

Video Output with MCUs

From
Composite Video
to DVI

p. 6

p. 70

THE TUBE
An Unusual
Tube Amplifier

p. 16

ESP32 CAMERA
A Simple Approach with
Inexpensive Modules

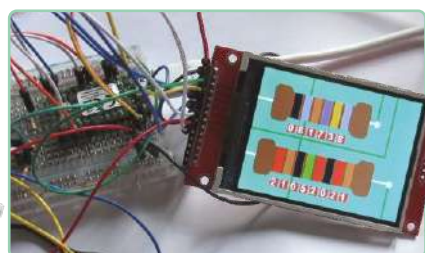
FOCUS ON

**Audio &
Video**



ATX Power Supply for Raspberry Pi:
Different Voltages at Interesting
Power Levels!

p. 20



SDR Radio-Controlled Clocks:
Five Time Signals, Six Displays

p. 29



Using Light for Sound Effects:
LDR-Based Voltage-Controlled
24 dB/oct Synthesizer Filter

p. 98

From Arduino and Elektor with Engineering Love

OUT NOW

A unique Elektor Magazine edition curated by guest editor Arduino!

DIY electronics projects, engineering insights, and more from Arduino and Elektor engineers

Packed with projects and tutorials

Dive into hot topics such as MicroPython, TinyML, and home automation with Arduino

Get to know Arduino: Insights from Fabio, Massimo and David

Get Started with the Portenta x8

Get it now! From your favorite newsstand or buy in the Elektor web stores!

Direct links in articles give you easy access to Arduino products and solutions

More information

www.elektor.com/arduino-magazine



Elektor Magazine,
English edition

Volume 49, No. 519
January & February 2023

ISSN 1757-0875 (UK / US / ROW distribution)

www.elektor.com
www.elektormagazine.com

Elektor Magazine, English edition
is published 8 times a year by
Elektor International Media

Head Office:

Elektor International Media b.v.
PO Box 11
6114 JG Susteren
The Netherlands
Phone: (+31) 46 4389444

Memberships:

E-mail: service@elektor.com
www.elektor.com/memberships

Advertising & Sponsoring:

Raoul Morreau
Phone: +31 (0)6 4403 9907
E-mail: raoul.morreau@elektor.com

www.elektor.com/advertising
Advertising rates and terms available on request.

Copyright Notice

The circuits described in this magazine are for domestic and educational use only. All drawings, photographs, printed circuit board layouts, programmed integrated circuits, disks, CD-ROMs, DVDs, software carriers, and article texts published in our books and magazines (other than third-party advertisements) are copyright Elektor International Media b.v. and may not be reproduced or transmitted in any form or by any means, including photocopying, scanning and recording, in whole or in part without prior written permission from the Publisher. Such written permission must also be obtained before any part of this publication is stored in a retrieval system of any nature. Patent protection may exist in respect of circuits, devices, components etc. described in this magazine. The Publisher does not accept responsibility for failing to identify such patent(s) or other protection. The Publisher disclaims any responsibility for the safe and proper function of reader-assembled projects based upon or from schematics, descriptions or information published in or in relation with Elektor magazine.

© Elektor International Media b.v. 2023
Printed in the Netherlands

Jens Nickel

International Editor-in-Chief, Elektor Magazine



For Eyes and Ears

It is all in the mix — and it has always been at Elektor. You can learn just as much from diving into a circuit diagram as from delving into source code. In this edition, we approach the hot topic of audio and video — both analog and digital, in theory and in practice. After all, it's not just the eyes and ears that want to get their money's worth; the gray matter in the head also wants its due.

Our cover article is especially inviting. An inexpensive controller board, a few external components, and some open-source software are all you need to output video signals that can be viewed on a monitor. This works not only with a video composite signal; VGA and HDMI are also possible. My colleague Mathias Claußen put together an article full of background knowledge, but it became so extensive that it pushed our editorial workflow to the limit. Instead of printing a 20-page article in this issue, we decided to split Mathias's article into two parts. This issue is about how to coax a composite signal out of a small 8-bit controller, in black and white, as a grayscale image or in color (page 6). In the next issue, somewhat stronger computing artists will be in action to generate VGA, DVI, and HDMI signals.

For advanced users, I'd like to recommend Tam Hanna's article, which introduces Espressif's Audio Development Framework, using practical applications such as an MP3 player (page 88). I was amazed at what is possible with an ESP32 and a few lines of software in terms of audio processing. Granted, the learning curve is steep. But for all those who want to develop professional projects with voice or music output, the struggle should pay off in every respect.

Our amplifier for 32-Ω headphones does not require a microcontroller at all. A dual audio op-amp OPA2134 is complemented by two BUF634As, which take care of the current amplification. If you like, you can replace these ICs with a discrete circuit.

Join in and learn!

The Team

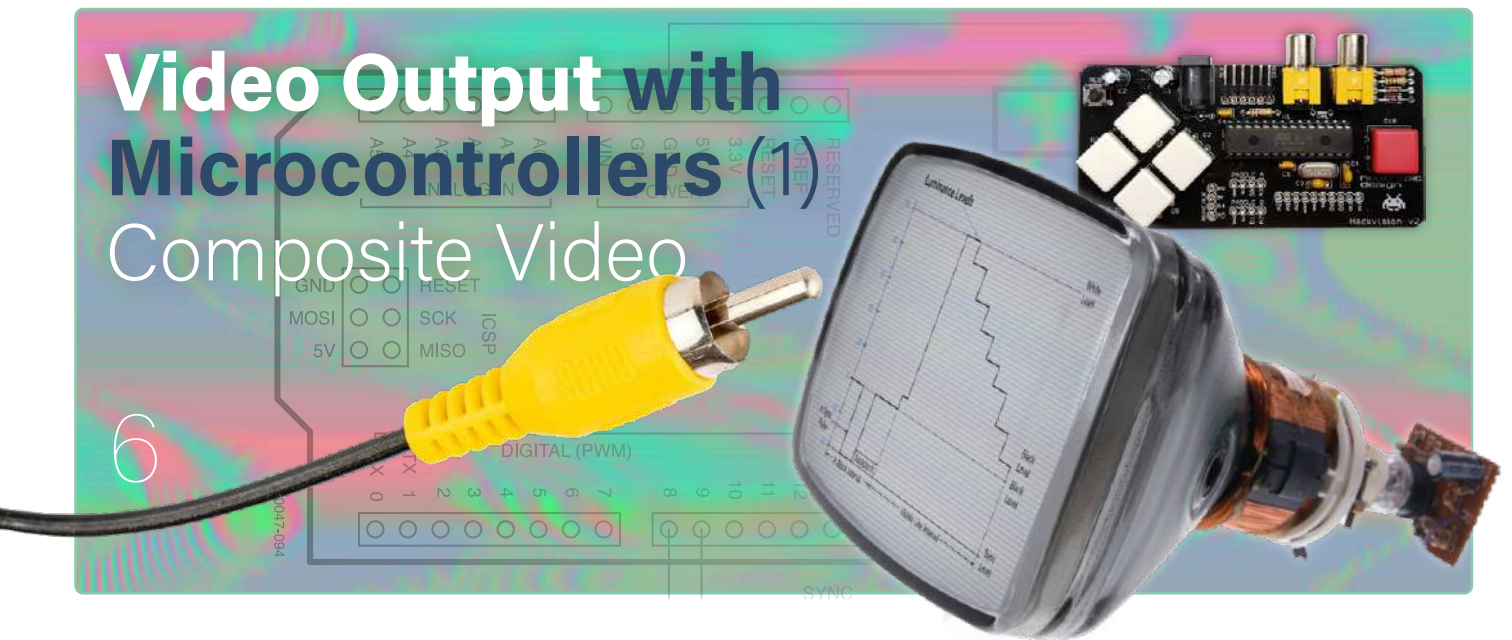


International Editor-in-Chief:	Jens Nickel
Content Director:	C. J. Abate
International Editorial Staff:	Eric Bogers, Jan Buiting, Stuart Cording, Rolf Gerstendorf, Alina Neacsu, Dr Thomas Scherer, Brian Tristram Williams
Laboratory Staff:	Mathias Claussen, Ton Giesberts, Clemens Valens
Graphic Design & Prepress:	Giel Dols, Harmen Heida, Sylvia Sopamena
Publisher:	Erik Jansen

DEUTSCHE

FACHPRESSE

Elektor is a member of VDZ (Association of German Magazine Publishers), which "represents the common interests of 500 German Consumer and B2B publishers."



Regulars

- 3 Colophon**
- 34 Starting Out in Electronics**
Special Diodes
- 38 From Life's Experience**
Musings on the Quality of Things
- 52 Developer's Zone**
THD Measurement with an Oscilloscope and FFT
- 77 Ethics in Action**
Biomaterial in Electronics: Ready or Not
- 103 Retronics**
Elektor High-Power AF Amplifier
- 106 HomeLab Tours**
A Volumetric Display Made in Canada
- 110 Err-electronics**
Corrections, Updates, and Readers' Letters
- 114 Hexadoku**
The Original Elektorized Sudoku

Features

- FOCUS**
- 6 Video Output with Microcontrollers (1)**
Composite Video
- 14 electronica 2022**
News from the World's Leading Electronics Trade Show

- 47 MakePython ESP32 Development Kit**
Everything in a Box
- 80 Opera Cake Antenna Switch for HackRF One**
Connect Up to Eight Antennas to Your SDR
- 82 Interview**
Engineering with Arduino and More
- FOCUS**
- 88 Audio Signals and the ESP32**
The ESP-ADF Environment in Practice

Industry

- FOCUS**
- 56 All-Seeing Machines**
The Technology Behind Today's Industrial Vision Systems
- FOCUS**
- 60 Infographics**
Facts and Figures
- FOCUS**
- 62 The Evolution of Voice and Audio Control for Electronic Devices**
- 66 WEEF 2022 in Review**
- 68 FFWD electronica 2022 in Review**
Innovators Did Not Fail to Impress

The Tube
An Unusual Tube
Amplifier



70



SDR Radio-
Controlled Clocks

29



Using Light for
Sound Effects

98

Projects

FOCUS

16 ESP32 Camera

So Simple, It Doesn't Even Have to Use Wi-Fi

20 ATX Power Supply for Raspberry Pi

FOCUS

26 32 Ω Headphone Amplifier

Simple But High-Quality 3-Chip Solution

29 SDR Radio-Controlled Clocks

Five Time Signals, Six Displays

40 Reverse-Engineering a Bluetooth Low Energy LED Badge

How to Control a BLE Device with a Python Script

FOCUS

70 The Tube

An Unusual Tube Amplifier

85 LiDAR Precision Gauge

Measure Up to 12 Meters

FOCUS

98 Using Light for Sound Effects

LDR-Based Voltage-Controlled 24 dB/oct Synthesizer Filter

Next Edition

Elektor Magazine Edition 3-4/2023 (March & April 2023)

As usual, we'll have an exciting mix of projects, circuits, fundamentals, and tips and tricks for electronics engineers and makers. We will focus on Embedded and AI.

From the Contents:

- > Alarm Clock 2.0
- > Video with Microcontrollers: VGA, HDMI, DVI
- > Raspberry Pi Pico: PIO in Practice
- > Programming Voice-controlled IoT Applications
- > The Android WiFi API
- > My First Software-Defined Radio
- > Poor Man's ChipTweaker
- > USB True Random Number Generator

And Much More!

Elektor Magazine edition 3-4/2023 (March & April 2023) will be published around March 16, 2023. The arrival of printed copies for Elektor Gold Members is subject to transport. Contents and article titles are subject to change.

47

**MakePython ESP32
Development Kit**
Everything in a Box



Video Output with Microcontrollers (1)

Composite Video

By Mathias Claussen (Elektor Lab)

The topic of video output with microcontrollers goes back to the beginnings of these small all-round chips. Today's microcontrollers have considerably more computing power than, for example, the almost 42-year-old Sinclair ZX81 home computer, but even current microcontrollers are a long way from the memory size of modern graphics cards measured in gigabytes. Nevertheless, developers still manage to output amazing moving images with an ATmega, an ESP32 or an RP2040. In the first part of this series, we will cover the output of composite video. In the next issue, we will continue with VGA and even DVI. In any case, a few tricks and exact timing are necessary. Therefore, this is not only about theory, but also about practical examples as a starting point for your own experiments.



The history of video formats goes far back to the beginnings of television. As with radio transmissions, standardization and norms were also sought for television. During the analog color television era, the most common standards were National Television Systems Committee (NTSC — North and South America, Japan), Phase Alternating Line (PAL — Europe, South America, Africa and Asia), and Séquentiel Couleur à Mémoire (SECAM — France, Africa and USSR).

The basic procedure for analog transmission of video signals was carried out according to Video Blanking and Sync (VBS) for these standards. The resolution and frame rate for VBS depends on the underlying television standard. For NTSC, it is 480 visible lines with 640 visible pixels at 59.94 fields per second. PAL and SECAM have 576 visible lines at 720 visible pixels and 50 fields per second (576i). In addition, NTSC, PAL and SECAM also differ in modulation and the way color information is added. In the age of digital image transmission, these three transmission methods have largely lost their significance. However, they have remained with us in the form of digital image formats for DVD video or Standard Definition Television (SDTV).

Composite Video with CVBS

The first standardized television signals were designed for the transmission of monochrome images. Different picture formats were developed in the USA and Europe. The timing of a VBS signal in the PAL format is shown in **Figure 1** as an example of these standards.

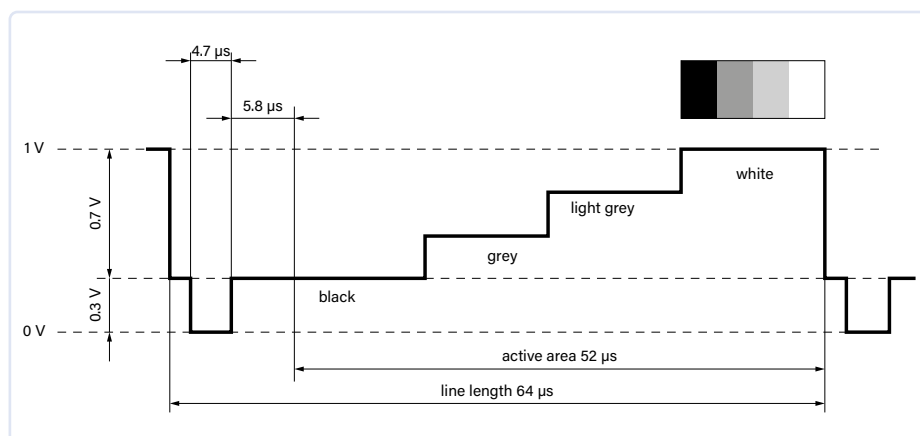


Figure 1: VBS signal with PAL timing. (Source: Wikipedia)



Figure 2: Opened TV set with cathode ray tube. (Source: Shutterstock / Sergio Sergio)

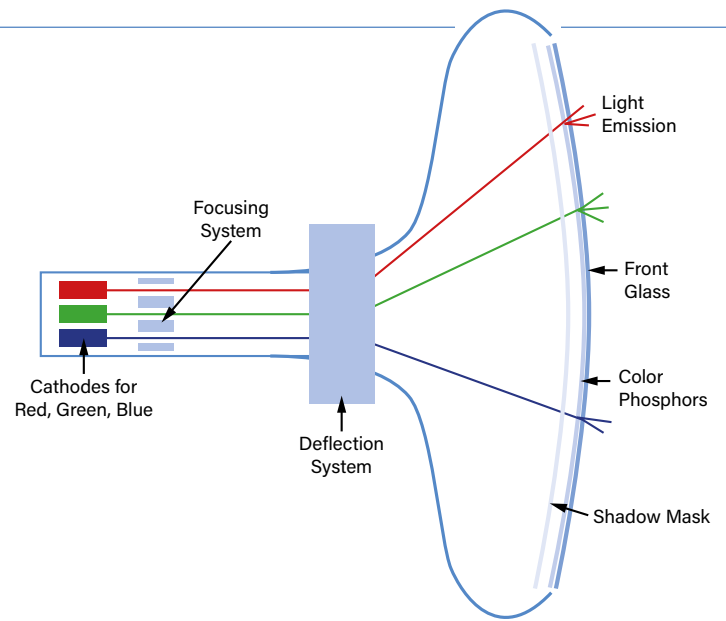


Figure 3: Schematic structure of a color picture tube. (Source: ITWissen.info)

This standard was especially suitable for image reproduction with cathode ray tubes (**Figure 2**). The operating principle of such picture tubes differs massively from LCDs or OLED screens. In a picture tube, a modulated electron beam deflected horizontally and vertically is directed onto a phosphor layer. At the point of impact, the phosphor then glows in proportion to the intensity of the electron beam (**Figure 3**). Time-synchronized horizontal and vertical deflection produces a two-dimensional image.

From the viewer's perspective, an image

(black and white) is built up in lines from left to right and the lines then from top to bottom (**Figure 4**). The first "pixel" per image is therefore located at the top left. This logic has been largely retained in modern digital displays.

The historical knowledge about image generation by means of a traveling electron beam makes it easier to understand modern video signals. The signal in **Figure 1** contains the structure of a picture line. In the front area (**Figure 5**) of the signal, the "horizontal blanking" can be seen. At the beginning of "horizontal blanking," the electron beam is at the right

edge of the picture tube and must therefore be returned to the far left for the next line. During this jump from right to left, the electron beam is blanked by setting the brightness signal to black or even "blacker than black."

Horizontal blanking takes $12\text{ }\mu\text{s}$ for PAL and $10.9\text{ }\mu\text{s}$ for NTSC. It consists of three sections called the "front porch," the "sync tip" and the "back porch." The brightness level of the front porch (PAL = $1.65\text{ }\mu\text{s}$ / NTSC = $1.4\text{ }\mu\text{s}$) is at or slightly below the value of black at 0.3 V , while the electron beam travels even further to the right out of the visible image. The voltage at

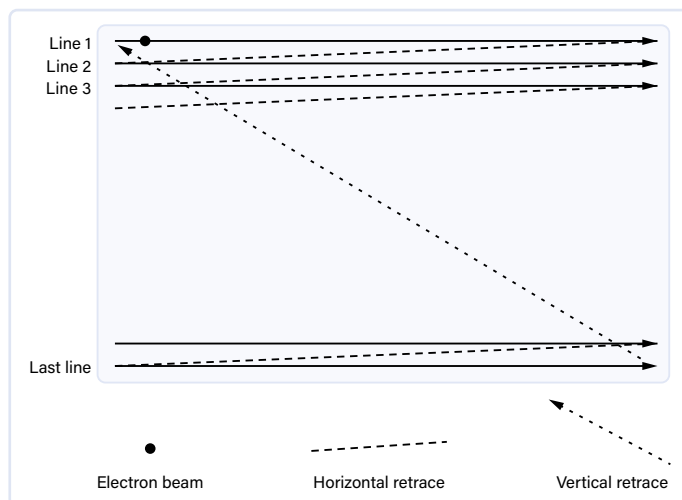


Figure 4: Line-based image composition.

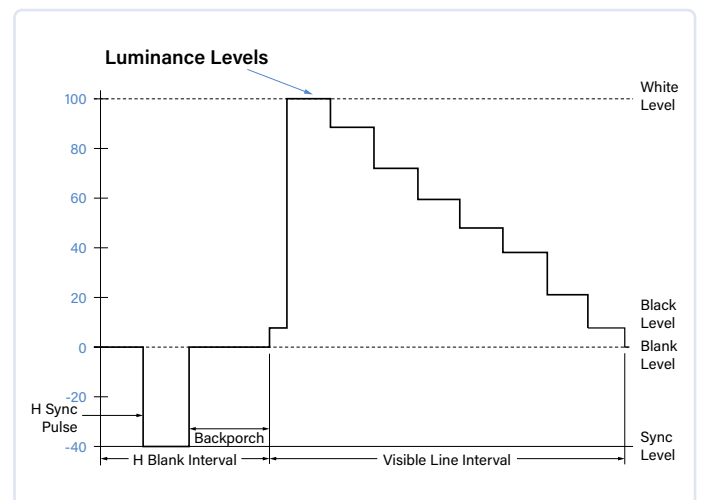


Figure 5: Horizontal blanking. (Source: www.edn.com)

the sync tip (PAL and NTSC = 4.7 μ s) is 0 V and is thus significantly lower than the black level. The sync tip causes the electron beam to jump from right to left. The back porch (PAL and NTSC = 4.7 μ s) provides the reference for the black value of 0.3 V for the line. The content to be drawn, i.e., the image information, starts at the end of the back porch. 52 μ s is then available for the visible part of the line. The number of pixels or sample points packed into this 52 μ s depends on the video format. In PAL, there are usually 720 visible pixels. When all lines are drawn and the electron beam has reached the bottom right, a jump to the starting point of the image (top left) must be performed. This process will be described in a separate section.

Synchronization and Interlacing

The analog bandwidth for transmitting television images was sufficient for only 25 frames per second for PAL and 29.97 frames per second for NTSC. For the human eye, this results in a moving picture, but it is not very smooth and creates a very unpleasant flickering effect. With a faster frame rate of 50 or even 59.94 frames per second, the flickering is reduced noticeably. Most people then perceive this as a smooth visual impression. Since the available radio bandwidth was limited, the interlaced scan technology was introduced: Instead of transmitting all image lines at a higher frequency as full frames, the lines with even or odd numbers are alternately transmitted at twice the frequency, i.e., 50 or 59.94 Hz, respectively (Figure 6).

So, first a half frame ("field") with all odd line numbers is transmitted, followed by the second field with the even line numbers. In



Figure 6: Frame composition from two fields.

order for the lines of the two fields to be drawn into the gaps of the respective other field, there is a detection signal at the beginning of each field.

Since the lines of a complete frame are numbered according to the transmission sequence, the first field in PAL has the lines 1 to 313 and the second field then continues counting up to line 625. The vertical synchronization for the end of one field and the beginning of the next is a bit tricky: Lines 311 to 317 at the transition from odd to even fields as well as lines 623 to 5 at the transition from the even to the odd field of the next complete frame are

not visible. In these lines, faster synchronization pulses with levels between 0.3 V and 0 V are transmitted. Figure 7 shows the sequence for a complete PAL frame. By means of these synchronization pulses, the electronics can recognize whether it is the field with even or odd line numbers.

Monochrome Video with AVR Controllers

Basic information about composite signals like CVBS or FBAS can be found on Wikipedia [1]. If you are looking for more details and circuit tricks, you will surely find them in the book *Analogue Video* [2] by Angelo La Spina.

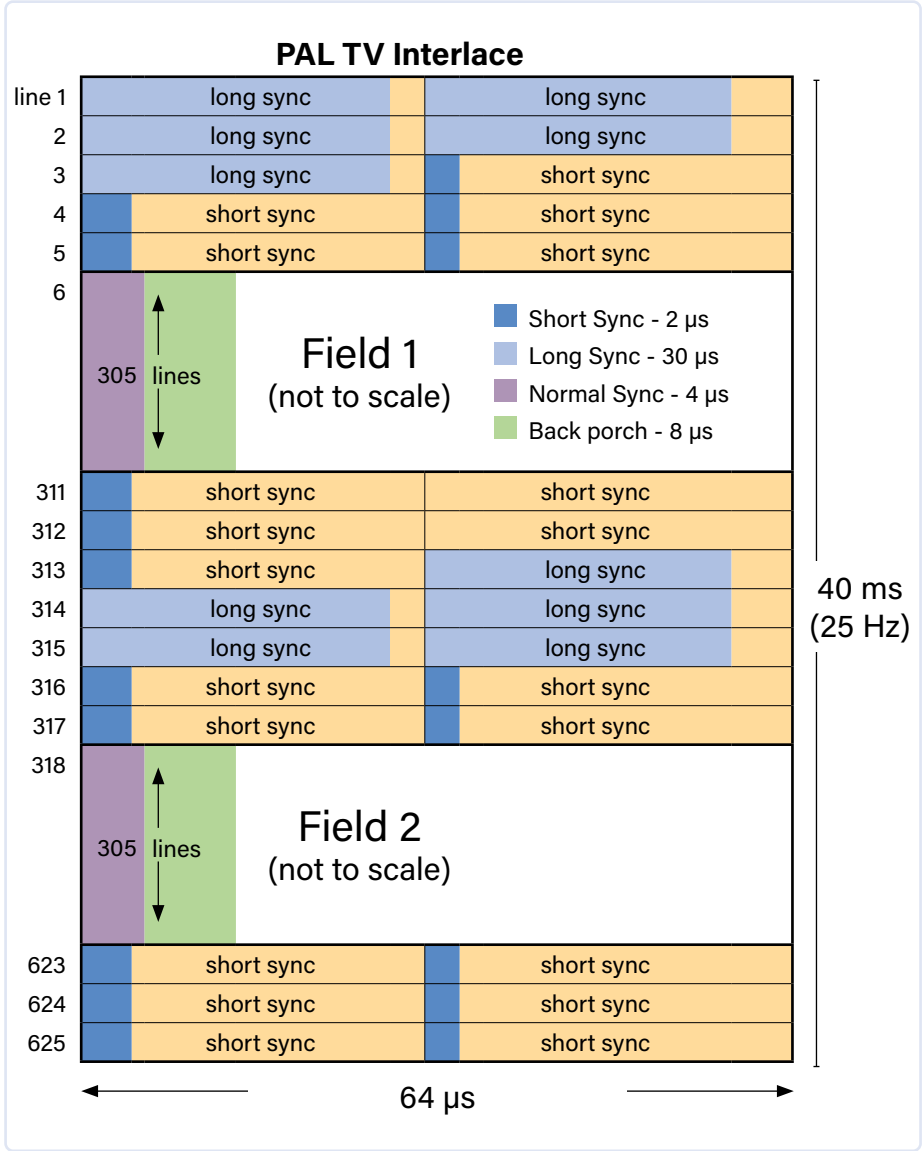


Figure 7: Composition of a complete PAL image. (Source: martin.hinner.info)



If the video signal is transmitted as a VBS signal (no color information, only brightness) using a suitable composite video cable (**Figure 8**), the generation of the brightness information can be quite simple. If you use only white and black instead of grayscale, you only need to output two suitable voltages plus synchronization. Besides simplifying the circuit, this also reduces the memory required for image generation.

The ATmega328P microcontroller of an Arduino UNO is thus able to generate a monochrome image with 128×96 pixels. All pixels can be stored in the internal memory of the Arduino UNO because only 1536 bytes is needed. The wiring of an Arduino UNO with composite output is shown in **Figure 9**.

As you can see, two resistors are sufficient to output a suitable signal. For the image output, you don't have to reinvent the wheel with your own code, but you can use the *TVOut* library [3]. This library supports timing for NTSC and PAL. The video output that can be done with it ranges from simple text to your own games. Hackvision [4] is a platform for a game console. This is open hardware for which, in extreme cases, a breadboard and a few components are all that is needed. An Arduino can also be converted into a Hackvision platform (**Figure 10**) and thus use the associated library of games.

Video output with an AVR microcontroller requires one of its timers. The horizontal and vertical synchronization is provided by Timer1 and its output pin PB1 (OC1A). Based on the timer, the pixels are output line by line via a dedicated pin (PD7).

The values for the resistors at PB1 and PD7 can be determined quite easily. A composite input on monitors or TV sets has an input impedance of 75 Ω. The level for synchronization is between 0 V and 0.3 V. The voltage for the brightness values of the pixels is between 0.3 V for black and 1 V for white, respectively. In the following, it is assumed that the microcontroller is supplied with 5 V.

Figure 11 shows the voltage divider for the video output at the typical impedance of 75 Ω. Pin 9 supplies the synchronization signals.



Figure 8: Cable for composite video with RCA plug. (Source: Shutterstock / Woodpond)

Mathematically, a value of 1175 Ω would be required for R1, so that the voltage drop at R2 is 0.3 V with a high level at pin 9. 1 kΩ provides a maximum of 0.34 V at R2, which is accurate enough for this purpose. Pin 7 outputs the brightness values. Values between 0.34 V (black) and 1 V (white) are now required at R2. In order to achieve 1 V at R2, a value of 375 Ω is required for the parallel connection of R1 and R3, which would result in 600 Ω for R3. The next larger value of 470 Ω from the E-12 series ensures that the voltage at R2 can never exceed the value of 1 V.

The fact that this works with an Arduino UNO or an ATmega328P proves that low computing power and little memory is sufficient to output graphics on a screen. Since the complete image can be kept in the internal memory, drawing a new image is not time-critical, only the generation of the video signals using Timer1.

Is it also possible to output grayscale = multiple brightness values with the Arduino? This is possible, but the internal memory is the limiting factor. With 16 shades of gray (4-bit grayscale) and suitable resistors, 6144 bytes are needed for video memory if you want to keep the resolution of 128×96 pixels and a complete image has to fit into the memory.

Raspberry Pi Pico and Composite

A Raspberry Pi Pico can also output a composite signal, and it easily supports more than 50 shades of gray. For this, an R-2R resistor ladder [5] is used as digital/analog converter. **Figure 12** shows a suitable circuit consisting of resistors with 180, 320 and 360 Ω. As can be seen from the GitHub page of the project "pico-composite8" [6], the internal resistance of the GPIO pins cannot be

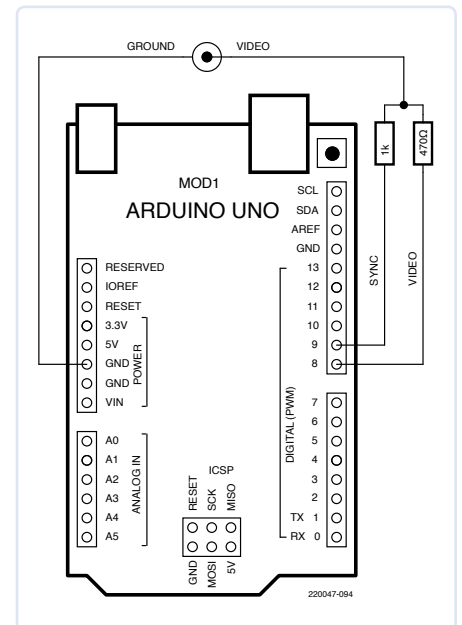


Figure 9: B/W video output with two resistors on an Arduino UNO.

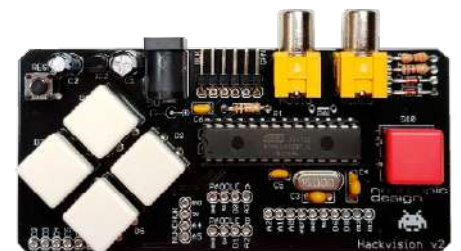


Figure 10: Hackvision hardware. (Source: nootropic design)

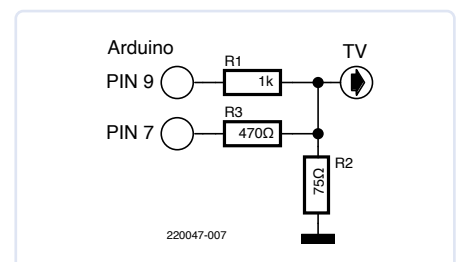


Figure 11: The circuit of the video output of Figure 9.

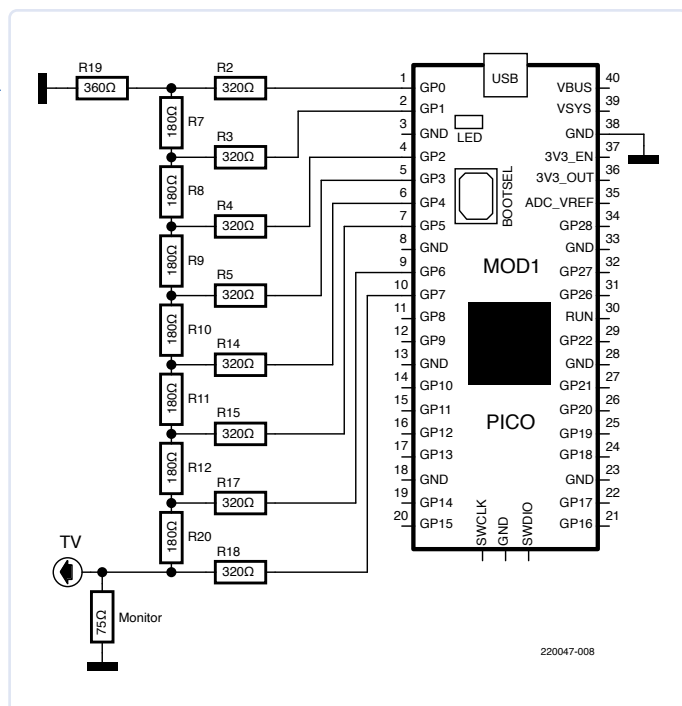


Figure 12: An R-2R network as DAC on the Raspberry Pi Pico.

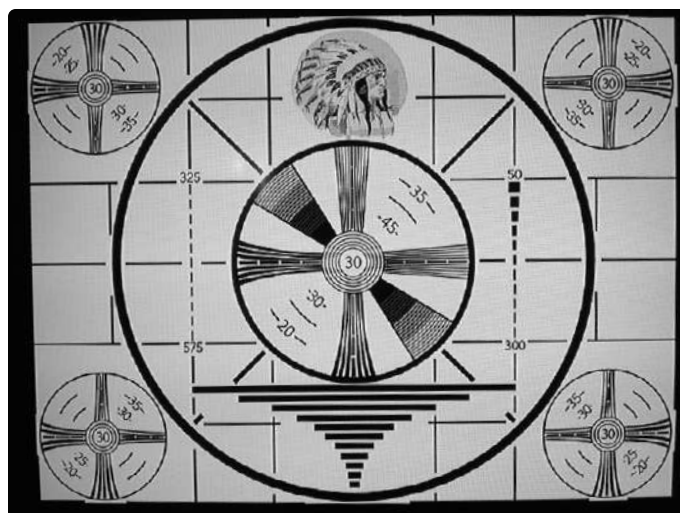


Figure 13: Grayscale image from a Raspberry Pi with 512x384 pixels. (Source: tinyurl.com/2p8z27a2)

neglected here. The developer of the project has determined values of about 40 Ω for this. The Pico outputs a composite signal according to NTSC here. It loads the images either from RAM or Flash and can output a maximum of 512x384 pixels. The result can be admired in **Figure 13**.

However, the project is only a feasibility demonstration and does not provide a ready-made generic library. The fact that a Raspberry Pi Pico can output grayscale video with 512x384 pixels shows that it is not so much a matter of computing power but of the right timing. If a complete image is kept in the RAM of the Raspberry Pi Pico, only about 64 KB of the 264 KB remain available for your own applications. But if you consider that even an ATmega can handle games like Tetris or Pong including video output, this should offer more than enough room for your own creations.

Color for Composite Video

More than 50 shades of gray is good, but color is simply better. When it comes to composite video and color, it becomes much more difficult than before to generate a suitable analog signal. In 2003, Rickard Gunée demonstrated the generation of a composite signal (PAL or NTSC) at a Hackaday event [7] using a Scenix / Ubicom SX28 at about 50 MHz.

But how difficult is it to add color to a video signal? The answer depends on the way color information is added to the composite signal.

Colorful Thanks to PAL and NTSC

When switching from black-and-white to color television a few decades ago, no entirely new signal was defined, since it had to be ensured that existing black-and-white televisions could still display the picture. This problem was solved internationally in three different ways, which is why the NTSC, PAL and SECAM standards came to coexist. What they have in common is that the color information was added to the existing monochrome signal.

While PAL and NTSC are similar in principle (quadrature modulation for color), SECAM differs in embedding color by using frequency modulation.

Colors that are provided in their three basic components — red, green, and blue — must be converted to a YUV or YCbCr color space [9] before they can be output via a composite video signal. The conversion from RGB to YUV is calculated as follows:

$$\begin{aligned} Y &= 0.299 * R + 0.578 * G + 0.144 * B \\ U &= 0.493 * (B - Y) \\ V &= 0.877 * (R - Y) \end{aligned}$$

This transforms the values for R, G, and B into a range of between 0 and 1. The formulas show that an implementation for microcontrollers requires either computing power or some programming skills.

The following is a basic explanation of the embedding of color based on PAL and NTSC.

Figure 1 shows a monochrome PAL signal to which no color information has been added yet. For color information, PAL uses a subcarrier at 4.43361875 MHz, which is used as a reference signal. This reference is sent in each line as a “color burst” (**Figure 14**) during horizontal synchronization. The test pattern (**Figure 15**) is used to explain how the different colors of the test image are coded.

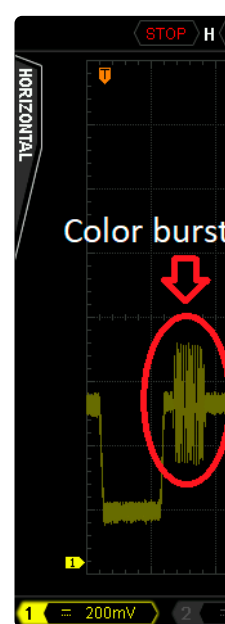


Figure 14: Oscillogram of the color burst of a PAL signal.

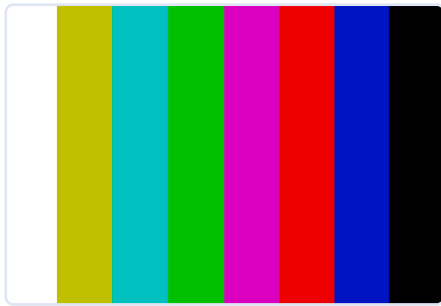


Figure 15: Test pattern with color bars.

The test image corresponds to the EBU color bars [10]; a Raspberry Pi Zero serves as the image signal generator.

The signal for one image line can be seen in **Figure 16**. The brightness (white line as average value of the signal) and the amplitude of the color information are easily visible. There is a third piece of information in the signal, which is hidden in the signal phase. **Figure 17** shows (in red) the phase change at the transition to another color.

Thus, the information for the brightness and the amplitude of the color signal and the phase shift is contained in the signal. If there is an error in the phase or its evaluation during transmission or reception of the signal, the color impression of the image changes, which NTSC receivers allowed to be compensated manually by means of "tint control" [11] via a rotary knob (later also with electronic solutions). With PAL, this problem was avoided because the phase information is shifted by 180° with every second line. A phase error is thus balanced out between two adjacent lines.

Color Composite Video with ESP32

An ESP32 can easily output a monochrome composite signal with a few small tricks, and even an Arduino is capable of doing so. But the generation of a color composite signal imposes significantly higher requirements on the modulation.

In 2018, bitluni demonstrated his approach to generating a color composite signal using an ESP32 [12]. It was shown that the 13.33 MSa/s of the ESP32 DACs is sufficient to integrate a color carrier and color information into the signal. As with the monochrome signal, one processor core of the ESP32 is used to generate the video signal. The setup uses the I2S block of the ESP32 to send data to the DAC (**Figure 18**).

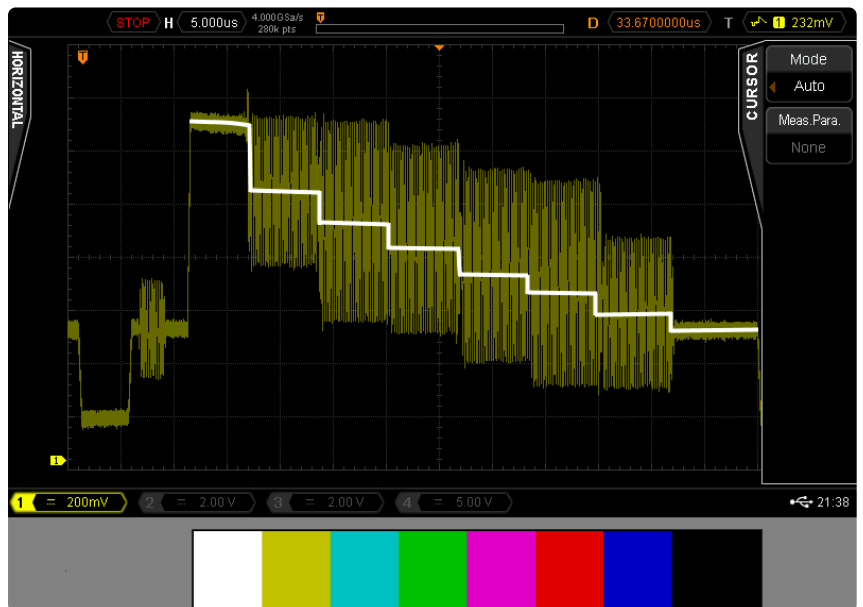


Figure 16: Oscillogram of a PAL color signal.



Figure 17: Phase changes during color change are marked.

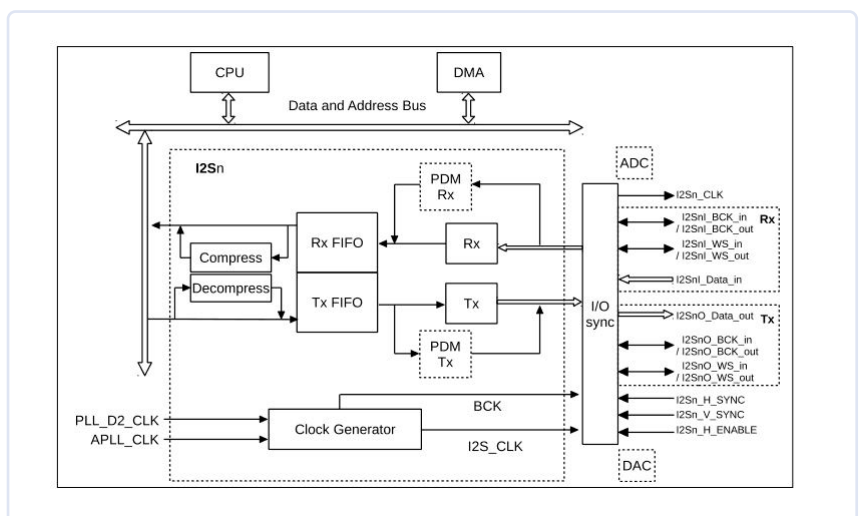


Figure 18: The I2S block of the ESP32. (Source: Espressif / tinyurl.com/yrrbnjak)

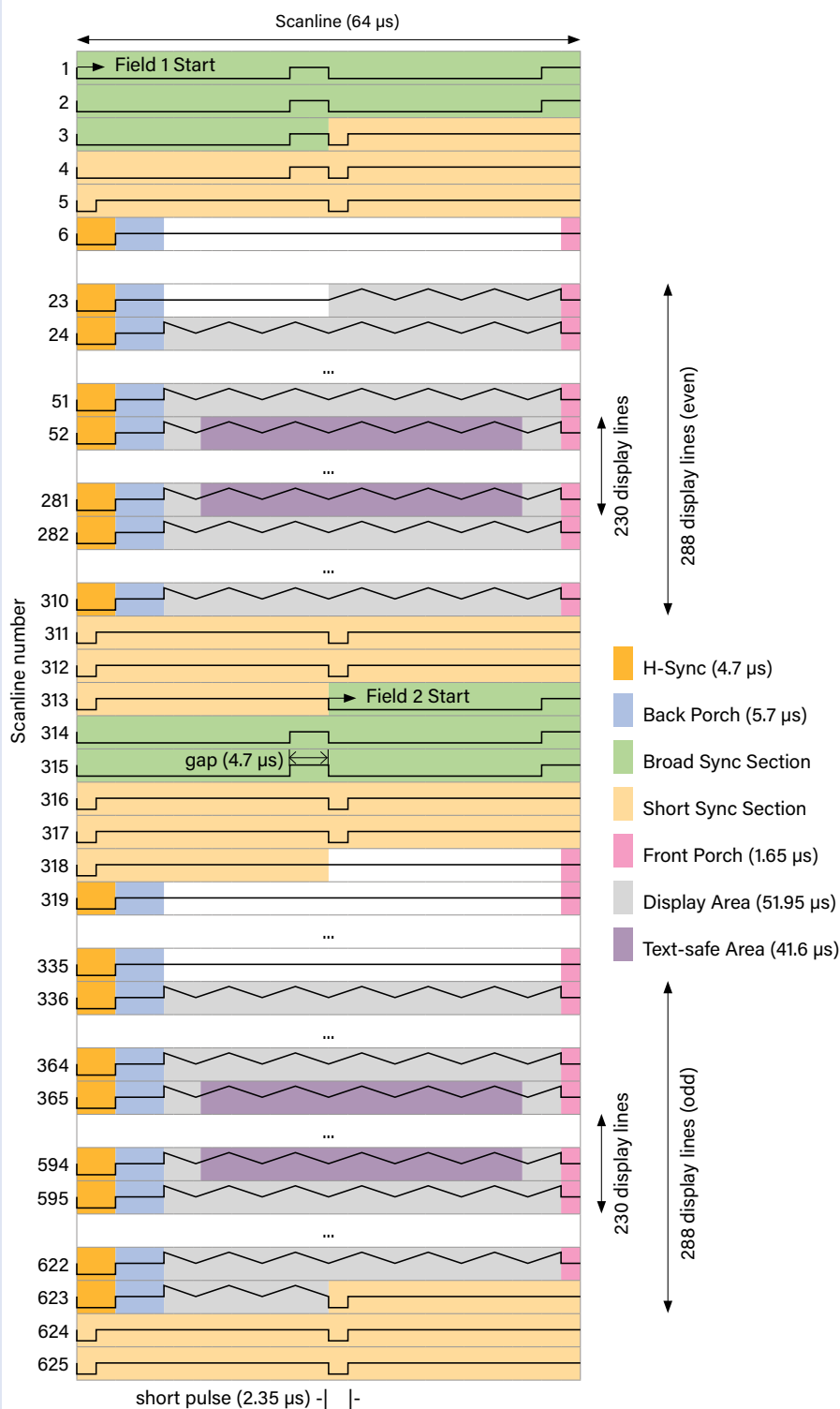


Figure 19: Signal of image output with half image line.
(Source: batsocks.co.uk / <https://tinyurl.com/4fychhmk>)

In PAL and NTSC, the picture is interlaced, i.e., output successively in fields with odd lines and even lines. As already mentioned, this doubles the perceived refresh rate (at the same bandwidth), which significantly reduces the perceived flicker. The timing of these two fields has a few peculiarities, such as only half

a frame line at the end of the image in one of the two fields (Figure 19).

Since the RAM of an ESP32 is only sufficient to output an image with half the resolution, lines must be drawn twice: The content of line one therefore also appears in line two.

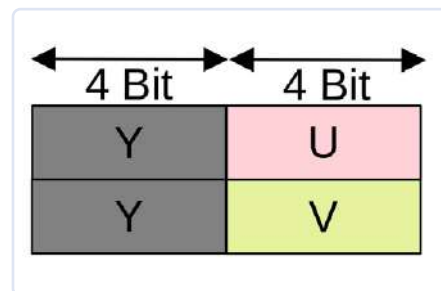


Figure 20: Storage of color information by bitluni.

The interlacing thus ensures that the same content is drawn twice. Now the question arises whether one should really output both the odd field and the even field, or whether it would also be sufficient to use the even field twice and thus simplify the code for the output significantly. Exactly this trick was already used by the Super Nintendo Entertainment System (SNES). And bitluni's code also uses this method to simplify the code for synchronization. So only 288 real lines at 50 Hz are output (288p) — for NTSC, the equivalent mode would be 240 lines at 60 Hz (240p).

In contrast to the monochrome image output, color information must now also be held in the RAM of the ESP32. Generally, the colors are represented by their components — red, green and blue. For the generation of a standard-compliant composite signal, however, the color information should rather be stored in RAM as YUV values. This is the only way that the ESP32 and its DAC can output this data fast enough. Since the amount of RAM in the ESP32 is limited, some tricks are required when encoding the data. bitluni stores the information in RAM as combined YU, YV, and V values with 4-bit resolution each (Figure 20).

Unfortunately, with bitluni's approach, composite video output is only possible according to the PAL standard — with NTSC, it does not work. The ratio between the 3.579545 MHz of the NTSC color burst and the sample rate of the DAC is so unfavorable that no usable color burst can be output and receivers cannot synchronize.



Figure 21: Emulators for 8-bit game consoles. (Source: tinyurl.com/ykd9ezap)

Based on the “esp_8_bit” project, “espflx” was developed. It allows the ESP32 to play videos that are stored on a server of the espflx project. However, this requires some concessions to the video material. Details about this can be found on the associated Github page [14].

Looking Ahead

The second part of the article will deal with VGA, DVI, and sprites. Especially with an ESP32 or the RP2040 of the Raspberry Pi Pico, amazing effects are possible here. If you don't want to wait until the second part, you can register for the webinar “Microcontroller as Pixel Artist” [15], where the topics VGA, DVI, and sprites will be discussed as well. ◀

220047-01



Webinar “Microcontroller as Pixel Artist”

Since moving pictures say more than words, especially on this topic, and extensive listings did not fit into the article, you should check out the the Elektor webinar for this issue. You can already register at [15]. It will be about graphics with Arduino, ESP32, and Raspberry Pi Pico.

Questions or Comments?

Do you have technical questions or comments about this article? Send an e-mail to the author at mathias.claussen@elektor.com or contact Elektor at editor@elektor.com.



Related Products

- **Raspberry Pi Pico RP2040 (SKU 19562)**
www.elektor.com/19562
- **ESP32-DevKitC-32D (SKU 18701)**
www.elektor.com/18701
- **Arduino Uno Rev3 (SKU 15877)**
www.elektor.com/15877

Color with ESP32 in PAL and NTSC

With his project “esp_8_bit,” the GitHub user rossumur has proven that it is also possible to create color according to NTSC with an ESP32. This project is a collection of ESP32-based emulations of various 8-bit consoles, from the Atari 400 to the Nintendo Entertainment System and the Sega Master System (Figure 21).

The project simply uses one pin of the ESP32 to output a composite signal in NTSC or PAL. The trick is hidden in the audio PLL of the ESP32. It can be used to operate the DAC with sample rates up to about 20 MHz. Four times the frequency of the NTSC color carrier is 14.318182 MHz (for PAL 17.734475 MHz). With the APLL, 14.318180 MHz and 17.734476 MHz

can be generated, which is close enough to the necessary frequencies for PAL and NTSC. This way, the DAC now outputs an integer multiple of the color subcarrier frequency, resulting in processable video signals.

With the APLL, the DAC of the ESP32 can not only output a video signal, but this approach is also very interesting for other Direct Digital Synthesis (DDS) applications. However, using the APLL has a disadvantage at high sample rates. The DAC of the ESP32 has two channels. If the APLL supplies one of them with data for video output and an attempt is made to supply the second channel with audio data from the I2S interface, the DAC will suffer from dropouts. These dropouts show up on both channels, so you have to resort to other methods for audio output.

WEB LINKS

- [1] Composite Video: https://en.wikipedia.org/wiki/Composite_video
- [2] Angelo La Spina, “Analogue Video”: <https://www.elektor.com/analogue-video-e-book>
- [3] Arduino TVOut Library: <https://github.com/Avamander/arduino-tvout>
- [4] Hackvision: <https://nootropicdesign.com/hackvision/>
- [5] Resistor ladder: https://en.wikipedia.org/wiki/Resistor_ladder
- [6] pico-composite8: <https://github.com/obstruse/pico-composite8>
- [7] Hackaday event:
<https://hackaday.com/2022/08/17/chips-remembered-the-scenix-ubicom-parallax-sx>
- [8] Color with SX Chips: <https://elinux.org/images/e/eb/Howtocolour.pdf>
- [9] YCbCr color space: <https://en.wikipedia.org/wiki/YCbCr>
- [10] EBU color bars: <https://en.wikipedia.org/wiki/YUV>
- [11] NTSC tint control: https://en.wikipedia.org/wiki/Tint_control
- [12] bitluni, “ESP32 Composite Video”: <https://bitluni.net/esp32-composite-video>
- [13] Github user rossumur: <https://github.com/rossumur>
- [14] espflx: <https://github.com/rossumur/espflx>
- [15] Webinar “Microcontroller as Pixel Artist”:
<https://www.elektormagazine.com/webinars>



electronica 2022

News from the world's leading electronics trade show

By Stuart Cording and Jens Nickel

That's what you call a complete success: Around 70,000 visitors flocked to electronica last November. Sometimes it could get quite crowded — at least at the big distributors' and semiconductor manufacturers' booths. And, at the world's largest trade show of its kind, they didn't go for any half-measures — instead, they came up with extravagant booths and attractive exhibits. There were motorcycles with built-in AI to marvel at, special vans for drone detection, lots of racing cars, and also a single-person quadcopter simulator. Here's a small selection of the exciting new technology we saw on display.



Aaronia, specialist for (RF) measurement technology, displayed this van for drone detection. The fully equipped vehicle is available for around 1.5 million euro, but far less expensive solutions for detecting unmanned aircraft are also available as rack-mounted equipment.

<https://drone-detection-system.com>



Digital microscopes are increasingly replacing those with eyepieces. They are user-friendly, even for inspection personnel unfamiliar with microscopy, and allow samples to be scanned automatically. The Olympus model shown here even produces 3D images that can be viewed from all angles.

<https://olympus-ims.com/en/microscope/dsx1000/high-end-model>



While virtual reality struggles to go mainstream, augmented reality (AR) offers tangible benefits for industry. Technicians can get access to instructions or guidance beamed in front of their eyes as they repair a machine, while still having full view of their environment. TDK's color laser module, embedded into some prototype AR glasses, beams color images directly onto the retina — and we survived to tell of our experience!

www.tdk.com/de/news_center/press/20221013_01.html





This was certainly the highlight at the booth of semiconductor manufacturer Infineon: Those who brought a little patience had the chance to dare trying a virtual flight with this single-person quadcopter simulator.
<https://infineon.com/cms/en/product/promopages/electronica/>



E-paper displays are on the rise because they offer a crisp image at any viewing angle. Distributor Beck Elektronik showed these specimens, sized larger than A4, in monochrome and color. Compared to LCDs, however, the displays are (still) expensive. Moreover, they do not have their own light source, just like printed paper.



<https://beck-elektronik.de/en/products/displays/e-paper-display-epd>



The automotive industry will increasingly rely on software to define their vehicles' capabilities, using standardized, programmable hardware. NXP launched their S32K39, which contains two dedicated motor controller peripherals. Supporting 100 kHz control loops, they are ready to combine with wide-bandgap devices, such as silicon carbide, to deliver high-efficiency traction inverters for electric vehicles.

<https://elektor.link/NXPS32K>



OKdo is a subsidiary of RS Components that offers boards and kits for makers and professionals. ROCK brand single-board computers are well-equipped, powerful, and (mostly) available. The new Rock 5 Model B has an 8K video output, which was demonstrated with a surveillance application.

<https://okdo.com>
<https://wiki.radxa.com/Rock5>



Range anxiety is a core concern for consumers as they switch from fossil-fuel-powered cars to electric. Hoping to allay those fears, Mercedes-Benz recently announced that their VISION EQXX completed a 1,202 km (747 mile) journey on a single charge. Sitting on the onsemi stand, the vehicle utilizes their VE-Trac SiC modules in the traction converter to help deliver 95% of the battery's energy to the wheels.

<https://elektor.link/onsemiSiC>



Elektor editors conducted several in-person interviews with engineers and industry thought leaders at electronica 2022 in Munich. Visit the Elektor TV channel to watch all the videos: www.elektor.tv 

220652-01

ESP32 Camera

So Simple, It Doesn't Even Have to Use Wi-Fi



By Bera Somnath (India)

Have an idea for a camera application? Before you buy a new off-the-shelf camera, consider a more do-it-yourself approach. With an ESP32 board with a camera and a few additional components, you can build a simple, yet effective, custom solution.



All smartphones can take and store pictures and display them. Therefore, building yet another camera is not a task to be considered seriously by anybody, and certainly not by me. Unless it is really easy to do. With a few inexpensive modules like an ESP32 board with camera, a small OLED display, a real-time clock and a trigger mechanism, you can create a battery-operated camera of your own which will take pictures completely under your control. By pressing a button or by presence detection with a motion sensor or even triggered by a non-contact temperature sensor, it (discretely) takes the picture of someone whose body temperature is too high. Such applications make sense because you cannot deploy your smartphone in this way.

ESP32-Cam

By adding a PIR sensor to the ESP32-Cam board, you can rejig it to take pictures of intruders secretly! Or using just the ESP32-with-camera and OLED display, you can make a neat little webcam that connects to any of the available networks in the vicinity and then publish the video stream on the Intranet. Also, by deploying the port-forwarding feature of your modem or router, you can publish the picture on the Internet. This feature comes very

handy for those who need a second camera for demo purposes of their webinars and/or online classes. The cost of the ESP32-Cam module that we will be using in this article is around €10.

First, a Cool Demo

After installing the latest Arduino IDE and the most recent ESP32 boards package using the IDE's Boards Manager, select as board the "AI Thinker ESP32-CAM" (*Tools → Board → ESP32 Arduino*) board and the port it is connected to (*Tools → Port*). Then open the ESP32 webcam example project (*File → Examples → ESP32 → Camera → Webserver*). Try it out, you will be surprised. The two projects that follow below are extensions of this project.

Project 1: Webcam with OLED Display

The webcam project does not allow selecting a Wi-Fi network on the fly (i.e. the network credentials are hardcoded in the program) and the only way to know the IP address of the webcam is to connect it to a PC with a serial terminal running. Here we augment this project to allow it to connect to any network available in the vicinity. Once it is connected, it displays into which network it got connected and what its IP address is on

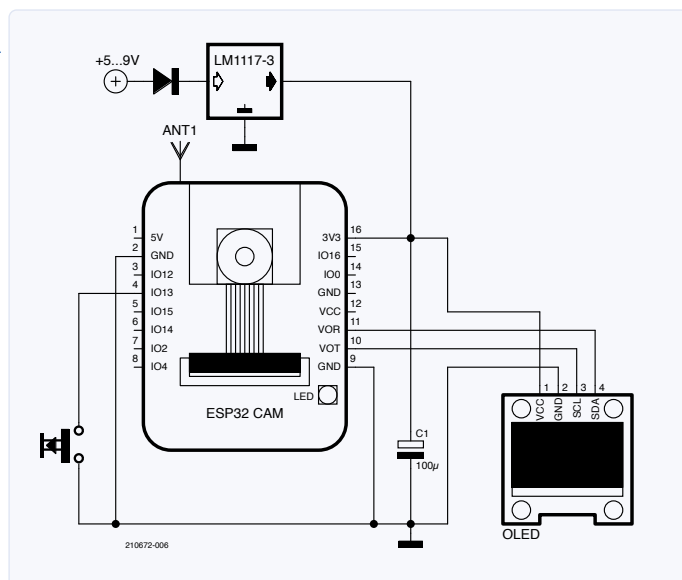


Figure 1: Add an OLED display to the ESP32-Cam for some extra comfort.

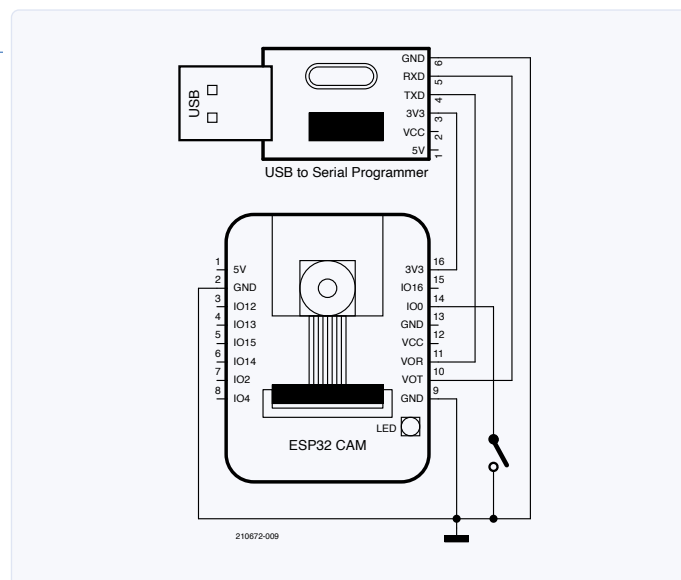


Figure 2: The ESP32-Cam does not come with built-in USB-to-Serial interface. Therefore, to upload the sketch one needs to have a USB-to-Serial adapter. To enter program-upload mode the reset button of the ESP32 must be pressed (or the power switched off and on) while IO0 is pulled to ground.

the tiny OLED. If, for some reason, it gets disconnected, it tries to reconnect to one of the known networks.

Building the Webcam

Figure 1 shows how to connect the OLED display to the ESP32-Cam board. Very few GPIO pins are left for use after the SD card and the camera are connected. The only free pins available for our use are IO1, IO3, IO4, IO12, IO13 (see inset 'Elektor Lab Notes'). IO1 ('UoT') and IO3 ('UoR') are also used as serial port for uploading the program to the ESP32 in flash programming mode. Therefore, these pins are to be completely free from any connections while uploading programs.

In this webcam project we can use pins reserved for the SD card pins easily as we are not using the SD card feature. However, we have used IO1 and IO3 as I²C pins for connecting the OLED display. That way we have maximized our free GPIO pins availability. Do not forget to disconnect the I²C bus when uploading a program to the ESP32 (**Figure 2**).

Usage

The working principle of the ESP32 webcam with OLED display has remained very simple. The ESP32 searches for the Wi-Fi networks provided as a list in the program and tries to connect to one of them. As soon as a connection succeeds, the camera-ready status shows up on the OLED display along with the network ID and the IP address that was allocated to the webcam. To access the camera, connect to the shown network and point a browser to the webcam's IP address. The port address is kept default (8080).

The software for the ESP32 webcam with OLED display can be downloaded from [1]. Note that it requires the library "ESP8266 and ESP32 OLED driver for SSD1306 displays" (we used version 4.1.0) [2]. It can be installed with the Arduino IDE's library

manager. Do not forget to disconnect the I²C bus before uploading a sketch.

Project 2: Triggered Camera with OLED & RTC

In this project a picture is taken every time an event occurs. Pictures are stored with a serial number and timestamp on an SD card.

The circuit (**Figure 3**) used is an extension of the previous circuit. A DS3231 (or DS1307) real-time clock (RTC) module is added on the I²C bus, and a trigger source is connected to IO13. This can

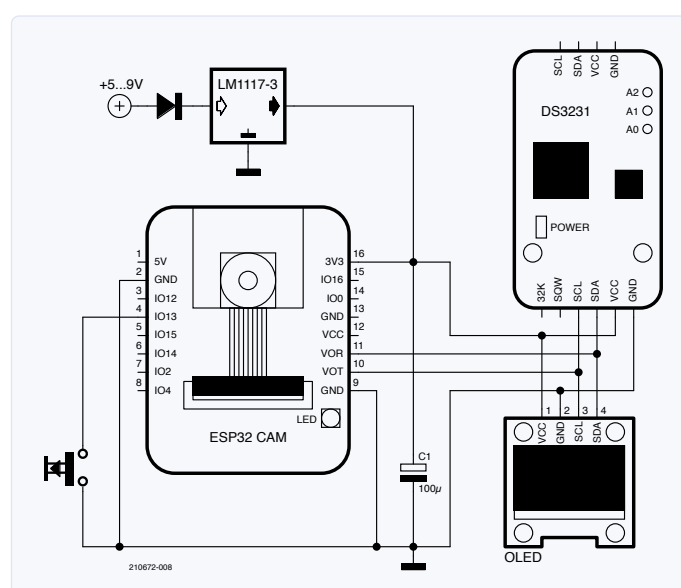


Figure 3: A push button and a real-time clock module turn the webcam from Figure 1 into a triggered time-stamping surveillance camera.

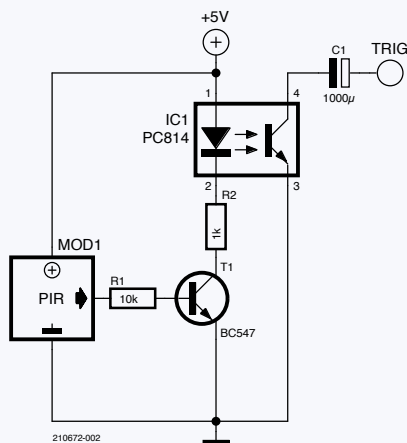


Figure 4: We can use a 5-V PIR sensor — some even work on 3.3 V — to trigger the ESP32 Camera. However, to make it fully isolated, we recommend using an optocoupler like this (or a relay).

be a pushbutton or a PIR sensor (Figure 4) or anything else that generates a short active-low pulse when a picture is to be taken.

IO4 is used as flashlight, deploying the ESP32-Cam on-board super bright white LED as an extra light source for the camera. IO13 is used as the trigger pin for taking pictures. The program stamps the date and time on the file name and stores it on the SD card. It then enters deep-sleep mode to preserve battery power during idle time when no pictures are being taken. An active-low pulse on IO13 wakes up the camera to take a new picture.

This program only consists of the function `setup`; the function `loop` is not used. The reason for this is the use of the deep sleep mode. When the device wakes up from this mode, it reboots and executes the function `setup` and takes a picture before going back to deep sleep, so the function `loop` is never reached.

The software for the triggered camera can be downloaded from [1]. Besides the OLED driver library required for the first project [2], this project also needs the library “Rtc by Makuna” (we used version 2.3.5) [3]. This library too can be installed with the IDE’s library manager. Do not forget to disconnect the I²C bus before uploading a sketch (Figure 2).

Conclusion

The ESP32-Cam is very cheap, yet it has got wide possibilities. By installing several of these cameras in line they can be used as a fixed and wide object scanner (50 to 100 meters) for taking pictures of approaching vehicle number plates to check for (e.g., speeding). In

solo mode with a PIR sensor, it can be used as a bird or other animal ‘snapper’. More elaborate applications can be imagined too. For instance, by adding face detection and a non-contact temperature scanner it can provide some anti-COVID security. ◀

210672-01

Questions or Comments?

Do you have technical questions or comments about this article? Contact Elektor at editor@elektor.com.



COMPONENT LIST

- ESP32-Cam board
- SD card
- 0.96" I²C SSD1306-compatible OLED display
- DS3231 / DS1307 I²C RTC module
- USB-to-Serial adapter (cable)
- LM1117 3.3 V regulator
- PIR Sensor or push button
- Optional: battery, capacitors, etc.



RELATED PRODUCTS

- ▶ **ESP32-Cam-CH340 Development Board (SKU 19333)**
www.elektor.com/19333
- ▶ **0.96" I²C OLED Display (SKU 18747)**
www.elektor.com/18747
- ▶ **SparkFun Real Time Clock Module – RV-8803 (Qwiic) (SKU 19646)**
www.elektor.com/19646

WEB LINKS

- [1] Downloads for this article at Elektor Labs: www.elektormagazine.com/labs/esp32-camera
- [2] ESP8266 and ESP32 OLED driver for SSD1306 displays: <https://github.com/ThingPulse/esp8266-oled-ssd1306>
- [3] Rtc by Makuna: <https://github.com/Makuna/Rtc>
- [4] Triggered Camera version Elektor Labs: www.elektormagazine.com/210672-01

Elektor Lab Notes

Many ESP32-Cam boards are sold together with a practical USB-to-serial-converter daughterboard.* (Maybe we should call it a motherboard as the ESP32 module plugs onto it and gets its power supply from it?). This combo (**Figure 5**) makes programming the ESP32 very easy, but it conflicts with the I²C port on ports IO1 and IO3 as used in the author's project. Using these pins, the I²C port must be disconnected every time you want to reprogram the ESP32, and so we looked for another way to connect the I²C bus.

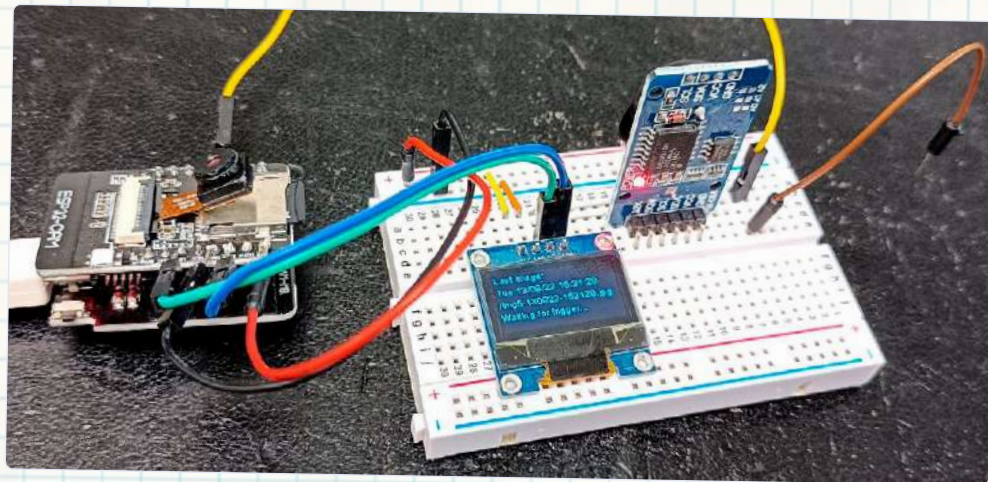


Figure 5: The Elektor Labs prototype built on a breadboard. The ESP32-Cam with daughterboard is on the left, the vertical module on the right is the RTC.

In theory, on the ESP32, the I²C bus can be configured to use almost any of its IO pins but doing so on the ESP32-Cam board results in all sorts of boot problems and PSRAM memory errors. This is due to the pull-up resistors on the I²C bus.

The ESP32-Cam board has two 8-way extension connectors. At a first glance that may seem okay but looking closer things are not so simple. First of all, six of the sixteen pins are used for power, leaving only ten for IO ports. Six of these (IO2, IO4, IO12, IO13, IO14 and IO15) are shared with the SD card slot. This leaves ports IO0, IO1, IO3 and IO16.

Uploading a program to the ESP32 with the daughterboard attached requires ports IO0, IO1 and IO3.

IO16 is shared with the chip select pin of the PSRAM chip, so it can only be safely used when PSRAM is not needed.

The SD card library defaults to 4-bit-data mode, but it can be put into 1-bit-data mode with

```
SD_MMC.begin("/sdcard",true);
```

Doing so liberates ports IO4, IO12 and IO13. Port 4 is shared with the (bright) white flash LED. It has a 47 k Ω pull-up resistor and a 10+1 k Ω pull-down resistor driving a transistor. Port 13 also has a 47 k Ω pull-up resistor and is used in this project as the camera trigger input.



Figure 6: It works, thumbs up!

IO12 is one of those ESP32 pins that must be handled with care at power-up as it determines the value of VDD_SDIO, the flash memory interface voltage. It must be pulled down at power-up on the ESP32-Cam board.

After a bit of experimenting, it was found that the I²C bus may be safely moved to ports IO0 (SDA) and IO3 (SCL, or the other way around. It doesn't matter as long as the software is configured correctly). Using these ports the I²C bus does neither interfere with the serial programming port nor with the boot mode as both IO0 and IO3 are supposed to be high at power up.

Finally, and unrelated to the above, know that IO33 is connected to a red LED on the ESP32-module-side of the board that you can use any way you like (as long as you keep in mind that it is active low).

We got a bit carried away with the second project and modified the program (**Figure 6**). It no longer uses emulated EEPROM for storing the image number but a file on the SD card instead. Simply delete the file "counter.txt" to reset the counter. Our program can be downloaded from [4].

* See the Related Products box for yet another version of the ESP32-Cam module.

ATX Power Supply for Raspberry Pi

By Sébastien Guerreiro de Brito (France)

Have a Raspberry Pi that needs power, as well as an old ATX power supply? A PCB, some components, and an ATtiny will connect the dots.



Figure 1: ATX power supply. (Source: Shutterstock)

The ATX form-factor PC power supply (**Figure 1**) was introduced by Intel in 1995. It's been the most common switch-mode power supply form factor for PCs since Pentium II-type processors. All its electronic, mechanical, environmental and other characteristics are specified by Intel to define the ATX standard.

The purpose of this circuit is to use an ATX power supply to power a Raspberry Pi board as well as the many peripherals that may be connected. The advantage of this power supply is the number of different voltages available – at interesting power levels.

ATX Power Supply Connector

However, in this article, we are not going to describe the internal workings of the ATX switch-mode power supply, but only try to

get it up and running. To begin with, let's look in detail at the connector (**Figure 2**) that is to be connected to the PC's motherboard.

The connector is composed of

- a 24-position, two-row Molex Mini-Fit Jr female connector (ref: 39-01-2240) or equivalent.
- Molex Mini-Fit HCS female contacts (ref: 44476-1112)

The pinout of this connector can be seen in **Table 1**.

The signals of interest for driving the power supply are PWR_OK and PS_ON#. The +5 VSB supply will also be used in our assembly.

The PWR_OK signal is a "Power good" signal

used by the power supply to tell the system that the +5 VDC, +3.3 VDC and +12 VDC outputs are present and conform to the thresholds.

The PS_ON# signal is the one that will allow us to start the power supply. It is a TTL-compatible signal, active-low, which allows the motherboard to control the power supply remotely. This allows software-driven power ON/OFF, Wake-on-LAN, wake-on-USB activity, etc.

When PS_ON# is pulled low, the power supply must activate the four main DC power rails: +12 VDC, +3.3 VDC, +5 VDC and -12 VDC. When PS_ON# is pulled to a TTL high level, or open circuit, the DC output rails must deliver no current and must be held at zero potential to ground. The ATX power supply circuitry provi-



Figure 2: ATX power connector.
(Source: Shutterstock)

des an internal anti-bounce circuit to prevent power ON/OFF oscillations if the PS_ON# signal is activated by a mechanical switch.

The characteristics of the PS_ON# signal are listed in **Table 2**. PS_ON# signal characteristics can be seen in **Figure 3** as graphs.

The PS_ON# signal has no effect on the +5 VSB output. The +5 VSB (SB = standby) supply is a supply that is present whenever the power supply is connected to the mains. This output provides power for circuits that must remain operational when the main power rails are disabled.

Description of the Electronic Board

To let the ATX power supply be controlled by the Raspberry Pi, a PCB was designed [1]. It contains the required logic to enable startup and shutdown of the Raspberry Pi by adding a small MCU to it. This also gives some other nice benefits. As we have an ATX power supply, we will use the 5 V rail to add a set of USB ports to the PCB that will allow peripherals that demand higher currents to take advantage of it. The PCB at work can be seen in **Figure 4**.

Parameter	Minimum	Maximum
V_{IL}	0 V	0.8 V
I_{IL}	–	–1.6 mA ¹
V_{IH}	2.0 V	–
V_{IH} open circuit	–	–5.25 V
Ripple/ Noise		400 mV pk-pk

¹ Note: Negative current indicates that current is flowing from the power supply to the motherboard.

Table 2: PS_ON# signal characteristics.

Pin	Signal	Wire color
1	+3.3 VDC	
2	+3.3 VDC	
3	COM	
4	+5 VDC	
5	COM	
6	+5 VDC	
7	COM	
8	PWR_OK	
9	+5 VSB	
10	+12 V1DC	
11	+12 V1DC	
12	+3.3 VDC	
13	+3.3 VDC	
14	–12 VDC	
15	COM	
16	PS_ON#	
17	COM	
18	COM	
19	COM	
20	Reserved	NC
21	+5 VDC	
22	+5 VDC	
23	+5 VDC	
24	COM	

Table 1: ATX connector pinout.

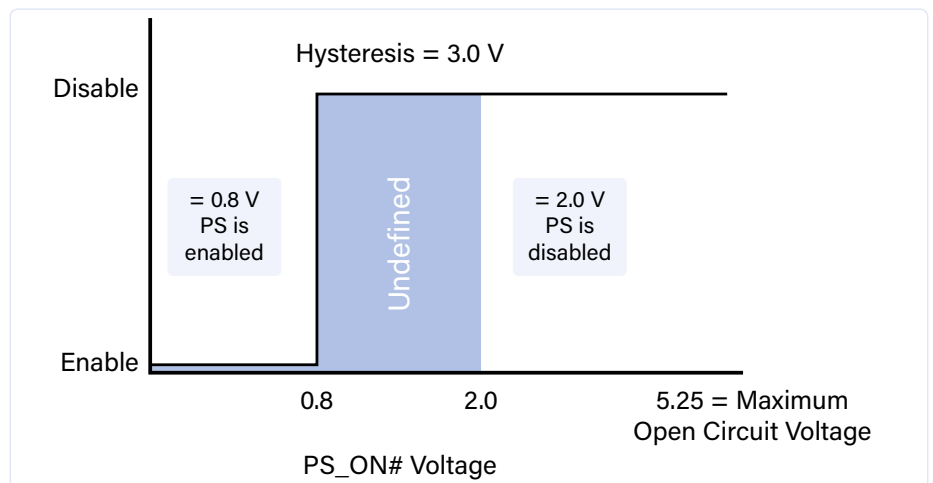
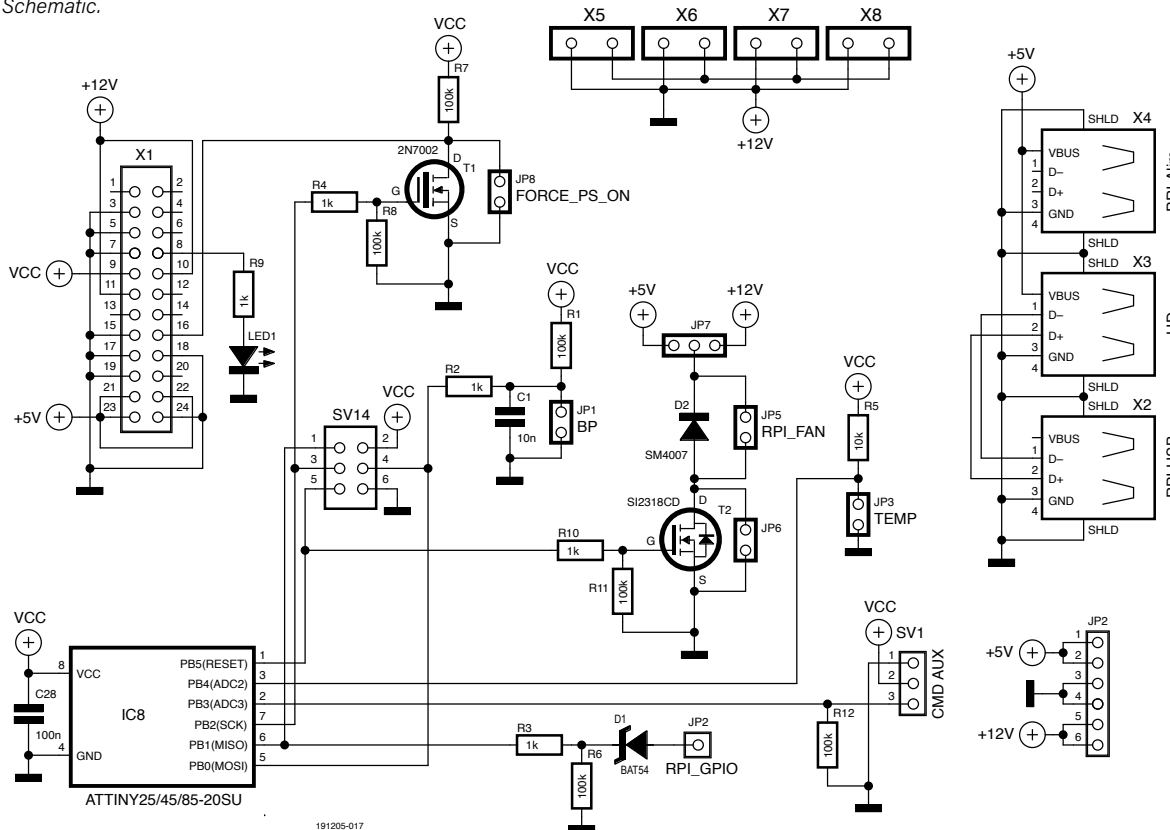


Figure 3: PS_ON# signal characteristics. (Source: ATX/ATX12V Power Supply Design Guide Version 1.1, Section 3.3.2, Figure 3)



Figure 4: ATX power supply connected to the PCB.

Figure 5: Schematic.



You can see the schematic of the PCB in **Figure 5**. The ATX power supply control scheme is relatively simple: It's based on the use of an ATtiny85 microcontroller powered directly from +5 VSB. The advantage of using a microcontroller is that, although the power supply control is simple, it allows for the addition of other features.

Starting the power supply

The ATtiny is instructed to start the power

supply by an external pushbutton soldered to jumper JP1.

When the pushbutton is activated, the microcontroller drives transistor T1, which starts the power supply. Jumper JP8 allows for starting the power supply in automatic mode.

Temperature management of the Raspberry Pi processor

We offer additional air-cooling for the

Raspberry Pi board. To do this, a 10 kΩ NTC thermistor is placed on it and, depending on the measured temperature, the board will trigger a fan. We select a fan voltage of either 12 V or 5 V by using jumper JP7.

Communication with the Raspberry Pi

To be able to turn off the ATX power supply from the Raspberry Pi, as on a PC, we set up a little stratagem.

The RPI_GPIO signal is connected to GPIO27 on the Pi board. The Raspberry Pi GPIO connector can be seen in **Figure 6**.

At operating system level, it is necessary to ensure, first of all, that GPIO27 is accessible. To do this, we create the GPIO initialization file (running at startup) with the nano editor:

```
nano S75gpiointit
```

Then, we enter the code from **Listing 1**.

We make the file executable and place it in the initialization folder.

```
sudo chmod +x S75gpiointit
sudo mv S75gpiointit /etc/init.d
```

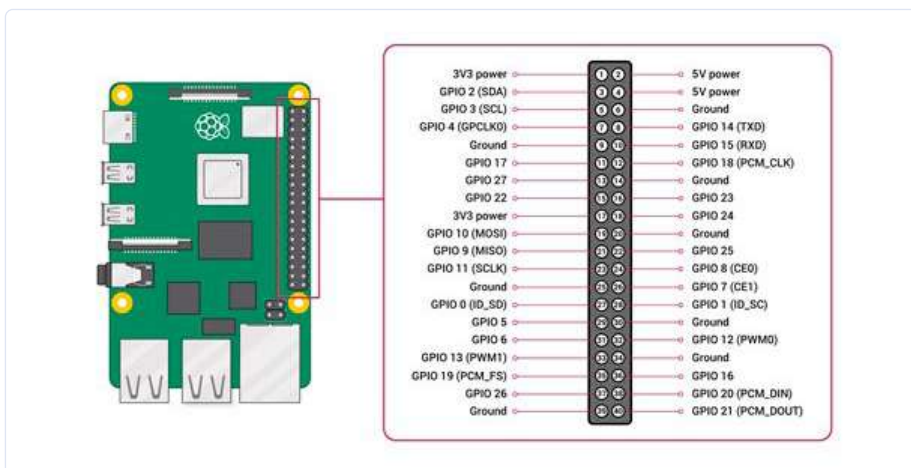


Figure 6: Raspberry Pi pinout (Source: [2]).

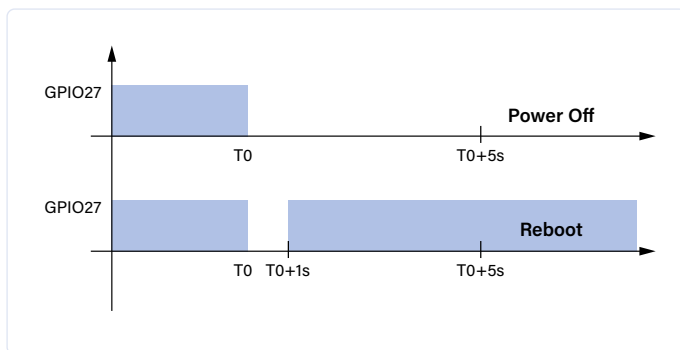


Figure 7: Management of the GPIO27 pin.

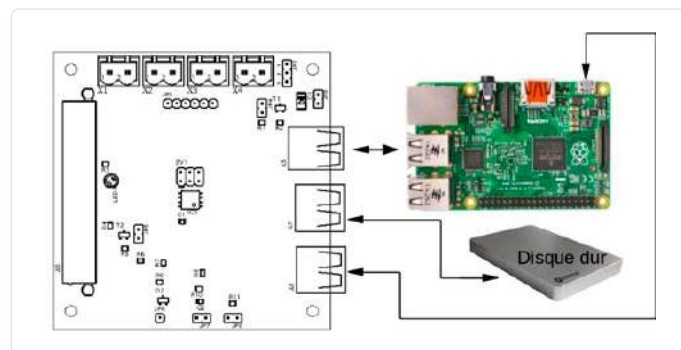


Figure 8: Connecting the Raspberry card.

You will notice in the initialization script that GPIO27 is initialized with a value of 1. So, when we turn off the Raspberry Pi via the OS (poweroff, for example), this entry will change back to 0 and the microcontroller will know that it's time to turn off the ATX power supply.

But, for a little more fun, we want to allow a reboot of the Raspberry card using the power supply reset. To do this, the trick we use is to vary GPIO27 in order to signal to the microcontroller that it's not a poweroff, but a reset, that's requested.

As you can see in **Figure 7**, the details of the management of GPIO27 is as follows: When the microcontroller sees the GPIO27 pin at 0, it waits for 5 seconds, then, if the pin is still at 0, it means that the RPi has been turned off, so it turns off the ATX power supply. If not, it



Listing 1: Shell script.

```
#!/bin/sh
#####
# I/O Init script #
# Author : Sebastien Guerreiro #
# Versions : #
# ----- #
# Mars 2019 V1.0 Création #
#####
CHEM_GPIO=/sys/class/gpio
#-----
register()
{
#Registers the outputs
if [ $SENS="out" ]; then
#Outputs
echo "$NUM" > $CHEM_GPIO/export
echo "$SENS" > $CHEM_GPIO/gpio$NUM/direction
echo "$VALEUR" > $CHEM_GPIO/gpio$NUM/value
chmod g+w $CHEM_GPIO/gpio$NUM/value
else
#Inputs
echo "$NUM" > $CHEM_GPIO/export
echo "$SENS" > $CHEM_GPIO/gpio$NUM/direction
fi
ret=$?;
if [ $ret -eq 0 ]; then
#echo_success;
echo "Registering GPIO$NUM : OK"
```

```
else
# echo_failure;
echo "Registering GPIO$NUM : ERROR"
exit $ret;
fi
}
#-----
unregister()
{
echo "$NUM" > /sys/class/gpio/unexport
ret=$?;
if [ $ret -eq 0 ]; then
echo_success;
else
echo_failure;
exit $ret;
fi
}
#-----
start()
{
echo "Registering GPIO--Setting Power GPIO ON"
#Output for power GPIO
NUM=27; SENS=out ; VALEUR=1
register;
}
#-----
start
exit 0
```



Listing 2: The shutdown script.

```
#!/bin/sh
#####
# Reboot/poweroff Management #
# Author : Sebastien Guerreiro (www.sebelectronic.com)#
# Versions : #
# ----- #
# Feb 2020 V1.0 Création #
#####
case $1 in
-r)
echo 0 > /sys/class/gpio/gpio27/value
sleep 1
echo 1 > /sys/class/gpio/gpio27/value
sleep 1
/sbin/shutdownSeb $@
;;
*)
/sbin/shutdownSeb $@
;;
esac
```

receives a reboot command and causes the ATX power supply to shut down for 2 seconds, before restarting it.

For all this to be possible we have to intervene with some Linux commands. On the operating system side, we rename the `shutdown` and `reboot` instructions contained in `sbin` to `shutdownSeb` and `rebootSeb`, respectively

```
sudo mv /sbin/shutdown /sbin/
shutdownSeb
sudo mv /sbin/reboot /sbin/
rebootSeb
```

Then, we create two scripts: `shutdown` and `reboot`, which we place in the `/sbin` folder. In the terminal, type:

```
nano shutdown
```

and enter the following (**Listing 2**) into the file. Save the file and exit nano.

In the terminal, type:

```
sudo cp shutdown /sbin
```

This will copy your script to `/sbin` so you can execute it later.

Next, we need to do the `reboot` script. Also in the terminal, enter:

```
nano reboot
```

and enter the following script (**Listing 3**) into that file. Save the file and exit nano. Then, in the terminal, type:

```
sudo cp reboot /sbin
```

This will copy your reboot script to `/sbin` and allow it to be executed later.

That's it!



Listing 3: The reboot script.

```
#!/bin/sh
#####
# Reboot Script #
# Author : Sebastien Guerreiro #
# Versions #
# ----- #
# Feb 2020 V1.0 Création #
#####
echo 0 > /sys/class/gpio/gpio27/value
sleep 1
echo 1 > /sys/class/gpio/gpio27/value
sleep 1
#/sbin/rebootSeb
/sbin/shutdownSeb -r now
```



RELATED PRODUCTS

- > **Raspberry Pi 4 2 GB (SKU 18965)**
www.elektor.com/18965
- > **Velleman VTSS220 Temperature-controlled Soldering Station (SKU 19865)**
www.elektor.com/19865

WEB LINKS

- [1] Project on Elektor Labs: www.elektormagazine.com/labs/atx-powersupply-for-rpi
- [2] Image Source: www.raspberrypi.com/documentation/computers/images/GPIO-Pinout-Diagram-2.png

Connecting the Peripherals

Using an ATX power supply allows us to connect the board to an IDE hard disk, for example, or an IDE CD-ROM drive in my case. In short, the advantage is that you can use the peripherals of your old PC for your new

projects. The connection is done as you can see in **Figure 8**.

If you'd like to know more about the project, you can visit the Elektor Labs page [1].

191205-02

Questions or Comments?

Do you have questions or comments about his article? Contact Elektor at editor@elektor.com or leave a comment on the projects page at Elektor Labs.



COMPONENT LIST

Resistors (0805, 0.1 W)

R1, R3, R6, R7, R8 = 1 k, 1%
R2, R4, R5, R9, R10 = 100 k, 1%
R11 = 10 k, 1%

Capacitors

C1 = 100 nF, 0805
C2 = 10 nF, 0805

Semiconductors

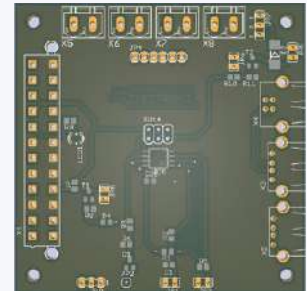
D1 = SM4007
D2 = BAT54
IC1 = ATTINY85
T1 = SI2318CDS

T2 = 2N7002

LED1 = LED, green, 3 mm

Miscellaneous

SV1, JP7 = Header male 3 points pitch 2.54 mm
JP1, JP3, JP5, J6, JP8 = Header male 2 points pitch 2.54 mm
JP2 = Header male 1 point pitch 2.54 mm
SV14 = Header male 2 rows 6 points pitch 2.54 mm
X5, X6, X7, X8 = MSTBVA 2.5 / 2-G-5.08 2-pin header
X4 = USB Type B connector



X1 = Molex Mini-Fit Jr straight connector, 5566, 24-way, 2 rows
X3, X2 = USB Type A connector PCB

Advertisement

Join the Elektor Community

- ✓ The Elektor web archive from 1974!
- ✓ 8x Elektor Magazine (print)
- ✓ 8x Elektor Magazine (digital)
- ✓ 10% discount in our web shop and exclusive offers
- ✓ Access to more than 5000 Gerber files



Take out a
membership!



www.elektormagazine.com/Member

elektor
design > share > earn

32 Ω

Headphone Amplifier

Simple But High-Quality 3-Chip Solution

By Thierry Clinquart (Belgium)

In the past, most good headphones had an impedance of 600 Ω . Today, 32 Ω versions dominate in the middle and higher price categories. Therefore, modern headphone amplifiers need a bit more power.

While, in principle, a normal opamp is sufficient for driving 600 Ω headphones very loudly, at comparable volume in the voice coils of modern 32 Ω headphones, significantly higher currents flow, which would cause normal opamps to clip. The classic solution with the old-but-good NE5532 audio opamps and comparable ICs is thus obsolete. So, in order for modern headphones not to sound distorted, the classic opamp circuit must be tricked out. Today, this very easy, not only in principle, but in practice. All you need to add is one BUF634A per channel.

The BUF634A

This is basically a small and fast push-pull power amplifier stage in an IC package. You only have to connect it behind an opamp and include it in the negative feedback – done!

Figure 1 shows the complete stereo circuit. A dual audio opamp of type OPA2134 [1] forms the two input stages and is responsible for the voltage amplification. Two BUF634As handle the amplification of current. Its basic internal circuit can be admired in **Figure 2**.

The special feature of this integrated push-pull output stage is its very high bandwidth, which can be set between 35 and 210 MHz via the current through Pin 1. If resistor Rx in Figure 1 is omitted (Pin 1 = open), the bandwidth is 35 MHz, and the quiescent current is only about 1.5 mA. The maximum bandwidth results if a value of 0 Ω is selected for Rx. In this case, the quiescent current rises to a tolerable 8.5 mA.

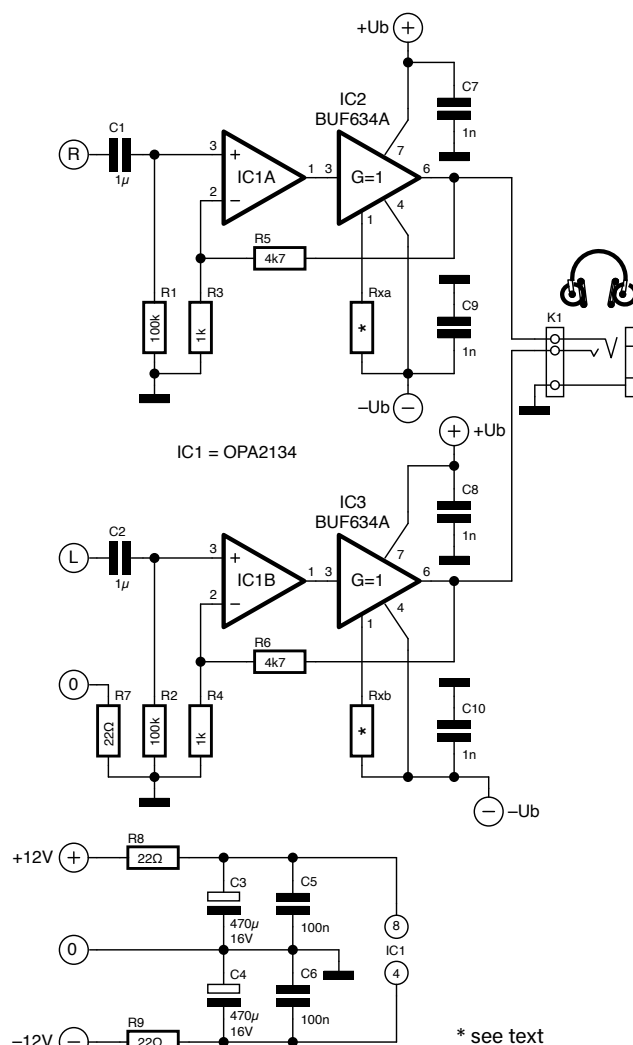
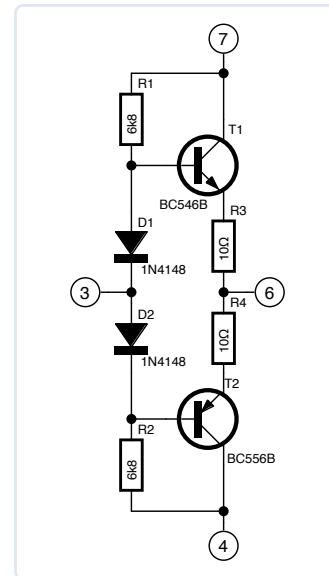
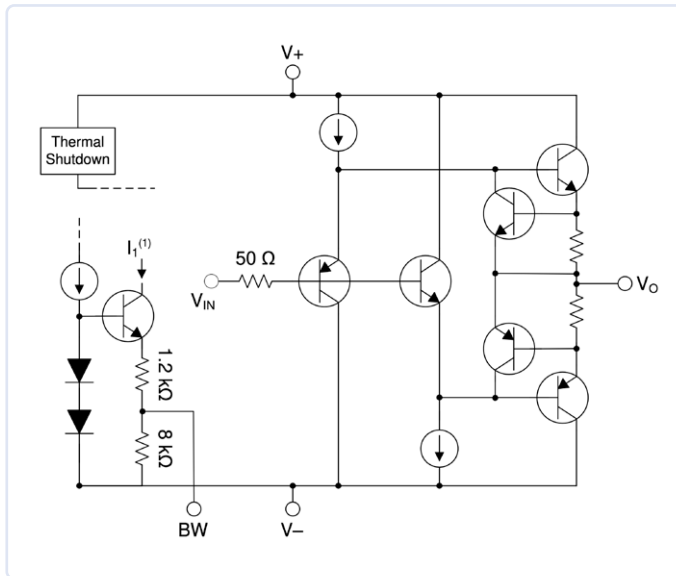


Figure 1: The circuitry of the complete headphone amplifier relies on integrated semiconductors.



The output stage can deliver currents of up to 250 mA - enough for headphones with impedances down to 8 Ω . For detailed information, please refer to the data sheet [2].

What bandwidth, and therefore what quiescent current, is the right one? For application as a small headphone amplifier, normally the low bandwidth should be okay, and you can do without Rx. For higher capacitive loads, however, the higher quiescent current with $R_x = 0\ \Omega$ is the better choice. The latter also applies to opamps, which are not stable at unity gain. Quiescent currents between extremes are easily adjustable with other values of Rx.

Fine-Tuning and Modification

With SMD components, you can build a superb headphone amplifier with minimal distortion, very low noise, and high bandwidth in a tiny space. But also, those who like wired components will be happy with this circuit. IC1 is also available in an 8-pin DIP version, and for IC2 and IC3, you can easily use a substitute for type BUF634. This IC is still available in an eight-pin DIP package. It will also work, although it is a bit slower, has a slightly higher quiescent current, and is no longer recommended for new designs.

An alternative is to replace the integrated BUF634A with a discrete equivalent circuit of small signal transistors, as shown in **Figure 3**. The quiescent current then depends on the semiconductors' respective combined features, and can be adjusted via the values of R1 and R2. Furthermore, for thermal stability, it should be ensured that D1 and D2 are in direct contact with T1 and T2. If necessary, they can also be glued together. Since the quiescent current is also dependent on the supply voltage, the amplifier should be operated with a stabilized ± 12 V supply, e.g. using 7812 or 7912 voltage regulator ICs. For $32\ \Omega$ headphones,

a power supply of ± 100 mA is sufficient. For $8\ \Omega$ headphones, about twice that is required. R7 reduces hum from potential ground loops.

Of course, you can experiment with other types for IC1 or T1 and T2 and get good results. When using headphones with impedances $\geq 32\ \Omega$, there are no heat problems with the BUF634A. At $8\ \Omega$ and prolonged

Resistors

(default: metal film, 1%)

 $R_1, R_2 = 100 \text{ k}$ $R_3, R_4 = 1 \text{ k}$
$$R5, R6 = 4k7$$
$$R7...R9 = 22 \, \Omega$$

Rxa,Rxb = see text

Capacitors

C1,C2 = 1 μ , 25 V, film

C3,C4 = 470 μ , 16 V, electrolytic

$$C5, C6 = 100 \text{ n}, 25 \text{ V}$$

C7...C10 = 1 n, 25 V, ceramic

Semiconductors

IC1 ≡ OPA2134

IC2,IC3 = BUF634A (see text)

Other

K1 = Headphone socket, stereo, 6.3 mm



listening at high volume, SOIC ICs can get quite warm. Then, when designing a board, make sure that the heat is dissipated via the pads to a sufficient copper surface, or choose the 8-pin DRB package that has a “thermal die pad” on the bottom. The BUF634 type (without the “A”) is even available in an easily cooled TO-220 or TO-263 package.

It's best to use film capacitors at the input for C1 and C2. There is no need for capacitors at the output because the typical offset voltage will be around a few mV, which doesn't cause significant movement of the voice coils. However, if you want to be absolutely sure about this, you can put a capacitor between the output and the headphones. For 32 Ω impedance, a 470 μ F (25 V) bipolar type would be sufficient. 8 Ω headphones need at least 1000 μ F. The output is short-circuit proof – but not if you use the BUF634 replacement from Figure 3.

Conclusion

Modern analog electronics enable tiny headphone amplifiers with distinctly audiophile characteristics. Distortion factors in the range of <0.01% at medium volumes are readily achievable and noise is virtually inaudible thanks to IC's good properties. The amplifier can deliver up to 300 mW to 32 Ω headphones (and still up to 250 mW to 8 Ω loads with good cooling), which is much more than you should present to your ears. ◀

200441-01

About the Author

A trained electronics technician, Thierry Clinquart has directed his passion toward audio. All his projects revolve around analog audio – preamplification, dynamic processing, corrections, signal distribution, etc. He makes his own PCBs with Sprint-Layout from Abacom and sPlan for the diagrams.

Questions or Comments?

If you have technical questions, feel free to e-mail the Elektor editorial team at editor@elektor.com.



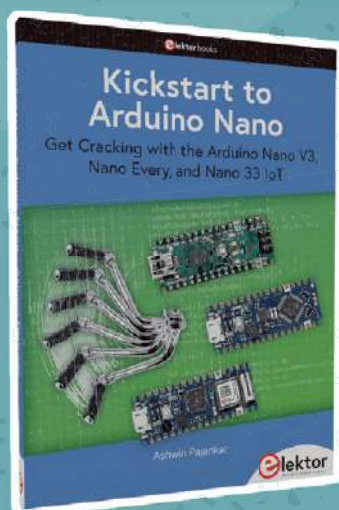
Related Products

- > **OWON SDS1102 2-ch Oscilloscope (100 MHz) (SKU 18782)**
www.elektor.com/18782
- > **PeakTech 3350 True RMS Digital Multimeter (6000 Counts) (SKU 19986)**
www.elektor.com/19986
- > **Joy-IT JDS6600 Signal Generator & Frequency Counter (SKU 18714)**
www.elektor.com/18714

WEB LINKS

[1] OPA2134 Datasheet: <https://ti.com/product/OPA2134>

[2] BUF634A Datasheet: <https://ti.com/product/BUF634A>



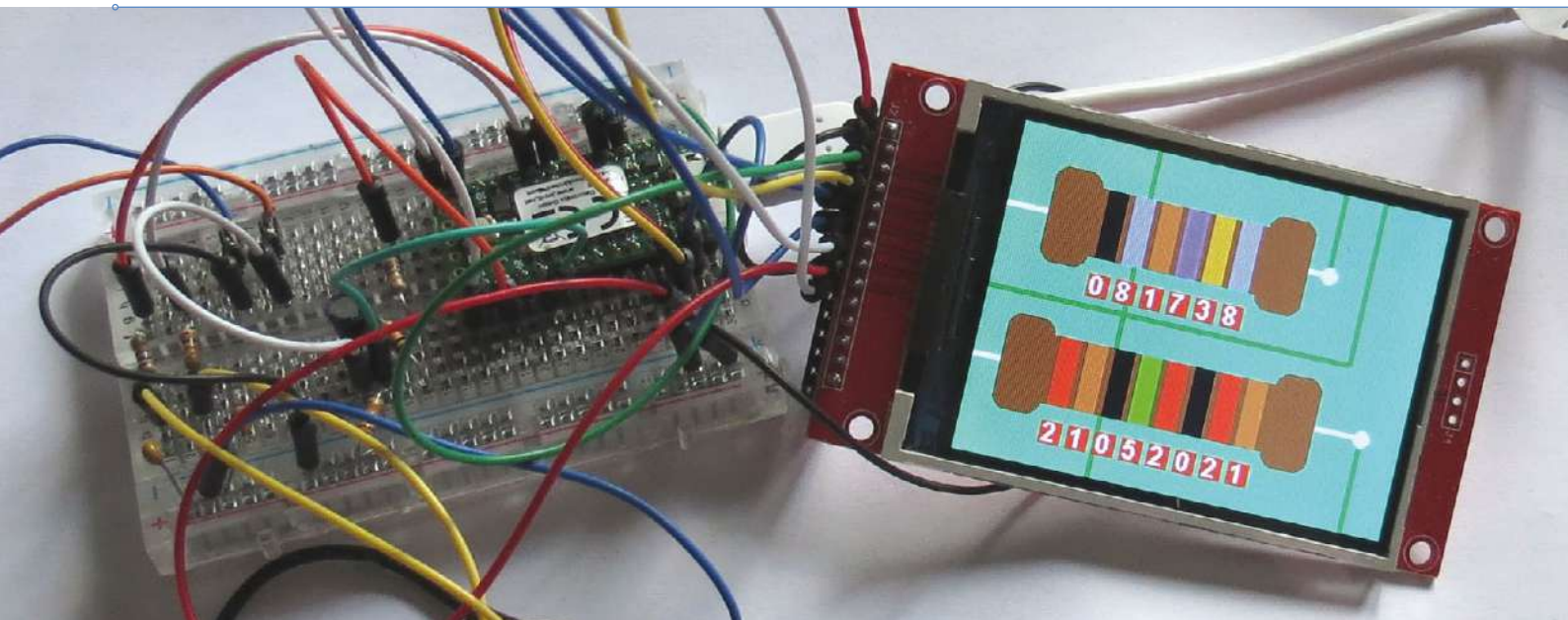
Kickstart to Arduino Nano

This book serves as the first step for novices and microcontroller enthusiasts wishing to make a head start in Arduino programming. The book follows a step-by-step approach to explain concepts and the operation of things. Each concept is invariably followed by a to-the-point circuit diagram and code examples. Next come detailed explanations of the syntax and the logic used. By closely following the concepts, you will become comfortable with circuit building.

www.elektor.com/20241



elektor



The SDR radio clock is formed by the Teensy 4.0 microcontroller board with connected display on a breadboard. The time is displayed in resistor color code here.

SDR Radio-Controlled Clocks

By Prof. Dr. Martin Ossmann

Today's microcontrollers are so powerful that they can be used to build radio-controlled clocks based on the software-defined radio (SDR) principle. How this works in principle has been shown by the MSF radio clock [1] built around a Raspberry Pi Pico. This article, too, is about time signals, but this time the fast Teensy 4.0 board is used as the microcontroller.

Table 1: Receivers are implemented for these transmitters.

60.0 kHz	MSF	British time signal transmitter NPL
77.5 kHz	DCF77	German time signal
129.1 kHz	EFR	Radio ripple control
162 kHz	TDF	French time signal
198 kHz	BBC	AM broadcasting with phase modulation

In this project, we are developing a receiver for the signals of the EFR radio service, which is used by energy suppliers for ripple control. It also transmits time information at regular intervals. Building a radio clock with it is something new. The same SDR-based concept and hardware can be used to receive other time signals, which we plan to use extensively (**Table 1**). Another important development goal is to show the time on various innovative displays.

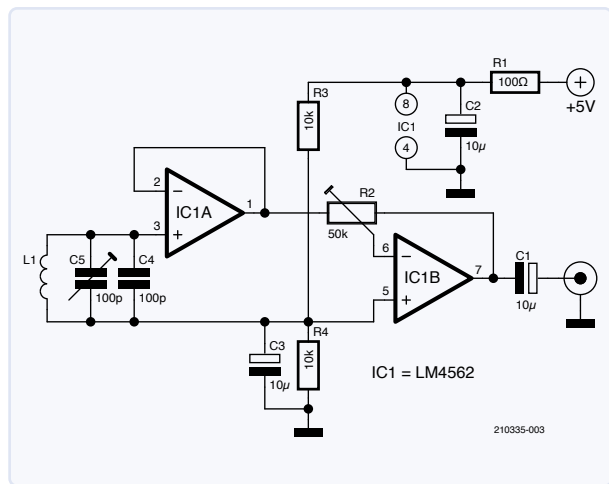


Figure 1: Circuit diagram of the active antenna for 129.1 kHz. The 250 turns of 0.2 mm² enameled copper wire are wound on an ETD29 coil former. The coil is pushed onto a 150 mm × 8 mm ferrite rod made of material 3B1. By adjusting C_x, the circuit is usable for frequencies from 60 kHz to 200 kHz.

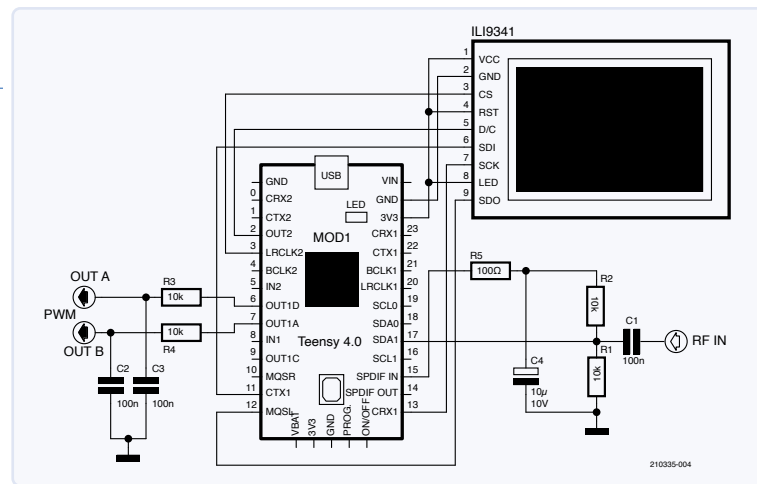


Figure 2: Circuit diagram of our software-defined radio for radio-controlled clocks.

One Concept for All Frequencies

To receive the radio signals, the first thing we need is an active antenna, as in the circuit diagram in **Figure 1**. The signal of the ferrite rod (150 mm × 8 mm, material 3B1, coil former ETD29, winding 250 turns CuL 0.2 mm²) is filtered with an resonating circuit and then buffered by an op-amp. The second stage is another op-amp, the gain of which can be adjusted to local conditions using trim pot R2. Our receiver works best when a signal of around 50 mV_{SS} is present at the output of the two-stage amplifier circuit. The circuit is suitable for all time signal frequencies used here; if you want to use the antenna for other frequencies, you will have to adapt capacitor C4.

The complete system schematic is shown in **Figure 2**. The main component is a Teensy 4.0 board. With resistors R1 and R2, we set the offset of the A/D converter, and with C1 we couple the signal from the active antenna to the A/D converter of the Teensy board. We have built our prototype on a breadboard, as was shown in Figure 1. The two low-pass filters, R4/C2 and R3/C3, filter the PWM signals from outputs 6 and 7. We use the PWM signals for debugging, for example to display the I and Q signals on an oscilloscope. The display is connected via the SPI connector and some additional pins. Power is supplied via the USB port, which is also used to program the board and output the debugging texts. This allows you to watch the reception of the data in the serial monitor of the Arduino IDE. The Teensy board can be programmed in the normal Arduino development environment.

Receiving Time Codes

The “universal” concept refers not only to the receiver’s hardware, but also to its software. The structure remains similar for all time signal transmitters listed here. But, since the demodulation and decoding routines are very different, it seemed to make more sense to me to write separate firmware for each time signal transmitter than to pack all routines into a single firmware and activate the desired one via a switch, for example. The complete and documented software package can be downloaded from the project page at [2]. However, the basic operation of the receiver will first be described in more detail, using the reception of the EFR long-wave transmitter on 129.1 kHz as an example.

EFR Reception

EFR GmbH operates three long-wave transmitters (129.1 kHz, 139 kHz, and 135.6 kHz) for ripple control at energy plants [3]. The transmitters in Germany and Hungary each have a range of at least 500 km and thus cover most of Central Europe. The modulation type used is frequency-shift keying (FSK)

with a deviation of ±170 Hz and a data rate of 200 bps. In addition to the encrypted proprietary messages, time information is sent as well. The protocol is specified in IEC 60870-5.

The concept of our EFR receiver is shown in **Figure 3**. The signal from the active antenna reaches the internal ADC of the Teensy 4.0 board, which is clocked at 333.333 kHz. Thus, the sampling theorem is fulfilled. As usual with SDR receivers, IQ multiplication is then performed. The oscillator signal is generated with a DDS generator implemented in software. After multiplication of the input signal with the I or Q oscillator signal, low-pass filtering is performed. The low-pass filters are dimensioned in such a way that the received signal is not yet clipped, but no other signal components can pass through, either.

To demodulate the RTTY signal, we must determine the currently-received frequency. At +170 Hz offset, it is a one, at -170 Hz, it is a zero. For this purpose, we determine the current phase from the I/Q signals.

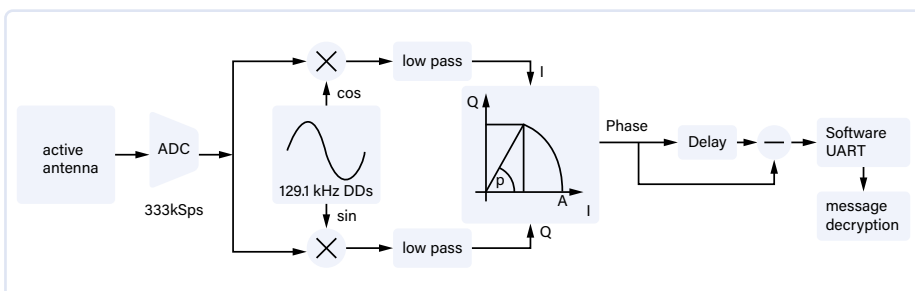


Figure 3: The concept of the EFR receiver.

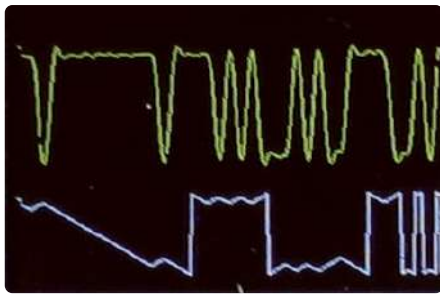


Figure 4: Debug information on the display. Upper graph: Demodulated RTTY signal (instantaneous frequency); Lower graph: Instantaneous phase.

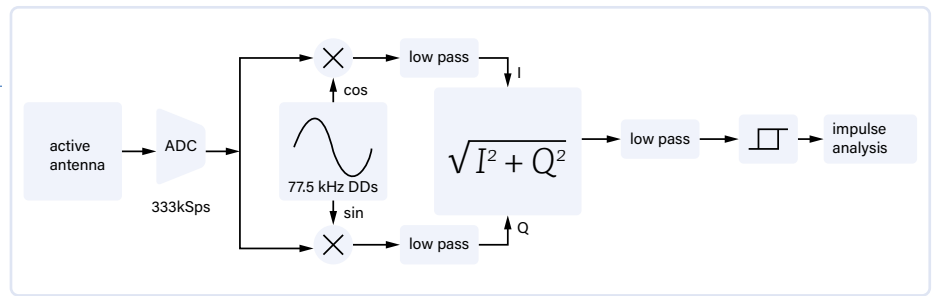


Figure 5: Concept of the DCF receiver.

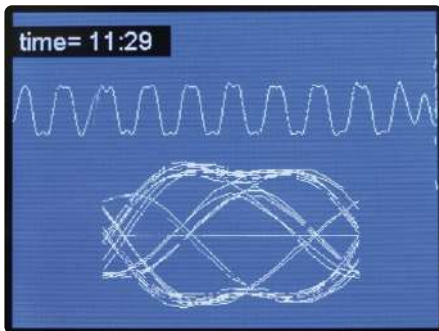


Figure 6: Debug information from BBC reception. Top: Recovered data signal; Bottom: Eye diagram.

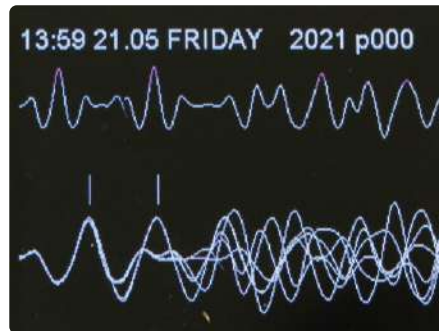


Figure 7: Debug signals from TDF reception.

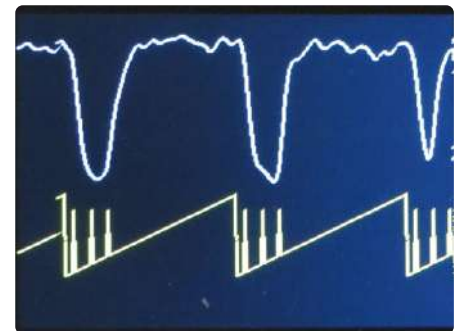


Figure 8: MSF debug info. Top: Receive level with second markers; Bottom: Timer variable with sample points.

We do this with the `atan2()` function. From the phase change over time, we can determine the frequency. To achieve this, we calculate the difference between the original and the delayed signal, resulting in the demodulated signal, which we pass to a software UART to recover the transmitted bytes. That way, we recover the transmitted messages, from which we can extract the time information. **Figure 4** shows the debug signals on the display, at the top the demodulated RTTY signal (instantaneous frequency), and at the bottom the instantaneous phase.

DCF Reception

The concept of the DCF receiver (**Figure 5**) is very similar to the EFR concept. However, this time, the instantaneous transmit amplitude is determined from the I/Q signals. For this purpose, we calculate $\sqrt{I^2 + Q^2}$.

This signal is low-pass filtered and routed to a Schmitt trigger. This makes the second pulses available for evaluation, and we can determine the date and time from the bits.

BBC Reception

The BBC broadcasts a radio signal on

198 kHz. The carrier is phase-modulated and used for data transmission similar to the RDS system in FM radio [4]. The phase modulation is demodulated in the same way as in the EFR system. The bit clock of 25 bps must be recovered from the demodulated signal (**Figure 6** above). To do this, the receiver forms the eye diagram, so to speak, and varies the clock so that the eye opens to the maximum. The data is then sampled in the center of the eye. From the bit stream obtained that way, the block boundaries must now be recovered. The data is sent in blocks of 50 bits each, of which 13 bits are check bits. The receiver searches the bit stream for error-free blocks and recognizes the block boundaries by their positions. There are different block types that contain different data. Block type 0 is intended for the transmission of the time information and is evaluated by our receiver.

TDF Reception

Until 2016, TDF (formerly *Télédiffusion de France*) broadcast a radio program on 162 kHz, but, today, only the carrier is transmitted using phase modulation. Time information is included in the data. A zero bit is sent as

a single pulse, a one bit as a double pulse, each starting at the beginning of the second. **Figure 7** shows the received phase information (upper curve) and at the bottom, as a kind of eye diagram, superimposed curves showing the pulses on the left between the vertical lines. In the 59th second of each minute, the signal is not phase-modulated. Our receiver uses this pause for synchronization. The individual bit information is structured quite similarly to DCF77.

MSF on 60 kHz

On 60 kHz, the NPL (National Physical Laboratory) in Great Britain operates a long-wave transmitter which broadcasts time information at 15 kW. Simple on-off carrier modulation is used [5]. The minute starts with a pause of 500 ms. The subsequent seconds each start with a carrier gap of 100 ms and then further two gaps of 100 ms if necessary, depending on the bits to be transmitted. Our MSF receiver is built quite similar to the DCF receiver, only the pulse-decoding is done differently. Compared to the other signals, the MSF signal is relatively weak, but can still be received quite well at the western border of Germany (**Figure 8**).

Display Options

When the author decided to build an EFR-based radio-controlled clock, he also had to decide which display should be used. The decision was made to use a 2.8" LCD with 320x240 pixels, which can be used to realize a wide range of graphic options. The display is based on an ILI9341 controller and can be obtained, for example, at eBay. Alternatively, the little brother of the display with a diagonal of 2.2" but the same resolution from the Elektor store (see Related Products text box) can be used. The display is connected to the Teensy board via the SPI interface.

Resistor Display

Our EFR Clock offers a somewhat unusual new display option, the "resistor display" (Figure 9). The digits of time and date are displayed color-coded as on electrical resistors. There are two resistors in the display, one with six rings for the six digits of the time (hours:minutes:seconds, for example, 12:23:45). The second resistor has eight rings for the date (day.month.year, for example, 23.08.2012). The time is updated

every second, and the local clock is always synchronized when time data is received from the EFR service. For users who do not yet know the resistor code, the time and date are also displayed using standard digits. Purist readers might point out here that resistors have a maximum of six rings, but at least this allows beginners to learn the DIN41429 color code in a fun way.

Domino Display

As another funny display idea, a domino display has been implemented (Figure 10). On dominoes, you can display the numbers from 0 to 9. Each domino provides space for two digits, so you need three dominoes to represent the time and four to represent the date. Whether the resistor display, the domino display, or another option is used is determined in the source code using option bits (Listing 1). The option to be chosen is selected with a "1," and the other options are switched off with a "0."

Word Clock and Analog Style

In addition to the displays described so far, you can also select a word clock display,

where the time is shown as plain (German) text in the display (Figure 11). With this option, only the time is displayed, not the date. It is updated every five minutes. The classic display format of an analog clock is also implemented (Figure 12). The time is displayed on a round dial, and the date as text below it.



Listing 1: The display is selected via these constants.

```
#define scopeShow 0
#define scopeClkShow 1
#define resShow 0
#define dominoShow 0
#define wordShow 0
#define sevenSegShow 0
#define AP6571Show 0
```

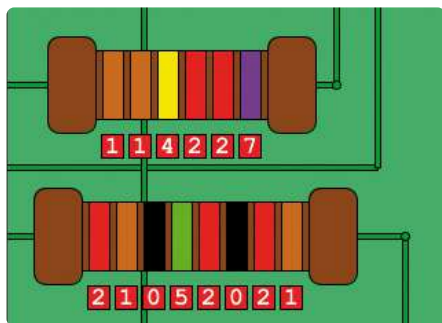


Figure 9: Resistor color code display option.

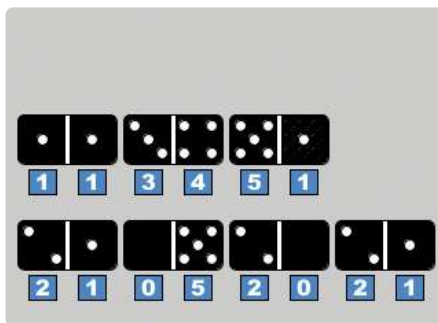


Figure 10: Domino display.



Figure 11: A word clock is also implemented.



Figure 12: Classic analog clock.



Figure 13: Nostalgic seven-segment display.



Figure 14: The time represented as if from a dot-matrix printer.

Seven-Segment Display

Of course, we also have a simple, nostalgic seven-segment display on offer. So, if you want to be reminded of the old days, this is the right choice (**Figure 13**).

Character Generator Display

As a last option, we offer a simple text display based on the legendary 6571AP character generator ROM, once widely used in 9-pin dot matrix printers (**Figure 14**).

Overall, this project provides many combinations and options. You can choose between five radio clock variants from EFR to DCF and

between six display options, so there should really be something in the mix for everyone. If you'd like, you can also use this software, for example the display options, as a starting point for your own developments. ◀

210335-01

Questions or Comments?

Do you have technical questions or comments about this project? Then contact the author at ossmann@fh-aachen.de or the Elektor editorial team at editor@elektor.com!



Related Products

- > **Teensy 4.1 Development Board (SKU 19311)**
www.elektor.com/19311
- > **2.2" SPI TFT display module ILI9341 (240x320) (SKU 18419)**
www.elektor.com/18419

WEB LINKS

- [1] Martin Ossmann, "Raspberry Pi Pico Makes an MSF-SDR," ElektorMag 7-8/2022: <https://elektormagazine.com/magazine/elektor-260/60555>
- [2] Firmware on the project page: <https://elektormagazine.com/210335-01>
- [3] EFR signal: https://mee.hu/files/images/5/B_Sbick_EFR-CEE_Lakihegy.pdf
- [4] L.F. Radio Data specification: <https://downloads.bbc.co.uk/rd/pubs/reports/1984-19.pdf>
- [5] MSF 60 kHz Time and Date Code: www.npl.co.uk/products-services/time-frequency/msf-radio-time-signal/msf_time_date_code

elektor e-zine

Your dose of electronics



Every week that you don't subscribe to Elektor's e-zine is a week with great electronics-related articles and projects that you miss!

So, why wait any longer? Subscribe today at www.elektor.com/ezone and also receive free Raspberry Pi project book!

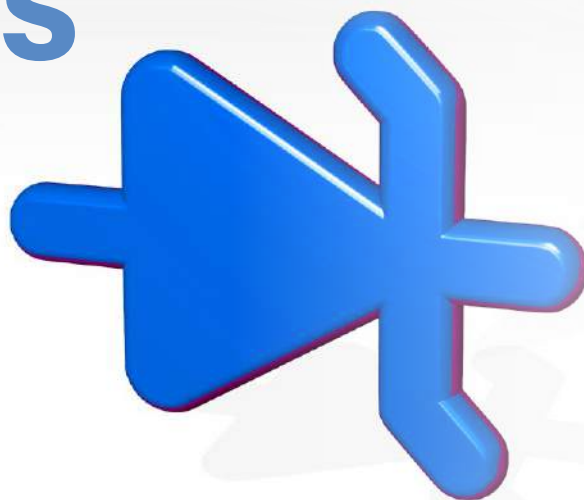


Starting Out in Electronics

Special Diodes

By Eric Bogers (Elektor)

In this installment, we conclude the chapter on diodes with a few rather exotic ones, which still exist even today, such as thyristors, triacs, LEDs and optocouplers.



The first generation of dimmer circuits was built with thyristors - triacs did not exist yet (at least not for the required levels of power). Nowadays, dimmers make use almost exclusively of triacs.

The Thyristor

A thyristor (**Figure 1**) is basically nothing but a diode with a control terminal – when the thyristor conducts, it will only do so in the forward direction, just like a regular diode. Therefore, for AC applications, we need two thyristors (or a rectifier).

To switch a thyristor into conducting mode, it has a third connection called the gate (the other two connections are called the *anode* and *cathode*, just as with a normal diode), where it needs an 'ignition pulse.' Once a thyristor is ignited, it continues to conduct as long as a current flows through it – no need to keep a voltage on the gate. To turn the thyristor off (returning it to reverse state), the current flow through it must be interrupted. In a DC circuit, a thyristor will conduct 'forever,' in an AC circuit, only until the next zero crossing.

There are two ways to construct a dimmer with a thyristor: The alternating voltage could be rectified, or we can use two thyristors

connected antiparallel. As dimmer circuits constructed in this way no longer have any practical use, we will not go into further detail.

Thyristors are often used as protection against overvoltage, as shown in **Figure 2**. If the stabilizer or a power transistor burns out in the grid supply, then, without protection, (much) too high a voltage will be applied to the connected components and devices, which could destroy them. This is a situation we should try to avoid by all means.

However, as soon the voltage in the circuit of Figure 2 becomes too high, the Zener diode trips and ignites the thyristor, and, as a result, the thyristor will conduct continuously. This causes a short circuit, causing the fuse to trip. This is the lesser of two evils when compared with a completely destroyed electronic circuit.

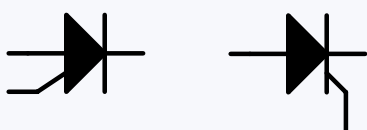


Figure 1: The thyristor.

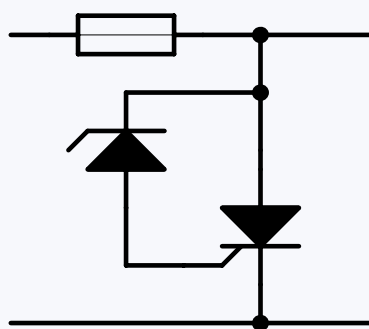


Figure 2: Overvoltage protection (principle).

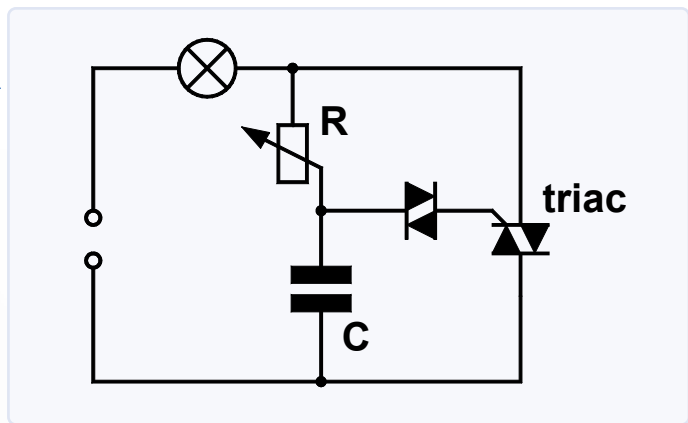


Figure 3: Conventional (read: obsolete) dimmer circuit.



Figure 4: Phase-cut.

Depending on the type of thyristor, it requires a voltage ranging from 0.8 ... 2.5 V for ignition. The Zener voltage should therefore be carefully selected, so that in case of an overvoltage this ignition voltage level will definitely be reached.

The Triac

Just like a diac, which is basically a combination of two Zener diodes, a triac is a combination of two thyristors – it exists solely for the purpose of realising AC applications using just one component. Let's have a look at the triac in a practical circuit – the conventional dimmer circuit, as shown in **Figure 3**. This kind of dimmer operates on the principle of phase cutoff: depending on whether a smaller or larger part of the phase (period) is passed through, the connected light will illuminate dimmer or brighter. Refer to **Figure 4** for details.

The triac turns on when the voltage across the diac increases above approximately 33 V and provides the ignition pulse for the triac. When potentiometer R in Figure 3 is set to its lowest stop, the voltage is divided by the RC network on one hand and shifted by 90° in phase on the other – the voltage across the diac will never reach a value of 33 V. Subsequently, if the potentiometer is gradually adjusted, the voltage across the diac increases (and its phase also shifts less and less) with the result being that the triac will ignite at an accelerated, earlier point in time.

The circuit in Figure 3 is one of the simplest (grid voltage) dimmers imaginable: it lacks any form of interference suppression, while no measures have been taken against switching hysteresis either. Many improved dimmer circuits are available nowadays, but they are much more complicated.

Of course, one could dim a light bulb by connecting an adjustable (power) potentiometer in series, but a large part of the mains AC power in that potentiometer would be lost as dissipated heat. The

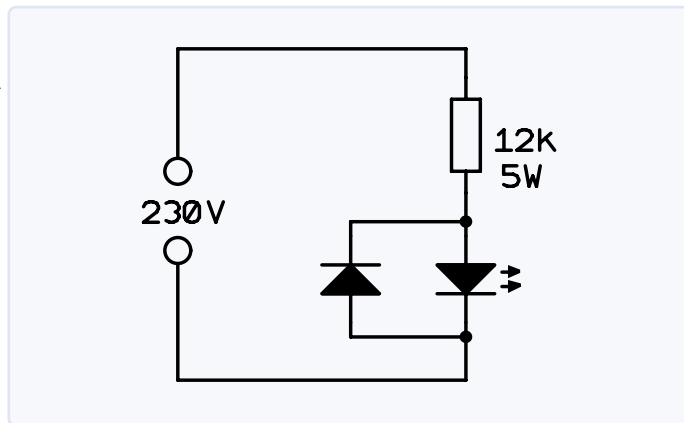


Figure 5: LED with flyback diode.

advantage of the phase-cut is that this power loss will not occur.

Light-Emitting Diodes and Optocouplers

Light-emitting diodes are diodes that emit light as soon a current flows through them. The forward voltage is about 2 V, while the maximum current is normally about 20 mA (for so-called *low-current* LEDs, this maximum current is in the range of only 2 mA).

LEDs have replaced regular light bulbs in control panels for many years. They are characterized by a very long lifespan, relatively low power consumption and great shock and vibration resistance.

Nowadays, LEDs are available in all possible colors, and laser LEDs are also available (to which we owe the CD and DVD player) whilst Neopixel LEDs with a built-in microcontroller enable all colors to be emitted, not to mention various lighting effects. LEDs have also made their appearance in lighting technology as (mandatory) replacements for incandescent bulbs.

Although all this is very interesting, a detailed discussion of these topics is far beyond the scope of this basic course. We will, therefore, limit ourselves to a brief general discussion, in honor of this small but very important component.

LEDs Powered by AC Voltage

In many cases, the allowable LED reverse voltage is only a few volts. In case they are powered from an AC voltage, a flyback diode must be connected to limit the reverse voltage across the LED to a safe value (see **Figure 5**). By the way, you can see that the schematic symbol of an LED is a regular diode, but with a double arrow symbolizing its light emission.

Optocouplers

An optocoupler consists of a light-emitting diode and a light-sensitive component (often a phototransistor) combined in one package (**Figure 6**). As soon as a voltage is applied across the LED, the LED will light up, causing the phototransistor to conduct. The major advantage of this seemingly-complicated approach is the galvanic separation between LED and phototran-

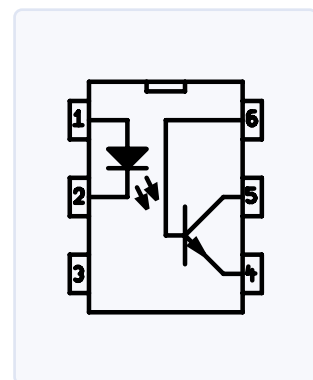


Figure 6: Optocoupler.

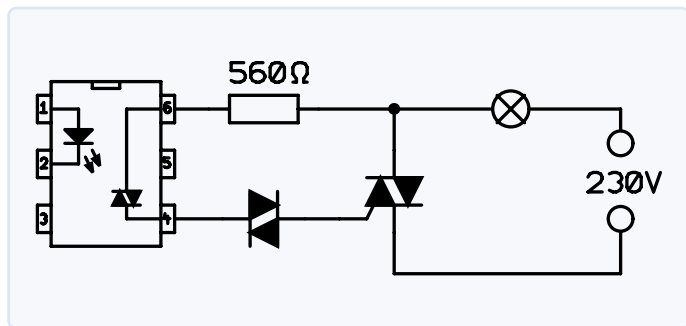


Figure 7: Optocoupler with phototriac.

sistor: A voltage difference of (often) several hundred volts between these components may be possible without causing any problems. Optocouplers are widely used to keep the power section of dimmer or other circuits separated from the control circuitry. For example, the high-power section can be supplied from the grid while the control circuitry remains unaffected by it (just for safety reasons).

Optocoupler with Phototriac

A transistor only conducts in one direction (as we will see in the next installment), and this can be inconvenient for some applications. If we wanted to drive a triac with that, we would need a dedicated voltage and the triac would be triggered with an ignition pulse of opposite polarity every second half of an AC voltage period, resulting in asymmetric behavior. To make a long story short, a regular optocoupler is by no means ideal for such applications. Therefore, in such cases, we prefer using an optocoupler with a phototriac rather than a phototransistor (**Figure 7**).

When any current is supplied to the LED and it lights up, the phototriac starts to conduct and in turn switches the external triac. For light loads (such as a small fan), the external triac can be eliminated, if necessary, and the load can be powered directly by the phototriac.

The circuit in Figure 7 should be able to function even without a resistor and diac. However, two problems can occur:

- Without the diac, the circuit tends to trip in an uncontrolled fashion: the triac starts to conduct without a control pulse from the LED.
- Many power triacs cause the phototriac to burn out if the current is not limited by a resistor.

Optocouplers with Zero-Crossing Detection

When the current is turned on somewhere in the middle of the current phase, it generates a high-frequency interference pulse which can, for example, cause audible interference on audio systems. This HF interference is the cause of the infamous “dimmer buzzing.” Unfortunately, dimmers operating according to the principle of phase-cut have such a disadvantage: We can only try to keep the problem under control by using appropriate interference-suppression filters.

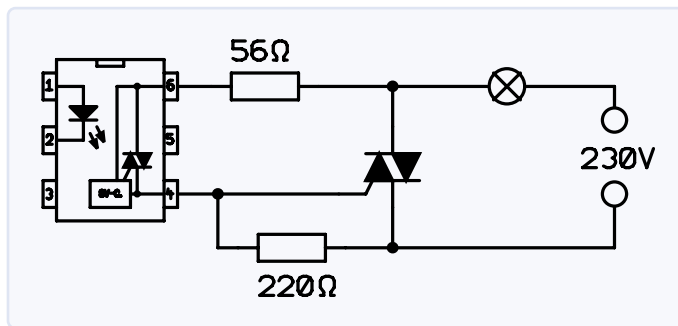


Figure 8: Optocoupler with zero-crossing detector.

But, in all situations where a load only needs to be switched on or off, the switching moment should coincide with the zero crossing of the grid AC voltage: In that case, interference will not occur at all. You can design and build your own sophisticated circuits for this, or you could use an optocoupler that already has zero-crossing detection integrated.

Of course, it's not possible to use this to switch a triac exactly at the zero crossing point, because, at that point, there is no voltage available to supply the required ignition current. However, it is possible to switch as close as possible to the zero-crossing point, resulting in a negligible level of interference.

Here, we conclude our exploration of the wonderful world of diodes. In the next installment, we will introduce the transistor. ◀

220446-01

*Editor's Note: The 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?

Do you have any questions or comments about this article? Contact Elektor at editor@elektor.com.



Related Products

➤ **B. Kainka, Basic Electronics for Beginners (Elektor, 2020) (SKU 19212)**
www.elektor.com/19212

➤ **B. Kainka, Basic Electronics for Beginners (Elektor, 2020) (E-Book, SKU 19213)**
www.elektor.com/19213

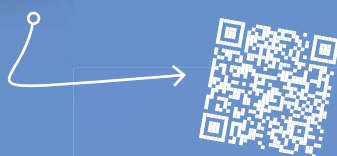
Elektor TV Shows



Elektor Engineering Insights

Elektor Industry Insights is a go-to resource for busy engineers and maker pros who want to stay informed about the world of electronics. During each episode, Stuart Cording (Editor, Elektor) will discuss real engineering challenges and solutions with electronics industry experts.

www.elektormagazine.com/elektor-engineering-insights



Elektor LabTalk

Are you passionate about DIY electronics, embedded programming, or engineering theory? Join Elektor Lab team engineers and editors as they share engineering tips, plan future electronics projects, discuss Elektor Mag and answer community questions.

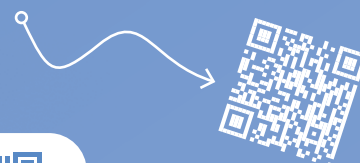
www.elektormagazine.com/elektor-lab-talk



elektor academy

Do you want to enhance your electronics skills? Turn to Elektor Academy for resources to level up your engineering capabilities. Our Expert Stuart Cording will guide you through the Elektor academy courses

www.elektormagazine.com/elektor-academy



Stay informed and join our
YouTube channel Elektor TV
www.youtube.com/c/ElektorIM





Photo: Shutterstock/ Jackrit Singhananta

From Life's Experience

Musings on the Quality of Things

By Ilse Joostens (Belgium)

You often hear older people say, “things were better back then.” They’re referring to the “good old days,” and in some cases also lashing out at young people today. I’ve been in this world for over 50 years, and I know that things weren’t always better in the past. The idea that things used to be better has been with us forever, and mainly comes from the fact that we remember positive experiences better than negative experiences.

Even so, I don’t think my brain is playing tricks on me when I have the feeling that the quality of consumer goods used to be better. Of course, some things sold back then were junk, but the situation now is pretty bad. Technolog-

ical progress does not always lead to improvements and more possibilities. Unfortunately, it also restricts the freedom of consumers, and secretly shortening the lifetime of products makes things even worse.

Royal Lamps and Non-Replaceable Batteries

The invention of bright blue LEDs in 1993 launched a revolution in lighting technology, although it took a while before LED lamps were available on shop shelves. Along with very low power consumption, we were promised an amazing lifetime of several tens of thousands of hours. In practice, it sometimes feels more like minutes than hours – I’ve lost count of how many defective LED lamps I’ve had to replace. Companies simply want to sell products, and products with a long lifespan are not good for the bottom line. Designing products at the edge of the maximum component ratings — or even beyond — has become all too common in the electronics industry. That’s more difficult than you might think, since products should fail as soon as possible after the end of the warranty period but not earlier, so the manufacturer doesn’t have to pay for warranty claims. Unfortunately, consumers keep falling for the same false promises and go out to buy new models, hoping they will be better. That’s how it is with LED lamps. Many of them are designed to force the maximum permissible current through the LEDs, or even a bit more.

The relationship between light output and current is not linear, so the efficiency drops and more power is converted into heat. With this approach, both the LEDs and the other components have a hard life, resulting in premature failure of the lamp. Then you can buy a new one: more income for the supply chain.



Photo: Shutterstock / Jirakorn

It doesn't have to be this way. To make the city of Dubai more energy efficient, Sheikh Mohammad Bin Rashid Al Maktoum made a deal with Philips for the production of millions of LED lamps, known as Dubai lamps [1][2][3]. Driving the lamps at half of their maximum rated power not only considerably extends their lifetime, but also means that these lamps use only half as much current as ordinary LEDs for the same light output. Needless to say, you couldn't buy these lamps in your local shop for a long time. Only recently has Philips released comparable LED lamps here with their Master Ultra Efficiency series.

Years ago I was asked to repair a satellite receiver that was suffering from a form of early dementia and displaying random characters. It turned out that all the electrolytic capacitors on the 12 V power rail and on the main circuit board were rated at 10 V. I have no idea how long the average 10 V electrolytic can withstand operation at 12 V, but this must have been intentional because the warranty had just expired. After I replaced the capacitors by 16 V types, the receiver worked perfectly again. This isn't an isolated case – in a defective battery charger of a top-brand lawnmower, I found a 10 V electrolytic capacitor in the power supply filter circuit, which had to handle peak voltages of up to 14 V.

Everyone knows that lithium-ion batteries don't last forever, so they should be replaceable. That's not possible with my cordless upright vacuum cleaner. Shortly after the end of the warranty period (again!) the operating time started dropping sharply. After a lot of finicky work, cursing and a lacerated finger, I managed to replace the 18650 cells – but the average vacuum cleaner user is not an expert in spot welding and soldering. Our old energy-guzzling cylinder vacuum cleaner, by contrast, is still working well after 24 years. It's the same story with recent smartphones, where battery replacement has increasingly come to resemble open-heart surgery.

The Software Gremlin and the New European Bauhaus

All the above is just 'old school,' but the number of ways to harass consumers has become nearly infinite thanks to increasing digitalization and the use of software. This ranges from ink cartridges with built-in chips for printers to endless software updates that make your devices (such as computers or smartphones) slower or even unusable because they are supposedly outdated. Or even worse: On a given day, your printer happily tells you it has reached the end of its life and immediately stops working [4].

A well-known manufacturer of label printers [5] has even managed to equip the labels with RFID tags so that you have to buy their expensive paper – all supposedly to give you a better user experience. Of course, things like this are always presented as good for the user, consumer, or ordinary person, but, in the end, the user is simply the victim.

In terms of planned obsolescence, the green bigwigs with their New European Bauhaus [6] may be the worst of all. I'm not talking here about the German building style, but I'm worried by the increasing pressure on ordinary people to throw their still perfectly good devices on the scrap heap and replace them by more energy-efficient 'green' versions. Aside from the idiotic costs, I think the production and recycling of household appliances causes considerable CO₂ emissions. And now I'm compelled by law to discard my perfectly functioning gas water heater and replace it by a new one for the sake of a few per cent higher efficiency. At least we know what we'll be working for... ❏

220461-01



WEB LINKS

- [1] LEDs from Dubai: The Royal Lights You Can't Buy: <https://hackaday.com/2021/01/17/leds-from-dubai-the-royal-lights-you-cant-buy/>
- [2] The lamps you're not allowed to have. Exploring the Dubai lamps: <https://youtu.be/klaJqofCsu4>
- [3] Making "Dubai lamps" from 13 W Philips bulbs with a simple hack!: <https://youtu.be/rXJq2vLJhLA>
- [4] Planned Obsolescence Rears Its Ugly Head in Epson Printer Spat: <https://elektor.link/epsonobsolescence>
- [5] EEVblog 1462 - Why Dymo Label Printers SUCK!: <https://youtu.be/xzSDJRC0F6c>
- [6] Report on the New European Bauhaus: https://www.europarl.europa.eu/doceo/document/A-9-2022-0213_EN.html

Reverse-Engineering a Bluetooth Low Energy LED Badge

How to Control a BLE Device with a Python Script

By Koen Vervloesem (Belgium)

Many Bluetooth Low Energy (BLE) devices come with their own mobile app offering user control and typically implementing a custom protocol that's understood by the device and the corresponding app only. Typically, again, there's no specification that you can read up to create your own implementation. Now comes the good news: reverse-engineering the BLE device is the way to crack it open and use it with your very own software. Here's how.

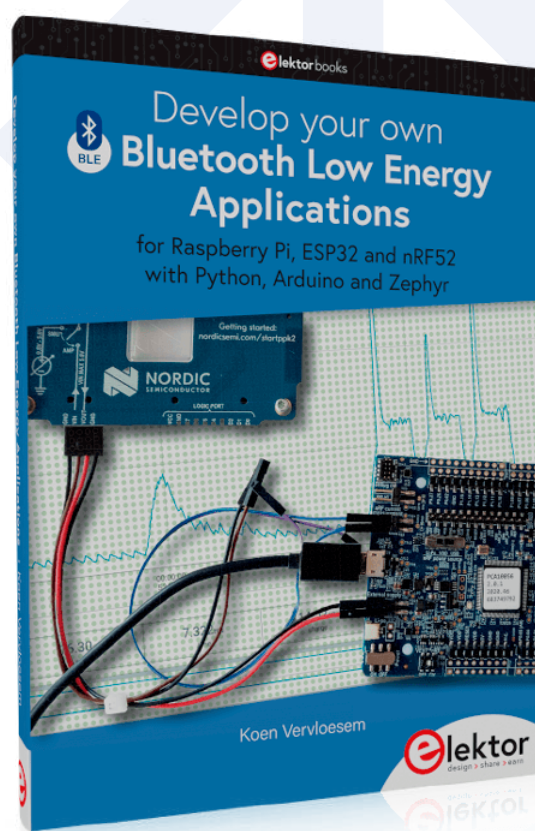
Editor's Note. *Although this article is based on a chapter from the 257-page book Develop your own Bluetooth Low Energy Applications (Elektor, 2022), the contents were adapted by the author especially for Elektor Magazine to form a closed, fully replicable project.*

In this article, I'm going to reverse-engineer a BLE LED badge as an example of this challenging and highly educational activity. Join me and discover how the device works by investigating its BLE services and characteristics. Watch me decompile the accompanying mobile app and sniff BLE traffic between the app and the device. My goal is to create a Python script to control the LED badge so that I no longer need the official mobile app.

Investigating the LED Badge

The LED badge from AliExpress as pictured in **Figure 1** has an 11 × 55 LED-pixel display that's available in various colors. It supports Bluetooth, with no version specified.

Scanning the QR code on the back of the badge results in an "HTTP 404" error. On Google Play, I found an app called "Bluetooth LED Name



Badge," made by Shenzhen Lesun Electronics Co., Ltd (**Figure 2**). This app can send text and icons to the badge and has some settings, including scroll speed.

Let's turn on the LED badge and use the BLE-scanning "nRF Connect for Mobile" app [1] to see what the device is "advertising," as it's called. Press the lower button twice. The display then shows a BLE icon. In nRF Connect, you'll notice that it's now advertising manufacturer-specific data and two custom BLE services: 0xfee7 and 0xfee0 (**Figure 3**). The device goes by the name: LSLED. Connecting to it reveals its services along with the characteristics listed in **Table 1**.

Table 1. BLE Badge Characteristics

Service	Characteristic	Property
0xfee7	0xfec7	WRITE
0xfee7	0xfec8	INDICATE
0xfee7	0xfec9	READ
0xfee0	0xfee1	NOTIFY, READ, WRITE

Reading the characteristic "0xfec9" in nRF Connect for Mobile returns the same value as the manufacturer-specific data. Reading 0xfee1 doesn't return any data, and subscribing to its notifications doesn't return anything either. The Characteristic User Description of 0xfee1 is "Data." This method yields nothing useful.

Decompiling the Mobile App

Since the Bluetooth LED Name Badge app for Android is obviously able to communicate with the LED badge, let's try and find out what the app actually does. We'll do so by decompiling the app and fathoming the workings of its source code.

Download the APK file for the Android app using a third-party APK downloader site like [2]. Just paste the app's Google Play Store URL [3] into APKPure's search field. Next, you're able to download the file. This APK file contains the Dalvik byte code for Android to execute. To understand what the app does, we need its source code. While the developer has compiled its source code to Dalvik, we can reverse this using a decompiler like jadx [4], which turns Dalvik byte code into Java source code.

With the most recent release for Windows [5], we can start the graphical interface by double-clicking the **jadx-gui** file in the **bin** directory. On other operating systems (Linux; macOS), start the graphical interface by running **bin/jadx-gui** from the command line.

Open the downloaded APK file. The program now decompiles the app and shows a tree structure of its packages and Java files at the left. Now the search begins. We already have a couple of leads: the UUIDs of the services and characteristics found in nRF Connect for Mobile. In the **Navigation / Text search** menu, you can enter some search terms.

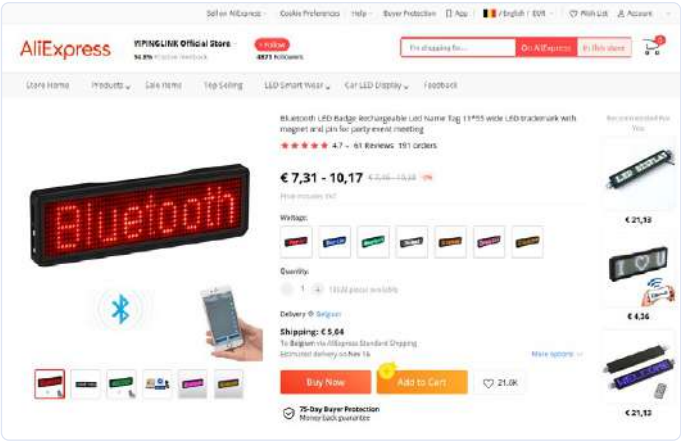


Figure 1: This Bluetooth LED badge seems like an interesting device to reverse-engineer!

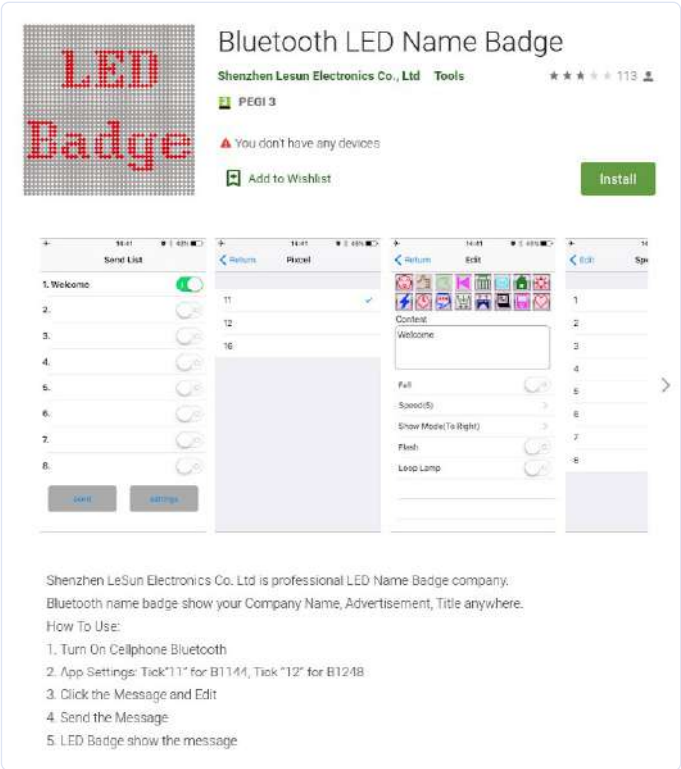


Figure 2: The "Bluetooth LED Name Badge app" in the Google Play Store can send commands to the LED badge.

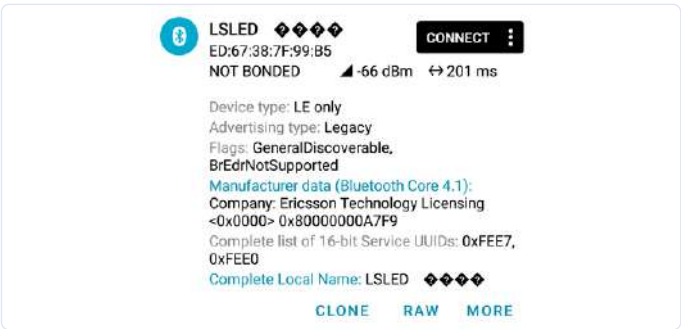


Figure 3: The Bluetooth LED badge as seen in nRF Connect for Mobile.



Listing 1.

```
public static final String UUID_CHARACTERISTICS_WRITE = "fee1";
public static final String UUID_SERVICE = "fee0";
```



Listing 2.

```
public static byte[] get64(List<SendContent> list, int i) {
    Iterator<SendContent> it = list.iterator();
    while (it.hasNext()) {
        Log.d("abcdef", "get64-----SendContent:" + it.next().toString());
    }
    byte[] bArr = new byte[64];
    bArr[0] = 119;
    bArr[1] = 97;
    bArr[2] = 110;
    bArr[3] = 103;
    bArr[4] = 0;
    bArr[5] = 0;
    bArr[6] = getFlash(list);
    bArr[7] = getMarquee(list);
    byte[] modeAndSpeed = getModeAndSpeed(list);
    for (int i2 = 0; i2 < 8; i2++) {
        bArr[i2 + 8] = modeAndSpeed[i2];
    }
    byte[] msgLength = getMsgLength(list, i);
    for (int i3 = 0; i3 < 16; i3++) {
        bArr[i3 + 16] = msgLength[i3];
    }
    bArr[32] = 0;
    bArr[33] = 0;
    bArr[34] = 0;
    bArr[35] = 0;
    bArr[36] = 0;
    bArr[37] = 0;
    byte[] date = getDate();
    for (int i4 = 0; i4 < 6; i4++) {
        bArr[i4 + 38] = date[i4];
    }
    for (int i5 = 0; i5 < 19; i5++) {
        bArr[i5 + 44] = 0;
    }
    bArr[63] = 0;
    return bArr;
}
```

This is easier said than done. The code doesn't mention the 0xfeef service or its characteristics! It does mention the other service and its characteristic in the class `com.yannis.ledcard.ble.BleDevice`.

Listing 1 shows the relevant parts of this class..

In the code, you can now search for these two constants:

```
UUID_CHARACTERISTICS_WRITE
UUID_SERVICE
```

Probably the easiest way to do this is to right-click on the name of the constant and then select **Find usage**. Then click on one of the instances found to open the corresponding file at that location.

While searching for some other clues in the code, I came across some promising code that deals with pictures and display modes in this class:

`com.yannis.ledcard.util.LedDataUtil`. Specifically, I found this Java method given in **Listing 2** that seems to create some kind of header for data.

This shows some fixed header (6 bytes), a mode and speed, a message length, and a date. It is called by a `getSendHeader()` in the class mentioned.

To reverse-engineer what the app is doing just by delving further into this code wasn't enough. So I started using the app with the LED badge while sniffing the traffic between both devices. For a better understanding, I was now able to combine the knowledge gained from the app's source code with live traffic.

Sniffing BLE Traffic between Badge and App

Wireshark [6] sniffs your phone's live Bluetooth traffic using the Android Debug Bridge. Connect your phone to your computer with a USB cable and allow the debug connection on your phone. Start Wireshark. This should show **Android Bluetooth Btsnoop Net** as one of the available interfaces. After double-clicking on the interface, you'll see live Bluetooth traffic scrolling by! An even better option is to use the nRF Sniffer for Bluetooth LE plugin for Wireshark [7], using an nRF52840 Dongle as the hardware part of the sniffer.

On the BLE badge, press the lower button twice until it shows the Bluetooth icon. Then, select the device in Wireshark to show only packets to and from that specific device.

Install "Bluetooth LED Name Badge" on your Android phone and open the app. In Wireshark, you see that your phone does a scan request and the BLE badge answers with its device name inside the scan response. Select the device type in the app. For the 11 x 5 pixel LED badge, whose type is 11. Tap **yes**.

The app shows a list of messages to send. The default configuration is to send one message: **Welcome**. Tap **Send**. The app then connects to the LED badge and shows the **Welcome** message on its display (**Figure 4**).



Figure 4: The Bluetooth LED badge shows a welcome message.

In Wireshark, we see a `CONNECT_REQ` packet followed by a request for attributes. Then come a couple of Write Requests and responses. Get a better overview by right-clicking on the Write Request opcode in the Bluetooth Attribute Protocol part of the packet details and then choosing **Apply as Filter / Selected**. Alternatively, enter: `btatt.opcode == 0x12` as the display filter.

In this case, the app sent nine Write Request packets, each having 16 bytes of data as shown below:

```
77 61 6e 67 00 00 00 00 30 30 30 30 30 30 30 30
00 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 e5 0a 18 0c 37 25 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 c6 c6 c6 c6 d6 fe ee c6 82 00 00 00 00 00 7c
c6 fe c0 c6 7c 00 00 38 18 18 18 18 18 18 18 3c
00 00 00 00 00 7c c6 c0 c0 c6 7c 00 00 00 00 00
7c c6 c6 c6 c6 7c 00 00 00 00 00 ec fe d6 d6 d6
c6 00 00 00 00 00 7c c6 fe c0 c6 7c 00 00 00 00
```

If you review the `get64()` Java method from the previous section, you'll recognize the parts of the header: 77 61 6e 67 (hexadecimal equivalent of decimal 119 97 110 103). Then we get 00 twice, and then again 00 twice, which are the values for `getFlash()` and `getMarquee()`, respectively. The next eight bytes are all 0x30, which represent the mode and speed. The next 16 bytes represent a message length. Because the app shows a list of eight messages, this could be the list of the lengths of these messages. The first two bytes here are 00 07, which indeed is the number of characters in the text **Welcome**. The hypothesis is that the next 14 bytes represent the lengths of the next seven messages (none, in this case).

Then, there are six 00's and the next six bytes represent the date, which is e5 0a 18 0c 37 25, in this case. Converted to decimal, that's 229 10 24 12 55 37. I ran this app on October 24, 2021, at 12:55:37. The month, day and time were correct. The header concludes with 20 00's

After this, five packets of 16 bytes somehow represent the seven characters of the **Welcome** text. Let's find out how it's done, starting with a simpler message. Open the app again and click on the first message. Change the text to **W** and enable **Flash**. Return to the main screen, click on the second message, and add the text: **e**. For this second message, set the speed to 8, the mode to **right**, and enable **Marquee**. Return to the main screen. Then enable the slider next to the second message and click on **Send** while looking at the Bluetooth Attribute Protocol packets in Wireshark.

The LED badge now shows the letter W moving to the left and flashing on and off. After this, it shows the letter e moving to the right at double speed and a frame of moving pixels around it. In this case, the app sent six Write Request packets, each with 16 bytes of data — see below:

```
77 61 6e 67 00 00 01 02 30 71 30 30 30 30 30 30
00 01 00 01 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 e5 0a 18 10 06 13 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 c6 c6 c6 c6 d6 fe ee c6 82 00 00 00 00 00 7c
c6 fe c0 c6 7c 00 00 00 00 00 00 00 00 00 00 00
```

The first packet starts with the same 6 bytes, but then the value of `getFlash()` is 01 and that of `getMarquee()` is 02. This is because we enabled **Flash** for the first message and **Marquee** for the second message.

The next byte is still 0x30, because we didn't change anything to the mode and speed of the first message. But the byte after this now reads 0x71. We changed the speed to 8 and the mode to Right. So, the left nibble of that byte apparently encodes the speed (speed 4 is encoded as 3 and speed 8 as 7) and the right nibble the mode. The second packet shows 00 01 as the length of the first message and the same for the length of the second message. This confirms our hypothesis. Then the date is encoded.

There are two packets left now. Convert the hexadecimal values to binary and format them byte by byte and in sequence, then split them into eleven rows (yes, because the display is 11 pixels tall) like so:

```
00000000
11000110
11000110
11000110
11000110
11010110
11111110
11101110
11000110
10000010
00000000

00000000
00000000
00000000
00000000
01111100
11000110
11111110
11000000
11000110
01111100
00000000
```



Listing 3.

```

"""Find BLE LED badges.

Copyright (c) 2022 Koen Vervloessem
SPDX-License-Identifier: MIT

"""

import asyncio
from bleak import BleakScanner

num_devices = 0

def device_found(device, advertisement_data):
    """Show device details if it's a BLE LED badge."""
    global num_devices
    if device.name.startswith("LSLED"):
        num_devices += 1
        print(
            f"() - RSSI: "
        )

async def main():
    """Scan for BLE devices."""
    print("Searching for LED badges...")
    scanner = BleakScanner()
    scanner.register_detection_callback(device_found)
    await scanner.start()
    await asyncio.sleep(5.0)
    await scanner.stop()
    if not num_devices:
        print("No devices found")

if __name__ == "__main__":
    asyncio.run(main())

```

Towards the end, filler 0's are used to pad out a block of 16 bytes (not shown here). If you apply the same decoding to the first message with **Welcome** and then put the bitmaps of the letters next to each other, you'll get a 1 for each enabled pixel and a 0 for each disabled one as illustrated here:

[illegible]

You now have a pretty good idea of the format of the data to send to the LED badge in order to show some text on the display. Let's proceed by putting all of this together and creating a Python script to send your own images to the LED badge.

Writing Random Images to the LED Badge using Bleak

The Bluetooth LED Name Badge also allows you to send some pre-defined little 11 × 11 bitmaps. But now that you know the data format to send, you can also send an arbitrary bitmap, for example, to flood the entire 11 × 55 display. Let's do this with the help of Bleak [8], a cross-platform Python library for BLE. Type:

```
pip3 install bleak
```

Let's first create a script that scans for BLE badges based on their names for five seconds, and then shows their Bluetooth addresses and RSSIs — see **Listing 3**.

If you run this with two LED badges that are both in Listen mode, the script shows:

```
$ python3 find_led_badge.py
```

Searching for LED badges...

ED:67:38:80:0D:E2 (LSLED) - RSSI: -74

ED:67:38:7F:99:B5 (LSLED) - RSSI: -83

Let's now build on the knowledge gained in the previous section. Check out this script that writes the appropriate commands to the LED badge to show an image — see **Listing 4**.

Apart from Bleak, this script also uses the Pillow library [9]:

```
pip3 install Pillow
```

The first four BLE commands are hard coded. In the first command, 0x34 defines the mode and speed for the first message. The 4 means static mode: the message is shown as a still image. In the second command, 0x07 defines the length of the message as the number of 8-pixel characters. The display is 55 pixels wide, and 7 times 8 equals 56, so we need to encode the image as 7 characters. The next two commands should encode the current date, but this isn't really needed. Just send 16 0s for each of these commands.

`chunks()` is a helper function to split a byte array into a list of byte arrays of 16 bytes each. This is used in the `commands_for_image()` function that converts an image file into bytes that represent characters on the display. At the end of the function, these bytes are split into 16-byte chunks.

`commands_for_image()` opens an image with Pillow and loads its pixels. For each of the seven characters on the display, it encodes the character as 11 bytes: one for each row. All of these bytes are appended to `image_bytes`.

At the end of the image, the byte array is padded with additional bytes so that it's a multiple of 16 bytes. And, ultimately, it's split into 16-byte chunks.

The `main()` function puts this all together. It connects to the device and creates a list of commands: the four commands of the header are extended with the commands for the image. Then, each of these commands is written to the characteristic with UUID 0xfee1. The main code at the bottom checks whether you have supplied two arguments



Listing 4.

```
"""Display an image on a BLE LED badge.
Copyright (c) 2022 Koen Vervloesem
SPDX-License-Identifier: MIT
"""

import asyncio
import sys
from PIL import Image
import bleak

WRITE_CHAR_UUID = "0000fee1-0000-1000-8000-00805f9b34fb"
COMMAND1 = bytes([0x77, 0x61, 0x6E, 0x67, 0x00, 0x00, 0x00, 0x00, 0x34, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30])
COMMAND2 = bytes([0x00, 0x07, *(14 * [0x00])])
COMMAND3 = bytes(16 * [0x00])
COMMAND4 = bytes(16 * [0x00])

def chunks(lst, n):
    """Yield successive n-sized chunks from lst."""
    for i in range(0, len(lst), n):
        yield lst[i : i + n]

def commands_for_image(image):
    """Return commands to show an image on the BLE LED badge."""
    image_bytes = bytearray()
    with Image.open(image) as im:
        px = im.load()
        for i in range(7): # 7x8 = 56 -> 7 bytes next to each other
            for row in range(11):
                row_byte = 0
                for column in range(8):
                    try:
                        pixel = int(
                            px[(i * 8) + column, row][3] / 255
                        )
                    except IndexError:
                        pass # Ignore the 56th pixel in a full row
                row_byte = row_byte | (pixel << (7 - column))
            image_bytes.append(row_byte)
    # Fill the end with zeroes to have a multiple of 16 bytes
    image_bytes.extend(bytes(16 - len(image_bytes) % 16))
    # Split image bytes into 16-byte chunks
    return list(chunks(image_bytes, 16))

async def main(address, filename):
    """Connect to BLE LED badge and send commands to show an image."""
    try:
        async with bleak.BleakClient(address) as client:
            commands = [COMMAND1, COMMAND2, COMMAND3, COMMAND4]
            commands.extend(commands_for_image(filename))
            for command in commands:
                await client.write_gatt_char(WRITE_CHAR_UUID, command)
    except asyncio.exceptions.TimeoutError:
        print(f"Can't connect to device .")
    except bleak.exc.BleakError as e:
        print(f"Can't write to device : ")

if __name__ == "__main__":
    if len(sys.argv) == 3:
        address = sys.argv[1]
        filename = sys.argv[2]
        asyncio.run(main(address, filename))
    else:
        print(
            "Please specify the BLE MAC address and image filename."
        )
```

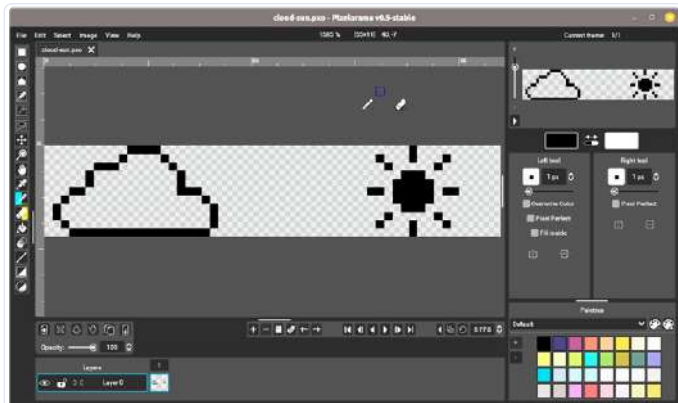



Figure 5: Pixelorama is an open-source 2D image editor.



Figure 6: By sending the appropriate BLE commands, you can show off your very own 11 × 55 pixel images on the LED badge.

on the command line. The first one is assigned to the Bluetooth address, and the second, to the filename.

Before running this code, you should prepare an image of exactly 11 × 55 pixels, using a pixel editor like Pixelorama [10]. Fill a pixel for each LED that you want to light up on the display. **Figure 5** shows an example of a cloud & sun image I created in Pixelorama.

Export the picture as a .PNG file. Now, put the LED badge in Discovery mode by pressing the button at the bottom twice until the Bluetooth icon appears. Run the Python script with two arguments; first the Bluetooth address of the badge and then the picture's filename:

```
$ python3 display_led_badge.py ED:67:38:7F:99:B5 cloud-sun.png
```

Conclusion

The LED badge should now display "your" image (**Figure 6**). So there you have it: you've successfully reverse-engineered the BLE badge and are now in a position to use it within our own code. Surely my proposed script is subject to improvement, especially to make it more generally accessible. I hope this inspires you to apply the same sort of reverse engineering to other BLE devices. 

220439-01

Questions or Comments?

Do you have any technical questions or comments related to this article? Email the author at koen@vervloesem.eu or Elektor at editor@elektor.com.



RELATED PRODUCTS



> **K. Vervloesem, Develop your own Bluetooth Low Energy Applications, Elektor 2022 (Book; SKU 20200)**
www.elektor.com/20200

This book comes with a FREE nRF52840 USB Dongle!

> **K. Vervloesem, Develop your own Bluetooth Low Energy Applications, Elektor 2022 (E-Book; SKU 20201)**
www.elektor.com/20201

WEB LINKS

- [1] nRF Connect for Mobile App: www.nordicsemi.com/Products/Development-tools/nrf-connect-for-mobile
- [2] The APK file : <https://apkpure.com>
- [3] The app in Google Play Store : <https://play.google.com/store/apps/details?id=com.yannis.ledcard>
- [4] The jadx decompiler : <https://github.com/skylot/jadx>
- [5] Latest release: <https://github.com/skylot/jadx/releases>
- [6] Wireshark: www.wireshark.org
- [7] nRF Sniffer for Bluetooth LE : www.nordicsemi.com/Products/Development-tools/nrf-sniffer-for-bluetooth-le
- [8] Bleak : <https://bleak.readthedocs.io>
- [9] The Pillow library : <https://pillow.readthedocs.io>
- [10] Pixelorama: <https://orama-interactive.itch.io/pixelorama>
- [11] Book resources/info page: www.elektor.com/develop-your-own-bluetooth-low-energy-applications
- [12] Code on GitHub <https://github.com/koenvervloesem/bluetooth-low-energy-applications>

MakePython ESP32 Development Kit

Everything in a Box

By Tam Hanna (Slovakia)

Modern microcontrollers, such as an ESP32, are so powerful that they can be programmed in MicroPython – thanks to the powerful libraries, this allows you to quickly achieve a finished project. With the MakePython ESP32 Development Kit – which is a textbook on the one hand and a hardware kit on the other hand – well-known Elektor author Dogan Ibrahim now presents a starter package that illustrates how to work with MicroPython by means of real-world examples.



We don't really need to discuss the fact that MicroPython is not the way to maximum software efficiency. On the other hand, it is true that modern microcontrollers like an ESP32 are more than able to keep up with a 486 in terms of performance – especially in small series it can therefore be reasonable to trade off programming effort for speed by using high-level languages.

The Book

Let's start with the textbook, which is available in English: It comes with a quite fancy design, which pretends a (fake) spiral binding.

The actual didactic structure then begins with a brief explanation of the ESP32 as a whole. As an Elektor reader who already has extensive or at least basic experience with microcontrollers, it will provide you with a “quick” overview of the various peripherals provided by Espressif in the controller. If you have no experience

at all with microcontrollers, the explanations — at least in places — might be too short for solid understanding.

The installation of the IDEs — Ibrahim introduces both uPyCraft and Thonny, but later works almost exclusively with Thonny — is explained in detail for Windows, whereas Linux is hardly covered. This is followed by a chapter that illustrates the execution of some “small” Python snippets using Thonny — if you don't have any knowledge of Python syntax, you can't really get into it at this point. On the other hand, the explanations are more than sufficient to understand the “basics” of working with MicroPython and the IDE.

The actual presentation of the 46 projects contained in the book is then done in the classic Dogan Ibrahim style: In the first step, the professor always presents the “battle task to be completed”, and then presents code and explanations of the hardware construction.



Figure 1: The kit. Plastic packaging reliably protects the components from damage.

Fans of compact coding in particular might be somewhat annoyed by the fact that the listings always include a standardized header with a good ten lines. On the other hand, to Ibrahim's credit, the examples — while not complicated — do aptly illustrate the essential aspects of working with MicroPython.

After reading this, you will certainly no longer have to fear the desperate search for how to put certain peripherals into operation. Very commendable, in the reviewer's opinion, is the fact that Ibrahim explains how to equip a "messed up" board with new MicroPython firmware.

A Look at the Board

Speaking of the evaluation board, the textbook is available in combination with the kit shown in **Figure 1**, which is also suitable to take along on vacation or to prevent boredom on a business trip due to its quite robust plastic packaging. But let's take a look at the actual hardware now. The author's practical teaching experience has shown that especially developers not familiar with embedded systems will have a more comfortable learning experience if the development platform provides an (ideally fully graphical) display.

Elektor cleverly gets around this problem by providing the red PCB shown in **Figure 2**. The black area is not a black hole, but one of the widely used SSD1306 OLEDs with a resolution of 128 by 64 pixels.

The reviewer's biggest point of criticism is hidden here: The 2.54-mm headers are not soldered in. This is a pity because a person who buys the kit directly in a store and takes it with them on vacation is screwed at this point — at least if you don't have a soldering iron at hand for its installation and the cell phone store next door doesn't help.

Otherwise, the board offers no reason for criticism. The ESP32 variant used is one of the larger models and has enough memory, the MicroUSB interface ensures that "command device cables" can be borrowed quickly from any older cell phone. In addition, the kit also includes a (very short) cable of decent quality.

A Demo

While the textbook is primarily oriented towards the needs of developers working with Windows, the reviewer will refer to Ubuntu 20.04 LTS in the following steps — if only for the sake of convenience. The IDE available at [1] is a rather bad choice, because its

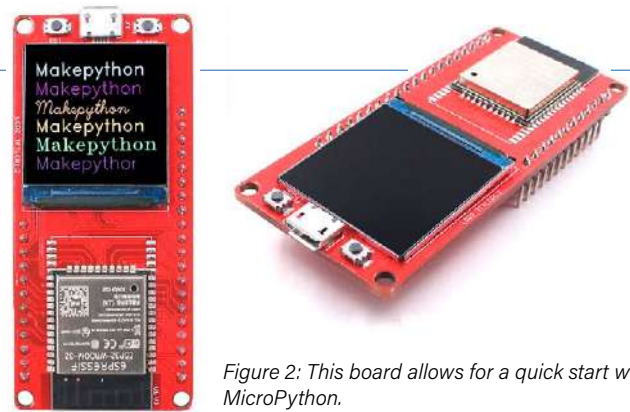


Figure 2: This board allows for a quick start with MicroPython.

execution in all Ubuntu versions later than 16.04 fails with an error message like:

```
ImportError: /tmp/_MEI0hQKhZ/libz.so.1: version 'ZLIB_1.2.9' not found (required by /usr/lib/x86_64-linux-gnu/libpng16.so.16)
```

A "nicer" option is Thonny, which can be installed automatically by entering

```
bash <(wget -O - https://thonny.org/installer-for-linux)
```

and can then be started by entering the command

```
/home/tamhan/apps/thonny/bin/thonny
```

To uninstall, we use the command `/home/tamhan/apps/thonny/bin/uninstall`. All that remains to be done after the first start is confirming the language selection.

The next step is to connect the evaluation board to a workstation. Fortunately, Elektor thoughtfully delivers the board with a preconfigured MicroPython runtime. In this context, it is especially convenient that the board also switches on the display at the same time, which facilitates a function check. By the way, a Silicon Labs CP210x UART bridge chip is used as converter.

At this point, you can switch to Thonny (**Figure 3**). Make sure that you click on the Python version in the selection menu that appears at the bottom right and select the *MicroPython (ESP32)* version. Thonny will then automatically search for ESP32 boards — in this case, the board used here was automatically detected.

The question of whether you need to run Thonny with superuser privileges is debatable. Since the reviewer's user account is a member of the *plugdev* group, he was able to run Thonny in his normal user account and still interact with the ESP32.

A First Welcome

As the next action, let's move on to putting the device's OLED display into operation. To do this, we open the URL: www.makerfabs.com/makepython-esp32-starter-kit.html. Download the archive *MakePython ESP32 Lessons Source Code*. Using RAR is a bit annoying in Linux. In any case, extract the archive to a conveniently accessible location.

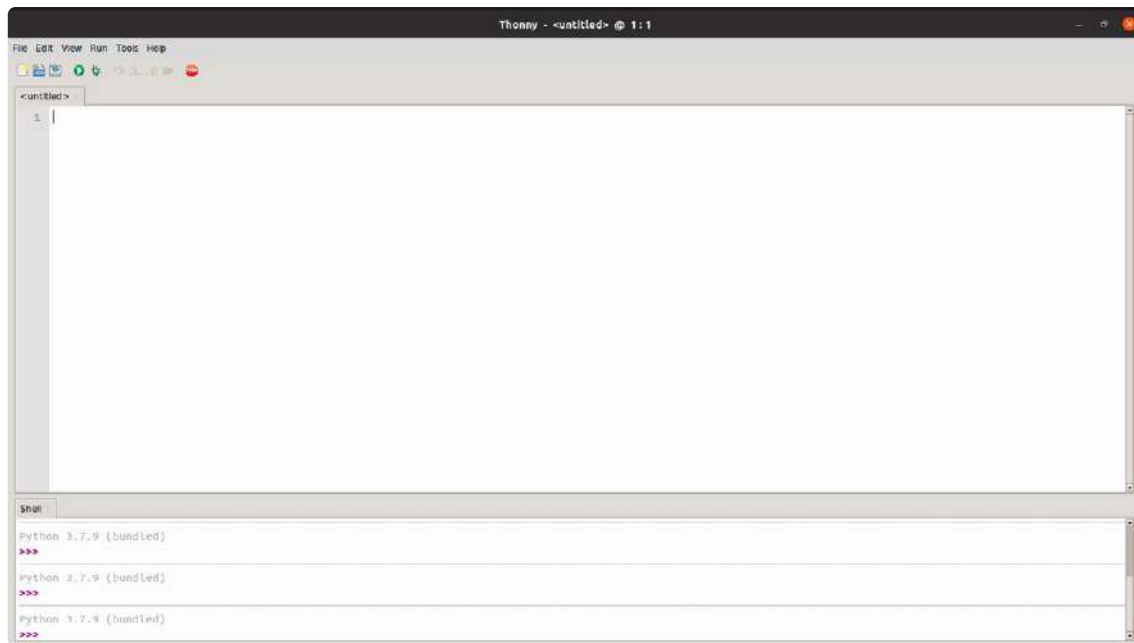


Figure 3: Screenshot of Thonny

For the following steps, we especially need the file `ssd1306.py`, which provides the OLED driver code.

In the first step, click *File* → *Open*, and then choose the *This Computer* option. In the next step, navigate to the file to load it into the editor. Then we choose *File* → *Save as...* and finally the option *MicroPython Board*.

If Thonny gets upset at this point about the backend being “busy”, you can click on the red stop icon in the toolbar to command a halt. In the next step, you save the file into the file system under the known name — at least if you did not leave the original firmware unchanged. In this case, the file is already in place.

As is common with MicroPython, we first need to install some libraries and also define constants with additional information about the specifications of the display:

```
import machine
import ssd1306
import time
WIDTH = const(128)
HEIGHT = const(64)
```

For the actual communication with the display connected via I2C — which is uncommon, as you would usually expect SPI here — we then use an instance of the software I2C class:

```
pscl = machine.Pin(5, machine.Pin.OUT)
psda = machine.Pin(4, machine.Pin.OUT)
i2c = machine.I2C(scl=pscl, sda=psda)
oled = ssd1306.SSD1306_I2C(WIDTH, HEIGHT, i2c)
```

Finally, we need to send text to the display according to the following scheme:

```
while True:
    oled.fill(0)
```

```
oled.text('Hello World!',10,0)
oled.show()
time.sleep(1)
```

At this point, you can click on the play icon in the IDE to enjoy the text appearing on the display of the evaluation board.

MQTT Setup

As a next task, we want to go a little beyond the “ordinary” project scope included in the kit and realize an advanced task — this also shows that the kit is suitable for higher tasks. Specifically, we want to use the universally known MQTT protocol. The task becomes especially interesting by the fact that we want to work with a brand-new version of the MQTT broker Mosquitto — at the URL [2], you can find a (recommendable) description of the issues that the parsers — which are configured more and more strictly for security reasons — can run into in interaction with the MicroPython drivers.

Speaking of MicroPython drivers, meanwhile there are two competing implementations of MQTT for MicroPython. However, here we want to rely on the simpler variant available at [3]. Download the file:

<https://github.com/micropython/micropython-lib/blob/master/micropython/umqtt.simple/umqtt/simple.py>

And save it under the name `simple.py` on the process computer.

In the next step, we need to set the Wi-Fi transmitters of the board into station mode according to the following scheme:

```
...
wlan=network.WLAN(network.STA_IF)
```

For the actual connection setup, we then use a method built according to the following scheme — it is largely taken from the project skeleton provided by MakerFabs and implements a countdown timer including connection establishment. Of course, make sure to



Listing 1.

```
def connectWiFi():
    i=0
    wlan.active(True)
    wlan.disconnect()
    wlan.connect("ssid", "pass")
    while(wlan.ifconfig()[0]!='0.0.0.0'):
        i=i+1
        oled.fill(0)
        oled.text('connecting WiFi',0,16)
        oled.text('Countdown: '+str(20-i)+'s',0,48)
        oled.show()
        time.sleep(1)
        if(i>20):
            break
    oled.fill(0)
    oled.text('Makerfabs',25,0)
    oled.text('MakePython ESP32',0,32)
    if(i<20):
        oled.text('WIFI connected',0,16)
    else:
        oled.text('NOT connected!',0,16)
    oled.show()
    time.sleep(3)
    return True
```

adapt the strings passed to `wlan.connect` to your wireless network situation (see **Listing 1**).

Establishing the actual test connection then couldn't be easier:

```
connectWiFi()
while True:
    time.sleep(1)
```

In the next step, we want to provide ourselves with a Mosquitto server. While Mosquitto is not the fastest MQTT broker, it is open source, free and very strict in its MQTT implementation.

Due to its widespread use, it is also ensured that there is a more or less turnkey image in the Docker Hub that allows the deployment of the server without deep configurations of the host computer.

For a first attempt, it is sufficient to enter the following command:

```
docker run -it -p 1883:1883 -p 9001:9001 -v mosquitto.
conf:/mosquitto/config/mosquitto.conf eclipse-mosquitto
```

Firstly, the parameters `-p 1883:1883` and `-p 9001:9001` are important here. After all, each Docker container is provided with a full complement of TCP/IP ports by the runtime. Each of the parameters then connects one of these ports to the host computer's network card. Via the parameter `-v mosquitto.conf:/mosquitto/config/mosquitto.conf`, we also integrate a configuration file.

At this point, it is a good idea to make an initial download attempt with an existing Internet connection. Docker will normally connect



to the hub and download the required components, and then exit with the following error message:

```
docker: Error response from daemon: source /var/lib/
docker/aufs/mnt/a22b9e557c37d99eb71f17e7bc6d38df6e7677
d09225376d416612adf0977ccd/mosquitto/config/mosquitto.
conf is not directory.
```

The cause of the error is that there is not yet a file named `mosquitto.conf` in the host workstation that we could make available to the container.

To solve the problem, it is sufficient to create a new file via the command line in the working directory with `gedit`:

```
(base) tamhan@tamhan-thinkpad:~$ gedit mosquitto.conf
```

The switch from Mosquitto version 1 to version 2 was accompanied by massive tightening in the area of security configuration. While a Mosquitto 1.X (started with default parameters) accepted server requests from any clients, Mosquitto 2 refuses this. However, if you place the following lines in the configuration file, everything will work as usual (and insecure) again:

```
listener 1883
allow_anonymous true
```

The actual execution can then be commanded according to the following scheme. Note that the `-v` parameter always requires a fully qualified path, which is why we call the `pwd` command and "assemble" the results by shell magic. `pwd`, by the way, stands for Print Working Directory, **Figure 4** shows the behavior:

```
(base) tamhan@tamhan-thinkpad:~$ docker run -it -p
1883:1883 -p 9001:9001 -v $(pwd)/mosquitto.conf:/
mosquitto/config/mosquitto.conf eclipse-mosquitto
. . .
```

If the Docker container acknowledges its successful start by returning `1658673183: mosquitto version 2.0.14 running`, we are all set at this point.

```
tamhan@tamhan-thinkpad: ~/Desktop/Stuff
Support: command not found
(base) tamhan@tamhan-thinkpad:~$ pwd
/home/tamhan
(base) tamhan@tamhan-thinkpad:~$ cd Desktop/Stuff/
(base) tamhan@tamhan-thinkpad:~/Desktop/Stuff$ pwd
/home/tamhan/Desktop/Stuff
(base) tamhan@tamhan-thinkpad:~/Desktop/Stuff$
```

Figure 4: The `pwd` command returns the current working directory of the shell.



Initializing the MQTT Software on the ESP32

First of all, we need some constants for working with MQTT. Make sure to adapt the IP address passed in the server string to your local operating situation:

```
SERVER = "192.168.178.146"
CLIENT_ID = ubinascii.hexlify(machine.unique_id())
TOPIC = b"led"
state = 0
```

We also need a callback function that the MQTT driver will always call when a message is received:

```
def sub_cb(topic, msg):
    global state
    print((topic, msg))
    . . .
```

The actual setup of the MQTT connection is then comparatively simple:

```
connectWiFi()
c = MQTTClient(CLIENT_ID, SERVER, keepalive=30)
c.set_callback(sub_cb)
c.connect()
c.subscribe(TOPIC)
print("Connected to %s, subscribed to %s topic" % (SERVER,
TOPIC))

while True:
    c.wait_msg()
```

The call of `c.subscribe` ensures that the MQTT driver informs the server which message channels are currently of interest. It is also important to periodically call the method `c.wait_msg()` to provide the MQTT server with CPU time for processing the information that is incoming or to be sent. In the command line or in the output window, you can then see the output `Connected to 192.168.178.146, subscribed to b'led' topic`.

In the next step, we want to connect one of the included LEDs, along with the included resistor, to the GPIO pin 12. The author assumes that the reader has sufficient knowledge of electronics. The initialization of the GPIO port then only requires ordinary ESP32 code:

```
led = Pin(12, Pin.OUT, value=1)
```

To process the incoming messages, we then need to extend the callback by combining the string supplied in the value `topic` with the constants provided for the various commands:

```
def sub_cb(topic, msg):
    global state
    print((topic, msg))
    if msg == b"on":
        led.value(1)
        state = 1
    elif msg == b"off":
        led.value(0)
        state = 0
    elif msg == b"toggle":
        led.value(state)
```

For the actual testing, we want to use the command line utility `mosquitto_pub`: This is a tool available in Linux that allows commands to be sent directly to an MQTT server. The following commands are then used to switch the LED on and off:

```
(base) tamhan@tamhan-thinkpad:~$ mosquitto_pub -t led
-m 'on'
(base) tamhan@tamhan-thinkpad:~$ mosquitto_pub -t led
-m 'off'
```

If the ESP 32 and the server are in the same network, you can now switch the LED on and off.

Conclusion and Outlook

The MakePython ESP32 Development Kit is a fascinatingly compact toolbox that allows Python developers and/or electronics engineers to quickly unite the two worlds. Especially because everything is so nicely packaged in a turnkey manner, this is a box the reviewer is happy to recommend. ◀

220429-01



Related Products

> **MakePython ESP32 Development Kit (SKU 20137)**
www.elektor.com/20137

WEB LINKS

- [1] uPyCraft IDE: <https://github.com/DFRobot/uPyCraft>
- [2] umqtt.simple and mosquitto 2.0.12: <https://github.com/micropython/micropython-lib/issues/445>
- [3] umqtt.simple - simple MQTT client for MicroPython:
<https://github.com/micropython/micropython-lib/tree/master/micropython/umqtt.simple>

THD Measurement with an Oscilloscope and FFT

Easily Calculate the Distortion Factor

By Sebastian Westerhold (AI5GW) (Germany)

Total Harmonic Distortion, or THD for short, is an important measure of the harmonic content of signals in power supply networks and electronic assemblies, such as audio amplifiers. Usually, special measurement devices are necessary for determining the THD. However, if you have a digital oscilloscope with FFT functionality, you can calculate the THD with a little mathematics.

Firstly, some terminology: Harmonics describes both the fundamental frequency and the overtones of a signal by their order numbers. Overtones describe all integer multiples of the fundamental frequency, starting with twice the frequency of the fundamental wave. The first harmonic corresponds to the fundamental, the second harmonic corresponds to the first overtone.

There are two important representatives of the THD family, THD_R and THD_F . The following applies to both: The smaller the THD, the lower the harmonic content of the measured signal. The greater the harmonic content, the greater the THD. Since harmonics typically occur as an effect of distortion, it is basically a measure for quantifying non-linear distortion.

In this article, the term "overtones" is used when multiples of the fundamental frequency are referred to in their entirety. The term "harmonics" is used when either the totality of all frequency components (including the fundamental wave) or a specific multiple of the fundamental frequency, characterized by an order number, is meant. Example: 3rd harmonic = 3 times the frequency of the fundamental wave.

THD_F denotes the ratio of all overtones to the fundamental wave. This quantity is used, for example, in the measurement of harmonics in power supply networks. THD_R , on the other hand, means the ratio of

all overtone components to the total signal. This quantity is especially popular in audio engineering and is also known as "distortion factor."

The distinction between the two metrics is very important because THD_F and THD_R , although closely related, produce different results. In literature and data sheets, the two values are often confused with each other [1]. In some places, you can find only the term "THD" without further definition of the method used, e.g., in Texas Instruments's data sheet for the LM386.

In practice, THD_F can be calculated relatively easily from the sum of the power level differences of all overtones in relation to the fundamental. From this, in turn, THD_R can be calculated.

$$THD_F = 100 \cdot \sqrt{\sum_{n=2}^{\infty} 10^{\left(\frac{P_n}{10}\right)}} = 100 \cdot \sqrt{10^{\left(\frac{P_2}{10}\right)} + 10^{\left(\frac{P_3}{10}\right)} + 10^{\left(\frac{P_4}{10}\right)} \dots}$$

Formula 1: THD_F (in %), n = order no. of harmonic, P = power level difference to fundamental wave in dBc.

The formula describing this principle may seem daunting at first sight. In simplified terms, THD_F is the square root of the sum of all voltage ratios calculated from the relative power level differences of an infinite number of harmonics relative to the fundamental wave. At first glance, this sentence may seem just as daunting as the formula itself. However, the matter is not as complex as it seems.

The THD_F value calculated this way can then be converted into the distortion factor using the following formula:

$$THD_R = \frac{THD_F}{\sqrt{1 - THD_F^2}}$$

Formula 2: THD_R (in %), THD_F (in %).

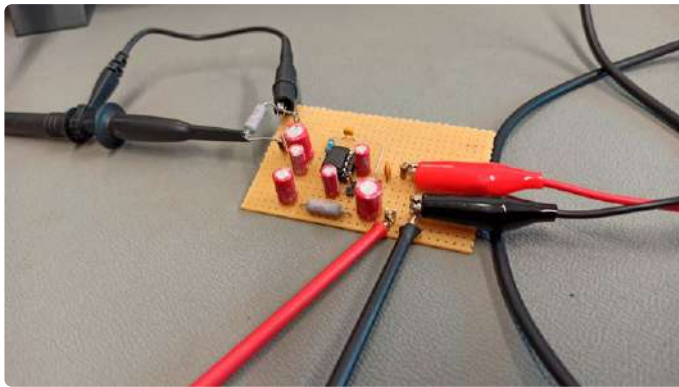
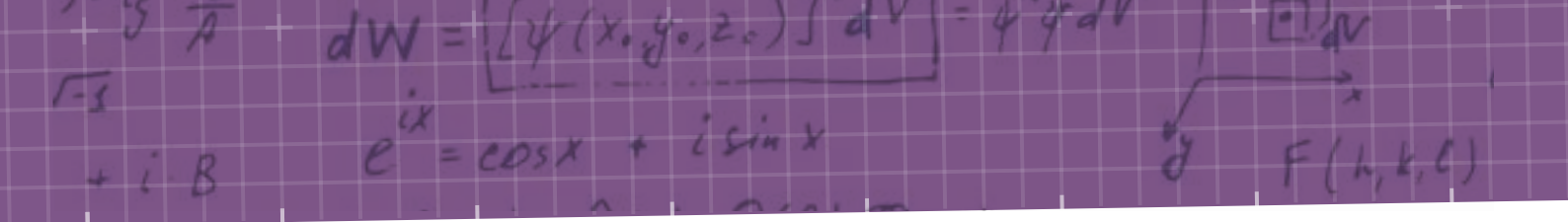


Figure 1: Audio amplifier with an LM386N-1 as DUT.

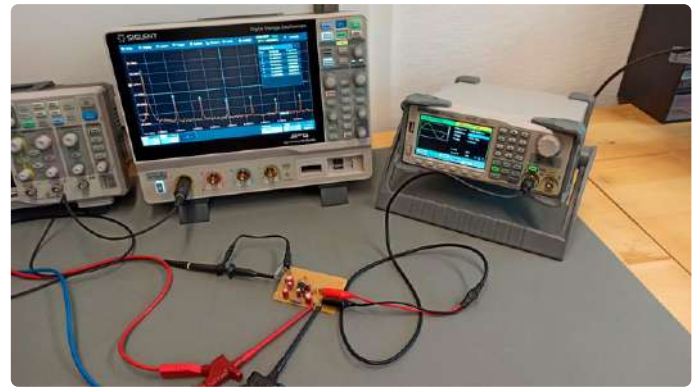


Figure 2: Complete measurement setup with signal generator, oscilloscope, and DUT.

The difference between THD_F and THD_R decreases the smaller the values themselves are. Below 1%, the difference is almost negligible.

In power supply networks, for obvious reasons, the mains frequency, e.g. 50 Hz, is used as the fundamental frequency when calculating THD. In audio technology, 1 kHz is a common standard.

In practice, of course, an infinite number of harmonics cannot be taken into account. For example, 50 harmonics are the maximum considered in power supply networks. In audio technology, it makes sense to neglect non-audible harmonics.

Practical Example

To show that the dull mathematical theory is not so bad in practice, consider the following example: A sine signal with a frequency of 1 kHz is fed to the input of an amplifier (LM386N-1) (Figure 1). The output signal of the amplifier is connected to a 10 Ω resistor (serving as a rough substitute for the impedance of a loudspeaker) and fed through a 1:10 probe to a Siglent SDS2104X HD oscilloscope (Figure 2).

Viewed in the time domain, the output signal initially looks quite clean and sinusoidal (Figure 3). However, this first impression is deceptive,

as a look at the FFT spectrum clearly shows: The harmonic components — here excerpted up to 7 kHz — are clearly visible next to the fundamental at 1 kHz (Figure 4).

For reasons of clarity, I dismissed my initial idea to approach infinity in the number of harmonics considered, at least up to the bandwidth limit of the oscilloscope. I ended up recording the power levels of the first seven harmonics, corresponding to the power level of the fundamental wave and the first six overtones.

Most oscilloscopes offer the option of automatically displaying the measured power levels in a convenient table. Power levels are usually displayed in the unit dBm. This is ten times the decadic logarithm of the ratio of the measured value in relation to a reference quantity. In the case of dBm, the reference quantity is 1 mW.

Before calculations can be made, the absolute power levels must be converted into a signal level difference relative to the fundamental wave. The unit dBc (dB relative to the carrier) is used here. For conversion, the power level of the fundamental wave must be subtracted from the power levels of all harmonics. For example, for the 2nd harmonic (= 1st overtone): -31.7 dBm - 22.1 dBm = -53.8 dBc.



Figure 3: Output signal of the DUT in the time domain.



Figure 4: FFT spectrum of the output signal.

Table 1: Absolute and relative amplitudes of the first seven harmonics (rounded).

Order number of the harmonic	Power level (dBm)	Relative power level (dBc)
1. (1 kHz) = fundamental	22.1	0
2. (2 kHz) = 1st overtone	-31.7	-53.8
3. (3 kHz) = 2nd overtone	-38.7	-60.8
4. (4 kHz) = 3rd overtone	-43.8	-65.9
5. (5 kHz) = 4th overtone	-63.5	-85.6
6. (6 kHz) = 5th overtone	-47.0	-69.1
7. (7 kHz) = 6th overtone	-58.0	-80.1

Table 1 summarizes the harmonics with their order numbers, the absolute power levels in dBm (rounded) and the calculated power level differences relative to the 1st harmonic. The relative power levels from Table 1 can be directly inserted into Formula 1.

$$100 \cdot \sqrt{10^{\frac{-53.8}{10}} + 10^{\frac{-60.8}{10}} + 10^{\frac{-65.9}{10}} + 10^{\frac{-85.6}{10}} + 10^{\frac{-69.1}{10}} + 10^{\frac{-80.1}{10}}} \approx 0.232\%$$

The THD_F value of the amplifier (considered up to 7 kHz) is thus approximately 0.232%. With the second formula, THD_R (the distortion factor) can then also be calculated.

$$THD_R = \frac{0.232}{\sqrt{1 - 0.232^2}} \approx 0.239\%$$

The distortion factor is thus approximately 0.239%. The LM386's data sheet specifies a THD of 0.2%. This is quite close to the values calculated here. Unfortunately, Texas Instruments does not reveal the THD type stated in the data sheet, nor does it reveal how many harmonics were included in the measurement.

Limitations and Notes

The method presented here finds its limits in the dynamic range of the analog-to-digital converter (ADC) of the oscilloscope used. Typically, these are 8-bit ADCs with a theoretical dynamic range of about 48 dB. These can be used up to a minimum THD of approximately 3%.

Attentive readers may have noticed that the THD calculated above as an example is clearly below this limit. This is possible because I used a 12-bit oscilloscope. With 12-bit oscilloscopes, the theoretical dynamic range is around 72 dB. This means that THD values from approximately 0.2% are within the realm of possibility.

However, to make the best use of these limits, caution is advised: The actual available dynamic range also depends significantly on the vertical deflection! If the ADC is over-driven, distortions occur; if it is under-driven, valuable dynamic range is lost.

If you want to try this method directly, you may wonder where to obtain reference signals with known THD values. A simple signal generator can help here: A reasonably symmetrical square wave signal with a 50% duty cycle has a THD_F of about 48.3%. With an equally symmetrical triangle signal with the same duty cycle, the THD_F is still around 12.1%.

If you want to delve a little deeper into the matter and also understand the derivation of the formulas, you can find more information under [2] and [3].

220398-01

Questions or Comments?

Do you have technical questions or comments about this article? Email the author at sebastian@baltic-lab.com or contact Elektor at editor@elektor.com.

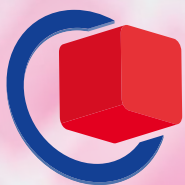


Related Products

- > **Siglent SDS1204X-E 4-ch Oscilloscope (200 MHz)**
(SKU 18524)
www.elektor.com/18524
- > **Siglent SDG1032X 2-ch Signal Generator (30 MHz)**
(SKU 20276)
www.elektor.com/20276
- > **Adafruit 2.5 W Class D Mono Amplifier (PAM8302)**
(SKU 18745)
www.elektor.com/18745

WEB LINKS

- [1] Doron Shmilovitz, "On the definition of total harmonic distortion and its effect on measurement interpretation," Power Delivery, IEEE Transactions, S. 526 – 528, 2005: <https://doi.org/10.1109/TPWRD.2004.839744>
- [2] Sebastian Westerhold, "Total Harmonic Distortion (THD) from dBc," 2022: <https://baltic-lab.com/thd/>
- [3] Sebastian Westerhold, "Total Harmonic Distortion (THD) analysis utilizing the FFT capabilities of modern digital storage oscilloscopes," 2022: <http://dx.doi.org/10.5281/zenodo.6969825>



embeddedworld2023

Exhibition & Conference

... it's a smarter world



JOIN THE EMBEDDED
COMMUNITY

14–16.3.2023



Get your
free ticket now!

embedded-world.com/voucher

Use the voucher code **GG4ew23**

Media partners

Markt & Technik
Das unabhängige Wochenblatt für Elektronik

Elektronik

SmarterWorld
Solutions for a Smarter World

DESIGN &
ELEKTRONIK
KNOW-HOW FÜR ENTWICKLER

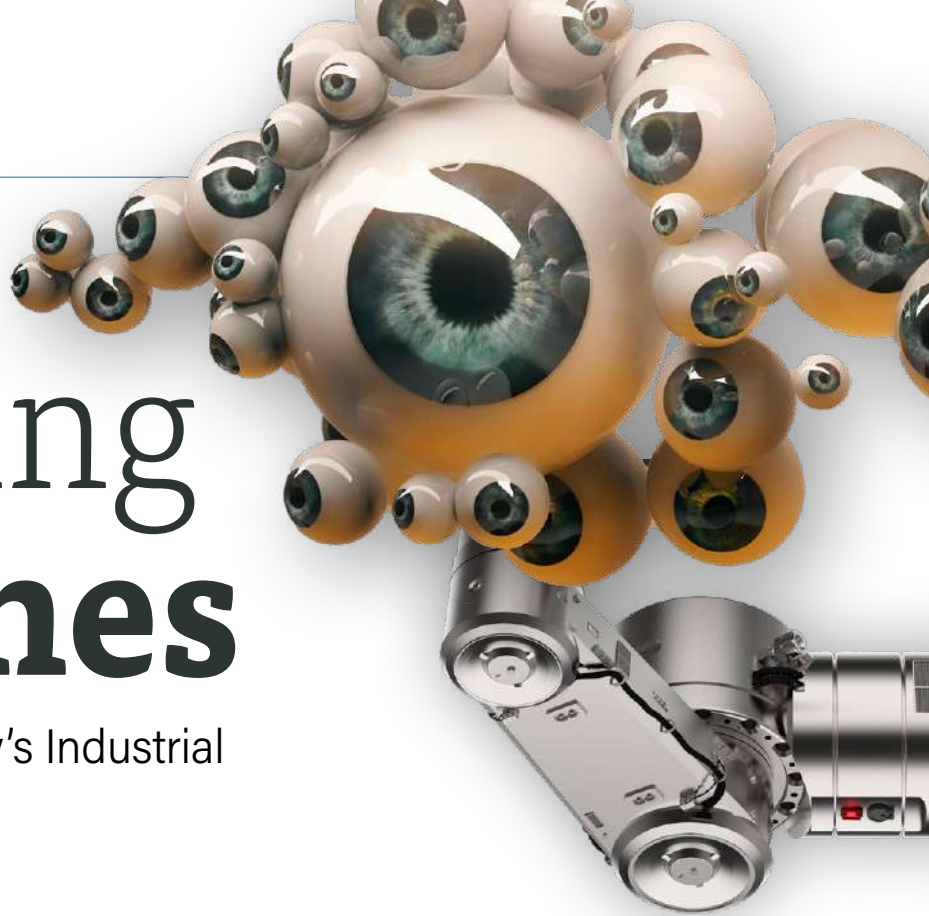
Elektronik
automotive

•medical-design

computer &
automation

elektroniknet.de

NÜRNBERG MESSE



All-Seeing Machines

The Technology Behind Today's Industrial Vision Systems

By Stuart Cording (Elektor)

With the advent of CMOS imaging sensors, low-power cameras are now easily mass-produced in an economic semiconductor manufacturing process. Today, we all carry at least two such devices in our smartphones. Thanks to ever more powerful microcontrollers, image capture and object processing are available to all, and they even support 3D measurements in industrial sensors.

Cameras are never far away in today's society. If you don't have one in your pocket, chances are the person you are sitting next to on the train does. And, if they don't, there is probably one integrated into the train carriage, railway station, or security system in the café you walked past on the way in. Cameras have become this ubiquitous thanks to CMOS image sensors, a technology that significantly reduced their cost, power consumption, and size compared to the old charge-coupled devices (CCD) technology they've replaced. While capturing images and video is one task, the past decade's rapid growth in computing power has made it possible for computers to analyze incoming visual data in real-time. This enables automated inspection for various industrial applications, such as detecting misplaced components on PCBs, locating parts on conveyor belts, and even assessing the quality of a welded joint.

CCDs — Imaging Tech of the Past

Image sensors use an array of photodiodes to convert photons into an electrical current. In CCDs, a string of photodiodes is linked via switches. After light has been allowed to strike the imaging surface, the charges present in each pixel are relayed through the string (charge passing), one at a time, to the edge of the sensor (**Figure 1**). Here the charge is converted into a voltage before being converted into a digital value by an analog-to-digital converter (ADC). CCD sensors are manufactured in a metal-oxide semiconductor (MOS) process, which differs from the complementary MOS (CMOS) technology used for most silicon devices.

The longevity of CCD as a sensor technology is linked to its high quantum efficiency, a measure of how well the device converts

photons into electrons, which can be as high as 95%. Beyond commercial and consumer camera applications, CCDs were used in scientific apparatus, such as spectrometers and even the Hubble telescope [1]. However, the evaluation of the images they captured required external circuitry.

CMOS image sensors emerged as a viable technology in the 1990s. Initially, the image quality was poor, with low resolution, noisy images, and poor color definition. Due to necessary differences in design, the photodiodes in these sensors were much smaller than those in a CCD sensor with an equivalent number of pixels. Rather than implementing a charge-passing circuit, each pixel required its own amplifier, reducing the surface area available to each pixel (**Figure 2**). However, thanks to the immense development and research driven by clear market needs, image quality between the two technologies today is minimal. The ease of production that comes from using a CMOS manufacturing process has also pushed prices down, improving integration with much of the necessary image processing support included on the sensor die. Furthermore, power consumption is around 100 times lower than an equivalent CCD sensor, something critical for today's battery-powered portable devices.

Reduction of noise in pixels and increase in frame rate is achieved through massively

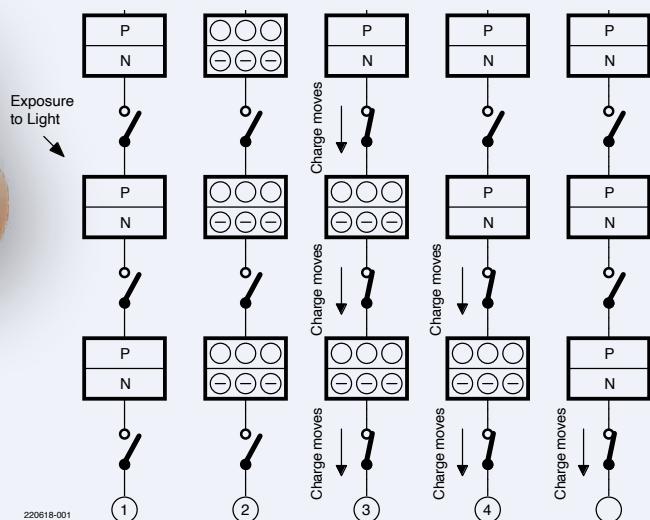


Figure 1: CCD image sensors used charge transfer between pixels to read out their values. (Source: Canon Inc.)

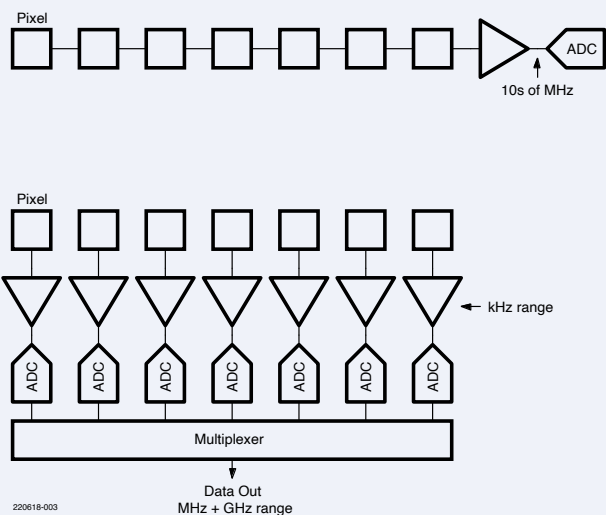


Figure 3: Because CMOS image sensors are massively parallel, the analog front end does not need to be as high speed as that used in CCD sensors. (Source: Teledyne Digital Imaging Inc.)

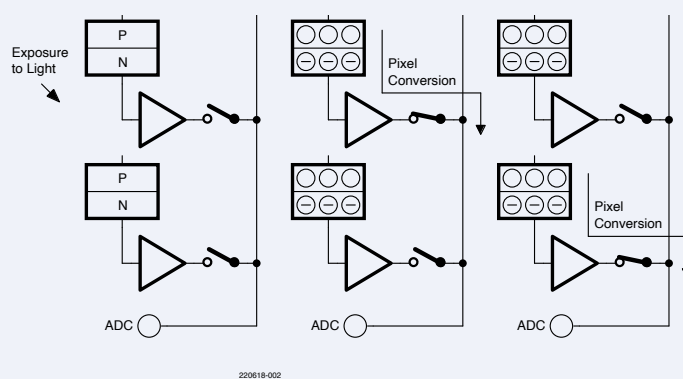


Figure 2: CMOS image sensors use an ADC to convert the value of each pixel individually. (Source: Canon Inc.)

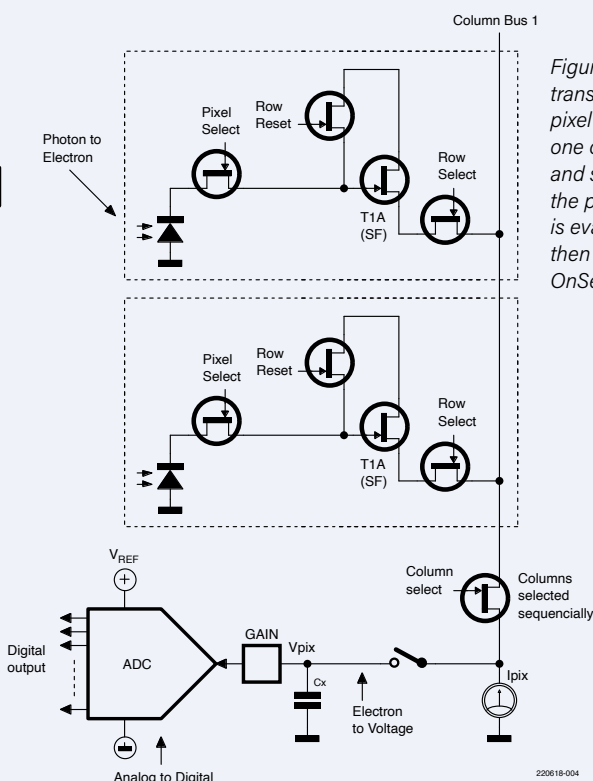


Figure 4: This four-transistor CMOS pixel architecture is one of many used and shows how the pixel's charge is evaluated and then reset. (Source: OnSemi)

parallel data conversion, with multiple ADCs converting the charges in a line of pixels simultaneously. Compared to CCDs, this allows the use of lower-bandwidth amplifiers and ADCs (**Figure 3**). Pixel designs vary but usually consist of a high-impedance amplifier connected to the photodiode junction and switches for row and column selection. Finally, a switch is provided to clear the charge on the pixel (**Figure 4**).

Adding Color

While using the popular CMOS production process may have reduced the cost of image sensors, they are anything other than normal when it comes to manufacturing. The

photodiodes are sensitive to all visible light, so any image captured would only deliver a greyscale image. To produce color images, a color filter array (CFA) is needed. A Bayer filter is typically used, named after Bryce Bayer of Eastman Kodak, who invented it. It uses red, blue, and two green color filters arranged in a pattern that ensures any square of four pixels is half green, quarter red, and quarter blue. The higher weighting for green is due to the physiology of the eye, which perceives green more readily [2].

The resultant raw image is not suitable for immediate human consumption without

pre-processing. This is undertaken using a process called demosaicing. At its simplest, each pixel is converted to an RGB (red/green/blue) value based on the values of specific pixels surrounding it. However, some demosaicing approaches, such as linear interpolation, can cause artifacts in the final image, especially around the edges of objects. To combat this, a selection of algorithms are available that attempt to minimize artifacts while delivering a fast final image [3].

Another difference to typical chip manufacture is the need for a microlens layer. Since the photosensitive area of each pixel is so small,

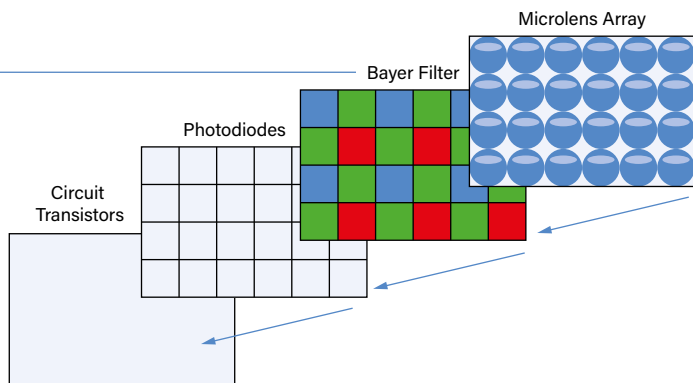


Figure 5: Unlike typical semiconductors, image sensors require a color filter and microlens array in their construction.

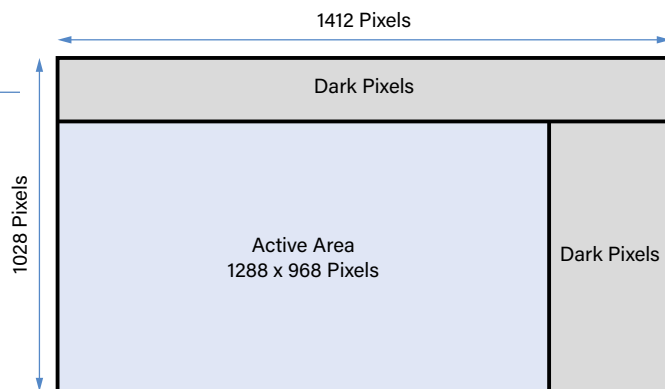


Figure 6: Additional pixels may be implemented in a CMOS image sensor to provide black-level compensation. (Source: OnSemi)

the lens helps to concentrate the incoming light on the pixel that, in turn, increases the quantum efficiency (Figure 5).

Understanding Electronic Shutters

How the image sensor shutter works can also be challenging to understand for the uninitiated. An electronic shutter is used to expose the pixels to light for the correct amount of time. Typically, two main options are available, global and rolling, and some image sensors support both. With a global shutter, all the image sensor pixels are electronically controlled to be exposed at the same time. Afterward, each line of pixels is converted to digital values and passed to the host processor that handles image processing. With a rolling shutter, each row of pixels is exposed to the light one after the other, with the conversion taking place immediately for each line.

The difference in the image that results is partially impacted by the shutter approach used. A global shutter captures the image in a single moment. By comparison, the rolling shutter captures each line with a slight delay compared to the previous line. When the object being captured by the sensor is moving, artifacts in the resultant image can occur. For example, when photographing a locomotive moving from left to right, the final image will show smear from left to right starting from the top of the image.

While the obvious answer would seem to be to select a global shutter, there are, as always, more details to be understood. The rolling shutter approach requires fewer transistors around each pixel, leaving more area for its photosensitive portion, resulting in improved

image quality. The simplified design also leads to lower noise in the pixel. Finally, the rolling shutter enables higher frame rates to be achieved than a global shutter design because the top rows of the next frame can begin exposure while the final rows are being converted.

The decision as to which is better depends on the application, and the issues caused by the rolling shutter can be resolved by using a flash synchronized to the shutter. Some scientific instruments require a global shutter approach, while a visual inspection camera will be fine with a rolling shutter and synchronized flash light source. A helpful resource to understand the issues in more detail is provided by PCO AG [4].

Another potential curiosity with image sensors is the difference between the number of pixels on the sensor and the number available to the user's application. The OnSemi AR0130CS [5] is an example, a 1/3-inch CMOS digital image sensor with a rolling shutter. This 1.2-megapixel sensor boasts 1280 × 960 active pixels in the datasheet's key performance parameters list. However, the actual sensor is 1412 × 1028 pixels with an active area of 1288 × 968. Some of the upper rows and a block of the columns on the right-hand side are optically black. These are used internally to monitor the black level and provide black level correction (Figure 6).

A further challenge in acquiring a good quality image lies with the power supply. The power supply rejection ratio (PSRR) of the low dropout regulators (LDO) used needs to restrict any noise from entering the CMOS image sensor's supply pins. If not, the risk

of pixel noise is increased. According to an application note by OnSemi [6], as image sensors go beyond 50 megapixels, a PSRR of greater than 90 dB up to 10 kHz and greater than 45 dB between 1 and 3 MHz is recommended. The dynamic load can also be quite considerable, with a few hundred milliamps of load change during row transitions. Here bulk capacitors are essential to attain optimal performance. Finally, consideration should be given to the output noise of the LDO (nV/√Hz), as anything in the 10 Hz to 1 MHz range can result in additional noise in the pixels, reducing their dynamic range.

Machine Vision with Microcontrollers

With their 32-bit processors and a megabyte of SRAM or more, today's larger microcontrollers are well-suited to simple vision applications and can even integrate some useful machine learning (ML) algorithms for object detection. One project to consider is OpenMV [7], a combination of hardware and software development environment specifically targeting machine vision. Using the STM32 family of microcontrollers, their hardware provides access to image sensors of up to 5 megapixels that can be exchanged as needed.

Software development is supported by the OpenMV integrated development environment (IDE) that is tuned to the needs of machine vision (Figure 7). On the left-hand side is an editor for writing code in MicroPython, and a serial terminal, displaying text output from the application code. On the right-hand side is the output from the camera. At the top is the frame buffer showing the image from the camera combined with any output injected into the video stream by the appli-

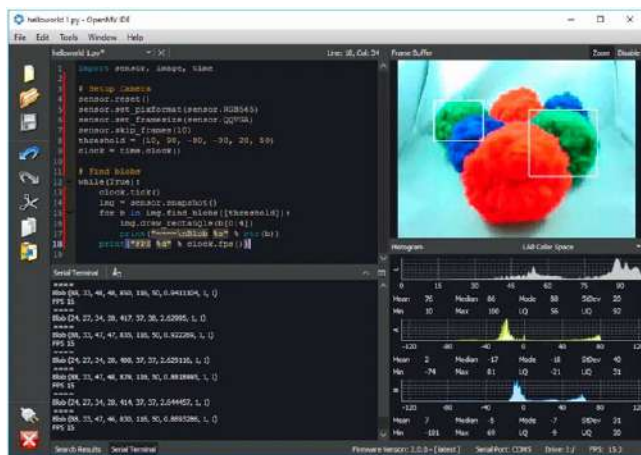


Figure 7: The OpenMV development environment provides an editor for coding and displays the image captured by the attached sensor. (Source: OpenMV)

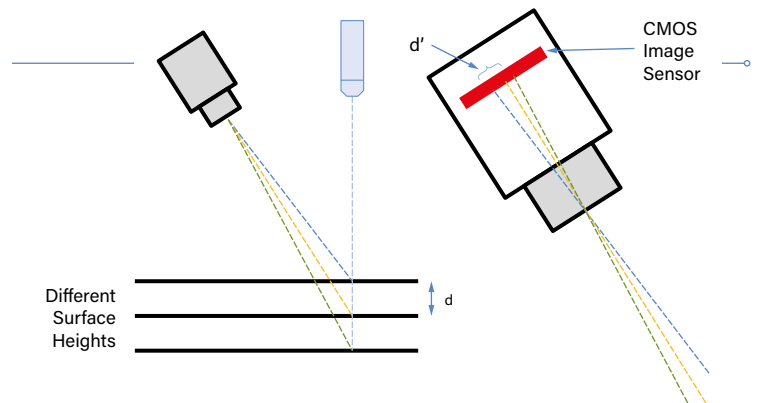


Figure 8: Laser triangulation uses a CMOS image sensor to detect the reflection of a laser on a surface to determine its height. (Source: MoviMED)

cation. This could include text messages or bounding boxes highlighting objects detected using ML. At the bottom, histograms are displayed, such as the image's red, blue, and green levels. Much like the Arduino environment, example projects are available. These also include ML models for applications such as face and person detection.

3D Imaging Using Laser Triangulation

While cameras with CMOS image sensors can be used for a multitude of vision applications, these typically lie in the domain of 2D imaging. However, a host of applications require accurate surface measurements that call for a 3D metrology capability. In the industrial space, laser triangulation is a common solution used to measure the surface of finished goods such as welds, glued beads, and the position of components on circuit boards.

The approach for these sensors is to project a laser spot or line onto the measurement object. Light reflected from the surface hits an integrated CMOS sensor which determines the distance to the object using triangulation (Figure 8). How the image sensor and laser are positioned determines the complexity of the calculations required for determining the distance, resolution, and likelihood of measurement errors. For example, the laser can be perpendicular to the object (standard geometry) with the camera at an angle, making for a good general-purpose sensor. However, when measuring highly reflective surfaces such as glass and polished metal, a look-away geometry with both the image sensor and laser at an angle compared to the measurement is a better design.

Sensor manufacturers such as SmartRay offer a wide range of 3D sensors based on this technology. Their ECCO 95.015G [8] sensors are specifically designed to scan flat glass and specular, highly reflective surfaces. Mounted at a distance of 23.5 mm from the measurement object, they offer a vertical resolution of 0.42 – 0.54 μm . The unit delivers its measurements as a series of points over a gigabit Ethernet interface that can be interfaced with typical industrial imaging software to evaluate the data.

CMOS Image Sensors for All

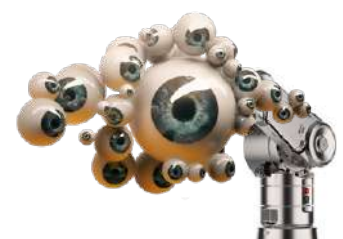
With their low cost and low power, CMOS image sensors have become ubiquitous in today's society. In sensing applications where ultrasound or infrared sensors would have been deployed, it is perfectly conceivable that a CMOS image sensor could be used today. However, with millions of pixels and functionality that is challenging to understand, they can be difficult to interface with microcontrollers. Thankfully, the computing

power and memory capacity of today's microcontrollers are capable of handling relatively complex machine-learning object detection functionality in conjunction with modestly-sized image sensors. And these devices are not solely about 2D images. Industrial sensors using laser-based triangulation incorporate CMOS imaging devices to deliver micrometer accuracy measurements in three dimensions, automating visual inspection of workpieces to assure their quality. ◀

220618-01

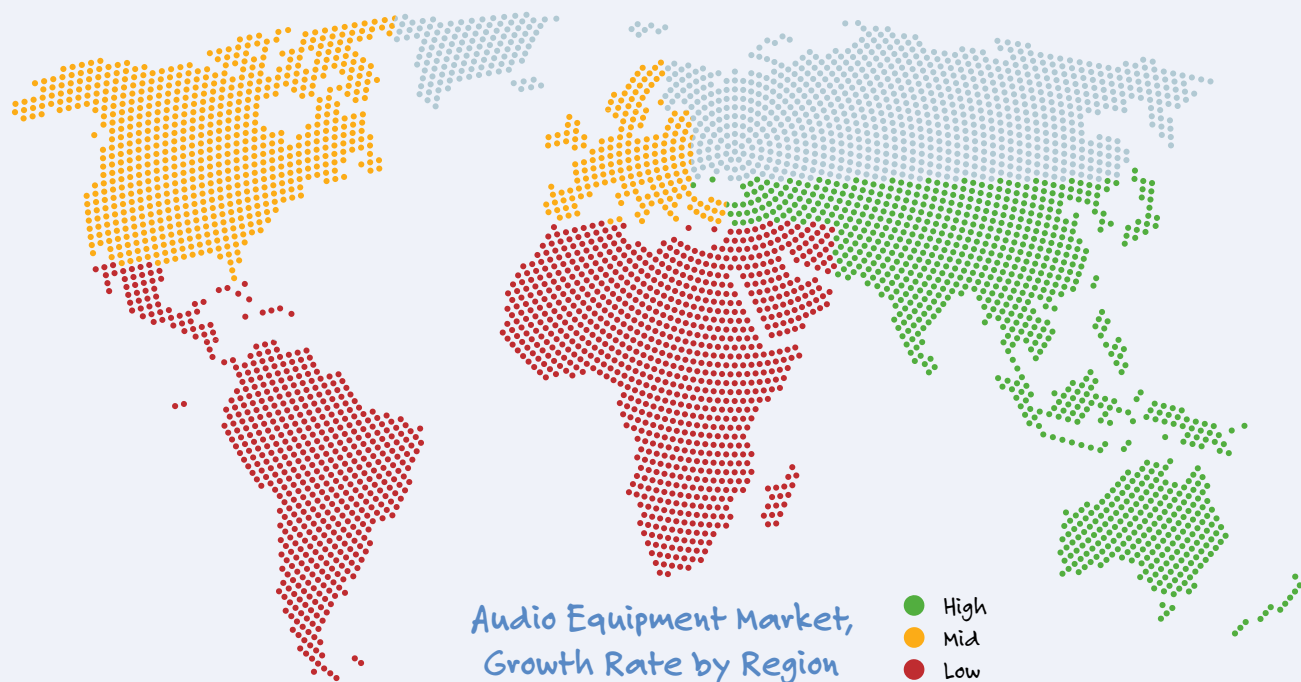
Questions or Comments?

Do you have technical questions or comments about this article? Email the author at stuart.cording@elektor.com or contact Elektor at editor@elektor.com.



WEB LINKS

- [1] "Hubble's Instruments: WFPC2 Wide Field Planetary Camera 2," European Space Agency: <https://bit.ly/3WBhN1Z>
- [2] "The Human Eye's Response to Light," Iowa State University: <https://bit.ly/3UsmVDU>
- [3] A. Rajwade, "Color Image Demosaicing," Indian Institute of Technology Bombay: <https://bit.ly/3UneoCg>
- [4] "What are all the discussions about global vs. rolling shutter?," PCO AG: <https://bit.ly/3h90h4R>
- [5] Product Page: AR0130CS CMOS Image Sensor: <https://bit.ly/3hgWq63>
- [6] M. Dadafshar, "Understanding Challenges in Powering High Resolution, High Frame Rate CMOS Image Sensors," OnSemi, July 2022: <https://bit.ly/3Dz7EtS>
- [7] Website: OpenMV: <https://openmv.io/>
- [8] Product Page: ECCO 95.015G: <https://bit.ly/3E7HakZ>



The Asia-Pacific Region Is Taking the Lead

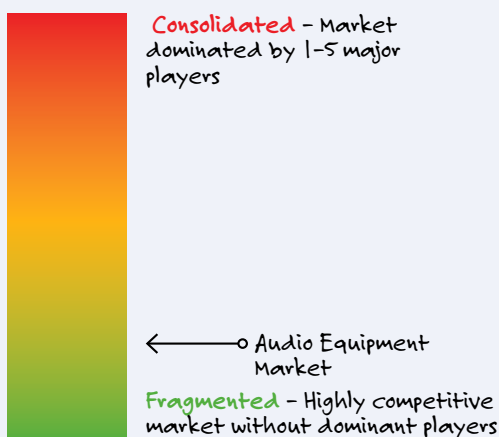
When it comes to audio equipment in a professional environment — mixers, amplifiers, microphones and speakers/monitors — experts envisage a growth rate of about 6% for the next four years. This includes all audio equipment within the automotive sector. The market will

grow from roughly USD \$13.5 billion in 2022 to USD \$18 billion in 2027. The Asia-Pacific is the fastest growing region, followed by the regions of Europe and North America. The difference in growth rates between the Asia-Pacific, Europe/North America and Africa/South America

can, in large part, be explained by a booming automotive market in China and India. A single car could carry as many as 10 speakers, maybe even more.

(Sources: Fortune Business Insights, Mordor Intelligence - map)

Market Concentration Audio Equipment Market



Source: Mordor Intelligence (Report Audio Equipment Market, 2022-2027)

Market Concentration? Not So Much

The professional audio equipment market has a bright future. As a consequence, there is no lack of interest from vendors in exploring this market. This, in turn, brightens up the prospects for the audio equipment buyers even more. The relative lack of market concentration means that price competition in the audio equipment market will certainly be stiff. Also, there are many innovations going on, whether they are related to amplifiers, microphones, or speakers. In short: The professional customer can expect high-quality sound input and output for reasonable prices. Thus, audio innovation will give a boost to video conferencing

(Sources: Microsoft, Mordor Intelligence)

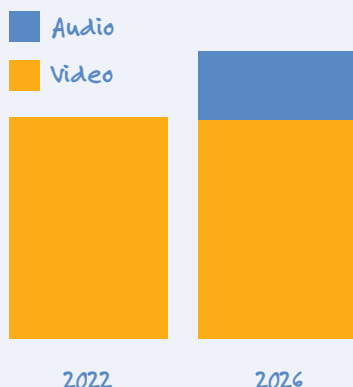
The Car of the Future Is... Sound Out/Sound In

When putting some numbers together, it becomes obvious that the market for professional audio equipment consists mostly of audio equipment in automobiles. Please see the graph. Why is the car so prevalent here? The first reason is that drivers spend a great deal of their time in their cars, with estimates of some 45 hours per year extra due to traffic delays. The second reason is that using infotainment services in a car can be dangerous: They can distract the driver from paying attention to what is happening on the road. That is why voice recognition is so important. Voice recognition requires audio processors and these are also part of the audio equipment market.

(Sources: Fortune Business Insights, Mordor Intelligence)



An Overview: Video Wins from Audio



There is a difference between the professional audio market and the total audio market. The latter includes home users. The total audio market is about all kinds of recorders and players, not only about professional mixers, etc. The same is true for video equipment: A television camera is different from a smartphone camera. However, when we bring professional use and home use together for

both audio and video, what will the numbers be? Well, this market was worth about USD \$158 billion last year, with a projected growth to USD \$196 billion in 2026. Roughly 80% of the total market consists of video equipment. Curved TVs will be a winner during the coming years.

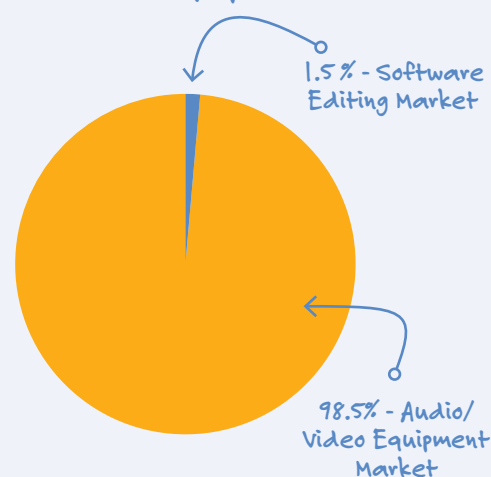
(Sources: Market Prospects, Research and Markets)

Hardware? OK, But What About Software?

Until now, we have been talking about hardware. What about software? There is, of course, a market for audio and video editing software, if only because of YouTubers. The software market grows at an even higher pace than the market for audio and video equipment: 7% instead of around 6% for the years to come. However, when compared to audio and video hardware, the market for audio and video software is just a drop in the bucket. Based on figures from Transparency Market Research, the market for audio and video editing software will grow from around USD \$2 billion in 2020 to roughly USD \$3 billion in 2030. This means that the audio/video software market represents about 1.5% of the hardware market.

(Sources: Research and Markets, Transparency Market Research)

Software Editing Market vs. Audio/Video Equipment





The Evolution of Voice and Audio Control for Electronic Devices

▲

Figure 1: Voice-controlled robot assistant.
(Source: PaO_STUDIO on Shutterstock)

By Mark Patrick (Mouser Electronics)

Speech is an efficient way for people to express ideas and desires. Before the industrial age, humans discovered that animals could be trained to recognize and respond to basic commands instructing them to perform a task. The next logical step was to develop a way to communicate with and command our machines using our voices. The use of voice and audio as a control interface for electronic devices has grown in popularity in recent years, and it is evolving to meet users' expectations and the requirements of new applications. In this article, we explain the benefits of voice and audio to control electronic equipment and machines and review how it is implemented. We also show how this control interface can now be embedded into offline devices and how the audio experience they provide can be greatly improved.

Using Voice to Control Electronic Devices

There are several obvious benefits to be derived from voice-machine interaction:

- For humans, speech is an intuitive form of communication, which makes it easy to convey commands verbally.
- Voice communication is still possible even when a person's eyes and hands are otherwise occupied. While simultaneous voice control is convenient, in some circumstances like driving, for example, it is illegal to attempt to control another device by touch.
- Voice is an efficient medium with which to control machines that can listen and respond without the requirement for complex commands.
- Voice integration minimizes the requirement for a touchscreen on many devices. This is especially desirable for remotely located or portable battery-powered equipment, where reducing the size and power consumption are common design challenges. Removing the requirement for touch is also more hygienic for applications that have multiple users.
- As shown in **Figure 1**, it can be an enabling tool for people with disabilities for whom touch may not be a viable option. Voice communication with machines can be used to perform tasks such as opening doors, or to remotely communicate updates on a person's health.

The audio front end (AFE) of a voice-controlled device includes a microphone array and signal-processing blocks. The AFE processes the signal from a multi-channel microphone array to cancel out interference from any background noise or from playback by the device itself. This signal is then sent to a 'wake-word' detection engine which recognizes words like "Alexa" or "OK Google," which are pre-programmed on the device. Multiple signal-processing algorithms are used to cancel out unwanted interference signals. Components of a voice control solution include:

Microphone array: Voice-activation systems require one or more microphones to capture the audio control signal. Size, cost, performance, and robustness are the main considerations when choosing a microphone array. Combining different signals from a multi-microphone array helps to improve the signal-to-noise ratio (SNR) for the audio signal chain.

Direction-of-arrival (DoA) detector: This is used to determine the position of the user relative to the device being controlled so that the microphone array can point a beam in the direction of the voice.

Beamformer: This accepts sounds from the DoA while rejecting sounds from other directions. Its performance depends on the geometry and the SNR of the microphone array as well as beam width, and the level of background noise.

Acoustic echo canceller (AEC): This rejects the playback signal on the device speaker itself (for example, if the device speaker is playing music) to allow the user's voice command to be picked up clearly.

Adaptive interference canceller (AIC): This cancels out external noise from other sources of sound that are difficult to cancel out with a traditional beamformer, for example, loud noise being produced by other equipment.

Wake-word detector: The processed voice signal from the AFE is compared to a library of wake words, such as "Hey Google" using a wake-word detection algorithm which is part of a machine-learning model. Larger models are more accurate – for example, a 1-MB trained model will be more accurate than a 64-KB model, but is more processor-intensive. Large wake-word models are required to accurately detect the wake word to reduce the number of false alarms.

Class D Audio Amplifiers

The voice processing part of this control interface has undergone much development, which now allows even low-cost devices to provide accurate voice recognition capability. However, the audio side of the interface has received significantly less attention, meaning the sound quality produced by many early smart speakers and other audio enables Internet-of-Things (IoT) devices was inferior when compared to that produced by higher end audio equipment.

The novelty associated with voice control was perhaps considered a sufficient distraction to deflect attention from this shortcoming. However, as smart devices gain wider adoption, consumer expectations of the audio experience which they provide are growing. The low efficiency of traditional class AB audio amplifiers makes them impractical for use in low-power IoT devices, but several chipmakers have recently introduced a range of cutting-edge Class-D audio amplifiers

that are a significant improvement on those previously available. Many of these have been specially developed for to enable high-quality audio in smart technology and IoT devices.

The TAS2770 [1] 15-W input audio amplifier by Texas Instruments [2] increases loudness and audio quality, and its increased ability to capture voice means easier and more natural operation of voice-controlled devices. It is the first audio front-end to combine a digital microphone input with a powerful I/V sense amplifier, allowing it to capture voice and ambient noises for echo cancellation or noise reduction in voice-enabled applications. Maxim Integrated [3] (now part of Analog Devices [4]) have designed their MAX98357 [5] and MAX98358 [6] class-D amplifiers to have an efficiency of a 92% and deliver 3.2 W of Class AB audio performance. A simplified block diagram for these amplifiers is shown in **Figure 2**. The PAM8106 [7] by Diodes Incorporated [8] has low power consumption, enabling it to work well in devices driven by 1.5-V lead-acid batteries, as well as 3.5-V lithium-ion batteries. It is 92% efficient and can be implemented without the need for a bulky heatsink in a compact design architecture.

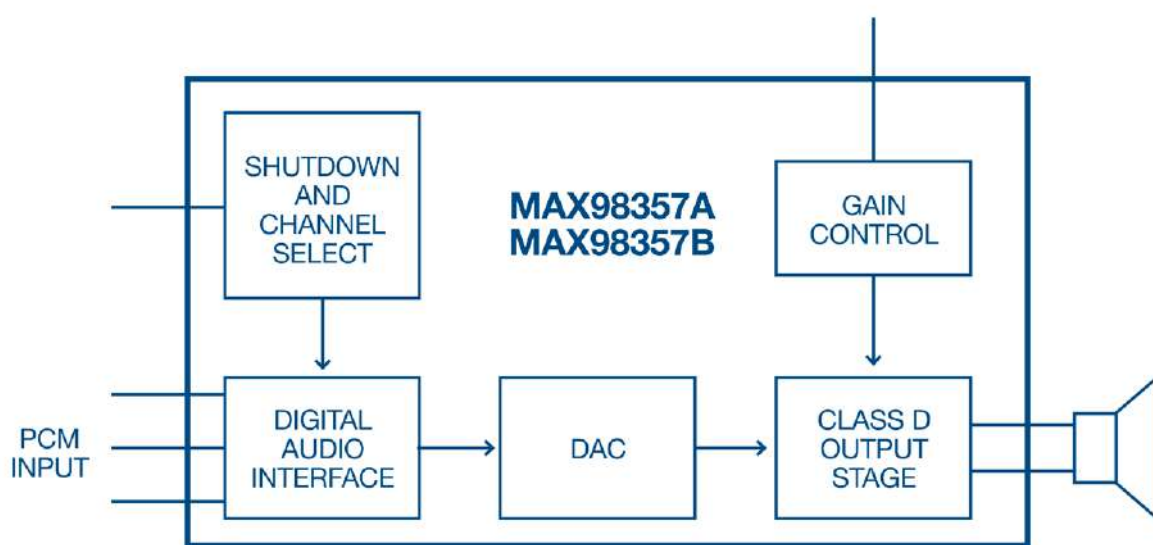
Figure 2: Simplified block diagram of Class D audio amplifiers by Maxim Integrated. (Source: Maxim Integrated)

Offline Voice Control

Cloud-based solutions like Amazon's Alexa and Google Assistant are readily available for devices with

Figure 3: SLN-LOCAL2-IOT offline voice control solution by NXP. (Source: NXP [10])

a stable internet connection, but for those with poor or no connectivity, offline voice control is a preferable solution. For example, if a product is required to respond to simple, single-word commands like go, stop, reset, etc. (often referred to as keyword spotting) then it makes sense for processing to be performed locally on the device itself. Implementing a simple keyword command system is possible using a low-cost embedded microcontroller, like, for example, NXP's EdgeReady MCU-based solution for offline local voice control. This uses the i.MX RT crossover MCU and enables developers to quickly add voice control to their products. NXP's i.MX RT106S based solution includes the SLN-LOCAL2-IOT [9] development kit shown in **Figure 3**.



This comes with fully integrated software, running on FreeRTOS and is provided with a software development kit (SDK) to allow for quick proof-of-concept. This solution enables manufacturers to easily add voice control to their smart home, smart appliance, smart building, and smart industrial products, without the need for Wi-Fi or cloud connectivity. Offline voice control also helps to address the privacy concerns of many consumers, who fear that their system could be vulnerable to online intruders. The kit also has a Windows-based tool that can be used to create speech models for over one hundred custom commands and multiple wake words from text input in over 40 languages.

Conclusion

Voice and audio are quickly becoming the interfaces of choice in many smart devices, but are particularly suitable for use in low-power and portable IoT devices because it removes the requirement for expensive and power-hungry digital displays. Many early systems had inferior audio quality and could only be implemented using a cloud-connected solution.

However, a new generation of high-efficiency Class D audio amplifiers now makes it possible for manufacturers to ensure that their equipment provides consumers with a high-quality audio experience. Solutions have also emerged that now make it possible to control devices that have poor internet connectivity or none. These innovations demonstrate the ability of this technology to be adapted to meet new

demands that are emerging as people become more accustomed to this control interface, and this trend is likely to continue. ◀

220620-01



About the Author

As Mouser Electronics' Technical Marketing Manager for EMEA, Mark Patrick is responsible for the creation and circulation of technical content within the region — content that is key to

Mouser's strategy to support, inform and inspire its engineering audience.

Prior to leading the Technical Marketing team, Patrick was part of the EMEA Supplier Marketing team and played a vital role in establishing and developing relationships with key manufacturing partners. In addition to a variety of technical and marketing positions, Patrick's previous roles include eight years at Texas Instruments in Applications Support and Technical Sales.

A "hands-on" engineer at heart, with a passion for vintage synthesizers and motorcycles, he thinks nothing of carrying out repairs on either. Patrick holds a first-class Honours Degree in Electronics Engineering from Coventry University.

WEB LINKS

- [1] TAS2770 Audio Amplifier Evaluation Module: <http://elektor.link/MouserTAS2770>
- [2] Texas Instruments @ Mouser: <http://elektor.link/MouserTexasInstruments>
- [3] Maxim Integrated @ Mouser: <http://elektor.link/MouserMaxim>
- [4] Analog Devices @ Mouser: <http://elektor.link/MouserAnalogDevices>
- [5] MAX98357: <http://elektor.link/MouserMAX98357>
- [6] MAX98358: <http://elektor.link/MouserMAX98358>
- [7] PAM8106 10W Audio Amplifier : <http://elektor.link/MouserPAM8106>
- [8] Diodes Incorporated @ Mouser: <http://elektor.link/MouserDiodesIncorporated>
- [9] SLN-LOCAL2-IOT Solution for Local Voice Control: <http://elektor.link/MouserSLNLOCAL2IOTDevkit>
- [10] SLN-LOCAL2-IOT Product: <http://elektor.link/MouserSLNLOCAL2IOT>

WEEF 2022 in Review

By C. J. Abate (Elektor)

The ethics movement in electronics was in sharp focus at the Lenthe Foundation's second World Ethical Electronics Forum, which took place at electronica 2022 in Munich, Germany.



The 2nd World Ethical Electronics Forum took place at the electronica 2022 in Munich.
(Source: Messe München)

On November 15, 2022, at the electronica fair in Munich, the World Ethical Electronics Forum (WEEF) brought together engineers, electronics industry executives, academics, and tech journalists to discuss many of the most important ethical electronics-related issues of our time. During the day-long event, the Lenthe Foundation — an Amsterdam-based nonprofit producer of events about social entrepreneurship — and its media partners — Elektor, ELEKTRONIKPRAXIS, and electronica — offered a platform for forward-thinking leaders to discuss the following topics: eco-friendly design practices, the importance of sustainable development goals, the planet and people before profit, ethical HR practices, sustainable resources, and much more.

The event featured stimulating talks and insights from a wide range of thought leaders: Prof. Dr. Stefan Heinemann (Professor, FOM University of Applied Sciences), Dr. Reinhard Pfeiffer (CEO, Messe München), Florian Kurz (KK Ventures), Michel Lorient (AGIGA), Tessa Quandt (VARTA), Dr. Ralf Bodelier (author, journalist), Hendrik-Jan Overmeer (founding team member, Deedmob), Andrea Barrett (VP Social Responsibility and Sustainability, RS Group), Robert van Kats (Co-Founder, bkvv architects), Milda Pladaitė (Initiator of Engineers Enterprise), Kelly Heaton



The World Ethical Electronics Forum Award 2022. (Source: Messe München)



WEEF Award Winner Milda Pladaitė — awarded by Beatriz Sousa (Elektor). (Source: Messe München)



Award winner Frank Stührenberg (CEO of Phoenix Contact) with Dr. Reinhard Pfeiffer (CEO of Messe München) and Don Akkermans (Lenthe Foundation / Chair of the Program Committee). (Source: Messe München)

(artist, USA), Johann Wiesböck (Editor-in-Chief, Elektronikpraxis), C. J. Abate (Content Director, Elektor), Udo Bormann (Project Director of electronica fast forward 2022 Startup Award), Stuart Cording (Journalist, Elektor), and Don Akkermans (Chair, WEEF). In addition to the talks, the WEEF jury honored four WEEF award recipients:

- **Sven Krumpel:** The owner of CODICO, which gives its employees cultivation areas for growing fruit and vegetables.
- **Gopal Kumar Mohoto:** An engineer working on electrifying transport in Bangladesh, he is the Co-Founder and CTO of Advanced Dynamics.
- **Frank Stührenberg:** The CEO of Phoenix Contact, a board member of

the KlimaWirtschaft Foundation, and a major contributor to the All Electric Society.

- **Milda Pladaitė:** A civil engineer from Lithuania who is part of the World Federation of Engineering Organizations (WFEO) Young Engineers Future Leaders Committee.

The full event was caught on camera by Elektor. Watch it at elektor.tv!



Curious about what's next for WEEF? Visit worldethicalelectronicsforum.com for up-to-date news about WEEF, the 2023 WEEF Index, and ethical electronics-related news items and articles. In early to mid-2023, the Lenthe Foundation (the producer of WEEF) will launch the annual WEEF GUIDE. It will be distributed (online and in print) to thousands of WEEF partners and stakeholders. The WEEF GUIDE will have the profiles of thought leaders listed in the 100 WEEF INDEX, winner interviews, controversial comments from outside the industry, interviews with ethical leaders, and much more. If you want to contribute, please contact Shenja Panik at shenja.panik@elektor.com. 📩

220642-01



Moderator Johann Wiesböck and Don Akkermans interviewing business ethics Professor Dr. Heinemann (FOM).



Stuart Cording was one of the two moderators leading through a day full of ethical debates. (Source: Messe München)



Don Akkermans and C. J. Abate discussing research results about electronica exhibitors and their ethical representation.

FFWD

electronica 2022 in Review

Innovators Did Not Fail to Impress

By Brian Tristram Williams

After months of preparation and building excitement, the Fast Forward 2022 start-up and scale-up awards, powered by Elektor, were held at electronica 2022 in Munich.

In November, at electronica Fast Forward 2022 — powered by Elektor, we came for unicorns, and we were not disappointed. We finally got to see all of the start-ups and scale-ups that we've been looking forward to over the preceding few months. At the Elektor booth, we hosted all of these friends at a made-for-the-purpose Fast Forward 2022 exhibition and presentation area.

The Participants

Touting their innovations, each in their own booths, were the start-ups:

- > **Opulo**, with their innovative DIY open-source pick-and-place machine.
- > **AMSEL** gave us an in-depth look at their groundbreaking surface pre-treatment solution.

- > **AirHood** showed us their range/cooker hood that isn't a hood, but a portable, aesthetically pleasing countertop solution for kitchen fumes.
- > **V-Juice** showed off their sleek and slimline wireless charging ideas, with more application angles than just the engineering — marketing, gamification and incentivization are all possibilities.
- > **Treesense** brought environmental sensors to the fore, allowing trees to talk to us and tell us what they want, saving many manual resources in the management and upkeep of flora.

The scale-up:

- > Already an earlier favorite, **wheel.me** showed off their autonomous wheel solutions, which look sure to put an end to back-breaking industrial and commercial lifting and hauling.

Innovation Lane:

- > **Voltera** demonstrated their existing products, plus an exciting new one that offers the ability to print (almost) anything on anything.
- > **Quadruped** was a fair-visitor favorite, drawing crowds with their four-legged robots boasting the company's software solutions.

And a university project:

- > **Ecurie Aix** was proud to present their RWTH Aachen University project, which iterates each year on their fully autonomous electric Formula Student racecars.



The Fast Forward 2022 exhibition and presentation area — powered by Elektor.



First-prize winner wheel.me's Kai Kreos-Nemcock.



A wheel.me autonomous wheel does heavy lifting intelligently.



Third-prize winner Treesense's Alba Dervishi.

All nine participants got to present their developments in depth on Tuesday and Wednesday, and then, on Thursday, the start-ups and scale-up wowed us with their official quick-pitches.

If the jury members were looking for an easy job, this was not the place for it. Deliberations were held late into the night Thursday, and finally, on Friday, the winners were announced.

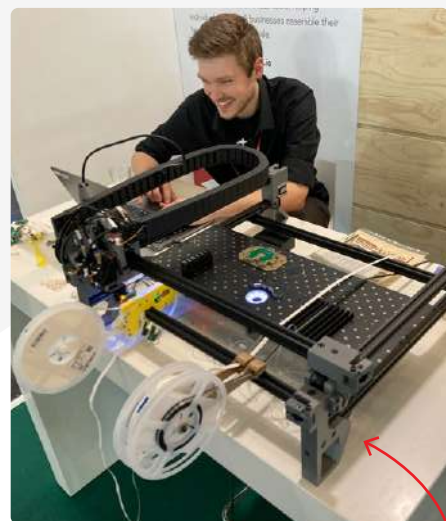
1st Prize

Walking away with a €75,000 media budget from Elektor was wheel.me, who've made

great strides in eliminating industrial drudgery, with applications from logistics and warehousing to hospitals [1]. Their wheels are not only powerful, taking a big load off wherever it's needed, but they're connected, sensed-up, and autonomous, too. We look forward to seeing them grow further and how they make an impact on industry!

2nd Prize

Stephen Hawes was infectious all week long, and his Opulo open-source pick-and-place machine, which he put together step by step, all painstakingly documented and the whole process vlogged on YouTube, took 2nd prize, meaning that Opulo gets the opportunity to brag about their offering with €50,000 worth of marketing to use from the Elektor Media Kit [2].



Opulo's Stephen Hawes sets up their open-source pick-and-place machine.



Second-prize winner Opulo's Stephen Hawes.

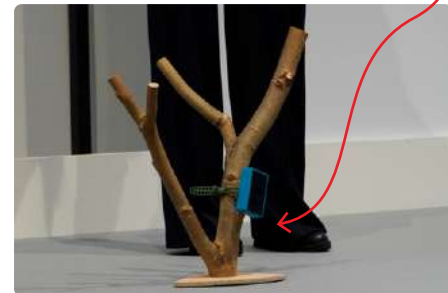
3rd Prize

Treesense was the environmental favorite of the pitches, bringing electronic innovation in the form of embedded sensors to take the pulse of trees and let the humans in charge know just what they need [3]. With their €25,000 Elektor media budget, Treesense gets to shine some light on a topic that doesn't get as much attention as it deserves: looking after our planet's trees.

That should be it. After all, we never spoke about a 4th prize. But, with a passionate, deadlocked late-night jury, we could not move on without recognizing a contender about whom we just had to admit when considering the potential: "There's definitely something there."

Encouragement Award

AMSEL won a €10,000 media budget to get their incredible eBlaze technology off the ground. Yes, we were forced to invent another prize this year because this cutting-edge surface pre-treatment technology,



Thanks to Treesense sensors, trees can tell us what they need.



Encouragement Award-winner AMSEL's Dr. Salman Asad.

although still in its infancy, could not go unnoticed. Watch this space, as we invited their CEO, Dr. Salman Asad [4], back for Fast Forward 2024, where he gets to prove that there really was something there.

Congratulations to all the winners. We will be watching your progress with great interest over the next couple of years! ◀

220651-01

WEB LINKS

- [1] wheel.me: <https://wheel.me>
- [2] Opulo: <https://opulo.io>
- [3] Treesense: <https://treesense.net>
- [4] AMSEL's Salman Asad on LinkedIn: <https://linkedin.com/in/syed-salman-asad-86337216>

The Tube

An Unusual Tube Amplifier

By Gerd Reime (Germany)

The key features of this special amplifier are good performance and an impressive visual appearance. This project combines craftsmanship in construction with audiophile circuitry and, thanks to tube technology, it links nostalgic elements to a modern appearance.

I built my first tube amplifier in 1971, when I was just 15 years old. It was built around two EL84 tubes operating in push-pull mode, using a circuit I found in Elektor. The rest of the components were cannibalized from old radios. A short while later, I started an apprenticeship as a radio and television technician at Grundig. Later I worked in R&D. When I retired, I wanted to work on a special project. What I had in mind was a nice, small power amplifier with around 10 to 15 W output power — using two EL84s, just like before, but better.

At first, I only wanted to have a special enclosure, but then I became fascinated by circuit technology and tried, with many test setups and measurements, to put together something as good as possible. What started out as a fun project turned into a serious endeavor. If you want to draw inspiration from my ideas, the following description forms a good introduction. The full construction instructions are rather elaborate and actually too large for an Elektor article. For all the details, you can consult the comprehensive PDF file available for download at [1].

Preliminary Considerations

Many audio enthusiasts swear by tube amplifiers. Why? The allegedly warm sound and mild response to overdrive are often cited. Nearly all circuits I am aware of operate with negative feedback, but what are the effects of this sort of feedback?

The basic task of an audio amplifier is to process the weak signal from an audio source so it can be reproduced by a loudspeaker. For this, both the voltage and the current must be amplified so that the coil and cone of the (commonly used) dynamic loudspeaker can be driven with enough excursion to make music or other audio signals loud enough to be heard. This is easy with solid-state amplifiers, but tube amplifiers operate with relatively high voltages and correspondingly low currents, which cannot be applied directly to the loudspeaker. An output transformer is usually necessary for impedance matching. Many generations of hi-fi enthusiasts have devoted their attention to the question of what is the best transformer.

Another aspect is that the operating range of tubes is not as linear as would be liked. Negative feedback is used to reduce the resulting distortion of the audio signal. Part of the signal is fed back in reverse phase from the output to the input, and the generally high, 'surplus' open loop gain (relative to the actual gain of the amplifier) reduces the nonlinearities that would otherwise occur. In **Figure 1a**, the negative feedback is provided by the voltage divider in the red rectangle. All nonlinearities and distortion signals

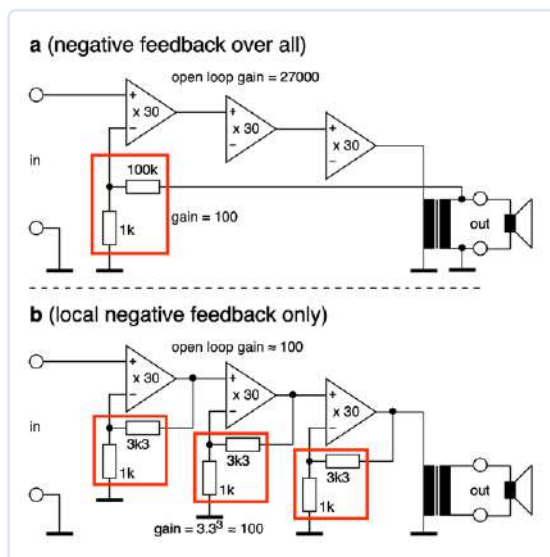


Figure 1: Different approaches to negative feedback: (a) conventional overall negative feedback and (b) local negative feedback, individually at each amplifier stage.



generated in the amplifier are reduced by a factor equal to the ratio of the open loop gain to the gain set by the negative feedback circuit. In the example in Figure 1a, this factor is $27,000 / 100 = 270$. As the negative feedback supposedly reduces all distortion by this factor, shortcomings in the design have less effect.

An alternative to strong overall negative feedback (from the loudspeaker output to the input of the first stage of the amplifier) is to limit the gain of each stage of the amplifier by local negative feedback (in Figure 1b, this is provided by the voltage dividers in the three red rectangles, functionally corresponding to the anode and cathode resistors of the tubes). In this way, the individual amplifier stages are linearized, resulting in a fairly linear, low-distortion amplifier that does not need overall negative feedback. The question is whether this makes a difference and whether it actually sounds different.

Compared to a solid-state amplifier, the most striking difference is that there is an output transformer ahead of the loudspeaker. In the circuit in Figure 1a, the transformer is inside the feedback loop, but in Figure 1b it is not.

It can be expected that the design according to Figure 1a will have a fairly straight frequency characteristic in terms of voltage. My measurements on this sort of amplifier, shown in Figure 2a, confirm this. The voltage characteristic is very flat, but the current characteristic has a sharp dip around 1 kHz because the impedance of the connected loudspeaker is the highest at this point. Consequently, a power dip can also be seen at 1 kHz. From Figure 2b, you can see that due to the higher load impedance at 1 kHz, the voltage rises and the current dip is not as strong. This means the influence of the loudspeaker impedance on the output power is partially compensated, which should result in a different (better) sound. Something similar occurs in the bass region at the resonance frequency of the woofer around 60 Hz (Figure 3). Here as well, the influence of the loudspeaker impedance is reduced if the output transformer is not in the negative feedback loop. I repeated these measurements with a variety of loudspeakers, and they generally show a similar picture.

The conclusion is that a tube amplifier without negative feedback from the loudspeaker to one of the earlier

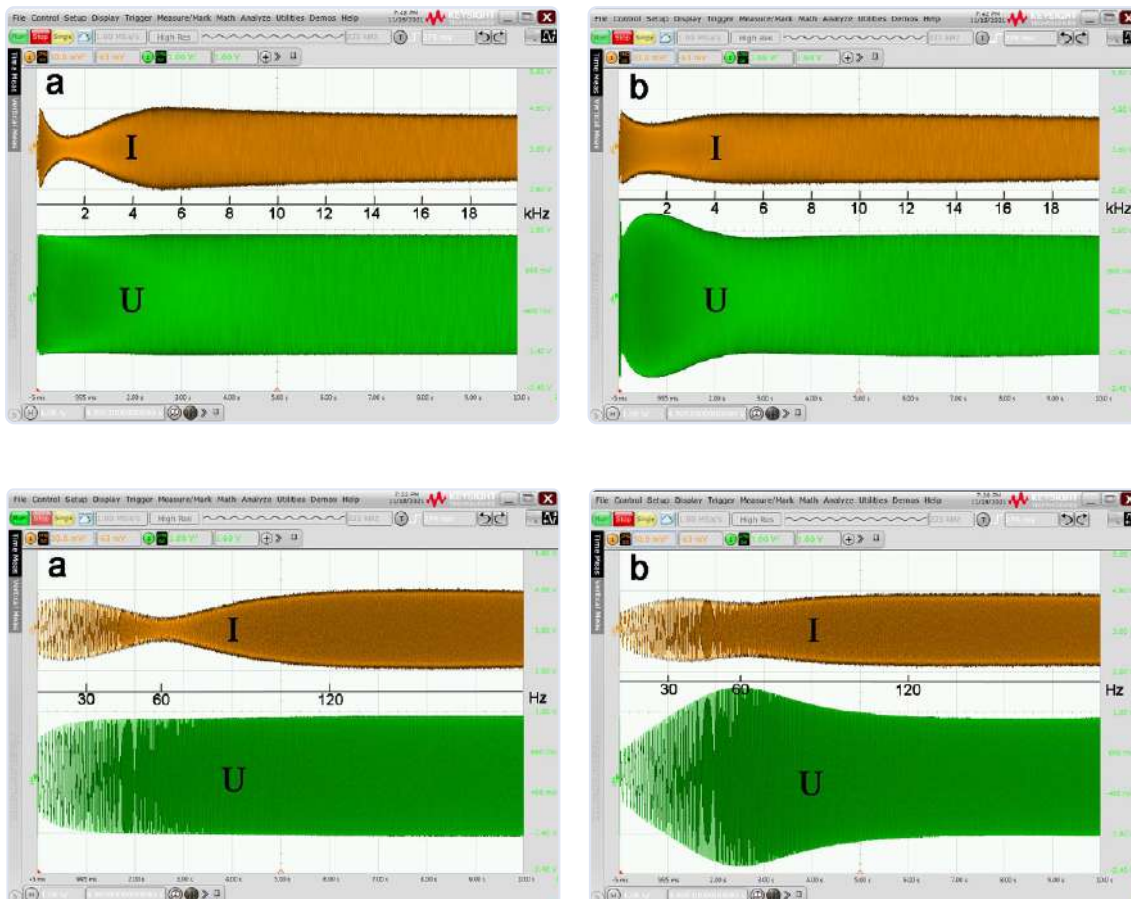
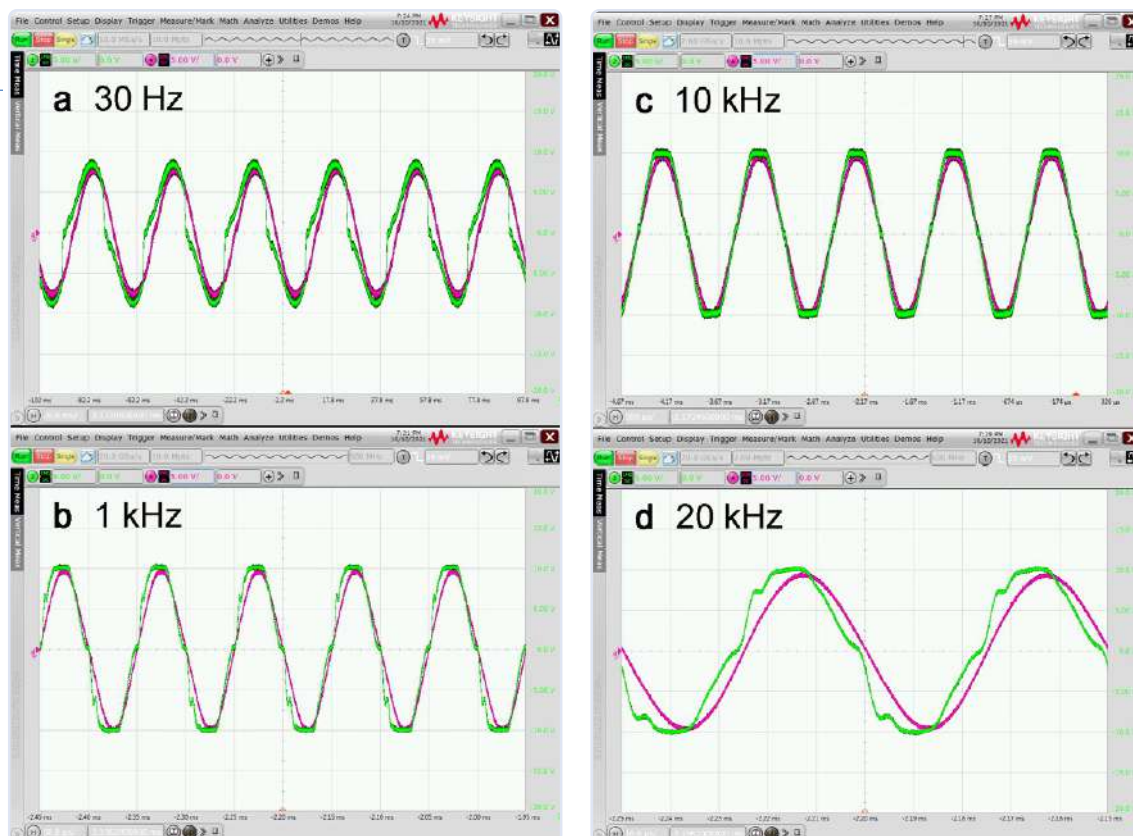


Figure 2: Current (ochre) and voltage (green) over the frequency range (20 Hz to 20 kHz): (a) with conventional negative feedback and (b) with local negative feedback.

Figure 3: Current (ochre) and voltage (green) in the bass region: (a) with conventional negative feedback and (b) with local negative feedback.

Figure 4: Overdrive behavior at various frequencies (green = conventional negative feedback; red = local negative feedback).



stages (version b) has significantly different behavior than a conventional amplifier (version a). Of course, I didn't stop with simple measurements of the two amplifier designs, but also asked colleagues to participate in listening tests. And the results were that version b, without overall negative feedback, always came out on top in these comparisons. The sound was experienced as significantly warmer and more balanced. Depending on the music genre concerned, the presence was experienced as more intense. Both results are not surprising, considering the different measurement curves.

Yet another difference is that at low-power levels or volumes, a push-pull A/B output stage operates in Class A and therefore naturally has less distortion. For this reason, the harmonic distortion with a good output transformer should be nearly the same at low volume, with or without negative feedback. Surprisingly, the design without overall negative feedback had distinctly lower harmonic content, particularly in Class A. By the way, in my opinion, the best output transformers are those from Menno van der Veen [2].

Clipping and Saturation

Just about everybody knows that tube amplifiers exhibit softer limiting at high volume levels, when the peak values of the signal approach the supply voltages. Usually, the output tubes limit the amplitude, together with the output transformer, which goes into saturation at that point. Particularly in this situation, overall negative feedback becomes problematic because when clipping occurs the feedback causes the non-limiting upstream stages

to drive the output stage even harder with the correction signal, leading directly to a strong rise in harmonic distortion. Without this form of negative feedback, the harmonic distortion behavior is significantly gentler in case of overdrive.

Particularly at low frequencies in the range of 20 to 30 Hz, output transformers are often unable to transfer large amounts of power. As a consequence, the adverse effects of negative feedback become audible in the deep base region even at low power levels. On top of this, there are the effects of the phase characteristic of the transformer, which also produce undesirable sound effects due to negative feedback.

My measurements with 10% overdrive at different frequencies support these considerations. The red curves of the amplifier without overall negative feedback in **Figure 4** look consistently better than the more or less distorted green signals of the amplifier with full negative feedback. The differences at 20 kHz are especially strong, but relevant degradations with a conventional amplifier design can also be seen at 30 Hz. Interestingly enough, the 10% overdrive at this frequency accounts for 85% of the nominal signal level. In summary, it can rightfully be asserted that truly soft clipping with almost no disturbing effect is only possible without overall negative feedback.

Aurally Compensated Loudness Control

You often read that a linear response is the only right one. According to this, top-end amplifiers must have a flat frequency response. By contrast, I suspect that the

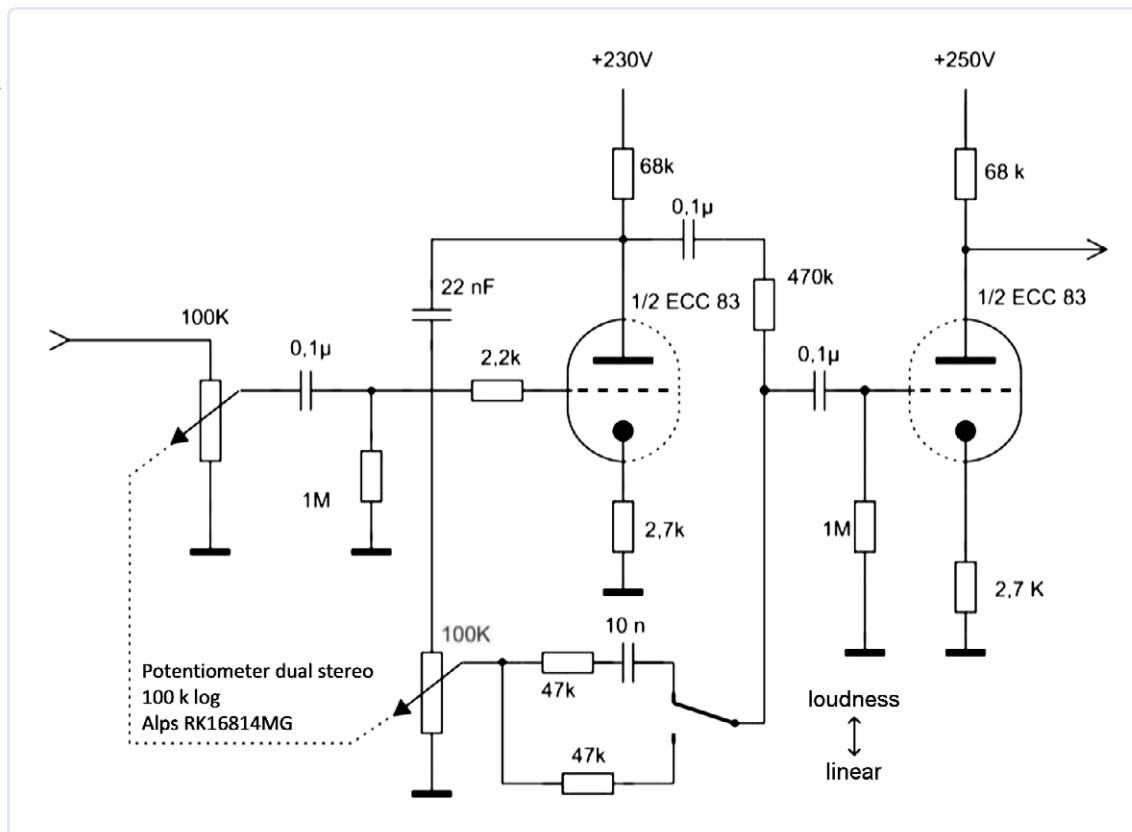


Figure 5: Circuit for aurally compensated loudness control using an Alps quad potentiometer.

requirement for a flat frequency response is only based on the fact that potentiometers with taps are no longer available now. The equivalent circuits that can be found online only work so-so. What's behind this is that human hearing is less sensitive to high and low frequencies at low-volume levels, or (depending on how you look at it) our ears prefer the midrange frequencies.

In the past, aurally compensated loudness controls (also called physiological loudness controls) were the usual standard. They emphasized the high and low frequencies at low volume levels, with this emphasis decreasing with increasing loudness. But how many modern amplifiers offer this function? Hardly any.

Instead of using vintage potentiometers with two or three taps, which were anyhow not ideal due to the steps, this problem can be solved in a different and more elegant manner. You use a dual stereo potentiometer, which in fact is a quad potentiometer, such as is still available from Alps, among others. With this and a simple circuit, you can build an excellent loudness control with aural compensation. In addition, you can use a simple switch to change between aurally compensated and linear, without any change to the amplifier.

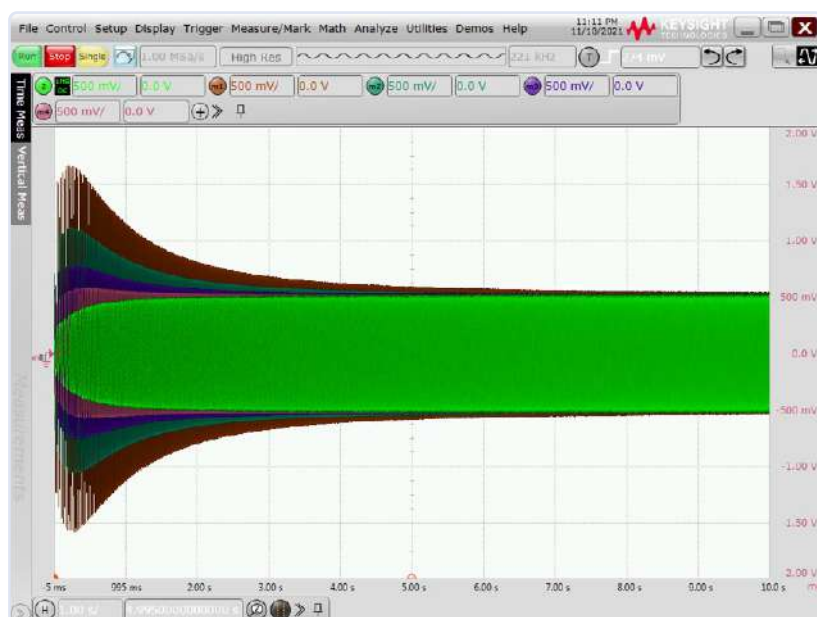
Figure 5 shows the circuit I built using the Alps RK16814MG potentiometer. Although this is a motor-driven potentiometer, the motor can be removed if needed so the potentiometer can be operated manually. To clarify the effect of aurally compensated loudness control, I measured the frequency responses at various settings,

normalized them to the same signal level at 1 kHz, and superimposed the curves. The effect in the bass range can be seen in **Figure 6**. As can clearly be seen, the frequency response correction gradually decreases with increasing loudness.

Driving the Output Tubes

As is well known, push-pull stages need signals with a 180° phase shift. Compared to simple circuits with just one tube, you can achieve better quality with a bit more complexity.

Figure 6: Effect of aurally compensated loudness control: **brown** = very quiet; **dark green** = quiet (background music); **blue** = medium loudness; **red** = rather loud (linear frequency response); **green** = full loudness just before overdrive.



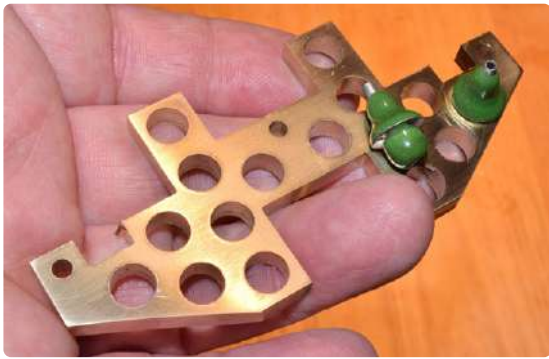


Figure 11: Entirely fitting for tube technology: a 'circuit board' made of solid brass.

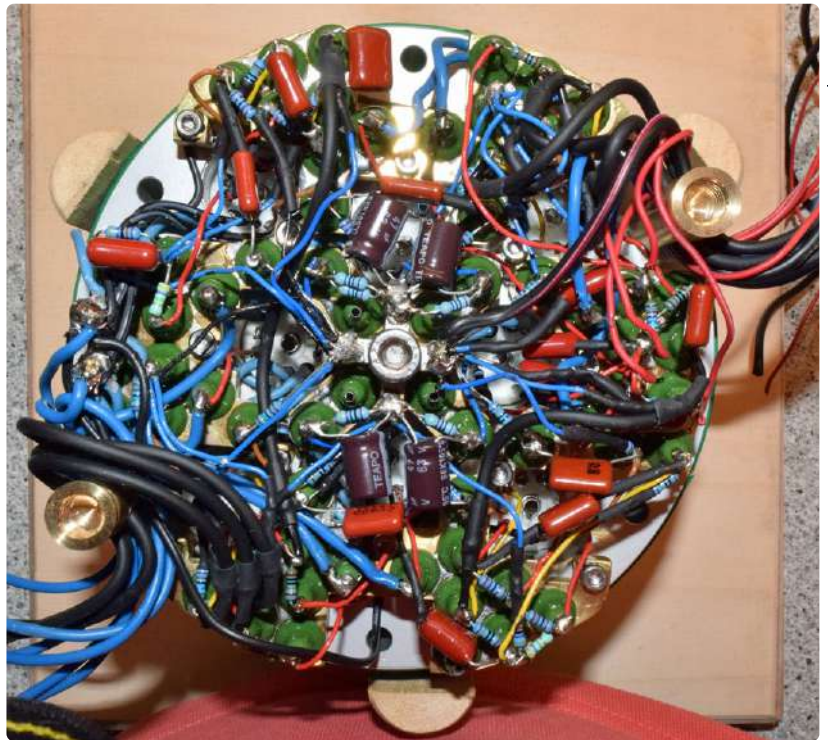


Figure 12: Solder posts with nearly vintage green glass insulators.

quotes for two reasons: firstly because a circuit board is not a common feature in a tube amplifier, and secondly because it is composed of component holders (**Figure 11**) milled from solid brass plates, with no epoxy in sight. After all, the internals of this sort of tube amplifier should also be pretty, shouldn't they? **Figure 12** shows the bottom of the aluminum plate with the finished brass plate and the attractive solder posts with green glass insulators.



Figure 14: The bulky electrolytic capacitors need their own brass plate.



▲

In fully assembled form, this plate looks like **Figure 13**. Another 'circuit board' holds relatively large components, such as electrolytic capacitors (**Figure 14**). Supporting brass plates were also made for the mains transformer, which is circular as befits a toroidal transformer, and can be admired in **Figure 15**. Of course, that's not the full story on the mechanical side. Logically enough, the front panel is an aluminum ring on which the input selector

Figure 13: Bottom view (solder side) of the fully assembled circuit board. A bit of a rat's nest, but perfectly stable.



Figure 15: A round transformer for a round amplifier. Here, as well, additional mechanical parts are required for fixing.

◀

Figure 16: The user interface components (selector switch, power switch and loudness potentiometer) are mounted on their own aluminum ring.



Figure 17: Provisional assembly of the complete amplifier, without the protective tubular enclosure.

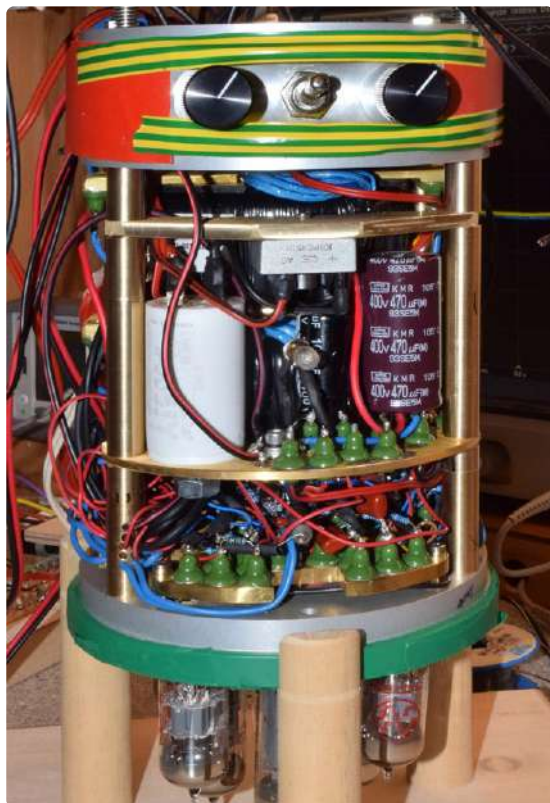


Figure 18: The finished amplifier in its full glory. A true gem — more than just living room compatible.

switch, power switch and volume control potentiometer are mounted (Figure 16). The provisionally assembled stack shown Figure 17 gives a good impression of this arrangement. And now — drum roll! — the finished amplifier in its enclosure (Figure 18).

With suitable attention to detail, a tube amplifier like this is always an especially aesthetic bit of technology. But isn't there something important missing? As with any amplifier, The Tube needs connectors where weak audio signals enter and connectors where powerful signals exit



Figure 19: A recess at the rear holds the input and output connectors and the mains power connector.

to the loudspeakers. And somehow, mains voltage has to reach the installed transformer. The solution to this problem is shown Figure 19. This recess in the tubular enclosure of the tube amplifier holds the gold-plated RCA / Cinch sockets for the inputs, the XLR sockets for the loudspeakers, and an appliance power connector for the mains cable.

Feeling Inspired?

As can be seen from my project, for a tube amplifier a good circuit is only half the story. To paraphrase Thomas A. Edison, along with the inspiration (electronics) it needs a lot of perspiration (mechanics). If you want to draw inspiration from my project, you need to be prepared to put a lot of work into the construction in order to build a tube amplifier that not only sounds good, but also looks great. The old saying, "no pain, no gain," is especially true when it comes to tube technology. ◀

220089-01

About the Author

After completing an apprenticeship as a radio and television technician, Gerd Reime worked successively in the development departments of Grundig and Nokia. He holds several patents.

Questions or Comments?

Do you have any technical questions or comments about this article? Contact the Elektor editorial team at editor@elektor.com.



Related Products

➤ **Elektor Audio Collection (USB Stick)**
(SKU 19892)
<https://www.elektor.com/19892>

➤ **R. A. Honeycutt, *The State of Hollow State Audio* (Elektor 2020)**
www.elektor.com/19170

WEB LINKS

[1] Comprehensive project description (PDF): <https://www.elektormagazine.de/220089-01>

[2] Van der Veen transformers: <https://www.mennovanderveen.nl>

Biomaterial in Electronics: Ready or Not



By Priscilla Haring-Kuipers (The Netherlands)

A lot of the damage from electronics comes from the materials used. Are we ready to shift to more eco-friendly biomaterials in electronics manufacturing?

Recycling

Sustainability in electronics usually refers to recycling, and there is indeed a lot on that end that needs to be done. Not even 20% of the electronics in the world are being recycled, and of that number most are handled by untransparent electronic waste recyclers or even urban miners and scrappers.[1] If we look at the other end of making electronics, we can start out doing

better. By making the base material different, we can also change the process of recycling. Biomaterials can be designed with circularity in mind. Clean, non-toxic and easier to process.

The Problem with FR4

Almost all electronics are built onto PCBs made out of FR4 and copper foil. FR4 stands for Flame Retardant class 4, and it is made by dipping fiberglass cloth in an epoxy resin. Epoxy resin is a thermoset polymer with the “epoxy ring” found somewhere in the molecular structure.[2] Epoxy resin tends to come from petrochemical origins and has a large CO₂ footprint (5.7–7.6 kg CO₂ per kg).[3] The chemical processing of epoxy is also a known cause of occupational asthma, which can cause permanent damage to the lungs. The high viscosity of resin at room temperature further requires the adding of petrochemical dilutants.

Epoxy resin is a thermoset plastic that is impossible to break back down into its chemical parts and cleanly recycle the building blocks, just as a baked cake cannot be turned back into flour, eggs and butter. The chemical reaction that gives this material fantastic mechanical, dialectic, flame retardant as well as heat and moisture resistant properties also makes it very hard to recycle. “Sustainable circularity will require more research to discover high-performance biobased polymers for replacing synthetic polymers in PCBs.”[4]

The making, processing and disposing of FR4 creates a lot of problems that could be solved by using other materials. One of the ingredients in FR4 used to be coal tar but we’ve developed into other fossil fuel derivatives. Seems like we are ready to develop further into other ingredients.

A few projects have used paper, flax and bio-based resin matrices while using less of the toxic flame retardants or using soy and hemp oil-based replacements. Durability is an issue with these bio-boards as — unlike FR4 — they do interact with moisture.

More Bio

PLA, or Polylactic Acid, is the most famous of bioplastics that is entirely synthesized from renewables. Unfortunately, it does not do well under heat stress and the processes of making electronics often puts high temperature demands on its material (vapor deposition of metals, reflow ovens or wave soldering). Additional chemical processing of the PLA is being developed to make it more heat resistant. As a chemist recently told me: “We can grow all kinds of bio-plastics now. You can just state your requirements.”

Carbon-based conductive material is more thermally resistant and has better electrical conductivity. There are a lot of potentially relevant conductive carbons that can all be synthesized from renewables, replacing the use of various metals in electronics. Carbon nanotubes can be grown from virtually any organic compound and graphene can be made by exfoliating graphite. Several universities in the USA are working on bio-based graphene ink to print electronics “manufacturing tools that currently live only in the laboratory or in the imagination” and the bio-based substrates to print them on.



Source: Michael Schiffer, Unsplash

Several Finnish universities are working on making flexible piezoelectric sensors from wood fibres and bacteria as well as making supercapacitors out of dandelion. They are developing “new green materials” such as biosensors, electrodes, circuit and component boards all based on cellulose and other bio-based materials.

The definition of bio-based is also developing. According to the American Food and Drug Administration (FDA), “bioplastics” now also includes products made with petroleum-based polymers and blended with natural fibres. Such materials would not be biodegradable and this may not be what we have in mind when we think of bioplastic.

Long Last

The problems of petrochemical plastics and biomaterials for electronics are different sides of the same thing: extremely durable versus biodegradable. Petrochemical materials used in electronics are not affected by natural aging agents such as moisture and temperature. They are so durable that they do not degrade at all and form mountains of e-waste, which we are not recycling properly. Biomaterials degrade over time, which sounds great when a device is at the end-of-life, but sounds less great when you are still designing the device.

Most of our electronics we actually want to degrade. Some things are one-time-use-only electronics such as RFID tags, sensors and medical kits. Most other devices we do not intend to use for decades — or even one decade. Perhaps instead of unlimited durability as the default in manufacturing, biodegradable electronics should be the default. Should you *really need* to manufacture using plastics or resins that stay around forever, you should also present a plan on how you are going to support your precious electronic device over the next 50 years at least.

Responsible stewardship of materials is underway and is encapsulated in policies like the EU/USA Green

Deals and incoming Climate Law. There are some fantastic developments in material science that look to “accelerate the realization of a bio-based economy” and are relevant to the making of electronics. But none that I’ve found are available at scale yet. Most research projects seem poised to enter the chemical and/or manufacturing market in the next five years. I am expecting big bio-things. ◀

220556-01



Source: Fukayamamo, UnSplash

WORLD ETHICAL ELECTRONICS FORUM

World Ethical Electronics Forum 2023

In November of 2021, Elektor launched the **World Ethical Electronics Forum** (WEEF) in Munich, Germany. The event inspired global innovators in electronics with an open discussion about ethics and sustainable development goals (SDGs). In addition to Elektor engineers and editors, the list of speakers and panelists included Dr. Stefan Heineemann (Professor of Business Ethics at the FOM University of Applied Sciences), Dr. Paula Palade (PhD, Jaguar Land Rover), Margot Cooijmans (Director, Philips Foundation), and several other thought leaders, including Priscilla Haring-Kuipers. Visit the WEEF webpage to stay informed and to get involved: www.worldethicalelectronicsforum.com.

WEB LINKS

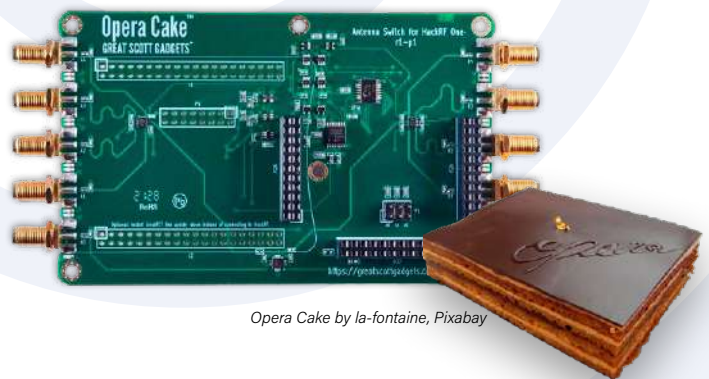
- [1] eWaste Ben: <https://www.youtube.com/c/eWasteBen>
- [2] Custom Materials, Inc., “Is Epoxy Resin Plastic?” January 2, 2022: <https://www.youtube.com/watch?v=i8nEQQ9S0V0>
- [3] NIH, “Bioplastics and Carbon-Based Sustainable Materials, Components, and Devices: Toward Green Electronics,” October 20, 2021: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8532127/>
- [4] O. Ogunseitan, et al, “Biobased materials for sustainable printed circuit boards,” Nature Reviews, September 12, 2022: <https://www.nature.com/articles/s41578-022-00485-2#citeas>

Opera Cake Antenna Switch for HackRF One

Connect Up to Eight Antennas To Your SDR

By Clemens Valens (Elektor)

The Opera Cake from Great Scott Gadgets is an antenna switch intended for the HackRF One SDR. With it, you can connect up to eight antennas to your system or insert a switchable filter bank.



Opera Cake by la-fontaine, Pixabay

Following the tradition of electronic gadgets named after cakes, pastries and sweets (think Raspberry and Banana Pi, Snickerdoodle [1], and early versions of Android), the Opera Cake antenna switch is a device that bears no relation whatsoever to its name (at least Pi referred to Python).

The Opera cake that I know (and even made a few times) is a multi-layer cake that mostly tastes like coffee. And maybe that's the relation to the device reviewed below? You'll need a lot of coffee to stay awake during the long hours of SDR fun that it will procure you.

The Opera Cake Antenna Switch...

As mentioned above, the Opera Cake is an antenna switch for HackRF One [2] as shown in **Figure 1**. Actually, it is a double four-way antenna switch, i.e. a pair of single-pole four-throw (SP4T) switches. You can use the two switches — also known as banks — in parallel as a DP4T switch or in series (sort of) as an SP8T switch. The switches are bidirectional, so they work in both RX and TX mode.

Let's start by describing what Great Scott Gadgets' take on Opera cake looks like. Basically, it is a 120 mm by 75 mm four-layer circuit board (wait a minute, four layers, layered cake, hmm...) with a lot of connectors. There are five female SMA connectors on either end of the board, and three stackable headers mounted on it, with footprints for three more.

... is an Add-On Board

The board has surprisingly few electronic parts. This is because the Opera Cake is an add-on board. It does not require any intelligence, as the board it plugs onto controls it. Size-wise, it is intended for plugging on top of an HackRF One board. But when you solder the right connectors on the board, you can also plug a GreatFET One [3] on it (upside down).

Note that when you have a boxed HackRF One, you will have to unbox it (with care!) before you can plug the Opera Cake on it.

Configurable

The first application of the antenna switch that comes to mind is, of course, connecting a variety of antennas to a single HackRF One so that you don't have to rewire your setup every time you want to change RF band.

Switching antennas can be done manually with a little software utility, but it can also be done automatically by the HackRF One firmware based on frequency or time. Frequency-based switching enables e.g. wideband spectrum analyzer applications, whereas time-based switching lets you do cool things like pseudo-Doppler direction finding. [4]

When using the switch in DP4T mode, you can use it to insert, for instance, attenuators or filters in the antenna path. This way, the Opera Cake can function as a switched filter bank.

Stackable

Also know that up to eight Opera Cakes can be stacked on top of each other (a layered SDR cake...), enabling different switch configurations. In manual mode each board can be controlled separately, but in automatic frequency and time switching modes all boards will switch at the same time and in the same way. A stack of two boards would allow for a DP8T configuration, switching, for example, eight filters.

Why Would I Want One?

Opera Cake is not a new design; it dates back to 2016 and maybe even longer. What's new about it is that you can now buy it as a fully assembled and tested module. Up to now, if you wanted one, you had to build it yourself from the design files published on GitHub. [5]

The nicely assembled Opera Cake antenna switch is, unsurprisingly, a perfect companion for HackRF One. Instead of obliging you to rewire your hardware or adjust the length of your antenna every time you want

to work in a different band, it lets you connect (and keep connected) up to eight of your favorite antennas at once.

A Few Remarks

Depending on the age of your HackRF One, you may have to upgrade its firmware to the latest version to make it work with the antenna switch. Instructions on how to do this are available on the Great Scott Gadgets website [6].

Please be aware that, as on HackRF One, the SMA connectors on the Opera Cake are female types. This means that it will only work with antennas equipped with a male connector (i.e., with a center pin, see **Figure 2**). Therefore, be careful and choose wisely.

A HackRF One with one or more Opera Cake boards stacked on top of it no longer fits in its (tight) enclosure. This, of course, makes the system vulnerable to dust and screwdrivers and other metal objects accidentally falling in- and onto it. Also, depending on how you wire your antennas and filters, the *Reset* and *Power* pushbuttons may become somewhat inaccessible (**Figure 3**). As the stackable headers P20, P22, and P28 only carry power and a few digital signals, and no RF signals, it is possible to insert headers to increase the distance between the boards. Finally, a (short) male-male SMA cable to connect the Opera Cake to HackRF One's antenna input is unfortunately not included, so bring your own. ◀

220602-01

Questions or Comments?

Do you have technical questions or comments about this article? Contact Clemens Valens at clemens.valens@elektor.com.



Figure 1: The Opera Cake from Great Scott Gadgets is an antenna switch for HackRF One.



Figure 2: Antennas with SMA connectors. Left is right and right is wrong for Opera Cake and HackRF One.

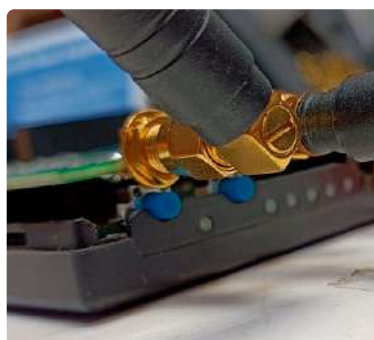


Figure 3: You may want to insert stackable headers to increase the space between the two boards.



Related Products

- ▶ **Opera Cake - Antenna Switch for HackRF One** (SKU 20083) www.elektor.com/20083
- ▶ **HackRF One Software Defined Radio (1 MHz to 6 GHz)** (SKU 18306) www.elektor.com/18306
- ▶ **ANT500 Telescopic Antenna (75 MHz to 1 GHz)** (SKU 18481) www.elektor.com/18481
- ▶ **ANT700 Telescopic Antenna (300 MHz to 1100 MHz)** (SKU 18480) www.elektor.com/18480



WEB LINKS

- [1] Snickerdoodle on Elektor Labs: <https://www.elektormagazine.com/labs/snickerdoodles-with-zynq>
- [2] D. Meyer, "HackRF One SDR Transceiver: From 1 MHz to 6 GHz," [Elektormagazine.com: https://www.elektormagazine.com/news/hack-rf-one-sdr-transceiver](https://www.elektormagazine.com/news/hack-rf-one-sdr-transceiver)
- [3] C. Valens, "Introducing the GreatFET One Dual-Core Microcontroller Board for Python," [Elektormagazine.com: https://www.elektormagazine.com/news/greatfet-one](https://www.elektormagazine.com/news/greatfet-one)
- [4] Pseudo-Doppler direction finding: <https://www.rtl-sdr.com/pseudo-doppler-direction-finding-with-a-hackrf-and-opera-cake/>
- [5] Opera Cake on GitHub: <https://github.com/greatscottgadgets/hackrf/tree/master/hardware/operacake>
- [6] HackRF One firmware: <https://github.com/greatscottgadgets/hackrf/releases/latest>

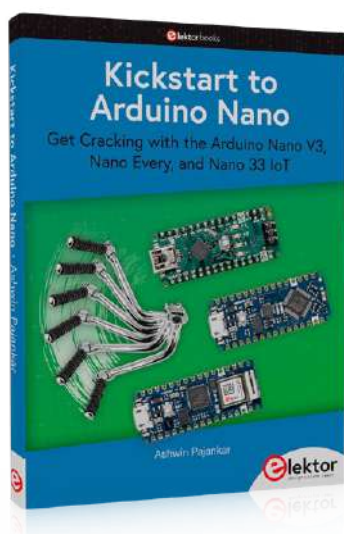


Engineering with **Arduino** and More

An Interview with Author Ashwin Pajankar

By Alina Neacsu (Elektor)

The Elektor community is full of engineers who enjoy sharing their knowledge with like-minded electronics enthusiasts. Take Ashwin Pajankar, an engineer, educator, Elektor author, and YouTuber working from Nashik, India. When he isn't tackling a new electronics project at his workbench, you will find him helping his peers — through books, courses, and videos — learn to engineer with Arduino and other cutting-edge tech.



Read Kickstart to Arduino Nano to start your journey with the Arduino Nano V3, Nano Every, and Nano 33 IoT.

Alina Neacsu: First of all, thank you for taking the time to answer these questions. Tell us about yourself. What is your current occupation?

Ashwin Pajankar: I live in the outskirts of the city of Nashik in India. I have earned a Bachelor of Engineering in Computer Science and Engineering from Shri Guru Gobind Singhji Institute of Engineering and Technology, Nanded. I also earned Master of Technology (again, in Computer Science and Engineering) from International Institute of Information Technology, Hyderabad. I speak five languages and have lived and worked in three states in India.

I currently work as a freelance technical writer, a YouTuber, and an instructor at Udemy. I also organize boot camps for programming and electronics for working professionals and students. I love math, physics, computers, and electronics.

Alina: Do you remember your first electronics project with microcontrollers? Can you describe it to us and what technologies were used?

Ashwin: My very first microcontroller project was a very humble one. I got started with microcontrollers (along with the 8085 and 8086 microcontrollers) in 12th grade. I was taught the 8051 microcontroller. (It is the standard microcontroller used across India to teach the topic.) However, we did not have any hands-on with that in our 12th grade. When I studied engineering (a Bachelor's level four-year undergraduate degree), I had an opportunity to work with 8051 kit using assembly language during the third year of the course. My very first project with the 8051 was a very humble one — blinking an LED using 8051 assembly code.

Alina: The Arduino ecosystem — can you tell us how this works? And what are the advantages and disadvantages?

Ashwin: Arduino is an open-source hardware and software ecosystem. Since the design is open source, people and organizations can manufacture their own boards and add ideas. The software is also open-source, and the source code can be found at github.com/arduino. People can contribute to this main branch or fork the project to have their own custom flavor. There are a lot of learning resources available on the internet. One can get started with electronics and programming with



C at the same time using Arduino without spending an astronomical amount of money. These are the advantages of Arduino ecosystem. The only disadvantage I can think of is that sometimes counterfeit products branded as original Arduino products are sold to unsuspecting enthusiasts.

Alina: What sparked your interest in the Arduino Nano 33 IoT board?

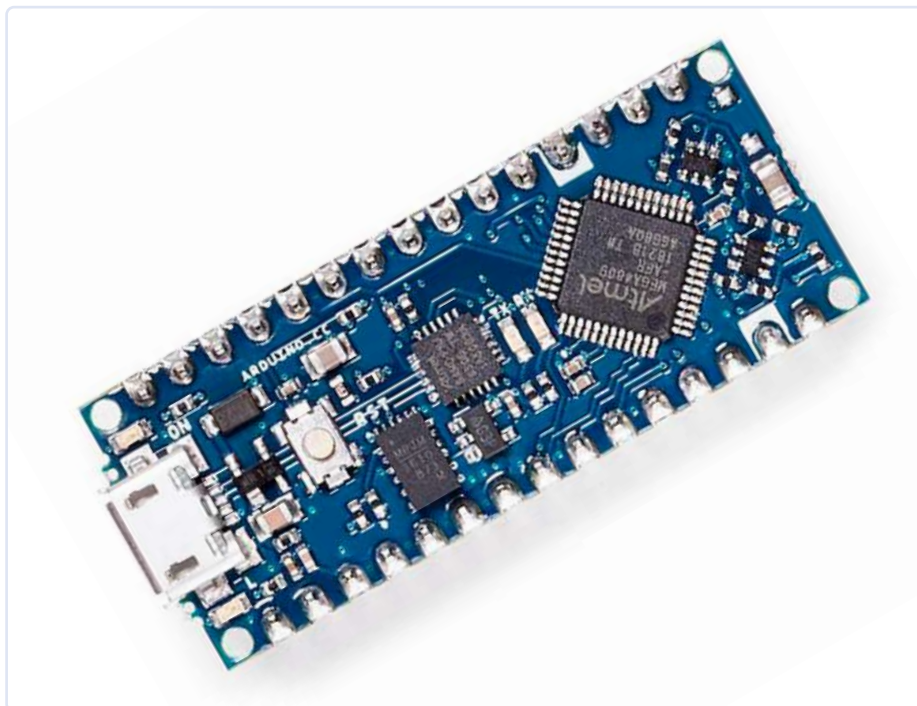
Ashwin: Well, earlier, when we had to connect any Arduino board to Wi-Fi, we had to use specialized shields (which are a bit difficult to get in India) or use the ESP-01. Working with the ESP-01 is a bit complicated, and beginners often find it overwhelming. Arduino Nano 33 IoT is a perfect IoT solution out of the box. It has a NINA-W102 Wi-Fi module embedded in it. I regularly check arduino.cc for new products. I was very excited to see the new board with built-in Wi-Fi. So, when it became available in India, I purchased it at the very first opportunity and experimented with it.

Alina: Do you have any advice for anyone interested in getting started with the Arduino Nano?

Ashwin: Yes. Check the online documentation at the Arduino homepage. My book, *Kickstart to Arduino Nano* (Elektor, 2022), has detailed and step-by-step instructions.

Alina: Tell us about the process of writing your first book. Was it challenging?

Ashwin: I was working as an IT Engineer in Bangalore, India when I wrote and published my very first book. The book was on Raspberry Pi. It focused on computer vision. While it was my very first experience of publishing a book, I did not find it very challenging as I love programming, electronics, and writing. Authoring books allows me to combine all of my passions into something very tangible.



Ashwin Pandakar provides a comprehensive kickstart to the Arduino Nano in his book.

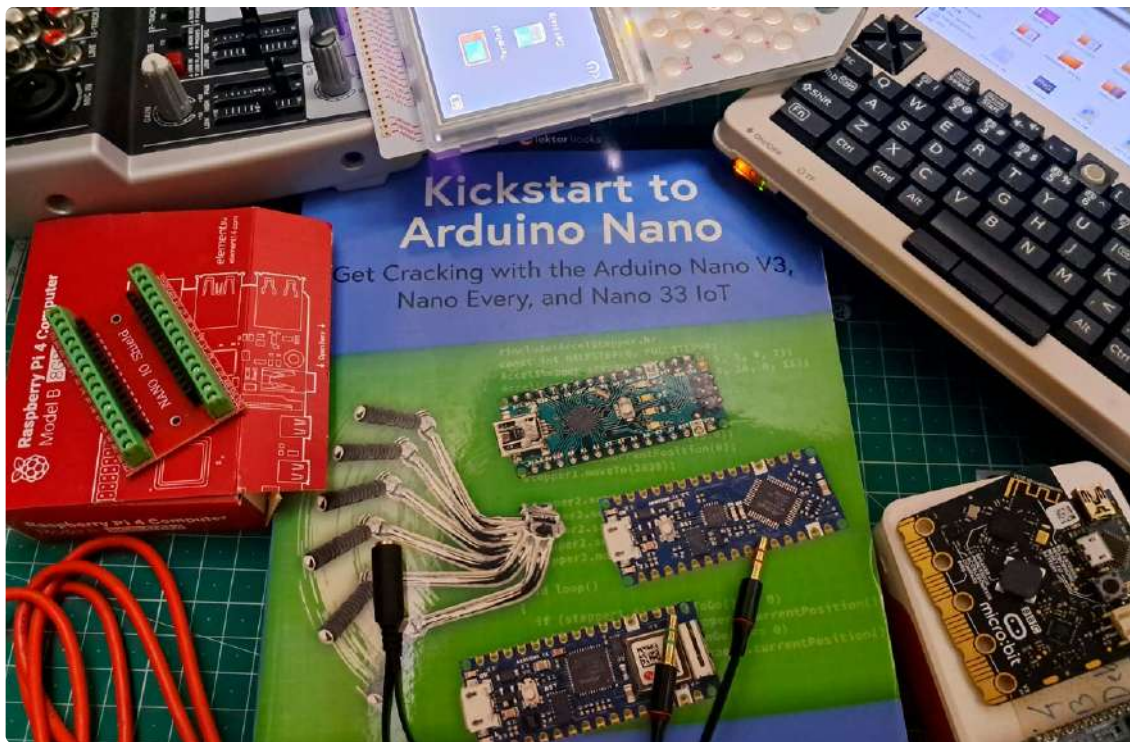
Alina: You mention in your book you were in India, where you currently reside, while writing. How did the book come to life during the COVID-19 pandemic?

Ashwin: The COVID-19 pandemic was a testing time for our entire human civilization. India was one of the worst affected nations, with one of the highest hospitalization rates and highest death tolls. Briefly, Nashik, my current city of residence, had the highest death toll in India. It was very unsettling as no one had ever experienced such catastrophe before. There were constant lockdowns, curfews, and restrictions. There was always a shortage of food, medicines, and other supplies. There were no beds available in hospitals. Ambulance and hearse services were overstretched. Even I contracted the disease twice. The first time, I was admitted to a government-run hospital and the second, I was treated at home. I distinctly remember running around to find anti-flu medications for myself and my neighbors, as the medicine was very short in supply.

Writing the book gave me purpose during these dark times. Due to isolation and the near halt of almost all the other aspects of life, I had the opportunity to focus on writing the book. Also, the team at Elektor has been very helpful and guided me through every step. Frankly, I cannot find words to express my heartfelt gratitude toward them for the great support they have always been.

Alina: You are greatly involved in online teaching, especially via Udemy. Can you describe this kind of experience to us? How are students engaging with you when compared to being in a traditional class?

Ashwin: I prefer teaching online as it increases my reach. I have also conducted programming boot camps for live audiences in the past. Online teaching helps me reach hundreds of thousands of students. People can engage with me by asking the questions on the portal. And since all the courses are self-paced, student engagement is much higher.



Ashwin Pajankar's workspace has various items.

Alina: When did you realize that you enjoy teaching other people about your favorite topics?

Ashwin: During the summer vacations in my engineering studies, I used to teach the students in senior high school and help them prepare for engineering entrance exams. I used to teach math, physics, computer science, and English. This is when I realized that I love teaching and making knowledge more accessible.

Alina: At the end of the day, what is the most important thing you would like your readers to learn?

Ashwin: The most important takeaway I have for my readers is to learn to explore the world of technology by themselves. The world of technology is very easy to navigate once you know where to look for relevant knowledge. In order to learn more, readers should go through all the online documentation, code examples, and various technical discussion forums by themselves.

Alina: Are you currently working on something? Any new ideas for a book?

Ashwin: Yes. I have been working with the Raspberry Pi 4. During the lockdown, it was short in supply. However, the situation has improved recently, and I was able

to procure one Raspberry Pi 4 with 8 GB RAM. I have been experimenting with it. Once I finish the personal project I have been working on, I will be ready to write a book on Raspberry Pi for Elektor. I love Elektor's book format, and Elektor has the best design team. I am planning to write many more technical books on diverse topics with Elektor.

Alina: Can you tell us about any personal projects you have currently going on?

Ashwin: Currently, I have embarked on the great journey of revamping my YouTube channel and expanding its audience. I will soon upload a lot of videos teaching school-level math in Hindi (a language spoken by a billion people all over the world). After that, I will cover basic Physics and Electrical Engineering (again, in Hindi).

Alina: Is there an achievement or contribution that you are most proud of?

Ashwin: Math, programming, and electronics are the things I do to earn a living. However, apart from my professional work, I have always been actively involved in giving back to the community through the social outreach programs at my workplaces and universities where I studied. Since I have started working as a freelancer, I have been doing it in my

personal capacity. As a result of my participation in educational outreach for underprivileged kids in the community through my university IIIT-H's community outreach program, my interview was featured in one of the most prominent newspapers in Hyderabad. I have received several awards for serving the community through educational outreach. This is the aspect of my life I am most proud of. ◀

220626-01

Questions or Comments?

If you have questions about this interview, feel free to e-mail Alina Neacsu at alina.neacsu@elektor.com or the Elektor editorial team at editor@elektor.com.



Related Products

- **A. Pajankar, Kickstart to Arduino Nano (Elektor 2022)**
www.elektor.com/20241
- **A. Pajankar, Kickstart to Arduino Nano (PDF, Elektor 2022)**
www.elektor.com/20242
- **Arduino Nano Every with Headers (SKU 19939)**
www.elektor.com/19939

LiDAR Precision Gauge

Measure Up to 12 Meters



By Somnath Bera (India)

LiDARs are great devices for detecting obstacles and measuring distances. Here we use one for taking precise distance readings of up to twelve meters with a resolution of 1 cm.

My original idea of buying a TFMMini-S LiDAR was to build some kind of radar with it. But, as happens so often, nothing really got done with the device until I had an enlightening experience at my job. Electric Resistance Welded (ERW) pipes are long pipes with a diameter of 300 mm or more and with a length of up to twelve meters. They are used in our power plant for ash slurry transportation. One day, while handling these long pipes for assembly line-up of ash slurry discharge pipelines, I found that my team had to measure each pipe precisely before taking them out through the exit gate. Every day, at least 30 to 50 times someone has to devote time for doing these measurements. Out in the open under a 47°C scorching sun, holding the tape measure at one end and a supervisor on the other end, then recording the length, is a very tiresome job. The idea of creating a precision measuring gauge took birth here.

LiDAR Modules

The TFMMini-S single-point ranging LiDAR (**Figure 1**) can take distance readings very fast (100 Hz) and very precisely (1 cm resolution, 1% accuracy) up to 12 m, which is exactly right for the job. There's also a *Plus* version that is faster (1 kHz) and has an IP65-rated enclosure. Both come with a serial interface.

Placing the LiDAR at one end of the pipe and then directing it towards the edge plate on the other end will give an accurate length of the pipe. Both versions of the LiDAR have a Field of View (FOV) of 3.4 degrees;

therefore, pointing it properly is very important for getting correct readings. To make this easier, we added a push-button-operated laser pointer to our system.

I used an ESP32 module for controlling the LiDAR and for doing the calculations. A small OLED display was added to show the distance readings. Incorporating a running average of a few measurements increases precision. Then the result can be printed as a floating-point value, but for speed and simplicity we have kept it as a plain integer value.

The TFMMini LiDARs use the reflection of invisible infrared laser light from the surface of the object they are aimed at. In case the object does not reflect or completely absorbs or diffuses the incoming laser light, the reading will be erroneous. For instance, avoid water and slanted and



Figure 1: The low-cost TFMMini-S LiDAR features a range of 12 m with a resolution of 1 cm.

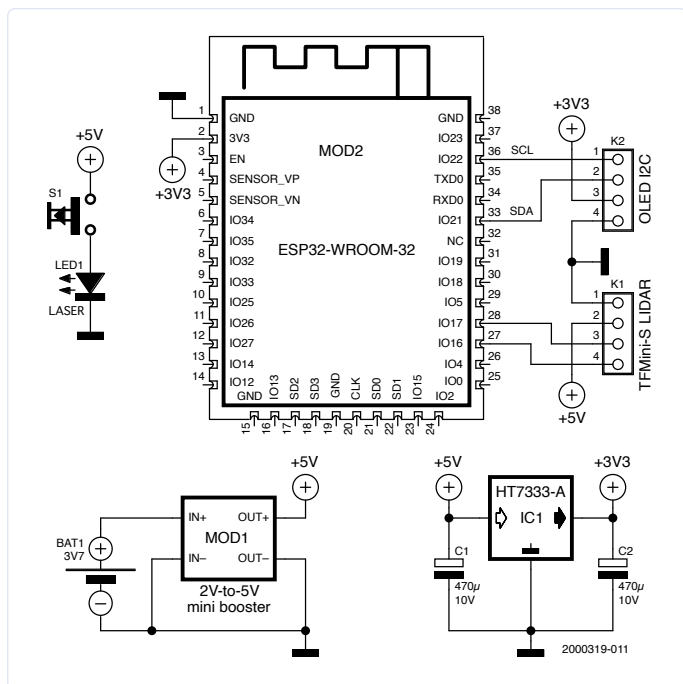


Figure 2: The prototype used an ESP32-WROOM module which does not have a 5 V regulator built in. When using a DevMod-C- or Pico-Kit-kind of module, IC1 can be omitted. In that case connect the 5 V supply to the module's 5 V input, not to its 3.3 V pin!

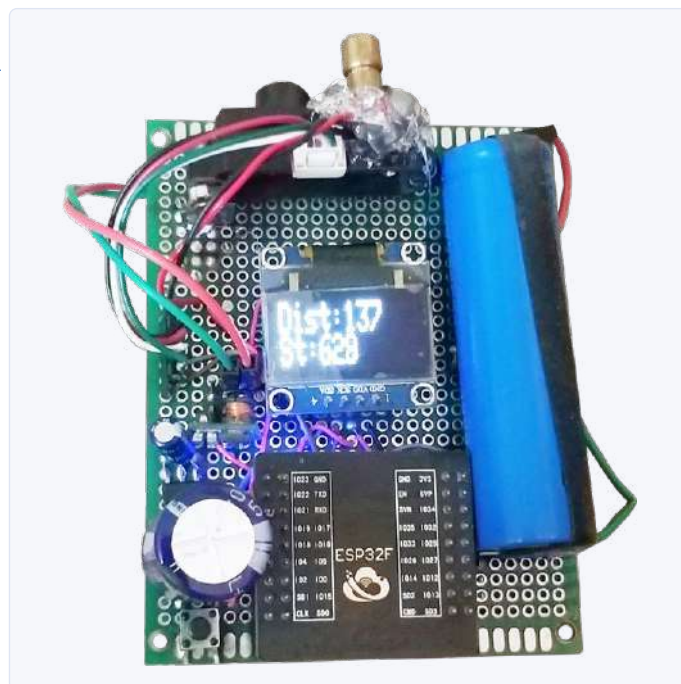


Figure 3: The prototype of the LiDAR-based precision gauge was built on a general-purpose prototyping board. Note how the laser pointer is hot-glued on top of the LiDAR module.

highly reflective glass windows. Most other objects reflect enough light and measurements are taken easily, even when the object is moving.

Constructing the Range Gauge

When building the device, I had to keep in mind that it had to be portable to allow my team to use it out in the field. The device must therefore be battery powered. The circuit I ended up with is shown in **Figure 2**. Both the LiDAR and the laser diode that I used for pointing need 5 V. The ESP32 WROOM module needs only 3.3 V. Therefore, I added a mini boost converter which can generate a 5 V supply out of an input voltage of as low as 2 V. I connected a low-dropout HT7333-A voltage regulator to the 5 V output to make 3.3 V for the ESP32. If you use an ESP32 module with on-board voltage regulator, then you don't need the HT7333-A. The complete setup runs from two 1.5-volt batteries or from a 3.7-volt LiPo battery. Note that the HT7333-A is a surface-mount device (SMD); therefore, be careful while mounting it.

Use the Automator!

The Elektor Automator [2] is the perfect platform for playing with the TFMMini-S LiDAR. Based on an ESP32 module, it has a Grove connector compatible with the LiDAR module, it has an OLED display and the necessary power supplies. We (at Elektor Labs) therefore created an Arduino sketch for this project that runs on the Automator. It uses the TFLidar and U8g2 libraries, available from the Arduino IDE's library manager. The software can easily be extended to add Wi-Fi or Bluetooth connectivity and to control the Automator's relay and LEDs. It can be downloaded from [2].

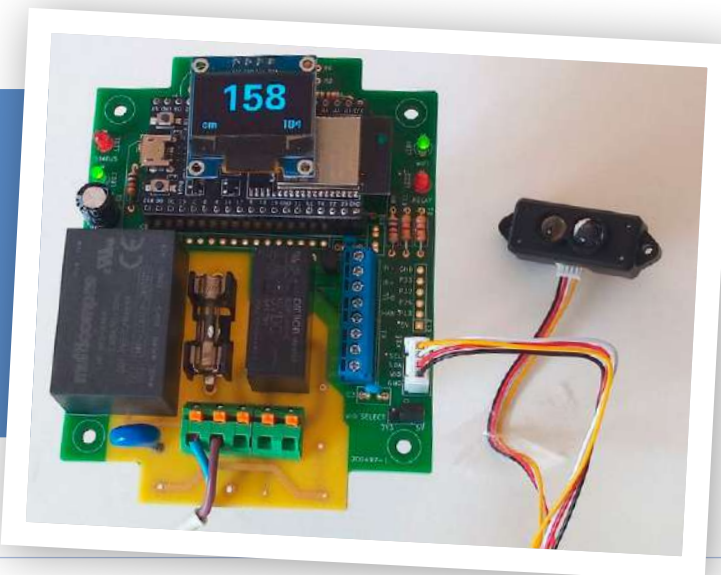
A Word About the Software

The software I wrote for the device is an Arduino sketch and can be downloaded from [1]. It requires the TFMMini library and Adafruit's GFX SSD1306 libraries that are available from the Arduino IDE's library manager.

The LiDAR and the ESP32 communicate with each other over a serial port (Serial1). Every 25 ms a new data point is requested. All the hard work is done by the TFMMini library while the sketch simply provides a user interface. Distance and signal strength values are printed on the OLED screen and are also transmitted on the "Arduino" serial port (Serial0).

Using the Device

After assembling all the parts on a general-purpose PCB (**Figure 3**), position the LiDAR on the edge of the board such that it has a clear view of the object. The laser pointer is a plain red laser diode controlled





by a push button. If possible, fix it on top of the LiDAR with hot glue (or something similar) such that both point in the same direction.

The minimum distance the TFMMini-S LiDAR can handle is 30 cm. Therefore, point it at an object or wall somewhere in the range of 30 cm to 12 m. Measure the distance and cross check with a tape measure. Eventually you will find that the reading is precise up to a centimeter.

The second line on the display indicates the strength of the received signal. As the object moves away, the strength diminishes. 

200316-01

Questions or Comments?

Do you have technical questions or comments about this article? Contact Elektor at editor@elektor.com.



RELATED PRODUCTS

- > **TFMini-S LiDAR (SKU 19691)**
www.elektor.com/19691
- > **TFmini Plus IP65 LiDAR (SKU 19690)**
www.elektor.com/19690
- > **ESP32-WROOM-32 (SKU 18421)**
www.elektor.com/18421
- > **0.96" OLED Display (Blue, I²C, 4-Pin) (SKU 18747)**
www.elektor.com/18747

WEB LINKS

[1] This project at Elektor Labs: www.elektormagazine.com/labs/tfmini-lidar-precision-gauge-30-cm-to-1200-cm

[2] The Elektor Automator: www.elektormagazine.com/labs/automator

Advertisement

Post your ideas and electronics projects

all sizes / all levels / all sorts

at www.elektor-labs.com
and become famous!



Create a project now at:
www.elektor-labs.com

design > share > earn



Audio Signals and the ESP32

The ESP-ADF Environment in Practice

By Tam Hanna (Hungary)

It is undoubtedly a challenge to develop any new consumer electronics device. Such products typically have an extremely short product life cycle so that time in development is crucial to success. This is particularly true for audio applications where algorithms will be used to implement various standard codec functions. The Espressif Audio Development Framework (ESP-ADF) is a powerful tool with resources which will save time and effort for developers of such applications.

Espressif's Audio Development Framework provides a software collection of algorithms and codecs which use a standardized format. To develop an application the designer just needs to link the appropriate "software ICs" in sequence without having to deal with their individual internal details.

The aim of this article is to give a basic understanding of how ADF works in practice. I have used Espressif's ADF tool many times to help implement solutions according to my clients' design requirements. I can share with you now many of the things I wish I had known when I first set out to use the ADF.

Hardware to Go

Let's start with the obvious: The ESP-ADF is compatible with any ESP32-based system. For the "audio part," the library offers a standardized driver interface that takes care of the data. Before

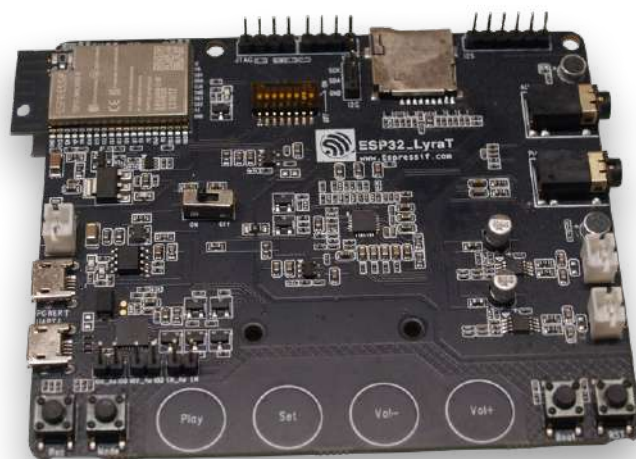


Figure 1: The small black PCB includes everything...

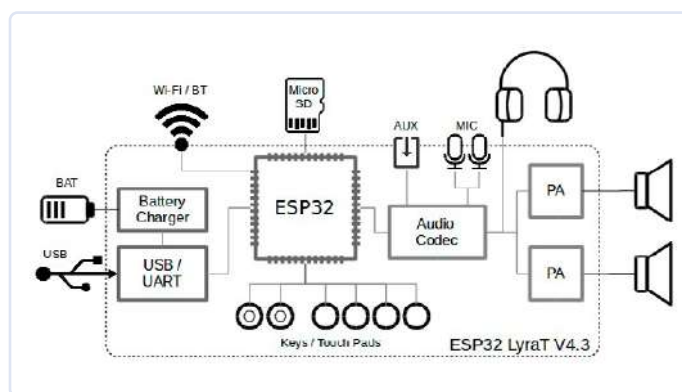


Figure 2 ...you need for prototyping audio applications (source: [6]).

you get down to finalizing the hardware design, much of the software can be written in advance and tested using an off-the-shelf development board. Espressif offers a range of such boards and for in-house development, I like to use the LyraT board shown in **Figures 1 and 2**.



Most important is the digitized “analog baseband” signal, which is supplied from the ES8388 (using two analog microphones) chip. Communication takes place both via I2S and via a group of dedicated pins — the circuit diagram and interconnects are provided by Espressif and can be easily replicated in the design of the finished device. The chip is readily available from UTSource and LCSC.

Working with the LyraT, the GPIO complement available from the on-board ESP32 is quite limited. As the design of the device hardware becomes more complex, you can quickly run out of interface options. In such cases, it may be necessary to consider the use of a second processor dedicated to taking care of the hardware communication.

Installing the ADF

Espressif provides the ADF as a plugin for its existing IDF environment. The ADF cannot be used in Arduino, and also is not even supported by some versions of Espressif’s own IDF.

The most convenient way to install the working environment is to download the full ESP-ADF repository. First, I use the following commands to create a subfolder, *esp4*, where the repository will be stored locally:

```
(base)tamhan@tamhan-thinkpad:~$ mkdir esp4
(base)tamhan@tamhan-thinkpad:~$ cd esp4/
```

The full repository can now be downloaded using the Git command line client:

```
(base)tamhan@tamhan-thinkpad:~/esp4$ git clone --recursive https://github.com/espressif/esp-adf.git
Cloning into 'esp-adf'...
```

Careful inspection of the output (not shown here) reveals that the ADF repository makes various references — for this reason, a download via the browser download function integrated in GitHub is not possible.

A version of the Espressif IDF is included along with the ADF. Some of the components require the presence of the `ADF_PATH` environment variable. If you want to use a command line window for processing ADF-based applications, you need to set the environment variable using the following commands:

```
(base)tamhan@tamhan-thinkpad:~/esp4$ cd esp-adf
(base)tamhan@tamhan-thinkpad:~/esp4/esp-adf$ export ADF_PATH=$PWD
```

The presence of the variable `ADF_PATH` sometimes affects normal IDF projects: I generally avoid this possibility by using a different terminal window for “normal” ESP32 work.

For the integrated IDF version, the usual installation run is required in the next step to provide compilers and other dependencies:

```
(base)tamhan@tamhan-thinkpad:~/esp4/esp-adf$ cd $ADF_PATH/esp-idf
(base)tamhan@tamhan-thinkpad:~/esp4/esp-adf/esp-idf$ ./install.sh
Detecting the Python interpreter
. . .All done! You can now run:

. ./export.sh
```

A script call is now required for the toolchain setup (the two consecutive dots are not a typo):

```
(base)tamhan@tamhan-thinkpad:~/esp4/esp-adf/esp-idf$ . ./export.sh
```

In practical development, it is advisable at this point to create a shell script for parameterization — with Bash, it can then configure itself automatically on demand.

Software Architecture: The Pipeline

After downloading and installing the ADF, it’s time to start thinking about the theoretical structure of the framework. True to the concept of a “software IC” originally developed by The Stepstone Corporation, the ADF also means that the developer essentially implements a pipeline consisting of a sequence of processing steps. For illustration, **Figure 3** shows a workflow model to implement an MP3 player.

In terms of the actual pipeline elements, Espressif does not seem overly expansive — **Figure 4** shows the roles and example implementations offered.

The only thing special about the present system is that the Unix concept where “everything is a file” also applies to peripheral devices.

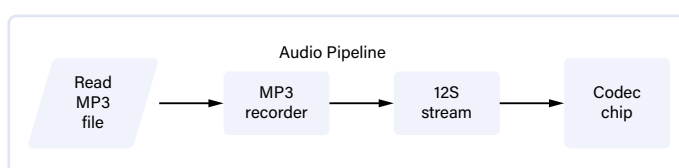


Figure 3: This pipeline decodes MP3s (source: [3]).

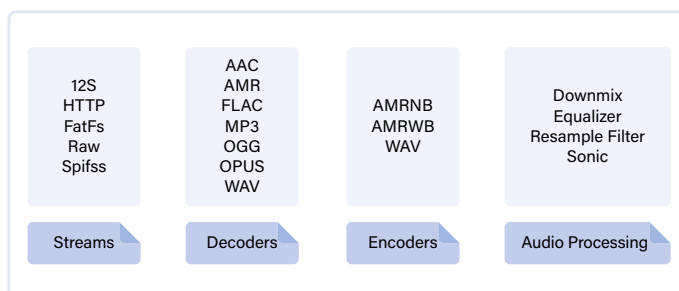


Figure 4: Pipeline elements can be divided into several types (source: [3]).

The API, which is documented in detail under [1] and is easiest to understand using the button functions provided under [2], allows the creation of “wrappers” that can be integrated directly into the pipeline. For application developers, however, this is only relevant insofar as the idea of also considering peripherals as pipeline elements sometimes needs careful consideration.

API documentation for the individual pipeline elements can be found under [3].

Software Architecture: Build an MP3 Player

For a practical understanding, let's take a look at one of the examples provided by Espressif: If you want to implement any project using the ADF, you are well advised to begin by looking for a similar or at least related template among the examples. Navigate to the example directory using the following command:

```
cd $ADF_PATH/examples/
```

We will see in the next few steps how to build a classic MP3 player. Go to the `~/esp4/esp-adf/examples/get-started/play_mp3_control/main` directory and open the `play_mp3_control_example.c` file using your preferred editor.

Most important is the entry point, which demonstrates the creation of some member variables. Our example needs a pipeline object and two element handles are required for the individual elements:

```
void app_main(void)
{
    audio_pipeline_handle_t pipeline;
    audio_element_handle_t i2s_stream_writer, mp3_decoder;
    ...
}
```

The hardware is commissioned in the ADF using an abstraction known as a *Board*. Your selection or parameterization takes place within the framework of *Menuconfig*. The commands to be created in the code-behind mainly evaluate the compilation constants:

```
audio_board_handle_t board_handle = audio_board_init();
audio_hal_ctrl_codec(board_handle->audio_hal, AUDIO_
    HAL_CODEC_MODE_BOTH, AUDIO_HAL_CTRL_START);
. . .
int player_volume;
audio_hal_get_volume(board_handle->audio_hal,
    &player_volume);
```

It is interesting here that the HAL also contains the volume control settings. Next up is the main pipeline object setup:

```
audio_pipeline_cfg_t pipeline_cfg =
    DEFAULT_AUDIO_PIPELINE_CONFIG();
pipeline = audio_pipeline_init(&pipeline_cfg);
mem_assert(pipeline);
```

Developers with previous ESP-IDF experience will find much of this familiar. The pipeline object is created by passing a configuration struct, which can then be passed from user to user.

This is followed by the generation of the actual pipeline elements. First up is the MP3 decoder:

```
mp3_decoder_cfg_t mp3_cfg =
    DEFAULT_MP3_DECODER_CONFIG();
mp3_decoder = mp3_decoder_init(&mp3_cfg);
audio_element_set_read_cb(mp3_decoder, mp3_music_
    read_cb, NULL);
```

The question now arises (especially in view of the pipeline shown in Figure 3) is how the data will be supplied. The answer to this is the `audio_element_set_read_cb` method, which receives a function (structured according to the following scheme) as a parameter:

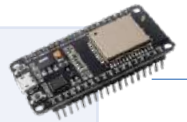
```
int mp3_music_read_cb(audio_element_handle_t el,
    char *buf, int len,
    TickType_t wait_time, void *ctx) {
    int read_size = file_marker.end
        - file_marker.start - file_marker.pos;
    if (read_size == 0) {
        return AEL_IO_DONE;
    } else if (len < read_size) {
        read_size = len;
    }
    memcpy(buf, file_marker.start +
        file_marker.pos, read_size);
    file_marker.pos += read_size;
    return read_size;
}
```

The callback supplies the information to be processed by the codec via the buffer. The I2S stream is then simpler; it is passed from `AUDIO_STREAM_WRITER` and parametrized as an output so that the incoming information gets directed towards the I2S sound hardware:

```
i2s_stream_cfg_t i2s_cfg = I2S_STREAM_CFG_DEFAULT();
i2s_cfg.type = AUDIO_STREAM_WRITER;
i2s_stream_writer = i2s_stream_init(&i2s_cfg);
```

Creating pipeline elements does not automatically declare them: This is accommodation by the ESP-ADF for developers who want to use multiple pipelines at the same time. The pipeline setup is instead done by registering the individual parts, each of which also has a string that serves as an ID:

```
audio_pipeline_register(pipeline, mp3_decoder, "mp3");
audio_pipeline_register(pipeline, i2s_stream_writer,
    "i2s");
const char *link_tag[2] = {"mp3", "i2s"};
audio_pipeline_link(pipeline, &link_tag[0], 2);
```

Listing 1. MP3-Player event processing.

```
while (1) {
    audio_event_iface_msg_t msg;
    esp_err_t ret = audio_event_iface_listen(evt, &msg, portMAX_DELAY);
    if (ret != ESP_OK) {
        continue;
    }
    if (msg.source_type == AUDIO_ELEMENT_TYPE_ELEMENT && msg.source == (void *) mp3_decoder
        && msg.cmd == AEL_MSG_CMD_REPORT_MUSIC_INFO) {
        audio_element_info_t music_info = {0};
        audio_element_getinfo(mp3_decoder, &music_info);
        ESP_LOGI(TAG, "[ * ] Receive music info from mp3 decoder, sample_rates=%d, bits=%d, ch=%d",
            music_info.sample_rates, music_info.bits, music_info.channels);
        audio_element_setinfo(i2s_stream_writer, &music_info);
        i2s_stream_set_clk(i2s_stream_writer, music_info.sample_rates,
            music_info.bits, music_info.channels);
        continue;
    }
}
```

In the second act, the pipeline is then linked using an array that supplies the individual string IDs in the correct order. With the MP3 player, only the activation of the pipeline is missing at this point:

```
ESP_LOGI(TAG, "[ 5.1 ] Start audio_pipeline");
...
audio_pipeline_run(pipeline);
```

In the case of the MP3 player, there is still a comparatively extensive event processing that reacts to various events (**Listing 1**). At this point, the exact event processing is irrelevant - it is more interesting to run through the compilation process just as a test:

```
(base)tamhan@tamhan-thinkpad:~/esp4/esp-adf/examples/
get-started/play_mp3_control$ make build
```

A note on this: To keep my business client data separate and secure, I am writing this article on my travel laptop. The Ubuntu version running here uses version 3.4.3 of CMake, which is why it is not possible to run *idf.py*. In practical use, *idf.py* is of course always preferable to the use of *make* and will also be mandatory in future versions of the IDF.

With a pristine project skeleton, the reward for the effort is the *menuconfig* screen, which offers a few additional options compared to a normal IDF project. The most important is the (Top) Audio HAL Audio board option, where you can select the appropriate board configuration as shown in **Figure 5**.

After saving, the build process begins in the same way it does for any IDF project. It is particularly important that the board has two Micro-USB ports: the POWER port is used to supply power to the board, while UART is only used for data communication. I normally just use two separate USB cables to connect the workstation to the board.

One thing to note about LyraT board is that (unlike many other development boards) it cannot be automatically reprogrammed; even before a connection can be established, you need to hold down the Boot push button while briefly pressing RST. You should then see:

```
Serial port /dev/ttyUSB0
Connecting.....
```

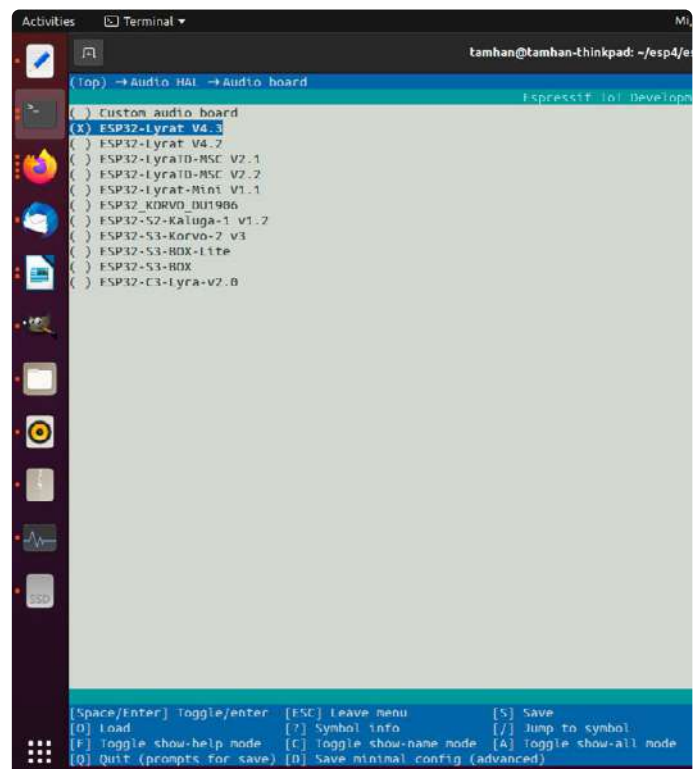


Figure 5: ADF extends Menuconfig with some settings.

If you got the push button sequence wrong, the system recognizes this as a corrupt communication message... try again:

```
A fatal error occurred: Failed to connect to ESP32:
Invalid head of packet (0x1B): Possible serial noise or
corruption.
```

Once the flash process has been successfully completed, you can plug headphones into the phono socket and press *Reset* again, if everything is in order you should hear a tune playing.

Finally, it should be remembered that running the configuration script *export.sh* is not sufficient to completely parameterize the environment. Before activating it, it is always necessary to set up the *ADF_PATH* environment variable correctly:

```
(base)tamhan@tamhan-thinkpad:~/esp4/esp-adf$ export
ADF_PATH=$PWD
(base)tamhan@tamhan-thinkpad:~/esp4/esp-adf/esp-idf$ .
./export.sh
Setting IDF_PATH to '/home/tamhan/esp4/esp-adf/esp-idf'
. . .
```

Processing Input Data

Using the ESP-ADF pipeline doesn't just make sense in the context of media playback: it's just as legitimate to put computations into the pipeline. For example, one of my recent jobs for a client required microphone data to be processed by an algorithm. The actual calculation was accommodated in a pipeline element, the general structure of which I would like to briefly show you here.

We begin by initializing the pipeline, and since the audio information arrives via the I2S bus, an I2S stream element is again required in the first step. In this case, however, its configuration now includes the *AUDIO_STREAM_READER* flag, which marks it as an input or data source:

```
i2s_stream_cfg_t i2s_cfg = I2S_STREAM_CFG_DEFAULT();
i2s_cfg.type = AUDIO_STREAM_READER;
#if defined CONFIG_ESP_LYRAT_MINI_V1_1_BOARD
    i2s_cfg.i2s_port = 1;
#endif
i2s_stream_reader = i2s_stream_init(&i2s_cfg);
```

Besides the I2S stream, we also need its companion, which encapsulates the proprietary algorithm. Here is its initialization:

```
tams_stream_cfg_t fatfs_cfg = TAMS_STREAM_CFG_DEFAULT();
fatfs_cfg.type = AUDIO_STREAM_WRITER;
tams_stream = tams_stream_init(&fatfs_cfg);
```

The actual pipeline construction then takes place by assigning strings and passing them on to *audio_pipeline_link()*:

```
audio_pipeline_register(pipeline, i2s_stream_reader,
    "i2s");
audio_pipeline_register(pipeline, tams_stream, "tam");
audio_pipeline_link(pipeline, (const char *[]) {"i2s",
    "tam"}, 2);
```

With this, we can switch to the header file that provides the elements necessary for commissioning the “Tam” task. The most important is the configuration struct, which contains, among other things, the meta information required for FreeRTOS (**Listing 2**).

TAMS_STREAM_CFG_DEFAULT is a convenience macro that allows the creation of the structs with default parameters. Space constraints mean we cannot print the constants here:

```
#define TAMS_STREAM_CFG_DEFAULT() {\
    .task_prio = TAMS_STREAM_TASK_PRIO, \
    . . .
}
```



Listing 2. Configuration of the Pipeline elements.

```
typedef struct {
    audio_stream_type_t    type;          /*!< Stream type */
    int                    buf_sz;        /*!< Audio Element Buffer size */
    int                    out_rb_size;   /*!< Size of output ring buffer */
    int                    task_stack;    /*!< Task stack size */
    int                    task_core;     /*!< Task running in core (0 or 1) */
    int                    task_prio;     /*!< Task priority (based on freeRTOS priority) */
} tams_stream_cfg_t;
```



Listing 3. Initialisation.

```
audio_element_handle_t tams_stream_init(tams_stream_cfg_t *config)
{
    audio_element_handle_t el;
    tams_stream_t *fatfs = audio_malloc(1, sizeof(tams_stream_t));

    AUDIO_MEM_CHECK(TAG, fatfs, return NULL);

    audio_element_cfg_t cfg = DEFAULT_AUDIO_ELEMENT_CONFIG();
    cfg.open = _tams_open;
    cfg.close = _tams_close;
    cfg.process = _tams_process;
    cfg.destroy = _tams_destroy;
    cfg.task_stack = config->task_stack;
    cfg.task_prio = config->task_prio;
    cfg.task_core = config->task_core;
    cfg.out_rb_size = config->out_rb_size;
    cfg.buffer_len = config->buf_sz;
    if (cfg.buffer_len == 0) {
        cfg.buffer_len = TAMS_STREAM_BUF_SIZE;
    }
    cfg.tag = "file";
    ...
}
```

The only exception we make is for `BUF_SIZE`. The supplied data is 16 bits wide, which is why we need the following constant value to process blocks with a length of 1024 slots:

```
#define TAMS_STREAM_BUF_SIZE (1024*2)
```

With that, we can turn to the initialization, see **Listing 3**.

The value of `cfg.buffer_len` determines the number of data words to be delivered per run. Otherwise, there is essentially only boilerplate code that does the housekeeping:

```
if (config->type == AUDIO_STREAM_WRITER) {
    cfg.write = _tams_write;
} else {
    cfg.read = _tams_read;
}
el = audio_element_init(&cfg);

AUDIO_MEM_CHECK(TAG, el, goto _tams_init_exit);
audio_element_setdata(el, fatfs);
return el;
_tams_init_exit:
audio_free(fatfs);
return NULL;
```

Audio elements created by the developer must integrate into a pipeline as shown in Figure 3. `audio_element_init()` needs some information for this – besides the thread-related parameters, the

routine also sets up a set of callbacks that the audio framework can use to inform the element about events that occur.

As an example, let's take a look at the `tams_open()` method, which takes care of the initialization of the element according to the following scheme:

```
static esp_err_t _tams_open(audio_element_handle_t self) {
    audio_element_info_t info;
    audio_element_getinfo(self, &info);

    initTamsWorkerAlgo();

    return audio_element_setinfo(self, &info);
}
```

Since the present algorithm does not need the information in `audio_element_info_t`, the implementation is limited to passing the parameter information to the pipeline.

The `read` and `write` events are responsible for data exchange. Our stream is only used as a “write point” and cannot be read, it therefore only outputs an error message in the system log when an attempt is made to read it out:

```
static int _tams_read(audio_element_handle_t self,
char *buffer, int len,
TickType_t ticks_to_wait, void *context)
{
    ESP_LOGE(TAG, "CENSORED");
    return 0;
}
```

Write operations are also implemented as “passthrough” and do not perform any actual operations:

```
static int _tams_write(audio_element_handle_t self,
char *buffer, int len,
TickType_t ticks_to_wait, void *context) {
    return len;
}
```

The next step requires an input buffer structured according to the following format:

```
int16_t DspBuf[4096];
```

`_tams_process()` is then responsible for the actual data processing. The most important thing here is the call to `audio_element_input`, which takes care of copying the information to the working buffer:

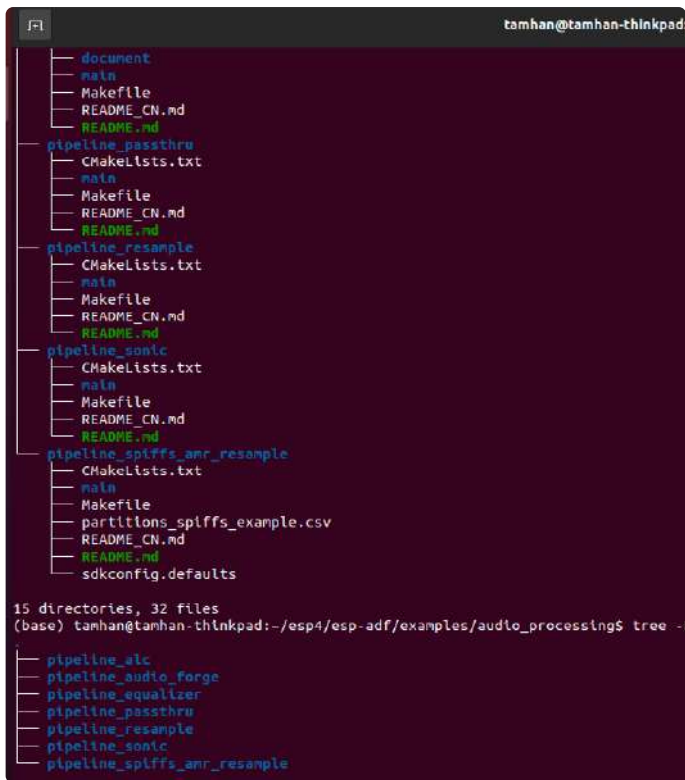


Figure 6: There's a whole bunch of pipelines to explore!

```
static int _tams_process(audio_element_handle_t self,
char *in_buffer, int in_len) {
    audio_element_input(self, (char *)DspBuf, in_len);
```

As a simple example of data processing, I included a simple normalizing operation on a range of values:

```
for (int i=0 ; i< in_len ; i++) {
    y_cf[i] = ((float)DspBuf[i]) / (float)32768;;
}
```

Finally, don't forget the housekeeping:

```
int r_size = audio_element_input(self, in_buffer,
in_len);
int w_size = 0;
if (r_size > 0) {
    w_size = audio_element_output(self, in_buffer,
r_size);
} else {
    w_size = r_size;
}
return w_size;
}
```

Operations to free up memory previously assigned for streams:

```
static esp_err_t _tams_close(audio_element_handle_t self)
{
    return ESP_OK;
}
static esp_err_t _tams_destroy(audio_element_handle_t
self) {
    tams_stream_t *fatfs =
```

```
(tams_stream_t *)audio_element_getdata(self);
audio_free(fatfs);
return ESP_OK;
}
```

The rest of the working code in the solution only needs to halt the core using the following instructions:

```
audio_pipeline_run(pipeline);
printf("halting.\n");
for(;;);
```

Note that the periodically called worker function in the pipeline can basically do whatever it wants - in this project, depending on the configuration, it communicates with either I2C or SPI peers in response to incoming sound data.

More Pipeline Elements

There are many more audio-related examples available in the ESP-ADF (**Figure 6**). Go to `~/esp4/esp-adf/examples/audio_processing` — the algorithms implement a wide range of additional pipeline operations.

I found the `examples/audio_processing/pipeline_equalizer` example particularly interesting. It implements a complete graphic equalizer function and is a class included in the ESP-ADF framework, so it doesn't need any advanced mathematics. Commissioning takes place according to the following, well-known scheme:

```
equalizer_cfg_t eq_cfg = DEFAULT_EQUALIZER_CONFIG();
int set_gain[] = {-13, -13, -13, -13, -13, -13, -13,
-13, -13, -13, -13, -13, -13, -13, -13, -13, -13,
-13, -13};
eq_cfg.set_gain = set_gain;
equalizer = equalizer_init(&eq_cfg);
```

You can find more about data fields under [4].

Integration into the playback stream then takes place as usual using `audio_pipeline_register()`:

```
audio_pipeline_register(pipeline, fatfs_stream_reader,
"file_read");
audio_pipeline_register(pipeline, wav_decoder, "wavdec");
audio_pipeline_register(pipeline, equalizer, "equalizer");
audio_pipeline_register(pipeline, i2s_stream_writer,
"i2s");
```

When an application reaches a certain degree of complexity, you get to a point where in-house signal processing elements will just not cut the mustard. Espressif provides help here with the ESP-DSP library available under [5]: it is a kind of framework that brings various DSP algorithms with high algorithmic stability and with various optimizations for the various chips common to Espressif.



Due to the organization of its folder structure, each ESP32 project is “directly” equipped to accept integration of additional ESP-IDF components. The installation of DSP is therefore easy. In the first step, create a copy of a project:

```
(base)tamhan@tamhan-thinkpad:~/esp4/esp-adf/examples$  
cp -r audio_processing/pipeline_equalizer/ tamsdspstest1  
(base)tamhan@tamhan-thinkpad:~/esp4/esp-adf/examples$  
cd tamsdspstest1/
```

Create a folder there with the name *components*. This then accepts the full version of the library, which can be downloaded from GitHub:

```
(base)tamhan@tamhan-thinkpad:~/esp4/esp-adf/examples/  
tamsdspstest1$ mkdir components  
(base)tamhan@tamhan-thinkpad:~/esp4/esp-adf/examples/  
tamsdspstest1$ cd components/  
(base)tamhan@tamhan-thinkpad:~/esp4/esp-adf/examples/  
tamsdspstest1/components$ git clone https://github.com/  
espressif/esp-dsp.git
```

When you next run the compilation toolchain, you will see the appearance of a new option for configuring the behavior of the DSP library, as shown in **Figure 7**.

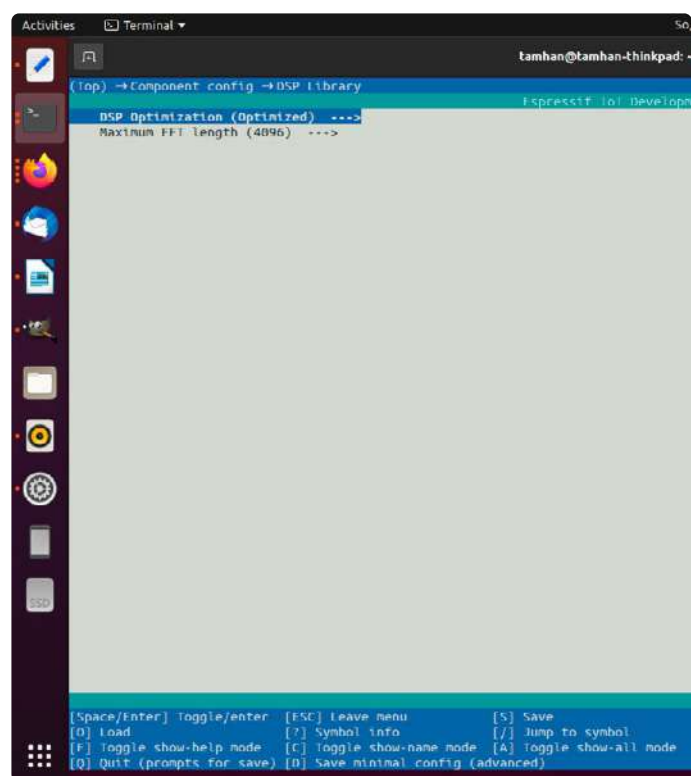


Figure 7: ESP-DSP nests under (Top) Component config DSP Library in the compilation process.

Summary

With the support provided by the ESP-ADF, Espressif gives developers a powerful and flexible framework to simplify the implementation of audio applications. With its help, my own business has already been able to save many hundreds of hours’ development time. I can highly recommend its use. ◀

220600-01

Questions or Comments?

If you have any questions or comments relating to this article, please contact the author at tamhan@tamoggemon.com or the editorial team via editor@elektor.com.

About the Author

Engineer Tam Hanna is a freelance developer, author and journalist (www.instagram.com/tam.hanna). He has been developing electronic products, computers and software for more than 20 years now. In his spare time, Tam designs and produces 3D-printed solutions and amongst other things has a passion to trade and enjoy high-end cigars.



Related Products

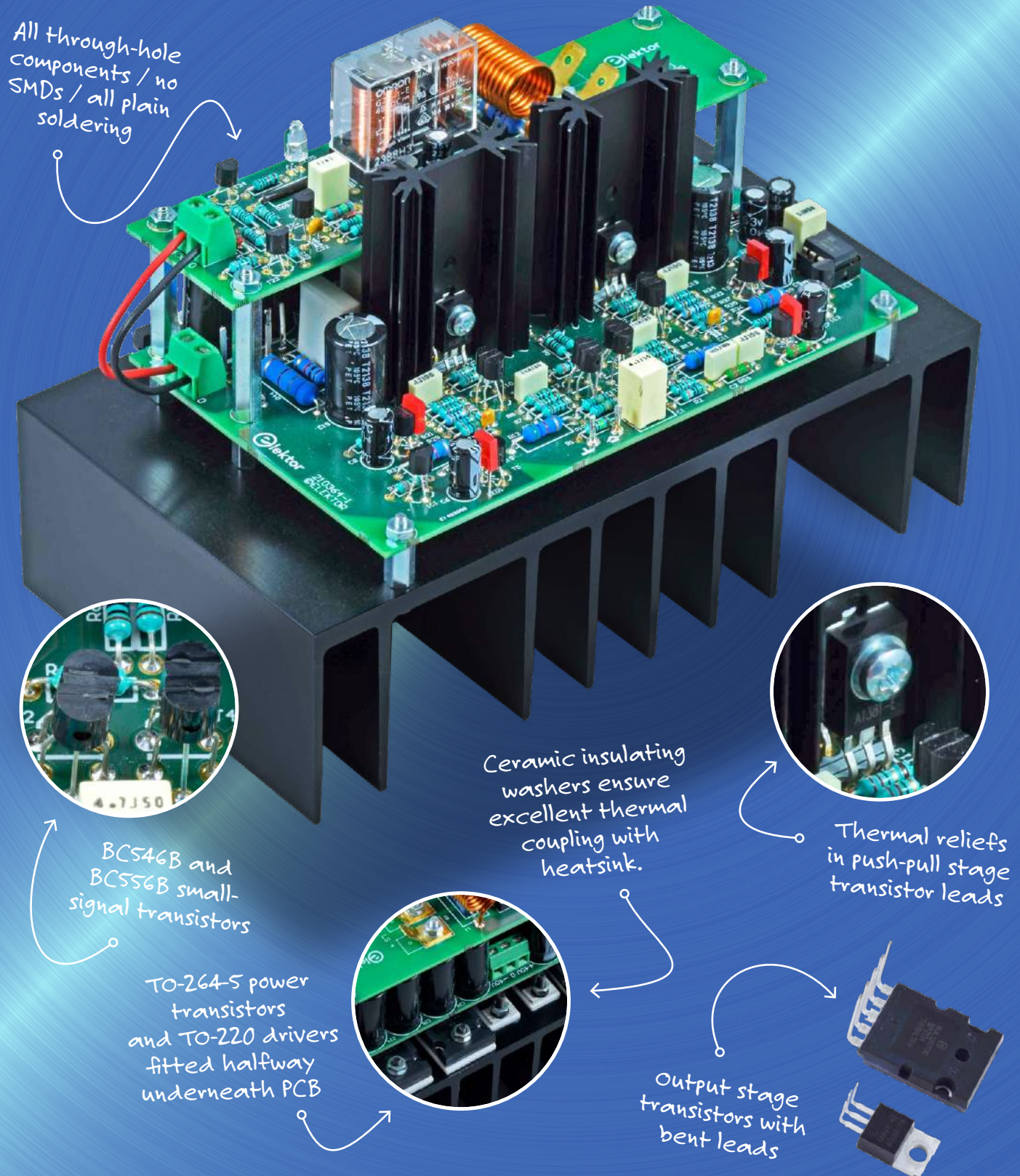
➤ ESP32-DevKitC-32D
www.elektor.com/18701



WEB LINKS

- [1] ESP-ADF Peripherals : <https://elektor.link/ESPPeripherals>
- [2] ESP-ADF Button Peripheral:
<https://elektor.link/ESPButtonPeripheral>
- [3] ESP-ADF API Reference:
<https://elektor.link/ESPAPIReference>
- [4] ESP-ADF Equalizer: <https://elektor.link/ESPEqualizer>
- [5] esp-dsp: <https://github.com/espressif/esp-dsp>
- [6] ESP32-LyraT Getting Started Guide:
<https://elektor.link/ESP32LyraT>

Elektor Fortissimo-100 Power Amplifier Kit





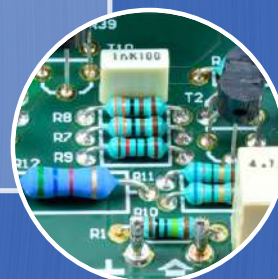
The Design

- All-analog, class-AB amplifier
- 100% symmetrical
- Extremely low distortion and noise
- ThermalTrack™ power transistors eliminate bias adjustment in output stage
- Symmetrical-bootstrap output stage for maximum output swing
- ± 40 V regulated power supply needed like SMPS800RE
- $3\ \Omega$ min. load impedance
- Solid, compact PCB-on-heatsink structure
- Mechanical work limited to 12 M3 holes drilled through heatsink
- +40 V supply voltage detection
- Output DC protection



The Components

- Elektor Kit = panelized PCBs + all THT components + heatsinks
- ThermalTrack™ transistors with integrated bias diode
- MJE15023 and MJE15033 drivers
- KSC3503 and KSA1381 push-pull amplifier stage transistors for high linearity
- OPA177 (DIP-8) driven DC-control loop
- 0.6 W 1 % metal-film resistors in all low-power positions
- Long-life, high-temperature, low-ESR electrolytic capacitors for bootstrap and supply decoupling



The Specifications

Input Sensitivity	1.076 V (94 W/8 Ω , THD = 0.1 %, B = 22 kHz)
Input Impedance	10 k Ω
Sine-wave Power	94 W (8 Ω , THD = 0.1 %) 181 W (4 Ω , THD = 0.1 %)
Bandwidth	3.3 Hz – 237 kHz (–3 dB, 1 W/8 Ω)
Open-loop Bandwidth	≈ 20 kHz
Open-loop Gain	$\approx 140,000$ (8 Ω load)
Slew Rate	45 V/ μ s
Signal-to-Noise Ratio	103 dB (B = 22 Hz – 22 kHz linear)
Harmonic Distortion Plus Noise	0.0008 % (1 kHz, 50 W, 8 Ω , B = 80 kHz) 0.002 % (20 kHz, 50 W, 8 Ω , B = 80 kHz) 0.0042 % (20 kHz, 100 W, 4 Ω , B = 80 kHz)
Intermodulation Distortion (50 Hz : 7 kHz = 4:1)	0.0015 % (50 W, 8 Ω) 0.0021 % (100W, 4 Ω)

More Info

www.elektor.com/20273



Using Light for Sound Effects

LDR-Based Voltage-Controlled 24 dB/oct Synthesizer Filter

By Raymond Schouten (The Netherlands)

This article describes a simple tunable filter circuit that needs no special chips, just two op-amps that can be running from a 5-V single supply. Despite its simplicity, the filter offers very low distortion ($\text{THD} < 0.01\%$), large signal handling and a wide dynamic range ($> 90 \text{ dB}$).

The 24-dB/oct audio low-pass filter design presented below is based on four light-dependent resistors better known as LDRs controlled by a single LED. The brightness of the LED controls the filter's cut-off frequency from approximately 20 Hz up to 20 kHz. Damping (i.e. Q factor, a.k.a. resonance) can be adjusted up to full oscillation, which is an attractive feature for music synthesizer applications. The circuit is easy to build, and component cost is around € 4.

This filter was designed with music synthesizers in mind, but its high signal quality might enable other audio applications too. The circuit was simulated first before building a prototype on a breadboard to measure its performance. Sound examples and a demo video are available on the Elektor Labs project page [1].

Two Versions

Two versions of the filter are available, the simplest one working from a symmetric

dual power supply. By adding a few resistors and capacitors, it can be turned into a version that works on a single supply. It was the second version that was built and measured (**Figure 1**).

The Description of the Circuit (Dual Supply)

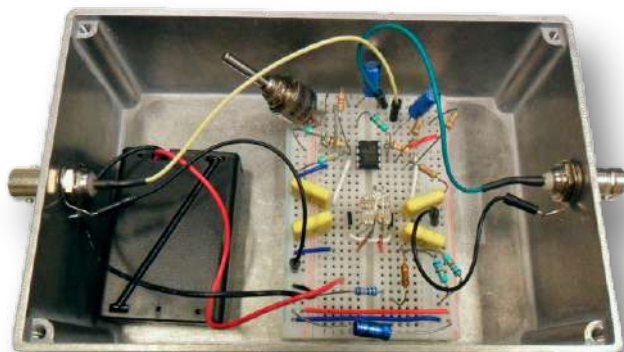
We will explain the circuit of the dual-supply filter as it uses fewer components. It is shown in **Figure 2**. The single-supply version (**Figure 3**) works exactly the same.

The heart of the filter is a 6-dB/octave RC low-pass filter with the fixed resistor R replaced by an LDR. Four of these filters are put in series to obtain a slope of 24 dB/octave. Controlling the resistance of the four LDRs at the same time by a single LED results in a filter with a cut-off frequency that can be tuned over a wide range ($> 1 : 1000$).

Cascading four identical RC circuits without buffers may look like a recipe for disaster, resulting in a very poor filter with sloppy transition at the cut-off frequency, but since the famous Moog ladder filter too is based on this principle, it can't be all bad. Robert Moog solved the problem in a clever way, a solution also used in our circuit. It is based on applying overall negative feedback that changes into positive feedback only at the cut-off frequency. The inset *Filter Q Control Explained* explains this in more detail.

Opamp U1a buffers the input and adds the global feedback signal via R9 and P1. Opamp U1b is a high-impedance buffer for the

Figure 1: The single-supply version of the filter was built on a breadboard and put inside a metal box for performance checking while being powered from three AAA batteries.



LDR/C cascade and adds some gain needed to compensate for the loss near the cutoff frequency.

P1 controls the amount of overall negative feedback. The filter moves from damped response to full oscillation by adjusting P1 from 100 k Ω to 0 Ω . Replacing P1 by another LED-LDR combination would allow for an electronically controlled Q factor. Refer to the inset entitled *Filter Response Simulation Results* for the frequency tuning and the Q-control curves.

A detailed explanation of selecting LED D1 and the concept behind its resistor circuit can be found in the inset *The LED-LDR Combination*.

Advantages of the Concept

Most analog tunable filters use more complex circuits that only work at small signal levels (< 100 mV_{pp}) to keep distortion at acceptable levels. Examples are filters based on tuned-bias transistors or diodes, discrete or as integrated into special ICs (OTAs like the famous LM13700). This LDR-based design, on the other hand, supports signals of several volts of amplitude with very low distortion figures. Actually, the voltage swing is mainly limited

by the opamps and the supply voltage as most LDRs can handle up to 100 V or more. You might even build this filter with valves! As the noise level of the circuit is fairly low - it is mainly determined by the opamps and the equivalent resistor noise of the LDRs - a wide dynamic range can be obtained.

Another advantage of the circuit is the extremely low (absent?) control-voltage leakage into the audio path because of having only optical coupling.

With suitable opamps it runs from a single low power supply (3 V to 5 V), making it compatible with microprocessor applications like Arduino.

Using voltage control on the LED (see inset *The LED-LDR Combination*) allows for a crude, but roughly exponential control curve that is needed for synthesizer applications. For more precise (but linear) control, a current source or PWM signal could be applied to the LED.

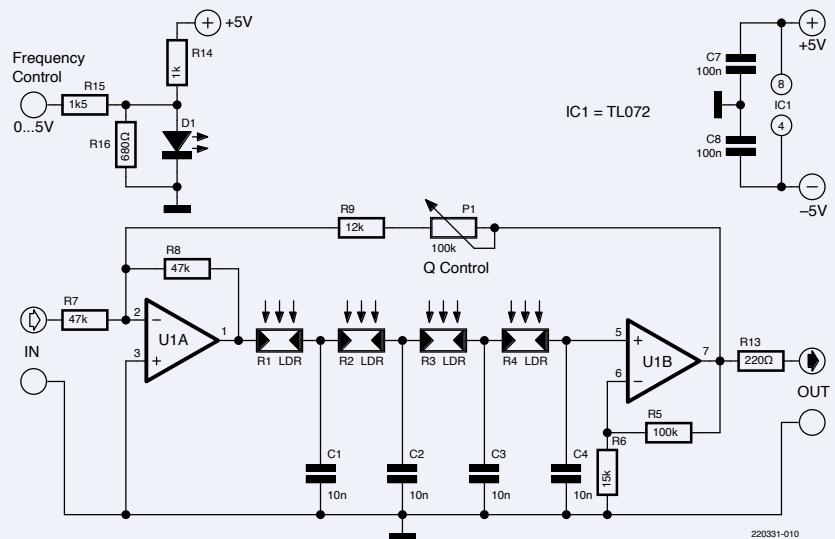


Figure 2: Circuit diagram for the dual-supply version of the filter.

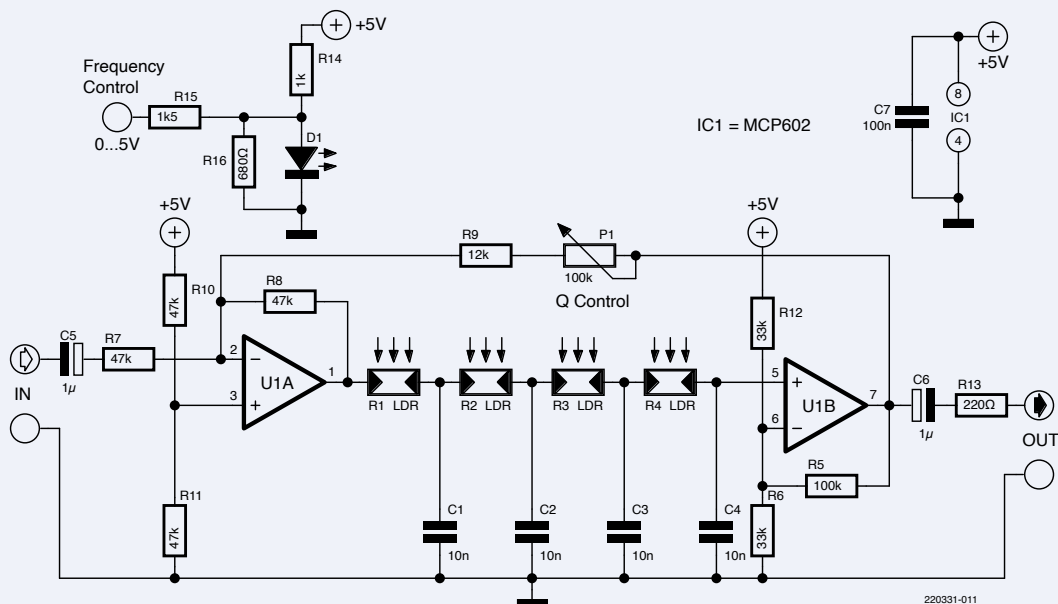


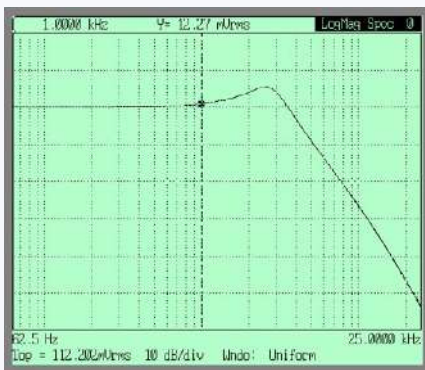
Figure 3: By adding a few resistors and capacitors the dual-supply filter can be turned into a single-supply one.



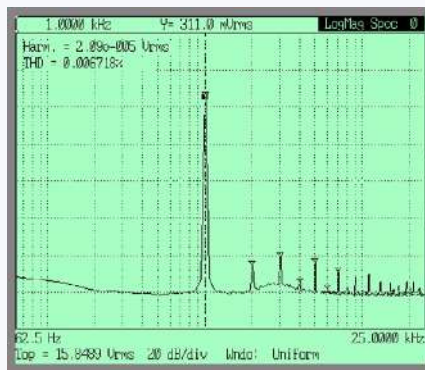
Distortion Measurements

For these measurements a Stanford Research SR770 FFT spectrum analyzer with built-in source was used. To prevent electrical and optical interference pickup, the circuit was placed in a closed metal box also containing the battery supply (see Figure 1). The circuit ran on a 4.5 V single supply here ($3 \times \text{AAA}$).

First, the analyzer was set to measure the filter response curve. Then the analyzer was set to generate a 1-kHz sine wave and measure the filter output spectrum up to 25 kHz. From that it calculated the total harmonic distortion (THD) for the first six harmonics. The filter was set to a cutoff frequency of 4 kHz. The reason for this choice is that there is positive feedback at the cutoff frequency and this increases distortion. So, this is a worst-case scenario for the second and third harmonic distortion (usually the dominant ones). One could argue that it is suppressing part of the higher harmonics but a check at $f_c = 13 \text{ kHz}$ showed almost the same harmonic pattern.



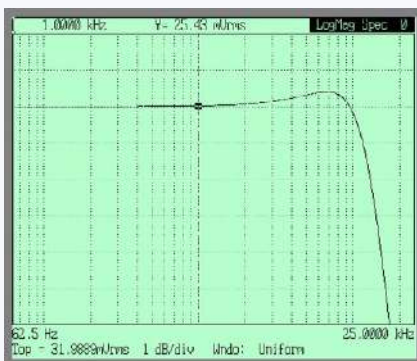
Measured filter response.



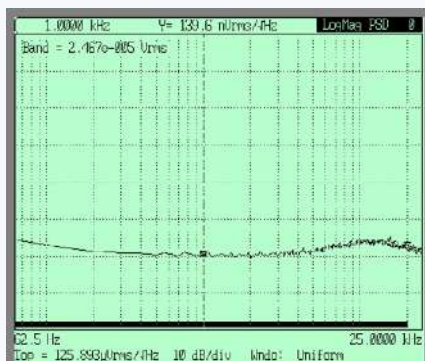
Measured filter distortion spectrum.

Noise Measurement and Dynamic Range

The output noise spectrum is measured with the source switched off and it shows in total $25 \mu\text{Vrms}$ in the audio band up to 20 kHz. In combination with the maximum signal handling of 4 Vpp (1.4 Vrms), this leads to a dynamic range of 95 dB. Note that you could improve this by using higher supply voltages.



Measured filter response curve (1 dB/div). The filter was set to -3 dB at 13 kHz.



Filter output noise spectrum as measured.

Limitations of the Concept

To be fair, the filter concept presented here also has some disadvantages when compared to the other concepts. The cutoff frequency setting is not very precise. It is sufficient for basic synthesizer use but a calibration table or an extra feedback circuit might be needed for more accurate applications.

Furthermore, the cutoff frequency control rate is limited by the relatively slow LDRs. When increasing the filter frequency, the measured response remains in the order of a millisecond but decreasing it can take up to 50 ms and even up to 500 ms for very low cutoff frequencies. Short 'attack' times are therefore possible for opening the filter at the start of a note. In music the attack time of a sound is usually shorter than its decay time, making this filter suitable for musical applications.

Suggestions and Remarks

The dynamic range can be increased further by using higher supply voltages. As mentioned before, LDRs can usually handle signals up to 100 V.

To obtain a more accurate and stable frequency control, you could use five LDRs instead of four and use one in the feedback loop of an opamp driving the LED. Because the filter was designed with music synthesizer applications in mind, the gain was made quite dependent of the Q-value to avoid large increases in the output signal. As an alternative, the input signal might be applied to the non-inverting input of U1a instead of to its inverting input. This would limit the influence of the Q-value on the filter gain and the filter would get a non-inverting transfer function. ◀

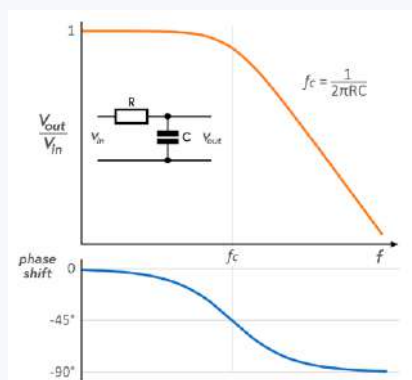
220331-01

Questions or Comments?

Do you have technical questions or comments about his article? Email the author at rs.elc.projects@gmail.com or Elektor at editor@elektor.com.



Filter Q Control Explained



An RC low-pass filter is not just a filter but also a frequency-dependent phase shifter, starting at 0°, going to -45° at the cut-off frequency and approaching -90° beyond that. With four of these RC filters in series, the phase shift at the cut-off frequency becomes $4 \times -45^\circ = -180^\circ$. This means that the signal changes sign at that frequency. Therefore, when we apply negative feedback from the output back to the input of the filter, the result is positive feedback at the cut-off frequency, producing gain peaking at that frequency. This gives your filter the sharp transition at the cut-off frequency. By adjusting the amount of negative feedback, the gain peaking can be varied. At a certain maximum feedback, it can become a (sine wave) oscillator that is tunable from 20 Hz to 20 kHz.

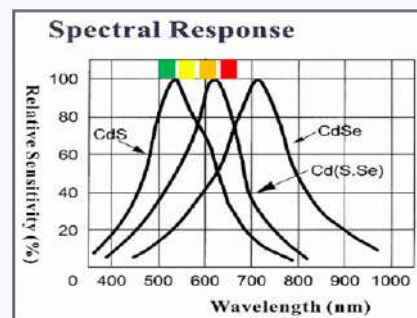
The LED-LDR Combination

Construction

The LED is mounted hanging some 5 to 7 mm above the four LDRs. Mechanical precision requirements are modest, measurements show no significant response difference with the LED being pushed 1 to 2 mm off axis. Make sure to properly shield the circuit from ambient light because it will modulate the filter also!

Choice of the LED Color

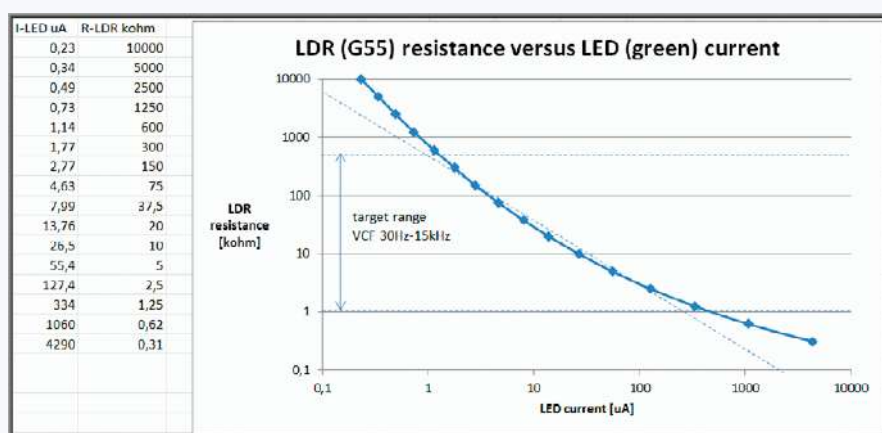
I used LDRs based on CdS, then green or yellow works best as shown in this spectral response diagram.



Source: JCHL datasheet of LDR GL5537-1

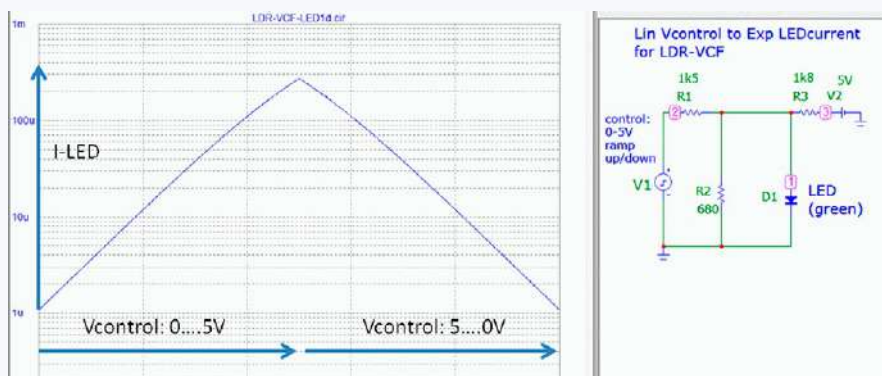
Sensitivity

Measurement showed that the LDR resistance is close to being a linear function of the LED current in the 500 kΩ to 1 kΩ resistance range. In total, the LDR resistance could be controlled from 10 MΩ to 300 Ω as shown in this measured curve.



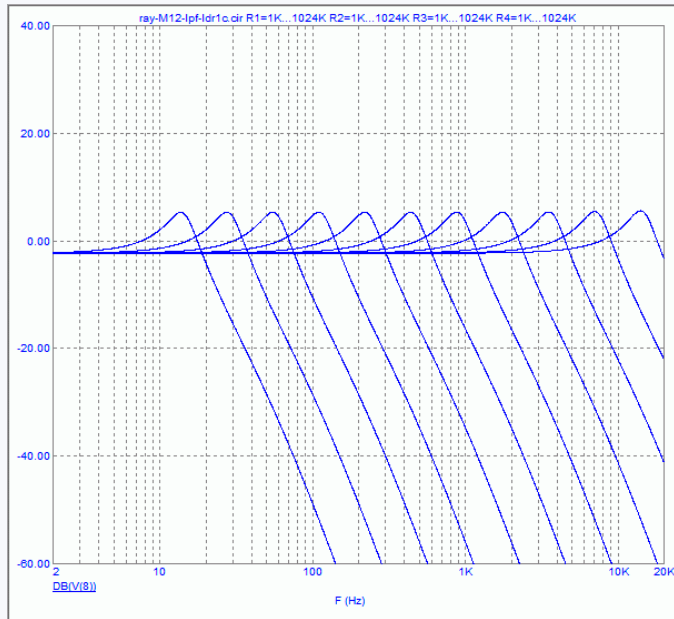
Note: A very high brightness green LED was used here (OVLBG4C7), 0,5 μA was enough to have visible light and to start lowering the LDR resistance.

A very simple but inaccurate way to approach an exponential control curve without using additional circuits is shown here. The concept is as follows. When controlling a diode (here the LED) with a voltage, the resulting current is an exponential function of that voltage. Connecting a voltage source directly to a LED is not a good idea but here we use low-enough resistor values to get close to this. The simulation below shows that when we sweep the control voltage in a linear way from 0 to 5 V, we get an exponential increase in LED current from 1 μA to 300 μA. In practice you might need to adapt the resistor values a bit to your LED. As mentioned, this is quite inaccurate and temperature dependent but the simplest way to approach exponential control for synthesizer applications. The sound examples were made using this control.

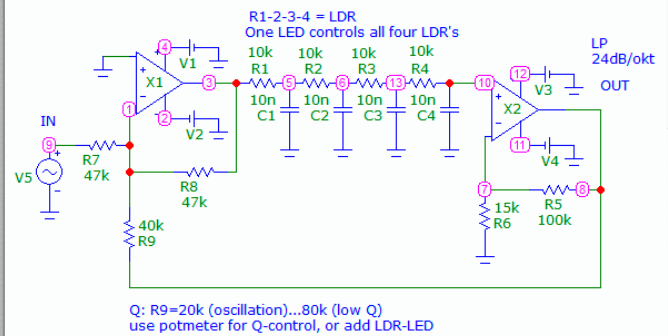




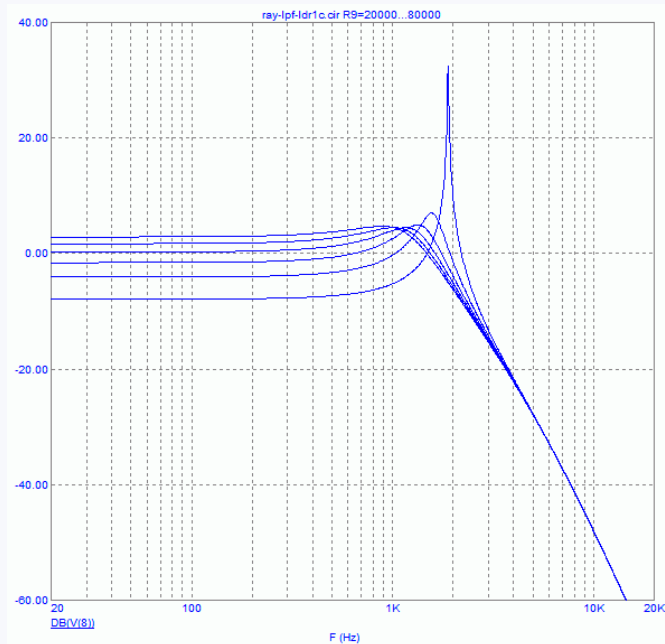
Filter Response Simulation Results



LDR based VC LPF 24dB/oct 20Hz-20kHz

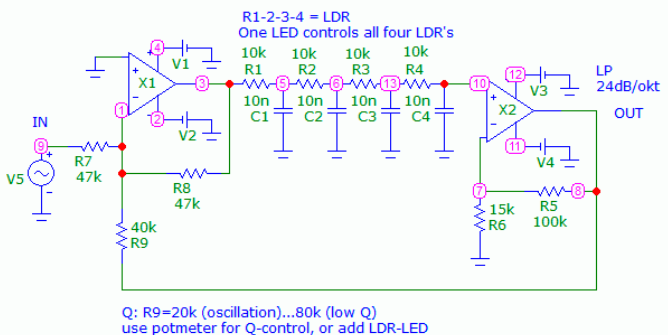


In this simulation the filter's cut-off frequency is swept from 20 Hz to 20 kHz while keeping the Q-control at a constant value.



LDR based VC LPF 24dB/oct 20Hz-20kHz

www.RS-ELC.nl



Here the filter's Q-control (R9) is swept from 80 kΩ to 20 kΩ while keeping the cut-off frequency control at a constant value.



Related Products

- Joy-IT ScopeMega50 USB Oscilloscope (SKU 18277) www.elektor.com/18277
- MIDI Input Board - Bare PCB (SKU 18382) www.elektor.com/18382
- The LTspice XVII Simulator (SKU 19741) www.elektor.com/19741

WEB LINKS

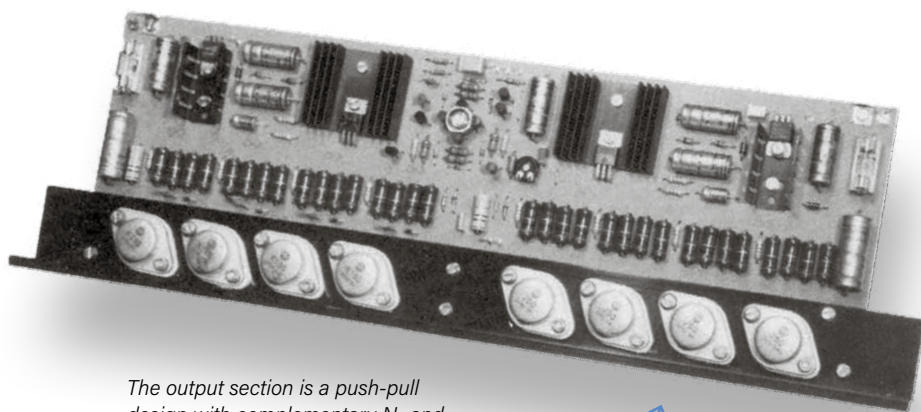
- [1] This project at Elektor Labs:
<https://www.elektormagazine.com/labs/voltage-controlled-24dboct-synthesizer-filter-using-ldrs>

Elektor High-Power AF Amplifier

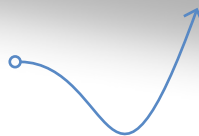
The **Loudest** of Them All!

By Larry Kossek (Elektor)

Audio amps have been in Elektor's DNA from the beginning. Thomas Scherer built them all, from 40 W to 200 W, before attempting, in 1986, to make a breakthrough with his own design capable of delivering high-quality sound at a then incredible output power of 1000 W.



The output section is a push-pull design with complementary N- and P-channel power MOSFETs of the horizontal type, selected for good transient response and linearity at all possible drive levels.

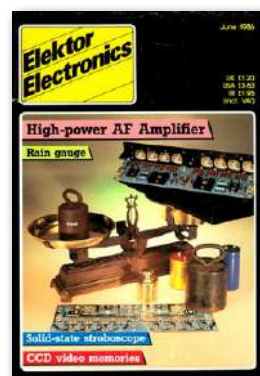


Segway Cloners Club

Delving into the Elektor archives brings back memories and surfaces surprising associations. Thomas Scherer, the man who designed this gigantic AF amplifier in 1986, later became a fan and promoter of the Elektor Wheelie. What do a 1000 W audio amp and a Segway clone have in common? It's a part with three legs, a metal case, and high current capability. Correct, it's the MOSFET power transistor, the workhorse that revolutionized modern electronics wherever power handling is concerned. Back in the eighties, these power components were not yet trivial in Elektor projects. And Thomas already had a penchant for high-current circuitry that put the fuses in the lab to the test.

"Wer Großes will, muß sich zusammenraffen"

Freely: *If you want it big, pull yourself together.* Of course, "regular" power amplifier projects were in Elektor's DNA right from the beginning. These amps came in quick sequence and Thomas built them all, from 40 to over 200 W. In 1986, he was a member of the Elektor editorial staff and dreamt of an audio amplifier so beefy it could deliver high-quality sound at the then-unimaginable power of 1000 W. Eventually, he designed one at home, in his spare time, and with his colleagues blissfully unaware. He went to bed late and many of his dreams featured complementary output transistor pairs. "I designed the analog circuitry using the power of mental arithmetic



Characteristics

- › Output power: 2 × 250 W (8 Ω), 2 × 500 W (4 Ω), or 1 × 1000 W (8 Ω)
- › Bandwidth: 8 Hz to 100 kHz
- › Distortion: 0.1 % (@ 1 kW); 0.01 % (@ 600 W)
- › Damping factor: > 100
- › Switch-on delay, fan control, voltage monitor
- › Input level: 0.775 V_{RMS} for full output
- › Weight: you don't want to know!

with a bit of help from a Sharp programmable calculator. [...] My proposal was accepted by Elektor for reviewing."

Exorbitantly Priced MOSFETs

Soon, Thomas had to proceed from the theoretical schematic to a tried, tested, grilled, and reproducible project: the "High Power AF Amplifier," without toasting too many parts, especially those exorbitantly priced Hitachi MOSFETs type 2SK135/2SK175 (T13-T16) and 2SJ50/2SJ55 (T17-T20).

"With supply voltages as high as ± 90 V, I faced peak voltages exceeding 125 V across the loudspeaker connections. Until then, I erred on the safe side of 48 V. [...] It was essential to delay switching on the loudspeaker and have a 'soft start' for the power transformer. [...] For distortion and some other measurements, my reference was the frighteningly expensive Brüel & Kjær spectrum analyzer that graced the Elektor Audio Lab to which I now had access."

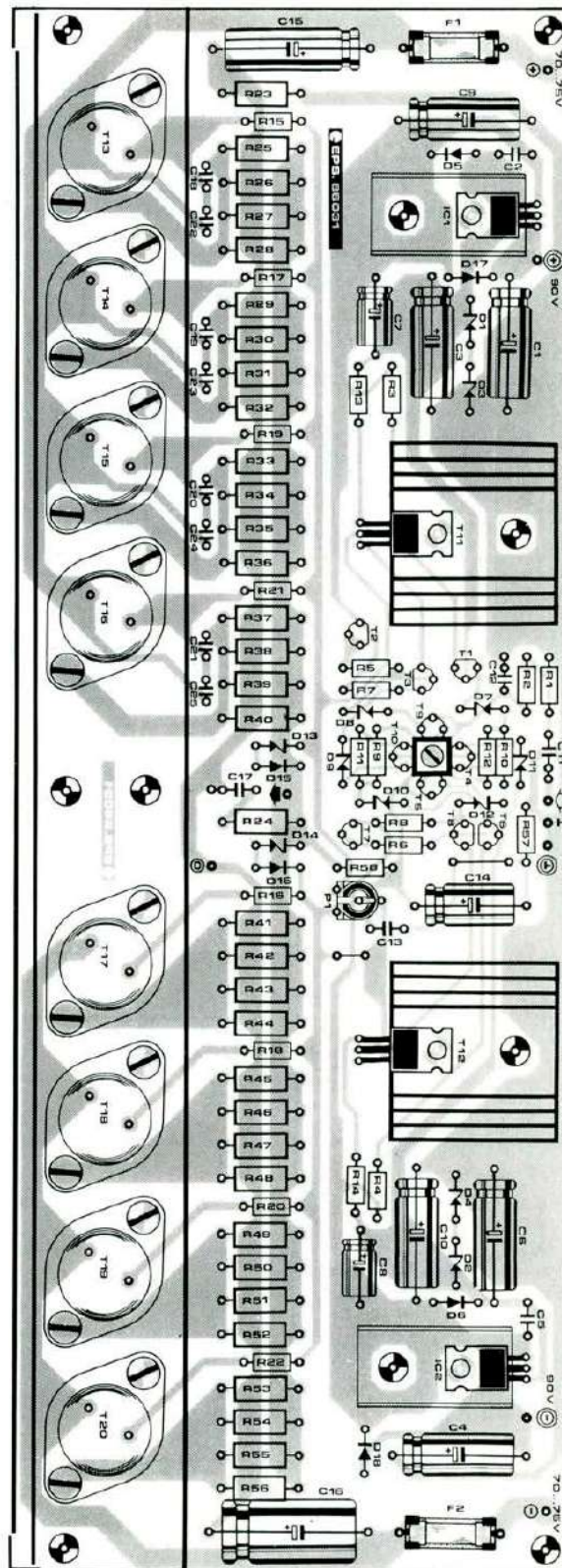
The one-kilowatt amplifier (also maliciously called *Killerwatts*) sparked considerable interest and resonated for many years [1][2][3]. Rarely and more than 35 years after its publication, a specimen of the kilowatt amp pops up for sale on the Internet. Many people have strutted their funky stuff and possibly "blown a fuse" to songs played LOUD through that beast of an amplifier. ◀

220234-01



Related Products

► **Elektor Archive 1974-2021**
(USB Stick) (SKU 20071)
www.elektor.com/20071



Protections aim at preventing disastrous bangs from the loudspeaker(s) and blown mains fuses when the amplifier is switched on.

WEB LINKS

- [1] T. Scherer, "High-Power AF Amplifier (1)," Elektor 5/1986: <https://www.elektormagazine.com/magazine/elektor-198605/46922>
- [2] T. Scherer, "High-Power AF Amplifier (2)," Elektor 6/1986: <http://www.elektormagazine.com/magazine/elektor-198606/46924>
- [3] T. Scherer, "Elektor High-Power AF Amplifier" (Retronics installment), Elektor 5/2016: <https://www.elektormagazine.com/magazine/elektor-201605/28977>



HomeLab Tours

A Volumetric Display Made in Canada

By Dan Foisy (Canada) and Eric Bogers (Elektor)

Using a phased array of ultrasonic transducers, it's possible to not only levitate a small foam ball but also move it back and forth at high speed. In combination with synchronized LED lamps, this allows a 'true' 3D display to be built thanks to the persistence of vision (POV) effect.

The idea of hovering a small foam ball on a cushion of ultrasonic waves is actually not new. What's more, this technique is now even available to the average hobbyist — see, for example, the article "Acoustic Wave Hovering" by Elektor editor Luc Lemmens, in which he discusses the Makerfabs Acoustic Levitation Kit [1].

The inspiration for a project can come from anywhere, but in the case of the Volumetric Display using an Acoustically Trapped Particle (more conveniently abbreviated as VATP) it was an article published by researchers at the University of Sussex [2], in which they described a method to use a particle captured by sound waves as a sort of display. When Toronto-based Dan Foisy saw the related video [3], he was totally entranced. Driven by his experience as an electronics hobbyist as well as the familiar wish to do the same thing himself, he went to work in his home lab. Let's hear it in Dan's own words.

"With the aid of a phased array of ultrasonic transducers, it's possible to hover a small (1 mm) foam ball and move

it quickly back and forth (at more than 1 m/s). Using the persistence of vision effect, it's then possible to draw figures in the air with this small ball, within a volume of approximately 100 × 100 × 140 mm. With the aid of RGB LEDs, the small ball can be illuminated at specific times along its path, resulting in multicolor images."

This sort of phased array is actually not new. For a long time now, the principle has been used in the reverse direction in fields such as radio astronomy, where the signals from many small antennas are combined to simulate a very large radio telescope (so large that it could never be built in practice). A good example of this is the LOFAR [4][5], an originally Dutch project that has now spread over all of Europe and consists of around 20,000 separate small antennas. Together, they form a phased array antenna with an unprecedented resolution of 0.2 arc-seconds.

"A lot of processing power is required to do more than just let the small ball hover, which means to control its



Figure 1: Two arrays, each consisting of 100 ultrasonic transducers, are used to make a small foam ball float in the air.

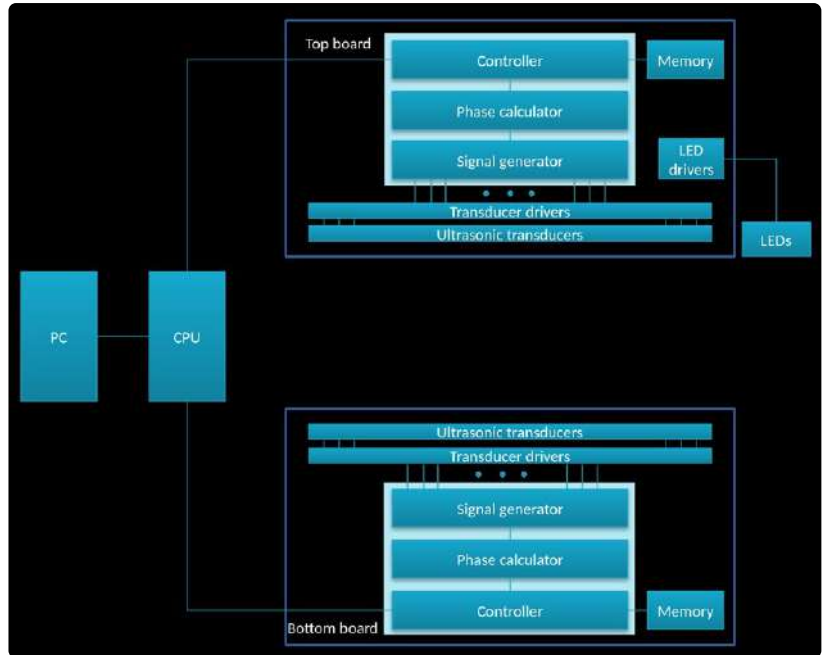


Figure 2: Block diagram of the VDTP. Except for the LED drivers, the top and bottom circuit boards are the same.

position. The VDTP that I built contains four FPGAs in addition to the main CPU, and they convert a position into the phased signals for the ultrasonic transducers. The FPGAs can also rotate, shift and scale the displayed images. Animation is also possible; I made an animation of a butterfly flapping its wings [6]."

Speaking of transducers: this project uses 200 of them, distributed over two circuit boards. Figure 1 gives a good impression of the layout. It should also be clear

that this project is far too extensive to be described in detail in Elektor magazine, so if you are interested in the VDTP, you should contact Dan Foisy by email (see the Questions or Comments? box).

"The block diagram of the VDTP was already a challenge in itself (see Figure 2). It consists primarily of two circuit boards holding the transducers — one at the top and the other at the bottom. Figure 3 shows one of the circuit boards in more detail, and Figure 4 shows

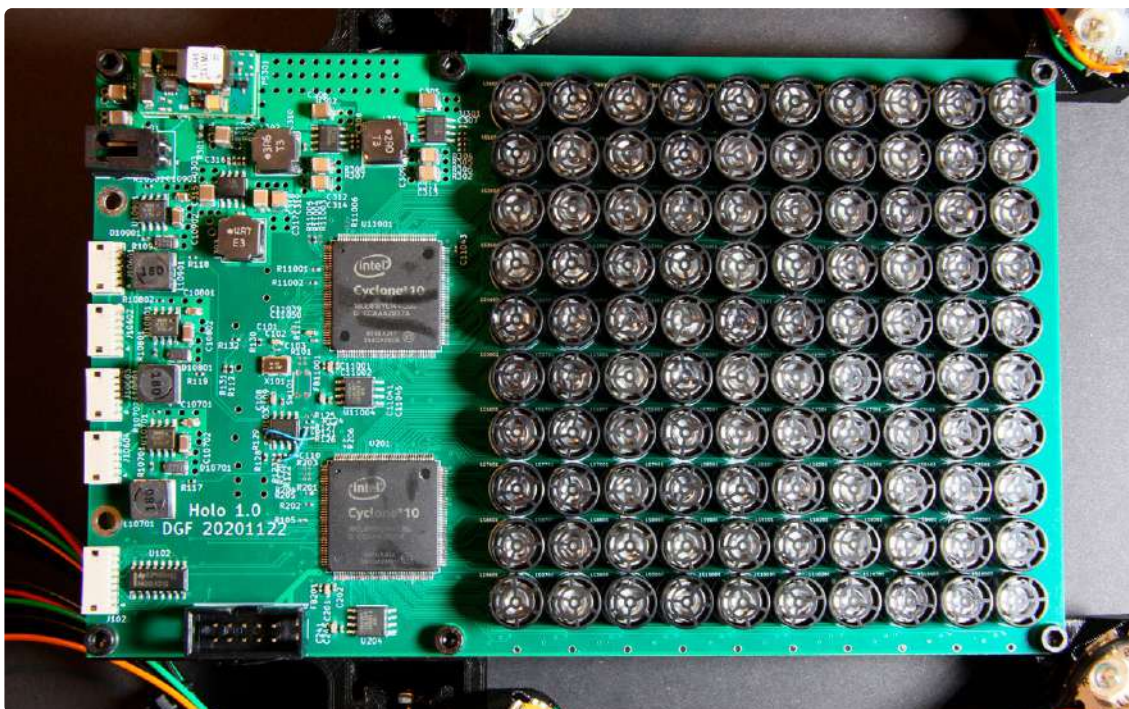


Figure 3: One of the fully populated circuit boards. Each of the 100 transducers was carefully tested before assembly.

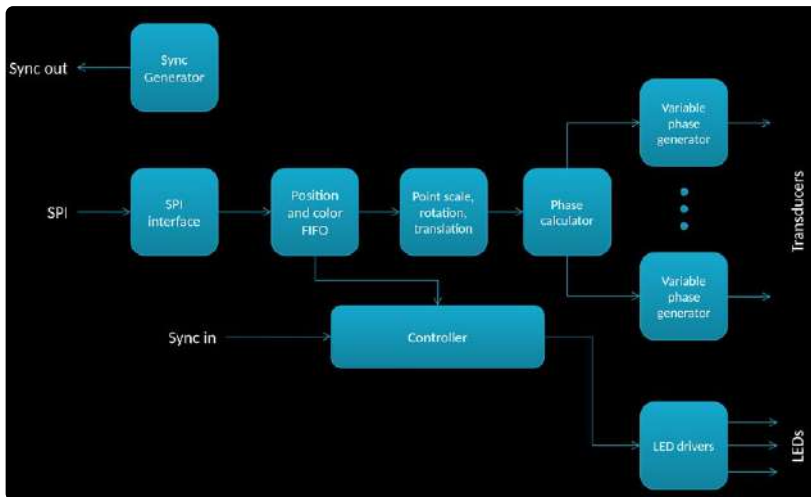


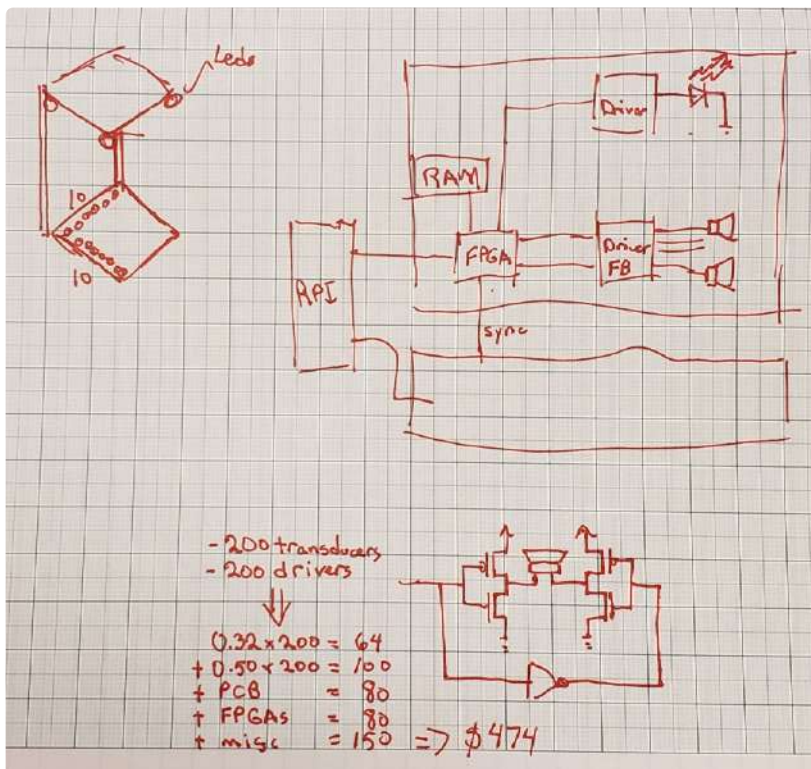
Figure 4: The block diagram of the FPGA code.

a more detailed block diagram of the top circuit board, which is also responsible for driving the LEDs."

As every maker knows, the journey from the initial idea to a working device is long, and the VDATP proved to be no exception in this regard. Figure 5 shows one of the initial sketches, and Figure 6 shows a fully realized part of the final design. As can be seen from Figure 7, KiCad was essential for the PCB design.

Figure 5: One of the earliest sketches of the VDATP.

"This was a very nice project, and it still gives me a lot of pleasure to watch the VDATP in action. There's



something magical about seeing an object floating in the air and then moving back and forth so fast that you can't really see it — it gives me a lot of satisfaction." ◀

210237-01

About the author

Dan Foisy was born in Montreal, Canada, in the mid-1970s and grew up in the early days of personal computers and the Internet — a really exciting time. At the end of the 1970s, his family moved to Toronto.

When he was 8 years old, he received a Radio Shack 160-in-1 electronics kit (equivalent to the well-known Philips experimenter kits in the Netherlands), which he used to build alarms, radios, amplifiers and other circuits. He also learned to design and build his own circuits.

Later, he received a Bachelor's and Master's diploma in Applied Science from the University of Toronto. During his Master's course of study, he went to work at the newly established Space Flight Lab, where he worked on the first Canadian space telescope, dubbed Microvariability and Oscillation of Stars (MOST) [7]. Following that, he was in charge of five other satellite programmes.

After holding a variety of other positions, he is now working at the Royal Bank of Canada, where he heads up several teams involved in AI, in particular natural language processing, as well as the appropriate and ethically responsible application of AI.

When he proposed to his future wife, he gave her a traditional diamond ring as an engagement present. His fiancée considered it entirely fitting to give him something in return, so she purchased a whole lot of equipment for his home lab — which he is still using. "For me that was a really great exchange :-)"

Questions or Comments?

If you are interested in building this yourself or in further developments of the volumetric display, you can always contact the author at danfoisy@gmail.com.



Related Products

➤ **Makerfabs Acoustic Levitator DIY Kit (SKU 19984)**
www.elektor.com/19984

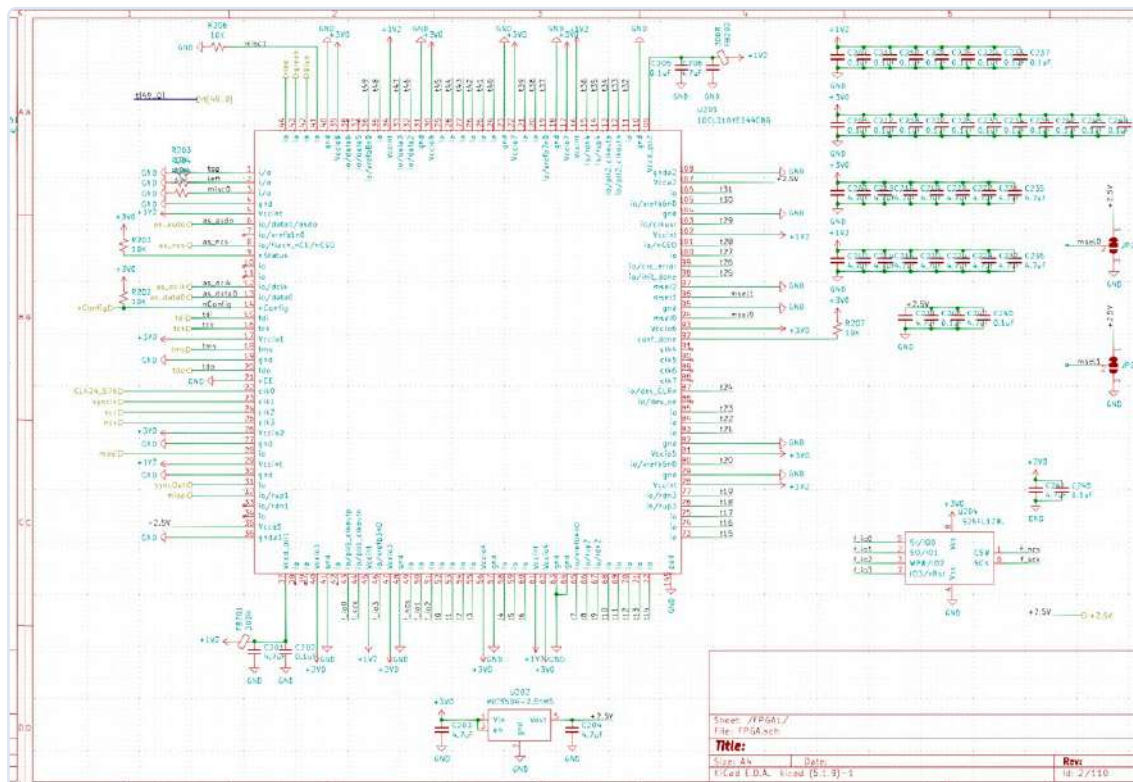


Figure 6: The final version of part of the circuit. Design software such as KiCad is virtually indispensable for projects of this scope; a paper-and-pencil approach is simply not good enough.

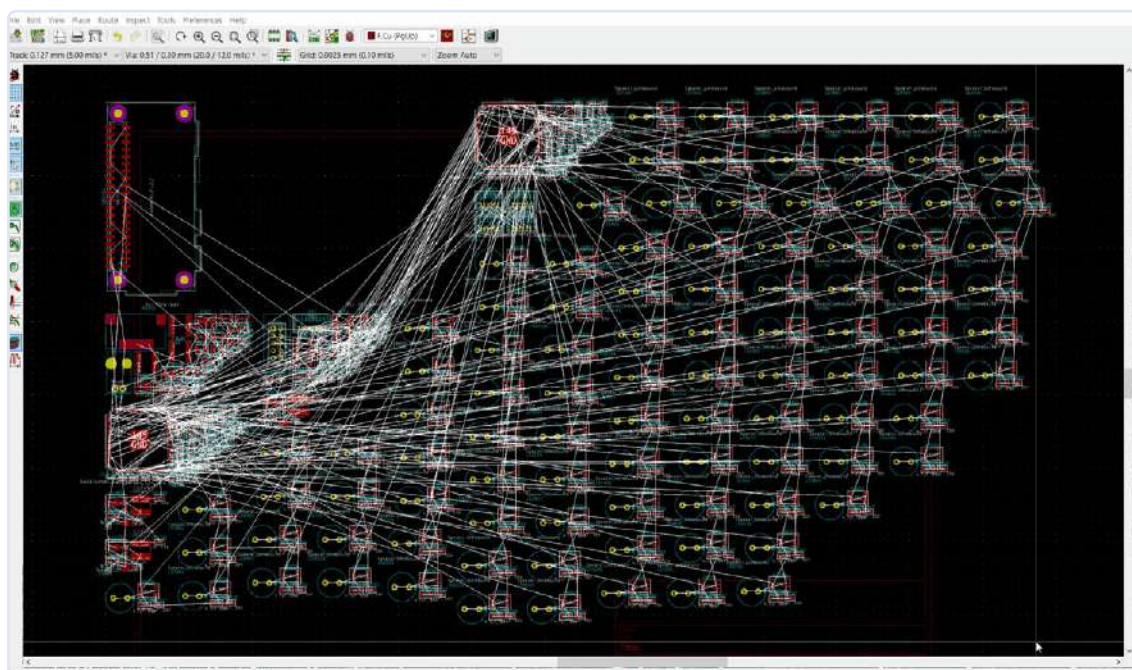


Figure 7: Untangling this rat's nest was a difficult job.

WEB LINKS

- [1] Luc Lemmens, "Acoustic Wave Hovering," Elektor July/August 2022: <https://elektormagazine.com/magazine/elektor-260/60551>
- [2] Original article from the University of Sussex (2019): <https://sro.sussex.ac.uk/id/eprint/86930/>
- [3] VDATP video from the University of Sussex: <https://youtu.be/Tm8JRIJ1q50>
- [4] LOFAR at Astron: <https://www.astron.nl/telescopes/lofar/>
- [5] LOFAR (Wikipedia): [https://en.wikipedia.org/wiki/Low-Frequency_Array_\(LOFAR\)](https://en.wikipedia.org/wiki/Low-Frequency_Array_(LOFAR))
- [6] VDATP video including butterfly animation by Dan Foisy: <https://youtu.be/hCC1C5KleUA>
- [7] MOST space telescope: <https://elektor.link/MOSTSpaceTelescope>

Err-electronics

Corrections, Updates and Readers' Letters

Compiled by Jens Nickel (Elektor)

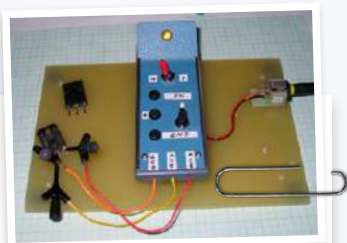
Power Semiconductor Tester

Elektor Summer Circuits 2022, p. 56 (210707)

This article was an excellent help for me in repairing a switching power supply: The “suspect” was desoldered and could thus be clearly identified as faulty. It was replaced by a working component from another board.

With materials from my stock, the tester was quickly assembled, but a suitable 12 V lamp was not available. Therefore, I used a $47\ \Omega / 5\ \text{W}$ resistor (in metal housing). The voltage drop (via a DIL bridge rectifier due to the different polarity) now lights up an LED with an $820\ \Omega$ series resistor (see photo). Thanks to the author, David Ashton!

Ulrich Strohmeyer



Fortissimo-100 High-End Amplifier

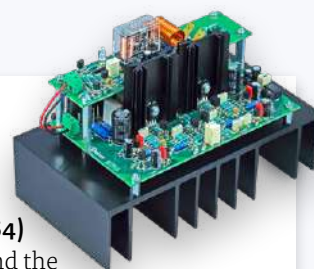
Elektor 11-12/2022, p. 6 (210364)

I find the amplifier interesting and the article contains a lot of data. But the info about power consumption is missing. My question is: How many watts does a channel consume at idle and at average power?

René Reynders

In idle state, the maximum total current after warm-up is about 350 mA, the power consumption is then 28 W. Since the power supply has a stable voltage of $\pm 40\ \text{V}$, the maximum average power is equal to the specified continuous power. However, the average output power for undistorted music playback is considerably lower than the specified continuous (maximum) output power. The total power of an AC current and a DC voltage is the average of the current times the DC voltage. For 181 W at $4\ \Omega$, the average current is $\sqrt{(181/2)/\pi} = 3.03\ \text{A}$. So, for a maximum undistorted sine wave, the total power input at $4\ \Omega$ is $2 \times 3.03\ \text{A} \times 40\ \text{V} = 242\ \text{W}$ - efficiency of the output stage is 74.7%.

Ton Giesberts



Obituary

It is with sadness that we announce that Peter Krengel, one of our authors, has died. We got to know Peter as a creative “Maker” who was always sprouting new ideas. From solder reflow ovens to milliohmeters, he still had many more projects in the pipeline, as well as Elektor books.

Peter, we will miss you!

The Elektor Team

A Fliege Notch Filter for Audio Measurements

Elektor 9-10/2022, p. 80 (210551)

The article immediately excited me: Finally, a really useful “old school” analog project — with contemporary components and high quality!

I contacted the author of the article, and I have to credit him for being very friendly and courteous! He had some come circuit boards left and sent them to me at an attractive price without much ado. We also discussed additional questions and variants.

A good mix is the key for this great magazine: Keep it up!

Jens Lemkamp

Thanks for the feedback and glad it worked out so well!
Jens Nickel, Editorial Team

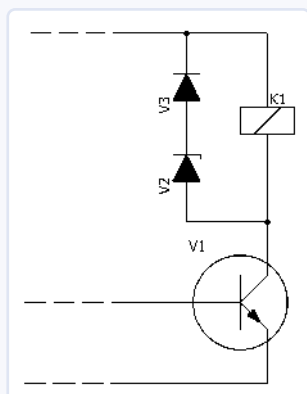


Figure 1

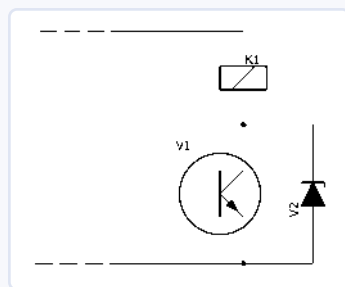


Figure 2

Starting Out in Electronics

Elektor 9-10/2022, p. 34 (220256)

Thank you very much for your article in the September/October issue 2022, where you show very nicely some applications of diodes, such as the freewheeling diode for a relay (shown in Figure 7 of that article).

I once used such a circuit to switch a solenoid valve in a filling plant. I noticed that the valve switched on immediately, but switched off with a delay. The delay time also differed from valve to valve, so that unfortunately I could not compensate for it, for example by switching it off a little earlier.

The solution to the puzzle was, of course, that the current through the coil decayed only gradually because the freewheeling diode only had a forward voltage of 0.6 V, so that at some point the closing spring gained the upper hand and tentatively closed the valve against the remaining magnetic force.

I was able to solve the problem by installing a Z-diode, as shown in **Figure 1**. The Zener voltage causes the current in the coil to decay much faster, which not only significantly reduces the switch-off time, but is also noticeable acoustically: A gentle “pling” turned into a loud “dong.”

To my surprise, I have never seen this trick elsewhere, neither in literature nor in real devices.

You can simplify the circuit a bit more by omitting the actual freewheeling diode and connecting the Z-diode in parallel with the transistor (**Figure 2**).

This might be an interesting variant for other Elektor readers.

Thomas Klingbeil

What Our Education System Can Learn from the Maker Community

At the age of 34, I find myself in the unusual situation of attending a technical college in Germany to obtain the advanced technical college certificate (for subsequent study). At the same time, I am admitted to a local university for “junior studies” as part of a program to promote gifted students. In this program, I attend courses in parallel with school with the possibility of already earning credit points for my studies.

In principle, nothing has changed at school for more than a decade; a teacher usually stands in front of the class and tells lesson after lesson about theoretical things, for which the students can imagine only few practical applications. The teachers do not seem to be able to understand why, after weeks of lessons on Ohm's law, an experiment to determine the internal resistance of a voltage source finds only moderate interest. Again and again, the question comes up: “What do we actually need this for?”

If the maker community were set up like this, we'd have a problem. Arduino and Co. are characterized by one thing in particular: A sense of achievement is generated very quickly. Extensive code examples and related projects almost always offer a starting point to get your project idea moving in the right direction as quickly as possible. People's natural curiosity to ask “Why?” is triggered, at the latest, when adjustments become necessary. I assume that virtually every reader of this magazine, including me, started by building circuits more or less “blindly.” Would anyone read a project article in Elektor if there were no finished circuit available? Or if it would be presented only after four weeks? Probably not.

Certainly, you can't quickly revolutionize something as huge as the education system. However, we could start at some point. The debate about “today's youth” is several decades old. In my youth, computer games were blamed; today, social media is the alleged culprit. Yet, both the computer game and social media arguments only support my hypothesis — both provide a sense of achievement early on. Imagine if every computer game or social platform required a theoretical preparation course lasting several weeks!

A very positive example is provided by Christian Albrechts University in Kiel, Germany, with its “Electrical Engineering Studies Entry Project” (www.einfachgutelehre.uni-kiel.de/allgemein/studieneingangsprojekt-elektrotechnik). Here, students in the first semester assemble a metal detector in groups before the regular start of lectures. The knowledge required for this is presented by the instructors in short, clearly understandable lectures. Each lecture conveys exactly as much knowledge as is necessary for the practical implementation. The main part of the project week was the practice; questions about the background came up all by themselves.

Sebastian Westerhold

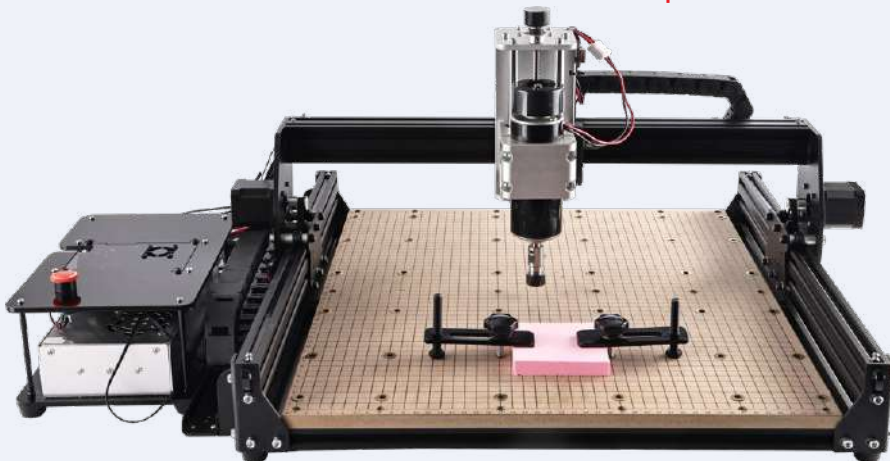
The Elektor Store

Never expensive, always surprising

The Elektor Store has developed from the community store for Elektor's own products like books, magazines, kits and modules, into a mature webshop that offers great value for surprising electronics. We offer the products

that we ourselves are enthusiastic about or that we simply want to try out. If you have a nice suggestion, we are here (sale@elektor.com). Our main conditions:
never expensive, always surprising!

Anet 4540 Desktop CNC Router Machine



-€200

€699.00

instead of

€899.00

Member Price: €629.10

 www.elektor.com/20260

Infrared Reflow Oven T-962

Revised Elektor Version

-€40

€229.00

instead of

€269.00

 www.elektor.com/20346





Great Scott Gadgets HackRF One SDR (1 MHz to 6 GHz)

-€40

€299.00

instead of
€339.00



www.elektor.com/18306

Raspberry Pi Pico W



+
FREE Pico W
Board

Price: ~~€47.90~~

Special Price: €39.95

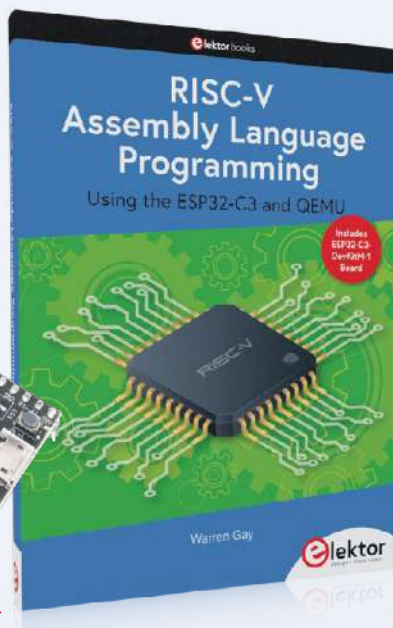
www.elektor.com/20335

RISC-V Assembly Language Programming

+
FREE ESP32
RISC-V Board

Price: ~~€59.90~~

Special Price: €39.95



www.elektor.com/20296

JOY-IT RD6006 Power Supply Bundle (360 W)



-€35

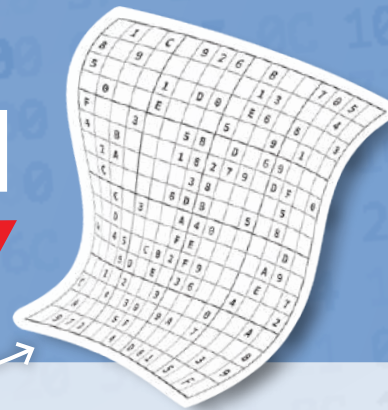
Price: ~~€199.95~~

Special Price: €164.95

www.elektor.com/19211

Hexadoku

Puzzles with an Electronic Touch



Traditionally, the last page of *Elektor* magazine is reserved for our puzzle with an electronics slant: welcome to Hexadoku! Find the solution in the gray boxes, submit it to us by email, and you automatically enter the prize draw for one of five Elektor store vouchers.

The Hexadoku puzzle employs numbers in the hexadecimal range 0 through F. In the diagram composed of 16 × 16 boxes, enter numbers such that **all** hexadecimal numbers 0 through F (that's 0-9 and A-F) occur once only in each row, once in each column and in each of the 4×4 boxes (marked by the thicker black lines). A number of clues are given in the puzzle and these determine the start situation.

Correct entries received enter a prize draw. All you need to do is send us **the numbers in the gray boxes**.



SOLVE HEXADOKU AND WIN!

Correct solutions received from the entire Elektor readership automatically enter a prize draw for five Elektor store vouchers worth **€50.00 each**, which should encourage all Elektor readers to participate.

PARTICIPATE!

Ultimately February 15th, 2023, supply your name, street address and the solution (the numbers in the gray boxes) by email to: **hexadoku@elektor.com**

PRIZE WINNERS

The solution of Hexadoku in edition 11-12/2022 (November & December) is: **DBE39**.
Solutions submitted to us before December 15th were entered in a prize draw for 5 Elektor Store Vouchers.
The winners are posted at www.elektormagazine.com/hexadoku.

Congratulations everyone!

E				B	8			3	F						1
	3			6	F		7	D		1	A				8
			7				A	B					2		
	D	8			1	3	5	C	7	E				A	F
	8	F	3			4			B			9	D	C	
B					2		9	8		A					7
	A	E		8		D	1	7	F		4		B	0	
	0	9		B							D		2	5	
	9	7		E							C		3	A	
	E	5		7		A	2	F	1		9		4	6	
D					3		B	4		6					5
	6	A	1			F			2			8	9	B	
	C	3			5	B	E	9	D	4			6	2	
			0				6	A				3			
	B			F	4		3	2		8	1			D	
2					9	C			6	0					B

7	B	8	C	2	A	6	D	1	4	5	E	F	0	9	3
9	D	E	5	1	0	3	8	A	F	6	B	2	7	4	C
F	0	2	3	4	B	E	7	9	C	D	8	5	6	1	A
A	6	4	1	9	C	5	F	7	0	2	3	B	8	D	E
4	7	9	8	A	6	B	1	5	D	E	0	3	C	F	2
B	1	F	A	C	3	0	E	2	6	4	7	D	9	5	8
C	2	D	6	5	7	8	9	B	1	3	F	E	4	A	0
E	3	5	0	D	F	2	4	8	9	A	C	1	B	6	7
D	8	B	9	7	4	C	5	F	2	0	1	A	E	3	6
2	A	7	F	6	8	D	B	E	3	9	4	0	5	C	1
1	C	0	E	F	9	A	3	6	7	8	5	4	D	2	B
6	5	3	4	E	2	1	0	C	A	B	D	7	F	8	9
5	9	1	2	8	E	F	6	D	B	7	A	C	3	0	4
0	E	A	B	3	D	9	C	4	8	F	2	6	1	7	5
8	4	C	D	0	5	7	2	3	E	1	6	9	A	B	F
3	F	6	7	B	1	4	A	0	5	C	9	8	2	E	D

The competition is not open to employees of Elektor International Media, its subsidiaries, licensees and/or associated publishing houses.



PROTEUS DESIGN SUITE

Design Quality Assurance

Constraint Driven Design

Flexible and scalable
rule system

Full support for design
rule rooms

Manufacturing
solder mask rules

Live display of
violation areas

Zone Inspector

Analyze plane coverage and
stitching

Grid view of plane
configurations

Edit plane settings and
draw order

Pre-Production Checklist

Set of board tests
before Gerber Output

Includes placement,
connectivity and
clearance testing

Completely independent
code for clearance checks

Dedicated Reporting Module

Tables automatically
populate with design
data

Compliance status for
diff pairs and length
matched routes

Make custom
reports with data
object tables

Generate reports
from templates

