled *MCUXpresso SDK Dashboard*, where it can be downloaded by clicking the *Download* option. At the time this article is written, this is a file of around 300 MB in size, which on the author's workstation is named *SDK_2_14_0_FRDM-MCXN947.zip*.

The next step is to actually download the IDE. Fortunately, there is a link at the bottom right of the *MCUXpresso SDK Dashboard* that we can click on in the next step: *Download MCUXpresso IDE*.

The reward for our efforts is the appearance of a new browser window, in which — analogies to the download process of STMs CubeIDE are recognizable — we click again on the yellow *Download* button in the first step. The result is that the browser has left us further down the download web page. The link that is relevant for us is *MCUXpresso Integrated Development Environment (IDE) updated*.

Occasionally, yellow error messages may appear on the website, indicating that the license service is temporarily offline. This is because NXP manages the distribution of MCUXpresso using the Flexnet license system, which is familiar from various games and other software. In this case, it is usually sufficient to wait a few minutes for the confirmation email to arrive. After that, refresh the page to get to the list of all licensed program packages.

Once there, a click on the *MCUXpresso IDE* link is required — at the time this article is written, the product is available in version 24.9. After accepting the licenses, NXP offers binary files for Windows, MacOS and Linux, all of which are around 1.1 GB in size. In the following steps, the author will work with the file *MCUXpressoIDE_24.9.25.exe*.



Build SDK for FRDM-MCXN947

Generate a downloadable SDK archive for use with desktop MCUXpresso Tools.

Developer Environment Settings

Selections here (operating host system, toolchain or middleware) will impact files and examples projects included in the SDK and Generated Projects

Host OS: Windows

Toolchain / IDE: MCUXpresso IDE

SDK VERSION: 2.14.0 (released 2024-01-30)

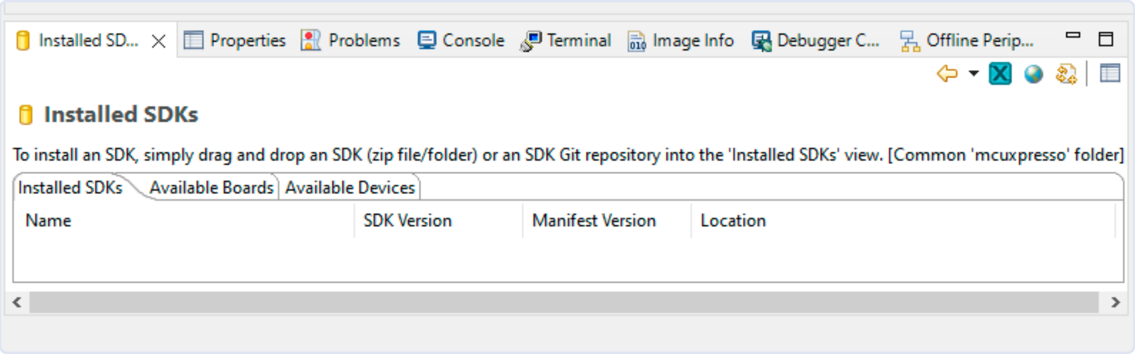
Filter by Name, Category, or Description...

SELECT ALL | UNSELECT ALL

	Name	Category	Description	Dependencies
☑	SDMMC Stack	Middleware	Stack supporting SD, MMC, SDIO	
☑	CANopen	Middleware	MicroCANOpen stack from Embedded Systems Academy	
☑	CMSIS DSP Library	CMSIS DSP Lib	CMSIS DSP Software Library	
☑	coreHTTP	Middleware	coreHTTP	llhttp
☑	eIQ	Middleware	eIQ machine learning (ML) middleware includes the TensorFlow... (more)	CMSIS DSP Library
☑	emWin	Middleware	emWin graphics library	
☑	Fatfs	Middleware	FAT File System stack	
☑	FreeRTOS		Real-time operating system for microcontrollers from Amazon	

BUILD SDK

Figure 5: NXP allows modular configuration of the SDK.



Installed SDKs

To install an SDK, simply drag and drop an SDK (zip file/folder) or an SDK Git repository into the 'Installed SDKs' view. [Common 'mcuxpresso' folder]

Name	SDK Version	Manifest Version	Location

Figure 6:
Lateral entrants could be in for a surprise here.

During the installation, which generally goes as expected, Windows sometimes displays several UAC warnings. These have to be acknowledged because developers in the NXP ecosystem have to deal with a wide variety of programming devices.

When using the default settings, the MCUXpresso installer creates a desktop icon that can be used to launch the integrated development environment "directly". Since the IDE is based on Eclipse in the background, the tool asks for a path to a workspace during the first start — this can be set more or less as you like, the default settings worked without problems in the test, at least on the author's workstation. It is also important to allow all access — in the firewall warning, you must explicitly allow the integrated development tool to interact with local networks.

For developers unfamiliar with MCUXpresso, a small pitfall lurks in the welcome screen: the window responsible for accepting the SDK downloaded in the previous step cannot be activated via the link for installing SDKs. Instead, you have to close the welcome screen to switch to the actual MCUXpresso working environment. There you will find a window, as shown in **Figure 6**, in which the SDKs can be managed.

In the next step, the .zip file is dragged and dropped into this window. You can safely ignore the warning message displayed by the IDE. A few seconds later, the new toolchain is ready for action as part of the MCUXpresso IDE.

Examining the (Complex) Example Code

Samsung's now defunct Bada operating system was popular with developers in part because it put handheld software developers in the role of "plumbers" to a certain extent. Developing an application for this operating system generally involves connecting the various building blocks provided by Samsung. In the opinion of the author, this development method, which at the time was revolutionary and ingenious, is also recommended when working with artificial intelligence systems.

Since most AI tasks are structured according to one of just a relatively small number of schemes, developers should start by looking for an example that can be run, and then adapt this to the needs of the given task.

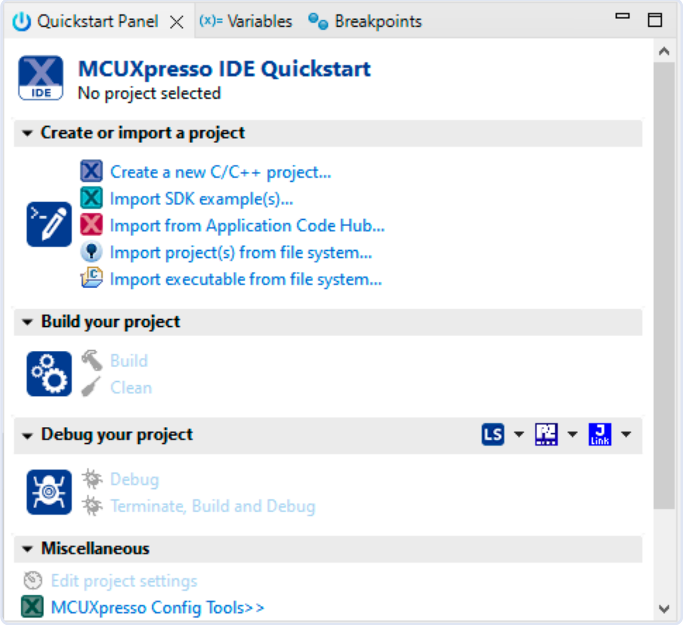
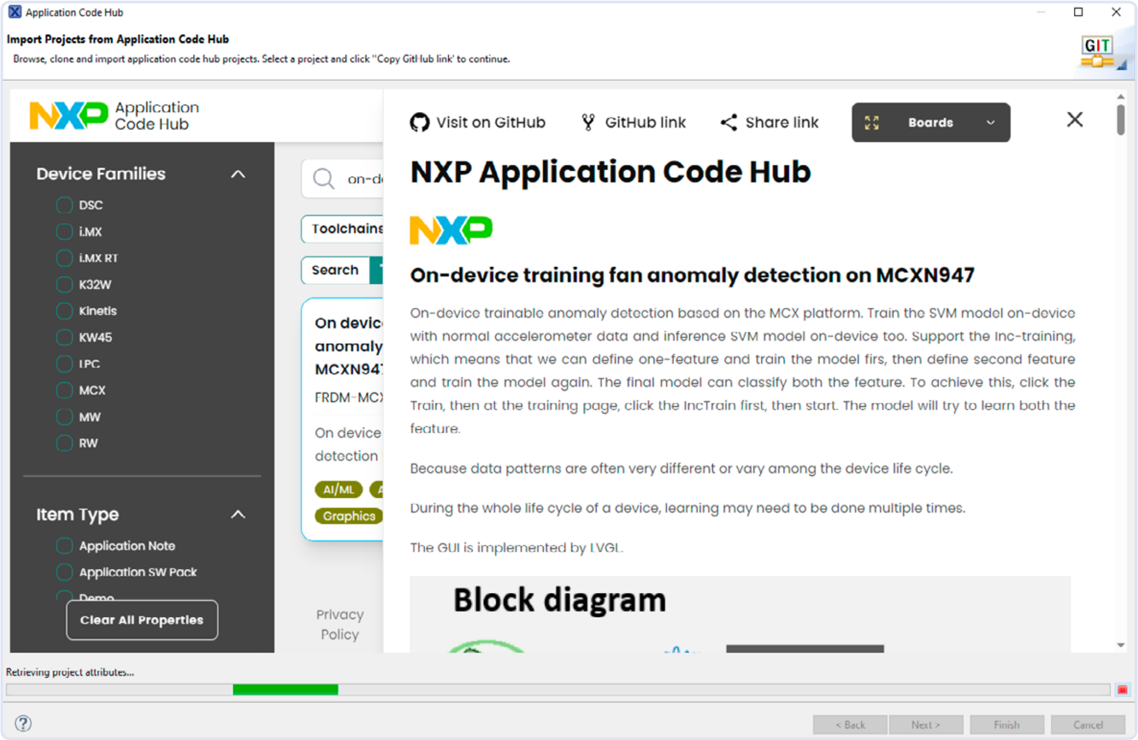

Quickstart Panel ✕
(x)= Variables
Breakpoints
IDE
MCUXpresso IDE Quickstart
No project selected
Create or import a project
Create a new C/C++ project...
Import SDK example(s)...
Import from Application Code Hub...
Import project(s) from file system...
Import executable from file system...
Build your project
Build
Clean
Debug your project
LS
Debug
Terminate, Build and Debug
Miscellaneous
Edit project settings
MCUXpresso Config Tools>>

Figure 7: NXP spreads the useful functions all over the system.

In the case of anomaly detection, NXP provides an example project on GitHub at [2] that we will use in the following steps. Note that the IDE allows direct rehydration of source code from GitHub — it is not necessary to use the command line client to load the project skeleton into the workstation file system in the first step. Specifically, the function *Import from Application Code Hub* is available for this purpose, which is hidden in the *Quickstart* panel shown in **Figure 7**.

The search string *on-device training fan anomaly on mcxn947* then leads to the display of a tab with the desired example, which you click on with the left mouse button in the next step. MCUXpresso responds by displaying a rotating progress bar, which you have to confirm after a while.

The appearance of a tab with the content shown in **Figure 8**, which appears on the right of the screen, then allows you to click on the GitHub link symbol. In the next step, a *Next* option appears in the Eclipse IDE.



Import Projects from Application Code Hub

Browse, clone and import application code hub projects. Select a project and click "Copy GitHub link' to continue.

NXP Application Code Hub

On-device training fan anomaly detection on MCXN947

On-device trainable anomaly detection based on the MCX platform. Train the SVM model on-device with normal accelerometer data and inference SVM model on-device too. Support the Inc-training, which means that we can define one-feature and train the model firs, then define second feature and train the model again. The final model can classify both the feature. To achieve this, click the Train, then at the training page, click the IncTrain first, then start. The model will try to learn both the feature.

Because data patterns are often very different or vary among the device life cycle.

During the whole life cycle of a device, learning may need to be done multiple times.

The GUI is implemented by LVGL.

Block diagram

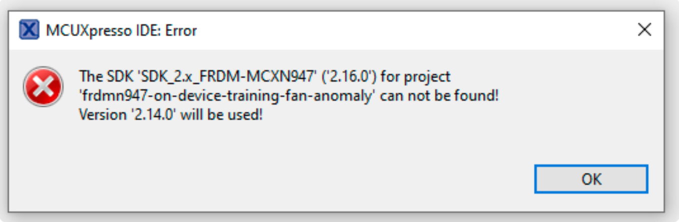
Retrieving project attributes...

Figure 8: Here you have to click on the GitHub link to get to the practical example.

The reward for your efforts is the display of a more or less standard GitHub deployment assistant. In tests by the author, it was possible to simply accept all the settings as proposed — due to the limited speed of GitHub, a little waiting time is sometimes required. If an error message appears as shown in **Figure 9**, you can update the SDK.

After successfully completing the loading process, we can start analyzing the code. The "entry point" of the example, which is based on FreeRTOS and LVGL, can be found in the file *source/main.c*, where the creation of the sensor and algorithm tasks is of particular importance:

```
int main(void) {
   . . .
  if (xTaskCreate(app_sensor_task, "SENS", 4096,
      NULL, configMAX_PRIORITIES-1, NULL) != pdPASS) {
    PRINTF("Failed to create sensor task.\r\n");
    while (1) {}
  }
  if (xTaskCreate(app_algo_task, "ALGO", 0x1000,
  NULL, configMAX_PRIORITIES-1, NULL) != pdPASS) {
    PRINTF("Failed to create sensor task.\r\n");
    while (1) {}
}
```



The SDK 'SDK_2.x_FRDM-MCXN947' ('2.16.0') for project 'frdmn947-on-device-training-fan-anomaly' can not be found! Version '2.14.0' will be used!

Figure 9: The SDK version mentioned in the documentation no longer matches the content of the Application Code Hub.

A queue will then work in the background, which — fed by the sensor task and harvested by the algorithm task — ensures the transfer of the incoming information from the accelerometer.

Brief Analysis of the Sensor Task

NXP supports two different sensor types here. In addition to the FXLS8974 from NXP itself, the MPU6050 from TDK InvenSense, which is well known for its low-cost availability on AliExpress and other sites, is also on the support list. In the sensor task, however, this selection does not have a significant effect. There is a kind of hardware abstraction layer that performs the following breakdown of the incoming accelerometer data and provides it for harvesting via a uniform interface:

```
int IMU_ReadSensorData(int16_t *pBuf,
uint16_t fifo_cnt, uint16_t readSize)
{
  if (sensor_id == IMU_6050)
    return MPU_ReadSensorData(pBuf, fifo_cnt, readSize);
  else if (sensor_id == IMU_FXLS8974)
    return IMU_FXLS8974_ReadSensorData(pBuf,
                fifo_cnt, readSize);
  else return 0;
}
```

The architecture of this example is interesting in that the raw data supplied by the accelerometer never actually reaches the ML model. Instead, the ML model is only presented with a frequency spectrum that the example code obtains by applying an FFT process to the raw data supplied by the accelerometer.

The way in which FFT is implemented is always an interesting topic — for example, in issue 3-4/2023, we reported on the K210 developed by Kendryte, which included an FFT accelerator unit and accordingly had "dedicated" FFT APIs. [3].

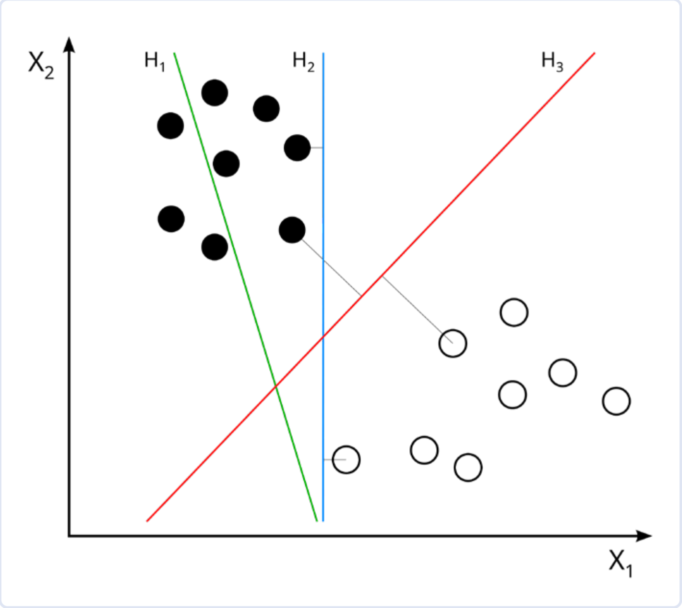


Figure 10: To understand how a support vector machine works, let's imagine that we only get two coefficients per measurement, which determine the coordinates in this two-dimensional diagram. The SVM then tries to group the points by imaginary lines.

In the case of working with an ARM controller, the *CMSIS DSP* library documented at [4] has established itself as a de facto standard. It is interesting to note that Arm points out in several places in the documentation that the name "DSP" is only retained for legacy reasons; the library itself now sees itself as a kind of general store for all kinds of code in the field of signal processing.

Developers can find out which types of hardware acceleration are used in the background by visiting the GitHub repository with the library's source code, which is provided at [5]. For example, the following statement can be found there:

It provides optimized compute kernels for Cortex-M and for Cortex-A. Different variants are available according to the core and most of the functions are using a vectorized version when the Helium or Neon extension is available.

The purpose of "windowing" the FFT raw data is to reduce the effects of inaccuracies occurring in the edge areas due to the limited length of the data array [6]. It is also interesting to note that the FFT data is windowed by a float field, which the NXP example code creates according to the following scheme:

```
void hanning_window(float *window, int length) {
  for (int n = 0; n < length; n++) {
    window[n] = 0.5 * (1 -
     cosf(2 * 3.141593f * n / (length - 1)));
  }
}
```

The actual sensor task then begins by creating various memory areas that are parameterized by the respective initialization function:

```
void app_sensor_task(void* parameters)
{
  . . .
   ringbuffer_init(&ringbuffer_handler,
    (uint8_t*)ring_buffer, sizeof(ring_buffer));
  hanning_window(hanning_ary, APP_FFT_LEN);
  arm_rfft_instance_q15 s;
  arm_rfft_init_q15(&s, APP_FFT_LEN, 0, 1);
```

The "meat" of the sensor data processing is then found in an endless loop. For reasons of space, I will just show the innermost working part here. What is important here is the call of the method `arm_rfft_q15`, which is responsible for the actual application of the transformation to the sensor data.

```
  for (int i=0; i < APP_FFT_LEN; i++) {
    g_app.rfftInBuf[j][i] -= g_app.dcEMAs[j];
    fft_buffer[i] = g_app.rfftInBuf[j][i] *
     hanning_ary[i];
    ftmp = (float)(g_app.rfftInBuf[j][i]) / g_rmsDiv;
    rmsAry[j] += ftmp * ftmp;
  }
  arm_sqrt_f32(rmsAry[j] / APP_FFT_LEN, rmsAry + j);
  arm_rfft_q15(&s, fft_buffer, g_app.rfftOutBuf[j]);
```

Data Analysis Using an SVM

If we assume, for the sake of simplicity, that the measurement data processing would only provide two coefficients per measurement, then these could be plotted in a two-dimensional diagram. In the best case, the various operating states are then located in different areas of the plane, which can be separated by imaginary lines (**Figure 10**). In a three-dimensional space, good and critical states could be separated by surfaces – and this basic principle can easily be applied in higher-dimensional spaces as well.

For years, this type of task has been solved by algorithms known as *support vector machines* (SVMs). For those seeking a detailed mathematical introduction, Wikipedia offers a surprisingly detailed explanation of the topic [7].

For the realization of SVMs, the LIBSVM library developed by Chih-Chung Chang and Chih-Jen Lin has become established as a de facto standard. At the time of writing this article, the website with the (open-source) library code can be found at [8].

Next, we will turn to the actual application evaluation logic. Pleasingly, it begins with the realization of a state machine:

```
typedef enum {
    kAppPredicting,
// in main screen, predicting data
    kAppWaitForCollect,
// in train screen, wait for "Start" button
    kAppCollecting,
// in train screen, collecting data
    kAppWaitForReturn,
}AppState_e;
```

Introduction to State Machines

If you are not familiar with state machines, you should get the textbook *Code Reading* written by Diomidis Spinellis. It is very inexpensive on the second-hand market, and in the opinion of the author, it is an absolute must-read for anyone who does not have a strong background in technical computer science.

To actually feed the model, NXP defines wrappers for the methods created in LIBSVM. Of particular importance are the two functions for training and analysis — note that AD_Predict returns a float value with the result of the algorithm, as shown here:

```
__WEAK void* ad_train(float features[][APP_FEATURE_DIM],
int featureCnt,float gamma, float nu) {
    return svm_train_mode(features, featureCnt,gamma, nu);
}
volatile float g_modelRetPlaceHolder;
__WEAK float ad_predict(void* pvModel,
     float feature[APP_FEATURE_DIM]) {
  return svm_model_pre(pvModel, feature);
}
```

The SVM task starts by harvesting the most recent accelerometer value, which is written to a FreeRTOS queue by the previously discussed task. From now on, the content of the feature variable is a representation of the most recent sensor values in both the training and prediction cases.

```
void app_algo_task(void* parameters){
  . . .
  float feature[APP_FEATURE_DIM];
  . . .
  for (;;) {
    xQueueReceive(g_app.qh_NewSample,
      feature, portMAX_DELAY);
    g_app.isNewFeature = 1;
```

In the next step, we can turn to the training logic. The first task is to collect the incoming sensor samples — a buffer is filled via the g_app.featureNdx variable:

```
    memcpy(g_features + g_app.featureNdx,
      feature, APP_FEATURE_SIZE);
    g_app.featureNdx++;
```

When the buffer reaches the limit stored in APP_FEATURE_CNT, the actual training takes place.

```
    if (g_app.featureNdx == APP_FEATURE_CNT) {
      int32_t pre_svs_cnt = 0;
      if(g_app.inc_train){
        ad_get_support_vector(g_app.pADModel,
         &(g_features[APP_FEATURE_CNT]), &pre_svs_cnt);
        g_app.inc_train = 0;
      }
      g_app.pADModel = ad_train(g_features,
APP_FEATURE_CNT + pre_svs_cnt, g_app.gamma, g_app.nu);
      deSlideAcc = 0;
      if (g_app.pADModel) {
        g_app.appState = kAppWaitForReturn;
      }
    }
```

In this area, two things are relevant: firstly, training the model and, secondly, updating the state machine.

The next task of the prediction code is to call the actual predict method, which — as discussed above — returns a result from the model.

Practical experience shows that the bulk of the intelligence in such AI tasks lies in the weighting of the results – here, in addition to a multiplication, we also note an inclusion of the sign. In any case, the reward for our efforts is that the variable g_app.health now shows the current state of health of the system to be maintained.

```
  ret = ad_predict(g_app.pADModel, feature);
  float retSign = -1.0f;
  if(ret > 0){
    retSign = 1.0f;
    ret *= 25;
  } else {
    ret *= 1;
  }
  g_app.health = (g_app.health + ret + 0.5 * retSign);
```

Practical experience in design shows that a plausibility check or value restriction of the results returned by an AI model always makes sense. This applies in particular if the values — as is the case here in the variable g_app.health — are integrated or accumulated over the program runtime.

```
  if (g_app.health > APP_HEALTH_ABS_MAX)
    g_app.health = APP_HEALTH_ABS_MAX;
  if (g_app.health < -APP_HEALTH_ABS_MAX)
    g_app.health = -APP_HEALTH_ABS_MAX;
```

The next step of the program then implements a hysteresis. This ensures that the model does not oscillate in borderline cases:

```
  if (g_app.fanState == kFanOn) {
if (g_app.health < APP_ABNORMAL_LEVEL - APP_SWITCH_BAND)
      g_app.fanState = kFanErr;
  } else if (g_app.fanState == kFanErr) {
if (g_app.health > APP_ABNORMAL_LEVEL + APP_SWITCH_BAND)
      g_app.fanState = kFanOn;
  } else {
      g_app.fanState = kFanOn;
  }
```

The rest of the code is then just responsible for switching the various LEDs on and off to make the ML model's results visible to the user:

```
  if(fanState == kFanErr){
    LED_OFF(GREEN);
    LED_ON(RED);
    LED_GREEN_OFF();
    LED_RED_ON();
  }else{
    LED_OFF(RED);
    LED_ON(GREEN);
    LED_RED_OFF();
    LED_GREEN_ON();
  }
```

Perspective

The experiments shown here have demonstrated that the realization of predictive maintenance is nowhere near as complicated as the AI development community, which is always eyeing "other people's money", likes to present it.

In particular, anyone who invests a little time in the background of SVM and Co. can quickly develop quite powerful systems. The author hopes that the experiments carried out here will provide a first incentive for readers to delve deeper into this fascinating area of software technology. ◀

Translated by Jörg Starkmuth — 240452-01

About the Author

Engineer Tam Hanna has been working with electronics, computers, and software for more than 20 years; he is a freelance developer, book author and journalist (www.instagram.com/tam.hanna). In his free time, Tam is involved in 3D printing and selling cigars, among other things.

Questions or Comments?

Do you have questions or comments about this article? Email the author at tamhan@tamoggemon.com, or contact Elektor at editor@elektor.com.

Related Product

- **Get Started with the NXP FRDM-MCXN947 Development Board (Bundle)**
  www.elektor.com/20990

H0W2
Get Started with the NXP FRDM-MCXN947 Development Board
Develop projects on connectivity, graphics, machine learning, motor control, and sensors
Dogan Ibrahim
NXP
elektor

WEB LINKS

[1] SDK downloader for MCUXpresso IDE: https://mcuxpresso.nxp.com/en
[2] Example project for anomaly detection (GitHub): https://github.com/nxp-appcodehub/dm-on-device-training-fan-anomaly-on-mcxn947
[3] Tam Hanna, "FFT with a Maixduino," ElektorMag 3-4/2023: https://www.elektormagazine.com/magazine/elektor-292/61493
[4] CMSIS DSP library, Documentation: https://arm-software.github.io/CMSIS_5/DSP/html/structarm__rfft__instance__q15.html
[5] CMSIS DSP library (GitHub): https://github.com/ARM-software/CMSIS-DSP
[6] Introduction to the windowing of FFT raw data (Application Note LDS Dactron): https://tinyurl.com/FFT-Windows
[7] Support Vector Machine (Wikipedia): https://en.wikipedia.org/wiki/Support_vector_machine
[8] LIBSVM — A Library for Support Vector Machines: https://www.csie.ntu.edu.tw/~cjlin/libsvm/
[9] NXP MCX portfolio: https://tinyurl.com/NXP-MCX-MCUS
[10] AI accelerators in the NXP MCX portfolio: https://www.nxp.com/docs/en/fact-sheet/MCXNFS.pdf

SPoE - Electromagnetic Compatibility

Single-Pair with Power-Over-Ethernet Through the Eyes of EMC

By Adrian Stirn (Würth Elektronik eiSos)

Due to the simple cabling with only one twisted pair of wires, single-pair Ethernet is extremely attractive. If the power supply function can be integrated in addition to data transmission, the solution "Single-Pair with Power-over-Ethernet" (SPoE) is unbeatable. This article describes the EMC behavior of the SPoE reference design from Würth Elektronik.

There are many possible applications for a 10 Mbit/s SPoE interface with integrated energy transmission, such as the operation of sensors at long distances. With the growing demand for data and power in areas such as industrial automation, automotive and IoT networks, SPoE (often referred to as Single-Pair Ethernet with Power-over-Data-Lines — SPE PoDL) offers an efficient solution that enables both data and power transmission over a single twisted pair of wires. This article deals with the EMC behavior of the SPoE reference design RD041 from Würth Elektronik [1], focusing on the low emissions and high immunity to continuous and transient disturbances.

SPoE Reference Design and Test Setup

The reference design RD041 in **Figure 1** is an Ethernet interface that transmits data over two wires and simultaneously supplies power, combining "Single Pair Ethernet" (SPE) with "Power over Data Lines" (PoDL).

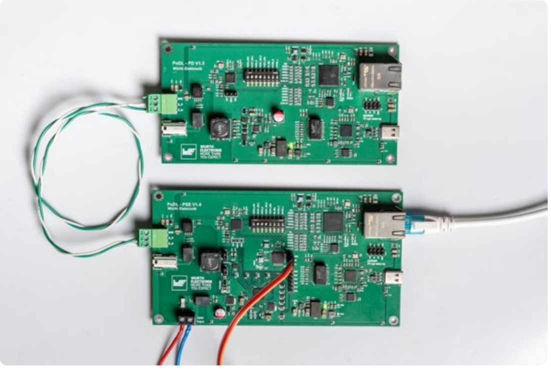


Figure 1: PSE (bottom) and PD (top) of the RD041 during start-up.

SPE enables Ethernet communication with only one pair of wires, making it ideal for space-saving and cost-sensitive applications in industrial, automotive and IoT devices. PoDL extends Power over Ethernet (PoE) to SPoE enabling power and data transmission over a single twisted pair of wires, simplifying installations and reducing costs.

To carry out the EMC tests, the reference design consisting of two boards (Powered Device — PD and Power Sourcing Equipment — PSE) is operated with a defined transmission path as the test object.

Figure 2 shows this transmission path with the auxiliary equipment required for operation shown in gray. A shielded SPE cable or an unshielded twisted pair cable can be used as the transmission path between the PD and PSE. Laptop 1 is connected to the PD via a shielded 10 Mbit/s Ethernet interface. It receives its IP address from the DHCP server (WLAN router) via the SPoE interface, which connects the PSE and PD. The PD is powered by the PSE board.

The PSE part of the reference design has a 24 V input that supplies the SPoE reference design. In the reference design, this connection is designed as a port on the local DC power supply network. The 10 Mbit/s interface is connected to the router via a shielded Ethernet cable and then on to the Laptop 2 being evaluated. Both laptops receive their IP addresses from the DHCP server. The reference design is therefore ready and can be used as an adapter to extend an Internet connection in a plug-and-play manner.

Test Software and Performance Criteria

Laptop 1 serves as a server that sends known test data to Laptop 2. This evaluates the error rate in the transmission and the transmission speed using a Windows application. This monitoring application has already been successfully used in the application notes for the GB-Ethernet design (ANP116 and ANP122) [2][3].

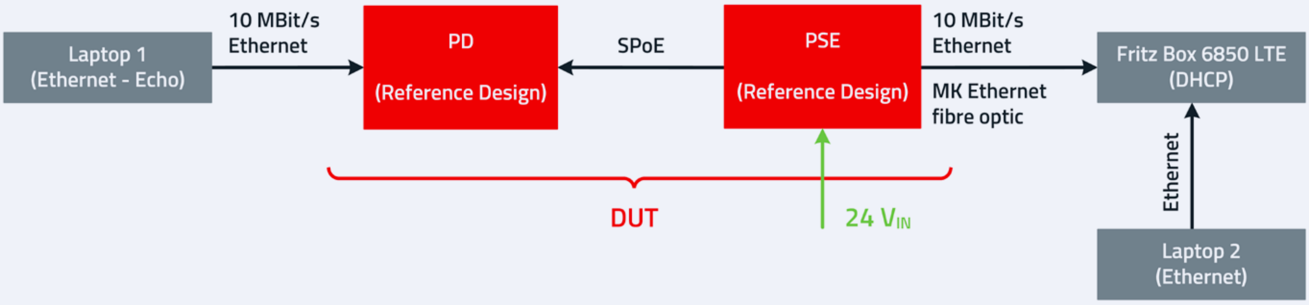


Figure 2: Test setup for evaluating the performance of the SPoE reference design.

Figure 3 shows that an average transmission rate of slightly less than 9 Mbit/s can be achieved. In the selected test case, the packets have a short transmission time compared to the tests on the Gigabit Ethernet interface. During the tests, it is therefore possible to test with a measurement and reaction time of one second.

During the immunity tests, the performance of the reference design is checked against the criteria in **Table 1**.

When evaluating the performance criteria, the auxiliary equipment also has a relevant impact on the performance of the transmission path. For example, the transmission error rate increased when a WiFi interface was used between the router and Laptop 2 instead of a shielded Ethernet cable.

Shield Connection of the Ethernet Interface

As already shown in Application Note ANP116 [2], the following components are suitable for connecting the shield of the Ethernet socket to the GND plane of the board as shown in **Figure 4**:

- 2× 10 nF X7R 1206 MLCC 100 V (part no. 885012208112)
- SMT varistor (part. no. 82551600)

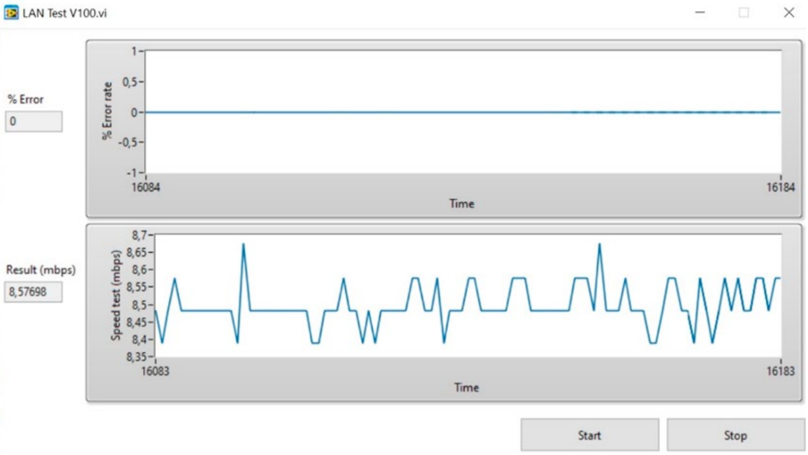
The shield connection is used for the SPoE interface and for the 10 Mbit/s Ethernet interface. The varistor is not shown in Figure 4. This can be placed parallel to the capacitor on one side. The USB interface on the boards is not used for the analyses in this article.

Emission of the Reference Design

The emission of the reference design in the configurations described above is analyzed below. The USB interface is not used for the emission measurements in accordance with CISPR 32; communication takes place exclusively via the Ethernet and SPoE interfaces.

Radiated Emissions

PD and PSE are operated and tested together to test radiated emissions. The cable length at the SPoE inter-



% Error
0

Result (mbps)
8,57698

Figure 3: Monitoring of the SPoE interface. PSE and PD are connected to the network via an Ethernet interface.

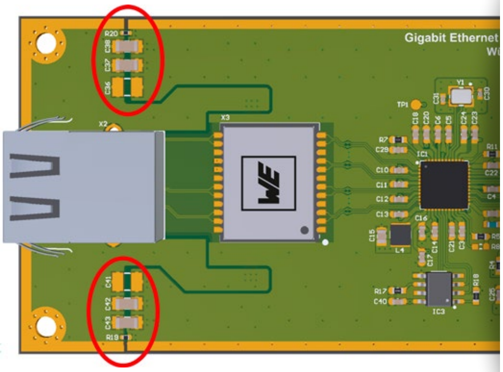


Figure 4: Shield connection of the Ethernet socket according to ANP116. There is an MLCC on each side; the varistor is not shown.

Measurand	Performance Criterion	Technical Evaluation Criterion
Data rate	A	Above 8 Mbit/s
	B	Below 8 Mbit/s
Error rate	A	0%
	B	Error rate increased or communication loss with automatic reconnection.
	C	100% = communication loss, no automatic reconnection possible.

Table 1: Performance criteria of the SPoE reference design.

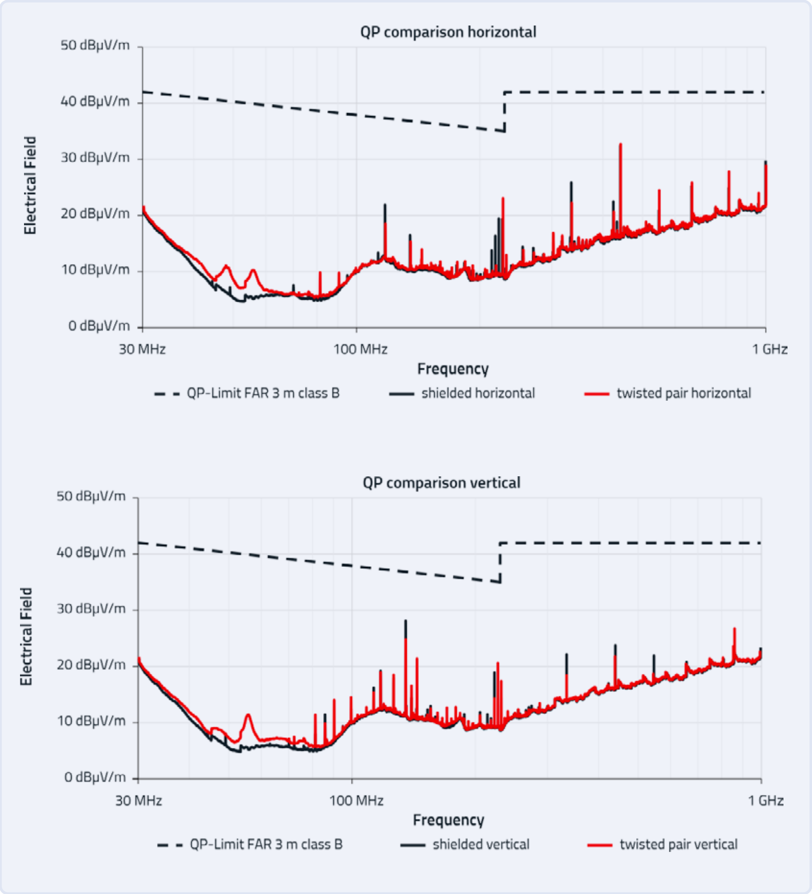


Figure 5: Comparison of the radiated interference emission of the SPoE reference design when using shielded SPE cable or unshielded twisted pair cable.

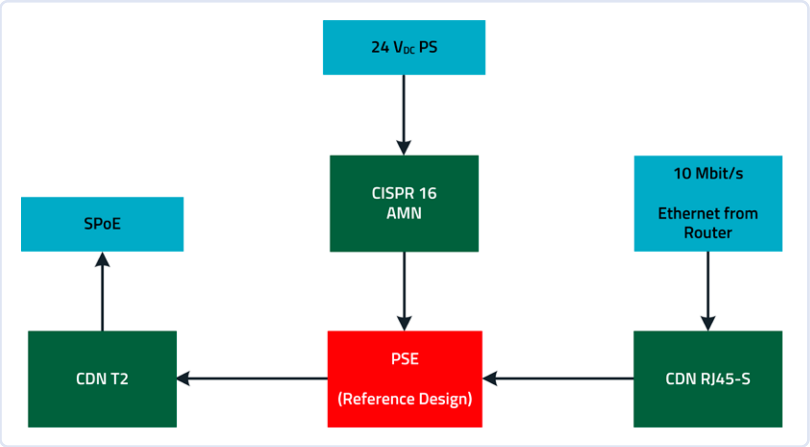


Figure 6: Measurement setup for testing the radio interference voltage on the PSE.

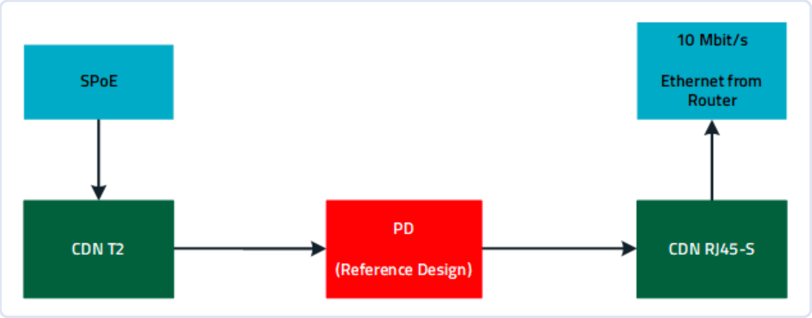


Figure 7: Measurement setup for testing the radio interference voltage on the PD.

face for the emission test is 1 m horizontally in the field between the two boards.

The PSE is supplied with 24 V via feed-through filters from a laboratory power supply unit located outside the anechoic chamber. The PSE and PD are located inside the full anechoic chamber, whereby the PSE is connected via 10 Mbit/s Ethernet with a shielded cable to an Ethernet fiber optic converter, which establishes the connection to the router and Laptop 2 outside the measuring hall. The PD is connected to Laptop 1 via a shielded Ethernet cable, which is inside a shielded box (Figure 2).

This setup prevents interference from the router or the laptops being measured during the emission tests. As this set-up is also used in the radiated immunity test, the laptops and the router are also protected from being affected by the electric field of the immunity test.

The interference spectrum of the reference design in **Figure 5** is at least 9 dB below the limit value for both cable types used, and the results are comparable.

Radio Interference Voltage

The PSE and the PD may be physically separated in the subsequent setup and are therefore tested individually. The DC input of the PSE is to be tested as a possible DC mains input, the Ethernet cable of the 10 Mbit/s Ethernet interface and SPoE interface are to be regarded as a long network cable. The emission of the DC input on the PSE is measured in accordance with CISPR 16 with a 50 µH AMN (Artificial Mains Network), the emission of the network ports is measured using 150 Ω CDNs (Coupling Decoupling Network), which act as an AAN (Asymmetric Artificial Network). The test setups for measuring the radio interference voltage at the PSE and PD are shown in **Figures 6** and **7**.

A CDN T2 is used to test the SPoE interface with an unshielded twisted-pair cable; when measuring the interference on the shielded SPoE interface, the emission is tested using a self-built CDN for shielded SPE cables. The interference on the shielded Ethernet cable is measured using a CDN for shielded Cat5e cables. Network replicas and CDNs must always be terminated with 50 Ω in the test setup (either by the test receiver or by means of a 50 Ω resistor).

The measurement setup of the conducted interference emission on the PSE board of the SPoE reference design is shown in **Figure 8**. The measurement results of the conducted emissions on the SPoE interface of the PSE are shown in **Figure 9**, with the results of the quasi-peak detector at the top and the mean value detector at the bottom. Further measurement results of the conducted emission test are shown in detail in Application Note ANP141 [4].

The SPoE reference design is characterized by low radiated and conducted emissions. Despite open circuit boards without a shielded housing, the emission results are well below the EMC limit values for residential areas. Even with an unshielded twisted pair cable, the results of the SPoE interface are well below the EMC limit values for all measurements. From the point of view of emission testing, shielding of the SPoE interface is therefore not necessary.

Interference Immunity of the Reference Design

A detailed examination of the immunity of the reference design to continuous and transient interference as well as the ESD behavior is beyond the scope of this article. Therefore, please refer to the comprehensive Application Note ANP141 [4], where you will find a detailed description of the test setups and measured values. ⏮

250085-01

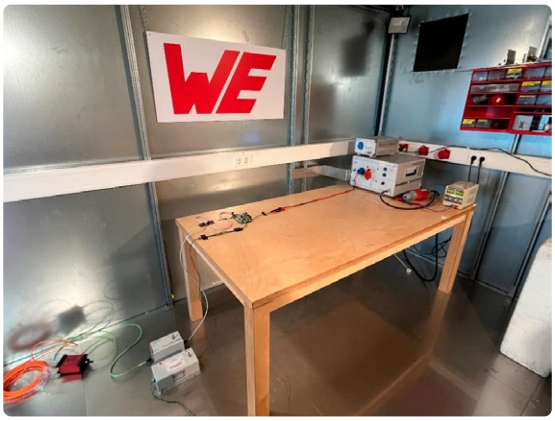
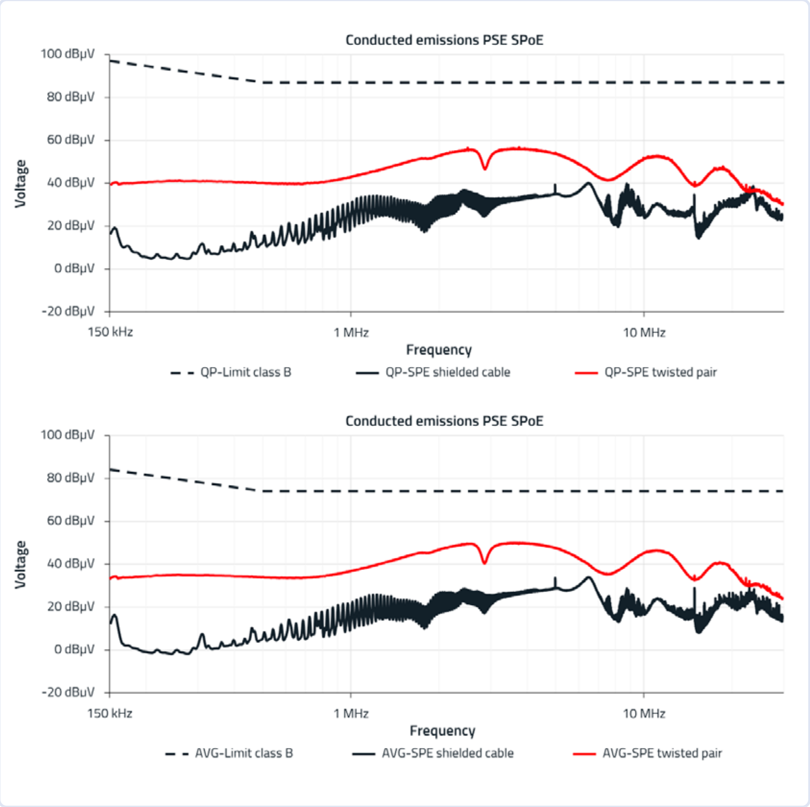


Figure 8: Measurement of the conducted interference emission on the PSE board of the SPoE reference design.

◄

Figure 9: Measurement results of the conducted interference emission on the SPoE interface of the PSE. The results of the quasi-peak detector are shown at the top and the results of the average detector at the bottom.

▼



About the Author

Adrian Stirn studied electrical engineering at Heilbronn University and then set up the company's own EMC laboratory at Würth Elektronik's Waldenburg site. Since 2016, he has been supporting customers in troubleshooting EMC problems during development. He also focuses on product conformity in the European Economic Area and the protection of ESD-sensitive components.

WEB LINKS

[1] Heinz Zenkner, "Design of a Single Pair Ethernet System with Power over Data Lines (SPoE)," Reference Design from Würth Elektronik RD041: https://www.we-online.com/RD041

[2] Adrian Stirn, "Gigabit Ethernet interface from an EMC perspective," Application Note from Würth Elektronik ANP116: https://www.we-online.com/ANP116

[3] Adrian Stirn, "Gigabit PoE Interface from an EMC perspective," Application Note from Würth Elektronik ANP122: https://www.we-online.com/ANP122

[4] Adrian Stirn, "The SPoE Interface from an EMC Perspective," Application Note from Würth Elektronik ANP141: https://www.we-online.com/ANP141

Color TV A Wonder of Its Time

Creating a New World

By Vince Sosko (Mouser Electronics)

Seventy years ago, the RCA CT-100 opened our eyes to a new visual experience. This groundbreaking device featured a 15-inch screen and was the first color TV to be mass-produced. Let's take a trip back!

(Source: Early Television Museum)

Earlier this year, over 120 million people tuned in to watch the Super Bowl, and many made a big purchase in the weeks prior to capture the spectacle in all its grandeur and glory. The newest smart TVs, with their amazing picture quality and immersive sound experience [1], afford viewers the ability to feel like they're at the game without paying an astronomical amount for a ticket. Today, an array of advanced light-emitting diode (LED) technologies lights up our home screens with magnificent colors. But to fully appreciate the majesty of modern televisions, let's take a trip back to 1954, when color televisions first broke into the market.

While the collector community debates between the Admiral C1617A and the Westinghouse H84OCK15 as the first true color TV to be released, we'll focus on the RCA CT-100, which came out right on their heels and was the first color TV to be mass-produced (while the Admiral and Westinghouse were both discontinued only months after release).

What Was the CT-100?

Marketed as "The Merrill" (nicknamed after RCA executive George H. C. Merrill), the CT-100 marked a turning point in the evolution of television, introducing vibrant hues to a world accustomed to monochrome screens. Manufactured by the Radio Corporation of America (RCA), this groundbreaking device featured a 15-inch screen that brought a spectrum of colors into living rooms, transcending the limitations of black-and-white broadcasting. At the center of the RCA CT-100's revolutionary design were its technical components that paved the way for the future of color television.

Cathode-Ray Tube

The heart of the system was the cathode-ray tube (CRT), responsible for rendering the images on the screen. Unlike its monochromatic predecessors, the CT-100 incorporated a shadow mask — a metal sheet with precision-drilled holes — to ensure precise placement of color phosphors. These phosphors emitted light when struck by the electron beam, allowing for a wide range of color hues and saturation. This breakthrough allowed viewers to experience a level of visual richness previously unseen, making the CT-100 a symbol of technological advancement and innovation.

CTC-2 Chassis

The CT-100 incorporated capacitors, resistors, transistors, and vacuum tubes into its CTC-2 chassis (**Figure 1**) to process and amplify the incoming television signal. Included among its complex layout of 36 vacuum tubes was the 15GP22 color picture tube, which used electrostatic convergence that necessitated static and focus adjustment knobs on the cabinet. Between their short service life and higher cost compared to other picture tubes of the time, the 15GP22 became a rare component that, while essential for synchronizing the color and brightness information and driving the CRT to produce the final image, has made restoration projects for the CT-100 rather difficult.

In addition to these new-age picture tubes, the CT-100 featured sophisticated color decoding circuits responsible for processing the color information contained in the broadcast signal. These circuits decoded, or "demodulated," the chrominance (color) signal and synchronized it with the luminance (brightness) signal to accurately reproduce colors on the screen.

Tuner and RF Components

Color was not the only marvel that set the CT-100 apart from other televisions during this broadcasting revolution. The CT-100 also featured a turret-style tuner and RF components that allowed it to receive broadcast signals over the airwaves. The tuner tuned into specific channels, which surprisingly included very high frequency (VHF) and ultra-high frequency (UHF) broadcasts emerging at the time, while RF components amplified and processed the incoming signal before it was decoded and displayed on the screen. These components, combined with color-rendering capabilities, forced television network companies of the time to play catch-up with the RCA CT-100.

The Way It Was

The release of the RCA CT-100 in 1954 unfolded against the backdrop of post-World War II optimism and the burgeoning consumer culture of the 1950s. Television, already a popular medium, was evolving rapidly. The United States was experiencing economic prosperity, and as households became more affluent, the desire for modern conveniences grew. The release of the CT-100 aligned with this cultural shift, offering consumers a glimpse into the future of television — a future painted in vibrant colors.

The timing of the CT-100's release also coincided with the concurrent efforts of major television networks to embrace color broadcasting. NBC made history by airing the first coast-to-coast color broadcast of the Tournament of Roses Parade on January 1, 1954. This event served as a dazzling showcase for the capabilities of color television and generated tremendous excitement among viewers.

The Bigger Picture

Since the launch of the RCA CT-100, color television has become a ubiquitous part of our daily lives. Technological advancements have propelled the evolution of television displays, moving from cathode-ray tubes to flat-panel technologies such as LED, liquid crystal display (LCD), and organic light-emitting diode (OLED). The journey from the CT-100 to contemporary displays has seen improvements in resolution, contrast ratios, and overall visual fidelity.

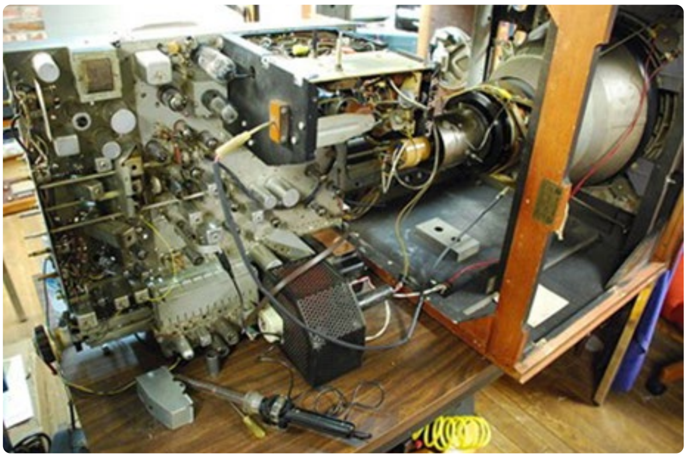


Figure 1: The RCA CT-100's CTC-2 chassis was not only a wonder of its time but has proven to be a treasure for collectors and restoration projects. (Source: HumanisticRationale, Wikimedia Commons, https://commons.wikimedia.org/wiki/File:CT-100_with_CTC-2_chassis.JPG)

Beyond hardware, the evolution of color television is evident in content production. Filmmakers and content creators now leverage the full spectrum of colors to tell compelling stories and create visually stunning experiences. High-definition content, streaming services, and smart TV capabilities have further transformed how we consume television, offering a diverse range of content accessible at our fingertips.

The transition from black-and-white to color television was not without its challenges. Concerns about compatibility, the cost of upgrading infrastructure, and the need for standardization initially slowed the widespread adoption of color broadcasting. However, as technology matured, these challenges were overcome, and color television became the norm rather than the exception.

The RCA CT-100's importance transcended its role as a technological marvel; it marked a cultural and visual revolution. The introduction of color television was a transformative moment in the history of mass media. It elevated the viewing experience, making it more immersive, engaging, and reflective of the vibrant world it aimed to capture.

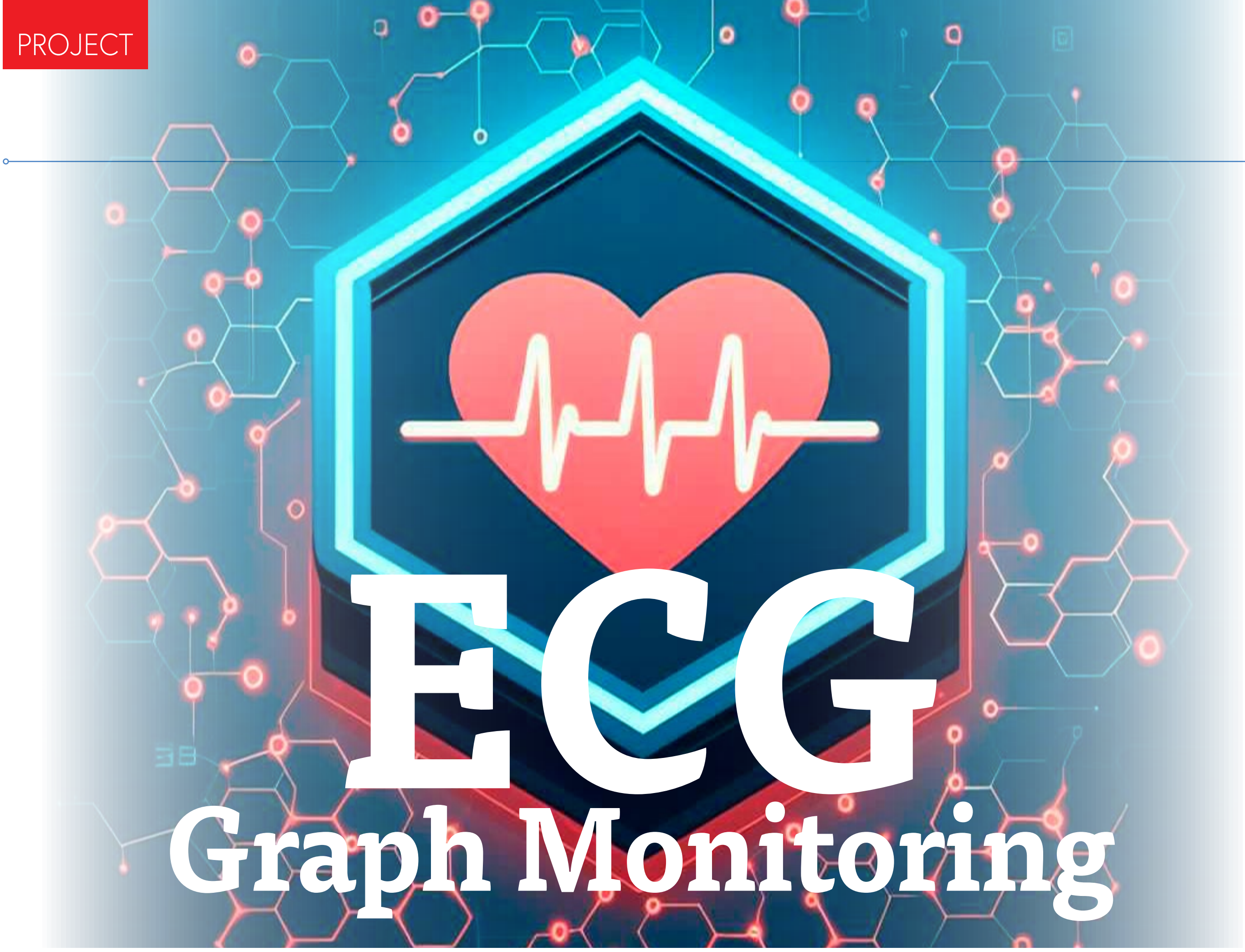
Recently, while watching an old black-and-white show on television, my daughter asked me if that's how the world looked back then. A silly question filled with curiosity and innocence, and one I likely asked my parents as a child, but it reveals just how magical the advent of color TV was. For many in the 1950s, the world outside their own town was accessible only through their television screens, and when vivid colors found their way onto those screens, other parts of the world became even more real and people's imaginations were lent a color palette of possibility.

As we look back at the journey from the CT-100 to the present, it's clear that color television has not only survived but thrived. It has become an integral part of our visual experience, enriching our lives in ways unimaginable to those first viewers in 1954. The evolution of color television mirrors the broader trajectory of technological progress — a journey marked by innovation, adaptation, and an unending quest to enhance daily experiences to new heights. Reflecting upon the introduction of the CT-100, we celebrate not just the evolution of technology but the enduring impact of a device that brought color to our screens and changed the way we view the world. ◄

250016-01

WEB LINK

[1] New Tech Tuesdays (Mouser Electronics), "A New Era of Movie-Watching," January 2024: https://resources.mouser.com/new-tech-tuesdays/new-era-movie-watching



ECG Graph Monitoring

Source: AI-generated by the author

An Implementation with Hexabitz Modules and an STM32CubeMonitor

By Aula Jazmati (Syria)

Imagine being able to monitor ECG data in real-time, wherever you are. This project aims to make this a reality by developing a portable and affordable ECG monitoring system, utilizing Hexabitz modules, STMicroelectronics IDE software, and sensor electrodes.

ECG monitoring is crucial for diagnosing heart conditions such as arrhythmias and heart attacks. Traditional ECG monitoring systems are often expensive and complex. Various ECG monitoring systems have been developed over the years, starting from the use of galvanometers in the early 20th century, up to modern systems that rely on advanced technologies. There are many existing systems around, that offer ECG monitoring, but they are often costly or complicated to use.

Why This Project?

What motivated me to carry out this project was the need for an ECG monitoring system that could be easily used by individuals at home without requiring complex or expensive equipment. In this article, I describe a real-time ECG monitoring system using the flexible and customizable Hexabitz modules. The goal is to create an affordable and portable solution that allows users to monitor their heart's electrical activity, wherever they are.

I conducted several experiments to validate the functionality and performance of our real-time ECG monitoring system. These tests include:

- *Signal acquisition and processing* — Testing the accuracy and reliability of the ECG signal acquisition. I analyzed the quality of the signals and ensured they were free from noise and artifacts.
- *Data transmission and storage* — Evaluating the efficiency of data transmission between the Hexabitz modules and the central monitoring system. I also tested the storage capabilities to ensure that the ECG data is securely saved for future analysis.
- *Real-time monitoring* — Assessing the system's ability to provide real-time monitoring of ECG signals. This involved checking the responsiveness and latency of the system to ensure it can deliver timely information to the user.
- *User interface and experience* — Testing the user interface to ensure it is intuitive and user-friendly. I gathered feedback from users to make any necessary improvements to the design and functionality.

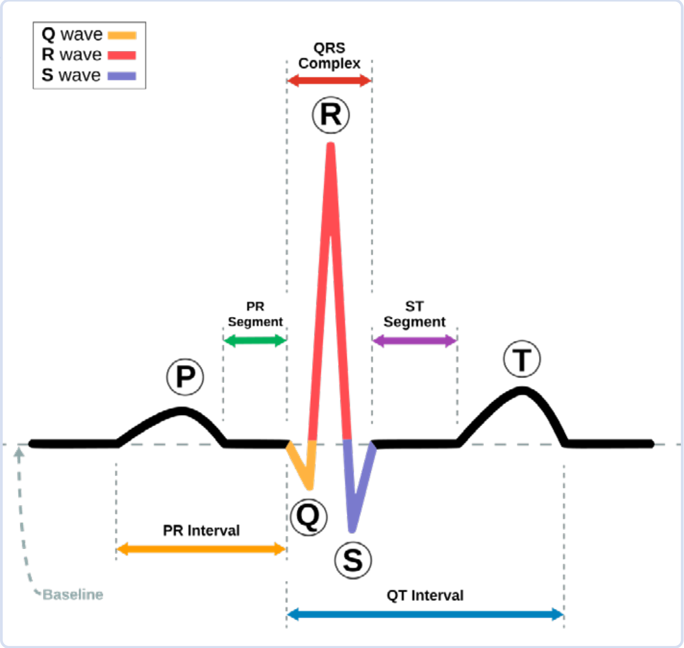


Figure 1: The typical ECG waveform. (Public Domain image created by Agateller [Anthony Atkielski] — https://commons.wikimedia.org/w/index.php?curid=1560893)

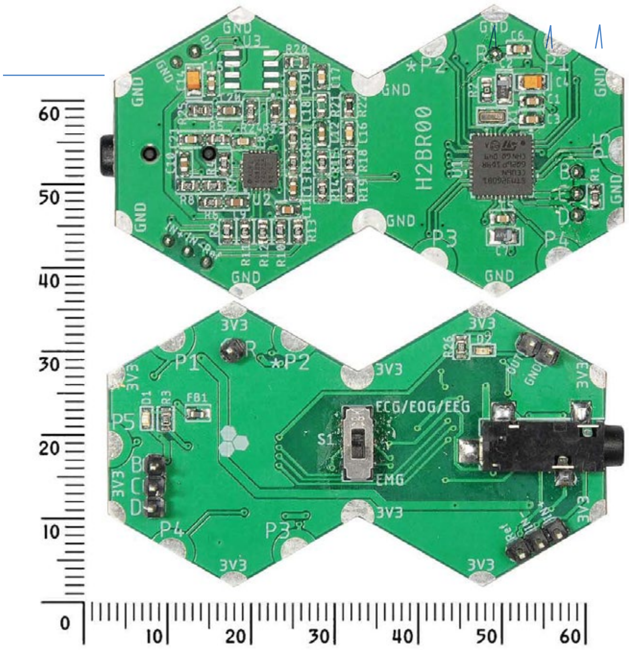


Figure 3: The compact, yet powerful EXG Monitor for heart signals acquisition. (Source: https://hexabitz.com/product/single-lead-exg-monitor-h2br0x/)

These experiments help us to refine the ECG monitoring system and demonstrate its potential for real-world applications.

What Is an ECG?

An ECG is a paper chart or a digital recording of the electrical signals in the heart. These signals — whose typical waveform is shown in **Figure 1** — can be analyzed by studying their components, representing the cardiac electrical activity. To know more about this, you might take a look at [1].

The Project

Figure 2 illustrates the functional diagram of this design, that could be divided into three main logic blocks: a signal acquisition and processing block; a programming and data transmission block; and a monitoring block.

The signal acquisition and processing block, shown on the right, is responsible for acquiring the bio-potential signals from the body and made by two main components, the Single-Lead, EXG Monitor (H2BR0x) and the Sensor cable with electrode pads (3 Leads).

The Hexabitz Single-Lead, EXG Monitor Module (H2BR0) is a one-of-a-kind module that can record publication-grade biopotential signals from your body be it from the heart (ECG), brain (EEG), eyes (EOG), and muscles (EMG). The two sides of this compact, yet powerful module are shown in **Figure 3**, whilst two YouTube introductory videos about

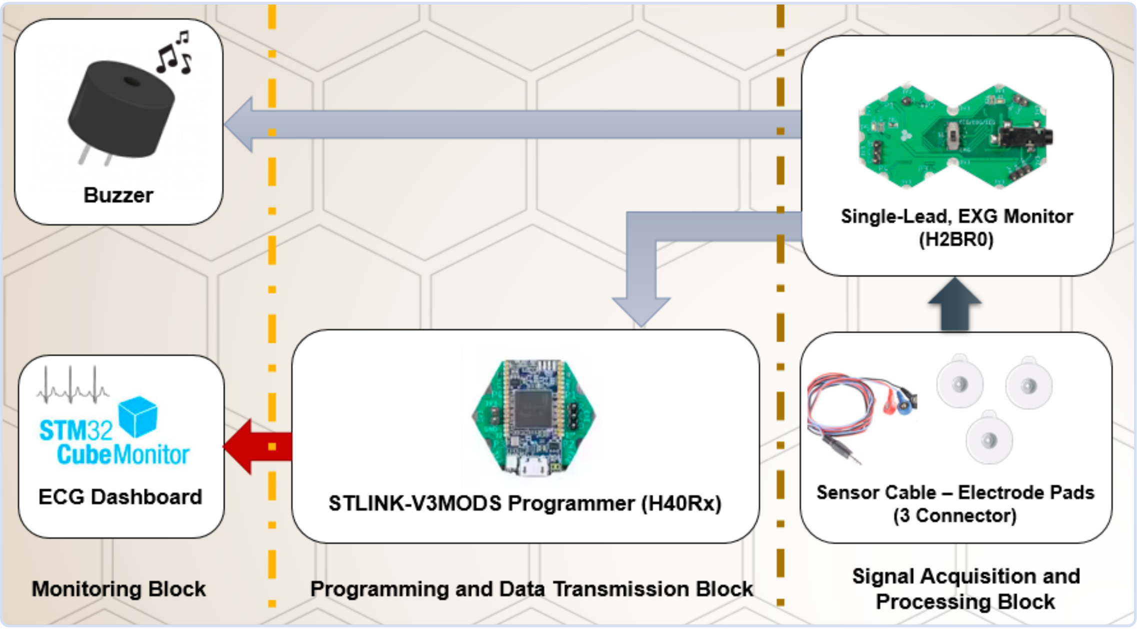

Buzzer
Single-Lead, EXG Monitor (H2BR0)
STM32 CubeMonitor
ECG Dashboard
STLINK-V3MODS Programmer (H40Rx)
Sensor Cable – Electrode Pads (3 Connector)
Monitoring Block
Programming and Data Transmission Block
Signal Acquisition and Processing Block

Figure 2: Functional diagram of the design.

it can be watched at [2] and [3]. The fact sheet of this EXG monitor is available at [4].

What are the points of strength of this unit?

- Record publication-quality biopotential signals like ECG, EMG, EOG, or EEG.
- Small size allows easy integration into mobile and space-constrained projects.
- Notch filter (second order) to remove 50-Hz noise from AC mains.
- H2BR0 is based on an STM32G0 MCU.
- Advanced C code can be programmed with easy-to-use APIs.
- Possibility to be connected to external hardware or combined with other Hexabitz modules!
- Equipped with an open-source MATLAB interface.

The Sensor Cable with three electrode pads is a simple, three-conductors cable with electrode pads (red, black and blue leads). The cable is 24" long and features a 3.5-mm audio jack connector on one end with snap-style receptacles for biomedical sensor pads. The latter are attached to the body to capture the ECG signals, which are then fed into the EXG Monitor module for processing. It can be sourced from local retailers or here [5], for example.

The programming and data transmission block, mainly made of the STLINK-V3MODS Programmer (H40Rx), is illustrated in **Figure 4** and shown in the mid-part of Figure 2. This block handles the programming and debugging of the Hexabitz modules and is responsible for transmitting the processed ECG data to the monitoring system.

The H40Rx is a stand-alone debugging and programming mini probe for STM32 microcontrollers (Hexabitz modules and other MCUs). It supports the Serial Wire Debugging (SWD) interface for the communication with any STM32 microcontroller located on an application board. It also provides bridge interfaces to several communication protocols allowing, for instance, the programming of the target through the bootloader.

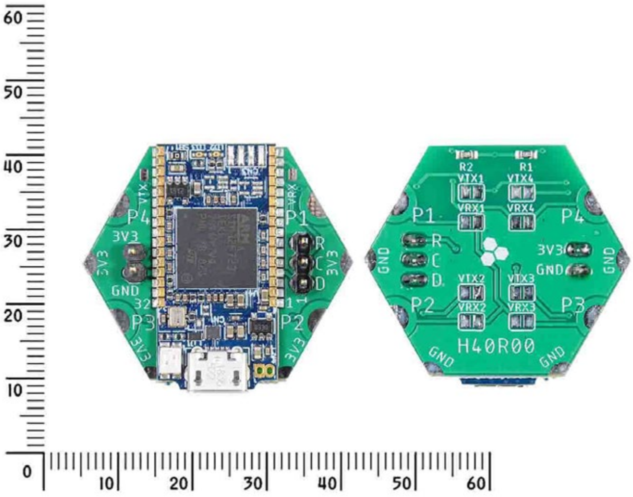


Figure 4: Component and solder side view of the multipurpose STLINK-V3MODS Programmer (H40Rx). (Source: https://hexabitz.com/product/stlink-v3mods-programmer-h40rx-2/)

The monitoring block, shown on the left of Figure 2, provides a real-time feedback. It includes the STM32CubeMonitor Dashboard to display the ECG signals and a buzzer, that emits a sound at each peak of the ECG signal indicating the heartbeats.

Software Tools, IDE, and Libraries

For the realization of this ECG monitor project, the software plays a crucial role. Therefore, I decided to implement it using a combination of specialized tools and libraries that ensure optimal performance and reliability.

- **STM32CubeMonitor**: a versatile tool that allows real-time monitoring and visualization of variables in STM32 applications. It provides a user-friendly interface to create custom dashboards.
- **STM32CubeIDE**: an integrated development environment (IDE) designed for STM32 microcontrollers. It provides a comprehensive suite of tools for coding, debugging, and optimizing embedded applications.
- **Hexabitz Single-Lead, EXG Monitor Module Firmware:** this firmware is essential for capturing high-quality biopotential signals such as ECG. You can download it from its respective GitHub page at [6].
- **Application Code (*main.c*):** the core functionality of this project is implemented in the *main.c* file, written by me. This includes the initialization of peripherals, data processing, and communication routines. You can find the detailed code at [7].

Writing Code with STM32CubeIDE Software

1) Download the latest version of STM32CubeIDE at [8].
2) Clone or download the firmware for the EXG Monitor Module. The latest stable release is always on the *master* branch, at the bespoken GitHub page [6].

The functionality of the project will be implemented in the *main.c* file of the module firmware. Add your one-time initialization code at the top of `UserTask` before the endless loop.

```
float Sample;
float ecgFilteredSample;
```

Add your recurring code inside the endless loop in `UserTask`. Ensure that MCU periodically checks the ECG sample value and ECG filtered sample value of the sensor, using the API.

```
ECG_Sample(& Sample, & ecgFilteredSample);
```

3) The module has several ports for communication with other modules. In our case, we want to control the pins of one port independently. This can be accomplished by configuring the module port to function as a stand-alone port.

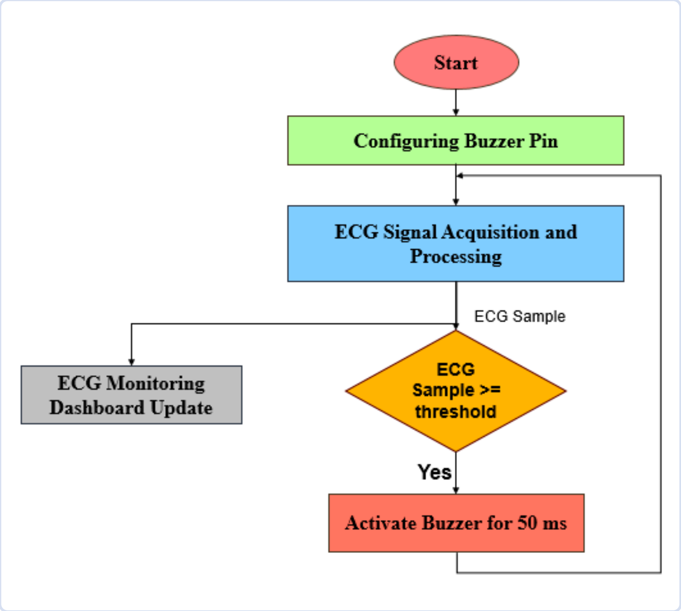

Start
Configuring Buzzer Pin
ECG Signal Acquisition and Processing
ECG Sample
ECG Monitoring Dashboard Update
ECG Sample >= threshold
Yes
Activate Buzzer for 50 ms

Figure 5: Flow chart of the sampling sequence.

First, reduce the number of ports in the *.h* file. This can be achieved by replacing the original number of ports with one less, and commenting out the last port along with its related USART port and UART Init prototype. For example, if the original number of ports is 6, it should be reduced to 5 by commenting out `//#define _P6`, `//#define _Usart6 1` and `//extern void MX_USART6_UART_Init(void);`

Second, comment the `MX_USART6_UART_Init()` port inside the `Module_Peripheral_Init()` function in the *.c* file, also commenting this port inside the `GetPort` function. Now we can use the pins as GPIO (input/output). In our case, to connect a buzzer.

Figure 5 illustrates the flow chart of the sampling sequence. If the ECG sampled value achieves the required condition, the buzzer is turned on at a certain time.

```
if(Sample > 2.2) {
 HAL_GPIO_WritePin(GPIOD,GPIO_PIN_3, 1);
 HAL_Delay(50);
 HAL_GPIO_WritePin(GPIOD,GPIO_PIN_3, 0);
}
```

The STM32CubeIDE debugging function can be used to monitor the sample values.

Designing and Implementing the Dashboard

The screenshot of **Figure 6** figures illustrates the overall flow design in STM32CubeMonitor, which is based on the popular Node-RED development tool. It shows how data are collected from the sensor, processed, and finally displayed on the dashboard. This dashboard provides a real-time interface for monitoring and visualizing sensor data. **Figure 7** shows the final layout of the dashboard, highlighting various interactive elements like charts and buttons.

The configuration of the nodes is another relevant step in the process. **Figure 8** shows the configuration of the *Acq IN* node, which is responsible for acquiring ECG data from the sensor, whilst **Figure 9** details the settings for the *Acq OUT* node. It plays a crucial role in transmitting the processed data to the subsequent nodes in the flow, enabling seamless data transfer and ensuring that the data is ready for further processing or visualization.

Figure 10 illustrates the configuration of the *Clear Button* node, which provides a user interface element to clear or reset the data displayed on the dashboard. This node enhances interactivity by allowing users to manage and refresh the data views as needed.

The chart configurations are illustrated in **Figure 11**. This node will be responsible for plotting the values of the sensor on the user interface. The *Variables* node will always contain ECG filtered sample variable,

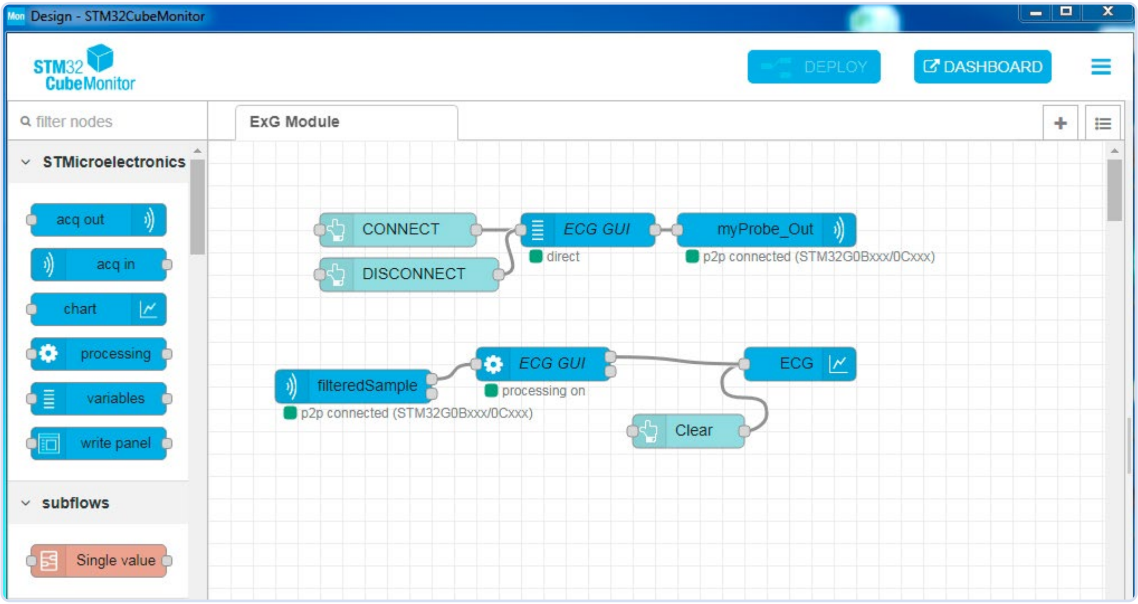


Figure 6: The flow-based, user-friendly graphical interface of STM32CubeMonitor, showing the building blocks of the design.

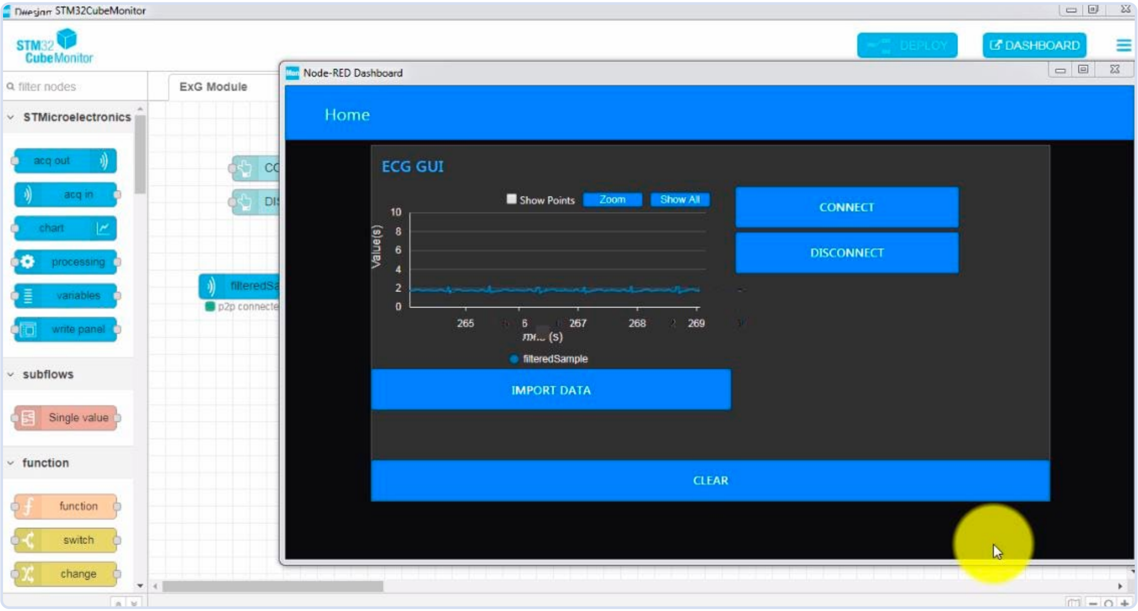
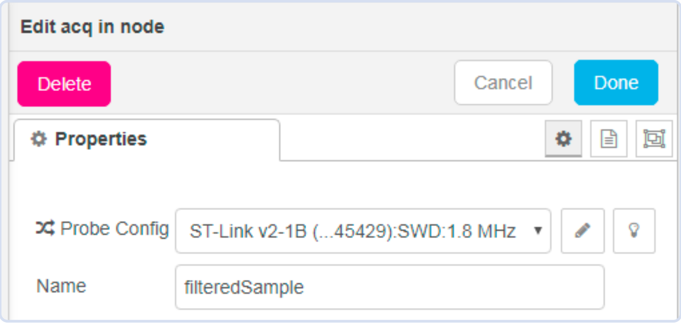


Figure 7: The Node-RED dashboard window, showing real-time sensor data.


Edit acq in node

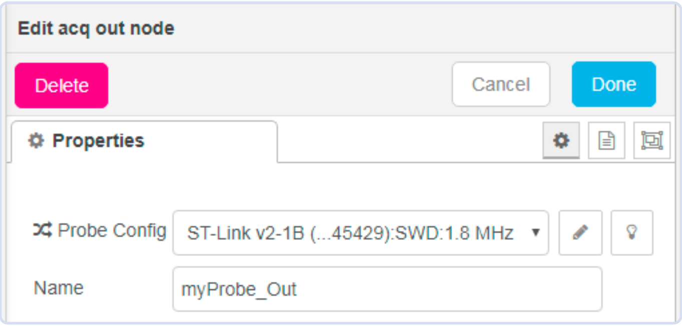
Delete | Cancel | Done

Properties

Probe Config: ST-Link v2-1B (...45429):SWD:1.8 MHz

Name: filteredSample

Figure 8: Configuration of the *Acq IN* node.


Edit acq out node

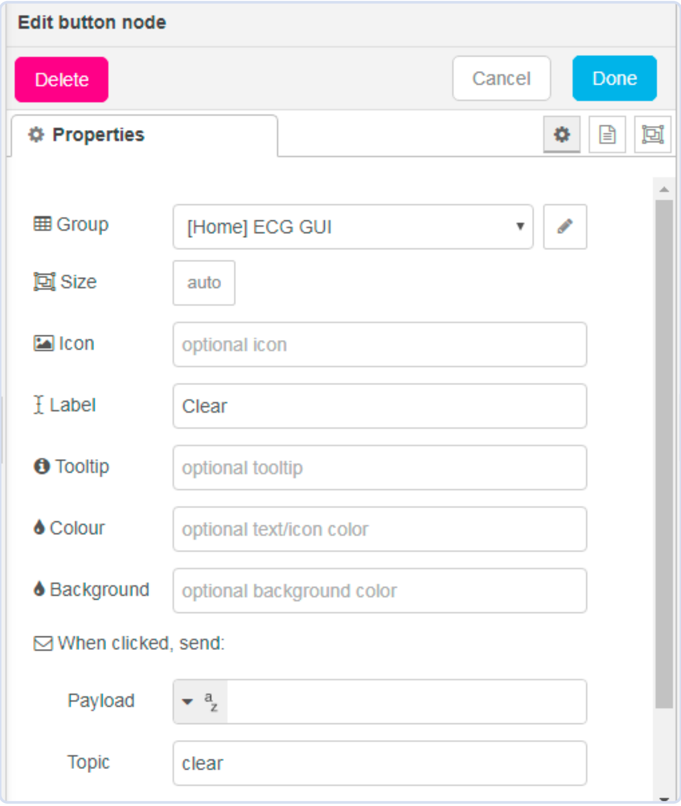
Delete | Cancel | Done

Properties

Probe Config: ST-Link v2-1B (...45429):SWD:1.8 MHz

Name: myProbe_Out

Figure 9: Settings for the *Acq OUT* node.


Edit button node

Delete | Cancel | Done

Properties

Group: [Home] ECG GUI

Size: auto

Icon: optional icon

Label: Clear

Tooltip: optional tooltip

Colour: optional text/icon color

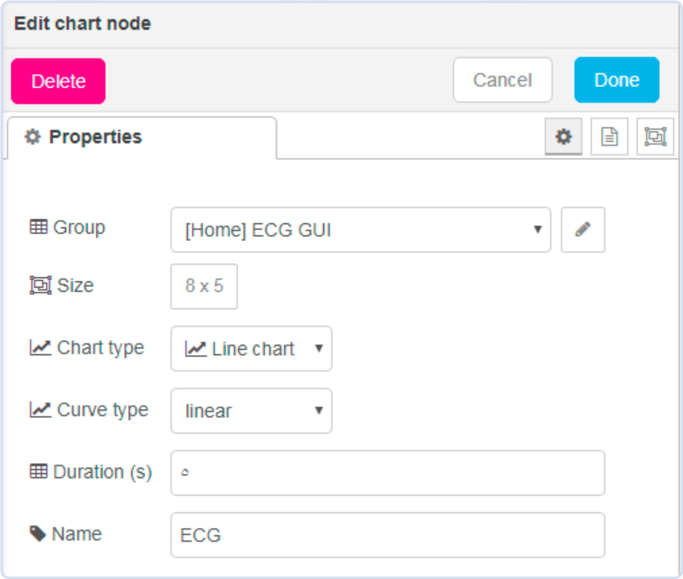
Background: optional background color

When clicked, send:

Payload: ^a_z

Topic: clear

Figure 10: Configuration of the *Clear Button* node.


Edit chart node

Delete | Cancel | Done

Properties

Group: [Home] ECG GUI

Size: 8 x 5

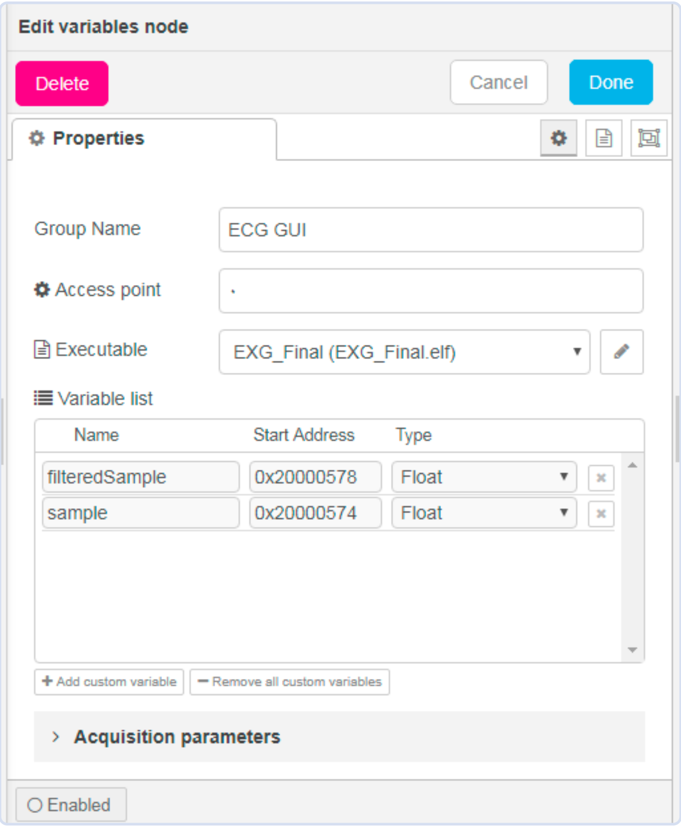
Chart type: Line chart

Curve type: linear

Duration (s): ○

Name: ECG

Figure 11: The chart configuration window.



Group Name: ECG GUI

Access point: .

Executable: EXG_Final (EXG_Final.elf)

Variable list

Name	Start Address	Type
filteredSample	0x20000578	Float
sample	0x20000574	Float

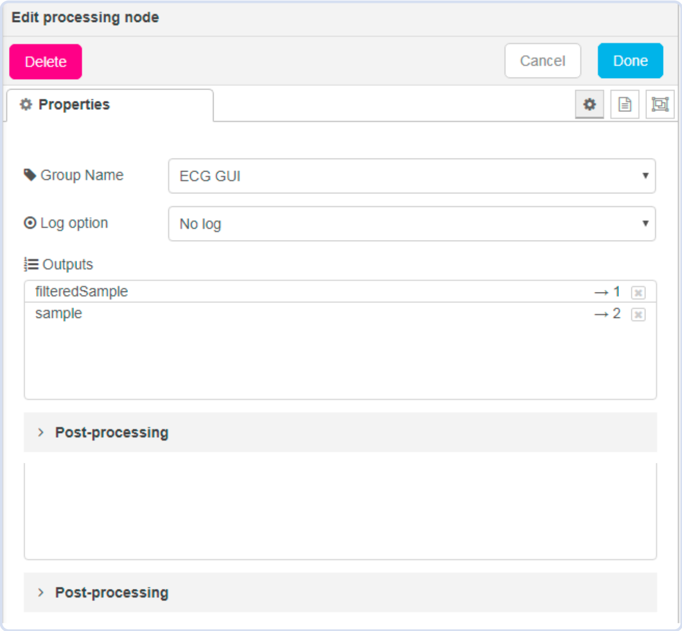
○ Enabled

Figure 12: Configuration settings for the *Variables* node.

which will be plotted on a line chart as curves. **Figure 12** details the configuration settings for the *Variables* node, which holds the necessary variables for collecting and displaying ECG data. It ensures the correct variables are tracked and updated.

In **Figure 13**, the settings of the *Processing* node are shown. We selected the filtered sensor readings without performing any additional processing, as the filtering is done within the sensor module itself.

To modify your layout, you need to click on the Dashboard menu, on the top-right side, select the layout and edit the layout; in *theme*, you can change the background color, the buttons color, etc.



Group Name: ECG GUI

Log option: No log

Outputs

filteredSample → 1
sample → 2

Figure 13: Setup window for the *Processing* node.

To simulate the remote access on our dashboard, we can use the link *http://localhost:1880/*. The dashboard can be controlled using any browser, as shown in **Figure 14**.

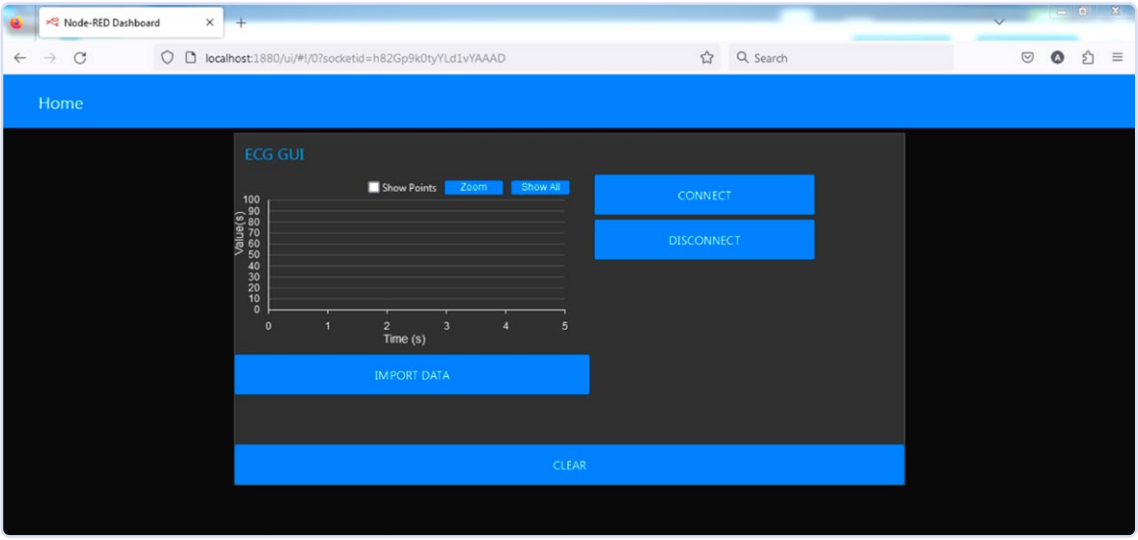
Hardware Setup

Figure 15 shows the wiring of the buzzer, that must be connected to P5 and GND pads. You can also solder it directly onto an Hexabitz Proto module.

You also need to connect the ST-Link (H40Rx Module) to the EXG module using SWD (Serial Wire Debugger), and supply the module with a 3.3 VDC source. The analog EXG module outputs can be checked with any oscilloscope.

Testing the System

The live testing of this system obviously implies the proper placement of the electrodes on different body parts. **Figure 16** shows three different options for the placement of LA, RA and LL electrodes: on



localhost:1880/ui/#!/0?socketid=h82Gp9k0tyYLd1vYAAAD

Figure 14: Simulation of remote access through *http://localhost:1880/*.

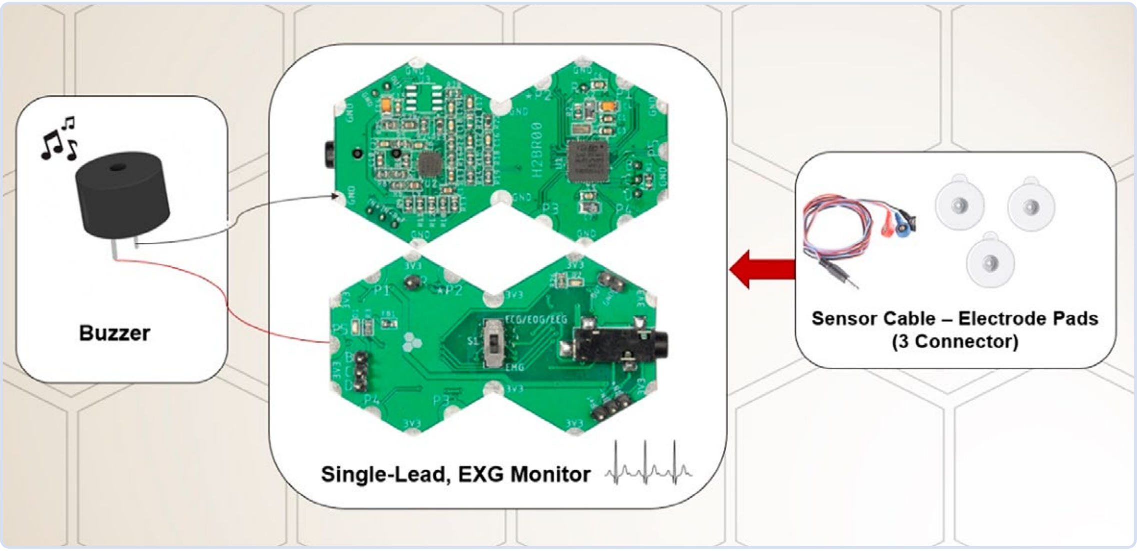


Figure 15: Buzzer connections to the EXG Monitor.

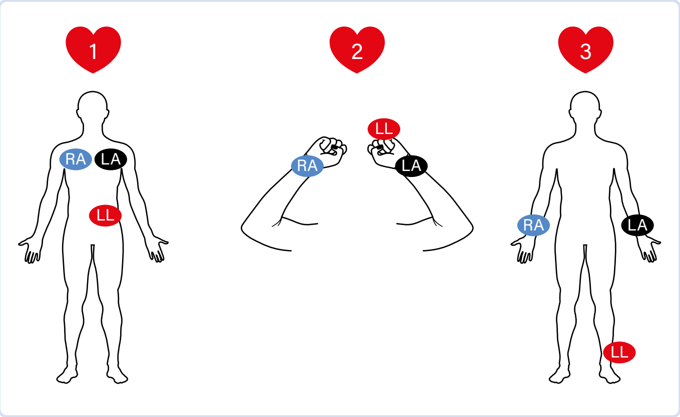


Figure 16: Examples of correct placement for LA, RA and LL electrodes.

chest, wrists and limbs. **Figure 17** illustrates the correct placement of the RA electrode on the right wrist, whilst in the background the actual ECG trace is displayed on the screen. **Figure 18** shows one of the final phases of dashboard testing, with all the involved hardware visible on the right.

General Hints

Building the ECG Graph Monitoring system involves several key steps. First, ensure all components are correctly connected according to the schematic; then, test your system. Performing an overall functionality check is an advisable procedure to assess the correct operation of the whole design. To this extent, you may use a known signal source from a reference ECG generator or, as an alternative, the output of any ECG pulses emulator, like the one described in this Elektor project [12].

Connect the chosen signal generator to the analog input of the ECG monitoring system. Although — on our Hexabitz module — offset and gain are fixed, this check will show us that the system is operating correctly. During testing, monitor the output closely, and compare it with standard ECG readings, to verify its consistency. Regularly check connections and components to maintain system reliability.

Operating the ECG Graph Monitoring system is straightforward. Ensure the electrodes are placed correctly on the subject to obtain clear signals. Practical tips include keeping the system away from electronic interference and ensuring the subject remains still during measurements. My experience with this project has been rewarding, as it provides real-time heart monitoring, which is invaluable for early detection of cardiac issues.

Future Developments

Here are some potential future developments for this project, that could make the system even more versatile and beneficial for the users:

- *Enhanced Data Analysis*, implementing advanced algorithms for more accurate and detailed analysis of ECG data.

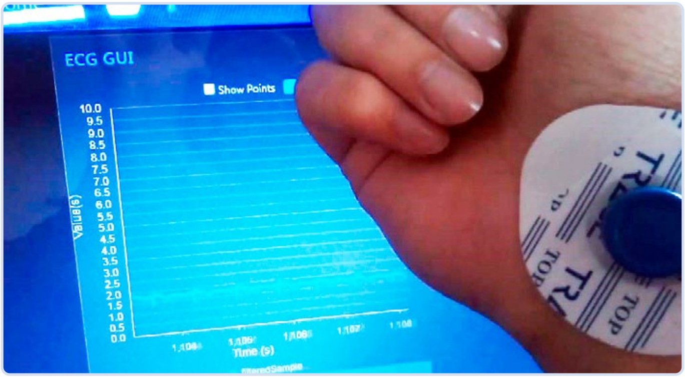


Figure 17: The RA electrode in place, with a "live" signal acquisition visible in the background.

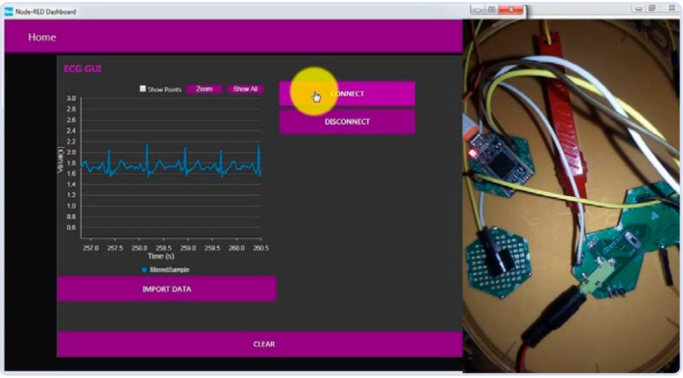


Figure 18: Final phases of dashboard testing. The involved hardware is shown on the right.

- *Wireless Connectivity*, adding Bluetooth or Wi-Fi capabilities for seamless data transfer to mobile devices or cloud storage.
- *Integration with mobile App*, allowing integration with monitoring apps for comprehensive health tracking.
- *Multi-Lead ECG Monitoring*, expanding the system to support multi-lead ECG monitoring for more comprehensive heart health analysis.
- *Portable Device Design*, developing a portable ECG monitoring device with a rechargeable battery. The device will use the LiPo Charger with USB-C for efficient charging.

240142-01

Questions or Comments?

Do you have technical questions or comments about his article? You may contact the author at aulajazmati7@hotmail.com or the editorial team of Elektor at editor@elektor.com.

About the Author

After her bachelor's degree in 2009 and a master's degree in 2015, Aula Jazmati obtained a PhD in Electronic Engineering in 2023. Since 2010, she has been working at the Advanced Electronic Systems Laboratory. In her spare time, she has been volunteering with the translation team at Raspberry Pi since 2016. Additionally, since 2019, Aula has served as an ambassador for the Hexabitz firmware team. You can find more of her projects at *www.hackster.io/aula-jazmati* and *https://github.com/aula9*.

Related Product

- **M. Pakdel, *Advanced Programming with STM32 Microcontrollers* (Elektor, 2020)**
  www.elektor.com/19520

WEB LINKS

[1] Wiki page for Electrocardiography: https://en.wikipedia.org/wiki/Electrocardiography
[2] EXG Monitor Module YT introductory video - Part 1: https://tinyurl.com/yc9hmmpf
[3] EXG Monitor Module YT introductory video - Part 2: https://tinyurl.com/48dx894h
[4] Hexabitz Single-Lead, EXG Monitor Module webpage: https://tinyurl.com/5b6a97dz
[5] BCRobotics sensor cable: https://tinyurl.com/tukpw7vb
[6] Hexabitz Single-Lead, EXG Monitor Module Firmware: https://tinyurl.com/4s9vdj7n
[7] Code for this project (GitHub): https://tinyurl.com/y6sumenz
[8] STM32CubeIDE download: https://www.st.com/en/development-tools/stm32cubeide.html
[9] How-To write code with STM32CubeIDE: https://tinyurl.com/2vujcv29
[10] How-To control ports independently: https://tinyurl.com/rftuharu
[11] Build GUI using STM32CubeMonitor: https://tinyurl.com/2zcfhy5p
[12] J. Holzhauer, "ECG simulator," Elektor 5/2000: https://tinyurl.com/4zaudfsd

The Battle for AI at the Edge

By Stuart Cording (Germany)

The news around AI mainly reports on the achievements of tools like ChatGPT and Midjourney — powerful, cloud-based tools. But there are plenty of other non-cloud applications where even a little intelligence can make a big difference.

Embedded systems have traditionally relied on procedural programming to solve their tasks, especially those constructed for battery operation and using microcontrollers clocked at tens of megahertz and with limited memory. This restricts these devices to tasks that can be implemented by a series of decisions supported by an algorithm such as a fast-Fourier transform or PID control loop. But many tasks are simply a pattern-matching activity. And this is something that AI is really good at.

Applying AI to Everyday Medical Diagnostics

Take, for example, an electrocardiogram (ECG). Used to monitor the activity of the heart, these time-varying signals have an amplitude of around 10 µV to 5 mV and contain frequencies of around 0.05 to 35 Hz [1]. Such tiny signals are the first challenge. The next is that the electrodes aren't permanently fixed in place like a sensor of an industrial system delivering well-defined signals. They are temporarily attached to the human body; even when a health professional performs this, there is wide signal variation and noise. Furthermore, humans move continuously, which causes deviations in the shape of signals.

The typical approach to this signal detection challenge is to build a high-quality analog front end, perform digital filtering, and then apply an ECG algorithm that is robust enough to respond to a wide range of differing and distorted pulse shapes, signal transitions, and amplitudes.

On the software side, ECG samples are collected, and then an algorithm is coded using procedural programming decisions. These choose different detection algorithm options and approaches depending on the distortions detected. This can lead to a long list of difficult-to-manage options.

The alternative approach is to use AI. Because AI algorithms are good at pattern matching, real ECG samples can be provided as training data. From this point, a model can be built that recognizes the six peaks (P, Q, R, S, T, and U), ECG anomalies, and delivers the results typically required by a physician (**Figure 1**). This will range from heart rate and regularity of the beat to deviation in the cardiac axis and other anomalies [2] (**Table 1**).

Such an AI model is also easier to modify when changes are made to the analog front end, different use cases are needed, or newly discovered ECG signal classifications arise through research. And the demand on a microcontroller is not exceptionally high, meaning a decent 32-bit device running at tens of MHz with a few hundred KB of flash and tens of KB SRAM can be used instead of relying on cloud-based AI resources.

Research Into Tiny Machine Learning

As a result, plenty of research is going on into AI at the tiny, non-cloud end of the user spectrum. The primary goals of such research are to support privacy by

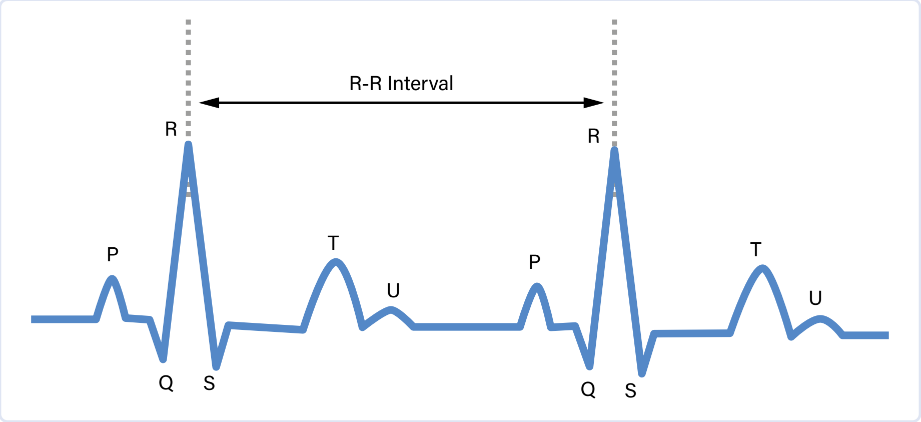


Figure 1: The key features of an electrocardiogram are labeled from P to U. The R-R Interval is the heart rate and normally lies between 0.6 and 1.2 seconds.

Table 1: Selection of ECG features, their typical duration, and how any anomalies may be interpreted. [1]

Feature	Typical Duration	Anomaly Interpretation
R-R Interval	0.6 – 1.2 s	Paroxysmal atrial fibrillation
		Congestive heart failure
P-R Interval	0.12 – 0.2 s	Stroke
QRS Complex	0.06 – 0.1 s	Ventricular enlargement
		Heart failure
		Tachycardia
		Acute Coronary Syndrome
T Wave	0.05 – 0.25 s	Myocardial infarction
		Pulmonary embolism

processing data locally; keep latency low by not communicating with cloud services, which is essential for time-critical systems; reduce communication bandwidth by transmitting results, not the raw sensor data; and often executing such tasks with minimal energy expenditure, an important consideration for battery-powered applications.

Since such embedded systems sit where the sensors are located and often directly control actuators based upon the measurements acquired at the farthest edge of a network in industrial systems, automotive, or Internet of Things (IoT), this field is typically termed Edge AI.

Organizations like the Edge AI Foundation (formerly the tinyML Foundation) organize regular summits that provide researchers and experts the opportunity to share their learnings in this field. They're supported by a range of the industry's key semiconductor vendors and technology suppliers, such as Qualcomm, STMicroelectronics, NXP, and Arm, but also startups like Greenwaves and single-board computing (SBC) suppliers like Arduino.

How Many Did You Want?

Sifting through the presentations from the EMEA Innovation Forum from June 2024, it's clear that the focus is on the sub-field of machine learning, as the name of the foundation infers, which is AI applied to a single and specific task. One paper from the Friedrich-Alexander-Universität [3] explores voice recognition for spoken numbers. This is a task that needs to function locally inside automotive telematics systems or smartwatches. They note that most datasets used for training are monolingual. Datasets are the initial challenge when developing machine learning models since, without some representative data in enough quantity, it is difficult to train the model to attain the desired accuracy. They also explain that simply combining the datasets isn't enough to produce a multilingual number recognition model.

Further issues arise because the time needed to express a number varies considerably between languages. Units and tens are short, with Mandarin being the shortest. The German numbers above twenty take the longest (except for the French in the 70s and 90s), needing as much as twice as long as Mandarin to express a value.

Their research delivers an artificially generated, multilingual spoken number dataset from four speakers consisting of 12,800 samples called SpokeN-100 [4]. They also propose using the EfficientNet convolutional network that has a scaling architecture that, essentially, scales uniformly with the size of the input data (**Figure 2**). This has led to work on their EvoNAS algorithm [5] (*End-to-end eVOlutionary Neural Architecture Search for microcontroller units*) running on an nRF52840 Arm Cortex-M4 device from Nordic Semiconductor. Current results indicate an inference (word recognition) time of 200 ms, a memory footprint of ~800 kB at an energy consumption of 2 mJ.

AI Made for MCUs

Most AI ideas start on a PC, trained in the cloud, with significant amounts of training data. Once they've achieved a certain level of maturity, the discussion of moving to an embedded platform starts. This can be a significant shock to development teams. Microcontrollers don't have the mature development environments used in MLOps (machine learning operations), as they are laid out to debug at register level and require a lot of manual peripheral configuration. And there typically isn't any support for Python, relying on C/C++ as the programming language.

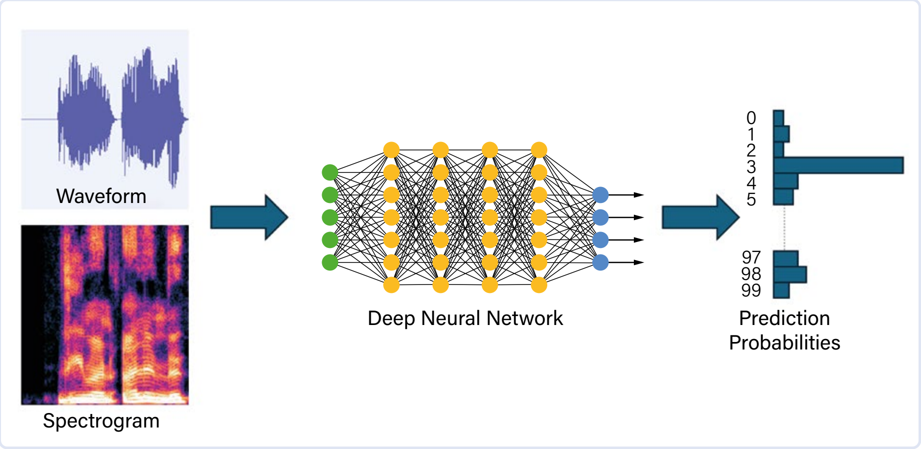


Figure 2: Speech algorithms typically turn audio into a visual image before presenting it to a deep neural network for analysis.

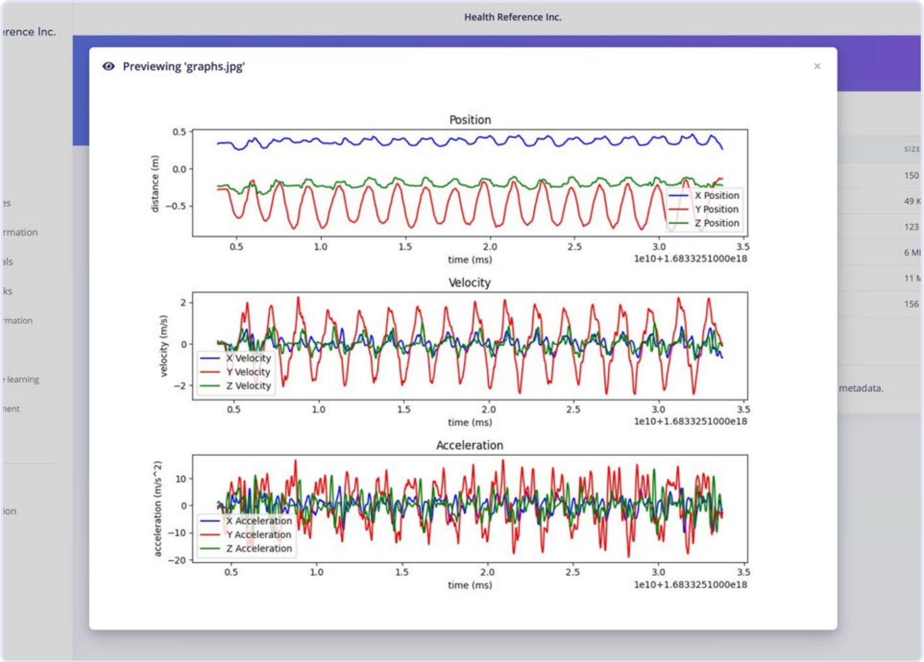


Figure 3: The Edge Impulse dashboard enables the ingestion of sensor data directly from your microcontroller.

Tackling these challenges is Edge Impulse, another sponsor of the Edge AI Foundation and one of the most prominent players in the Edge AI space. Their platform helps engineers from all walks of life, from beginners to machine learning practitioners, streamline data acquisition, preprocessing, and model optimization for embedded devices.

According to Alessandro Grande, their Head of Product, "the shift in mindset from cloud to edge development" is most developers' biggest challenge. "Instead of seemingly endless cloud resources, you're constrained by limited device memory, tight power budgets, and the need for real-time responsiveness. It's a new paradigm with unique challenges demanding a different approach that requires embedded expertise."

Grande notes that there are major challenges around "collecting and preparing suitable data from edge devices for ML development. This is particularly difficult when dealing with sensor data, as there is often a lack of publicly available, labeled datasets for modalities like accelerometer, sound, or vibration."

Getting Your Own Labeled Data

To tackle this, an ingestion service is provided. Even microcontrollers that are connected to your PC can directly upload raw data from attached sensors, providing the starting point for model training (**Figure 3**). Data engineering is also supported, helping to get your data classified if this step hasn't already been done. GPT-4o can be integrated into a labeling block, allowing acquired data, such as images or video, to be given labels. Once complete, the classifications can be reviewed for accuracy, deleting outliers that would reduce the accuracy of the resultant model.

The simplicity with which complex vision AI tasks can be trained and trialed is impressive. In one demonstration, Jan Jongboom, CTO and co-founder, uploads a video containing children's toys lying on the floor in a typical family home [6]. A training dataset is created using a simple GPT-4o prompt to classify video frames where toys are present, and an NVIDIA TAO image classification model is trained. The resultant model can be tested in a smartphone's web browser before being deployed on a Raspberry Pi. Further optimization enables the model to be shrunk, fitting into less than 250 kB of RAM and running at 10 frames-per-second on an Arm Cortex-M7-powered Nicla Vision [7] microcontroller board.

Hardware for Edge AI

Of course, the semiconductor industry is trying to ensure microcontrollers adapt to provide the processing performance required for Edge AI applications while retaining their reputation for low power. The Nicla Vision uses a standard STM32H747AI MCU from STMicroelectronics. In addition to the Cortex-M7 core, it also includes a Cortex-M4, which would probably be dedicated to real-time control tasks in this type of application. This leaves the Cortex-M7 core available for performance tasks, like running miniaturized machine learning models.

The PC industry quickly learned that more cores was the way to go once clock frequencies alone couldn't deliver more performance. So, is this the next development phase for MCUs?

Well, more cores alone in embedded systems don't help unless the power consumption issue is addressed in parallel. This is the approach taken by Greenwaves, supplier of the GAP8 and GAP9 processors. The devices target low-power, battery-operated applications like hearing aids that want to leverage neural networks to deliver next-generation electronic products.

The GAP9 includes the hardware interfaces needed for audio (I^2S) and video (MIPI and CPI) along with the traditional array of standard microcontroller peripherals. A Smart Filter Unit (SFU) provides low-latency, low-power audio stream processing and filtering (**Figure 4**). But it is the compute performance that is fascinating, delivered by ten RISC-V cores and an AI accelerator interfaced with a fabric controller that maintains the balance between power consumption, latency, flexibility, and ease of use.

Dr. Tinoosh Mohsenin provided the EMEA Innovation Forum participants insights on the capabilities of Greenwaves' processors when applied to Vision Language Models (VLMs). Such generative models take image and text inputs and generate text outputs. In cloud implementations, such AI models work with billions of parameters while examining images of disaster zones provided by satellite imagery.

The team's research explores whether a camera-equipped drone kitted out with a compressed VLM could help rescuers answer questions such as "Is this area flooded," "How many buildings are there,"

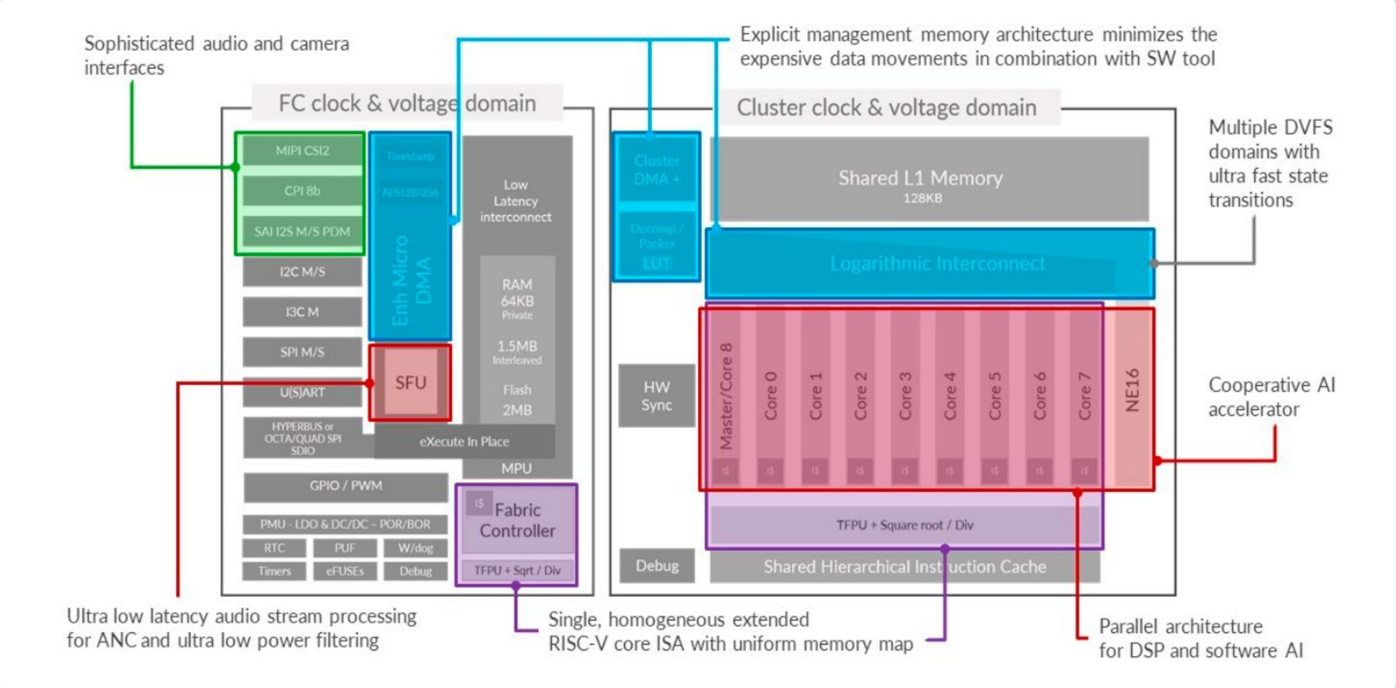

Sophisticated audio and camera interfaces
Explicit management memory architecture minimizes the expensive data movements in combination with SW tool
FC clock & voltage domain
MIPI CSI2
CPI 8b
SAI I2S M/S PDM
I2C M/S
I3C M
SPI M/S
U(S)ART
HYPERBUS or OCTA/QUAD SPI SDIO
Timestamp
AES128/256
Enh Micro DMA
SFU
Low Latency interconnect
RAM 64KB Private
1.5MB Interleaved
Flash 2MB
eXecute In Place
MPU
GPIO / PWM
PMU - LDO & DC/DC – POR/BOR
RTC
PUF
W/dog
Timers
eFUSEs
Debug
I$
Fabric Controller
TFPU + Sqrt / Div
Cluster clock & voltage domain
Cluster DMA +
Decomp / Packer
LUT
Shared L1 Memory
128KB
Logarithmic Interconnect
HW Sync
Master/Core 8
Core 0
Core 1
Core 2
Core 3
Core 4
Core 5
Core 6
Core 7
I$
NE16
TFPU + Square root / Div
Debug
Shared Hierarchical Instruction Cache
Multiple DVFS domains with ultra fast state transitions
Cooperative AI accelerator
Ultra low latency audio stream processing for ANC and ultra low power filtering
Single, homogeneous extended RISC-V core ISA with uniform memory map
Parallel architecture for DSP and software AI

Figure 4: The Greenwaves GAP9 processor targets low-power applications in audio and video and features a total of 10 RISC-V cores. (Source: Greenwaves)

and “Is this road accessible?” (**Figure 5**). A baseline AI for this task requires over 81 MB of memory for the model and delivers an accuracy of 81%. Their tiny version of this model for a microcontroller fits in just 339 kB of memory, with a loss in accuracy of just 1.5%.

This particular VLM leverages the capability of the GAP8 processor utilizing the cache memory for double-buffering, DRAM attached via a DMA (direct memory access), and the PULP-NN multicore computing neural network software library [8]. And the results are impressive. When the VLM is executed on an NVIDIA Jetson TX2 with its 256 Pascal GPUs, dual-core 64-bit Denver 2 cores, and quad-core Arm Cortex-A57, energy consumption was 5.6 J with an inference latency of 213 ms. The GAP8 achieved a latency of 56 ms while requiring just 200 mJ.

Such developments open up a wealth of opportunities in processing and power-constrained systems like MCUs.

Improving Getting Started

Now that you know that a billion-parameter AI model can be squished into a reasonably powerful Arduino, you’ll be eager to get started. But clearly, the path from one to the other requires a heap of skills, from programming to data analysis.

To make things easier, work is being undertaken on Automated Machine Learning, or AutoML [9]. This provides processes and methods that make machine learning more accessible to non-experts, improves algorithm efficiency, and supports researchers. The reach of activities is enormous and has led to tools like Auto-PyTorch that optimize AI network architectures and training hyperparameters.

AutoML also appears in Microchip’s MPLAB Machine Learning Development Suite [10], helping developers tune algorithms to reduce resource use while retaining the achieved model accuracy.

How Will Edge AI Change MCU Designs?

Microchip’s solution isn’t just for its 32-bit devices. The 16-bit and 8-bit MCUs can also be selected as targets for a machine-learning application. However, as we have seen in the examples provided, hundreds of kilobytes of memory are needed for complex AI models, meaning the small memory devices are only suited to the most simple pattern-matching type tasks.

What this means is that, as Edge AI grows in importance, we’ll see MCUs coming onto

Figure 5: A microcontroller connected to a camera fitted to a drone can perform basic real-time image analysis of a disaster area, answering human-generated questions. (Source: Adobe Stock/MariKa, AI generated)

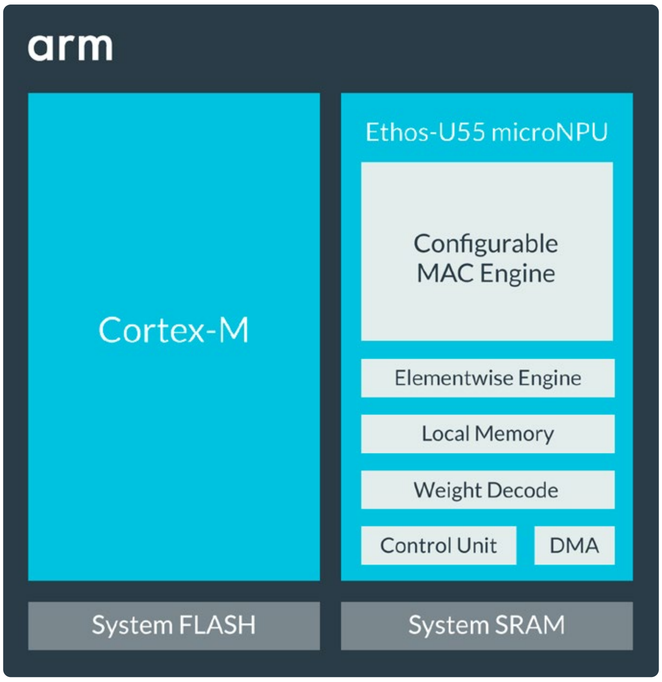


Figure 6: Arm's Ethos-U55 is a Neural Processing Unit (NPU) designed to operate alongside its range of Cortex-M processors typically found in microcontrollers. (Source: Arm)

the market with more flash and RAM, and potentially interfaces for external memory that provide the space for applications where integrating internal memory is no longer business viable.

Heterogeneous multicore processing, which uses several cores of different architectures, will also be more common. Both Arm and RISC-V are the key technology providers here. However, there is growing interest in AI accelerators, processors with an architecture well suited to the convolution and matrix calculations used by AI algorithms. Arm already offers its Ethos-U55 [11] Neural Processing Unit (NPU), which is available in the Alif Semiconductor Ensemble E5 [12] and E7 [13] families alongside Cortex-M55 and Cortex-M32 processors (**Figure 6**).

Of course, as always, engineers will ultimately decide which architectures and approaches are preferred. Referring back to Alessandro Grande from Edge Impulse: "Ultimately, the optimal choice between a dedicated accelerator, a basic core, or a hybrid solution will depend on the specific needs of each application, balancing performance, power efficiency, and cost considerations." ⏮

240687-01

About the Author

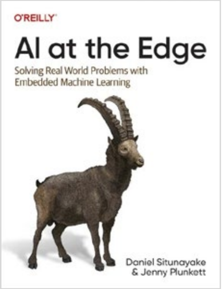
Stuart Cording is an engineer and journalist with more than 25 years of experience in the electronics industry. He specializes in video content and is focused on technical deep-dives and insight. This makes him particularly interested in the technology itself, how it fits into end applications, and predictions on future advancements. You can find many of his recent Elektor articles at www.elektormagazine.com/cording.

Questions or Comments?

If you have questions about this article, feel free to email the Elektor editorial team at editor@elektor.com.

Related Product

- **D. Situnayake and J. Plunket, *AI at the Edge* (O'Reilly)**
  www.elektor.com/20465


O'REILLY
AI at the Edge
Solving Real World Problems with Embedded Machine Learning
Daniel Situnayake & Jenny Plunkett

WEB LINKS

[1] Xie L, Li Z, Zhou Y, He Y, Zhu J., "Computational Diagnostic Techniques for Electrocardiogram Signal Analysis," National Library of Medicine, Nov 2020: https://pmc.ncbi.nlm.nih.gov/articles/PMC7664289/

[2] Dr M. Jackson, "How to Read an ECG," Geeky Medics, Nov 2024: https://geekymedics.com/how-to-read-an-ecg/

[3] R. Groh, N. Goes, A. M. Kist, "SpokeN-100," Presentation: https://cms.tinyml.org/wp-content/uploads/summit2024/Rene-Groh.pdf

[4] R. Groh, N. Goes, A. M. Kist, "SpokeN-100," Zenodo, March 2024: https://zenodo.org/records/10810044

[5] R. Groh, A. M. Kist, "End-to-end evolutionary neural architecture search for microcontroller units," tinyML: https://tinyurl.com/EvoNAS-algorithm

[6] Edge Impulse, "Label image data using GPT-4o": https://tinyurl.com/label-image-data-gpt-4o

[7] Arduino Nicla Vision: https://store.arduino.cc/products/nicla-vision

[8] PULP-NN library [GitHub]: https://github.com/pulp-platform/pulp-nn

[9] AutoML: https://www.automl.org/automl/

[10] Microchip, "MPLAB Machine Learning Development Suite": https://tinyurl.com/MPLAB-Microchip

[11] Ethos-U55: https://developer.arm.com/Processors/Ethos-U55

[12] Ensemble E5 Series from Alif Semiconductor: https://alifsemi.com/ensemble-e5-series/

[13] Ensemble E7 Series from Alif Semiconductor: https://alifsemi.com/ensemble-e7-series/

HaLow Hits Record 16-km Wi-Fi Distance at 900 MHz

By Nick Flaherty (eeNews Europe)

Morse Micro has achieved a record 16-km distance for a Wi-Fi HaLow link at 900 MHz.

The tests of Wi-Fi HaLow at the rural Joshua Tree National Park in the US covered 16 km [1], up from the 3 km shown in January in an urban environment. HaLow is a variant of the Wi-Fi standard designed for lower data rates and frequencies for the Internet of Things and the Morse tests achieved a data rate of 2 Mbit/s.

The tests used an evaluation kit as an access point (AP) at the edge of a quiet rural valley. The off-the-shelf MM6108-EKH01 evaluation kit is based around a Raspberry Pi 4 with the Morse MM6108-MF08651 Wi-Fi HaLow reference module.

The evaluation kit outputs 21 dBm (125 mW) through a standard 1 dBi low-gain dipole antenna, resulting in a total radiated power of 22 dBm without tweaking the 802.11ah parameters to increase the range or using high-gain directional antennas.

The Morse Micro [2] chips adhere to the 802.11ah standard, which specifies a slot time of 52 µs. Morse Micro's implementation allows for a maximum time of flight of 53 µs to allow for slight variations between devices. This results in a theoretical maximum range of 15.9 km (approximately 10 miles).

In theory, Morse expected a link at a sensitivity of -95 dBm, which provides a throughput of 4.5 Mbit/s or a UDP MAC throughput of 4 Mbit/s. The tests at the Joshua Tree National Park achieved a stable connection of 2 Mbit/s UDP throughput at 15.9 km.

240734-01

Editor's Note: Nick Flaherty Joosting first reported on this in eeNews Europe, a publication in the Elektor network. www.eenewseurope.com/en/domain/eenews-embedded/

WEB LINKS

[1] Wi-Fi HaLow delivers throughput at 16 km range: https://youtu.be/fBMgZah2Z7g
[2] Morse Micro: https://morsemicro.com

Source: Adobe Stock

First CHERI RISC-V Embedded Chip and Early Access Programme

By Nick Flaherty (eeNews Europe)

SCI Semiconductor in Cambridge has developed the first CHERI-enabled family of chips for embedded designs. The ICENI microcontroller chips are based on the RISC-V RV32E architecture and use the Microsoft CHERIoT-Ibex processor core. The chips are built by GlobalFoundries and are aimed at a wide variety of applications, from simple microcontrollers to advanced microprocessor applications, with availability in 2025.

The first device is a single-core microcontroller but the company is planning multicore and has designed an extended temperature version with a view to the automotive market, Haydn Povey, Chief Executive, SCI Semiconductor tells eeNews Europe.

The CHERIoT technology uses a hardware architecture that avoids memory safety issues. As well as developing its own devices, the company is supplying IP to a tier-one supplier for a system-on-chip (SoC) design.

"We are riding both horses at the moment as the primary goal is a device maker as the biggest challenge is being able to buy the technology for automotive, smart energy, and telecoms. We have a design that is qualified for extended temperature operation and we have partners looking at after-market telematics applications. Moving to autonomous vehicles [1] means more communications and that increases the attack surfaces," he said.

Over 70% of modern software vulnerabilities are based on these memory safety software issues, creating the explosion of cyber-security attacks taking advantage of memory misconfiguration and software-reuse issues to rapidly escalate attacks and are endemic in modern code bases globally.

The ICENI family of microprocessors from SCI Semiconductor use the 32-bit CHERIoT-Ibex RISC-V core for the world's first high-integrity intrinsically-Memory Safe devices. The device's architecture, integrating fine-grained hardware-enforced compartmentalisation, supports complete spatial and temporal memory safety.

The CHERIoT technology uses a hardware architecture that avoids memory safety issues.

This supports existing code with a compilation without rewriting the C code, and has an overhead of 1 to 3%, says David Chisnall, co-founder of SCI Semiconductor and Director of Systems Architecture.

"Everything on the market is embarrassing," said Chisnall at the High Integrity Systems Conference yesterday. "You are trying to replace appliances that people expect to last ten years but companies think three years is long-term support and they are building these in an incredibly cost-sensitive environment so the hardware is not able to support the features that we rely on for security. So we looked at designing the hardware and software stack from the ground up with all the things we have learned since the 1960s."

"We have a production quality core, and an open source core is great with an FGPA dev system but if you actually want to deploy this in production you just want to buy a chip and that's what we are doing at SCI Semiconductor," he added.

"The ICENI family marks the start of a new epoch of secured devices, secured applications, and secured society," said Povey. "The modern cyber-security industry is focused on treating the technological symptoms of poor hardware and software architecture, with CHERI and the new ICENI device family, we can now finally start to treat the disease, enabling rapid code reuse without importing vulnerabilities, accelerating development and reducing update requirements."

The ICENI family of devices uses the open-source CHERIoT Platform [2] originally developed by Microsoft Research and now maintained as a cross-vendor open-source project, with Microsoft and SCI Semiconductor as co-owners of the repository, along with contributions from Google and rapidly evolving host of ecosystem partners including open source hardware developer lowRISC.

"Microsoft is pleased to see that the open-source CHERIoT Ibex core is being used by SCI Semiconductor in an upcoming silicon product. We believe that CHERI is a promising technology that can be used to enhance computer security, and we are happy to see it making its way into production silicon. This is one of the main reasons why Microsoft developed and open-sourced the CHERIoT Ibex core," said David Weston, VP of Enterprise and OS Security at Microsoft.

SCI has also launched an Early Access Program, which will enable selected customers and partners early access to silicon devices, alongside advanced development systems. These systems use the lowRISC FPGA Sonata platform [3] enabled by the UKRI Digital Security by Design programme.

Selected partners can start development immediately on the EAC program with rapid portability to silicon devices in 2025, accelerating to transition to next-generation Memory Safe systems.

SCI Semiconductors has also signed a strategic distribution deal with EPS Global, which also handles IC Programming and Embedded Security for automotive Tier One suppliers and contract manufacturers.

"SCI's value proposition is compelling, and they're ahead of the market in terms of delivery," said Colin Lynch, CEO at EPS Global. "They are providing key security chips that meet customers' needs for secure-by-design solutions. The key markets are infrastructure, defense, automotive, and aerospace. EPS Global can add significant value in this space through our customer engagement, distribution expertise, and secure provisioning capabilities."

SCI Semiconductors is a founder member of the CHERI Alliance, and was formed to lead the commercialisation of CHERI technologies, which took on the SDK developed by Codasip in Germany for CHERI RISC-V cores for SoCs and automotive designs. ◀

240727-01

Editor's Note: Nick Flaherty first reported on this in eeNews Europe, a publication in the Elektor network. www.eenewseurope.com/en/domain/eenews-embedded/

WEB LINKS

[1] N. Flaherty, "£2.2m for CHERI automotive, embedded security projects," eeNews, 2023: https://www.eenewseurope.com/en/2-2m-for-cheri-automotive-embedded-security-projects/
[2] CHERIoT Platform: https://cheriot.org
[3] N. Flaherty, "CHERI moves into RISC-V, x86 for embedded," eeNews, 2024: https://www.eenewseurope.com/en/cheri-moves-into-risc-v-x86-for-embedded/

Source: Adobe Stock

Third-Generation Wildfire Detection Uses Satellite Links

By Nick Flaherty (eeNews Europe)

Dryad Networks in Germany has launched its third-generation Silvanet border and mesh gateways for early wildfire detection and forest management via satellite links.

The Silvanet gateways developed by Dryad [1] include direct-to-satellite connectivity via EchoStar for the first time and extended LoRa radio range, as discussed in *eeNews Europe* in June [2]. A new mounting bracket and ruggedized, weather-resistant design with an IP67 waterproof rating eases forest deployment.

The satellite connectivity in the third-generation gateway reduces reliance on terrestrial networks, offering real-time monitoring and bi-directional communication, including remote configuration and firmware updates. By including direct-to-satellite connectivity in mesh gateways in addition to border gateways, Dryad adds an extra layer of redundancy resulting in unparalleled network reliability, enabling fallback to satellite connectivity in case of a loss of terrestrial connectivity.

Other enhancements include improved reliability and extended network coverage, making the third-generation gateways ideal for large-scale deployments in challenging environments with maintenance-free operation of 10 to 15 years.

The Silvanet Border Gateway is placed at the border of a target forest area, and the Silvanet Mesh Gateway extends network coverage into the depth of the forest using a unique multi-hop mesh networking architecture. The gateways are the core of Dryad Networks' Silvanet Software Suite for wildfire detection and forest management, providing a robust communications network for the Silvanet Wildfire Sensors and additional sensors currently under development.

The built-in satellite communication in North America and Europe comes from EchoStar and also provides redundancy to 4G network communication, if gateways cannot connect over 4G. A built-in embedded

SIM card (NB-IOT / LTE-M) with multi-IMSI provides global out-of-the-box mobile network connectivity and there is an easily accessible, user-serviceable SIM card slot enables the replacement of the built-in embedded SIM card with a local SIM for optimized local connectivity.

Extended LoRaWAN (long-range wide area network) range enables the third-generation gateways to communicate over longer distances, up to 10 km (6 miles), reducing the number of gateways required per deployment.

The third-generation border gateways include two solar panels, doubling the energy available and adding thirty percent more energy storage than the second-generation gateways: An increased number of supercapacitors in the third-generation gateways provides more energy storage in the solar-powered gateways and helps ensure continual operation, especially in difficult lighting conditions and shaded environments.

A fully integrated PCB (printed circuit board) antenna design with no external antennas improves reliability and eliminates a weak point of other gateway designs and there is a single rigid-flex PCB for maximum durability: The advanced rigid-flex PCB design eliminates any internal connectors and cables, further improving reliability and longevity of the devices in challenging environmental conditions.

"At Dryad Networks, our mission is to protect the world's forests by developing innovative and scalable solutions for wildfire detection and forest management. Our third-generation Silvanet border and mesh gateways represent a significant leap forward in achieving this goal, offering unmatched reliability, network performance, coverage, and ease of use. The upgrades reflect our commitment to continuously advancing our technology to meet the evolving needs of our partners and end users," said Carsten Brinkschulte, CEO of Dryad Networks.

"The introduction of direct-to-satellite connectivity and extended LoRa range in our new gateways marks a pivotal advancement in wildfire detection technology. These features ensure that even the most remote and challenging environments can be monitored in real time, without reliance on terrestrial networks. This technological breakthrough sets a new standard in the industry," said Dryad Networks Chief Technology Officer Pedro Silva.

The third-generation gateways can be installed 50% faster than second-generation gateways, due to a new mounting bracket and locking system which is used to attach the gateways to trees or poles. This solution is substantially faster and easier to install and improves radio connectivity.

Built-in near-field communication (NFC) interface provides local configuration and testing. This is especially helpful in areas where there is no communication infrastructure that may prevent or impede installation or maintenance communication between a gateway and the Silvanet Cloud Platform. Technicians can access the gateway configuration and testing controls at the gateway site with an NFC-enabled device such as a smartphone, with no need to scan QR codes during installation and maintenance. ⏮

240729-01

Editor's Note: Nick Flaherty first reported on this in eeNews Europe, a publication in the Elektor network. www.eenewseurope.com/en/domain/eenews-embedded/

❝

The introduction of direct-to-satellite connectivity and extended LoRa range in our new gateways marks a pivotal advancement in wildfire detection technology.

WEB LINKS

[1] Dryad Networks: https://dryad.net

[2] "Satellite connection for solar-powered forest wildfire sensor network," eeNews June 2024: https://eenewseurope.com/en/satellite-connection-for-solar-powered-forest-wildfire-sensor-network

developer's zone

Tips & Tricks, Best Practices and Other Useful Information

From Life's Experience

Choice Overload

Figure 1: Plenty... (Source: Adobe Stock / nonnie192).

By Ilse Joostens (Belgium)

We live in interesting times, and I am not talking about geopolitical tensions in the world or crises, but the fact that we live in an era of unprecedented technological growth and societal social change. Our poor brains get a huge amount of information and stimuli to process every day, and it gets a little more every year. A person would get stressed for less. Even to make trivial everyday choices these days, you have to slog through such an abundance of choices and information every time that you end up with stress to making a decision [1].

In my local supermarket, you will find no less than 14 different types of table salt. Trivial as it gets, I almost dare not to mention the number of different types of butter and margarine (**Figure 1**). Even worse with all the apps that every supermarket offers these days, discounts and other promotions, some time pressure, and you'll feel like your brain is about to burst out of your head while shopping.

TUN, TUP and DUS

However, we do not become happier from these endless choices. We'd love to try everything, but we can't, and out of helplessness and the flood of information we have to wade through, we don't really know what to choose anymore. The result is an unfulfilled feeling, the gnawing subcutaneous fear of having overlooked something important, FOMO, an acronym for *Fear of Missing Out*.

And when we finally make a choice, doubt strikes us mercilessly. Did we make the right choice, or were there better options? Regret over our choice soon lurks, and in time we begin to search everywhere for the best option compulsively. There is an acronym for this too, FOBO or *Fear of Better Options*. Actually, we all unconsciously become "The very hungry caterpillar" [2], we are never satisfied and constantly looking for more and better, a time-consuming activity that gives a permanently unsatisfied feeling.

Enough whining; after all, this is not a psychology magazine. Recently, I was going through the fifth edition of the 1988 book *300 Schakelingen* (*300 Circuits*, in Dutch).

Source: Adobe Stock / Ivan.

Source: Adobe Stock / Avi.

Many designs in the book can be rebuilt with standard components that everyone still has lying around somewhere, called TUN (Transistor Universal NPN), TUP (Transistor Universal PNP) and DUS (Diode Universal Silicon) in the book. The huge difference with ordering online from a major electronics distributor could not be greater. For something as simple as an E12 series resistor, you already have dozens to hundreds of options per value and a choice of multiple manufacturers. For single N-channel MOSFETs, it's already over 10,000 references, albeit in various packages and with overlapping type numbers from different semiconductor manufacturers. In any case, there are still countless references, and the fact that new and better types are constantly coming onto the market makes the list even longer. Fortunately, parametric searching usually allows you to reduce the number of types to a more manageable list [3].

Even worse is when a part is out of stock, and you have to find an alternative part or also consider non-technical matters such as price, tiered pricing, availability, discount and other promotions, backorders, commercial sensitivities or even aesthetic considerations. Then you have already passed an entire order through the website and not kept track of time. Suddenly, you have only 10 minutes left to complete your order for delivery the next business day and need to order some more ICs. It turns out that the IC you typically order is out of stock, but you find that the IC in question, albeit with a different suffix after the type number and a different price, is still available. The housing is the same, but what on earth is the difference? Then quickly click open the datasheet and nervously start reading diagonally as the clock ticks relentlessly on (**Figure 2**). In the absence of information, you order just in time, stressed out. Afterwards, according to another, more general document from the manufacturer, the difference turns out to be something obscure such as packaging or the coating of the IC's pins.

Even equivalent components with varying prices always make me despair. The differences can be great, and then you wonder if the more expensive ones are better or if you are paying for a brand name. And what about the so-called "white products" of electronics? So the house brands of certain distributors are reviled by some or praised by others. Good or not, who's to say?

Brand-New Components

With each new version of a programming language, programmers tend to use the new syntax and language elements immediately within their projects. Similarly, I am tempted every time to use the latest components because it not only gives the impression that I am keeping up with my field, but also because I just think it's cool. Until I bang my head against the wall, of course.

For example, in a circuit, I needed multiple current sources, the type you can discretely build with a transistor, some resistors, and a few diodes. Something like that also exists in an integrated SOT353 package [4], so I selected that. During the global shortages of parts some time ago, this component was no longer available and also not replaceable with a pin-to-pin equivalent. Then I had to modify the design and just build everything with discrete components. The same with a modern DC/DC step-down module in an LGA package barely nine by nine millimeters and less than three millimeters high. When they were finally available again, the manufacturer's representative wanted to point me in the direction of the successor. Things can move that fast, and perhaps the slightly older but familiar components are not so bad after all. ⏮

Translated by Hans Adams — 240710-01

Figure 2: Ordering stress... (Source: Adobe Stock / Sergey Nivers).

WEB LINKS

[1] insideBE: Choice Overload – How Having Too Many Options Can Shut Down Your Brain: https://insidebe.com/articles/choice-overload/

[2] Wikipedia: The Very Hungry Caterpillar: https://en.wikipedia.org/wiki/The_Very_Hungry_Caterpillar

[3] Choosing MOSFETs / So Many Options! – Circuit Tips and Tricks (YouTube): https://youtu.be/f9Mgcf6EcTg

[4] PSSI2021SAY Constant current source in SOT353 package: https://www.nexperia.com/product/PSSI2021SAY

Starting Out in Electronics...

...Continues Filtering and Controls Tone

By Eric Bogers (Elektor)

In the previous installment, we overwhelmed you with a lot of formulas for calculating the various unity-gain filters. However, that is by far not the worst of it – after all, calculating machines were invented exactly for this purpose. In order to obtain the desired filter characteristic, resistor or capacitor values with "awkward" values are often required. This brings into doubt whether the theoretical performance of the filter can actually be realized. In this installment, we will try to do something about that.

Filters With More Than Unity Gain

The circuits that we discussed in the previous installment [1] all share the significant disadvantage that they require either resistors (in the high-pass filters) or capacitors (low-pass filters) that have widely varying and, additionally, unusual values. In the typical cross-over filters that are used in stage sound installations — read: fourth-order Linkwitz-Riley filters — the ratio of the component values is one to two. Although this double value is easily obtained with components connected in either series or parallel it is nevertheless inconvenient because precision capacitors are expensive, and therefore we would like to use as few of them as possible.

It becomes even more of a challenge when the cross-over frequency has to be adjustable: a further complication is the stereo potentiometer, which can be very difficult to obtain. To avoid these problems it is possible to use a filter where all the frequency-determining components have the same value — see **Figure 1**.

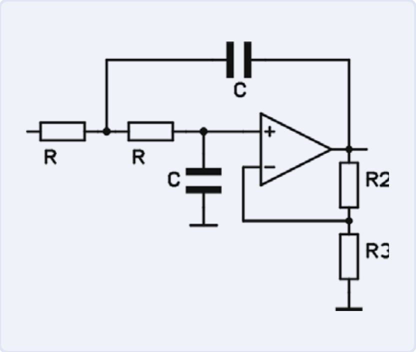


Figure 1: Low-pass filter with equal resistors and capacitors.

With R2 and R3 the gain of the opamp is set to a value greater than one. With this design all popular filter characteristics can be realized. If the overall gain is greater than 3, then the circuit will become an oscillator. Even with a gain between two and three the circuit already has a tendency to overshoot and exhibit underdamped behavior.

The gain of an operational amplifier follows the by now familiar formula for the non-inverting amplifier:

$$V = \frac{R_2}{R_3} + 1$$

For the different filter characteristics, the factors and values from **Table 1** need to be used.

Table 1: Gain factors/resistor values.

Filter characteristic	Gain	R2 (R3 = 10 kΩ)
Linkwitz-Riley	1	0
Bessel	1.268	2.68 kΩ
Butterworth	1.586	5.86 kΩ
Chebychev 3 dB	2.234	12.34 kΩ
Oscillator	3	20 kΩ

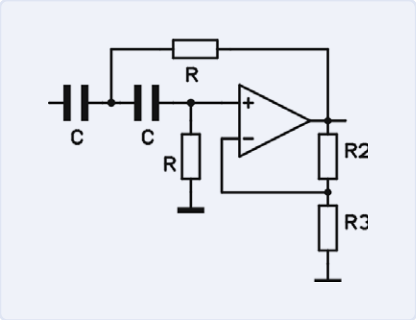


Figure 2: High-pass filter with equal capacitors and resistors.

In the final column you can read the value of R2 that you have to select when R3 has a value of 10 kΩ.

A high-pass filter looks exactly the same — except all the resistors and capacitors have swapped places (**Figure 2**). **Figure 3** shows the characteristics of the four filter types mentioned in the table. **Figure 4** shows the same characteristics, but here we have made the different gains equal by using a voltage divider at the input.

The formulas below only apply to Butterworth (–3 dB) and Linkwitz-Riley (–6 dB) filters:

$$f = \frac{1}{2 \cdot \pi \cdot R \cdot C}$$

$$R = \frac{1}{2 \cdot \pi \cdot f \cdot C}$$

$$C = \frac{1}{2 \cdot \pi \cdot f \cdot R}$$

To make a 24 dB/octave Linkwitz-Riley filter, we can simply connect two second-order Butterworth filters in series.

From the frequency characteristics of Figure 3 and Figure 4, it is very clear that the gain of the Chebychev filter around the transition region is greater than the gain in the pass-band. The unavoidable consequence is that the filter exhibits ringing behavior, as is shown in the waterfall chart in **Figure 5**. By the way, a waterfall chart is simply a series of frequency charts that show the behavior of the filter as a function of time; not that you need to remember that. Along the horizontal axis we see as usual the frequency, along the vertical axis the attenuation and diagonally on the right is the time, increasing towards the front. This goes hand in hand with a deterioration of the impulse behavior. This is why Chebychev filters are almost never used in audio applications.

Band-Pass Filters

Band-pass filters are generally made by connecting a high-pass filter and a low-pass filter (with the appropriate transition frequencies) in series. With these it is perfectly okay to combine filters that have different orders or are not of the same type.

Tone Control

We usually speak of a tone control when there are no more than four controls (and therefore filters) available; with five or more filters we speak of an equalizer. However, the basic operating principle is the same for both. We make a distinction between nonparametric, semi-parametric and parametric filters.

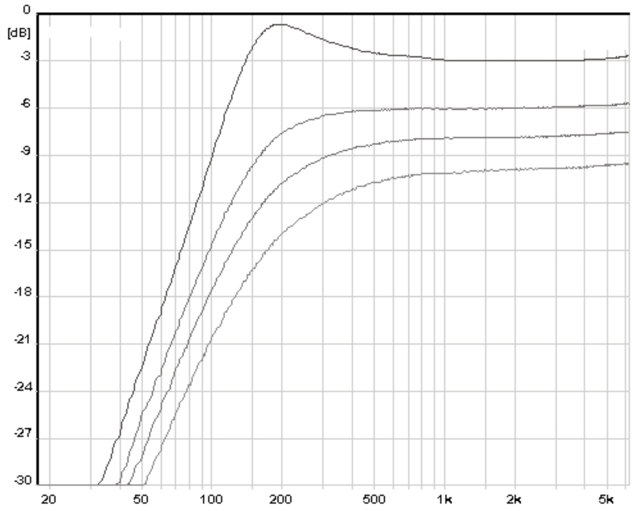


Figure 3: Frequency characteristics.

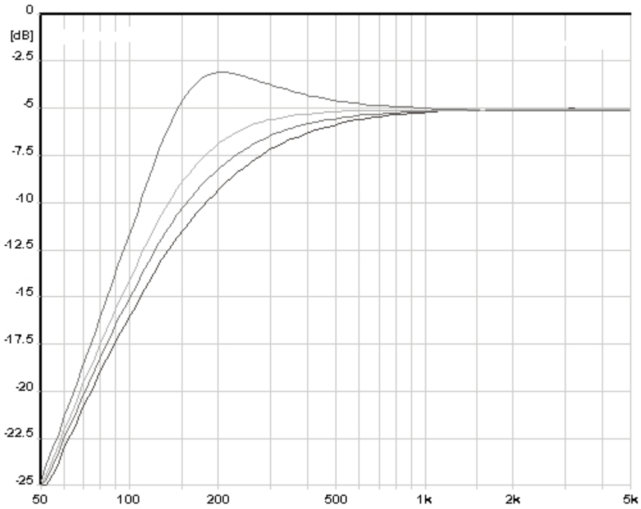


Figure 4: The same frequency characteristics, but now with equal gain in the pass band.

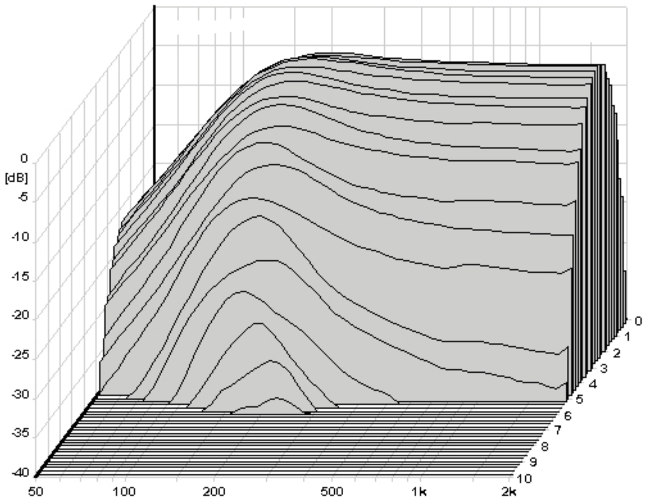


Figure 5: Ringing of a 3-dB Chebychev filter.

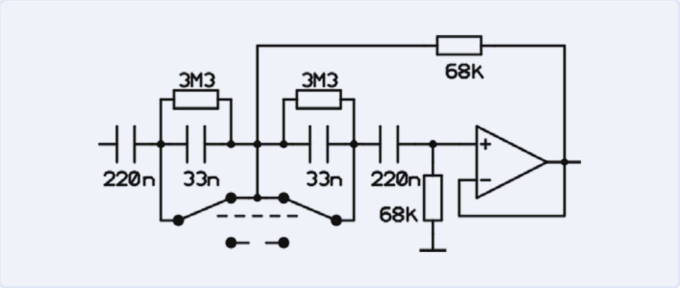


Figure 6: The high-pass filter.

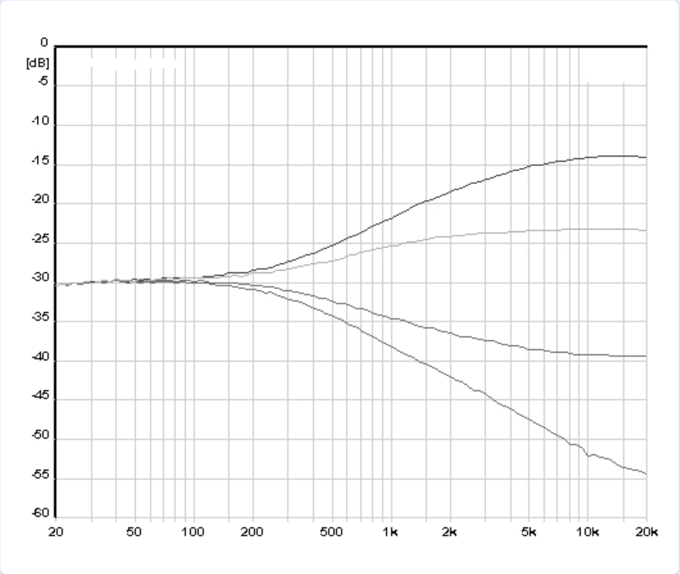


Figure 7: Frequency characteristic of the high tone filter.

- With nonparametric filters only the gain can be adjusted (where an attenuation is the same as a gain of less than one). Nonparametric filters are found in "simple" tone controls and in graphic equalizers.
- With semi-parametric filters, the gain as well as the cross-over frequency can be adjusted. Mixing panels in the medium price range typically have two parametric intermediate filters combined with nonparametric high and low tone filters.
- With parametric filters, besides the gain and cross-over frequency, the filter quality (Q-factor) is also adjustable. This latter parameter determines whether the filter operates over a narrow or a wide frequency band.

We will now look at a practical example, specifically the tone control of the Mitec Event mixing panel (with thanks to Mr. Pape for permission to publish the relevant schematics). This tone control is a typical example of a "middle-of-the-range" panel: two semi-parametric intermediate filters, nonparametric high and low tone filters and a switchable high-pass filter.

The High-Pass Filter

The high-pass filter is the first element in the signal path, so we will begin with that. This is a second-order, Linkwitz-Riley filter where the cross-over frequency can be switched between about 10 Hz and about 80 Hz (**Figure 6**). The purpose of the two 3.3-MΩ resistors is to ensure that there is no DC voltage across the 33-nF capacitors and so prevents spurious audible noise when switching on.

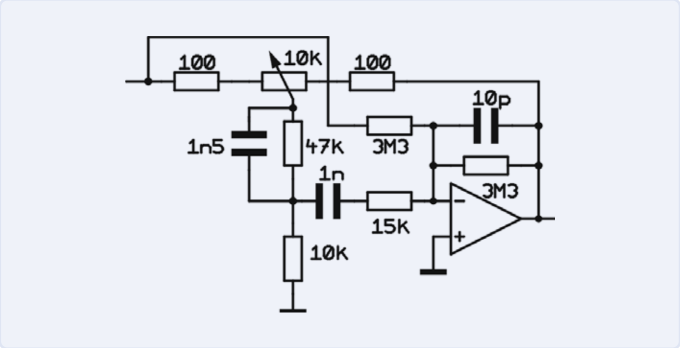


Figure 8: The high tone filter.

The High-Tone Filter

This filter exhibits a *shelving* characteristic (**Figure 7**): from the cross-over frequency until the end of the transfer range all frequencies are amplified. Of note is the broad range of this filter.

The wiper of the potentiometer (see the schematic in **Figure 8**) sits between the input and the output of the circuit and in this way provides either gain or attenuation. Between the wiper of this potentiometer and the inverting input of the opamp is the frequency-determining network, which is reminiscent of a Wien filter.

We will leave it here for the time being. In the next installment, we will continue with this tone control.

Translated by Arthur de Beun — 240711-01

Editor's note: This series of articles, "Starting Out in Electronics," is based on the book, Basiskurs Elektronik, *by Michael Ebner, which was published in German and Dutch by Elektor.*

Questions or Comments?

If you have any technical questions regarding this article, contact the Elektor editorial team at editor@elektor.com.

Related Product

- **B. Kainka, *Basic Electronics for Beginners* (Elektor, 2020)**
  Book: www.elektor.com/19212
  Ebook: www.elektor.com/19213

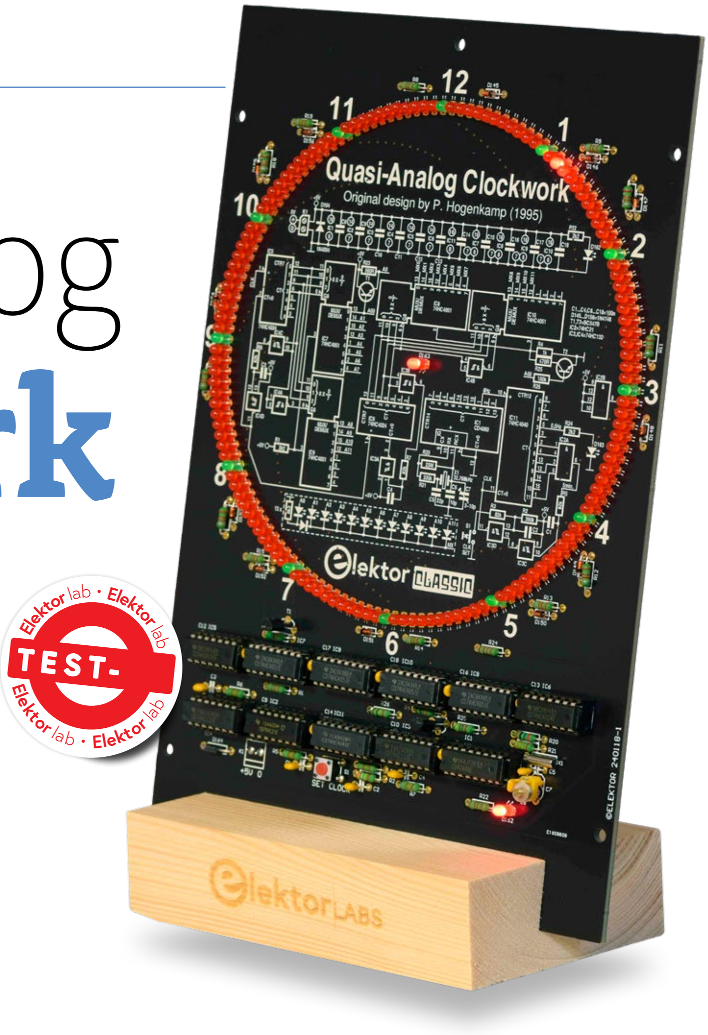
Analogue Electronics and Microcontrollers Projects
Basic Electronics for Beginners
Burkhard Kainka
elektor

WEB LINK

[1] "Starting Out in Electronics...Filters Actively," Elektor 1-2/2025: https://www.elektormagazine.com/magazine/240637-01

Quasi-Analog Clockwork

A Remake of an Elektor Classic



By Ton Giesberts (Elektor)
Original Project by P. Hogenkamp

Mechanical timepieces have always had a certain appeal because of the beauty of the analog dial and its convenience in reading, despite their limitations in accuracy. On the other hand, the digital models are very accurate, but cold and impersonal. Why, then, not combine the two worlds as is done in this analog-looking digital watch design?

Digital watches with displays consisting of numbers only have lost much of their popularity over the last few years. They are now outnumbered by traditional dials with hour and minute hands actuated by a digitally controlled stepper motor. These watches (and clocks) are the best of both worlds because they have an analog readout with digital control. The LED clock described in this article is based on the same principle, only there are no moving parts at all.

The dial is made of 144 × 3 mm LEDs in a circle, showing 12 hours with a 5 minute resolution. In this new design, 11 standard logic ICs are used, and all parts are through hole. The power supply can be sourced from any 5 V DC mains adapter.

This project is based on an article that is about 30 years old [1]. A kit is available at the Elektor shop [2].

Notes on the Remake

A number of changes were made by the Elektor Lab Team to make this 30-year-old design compatible with components, insights and technologies available.

- To make the circuit independent of the mains frequency (50 Hz or 60 Hz), a separate reference clock signal of 32.768 kHz is used instead of using 50 Hz only as was in the original 1995 design. A higher divisor is needed for a 5-minute pulse, i.e., 9,830,400 instead of 15,000, requiring an additional counter/divider circuit.
- 4000-logic CMOS ICs are no longer widely available, so their functionality is taken over by HCMOS logic throughout the design (except a CD4060, used for the reference clock). For example, the 4093 ICs have been replaced by 74HC132, the 4082 by 74HC21, and each 4067 by two 74HC4051s.
- The supply voltage is reduced from 12 V to 5 V. The external transformer, bridge rectifier, smoothing capacitors and stabilizing network using a Zener diode are no longer needed, avoiding the risk of having the mains voltage in the circuit. Instead, any small 5 VDC mains adapter can be connected via a small screw-terminal block on the board.
- Various changes affect the current settings of the LEDs. Apart from that, present-day LEDs have much higher efficiency, so the LED currents can be kept really low.
- The actual PCB design is double-sided, avoiding a mass of jumper wires.
- The blinking LED — rather than the power LED — is placed in the center of the dial.

- Two transistors were added to increase the current of the green LEDs lighting on the full hour — this is to compensate for their lower efficiency, compared to the red LEDs.

Circuit Description

As anticipated, the quasi-analog clockwork uses 144 LEDs to indicate the time on a round, quasi-analog dial with a diameter of about 143 mm. One of twelve green LEDs lights at maximum intensity to mark the hours, while the other eleven are dimmed. Between two green LEDs sit 11 red LEDs, each of which represents a period of five minutes. In this way, the time is indicated with an accuracy of five minutes. This would seem to be enough, considering the mostly decorative function of the present clockwork.

The complete circuit diagram of the clockwork is given in **Figure 1**. The oscillator section of counter/divider IC1 is configured as a quartz oscillator with a 32.768 kHz crystal. The high value of R21, 330 kΩ, ensures the total power delivered to the crystal is well below the maximum drive level of 1 µW (a more aggressive drive of this quartz would reduce its lifespan).

The total divider for the 5-minute pulse is created in two parts. The CT11 output of counter/divider IC1 (a CD4060 made by Texas Instruments, which also produced all the other ICs used for this project) ticks at 8 Hz and clocks the second counter divider, IC11 (a 74HC4040). IC11 must divide 8 Hz by 2400 to create a 1/300 Hz pulse rate (i.e., a 5-minute pulse).

A total divisor of 9,830,400 is achieved by detecting the binary code 2400 at the output of IC11 through IC2A. In numbers: 2400 = $2^{11} + 2^{8} + 2^{6} + 2^{5}$. When the value of 2400 is reached, a CLK pulse is generated via R5, C1, IC3C, and IC3D, resetting IC11 (CT=0). The CLK pulse also increments the count of IC5 by 1 (see symbol CLK at pin 1). The other input of IC3D is connected to pushbutton S1, whose purpose is to adjust the time after power-on. The pulse supplied by the pushbutton

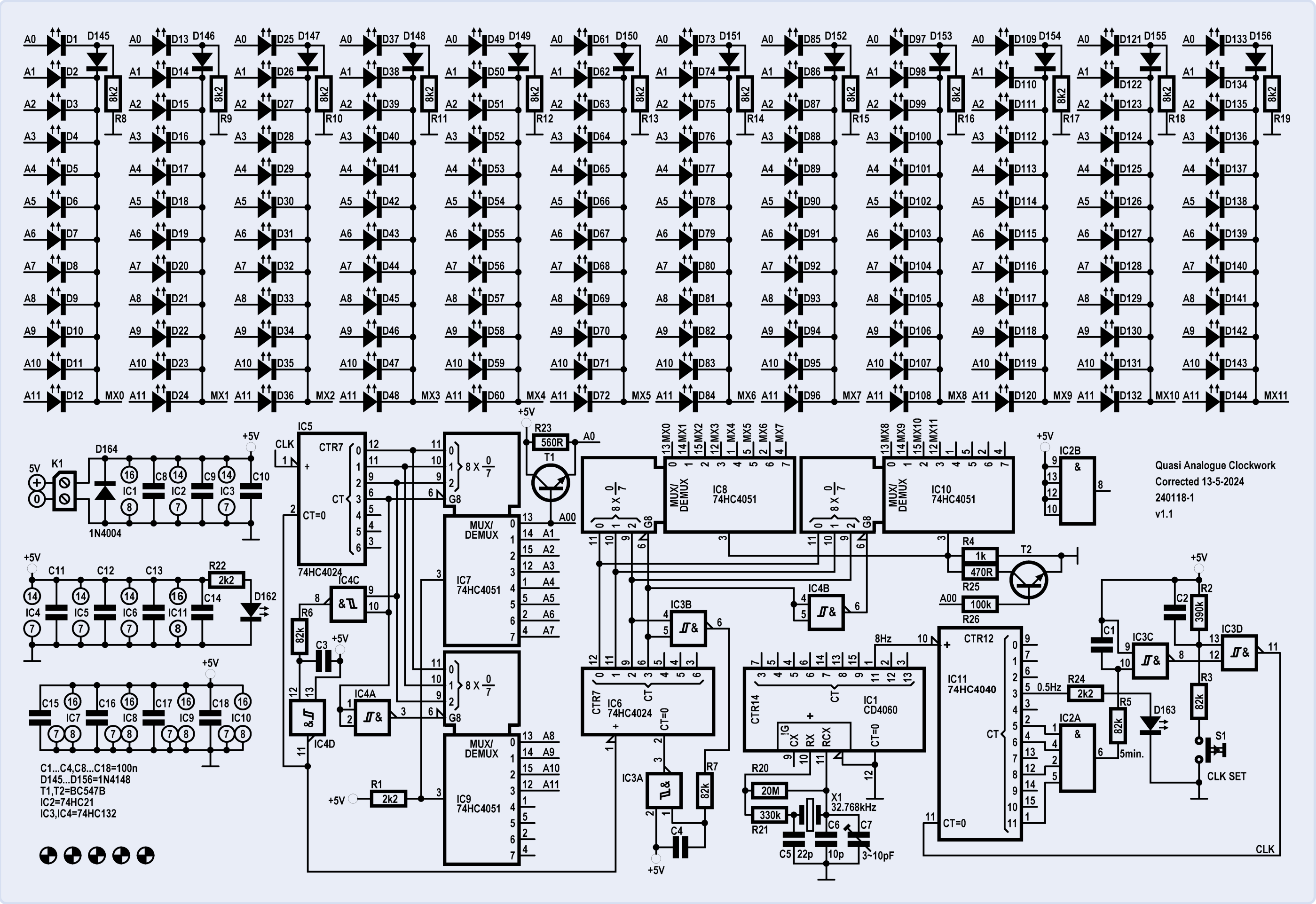


Figure 1: Schematic diagram of the Quasi Analog Clockwork.

▼

is debounced by network R3-R2-C2 and IC3D. Every time the button is pressed, the readout advances by one LED position.

IC5, a 74HC4024 counter/divider, operates as a 5-minutes counter and has 12 states, corresponding to 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 and 55 minutes. Its outputs determine which channel of multiplexers IC7 and IC9 (both 74HC4051) is actuated. The multiplexers' COM pins (pin 3) are connected.

For example, when all multiplexer SEL inputs and the active-low E (enable) pins (pin 11, 10, 9 and 6, respectively) are held at 0, the anode terminals of the LEDs connected to output A0 (D1, D13, D25, D37, D49, D61, D73, D85, D97, D109, D121 and D133) are connected to the +5 V supply rail via a 2.2 kΩ resistor, R1. IC4A inverts the fourth bit of counter IC5 to enable IC9 when IC7 is disabled and vice versa.

To make an LED light, however, its cathode must be pulled to ground and that is done by multiplexers IC8 and IC10 (both 74HC4051), which have their COM pins joined and tied to ground via R4 (1 kΩ). The binary pattern out of the counter/divider IC6 (74HC4024) is applied to the SEL and E inputs (IC8 and IC10, where IC4B has the same function as IC4A) and determines which LEDs have their cathodes connected to ground through the MX0…MX11 signal lines.

As soon as IC5 reaches state "12" (corresponding to 0 minutes), the counter is reset via IC4C and IC4D. Also, IC6 receives a clock pulse which marks the start of a new hour. Incrementing the count in IC6 results in the next output of IC8 or IC10 becoming active. The multiplexers ensure that the cathodes of the selected LEDs are connected to R4. As soon as IC6 supplies the binary code "12", a reset pulse is generated via IC3B, R7, C4 and IC3A. This pulse resets IC6 to state "0" enabling the counter to start counting another 12 hours.

As already mentioned, the display has 132 red and 12 green LEDs. The green LEDs light continuously at low intensity, due to their cathodes connected to ground via 8.2 kΩ resistors (R8…R19). Since the anodes of the green LEDs are connected to R23, a constant current of about 0.2 mA flows through each LED. At the full hour, the relevant green LED must light at maximum intensity.

This increase in brightness is achieved by connecting R4 (via MX0…MX11) through a diode (D145…D156). This trick explains why the LEDs connected to line A0 are wired differently in the matrix. The current is also increased by connecting R25 in parallel with R4 via T2. R26 is connected to virtually +5 V through IC7 and R1. Due to buffer T1, the voltage drop across R1 is then only 0.1 V.

LED D162 lights as soon as the supply voltage is present. The last LED of the project, D163, is located in the center of the dial and flashes slowly at 0.5 Hz to indicate that the clock is "ticking." The 5 V DC supply voltage for the

Component List

Resistors

(body 2.5 × 6.8 mm)
R1, R22, R24 = 2.2 kΩ
R2 = 390 kΩ
R3, R5, R6, R7 = 82 kΩ
R4 = 1 kΩ
R8-R19 = 8.2 kΩ
R20 = 20 MΩ
R21 = 330 kΩ
R23 = 560 Ω
R25 = 470 Ω
R26 = 100 kΩ

Capacitors

C1...C4, C8...C18 = 100 nF, 50 V, ceramic X7R, pitch 5 mm
C5 =22 pF, 50 V, ceramic C0G/NP0, pitch 5 mm
C6 = 10 pF, 50 V, ceramic C0G/NP0, pitch 5 mm
C7 = 3...10 pF trimmer, BFC280823109 Vishay/
BC Components

Semiconductors

D1, D13, D25, D37, D49, D61, D73,
D85, D97, D109, D121, D133 = LED, green, 3 mm
D2…D12, D14…D24, D26…D36, D38…D48, D50…D60,
D62…D72, D74…D84, D86…D96, D98…D108, D110…D120,
D122…D132, D134…D144, D162, D163 = LED, red, 3mm

Important: 3 mm round LED's D1…D144 must have a flat side, no ledge!

D145…D156 = 1N4148, DO-35
D164 = 1N4004, DO-41
T1, T2 = BC547B
IC1 = CD4060, DIP-16
IC2 = 74HC21, DIP-14
IC3, IC4 = 74HC132, DIP-14
IC5, IC6 = 74HC4024, DIP-14
IC7...IC10 = 74HC4051, DIP-16
IC11 = 74HC4040, DIP16

Miscellaneous

K1 = 2-way PCB terminal block, 3.5 mm grid
S1 = 6 mm tactile pushbutton
X1 = 32.768 kHz crystal, 20 ppm, C_{load} 12.5 pF,
8 x 3 mm cylinder package (X32K768L104,
AEL Crystals, pitch 1.1 mm)
Optional: DIP IC sockets, 14 contacts (IC2…IC6),
16 contacts (IC1, IC7...IC11)
PCB 240118-1

clock is provided by an external mains adapter that is connected to the screw-terminal block K1. D164, a 1N4004 diode, protects against accidental polarity reversals. The current demand from the clock varies from 6 to 11 mA, so the AC/DC adapter can be a low-power type.

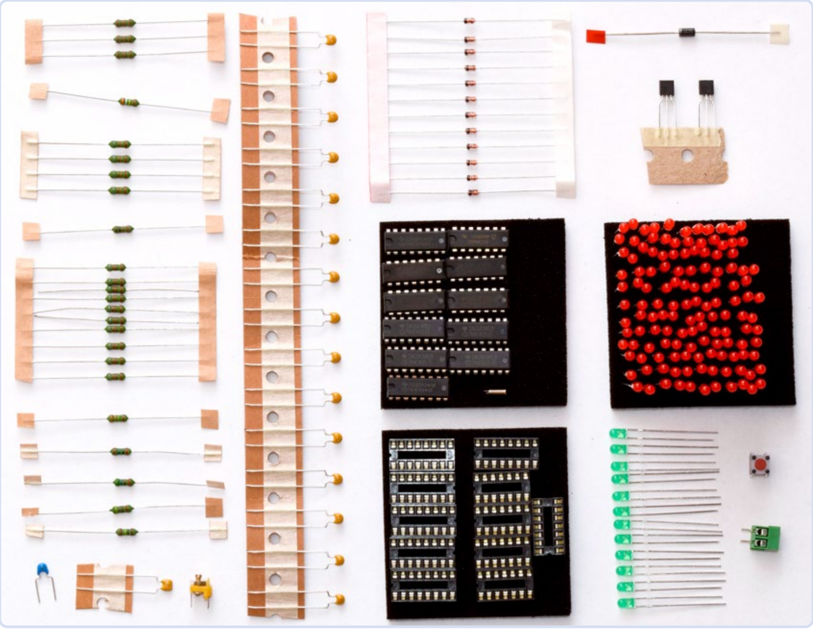


Figure 2:
All the components needed for this project are included in the kit. If you source them by yourselves, keep in mind to order rimless LEDs.

Construction

The PCB — whose layout is available at the Elektor Labs page for this article [3] — is doubled-sided and wire jumpers aren't necessary in this design. Special LEDs are required for this clock, as can be seen in **Figure 2**, which illustrates all the components included in the kit. Because of the type of mounting intended, all dial LEDs must be of the rimless type, in order to be mounted adjacent to each other.

Mount these LEDs close to the board surface, to keep the clockwork as flat as possible. Also use miniature ceramic decoupling capacitors. If you are confident about your soldering skills, you may fit the ICs without sockets. In that case, take the necessary precautions against ESD when soldering them. A fully detailed construction manual for this kit is available for download at [4].

Figure 3 shows a front view of the fully populated PCB, just after its first power-up. Maintaining Elektor's tradition for all Elektor Classic kits, the silkscreen print within the dial shows the complete circuit diagram, making this special clock a collector's item and an interior design piece at the same time!

The finishing touches to the clock depend on your own creativity. For instance, the dial may be covered by a bezel, which partly obscures the components but leaves the active LEDs clearly visible. Alternatively, a piece of transparent acrylic panel leaves "the works" in sight. And that, arguably, will be the best option for the true electronics enthusiast.

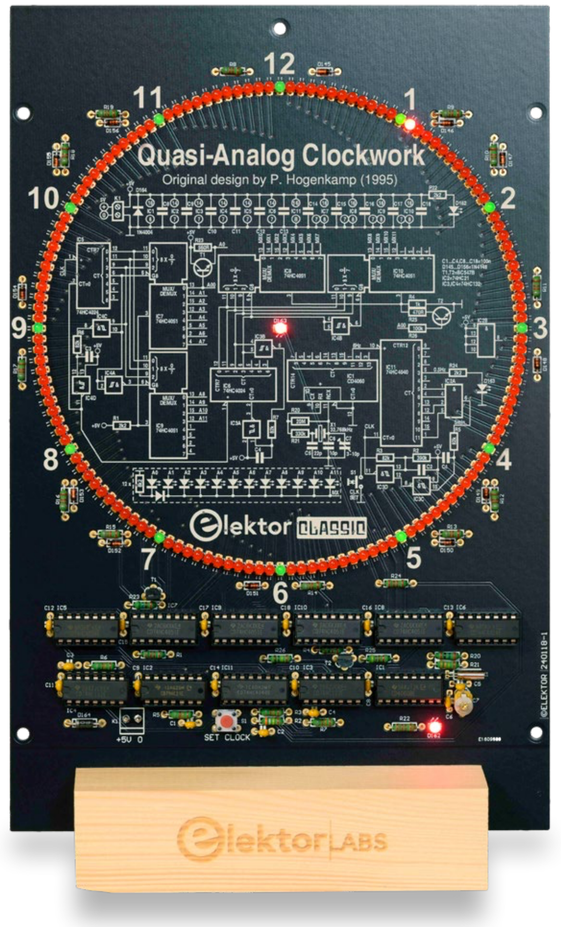


Figure 3: Front view of the completed PCB, just after the initial power-up.

32.768 kHz Reference Clock Testing

The clock should start to work the moment 5 VDC is applied to K1 (read 32.768 kHz reference clock). Which LED will light first is random and depends on the state of counters IC5 and IC6 at power-up. The LED at the top is green LED D1, indicating 12:00. Every time S1 is pressed, the clock is advanced by five minutes.

While testing the standard quartz oscillator with a 74HC4060 and a 32.768 kHz crystal, the oscillator worked fine, but produced a wrong and too high random frequency. Adding a small low-pass filter (330 Ω/100 pF) seemed to work. Anyhow, to be sure, a test was done with nine different crystals from eight manufacturers. One out of nine crystals didn't work with the extra filter.

Testing the same oscillator circuit with a standard 4060, from the 4000-logic series, the oscillator worked correctly with all crystals. The HC-logic version apparently causes enough interference in the circuit of the oscillator, due to its very high impedance, most likely caused by the higher switching speed, even on a PCB with a ground plane. And even when the frequency was almost correct, it wasn't as stable as is to be expected from a quartz oscillator. Fortunately, the CD4060 is still widely available and still manufactured.

A small capacitive trimmer (C7) was added to adjust the frequency to exactly 32,768.000 Hz, so the clock will show the correct time for a long while, despite the 5 minute resolution, of course. The frequency at pin 1 of IC1 is 8 Hz, and the oscillator frequency is divided by 2^{12}. On this pin, a probe can be easily connected to measure its signal with a frequency counter.

However, some readers have reported a bug that a 32.768 kHz crystal oscillator with a 4060 was not starting at power-up. I noticed this as well, at times; touching the crystal was enough to make the oscillator start. Changing values didn't help; replacing the crystal with different types didn't solve it either. Even stranger, adjusting the frequency to a sharp 8 Hz makes it most likely a permanent issue.

Observe if LED D163 is blinking at power-up. Tuning the oscillator to a slightly deviating frequency can make the oscillator start at power-up (but sometimes not). It's up to you if this is acceptable. Otherwise, simply touch the crystal and the oscillator will start. After a power outage, you will need to set the clock again anyway and then also watch if D163 LED is blinking. Another way is to place the crystal fully upright, this seems to solve it. So somehow the parasitic capacitance between the crystal and the ground plane is the culprit. ◀

240118-01

Questions or Comments?

Do you have technical questions or comments about this article? You can write to the editorial team of Elektor at editor@elektor.com.

About the Author

Ton Giesberts started working at Elektuur (now Elektor) after his studies, when we were looking for someone with an affinity for audio. Over the years, he has worked mainly on audio projects. Analog design has always been his preference. Of course, projects in other fields of electronics are also part of the job. One of Ton's mottos is: "If you want to have it done better, do it yourself." For example, for a PCB design for an audio project with distortion figures on the order of 0.001%, a good layout is crucial!

Related Products

- **Elektor Quasi-Analog Clockwork Kit**
  www.elektor.com/20944
- **Joy-IT JDS6600**
  Signal Generator & Frequency Counter
  www.elektor.com/18714

WEB LINKS

[1] P. Hogenkamp, "Quasi-analogue clockwork," Elektor 1/1995: https://www.elektormagazine.com/magazine/elektor-199501/33259

[2] Elektor Store webpage for the clock: https://www.elektor.com/products/elektor-quasi-analog-clockwork-kit

[3] Elektor Labs page for this project: https://www.elektormagazine.com/labs/quasi-analog-clockwork-an-elektor-classic-a-remake

[4] Quasi-Analog Clockwork construction manual: https://tinyurl.com/34nuvjtb

A Modular Approach to Sensor Testing

The ESP32-S3-Based Sensor Evaluation Board

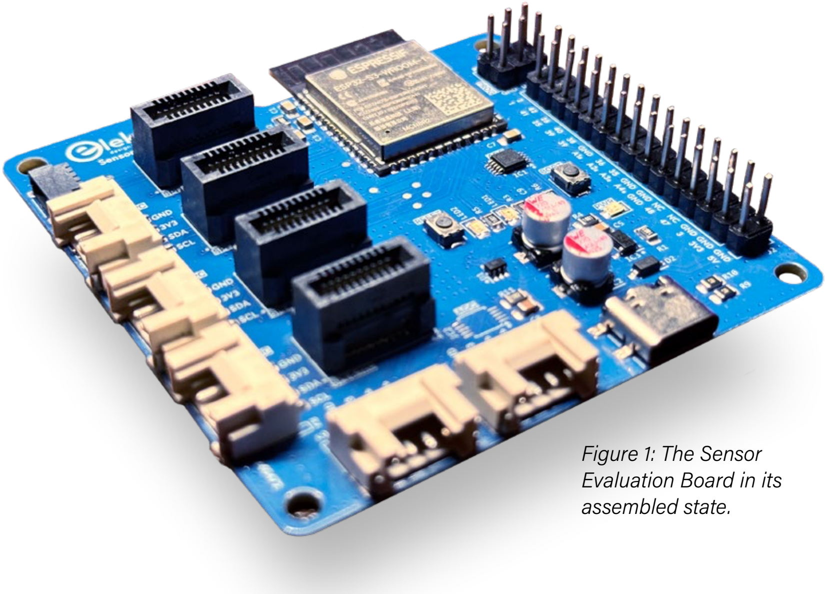


Figure 1: The Sensor Evaluation Board in its assembled state.

By Saad Imtiaz (Elektor)

The Sensor Evaluation Board was created to solve the hassle of repeatedly swapping sensors during development. Based on the ESP32-S3, it includes two high-precision ADS1015 ADCs, Grove connectors, and modular edge card slots. These features allow one to effortlessly test and swap sensors, addressing the ESP32's ADC limitations and ensuring consistent, accurate data for a smoother, more efficient prototyping process.

Before starting any project, we begin with an idea — a way to solve a problem or improve an existing solution. Then comes the planning stage, where we identify what's needed to bring that idea to life: the tools, equipment, and components that will make the project a reality. For sensor-based projects, this can often mean repeatedly testing different sensors, swapping out hardware, and troubleshooting to get everything working just right. That's where the Sensor Evaluation Board comes in (**Figure 1**).

Designed to make sensor testing simpler and more efficient, the board allows developers to quickly change sensors, optimize code, and troubleshoot without the usual hassle. In almost every embedded systems project, there's a core set of sensors we rely on time and time again — think temperature, humidity, light, motion sensors and more. These essential sensors form the building blocks for a range of applications, from straightforward environmental monitors to sophisticated IoT devices. But as projects and applications grow, so does the need for various setups, each requiring careful testing of both hardware and software compatibility. The process of rewiring, rearranging setups, and managing a cluttered workspace can quickly become tedious.

The idea of modular edge cards for each of these commonly used sensors can be employed to tackle this issue. By simply plugging in an edge card dedicated to a specific sensor, one can now swap sensors effortlessly, with no wire clutter or reconfiguration required. This modular approach streamlines testing and keeps desk space optimized, allowing multiple sensor setups to be tested on the same board seamlessly. Integrated with high-precision external ADCs, the Sensor Evaluation Board transforms the process, saving time and ensuring that every setup can be tested with accuracy and ease. Whether one is involved in environmental monitoring, IoT applications, or simply experimenting with sensor data, this board offers a seamless and structured approach to transforming concepts into fully functional prototypes.

Design and Architecture

The schematic in **Figure 2** outlines a design aimed at simplifying sensor testing with the ESP32-S3 module as the central controller. The ESP32-S3 was chosen for its dual-core capabilities and strong support for Wi-Fi and Bluetooth, making it suitable for IoT applications. Compared to options like STM32, the ESP32 offers broader community support and simpler integration for Wi-Fi-focused projects. Moreover, integrating a ESP32 WROOM Module is much easier in your projects as compared to other microcontrollers as the WROOM modules already have the

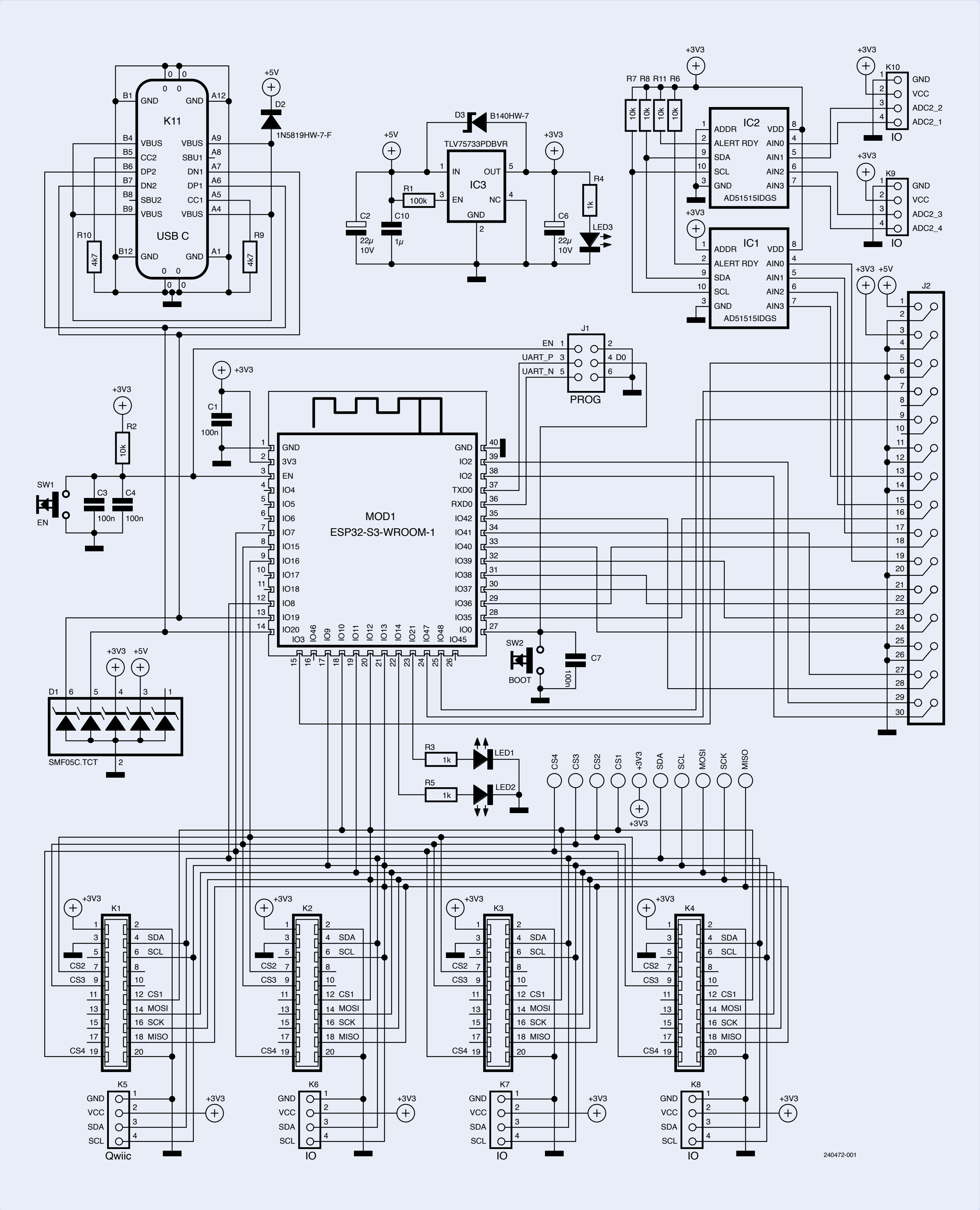


Figure 2: Schematic diagram of the project.

Component List

Resistors

R1 = 100 kΩ
R2, R11, R6, R7, R8 = 10 kΩ
R3, R4, R5 = 100 Ω
R9, R10 = 4.7 kΩ

Capacitors

C1 = 100 nF, 50V
C2, C6 = 22 µF
C3, C4, C7 = 0.1 µF
C5, C8, C9, C10 = 0.01 µF

Semiconductors

D1 = SMF05C.TCT
D2 = 1N5819HW-7-F
D3 = B140HW-7
IC1, IC2 = ADS1015IDGS
IC3 = Regulator TLV75733PDBVR
LED1 = NCD0805A0 (Amber)
LED2 = NCD0805G1 (Green)
LED3 = NCD0805R1 (Red)
MOD1 = ESP32-S3-WROOM-1

Others

SW1, SW2 = Button SKRKAEE020
J1 = 2×3 Pin, 2.54 mm Vertical Header
J2 = 2×15 Pin, 2.54 mm Vertical Header
K1, K2, K3, K4 = 408-52020-000-11 (ept Card Edge Connectors)
K5 = Qwiic Connector
K6, K7, K8, K9, K10 = Grove Connectors
K11 = USB-C GSB1C41110SSHR

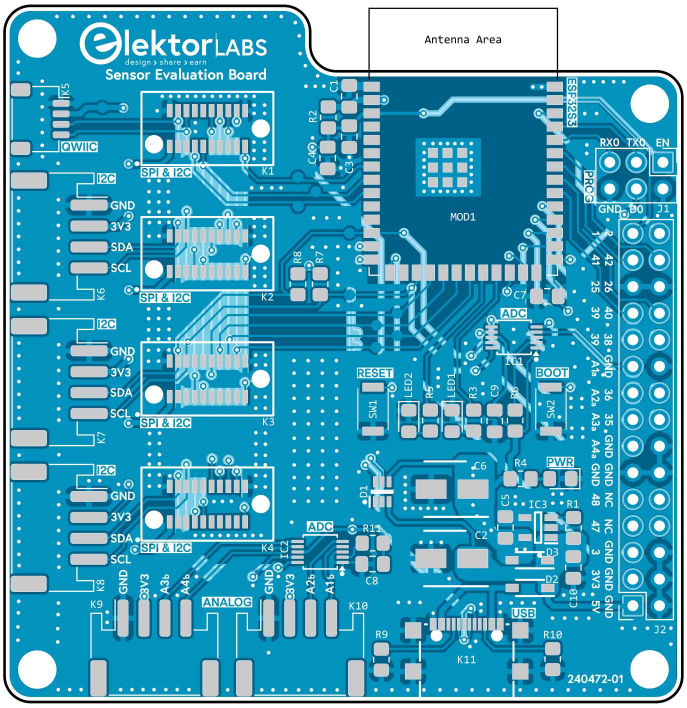


Figure 3: PCB layout, highlighting the reference numbers for connectors and component placement.

antenna front end designed along with other core components to use with the MCU, which further decreases the number of components to make your prototype work. You can address to **Figure 3** which shows the PCB layout of the board.

The board includes two ADS1015 ADCs (IC1 and IC2) to address the limitations of the ESP32-S3's internal ADCs. Both of them have four analog inputs (channels). The analog inputs of the first ADC are connected to 2.54-mm pitch header pins and the analog inputs of the second ADC to two Grove connectors (two channels each, together with VCC and GND). The ADS1015 provides reliable 12-bit resolution [1], which is adequate for most sensor tasks while remaining cost-effective. Although a higher-resolution ADC like the ADS1115 [2] was considered, the ADS1015's balance of performance (3300 Samples/s vs. 860 Samples/s) aligned better with the board's purpose. However, one can use ADS1115 instead of the ADS1015 on the same PCB as both of these ICs share the same pinouts and footprint. As you can see, the ADDR pin of one ADC is connected to GND, the other one to VCC — which gives these chips different I²C addresses (more on this can be found in the datasheet [1]).

For status indication, the board features two LEDs (LED1 and LED2) connected to GPIO14 and GPIO21 of the ESP32-S3. These LEDs can be configured for any application, such as displaying power status, communication activity, or custom debugging signals during sensor testing.

One I²C interface of the ESP32 (GPIO8 = SDA) and GPIO9 = SCL) is routed to the edge card receptables K1 to K4 and additionally to three Grove connectors K6 to K8 and one Qwiic connector K5. Similarly, the SPI communication interface is allocated as follows: SDI on GPIO11, SCK on GPIO12, and SDO on GPIO13. These lines are also connected to the edge card receptables; together with four CS (Chip Select) lines. This enables simultaneous use of SPI sensors across the edge card connectors without conflicts, enabling greater flexibility in sensor testing setups.

As said, the four edge card connectors (K1...K4) enable straightforward sensor swapping without any rewiring. These connectors are EC.8 edge card SMT connectors from ept [3], which are indeed quite robust for this application; they support up to 28-Gbps data transmission and a current capacity of 3.2 A (specs well above what is required here). However, their main advantage lies in their ease of use, allowing for seamless, frequent sensor module changes in a testing environment. Compared to traditional pin headers, these edge connectors make it significantly easier to streamline testing setups, with a claimed 500 insertion cycles that should hold up well over time.

Connector J1 is dedicated to programming the ESP32-S3 via UART, providing a straightforward interface for uploading firmware and making initial configurations. However, one can also flash firmware via the USB Type C Connection (K11) as well. J2, on the other hand, is a 2.54-mm header that grants access to the remaining GPIO pins of the ESP32-S3 and the first ADC (IC1). This additional access is valuable for connecting external modules or custom peripherals directly to the ESP32-S3, allowing flexibility for expanding the board's functionality as needed during development and testing.

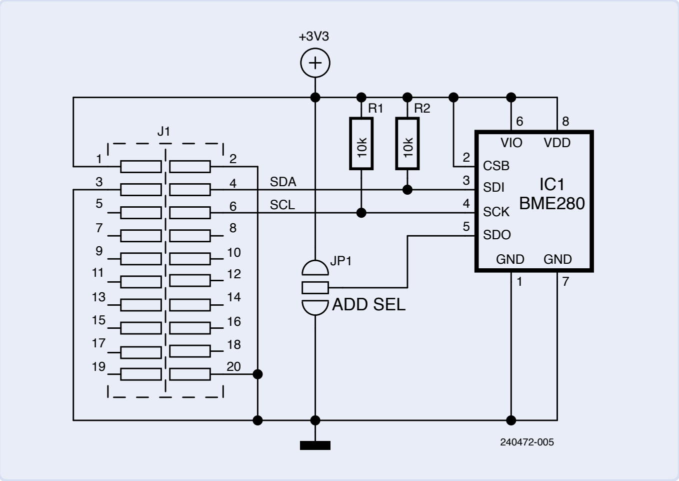


Figure 4: Schematic diagram of BME280 Edge Card module, outlining its pin configurations.

As stated before, the Chip Select (CS) line poses a unique challenge, as each sensor requires a dedicated CS pin when operating over SPI. To accommodate this, four separate CS lines are available on the edge card connectors, each with a unique GPIO assigned by the microcontroller. While this solution meets the needs of the current design, a multiplexing approach, or a GPIO expander could be considered for future versions to allow dynamic assignment of CS pins. For now, this configuration effectively supports simultaneous operation of multiple sensors, enabling flexible and straightforward sensor module swapping.

Modular Sensor Options

Two edge card sensor modules were created for the board: a BME280 environmental sensor [4] and an ICM42688 IMU sensor [5]. The schematic diagrams of these modules were based on their respective datasheets; however, for reference, you can see them in **Figure 4** and **Figure 5**. Both edge cards share identical dimensions 23.55 × 13.55 mm (**Figure 6**), with standardized pinouts for I²C and SPI lines to maintain compatibility.

For testing these sensors, I utilized libraries that simplified the integration process. For the ICM42688, the *ICM42688* library by finani was used [6], which proved to be user-friendly and included examples covering various functionalities such as I²C and SPI modes, interrupt handling, and more. For the BME280, I relied on the *Adafruit_BME280_Library* [7], which is well-known for its simplicity and ease of use.

When designing the edge card modules, I selected the *02x10 Connector odd/even* symbol in the schematic and used the *Samtec_HSEC8:Samtec_HSEC8-110-X-X-DV-BL_2x10_P0.8mm_Edge* footprint from KiCAD's default library for the PCB layout. For the edge card connectors on the Sensor Evaluation Board, I downloaded the footprint file directly from the manufacturer's website and added it to my KiCAD library using the Library Loader program [8].

Regarding the physical specifications, the EC.8 connectors used on the board are designed to support 1.6-mm-thick PCBs. To ensure compatibility, I specified a 1.6-mm PCB thickness when ordering the edge card modules from the PCB manufacturer.

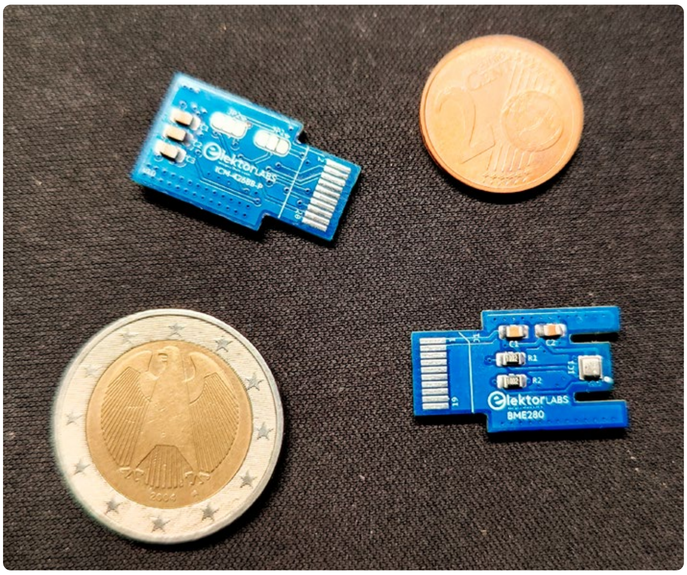


Figure 6: Size comparison of the Edge Card modules with a 1 Euro coin for scale.

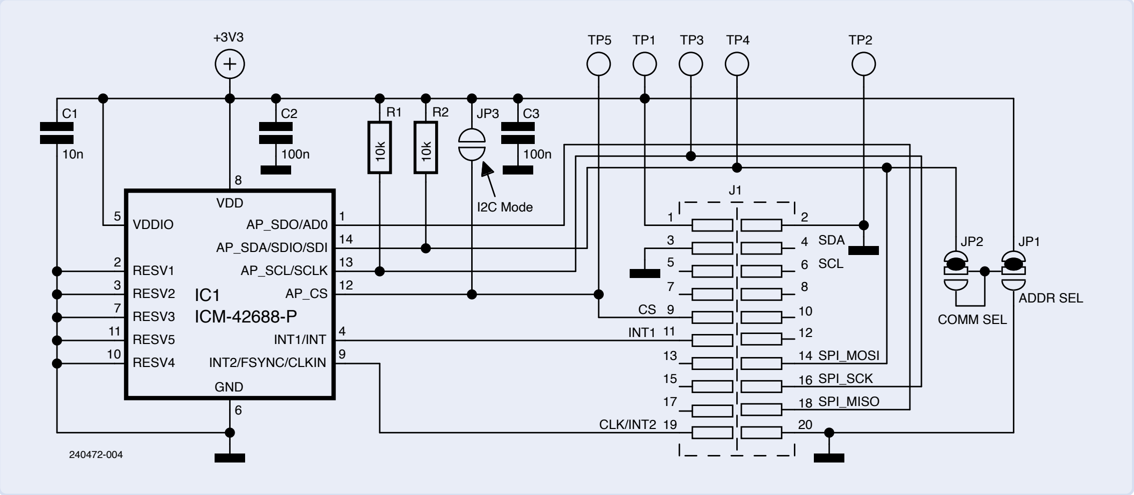


Figure 5: Schematic diagram of ICM42688 Edge Card module.

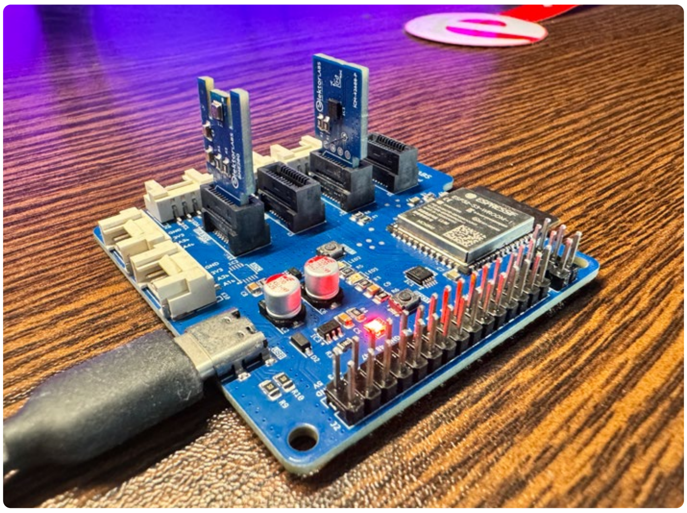


Figure 7: The Sensor Evaluation Board with sensor modules inserted, illustrating the modular approach.

Testing and ADC Limitations of ESP32

During testing, I evaluated both of the onboard ADS1015 ADCs. Unfortunately, only a single ADS1015 IC was available with me at the time, so I tested it in both locations separately. Despite this limitation, both configurations performed identically under normal conditions. However, the ADC at IC1 showed slightly better noise performance. This is likely due to its position, being closer to its header pins and further from the board's power section. That said, this noise difference is negligible for practical purposes.

When comparing these external ADCs to the ESP32-S3's built-in ADC, the improvement is stark. The ADS1015 offers far better accuracy and significantly reduced noise, even with the ESP32-S3 clocked at 20 MHz. The difference in performance is almost incomparable — truly a "day and night" contrast — making these external ADCs a worthwhile addition.

Beyond the ADC testing, this board's modular design truly shone when swapping and testing the edge card sensor modules. While I currently have only two modules — the BME280 and ICM42688 — the flexibility, they provide has inspired me to expand the collection. As shown in **Figure 7**, the compact size of the board, measuring just 64.7 × 66.6 mm, is another advantage. It's smaller than the mouse sitting next to it in **Figure 8**, making it a perfect desk-friendly tool. This modular approach has proven incredibly useful for testing sensors and has earned a permanent spot on my workbench.

Closing Thoughts and Future Enhancements

Like any project, there's always room for improvement, and reflecting on this design has given me several ideas for the next iteration. One area for refinement is the layout of the edge card connectors. While the present arrangement works, I encountered some difficulty soldering them using a rework station due to the tight spacing (**Figure 9**). This led me to use a hot plate method instead, just for these connectors. In future designs, I plan to space the connectors more generously to ensure better accessibility for soldering on both sides.

Another improvement would be relocating the bottom ADC (IC2) slightly further from the power section of the board. While the noise difference observed during testing was very minimal, optimizing its placement could further enhance signal integrity. At the time of designing, I speculated this might have a very minor impact, and testing confirmed that even small adjustments in layout can make a measurable difference.

Additionally, I would explore incorporating a multiplexer for the Chip Select (CS) lines on the edge card connectors. This would allow dynamic assignment of CS pins, making the design more scalable and reducing the reliance on predefined GPIOs for each edge card. It would streamline the integration of multiple edge cards and enable more flexibility in future designs.

Lastly, I'd consider adding an onboard OLED display for real-time feedback during testing. Features like displaying the sensor status or ADC readings directly on the board could enhance its usability and speed up debugging.

Overall, this project has been a valuable experiment in modular design and has proven its utility during sensor testing. While there's always room for improvement, this version serves its purpose well and provides

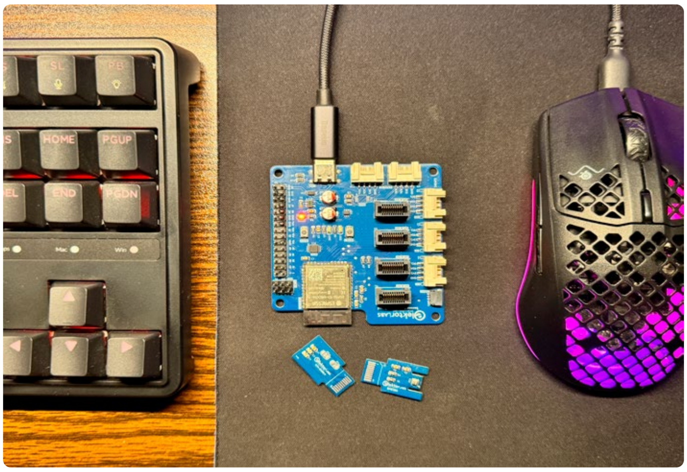


Figure 8: A size comparison of the Sensor Evaluation Board next to a keyboard and mouse, demonstrating its desk-friendly dimensions.

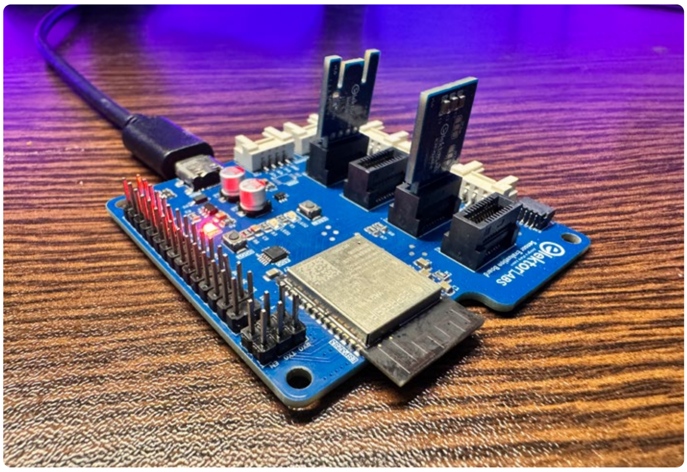


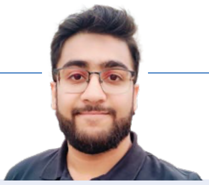
Figure 9: Side-angle view showing the tight spacing of the edge card connectors, highlighting potential for future improvements.

a strong foundation for future enhancements. With a few adjustments, I'm confident this board can evolve into an even more powerful and adaptive tool for embedded development.

240472-01

Questions or Comments?

If you have questions about this article, feel free to email the author at saad.imtiaz@elektor.com or the Elektor editorial team at editor@elektor.com.



About the Author

Saad Imtiaz, Senior Engineer at Elektor, is a mechatronics engineer who has extensive experience in embedded systems and product development. His journey has seen him collaborate with a diverse array of companies, from innovative startups to established global enterprises, driving forward-thinking prototyping and development projects. With a rich background that includes a stint in the aviation industry and leadership of a technology startup, Saad brings a unique blend of technical expertise and entrepreneurial spirit to his role at Elektor. Here, he contributes to project development in both software and hardware.

Related Product

- **Dogan Ibrahim, *The Complete ESP32 Projects Guide*, Elektor 2019**
  www.elektor.com/18860

WEB LINKS

[1] ADS1015 (12 bit, 3.3 kSamples/s, 4 channels), TI Datasheet: https://www.ti.com/product/ADS1015

[2] ADS1115 (16 bit, 860 Samples/s, 4 channels), TI Datasheet: https://www.ti.com/product/ADS1115

[3] EC.8 straight, ept connectors Product page: https://www.ept-connectors.com/index.php?EC8-SMT-HighSpeed-Direct-Connector

[4] BME280, Bosch Sensortec Product page: https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/

[5] ICM-42688-P High-Precision 6-Axis IMU, TDK Product page: https://invensense.tdk.com/products/motion-tracking/6-axis/icm-42688-p/

[6] ICM42688 Sensor Libaray (GitHub): https://github.com/finani/ICM42688

[7] Adafruit BME280 Library (GitHub): https://github.com/adafruit/Adafruit_BME280_Library

[8] Setting up Library Loader for use with KiCad: https://www.samacsys.com/kicad/

2025: An AI Odyssey

The Rise of Foundation Models and Their Role in Democratizing AI

Source: Adobe Stock

By Brian Tristam Williams (Elektor)

Foundation models are transforming artificial intelligence by making powerful, general-purpose tools available to a wider audience than ever before. From generating essays to creating artwork, these versatile AI systems are reshaping industries and empowering individuals. But with great potential comes great responsibility, as questions about ethics, equity, and control loom large. Let's explore how these models work, their impact, and the challenges they present.

Artificial intelligence is no longer the stuff of science fiction — it's reshaping our world in ways that affect everyone, from businesses to hobbyists. At the center of this transformation are foundation models, a groundbreaking type of AI that's suddenly everywhere (of the 10 biggest companies in the world [1], seven are actively creating foundation models) and unlocking new possibilities but also stirring important debates.

These powerful, general-purpose AI systems — think GPTs (generative pretrained transformers) [2], BERTs (bidirectional encoder representations from transformers) [3], or the Stable Diffusion model [4] for image generation — are paving the way for a new era of accessibility, creativity, and innovation. But, while they promise to democratize AI, they also raise important questions about ethics, equity, and control. Let's unpack what foundation models are, why they matter, and how they're shaping the future of AI.

What Are Foundation Models?

At their core, foundation models are AI systems trained on massive datasets to perform a wide range of tasks. Unlike traditional machine learning models, which are typically designed for individual specific functions, foundation models are generalists. For example, OpenAI's GPT-4 can write essays, draft code, answer complex questions, and much more. Similarly, models such as Stable Diffusion generate incredible images based on simple text prompts.

The secret sauce lies in their architecture, usually based on transformers, which enable these models to understand and generate patterns across various types of data. They're called "foundation" models because they serve as a base that can be fine-tuned or adapted for specific applications — a veritable Swiss Army knife for AI.

Transformers are a type of neural network architecture that revolutionized artificial intelligence when introduced [5] in 2017. According to AI hardware powerhouse NVIDIA, they "are among the newest and one of the most powerful classes of models invented to date. They're driving a wave of advances in machine learning some have dubbed transformer AI." [6]

The transformer's key innovation is the "self-attention mechanism," which allows the model to evaluate relationships between all elements in a sequence of data simultaneously, rather than processing them sequentially, like earlier models. This enables transformers to understand context more effectively, such as how words relate to each other in a sentence or across paragraphs.

Making AI Accessible to All

One of the most exciting aspects of foundation models is their potential to make AI accessible to a much broader audience. In the past, deploying AI solutions often required a deep understanding of machine learning, access to specialized hardware, and the resources to train models from scratch. Now, thanks to foundation models, anyone with an internet connection can tap into advanced AI capabilities.

Cloud-based platforms such as OpenAI's API [7] or Amazon's AWS SageMaker [8] make it possible for businesses and individuals to integrate AI into their workflows without breaking the bank. Meanwhile, open-source initiatives, such as Hugging Face's model repository [9] or Meta's Llama [10], offer free or affordable alternatives to proprietary systems, further lowering barriers to entry.

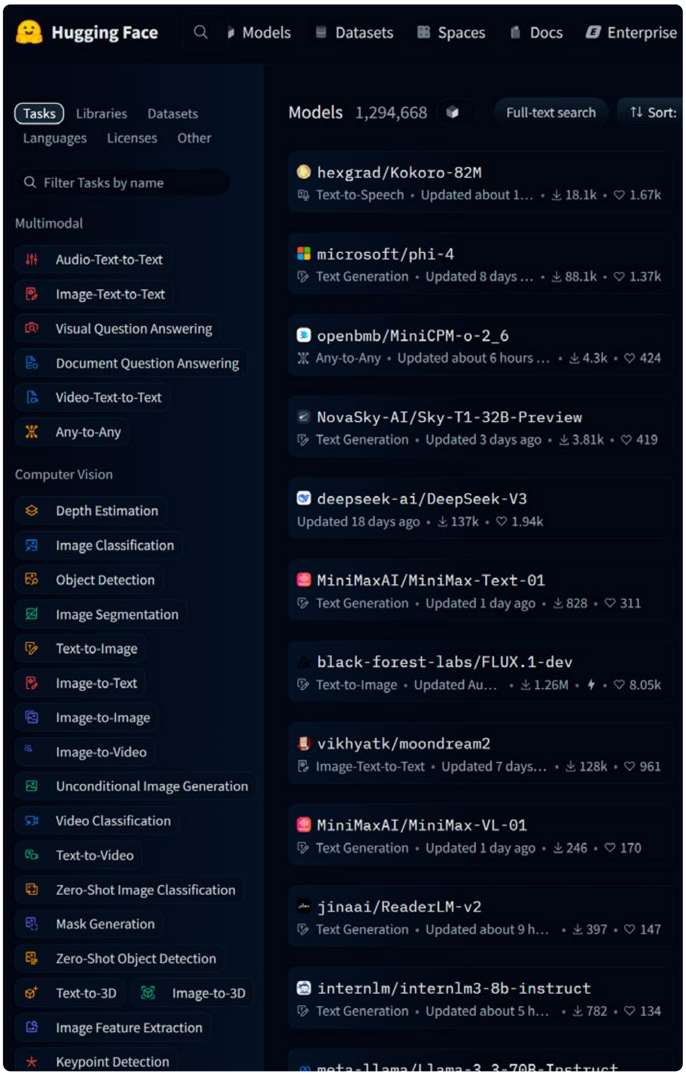
This accessibility has sparked a wave of innovation. Small businesses can use these models to analyze data or automate customer interactions. Independent creators are using tools such as DALL·E to generate unique art ("unique" may be the most charitable adjective, at least sometimes). Even we hobbyists are finding ways to apply AI to our projects — the possibilities are nearly endless.

Challenges and Ethical Questions

However, the rise of foundation models isn't without its challenges. First, there's the issue of bias. Because these models are trained on vast amounts of publicly available data, they can inadvertently learn and perpetuate societal biases. For instance, language models might generate responses that reflect stereotypes or exclusionary language.

Then there's the question of misuse. Just as with any useful tool since one of our ancestors picked up the first rock, foundation models can be used by bad actors — create deepfakes, spread misinformation, or automate cyberattacks. At scale, it poses a significant ethical dilemma.

Another concern is control. Many of the leading foundation models are owned by tech giants, raising questions about monopolies and power dynamics in the AI ecosystem. On the flip side, while open-source models promise transparency and accessibility, they can also be weaponized, leaving us to grapple with the trade-offs of openness versus security.



Models 1,294,668

hexgrad/Kokoro-82M
Text-to-Speech • Updated about 1… • 18.1k • ♡ 1.67k

microsoft/phi-4
Text Generation • Updated 8 days … • 88.1k • ♡ 1.37k

openbmb/MiniCPM-o-2_6
Any-to-Any • Updated about 6 hours … • 4.3k • ♡ 424

NovaSky-AI/Sky-T1-32B-Preview
Text Generation • Updated 3 days ago • 3.81k • ♡ 419

deepseek-ai/DeepSeek-V3
Updated 18 days ago • 137k • ♡ 1.94k

MiniMaxAI/MiniMax-Text-01
Text Generation • Updated 1 day ago • 828 • ♡ 311

black-forest-labs/FLUX.1-dev
Text-to-Image • Updated Au… • 1.26M • • ♡ 8.05k

vikhyatk/moondream2
Image-Text-to-Text • Updated 7 days… • 128k • ♡ 961

MiniMaxAI/MiniMax-VL-01
Text Generation • Updated 1 day ago • 246 • ♡ 170

jinaai/ReaderLM-v2
Text Generation • Updated about 9 h… • 397 • ♡ 147

internlm/internlm3-8b-instruct
Text Generation • Updated about 5 h… • 782 • ♡ 134

meta-llama/Llama-3.3-70B-Instruct

Figure 1: The tip of the iceberg from Hugging Face's over 1.2-million-model library. (Source: Hugging Face website)

The Role of Open Source

Fortunately, the open-source community is stepping up. Organizations such as EleutherAI [11], Hugging Face, and Stability AI are developing models that rival proprietary systems while emphasizing transparency and collaboration. These efforts are essential in ensuring that the benefits of AI are distributed more equitably.

For example, Hugging Face not only hosts a vast library of models (**Figure 1**), but also provides user-friendly tools to fine-tune them for specific tasks. Stability AI's Stable Diffusion model has democratized image generation, giving anyone with a decent GPU the ability to create professional-quality visuals on their own computer.

While open-source initiatives are a step in the right direction, they're not a silver bullet. Striking a balance between openness and responsible use remains a challenge that the community must address.

What Lies Ahead?

As foundation models continue to evolve, we can expect them to become more specialized, efficient, and multimodal — capable of handling text, images, audio, and video in a unified framework. This evolution will unlock new possibilities in fields ranging from healthcare to education, but it will also require robust governance and oversight.

The democratization of AI is a double-edged sword. On the one hand, it empowers individuals and small businesses to harness the power of advanced technology, and helps us get up and running with electronics passion projects quickly. On the other, it amplifies the risks associated with bias, misuse, and concentrated control. As a community, we must navigate these challenges thoughtfully, ensuring that AI serves as a tool for good rather than harm.

Looking Ahead

Foundation models represent a monumental leap forward in AI, offering unprecedented opportunities for innovation and creativity. However, their rise also underscores the need for vigilance, collaboration, and ethical responsibility. As these models become an integral part of our lives, it's up to all of us — developers, businesses, policymakers, and users alike — to shape a future where AI is truly accessible, equitable, and beneficial for everyone.

230181-L-01

About the Author

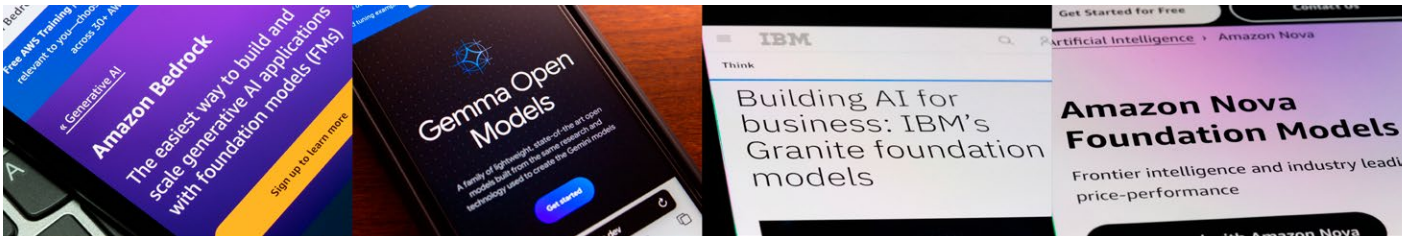
Brian Tristam Williams has been fascinated with computers and electronics since he got his first "microcomputer" at age 10. His journey with Elektor Magazine began when he bought his first issue at 16, and since then, he's been following the world of electronics and computers, constantly exploring and learning. He started working at Elektor in 2010, and nowadays, he's keen on keeping up with the newest trends in tech, particularly focusing on AI and single-board computers such as Raspberry Pi.

Questions or Comments?

If you have questions or comments, brian.williams@elektor.com is my email address. You can also catch me on Elektor Engineering Insights each month on YouTube, and you can find me @briantw on X.

Related Products

- **D. Situnayake & J. Plunkett: *AI at the Edge* (O'Reilly Media, 2023)**
  www.elektor.com/20465


Free AWS Training
Generative AI
Amazon Bedrock
The easiest way to build and scale generative AI applications with foundation models (FMs)
Sign up to learn more
Gemma Open Models
A family of lightweight, state-of-the art open models built from the same research and technology used to create the Gemini models
Get started
IBM
Think
Building AI for business: IBM's Granite foundation models
Get Started for Free
Artificial Intelligence › Amazon Nova
Amazon Nova Foundation Models
Frontier intelligence and industry leadi
price-performance

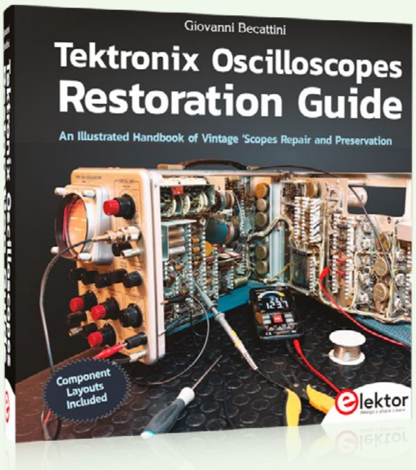
Foundation models are everywhere. (Source: @Tada Images/@Koshiro K/@Robert — Adobe Stock)

WEB LINKS

[1] Forbes India — "Top 10 Companies in the World by Market Cap in 2025": https://tinyurl.com/forbesmc
[2] GPT Wikipedia entry: https://en.wikipedia.org/wiki/GPT
[3] BERT (language model) — Wikipedia: https://en.wikipedia.org/wiki/BERT_(language_model)
[4] Stability AI: https://stability.ai
[5] "Attention Is All You Need," arXiv Cornell University: https://arxiv.org/abs/1706.03762
[6] "What Is a Transformer Model?" — NVIDIA blog: https://blogs.nvidia.com/blog/what-is-a-transformer-model
[7] OpenAI Developer Platform — Getting Started documentation: https://platform.openai.com/docs/overview
[8] Amazon SageMaker — build, train, and deploy ML models: https://aws.amazon.com/sagemaker
[9] Hugging Face: https://huggingface.co
[10] Llama: https://llama.com
[11] EleutherAI — empowering open-source artificial intelligence research: https://eleuther.ai

Tektronix Oscilloscopes Restoration Guide

Tektronix oscilloscopes are true masterpieces of electronics and have helped mankind advance in every field of science, wherever a physical phenomenon needed to be observed and studied. They helped man reach the moon, find the cause of plane crashes, and paved the way for thousands of other discoveries.


Giovanni Becattini
Tektronix Oscilloscopes Restoration Guide
An Illustrated Handbook of Vintage 'Scopes Repair and Preservation
Component Layouts Included
elektor

Price: €59.95

Member Price: €53.96

www.elektor.com/21051

SunFounder GalaxyRVR Mars Rover Kit for Arduino

The SunFounder GalaxyRVR Mars Rover Kit was designed to mimic the functionality of real Mars rovers, it offers a hands-on experience that's both educational and exciting. Compatible with Uno R3, the GalaxyRVR is equipped to navigate diverse terrains with ease.

Price: €134.95

Member Price: €121.46

www.elektor.com/21061

Raspberry Pi 5 (16 GB RAM)

Price: €139.95

www.elektor.com/21080

LILYGO T-Panel S3 Development Board

Price: €89.95

Member Price: €80.96

www.elektor.com/21026

Raspberry Pi Standalone MIDI Synthesizer (1)

Preparing a Platform for Some Edge AI Experiments

By Brian Tristam Williams (Elektor)

With AI on the edge in embedded systems now, single-board computers have been doing a lot of camera-related tasks such as object recognition and pose estimation. What if we tried something music-related instead? Let's get started.

I have been thinking of a novel task for portable AI: Perhaps it could help me musically — being a live backup player for me as I play my keyboard. In this first part, let's tackle how to actually get a keyboard MIDI controller talking to a Raspberry Pi, and having the latter make the hills come alive with the sound of music.

What Is MIDI?

MIDI (Musical Instrument Digital Interface) is a communication protocol that allows electronic musical instruments, computers, and other audio devices to connect to and control each other. Introduced in the early 1980s, MIDI transmits digital messages rather than audio signals. The types of digital messages include:

Note On/Off: Tells a device when to start or stop playing a note.
Velocity: Indicates how hard a key was pressed (volume expression).
Control Changes: Adjust parameters such as volume, modulation, pitch, or sustain.
Program Changes: Switch between instrument sounds (e.g., piano, strings).

Back then, MIDI data was sent via 5-pin DIN cables (legacy devices), but, nowadays, new gear will have USB (**Figure 1**). The protocol operates at a 31,250 baud rate and allows instruments to synchronize, play together, or control synthesizers and computers. It's lightweight, flexible, and integral to digital music production. See more about MIDI at [1].

Why I Want My Raspberry Pi to Make Sounds

Back in 1994, I bought a Roland JV-30 keyboard/synthesizer. It wasn't super high-end, but it did let me play the keyboard and let me hear what I was playing (via external speakers or headphones — it had no built-in speakers). It was also my first instrument with MIDI.

It so happened that in that same year, I got my first "multimedia" PC, meaning it had an actual sound card capable of playing stereo, 16-bit digital audio, rather than relying on the motherboard's digital On/Off signals from a single pin to provide beeps and buzzes to an internal speaker [2]. The sound card fed a couple of powered speakers next to the monitor, and it could also play CDs directly from its double-speed CD-ROM drive!

That certainly made the game of DOOM, which released in December of 1993, a lot more immersive to play (although early versions of DOOM were mono, not stereo).

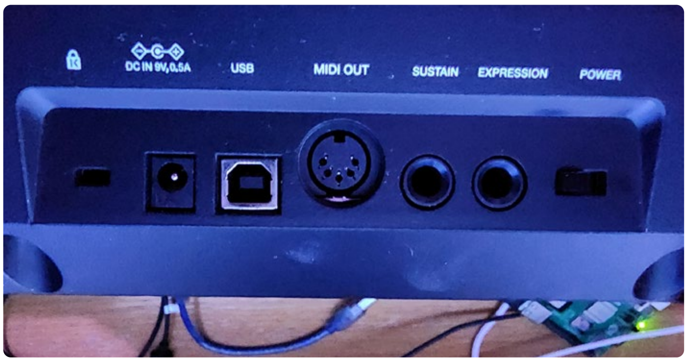


Figure 1: The M-Audio Keystation 88 MK3 has both USB and legacy 5-pin DIN connectors for its MIDI traffic.

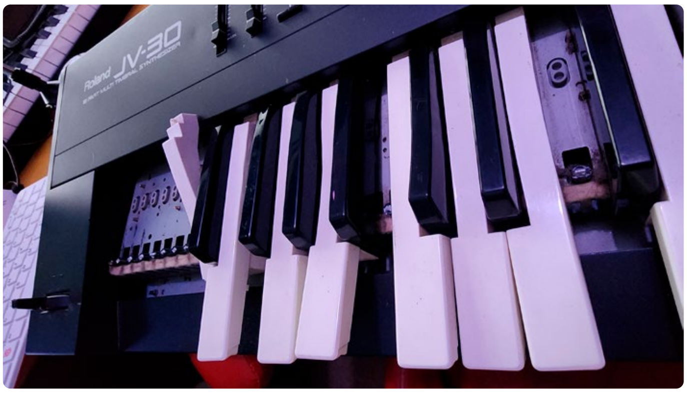


Figure 2: A badly damaged Roland JV-30. Viewer discretion is advised.

As was typical, my sound card had 3.5 mm minijack connectors for stereo line in, stereo line out, and a mono microphone input, but there was also another standard connector: the 15-pin joystick connector. Apart from the anomaly that sound cards were responsible for connecting PCs to joysticks, that connector had extra capabilities: MIDI input and output, which used only two of the 15 pins.

Ah yes, instead of using the cheap FM synthesis on my Media Vision Pro AudioSpectrum sound card [3] to play the music during a DOOM match, I could tell the game to send its music out via the MIDI output, and the Roland would render much higher quality music. Feeding the audio output from the keyboard back into the sound card's audio input, we could have all the game's audio come from the same pair of speakers.

So, that showed off the keyboard's synth responding to outputs from the PC, but, later on I got some music production software, such as MidiSoft Recording Session [4] that also allowed key presses from the keyboard to drive recordings and arrangement on the PC side, so it was a two-way street.

A couple of decades later, a self-made disaster struck. Rushing on a school morning, I tried to dry a damp jersey for my daughter by laying it over the conveniently located Roland JV-30 and using a hairdryer to get the job done. I had zero idea how sensitive the plastic keys were, and while the jersey was nice and dry at the end of my mission, when I lifted it off the keyboard, I was met with a disaster: All of the keys in the region were bent, buckled, and irreparable (**Figure 2**). I thought of buying new spare keys, but there were so many damaged that it would have cost less just to replace the keyboard.

I left it for a few years, but I was getting an itch to play again. I didn't want to spend a fortune on a stage-worthy keyboard/synthesizer, and I also wanted a full complement of 88 keys (like a piano), because the lack of keys down on the bass end (left-hand side) of the Roland JV-30 was a bit limiting for me in the past. I noticed that keyboards without built-in synthesizers were a lot cheaper, and I'd be able to get 88 keys for a reasonable price if I just bought a keyboard.

So, I opted for an M-Audio Keystation 88 MK3 [5], which is precisely that: a keyboard. Just like the one I'm typing on now, it has a USB interface, zero circuitry to create any kind of sound, uses very little power (so, while it has a power input, it can also be powered directly via USB), and simply sends details of every key I press (and subsequently let go of) back to the computer.

Then I discovered that the music keyboard's inability to play sound directly was more cumbersome than I expected. I found the process of connecting it to my PC every time and opening some complicated software unwieldy. After considering various solutions — such as upgrading to a keyboard/synthesizer, using my broken keyboard as a synthesizer unit for the new keyboard, or using a modular external synth (which you can see in **Figure 3**), I felt that all of these options took up too much space, used too much power, and were too heavy.

I thought of the Raspberry Pi, a compact option that might solve the problem, even though I doubted its compatibility with complex music software like you'd get on Windows or Mac, but I thought it was worth a try.

Having done some research, I found out that you don't even need slick, complicated music production software. You don't even need a monitor! If all you want to do is play your keyboard and have the computer synthesize the sounds you're going to hear, the Raspberry Pi is more than capable of doing that headlessly (i.e., without any need for monitor, keyboard, or mouse)! The Raspberry Pi is tiny enough to stick to the back of the music keyboard, and the keyboard uses so little power that you can power it directly from the Raspberry Pi's USB connector. Just think of the simple connection flow: Raspberry Pi Power Supply → Raspberry Pi → music keyboard powered by USB. That's all you need (well, almost — we still need a way to hear the sound, but we'll get to that).

Which Raspberry Pi?

While I used the Raspberry Pi 5 for my first experiments, the Raspberry Pi 4 is more than capable of doing what we need it to do in this context. In fact, the Raspberry Pi 4 and predecessors have one distinct advantage: They already have an analog audio output via a 3.5 mm minijack, whereas the Raspberry Pi 5, by default, pushes audio out via one of its HDMI connectors to a monitor.

Initially, for setup purposes, I used the Raspberry Pi Monitor, which not only has two tiny built-in speakers, but also a headphone output, which proved to be extremely useful — just plug in my studio monitor

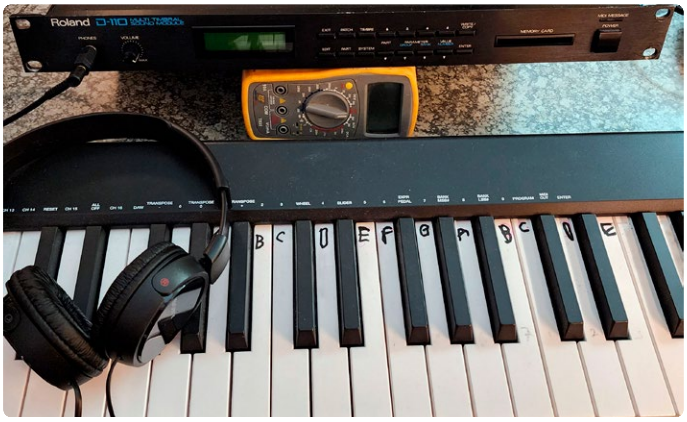


Figure 3: The 1988 Roland D-110 synthesizer promises "80s in a box!" But it sure isn't convenient.

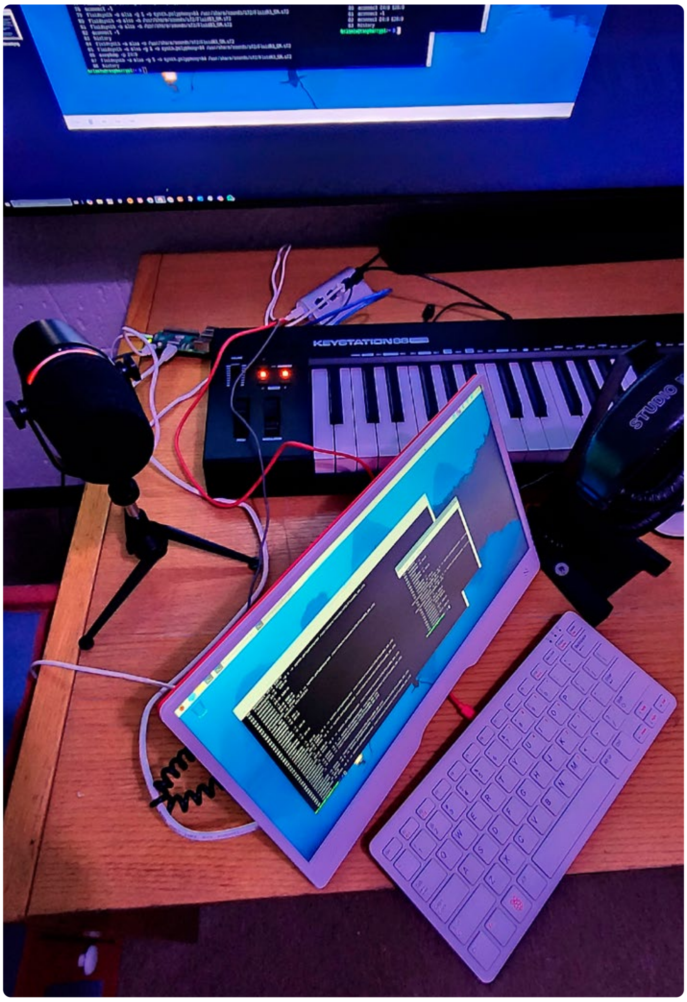


Figure 4: My test system up and running: Raspberry Pi, Raspberry Pi Monitor, keyboard, mouse, headphones and music keyboard connected via USB.

headphones, and I was set to play music and hear it at the same time — what a concept. Throw in the Raspberry Pi Keyboard and a mouse, and we have a little test system running (**Figure 4**).

Setting Up MIDI Playback on the Raspberry Pi

Here's the process I followed to get a Raspberry Pi 5 listening to my Keystation's activity and making music from it.

Firstly, connect the Raspberry Pi to the mouse, monitor, and (typing) keyboard. Then, connect a cable between the Raspberry Pi's USB-A and the music keyboard's USB-B connector (just like a printer). Finally, power up the Raspberry Pi and wait for it to boot into its desktop GUI (for now).

Open the *Terminal* application and start with the usual step of making sure all the software is updated, then reboot:

```
sudo apt update && sudo apt full-upgrade
sudo reboot
```

Once the board is back from its reboot, open the terminal again, and we'll turn to the *aconnect* program to give us some information about our setup. *aconnect* is a Linux command-line tool used to manage MIDI connections. It allows us to list MIDI devices, connect input devices (such as keyboards) to output devices (such as synthesizers or software), and dynamically route MIDI signals. This makes it essential for linking MIDI hardware or virtual ports in real-time during music production or performance.

The best part is that *aconnect* is included in Raspberry Pi OS, so just run

```
aconnect -l
```

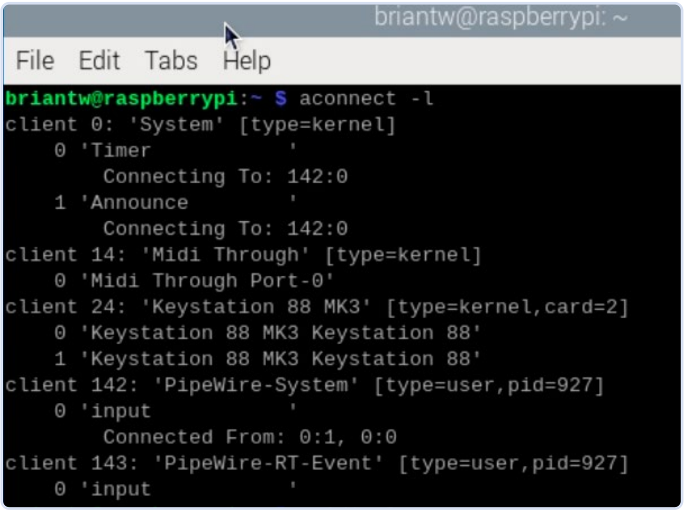
to get a list of detected devices. You can see the output I got in **Figure 5**. Under `client 24`, you see my keyboard listed as `'Keystation 88 MK3 Keystation 88'`. It has two ports, but port 0 is the one of significance here.

Great, so our keyboard, connected via USB, is detected. What we need now is some software that can turn the keyboard's low-bitrate digital chatter into music. *FluidSynth* is my go-to right now [6]. FluidSynth is an open-source software synthesizer that converts MIDI data into audio using SoundFont files. SoundFonts are collections of audio samples and instrument definitions that allow FluidSynth to generate realistic, high-quality sound. It operates in real-time, making it suitable for live performances, music production, and MIDI playback. It's widely used due to its flexibility and ability to produce professional-grade audio on various platforms, including Linux, macOS, and Windows. It boasts low-latency performance, making it ideal for interactive applications or live input from MIDI controllers. It also has a command-line interface and integration options, so it can be a powerful "Swiss Army knife" for MIDI debugging, composition, and performance.

Install it like this:

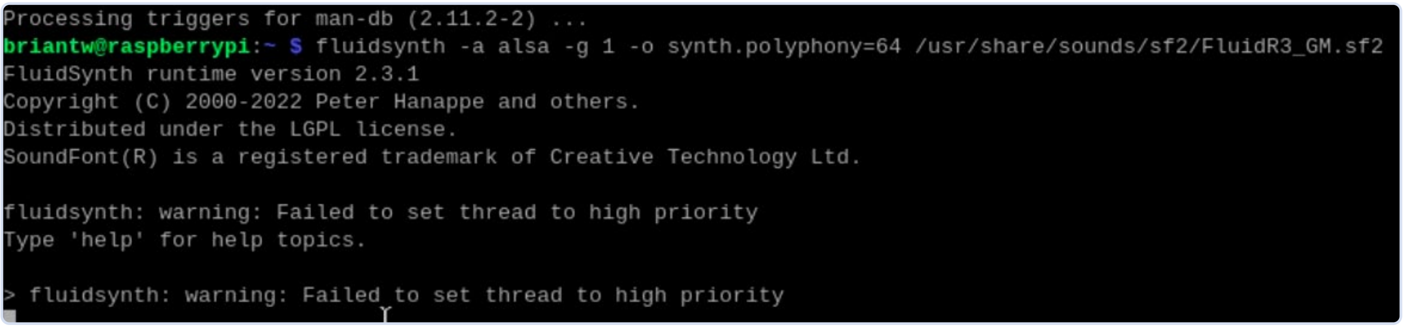
```
sudo apt install fluidsynth
```

Prepare to wait a couple of minutes, as it will likely download many files from the Bookworm repository. Once it's done, test it using:



```
briantw@raspberrypi:~ $ aconnect -l
client 0: 'System' [type=kernel]
    0 'Timer           '
        Connecting To: 142:0
    1 'Announce        '
        Connecting To: 142:0
client 14: 'Midi Through' [type=kernel]
    0 'Midi Through Port-0'
client 24: 'Keystation 88 MK3' [type=kernel,card=2]
    0 'Keystation 88 MK3 Keystation 88'
    1 'Keystation 88 MK3 Keystation 88'
client 142: 'PipeWire-System' [type=user,pid=927]
    0 'input           '
        Connected From: 0:1, 0:0
client 143: 'PipeWire-RT-Event' [type=user,pid=927]
    0 'input           '
```

Figure 5: `aconnect-l` shows the MIDI port configurations.



```
Processing triggers for man-db (2.11.2-2) ...
briantw@raspberrypi:~ $ fluidsynth -a alsa -g 1 -o synth.polyphony=64 /usr/share/sounds/sf2/FluidR3_GM.sf2
FluidSynth runtime version 2.3.1
Copyright (C) 2000-2022 Peter Hanappe and others.
Distributed under the LGPL license.
SoundFont(R) is a registered trademark of Creative Technology Ltd.

fluidsynth: warning: Failed to set thread to high priority
Type 'help' for help topics.

> fluidsynth: warning: Failed to set thread to high priority
```

Figure 6: FluidSynth's warning doesn't stop it from entering its interactive mode.

```
fluidsynth -a alsa -g 1 -o synth.polyphony=64 /usr/share/sounds/sf2/FluidR3_GM.sf2
```

In my case, FluidSynth complained about failing to "set thread to high priority" (**Figure 6**). It tries to do this to minimize latency and improve real-time audio performance. This is merely a warning, but FluidSynth is indeed now running in "interactive mode," so there's nothing to worry about. The "high priority" preference may be an issue on older Raspberry Pis, but given that I'm running on a Raspberry Pi 5, I discerned no noticeable lag. In fact, I was presently surprised at how responsive the whole system was. So, I moved on to the next step.

At that point, I was supposed to be able to play the keyboard and hear audio. Deafening silence. I tried telling FluidSynth to play a note manually. As mentioned, if it's already running in interactive mode, you just tell it what to do by typing a command. For example:

```
noteon 0 60 100
```

This tells it to play the Middle C note (60) at a velocity of 100 on channel 0. But still, no audio. Sigh.

After a bit of troubleshooting, it turned out that the Raspberry Pi's audio was being channelled to one of its two HDMI outputs — of course not the one I was using. Loath to do more software configuration, I just did a clean shutdown:

```
sudo shutdown
```

Then I plugged the HDMI cable into the other Micro HDMI output, and powered back up again. After starting FluidSynth again, the `noteon` command yielded the expected result and made a sound.

However, still no sound when playing the keyboard. Ah yes, there is another step: We need to route the keyboard's port (as listed in `aconnect -l`) with FluidSynth's port. From Figure 5, we saw that our keyboard was client 24, port 0. We just need to connect that (in software) to FluidSynth's port. And now that FluidSynth is installed, running the `aconnect -l` command tells us that FluidSynth is client 128, port 0. Let's connect them:

```
aconnect 24:0 128:0
```

Time to run the `fluidsynth` command again with bated breath and... Nope. `noteon` works, but still nothing when I play the keyboard. OK, I have to verify that MIDI data is, in fact, coming in to the system. Let's use the ***aseqdump*** tool. We'll first confirm that it recognizes the known ports:

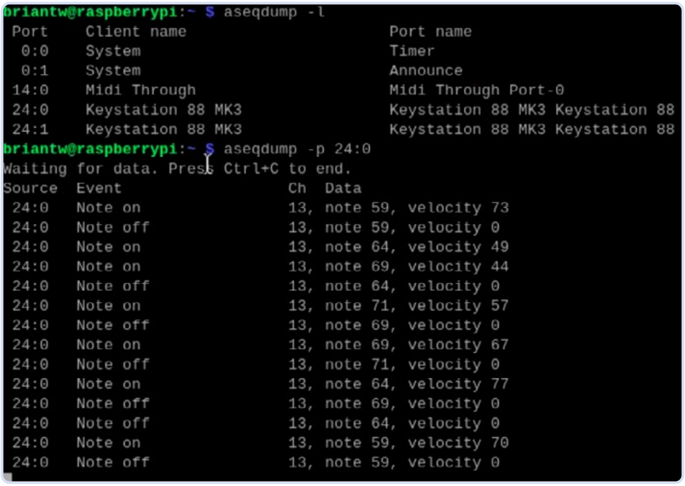
```
aseqdump -l
```

Yes indeed, it confirms that the keyboard is at 24:0. Now let's monitor any data that comes from the keyboard as we play a few notes:

```
aseqdump -p 24:0
```

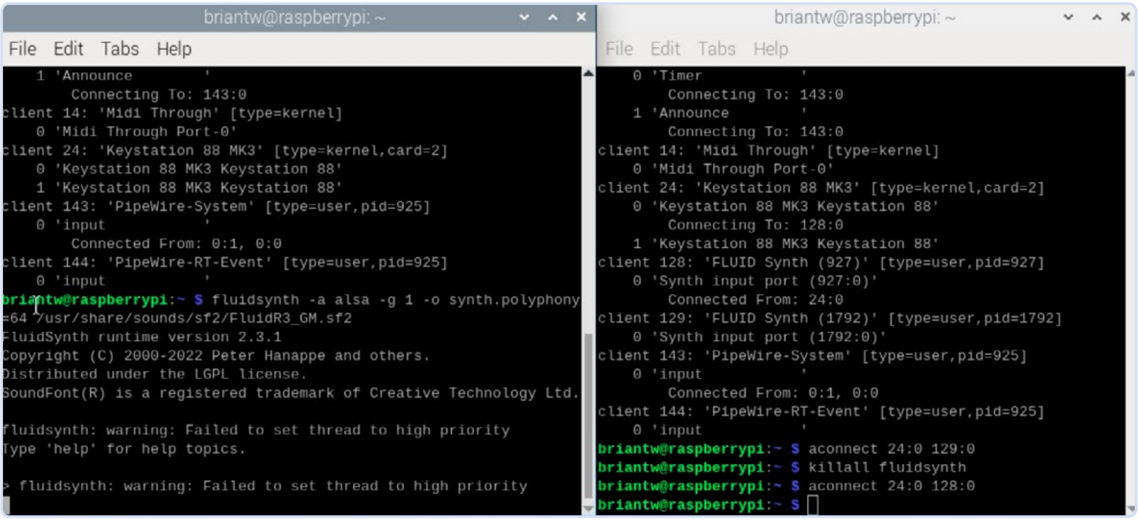
You can see the output in **Figure 7**. Well, the notes are coming in. (Side challenge: The first person to email me with the name of the tune gets a €20 Elektor Store voucher!)

Eventually, I got suspicious of FluidSynth — I found it showing up on port 128:0 before I even started it (even immediately after a reboot). That's fine, but what if I did explicitly start it from the terminal? To test this, I started FluidSynth as normal, then opened a separate, new terminal window and ran `aconnect -l` again. Well, what do



```
briantw@raspberrypi:~ $ aseqdump -l
 Port    Client name                      Port name
  0:0    System                           Timer
  0:1    System                           Announce
 14:0    Midi Through                     Midi Through Port-0
 24:0    Keystation 88 MK3                Keystation 88 MK3 Keystation 88
 24:1    Keystation 88 MK3                Keystation 88 MK3 Keystation 88
briantw@raspberrypi:~ $ aseqdump -p 24:0
Waiting for data. Press Ctrl+C to end.
Source  Event                  Ch  Data
 24:0   Note on                13, note 59, velocity 73
 24:0   Note off               13, note 59, velocity 0
 24:0   Note on                13, note 64, velocity 49
 24:0   Note on                13, note 69, velocity 44
 24:0   Note off               13, note 64, velocity 0
 24:0   Note on                13, note 71, velocity 57
 24:0   Note off               13, note 69, velocity 0
 24:0   Note on                13, note 69, velocity 67
 24:0   Note off               13, note 71, velocity 0
 24:0   Note on                13, note 64, velocity 77
 24:0   Note off               13, note 69, velocity 0
 24:0   Note off               13, note 64, velocity 0
 24:0   Note on                13, note 59, velocity 70
 24:0   Note off               13, note 59, velocity 0
```

Figure 7: `aseqdump -l` tells you exactly what traffic is coming in via the MIDI port.



```
    1 'Announce          '
        Connecting To: 143:0
client 14: 'Midi Through' [type=kernel]
    0 'Midi Through Port-0'
client 24: 'Keystation 88 MK3' [type=kernel,card=2]
    0 'Keystation 88 MK3 Keystation 88'
    1 'Keystation 88 MK3 Keystation 88'
client 143: 'PipeWire-System' [type=user,pid=925]
    0 'input            '
        Connected From: 0:1, 0:0
client 144: 'PipeWire-RT-Event' [type=user,pid=925]
    0 'input            '
briantw@raspberrypi:~ $ fluidsynth -a alsa -g 1 -o synth.polyphony
=64 /usr/share/sounds/sf2/FluidR3_GM.sf2
FluidSynth runtime version 2.3.1
Copyright (C) 2000-2022 Peter Hanappe and others.
Distributed under the LGPL license.
SoundFont(R) is a registered trademark of Creative Technology Ltd.

fluidsynth: warning: Failed to set thread to high priority
Type 'help' for help topics.

> fluidsynth: warning: Failed to set thread to high priority
```

```
    0 'Timer            '
        Connecting To: 143:0
    1 'Announce         '
        Connecting To: 143:0
client 14: 'Midi Through' [type=kernel]
    0 'Midi Through Port-0'
client 24: 'Keystation 88 MK3' [type=kernel,card=2]
    0 'Keystation 88 MK3 Keystation 88'
        Connecting To: 128:0
    1 'Keystation 88 MK3 Keystation 88'
client 128: 'FLUID Synth (927)' [type=user,pid=927]
    0 'Synth input port (927:0)'
        Connected From: 24:0
client 129: 'FLUID Synth (1792)' [type=user,pid=1792]
    0 'Synth input port (1792:0)'
client 143: 'PipeWire-System' [type=user,pid=925]
    0 'input            '
        Connected From: 0:1, 0:0
client 144: 'PipeWire-RT-Event' [type=user,pid=925]
    0 'input            '
briantw@raspberrypi:~ $ aconnect 24:0 129:0
briantw@raspberrypi:~ $ killall fluidsynth
briantw@raspberrypi:~ $ aconnect 24:0 128:0
briantw@raspberrypi:~ $
```

Figure 8: Using two terminal windows, while not strictly necessary, helped me track down the extra process.

you know, there was a new instance of FluidSynth operating on port 129:0 (see **Figure 8**). So, in the new window, I told *aconnect* to route to 129:0 instead of 128:0.

```
aconnect 24:0 129:0
```

I tickled a few keys, and was immediately greeted by the glorious sounds of FluidSynth's default piano. Finally! I'm not sure what I could have done wrong to get FluidSynth showing up on two different ports simultaneously — I never added any startup configuration or scripts to the Raspberry Pi earlier.

To restore default behavior, where I could use ports 24:0 and 128:0, I just killed all instances of FluidSynth from the new terminal window with

```
killall fluidsynth
```

and watched the process die in the original window. Then I ran `aconnect 24:0 128:0`, started FluidSynth again in the original window (just *up-arrow* followed by *Enter*, of course), and now it's working as expected. Wow, it "just works."

Getting Ready for More

Now that we have our gadgets talking and making sounds, we have the potential to write our own scripts to take output from the keyboard and turn it into sound using available tools. But, can we take it to the edge, i.e., doing some AI processing on our local hardware? I don't know how much brain power it really takes a computer to fiddle with a few plaintext MIDI messages containing a little bit of information. Stay tuned!

240714-01

Questions or Comments?

We'd love to hear about how you are using AI and how it has affected you over the past two years. If you have questions or comments, email me at brian.williams@elektor.com. You can also catch me on Elektor Engineering Insights each month on YouTube, and you can find me @briantw on X.

About the Author

Brian Tristam Williams has been fascinated with computers and electronics since he got his first "microcomputer" at age 10. His journey with Elektor Magazine began when he bought his first issue at 16, and since then, he's been following the world of electronics and computers, constantly exploring and learning. He started working at Elektor in 2010, and nowadays, he's keen on keeping up with the newest trends in tech, particularly focusing on AI and single-board computers such as Raspberry Pi.

Related Products

- **Pimoroni Piano HAT for Raspberry Pi**
  www.elektor.com/20552
- **Raspberry Pi 5 Ultimate Starter Kit (4 GB)**
  www.elektor.com/20720

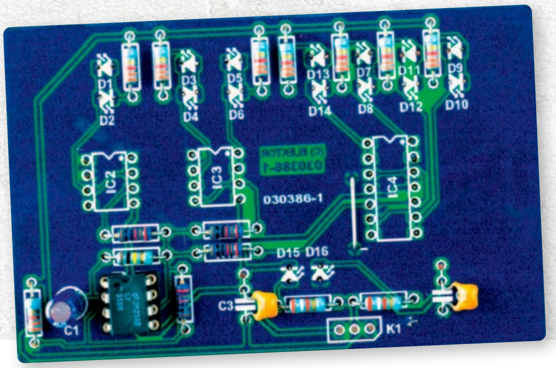
WEB LINKS

[1] What Is MIDI?: https://instructables.com/What-is-MIDI
[2] PC speaker: https://en.wikipedia.org/wiki/PC_speaker
[3] Media Vision Pro AudioSpectrum: https://en.wikipedia.org/wiki/Media_Vision_Pro_AudioSpectrum
[4] Demonstration of MidiSoft on Reddit: https://tinyurl.com/redditmidisoft
[5] M-Audio: Keystation 88 MK3: https://tinyurl.com/keystation88
[6] FluidSynth project wiki: https://github.com/FluidSynth/fluidsynth/wiki

Err-lectronics

Corrections, Updates, and Readers' Letters

Compiled by Jean-François Simon (Elektor)



Opamp Tester

Elektor 3/2005, p. 74 (030386)

I have noticed an error in the circuit. The schematic shows the power to IC4 as: +Ve on pin 11 and -Ve on pin 4. These are incorrect and need to be transposed.

Nigel Bowers

Thank you for your email. You're absolutely correct! I'm sorry about this error. Unfortunately the error is present in both the schematic and the PCB. Fortunately, it shouldn't be too difficult to correct the wiring. Pins 4 and 11 can be isolated from the tracks with an x-acto knife and connected correctly to +9 V and -9 V with a few solder bridges, as these power rails are close by. Also, note that the DIP sockets meant to hold the Opamps Under Test should be soldered on the back (copper) side of the PCB. This way the PCB can be mounted flush against a front panel that you may build for the tester, with the sockets accessible through holes in the panel. The LEDs are mounted on the back for the same reason. Enjoy the build!

Jean-François Simon (Elektor)

Low Voltage Converter for LED

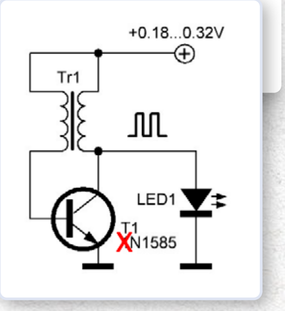
M. A. Shustov, *Electronic Circuits For All* (Elektor 2017), p. 35

I have a question about Shustov's book *Electronic Circuits for All*. The section on LED drivers is interesting. One circuit claims to run an LED on just a few hundred mV (e.g., p. 35). It needs a germanium transistor for these very low voltage circuits and more than once they use what they call a 1N1585. To me, that is a diode. Is this a typo or a mistake in translation from a Russian code?

Jonathan Hare (United Kingdom)

This is most probably a typo that went unnoticed. We are sorry about that! Thank you for pointing it out. Surely the authors meant 2N1585 instead. That one may be slightly difficult to buy, but it's supposedly similar to the GT311 which is available on eBay, but I don't have experience with either. Good luck with your experiments!

Jean-François Simon (Elektor)



Ideas or Feedback?
Got a bright idea or valuable feedback for Elektor? Reach out to us at editor@elektor.com. We're eager to hear from you!

Electronic Load Resistor

Elektor 1-2/2025, p. 94 (240201)

How can it be ensured that all four MOSFETs always draw the same partial current at every adjustment of the load current? Were all four MOSFETs selected in this regard, or measured in the finished device?

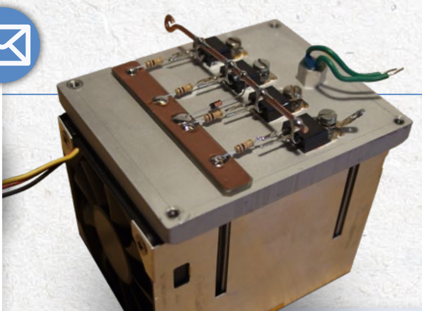
Henning Weddig (Germany)

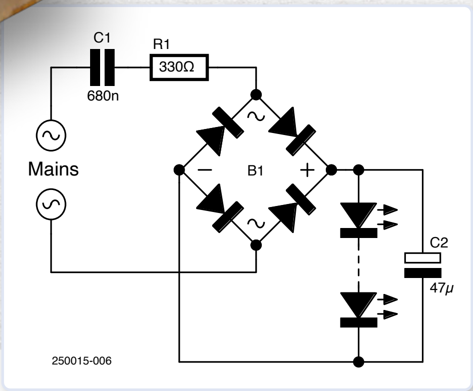
With this arrangement, there is no way to ensure that the current is perfectly equal in each of the four MOSFETs. If you want to recreate the setup, you can use four identical MOSFETs (preferably bought at the same time from the same place, so there is some probability that they come from the same batch and have similar properties), or you can match them by checking that they let through similar drain-source currents for a fixed gate voltage. This can be done using two regulated power supplies and a multimeter, or better yet, a transistor curve tracer if you have access to one. Indeed, paralleling MOSFETs is a complicated subject. Commercial electronic loads often have multiple current-sensing shunts (one per MOSFET) and multiple gate-driving op-amps (also one per MOSFET) to allow more control over current distribution.

Jean-François Simon (Elektor)

In my experience, the parallel connection of MOSFETs is significantly less problematic than with bipolar transistors. This configuration is also commonly found in commercial circuits. Unlike bipolar transistors, FETs have a positive temperature coefficient. This means that their on-resistance increases as temperature rises. If one FET draws more current than its counterparts, it heats up, its resistance increases, and consequently, the current through it decreases, along with the power dissipation and temperature. The datasheet for the IRF740 includes the graph "Normalized On Resistance vs Temperature" on page 5, which should be similar to other FETs. This graph clearly shows the significant increase in $R_{DS(on)}$ with rising temperature. If the four FETs are thermally well-coupled (as is the case in my design), there is no issue to be expected, provided that the parallel-connected FETs come from the same batch. In the days of bipolar transistors, this was much more critical: one transistor might perform slightly better, draw more current, heat up more than the others, and, due to its negative temperature coefficient, draw even more current as it got hotter — eventually leading to its failure. In my case, the device successfully endured a test dissipating 120 W without issues.

Peter Grundmann (Author of the article)





Failing LEDs

Our workshops are, of course, lit with LED tubes, typically for about five hours a day. Unfortunately, I've noticed that longevity isn't great. I had two failures of "Müller Licht" LED fixtures within a year, as well as one failure from "Osram", and one from "Philips". With the Müller Licht ones, one of the three LED strips failed first, then the remaining two got a bit brighter, and shortly afterward, it went completely dark. The Osram and Philips ones just stopped working abruptly. I also have an E27 LED lamp, it flickered at first and then stopped working. I opened it up, and the LEDs themselves are fine. What are your experiences?

Alfred Rosenkränzer (Germany)

Yes, unfortunately, that's how it is. Most of the time, it's the power supply that fails because it's not optimally cooled due to lack of space. For Philips tubes, I've sometimes replaced the miniature fuse (which blew without other damage) with a stronger one, and that was enough to fix it. With others, it's not so simple. Especially when LEDs themselves fail, it means their cooling is suboptimal. There are models with a kind of aluminum cooling fin on the back. I'd prefer those. For E27 lamps with a switch-mode power supply, it's likely that the power supply is defective as well. A cost-effective replacement could be a bridge rectifier with a series capacitor to limit the current, as shown in the small circuit diagram.

Thomas Scherer (Elektor)

Remote Water Heater Monitor

Elektor Circuit Special 2024, p. 50 (240039)

I have a few comments on the article. Capacitor C1 (150 nF/400 V) should be at least class X2. A diode (1N4148 or similar) should be added in anti-parallel to Q1 to protect the base-emitter junction, by connecting the anode to the emitter and the cathode to the base. Current transformers should not be operated without a load to prevent the output voltage from rising too much, which would be the case with a negative half-wave without an additional diode. Finally, C2's charging current should be limited with a series resistor.

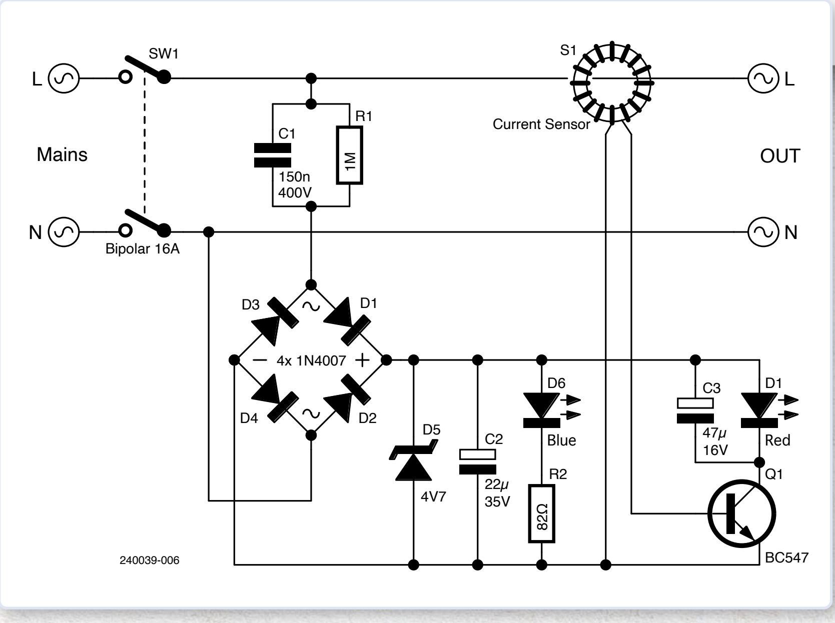
Frank Mammen (Germany)

There are always several ways of solving a given problem indeed! For capacitor C1, you're right, an X2 would probably be safer. For points 2 and 3, I can't comment. The author of the circuit told us that, in his case, the circuit had been working without a problem for many years, so it seems that for him the open-circuit voltage of the current transformer was not a problem in relation to the maximum emitter-base voltage of Q1. For your last point, it seems to me that the maximum current in capacitor C2 is in any case limited by the impedance of C1 (of the order of 21 kΩ at 50 Hz) to a value of around 15 mA, which seems reasonable to me.

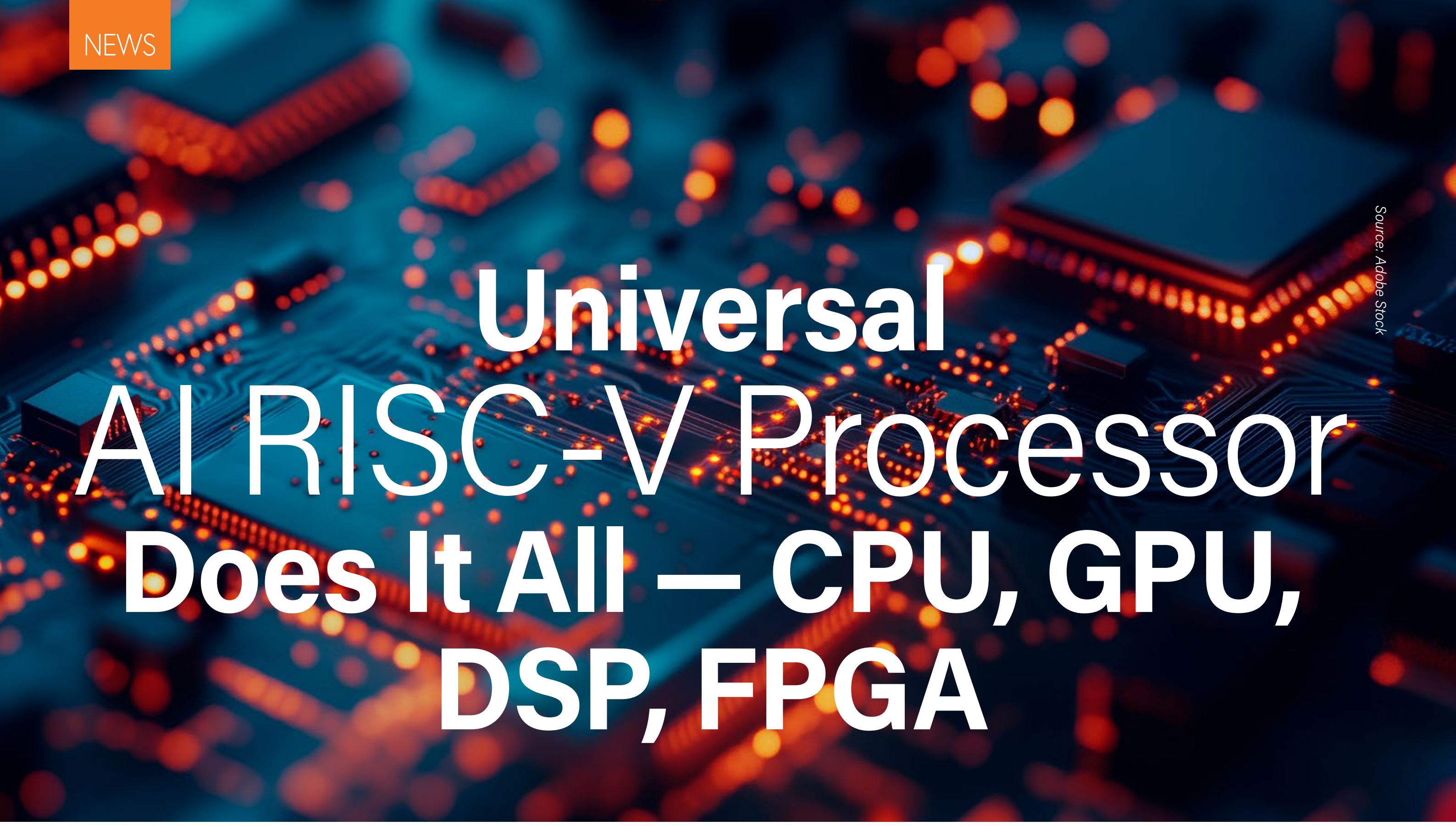
Jean-François Simon (Elektor)

Thank you for your comments. The device has been working for over 10 years without any problems. The toroid does not produce any harmful voltages / currents and no additional protective component is needed for the transistor. For C1 I have indicated the working voltage and the choice of an X2 is agreeable. As Jean-François points out, the current this capacitor lets through is very small, compatible with the characteristics of the limiting Zener.

Stefano Purchiaroni (Author of the article)



250015-01



Source: Adobe Stock

Universal AI RISC-V Processor Does It All — CPU, GPU, DSP, FPGA

By Jean-Pierre Joosting (eeNews Europe)

Ubitium has secured $3.7 million in seed funding to launch a universal RISC-V processor that eliminates the need for specialized chips, enabling advanced AI at no additional cost in embedded systems by reimagining IBM's 1967 processing approach.

For over half a century, general-purpose processors have been built on the Tomasulo algorithm, developed by IBM engineer Robert Tomasulo in 1967. It's a $500 billion industry built on specialised CPU, GPU and other chips for different computing tasks. Hardware startup Ubitium [1] has shattered this paradigm with a breakthrough universal RISC-V processor that handles all computing workloads on a single, efficient chip — unlocking simpler, smarter, and more cost-effective devices across industries — while revolutionizing a 57-year-old industry standard.

The new universal RISC-V processor does all functions — CPU, GPU, DSP, FPGA — in one chip, with one architecture.

The seed funding round, co-led by Runa Capital [2], Inflection [3], and KBC Focus Fund [4] will be used to develop the first prototypes and prepare initial development kits for customers, with the first chips planned for 2026.

Ubitium was founded by semiconductor veterans dedicated to revolutionizing processor architecture. CTO Martin Vorbach, who holds over 200 semiconductor patents licensed by major U.S. chip companies, spent 15 years developing this groundbreaking technology. Drawing from his pioneering work in reconfigurable computing, he created a workload-agnostic microarchitecture that allows the same transistors to be reused for different processing tasks — eliminating the need for multiple specialized cores and enabling AI at no additional cost. Working between Germany and Cupertino, California, Vorbach's innovation forms the foundation of Ubitium's mission.

Vorbach met CEO Hyun Shin Cho at the Karlsruhe Institute of Technology (KIT). After two decades of gaining insights across various industrial sectors, Cho reunited with Vorbach to commercialize the technology. Completing the team is Chairman Peter Weber, a veteran of Intel, Texas Instruments, and Dialog Semiconductor, who brings extensive industry expertise.

"The $500 billion processor industry is built on restrictive boundaries between computing tasks," says Hyun Shin Cho, CEO of Ubitium. "We're erasing those boundaries. Our universal RISC-V processor does it all — CPU, GPU, DSP, FPGA — in one chip, one architecture. This isn't an incremental improvement. It is a paradigm shift. This is the processor architecture the AI era demands."

"For too long, we've accepted that making devices intelligent means making them complex. Multiple processors or processor cores, multiple development teams, endless integration challenges — today, that changes. The universal RISC-V processor delivers workload-agnostic and AI-enabling compute capabilities to edge devices with a single chip, at a fraction of the cost to develop and manufacture compared to today's offerings."

With the semiconductor market projected to exceed $700 billion by 2025, Ubitium's technology initially targets embedded systems and robotics. By simplifying system architectures and reducing costs, the universal RISC-V processor makes advanced computing capabilities accessible across all industries without requiring specialized hardware for each application — enabling advanced AI at no additional cost.

Dmitry Galperin, a Berlin-based General Partner at Runa Capital, commented: “Ubitium’s unique approach to processor microarchitecture makes it possible to adapt to any type of workload — from simple control logic to massive parallel data flow processing.”

“What Ubitium brings will provide a real breakthrough to develop and launch any new product with embedded electronics. Their approach will reduce the cost as well as the complexity, allowing a much faster time-to-market. What previously required multiple teams to collaborate on hardware and software design now becomes purely a software project,” said Rudi Severijns, Investment Director at KBC Focus Fund.

> *The new universal RISC-V processor does all functions — CPU, GPU, DSP, FPGA — in one chip, with one architecture.*

Looking ahead, Ubitium plans to develop a complete portfolio of chips that vary in array size but share the same microarchitecture and software stack — enabling solutions from small embedded devices to high-performance computing systems. This super-scalable approach allows customers to scale their applications without changing their development process, while the workload-agnostic design ensures the processor can adapt to handle any computing task without specialized hardware modifications. The company’s goal is to establish its universal processor as the new standard that finally breaks down the cost and complexity barriers that have limited the deployment of advanced computing and AI capabilities across industries.

“We envision a future where every device operates autonomously, making intelligent decisions in real-time and transforming the way we interact with technology,” added Hyun Shin Cho.

240731-01

Editor’s Note: Jean-Pierre Joosting first reported on this in eeNews Europe, a publication in the Elektor network. www.eenewseurope.com/en/domain/eenews-embedded/

WEB LINKS

[1] Ubitium: https://ubitium.com

[2] Runa Capital: https://runacap.com

[3] Inflection: https://inflection.xyz

[4] KBC Focus Fund: https://www.kbcsecurities.com/en/investment-services/kbc-focus-fund.html

CEO Interview: **Ventiva's Thin and Cool Tech**

Carl Schlachte, chairman, president and CEO of Ventiva. (Source: Ventiva)

By Nick Flaherty (eeNews Europe)

After over a decade of development, miniature, silent cooling technology was reducing the thickness of laptop designs, shown at the CES 2025 show in Las Vegas.

Carl Schlachte, chairman, president, and CEO of the developer, Ventiva, is a veteran of the semiconductor industry, having worked at Motorola and ARM before taking over as CEO of the UK processor design company ARC. Back in 2011, he bought into a small US firm developing an innovative cooling technology.

Now in 2025, Ventiva [1] is showing a proof of concept for its ion cooling technology, replacing the fans in a laptop to show how the thickness of the design could be reduced by several millimetres with the potential to be used in other consumer designs.

"Between Dell, Intel, and Ventiva, we had 30 engineers working for over a year. This was a serious engineering effort. What was fascinating was that a community formed between the engineers," he tells *eeNews Europe*. "This is the thinnest laptop they have ever built, 2 mm thinner."

The technology of the 3-mm-high ICE9 blower is only one element. The technology for manufacturing and integration into designs has been as challenging. The latest version can address 40 W of thermal power, up from 25 W in earlier designs, and fits into the space previously occupied by the fan so that the motherboard design does not need to be changed. The control architecture also mimics a fan so that the software doesn't need to change.

"A lot of engineering went into making this easy to integrate, and that is hard," said Schlachte. "This is the benefit of having been around for a decade. We go out, test things in the market, and go back and re-engineer it. We initially went after spaces that couldn't be used in the laptop, such as the space around the hinge. Because of how it is built we can turn a corner, and we are not taking any more space."

"We spent a lot of time with designers to understand that to get great performance you have to rearchitect the design. We were using an existing Lunar Lake motherboard but they were investing in future products for the proof of concept and learning about the modeling and the airflow."

"The strategic vision I have is that in the next five to seven years, fans will be gone. If you go out to buy a laptop then, everyone will have the same reaction to fans that they had to hard drives that moved to solid state drives."

"That's what we are doing right now. We are taking out the existing blowers and putting our blowers in the exact same spot. That's not the best way to get the best performance out of the system. We take the exact same 4-pin connector and the system thinks it is talking to a fan, and the system needs more cooling it ramps up just like a fan. The economics clearly point that way by rethinking the thermals just a little bit."

"The blower is a 60 to 70 mm long device, and we designed it so it can be ganged up like Legos to provide more airflow."

"We use an electrohydrodynamic (EHD) flow, energizing a very thin wire in the housing, creating the ions that collide with the air to create the airflow. Making this low cost enough for consumers is the challenge. We use no novel materials, no rare earth elements, we use injection molded plastics, stamped metal and standard PCB manufacturing with a Microchip microcontroller so it's highly scalable," he said.

The company has its own factory in Malaysia, and it is getting ready to ramp production by the end of the year, along with second sourcing deals. The system that will ship to the laptop makers includes the heat pipe and the fins ready to replace the fan.

"Every piece of the manufacturing flow has been known for 40 years, the novelty went into the development of the software, the control of the power supply and the architecture of the blower to maximise the airflow," he said.

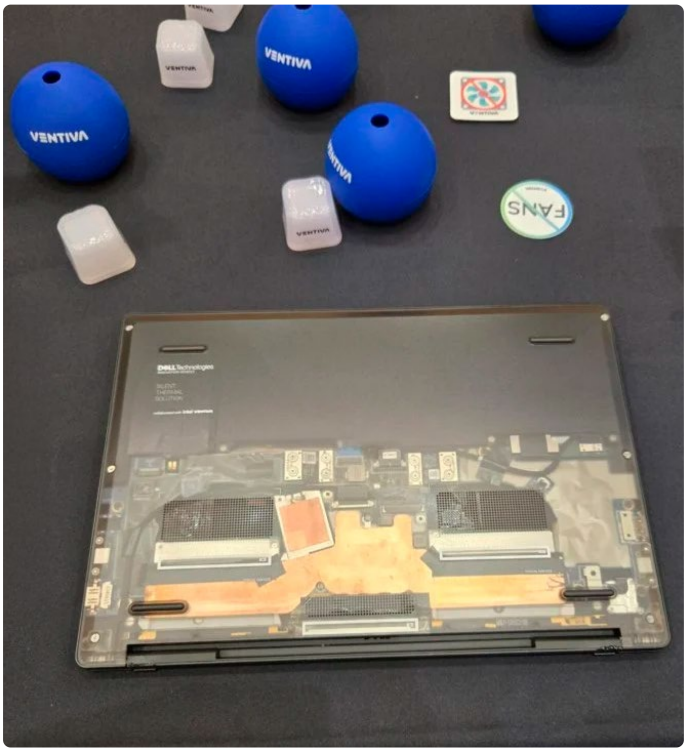
"I am the only connection back to 2011, the product is different, but the basic physics of the idea is the thing that has remained the same. My partner and I bought it, and scaled it back to bare metal, negotiated partnerships that helped pay for things."

"Even 18 months ago, we hit a product problem and I thought we were dead. So, going from that to the Intel keynote, with Dell people at our booth, is fantastic. We have relationships with all the top PC makers, but Dell is the furthest along. In six to eight months, we'll have more news to share," he said.

"We've spent the bulk of our time working with really smart laptop engineers, and our credibility will start to show up in the way these products are developed."

AI has been a boon for the company, as the higher energy consumption of AI in both the CPU and any accelerators has highlighted the need for more thermal management.

"The general figure of merit for large language models (LLMs) we use is an extra 10 W in heat load, so what has happened is that their problems have got worse overnight as everyone wants an AI PC that people want to talk to, and fans are loud. To get a blower quieter, you have to make it bigger and run it slower. For us, our technology is quiet no matter what. That converged on our technology, a random lovely accident that occurred at a time we had all the tools to tackle it," he said.



A proof-of-concept thin laptop developed by Dell and Intel using the Ventiva ICE9 cooling system. (Source: Ventiva)

The company also targets the $1000+ market as that has the margin and demand for thinner designs, as well as AI. He points to data that says 94% of the volume of these laptops is between 15 W and 40 W, hence the 40 W proof-of-concept design shown at CES.

"Laptop PCs are just the start. We have a counterintuitive view of the world — we don't see a play for us in the gaming laptop or data center. Instead, there are more compelling applications on the small side — in a handset, ultrathin tablets, at a certain level there is nothing else that will move air at this size. Giving a handset maker an extra 5 W of cooling is unheard of, especially if you add in wireless charging and the more performance requirements of AI."

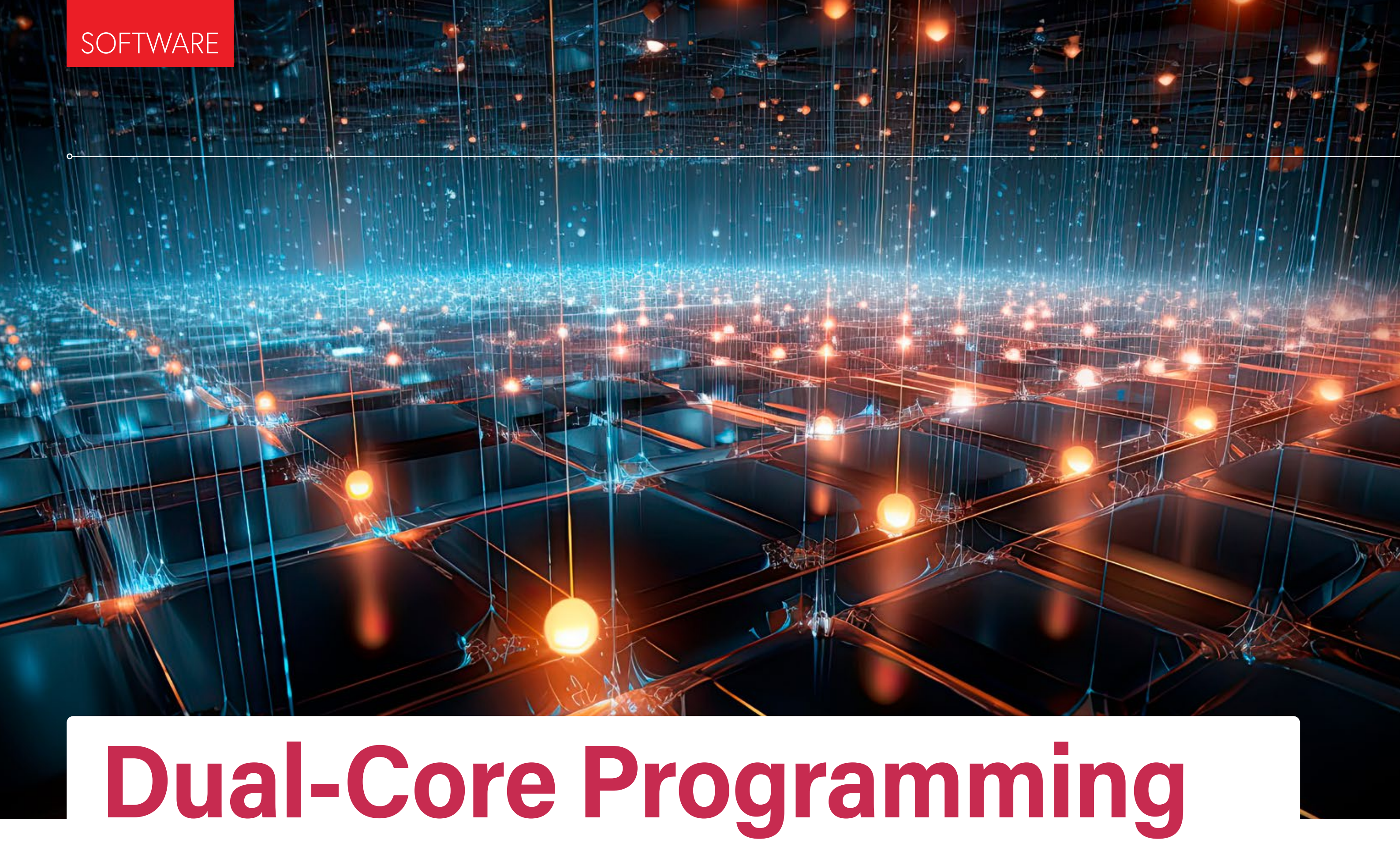
250043-01

Editor's Note

Nick Flaherty first reported on this in *eeNews Europe*, a publication in the Elektor network. *www.eenewseurope.com*

WEB LINK

[1] Ventiva: https://ventiva.com



Dual-Core Programming with a Raspberry Pi Pico

Venture Into the World of Parallel Programming

By Prof. Dr. Martin Ossmann (Germany)

The Raspberry Pi Pico is powered by the RP2040 processor, equipped with two CPU cores. This characteristic makes it an ideal platform for delving into the realm of "Parallel Programming." Through the utilization of tangible real-world examples in digital signal processing, this article illustrates the methodologies and considerations employed to extract maximum benefits from the dual-core architecture.

The examples showcased here employ various parallelization techniques. The acceleration of a program through the use of multiple cores is referred to as "Speedup" (abbreviated as S). With two cores, the maximum theoretical speedup is S = 2. The speedup score achieved by each example application will be provided in the following discussion.

Synchronization and the Volatile Attribute

When programming software to run on multi-core CPUs, it's often necessary for threads to synchronize their activity. In the simplest case, Core 0 can activate Core 1. This can be achieved easily using a `trigger` variable:

```
volatile int trigger ;

void core1do(){
   while(1){
      while(trigger==0){ } ;
      printf("trigger=%d\n",trigger) ;
      trigger=0 ;
    }
 }

void core0do(){
   int cnt=0 ;
   while(1){
      trigger=cnt++ ;
      sleep_ms(1000) ;
    }
 }
```

The variable trigger is used by both cores. To let the compiler know that the variable can change due to its use by multiple threads, it is assigned the volatile attribute. Without this attribute, the compiler might perform extensive code optimizations, potentially causing issues with the shared, parallel use of a variable by two threads.

Shared-Function Usage

The following example illustrates how Core 0 and Core 1 can simultaneously use the same function:

```
void delay(int nn, int cnt){
   volatile int k ;
   printf("core %d cnt=%d\n",sio_hw->cpuid,cnt) ;
   for(int k=0 ; k<nn ; k++){
      asm volatile( " nop\n" ) ;
      }
   }

void core0do(){
   int cnt=0 ;
   while(1){
      delay(10000000,cnt++) ;
      }
   }

void core1do(){
   int cnt=0 ;
   while(1){
      delay(20000000,cnt++) ;
      }
   }
```

The Function delay(nn,cnt) outputs a message indicating which core is active and the value of cnt. Then, it enters a wait loop with nn iterations using the local variable k.

Essentially, Core 0 and Core 1 do the same thing: they run an infinite loop. Within the loop, delay(..,..) is called. Core 0 uses nn = 10,000,000 as a parameter, while Core 1 uses nn = 20,000,000. Thus, Core 1 performs twice as many loop iterations as Core 0. The count of loop iterations is tracked in the variable cnt. Since both cores spend most of their time in the delay loop, they are mostly in the delay function simultaneously. How does this work? The code of the function is stored in memory only once. Therefore, both cores execute the same code. Since the code only involves read instructions, accessing the code alone does not cause a conflict.

Both cores naturally have their own program counter (PC) that addresses or steps through the code instructions. When a function is called, the following happens: the PC value is pushed onto the stack. The parameters are also placed on the stack. Space is then reserved on the stack to store local variables (here k). This collection of variables on the stack is often referred to as the *stack frame*.

The PC is now loaded with the start address of the called function. Both cores have their own stack; they also have their own stack frame. This means there will usually be two so-called incarnations of delay() using the same code but via separate stack frames. If delay were to use global variables or other global resources, for example, this separation would not exist, and parallel usage would be problematic. This example illustrates well what needs to be considered when parallel programming.

Core 0 Can Throttle Core 1

This next example illustrates how the two cores influence each other in ways that may not be immediately apparent.

```
void core1do(){
   asm volatile(
   "       nop        \n"
   "loop2: b     loop2\n") ;
   }

void ASMdelay625(){
   asm volatile( // 3*205=615 cycles
   "ldr r0, =#205\n"
   "loop1: sub r0,r0,#1\n" // 1 cycle
   "bne   loop1\n"
      // 2 cycles if loop taken, 1 if not
    ) ;
  }

void core0do(){
   while(1){
      gpio_put(GPIO2, 1) ;
      ASMdelay625() ; // approx 625 cycles
      gpio_put(GPIO2, 0) ;
      ASMdelay625() ;
      // without core1: 100.08055 kHz
      // with core1: 75.256 kHz
      }
   }
```

Core 1 is running a simple infinite loop. Core 0 also runs an infinite loop, but in addition sets GPIO2 high, calls a time delay of 625 cycles before it returns GPIO2 low and calls another 625 cycle delay. When Core 1 is not running, Core 0, (which is clocked at the Pico default clock frequency of 125 MHz), performs 100,000 loop iterations per second (125 MHz / (2 * 625 cycles)) which produces a square wave output signal of 100 kHz at GPIO2. When Core 1 is started, the throughput of Core 0 drops to produce a 75 kHz signal. Even though the two cores don't seem to have anything obvious to do with each other, they do influence each other. This is likely because there is only one memory interface used by both cores. This effect is often responsible for the achievable speedup factor of many of the programs used here being significantly below 2.

This example highlights the need to always be on the lookout for unexpected effects. Debugging parallel programs like this can be challenging.

FIFO Communication

You can use two implemented FIFOs to send data back and forth securely between the cores. There is a FIFO that works from Core 0 to Core 1 and another from Core 1 to Core 0. The advantage of the SDK's FIFOs is that they are termed "Thread-Safe," meaning they operate correctly even with parallel usage.

To test the FIFO's performance, we can implement a simple "server" on Core 1:

```
void core1do(){
   int buffer1 ;
   while(1){
      buffer1=multicore_fifo_pop_blocking() ;
      multicore_fifo_push_blocking(buffer1+1000)
      }
   }
```

It just waits for a value from the FIFO of Core 0 and sends this value back to Core 0 increased by 1,000.

Whilst this routine runs on Core 0:

```
void core0do(){
   for(int k=0 ; k<NN ; k++){
      multicore_fifo_push_blocking(k) ;
      buffer2=multicore_fifo_pop_blocking() ;
      if(buffer2!=k+1000){
       printf("communication error!\n") ;
      }
   }
```

The runtime of this routine was measured to find this program takes approximately 0.62 µs. Since twice 32 bits are transmitted during each run, this corresponds to a bit rate of about 104 Mbit/s. The following addition loop is used to compare this data transfer rate to the calculation speed of a Raspberry Pi Pico:

```
void doAdd(){
   int sum=0 ;
   for(int k=0 ; k<MM ; k++){
      sum +=k ; }
   result=sum ;
   }
```

The timing measurement results in a value of approximately 56 ns per loop iteration. If we let the server run the loop for N = 10 iterations, we have a computation time of about 560 ns and a communication time of about 620 ns. This results in a communication overhead in excess of 100%. The server, therefore, needs to provide a substantial amount of computing power to prevent communication overhead from impacting overall performance. When coding for parallel programming environments, it's advisable not to focus on too fine a level of programming granularity. In addition to FIFOs, Queues are also provided, offering more flexibility as they can send arbitrary data structures. Furthermore, the SDK offers various mechanisms for thread synchronization; it is worthwhile taking a closer look at the available SDK facilities.

SDR-Parallelization

The frontend of a software-defined radio (SDR) is often structured as shown in **Figure 1.** After multiplying the input signal with the Local Oscillator (LO) signal, you get the I-channel (In-Phase) and the Q-channel (Quadrature Phase) [1]. These two channels arise almost in parallel, making them ideal candidates for parallelization. Core 0 processes the data of the I-channel, and Core 1 handles the data of the Q-channel. Summation after mixing is still performed in the interrupt routine. The first parallelized part involves the implementation of Butterworth filters, which are performed with float precision. In **Figure 2**, you can see the timing of the low-pass filter evaluation.

The yellow curve is "high" when Core 0 calculates the I low-pass filter. The turquoise curve is "high" when Core 1 calculates the Q low-pass filter. Interestingly, calculation of the I component

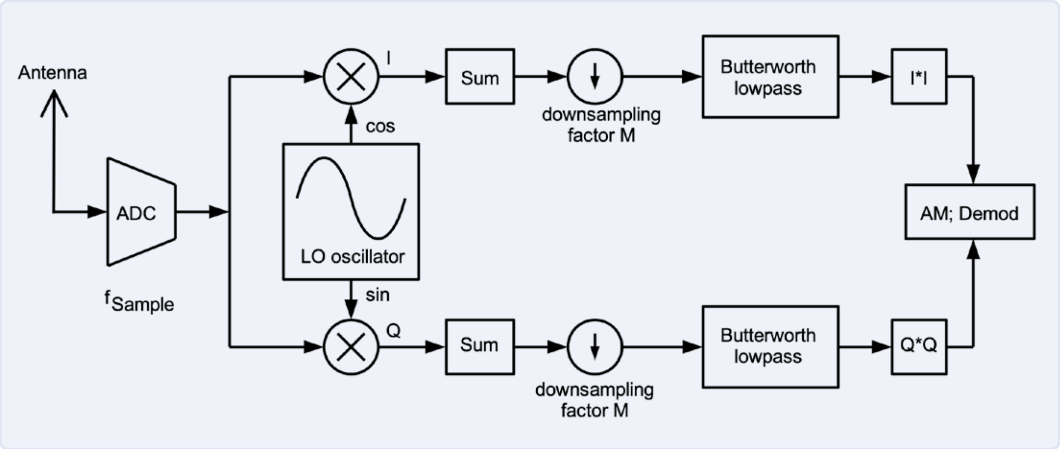


Figure 1: Signal flow diagram of an SDR frontend.

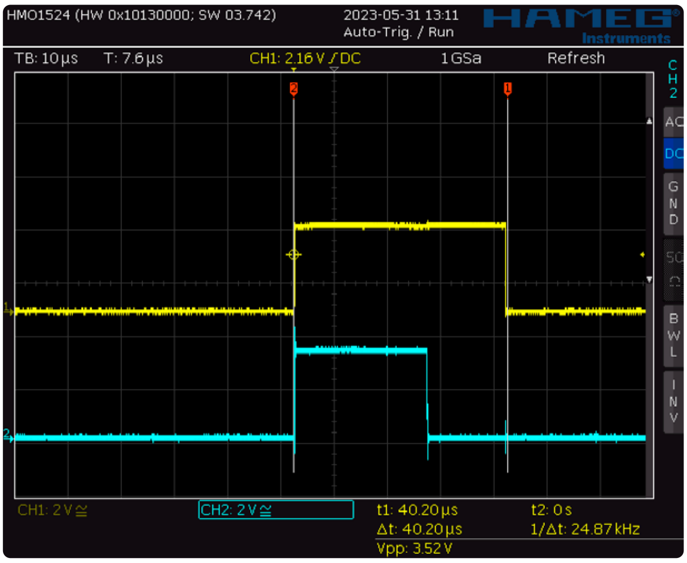


Figure 2: Execution of the I and Q low-pass routines.

takes significantly longer than for the Q. This is because Core 0 also executes the interrupt routine that processes the sampling. Therefore, Core 0 is not always available, and the calculations of the I-channel take accordingly longer.

Scalar Product With Loop-Splitting

Another example is the calculation of a scalar product. The parallelization is based on the following observation: A whole set of calculations is typically performed in a loop. If the individual loop elements can be calculated independently, the loop can be split into two parts and processed in parallel using the two cores. The formula for calculating the scalar product of two vectors *x* and *y* is as follows:

$$p=\sum_{k=1}^{N} x_k y_k$$

A routine called `skpLoop` in this program performs the summation of k_1 to k_2:

$$S(k_1,k_2)=\sum_{k=k_1}^{k_2} x_k y_k$$

In the non-parallel version, 1 to N are simply summed. In the parallel version, summations from 1 to N/2 and N/2 to N are performed in parallel.

$$p=S(1,N)=S(1,N/2)+S(N/2,N)$$

The end result is obtained by adding the two partial results. Core 0 and Core 1 simultaneously use the same summation loop `skpLoop(..,..)`. Since the loop is used by each core with separate sets of array elements x_k and y_k, there is no conflict. The speedup achieved here is S = 1.8. The programs are shown in **Listing 1**.

FIR Filters

Finite impulse response (FIR) filters are widely used for applications in digital signal processing. In an FIR filter of order N, the current output value is calculated by summing the last N input values, weighting the samples with the coefficients a_k of the filter impulse response. The formula is:

$$y_m=\sum_{k=0}^{N-1} a_k x_{m-k}$$

For the calculation, it's necessary to store the last N input values x_k. This can be most conveniently performed using a ring buffer.

Listing 1: Serial and parallel scalar product-calculation.

```
#define NN 10

float xk[NN] ;
float yk[NN] ;
float result ;

float skpLoop(int k1, int k2){
   float sum=0 ;
   for( int k=k1 ; k<k2 ; k++){
      sum=sum+xk[k]*yk[k] ;
      }
   return sum ;
   }

void core1Do(){
   while(1){
      multicore_fifo_pop_blocking() ;
      result=skpLoop(NN/2,NN) ;
      multicore_fifo_push_blocking(456) ;
      }
  }

void parallelSkp(){
   float sum ;
   multicore_fifo_push_blocking(123) ;
   sum=skpLoop(0,NN/2) ;
   multicore_fifo_pop_blocking() ;
   sum=sum+result ;
   printf("parallel sum=%10.3f\n",sum) ;
   sleep_us(10) ;
   }

void seriellSkp(){
   float sum ;
   sum=skpLoop(0,NN) ;
   printf("seriell sum=%10.3f\n",sum) ;
   sleep_us(10) ;
   }
```

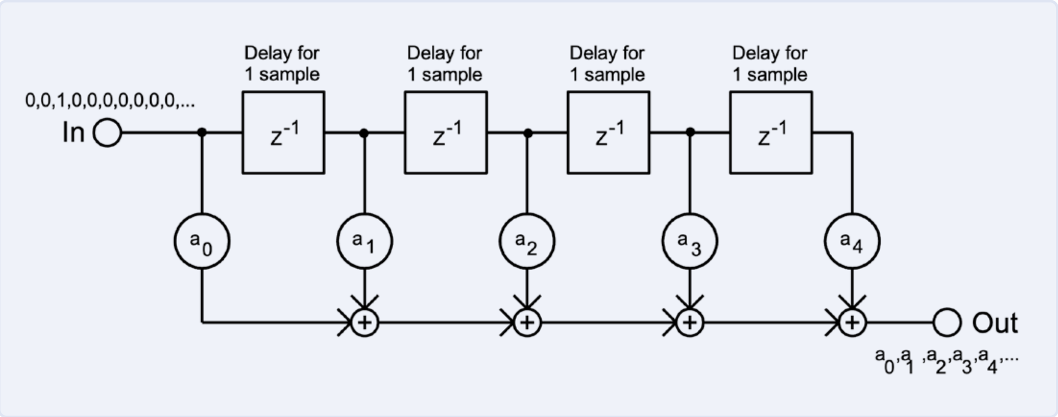


Figure 3: Signal flow in the FIR filter.

The parallelization can be carried out in principle in the same way as with the scalar product, using loop splitting. You just need to manage the ring buffer and correctly index its contents. The loop is represented here:

```
float doSumLoop(int k1, int k2, int readPtr){
   float sum=0 ;
   for( int k=k1 ; k<k2 ; k++){
      sum=sum+ak[k]*xBuf[readPtr]
      readPtr-- ;
      if(readPtr<0){ readPtr=N-1 ; }
      }
   return sum ;
   }
```

The parameter `readPtr` points to the current start position in the ring buffer `xBuf`. The parallel FIR calculation can be seen below. The speedup factor achieved here is S = 1.7.

```
float FIRdo(int x){
   xBuf[writePtr]=x ;
   int readPtr1=writePtr ;
   writePtr++ ;
   if(writePtr>=N){ writePtr=0 ; }
   int readPtr2=readPtr1-N/2 ;
   if(readPtr2<0){ readPtr2=readPtr2+N ; }
```

Program Development

Troubleshooting parallelized programs is often challenging and complex. It has proven useful to initially write a version of the program to run in a more flexible environment, such as the free, open-source Processing environment (reference version), and debug it there. During this process, you also generate test data for parallelization. Once this version of the software has been debugged it can be ported to the target system to initially use the same test data there. For troubleshooting, you can then compare the two versions.

IIR Filter with Pipelining

In the world of digital signal processing, Infinite Impulse Response (IIR) filters are commonly used. An IIR filter is constructed from so-called *Biquads* (**Figure 4**) connected in series. In the non-parallelized form, the input value of a Biquad is precisely the output value of the previous Biquad. It is necessary to compute *biquad1* before the computation of *biquad2* can begin.

The evaluation of a Biquad is dependent on the results of the previous stage:

```
out1=biquad1(x) ;
out2=biquad2(out1) ;
out3=biquad3(out2) ;
out4=biquad4(out3) ;
```

This makes it difficult to parallelize the evaluation of Biquads. There is, however, a pipelining technique that can be applied here. The term pipeline originates from the implementation of fast processors. The execution of an instruction is divided into *Fetch*, *Decode*, and *Execute* phases, each of these phases is executed in separate processing stages in parallel.

When Instruction I_k is fetched, the previous instruction $I_{(k-1)}$ has already been decoded and the instruction before that *I*(*k*-2) is being executed. This form of parallelization results in a higher throughput (**Figure 5**). The execution of a single instruction is not any faster.

This concept can now be applied to the IIR filter. The corresponding code is as follows:

```
inp1=x ;
inp2=out1 ;
inp3=out2 ;
inp4=out3 ;
```

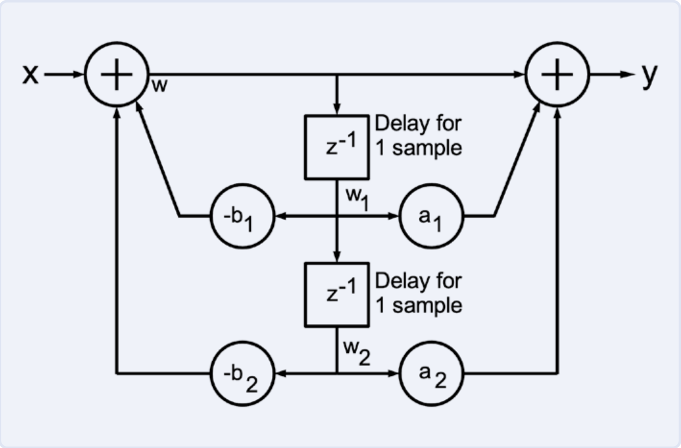


Figure 4: Signal flow of a Biquad.

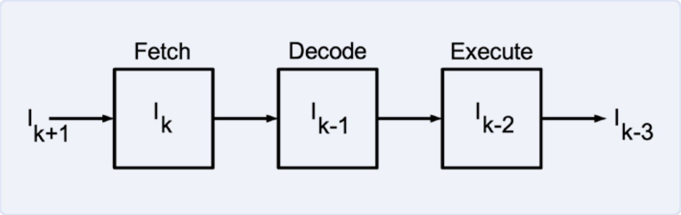


Figure: 5: Pipelining in a CPU.

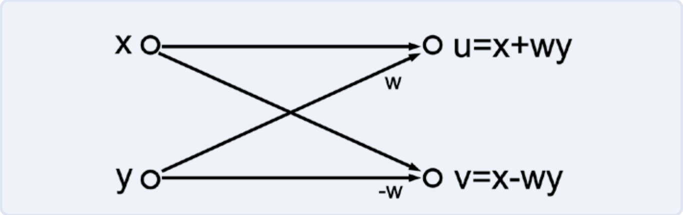


Figure 6: Signal flow in a Butterfly operation.

```
out1=biquad1(inp1) ;
out2=biquad2(inp2) ;
out3=biquad3(inp3) ;
out4=biquad4(inp4) ;
```

Here, there are additional intermediate values `inp1`...`inp4`. This makes the Biquad operations independent of each other. The Biquad evaluations can now be parallelized, and that's precisely how IIR filter routines are performed using Dual-Core CPUs. This method achieves a speedup S = 1.8 approximately.

The XIP Feature

If you measure the execution time of the IIR filter multiple times consecutively, you can make a surprising discovery (**Table 1**): In the first run, the runtime is around 64 µs which takes nearly 2.5 times longer than all subsequent runs. This acceleration can be explained by considering how the RP2040 accesses code memory. The code is stored in the serial QSPI flash memory, and the CPU can access this code as if it were stored in the CPU (hence XIP = eXecute In Place). Since the flash memory is accessed via a serial interface, accessing the data stored here is relatively slow. For acceleration, the XIP interface is equipped with a cache based on fast RAM. The recently read words are stored here, so a re-access does not occur through the QSPI interface but from the cache.

This process explains the values of the runtime measurements. In the first run, the code does not make use of the cache, so QSPI access is slow and serial. Once the code is in the cache, the accesses in the subsequent runs are correspondingly fast. This is how intricate the runtime behavior of modern CPUs can be. For correct and reliable results, you need to be aware of such mechanisms.

Fast-Fourier Transformation (FFT)

In digital signal processing, Fast Fourier Transforms (FFTs) are often used to analyze signals. An FFT calculation consists of so-called butterfly operations. **Figure 6** shows the data flow in a butterfly operation.

The factor *w* is the cosine of a phase that is dependent on the butterfly operation.

The diagram in **Figure 7** shows the data flow where N = 16. There are $\log_2 N = 4$ iterations performed, and each iteration consists of N/2 butterfly operations. Within an iteration, the butterfly operations are independent of each other. They can therefore be executed in parallel, and that's just what this program does. Half of the butterfly operations in one iteration are carried out by Core 0, while the other half are performed by Core 1. Each core performs N/4 Butterfly operations per iteration. This gives a speedup factor S of 1.85.

Table 1: IIR-filter execution times.

Measurement	#	µs
Timer32Measure:	0	64
Timer32Measure:	1	26
Timer32Measure:	2	25
Timer32Measure:	3	25
Timer32Measure:	4	25

Horner's Method

Now we can look at how to evaluate a polynomial at point *x*. In this example, the degree of the polynomial N = 5. According to the definition of a polynomial:

$$p(x)=a_5x^5+a_4x^4+a_3x^3+a_2x^2+a_1x+a_0$$

To evaluate the polynomial in this form, you need N - 1 multiplication operations to calculate the powers of *x*. In addition there are N - 1 multiplications of the powers with the coefficients a_k and N

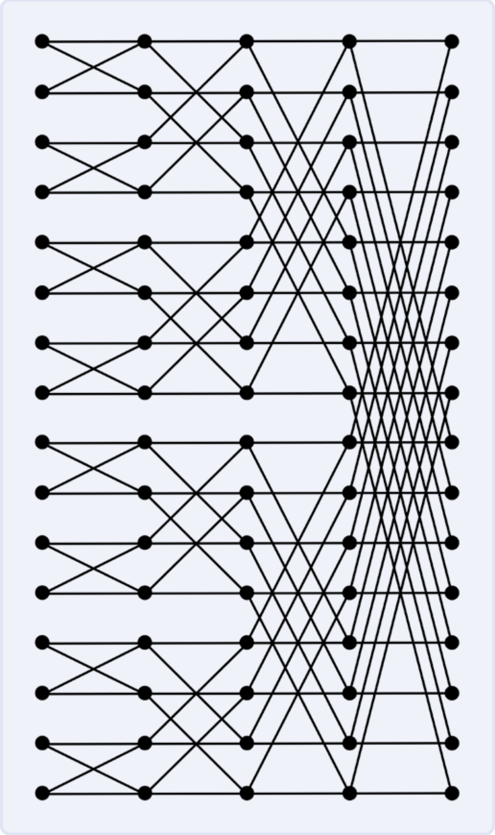


Figure 7: Signal flow of a 16-point FFT.

addition operations. A more efficient calculation can be performed using Horner's Scheme or Method. Here you can calculate the value p_k, k = N,..,0 using the formula:

$$p_5=a_5 \quad p_4=x\,p_5+a_4 \quad p_3=x\,p_4+a_3 \quad p_2=x\,p_3+a_2 \quad p_1=x\,p_2+a_1 \quad p_0=x\,p_1+a_0$$

$$p_0=a_0+x\,p_1=a_0+a_1x+x^2\,p_2=a_0+a_1x+a_2x^2+x^3\,p_3=a_0+a_1x+a_2x^2+a_3x^3+x^4\,p_4$$

$$p_0=a_0+a_1x+a_2x^2+a_3x^3+a_4x^4+x^5\,p_5=a_0+a_1x+a_2x^2+a_3x^3+a_4x^4+a_5x^5=p(x)$$

In the end, you get the value *p(x)* and have used N multiplications and N additions. Altogether, this has saved N-1 multiply operations. The Horner scheme cannot be parallelized as easily, because the evaluations of p_k are interdependent, the polynomial can, however, be split into two halves. The corresponding formulae are:

$$p(x)=x^3(a_5x^2+a_4x+a_3)+(a_2x^2+a_1x+a_0)=x^3q(x)+r(x)$$

$$q(x)=a_5x^2+a_4x+a_3 \quad r(x)=a_2x^2+a_1x+a_0$$

Now, we can evaluate the two polynomials *q(x)* and *r(x)* at the same time. To find *p(x)* just multiply q(x) by $x^3 = x^{(N+1)/2}$ and add it to *r(x)*. Fast calculation of $x^3 = x^{(N+1)/2}$ is really important here. This can be done using the exponents $1, x, x^2, x^4, x^8, x^{16}$ successively, and multiplying the necessary powers for $x^{(N+1)/2}$. For example, for x^9 calculate $x^9 = x^8 x^1$. Calculation of the powers can be performed in $\log_2$ N steps. Since $\log_2$ N is very small compared to N (at least for larger values of N), the necessary calculation of the powers is quick and does not impact too much on the runtime.

The described algorithm was implemented on the Raspberry Pi Pico using N = 100. The non-parallelized version takes 133 µs. The Horner calculation for both polynomial halves running in parallel each takes 71 µs. The fast polynomial calculation requires about 7 µs. The total runtime of the parallelized algorithm is therefore approximately 78 µs. This makes the parallel algorithm speedup S = 133 / 78 = 1.7 times faster. It's evident that the calculation of $x^3 = x^{(N+1)/2}$ necessary for the parallelization eats into the achievable speedup, but the value attained is nevertheless worthwhile.

Linear Algebra

In addition to parallelizing the dot product, the matrix-vector product was also parallelized, achieving a speedup S = 1.6. The solution for a linear equation system was also implemented, resulting in a speedup of 1.8. A simple loop-splitting technique was used for both tasks.

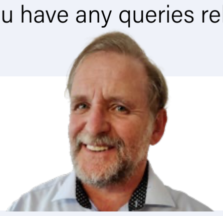
Truly Effective

By exploring the series of examples detailed in this discussion, it becomes clear how effectively tasks can be parallelized for execution on the dual cores of the Raspberry Pi Pico. The attained speedup values consistently approach around 1.8 — close to the theoretical limit S = 2. The pivotal strategy for optimizing the advantages of parallelization involves pinpointing suitable algorithms or methods capable of independent execution to attain the desired outcomes — only these will be truly effective when executed in parallel. You can download the source code and all examples for free on the Elektor website related to this article [2].

230440-01

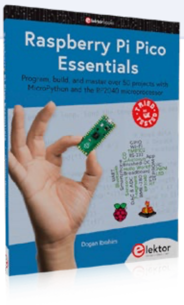
Questions or Comments?

Please contact the author at ossmann@fh-aachen.de or the team at Elektor at editor@elektor.com if you have any queries related to this article.



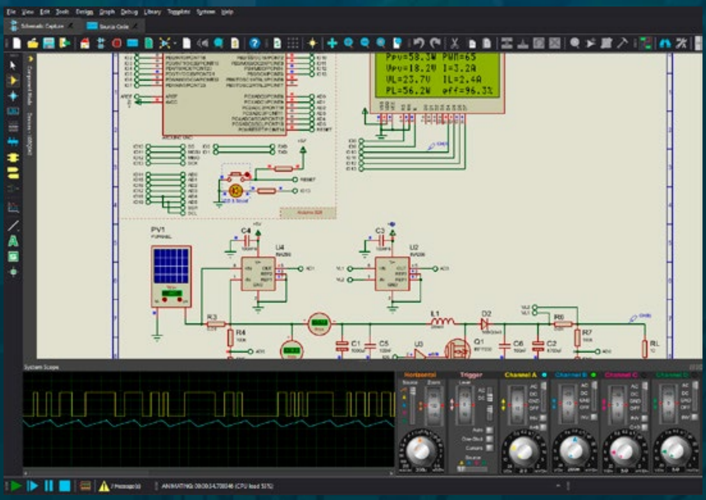
About the Author

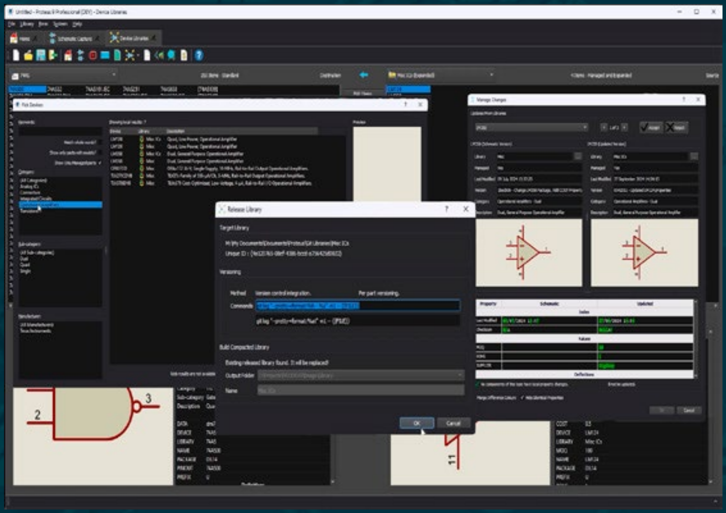
Martin Ossmann began reading Elektor, and tinkering, at the age of 12. After studying electrical engineering and working for several years as a development engineer, he became a professor in the Department of Electrical Engineering and Information Technology at FH Aachen. Not only is he an author of many scientific publications, but for more than three decades, he has been regularly publishing novel circuits and software projects in *Elektor Mag*, showcasing his wealth of technical expertise.


Raspberry Pi Pico Essentials
Program, build, and master over 50 projects with MicroPython and the RP2040 microprocessor
Dogan Ibrahim
elektor

WEB LINKS

[1] M. Ossmann, "Raspberry Pi Pico Makes an MSF-SDR," Elektor 7-8/2022: http://www.elektormagazine.com/220006-01
[2] Software Download: http://www.elektormagazine.com/230440-01





Introducing new electronic products every day. Mouser is the first choice among design engineers seeking the newest products and latest technologies. By focusing on delivering the newest technologies and newest products, Mouser helps engineers obtain a clear edge in product features, longer product lifecycles.

Order - with - **Confidence**

MOUSER
ELECTRONICS