



elektor

LEARN

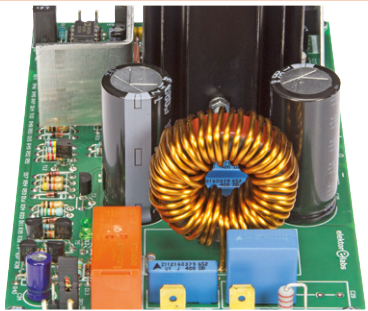
DESIGN

SHARE

Sand Clock

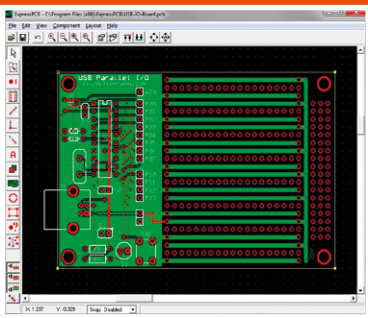


a real eye-catcher



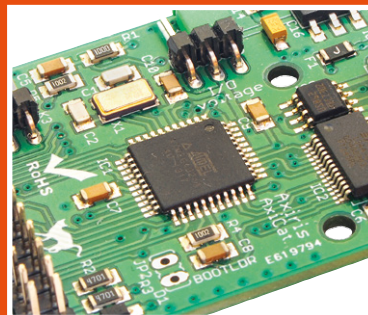
Cool Power with D-Watt

A high-performance amplifier with lots of output power



CAD for PCBs

Free and non-free PCB layout software



AxiCat

Link an IC to a Linux or Windows PC

Analog Random Numbers • AxiCat • BrainBox Arduino • Build a Chipamp! • CAD for PCBs • Catdoku • Color LEDs • Connect Objects with Genuino 101 • Cool Power with D-Watt • DAB+ Antenna Diplexer • Debugging the Arduino Zero & M0 Pro • electronica Fast Forward Award 2016 • Elektor Labs Pipeline • Elektor SDR Reloaded (4) • Err-lectronics • The Fetron • Retronics: From Verobox to Heavy Metal • Internet Radio with Fluorescent Display (2) • IoT Shield for Arduino • LED Stairway Lighting • LEGO Control Board for the Raspberry Pi • MAXREFDES99# Display Driver Shield • The Mendocino Motor • Opamp Power Supply Connexions • Programming Adapter for USBasp • Sand Clock • Sensors make Sense (2) • Simple Interfacing to Analog and Digital Position Sensors • Simulation with SystemVision • Super Simple Dice • SUPRA 2.0 MC Version

- ▶ More than 45 years of experience
- ▶ 24-hour shipping
- ▶ More than 70,000 products



THE REICHELTV ADVERT



rch.it/vuk
FIND OUT MORE ▶

“ Arduino microcontrollers make it easy for you to turn your ideas into reality! ”

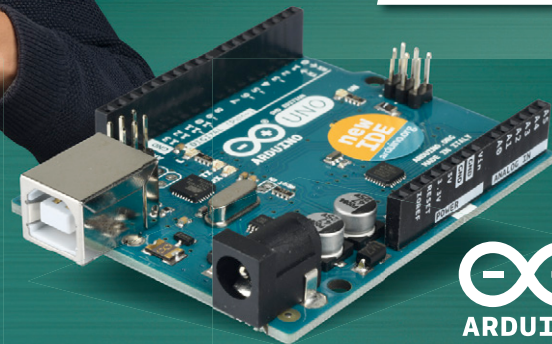
The most popular Arduino!

ARDUINO UNO REV. 3

- ATmega 328
- 14 digital I/Os (6 as PWM)
- 6 analog inputs
- USB port

ARDUINO UNO

14.²⁵



Small but not to be underestimated!

ARDUINO™ MICRO

- ATmega 32u4
- 20 digital I/Os (7 as PWM)
- 12 analog inputs
- Micro USB port



ARDUINO MICRO **13.52**

Small but mighty!

ARDUINO™ NANO

- ATmega 328
- 14 digital I/Os (6 as PWM)
- 8 analog inputs
- Mini USB port



ARDUINO NANO **16.73**

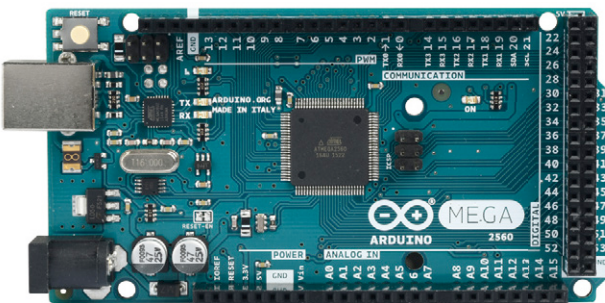
Just connect and you're ready to go!

ARDUINO™ MEGA 2560

- ATmega 2560
- 54 digital I/Os (14 as PWM)
- 16 analog inputs
- USB port





ALL ARDUINO PRODUCTS
rch.it/yb



ARDUINO MEGA **25.50**

Daily prices! Price as of: 01. 12. 2016

Prices in £ plus statutory VAT, plus shipping costs · reichelt elektronik, Elektroniking 1, 26452 Sande (Germany)

Onlineshop languages:  

PAYMENT METHODS:



SHOP CONVENIENTLY ONLINE!



www.reichelt.co.uk

ORDER HOTLINE: +49 (0)4422 955-360

Elektor Magazine

Edition 1/2017
Volume 43, No. 481 & 482
January & February 2017

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

www.elektor.com
www.elektormagazine.com

Elektor Magazine, English edition
is published 6 times a year by

Elektor International Media
78 York Street
London W1H 1DP
United Kingdom
Phone: (+44) (0)20 7692 8344

Head Office:
Elektor International Media b.v.
PO Box 11
NL-6114-ZG Susteren
The Netherlands
Phone: (+31) 46 4389444
Fax: (+31) 46 4370161

Memberships:
Please use London address
E-mail: service@elektor.com
www.elektor.com/memberships

Advertising & Sponsoring:
Johan Dijk
Phone: +31 6 15894245
E-mail: johan.dijk@eimworld.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. 2017
www.eimworld.com
Printed in the Netherlands



Creative repair work

I would argue with my learned colleagues in the labs that repairing or restoring a circuit can be as much fun as creating one from scratch, and the activity overall is equally educational potentially. Both activities are fraught with pitfalls and frustration but eventually should have the same moment of triumph as there is little difference between “hey it works” and “hey it works again”, and I’ve been either way.

The one comforting and reassuring thought when repairing a complex piece of equipment like a Tek 2235 ‘scope that’s as dead as a doornail must be this: it has worked for years and now there’s one or two parts in it thwarting its normal operation and hiding out for me. Which one is it? While hunting for that one flakey part (of 1202 in principle) I have to keep in mind all the time that the electronic design of the ‘scope will have imperfections (even the Tek), but not so many that it has never worked.

By contrast, sitting in front of his empty (CAD) screen, the creator of a new circuit has no other beacons to navigate by than his/her own intuition or strong conviction that the circuit will work. Some day — some year — possibly helped by others who were there before, successfully or not.

Both the creator and the repairman have to exercise their creative minds to overcome one obstacle after another and remain on the (hopefully) straight path to victory. After all, the relevant activities and skills are complementary rather than incongruent. Like so: many repair fans create fantastic workarounds to replace dead, obsolete parts and even manage to enhance existing circuitry; and the creative genius has to repair his/her project silently after a less fortunate choice of, say, a zener voltage or being “misguided” on the pinout of a BS170P.

Which leads me to the circuits published in this first edition of 2017: perfectly presented as they may seem — they are forever subject to improvement. It’s meant to be that way. Let me know how you fared.

Happy hooraying,

Jan Buiting, Editor-in-Chief

The Circuit

Editor-in-Chief:

Jan Buiting

Deputy Editors:

Thijs Beckers, Clemens Valens

Translators:

David Ashton, Jan Buiting, Martin Cooke,
Ken Cox, Arthur deBeun, Andrew Emmerson,
Tony Marsden, Mark Owen, Julian Rivers

Membership Manager:

Raoul Morreau

International Editorial Staff:

Thijs Beckers, Mariline Thiebaut-Brodier
Denis Meyer, Jens Nickel

Laboratory Staff:

Ton Giesberts, Luc Lemmens,
Clemens Valens, Jan Visser

Graphic Design & Prepress:

Giel Dols

Publisher:

Don Akkermans

THIS EDITION

Volume 43 – Edition 1/2017

No. 481 & 482 January & February 2017

- 21 electronica 2016**
Overview of what's hot
- 24 electronica Fast Forward Award 2016**
Who won the € 75,000 media budget plus a booth at electronica 2018?
- 26 ElektorBusiness: Simple Interfacing to Analog and Digital Position Sensors**
For industrial drive control systems
- 65 The Elektor Community**
- 112 Elektor Store**
- 128 Elektor World News**
- 130 Play & Win: Catdoku**
To kick in the New Year our traditional Hexadoku got a feline makeover.

LEARN

DESIGN

SHARE

- 6 Welcome to the LEARN section**
- 7 Peculiar Parts, the series**
The Fetron, a Solid-state Tube
- 8 Color LEDs**
Past, present and future of these colorful indicator lamps that have no filament or gas filling.
- 14 Tips & Tricks: Programming Adapter for USBasp**
A DIY adapter to program an ATtiny when it's mounted in a prototyping plug board.
- 15 CAD for PCBs**
An overview of a limited selection of currently available free PCB CAD programs.

LEARN

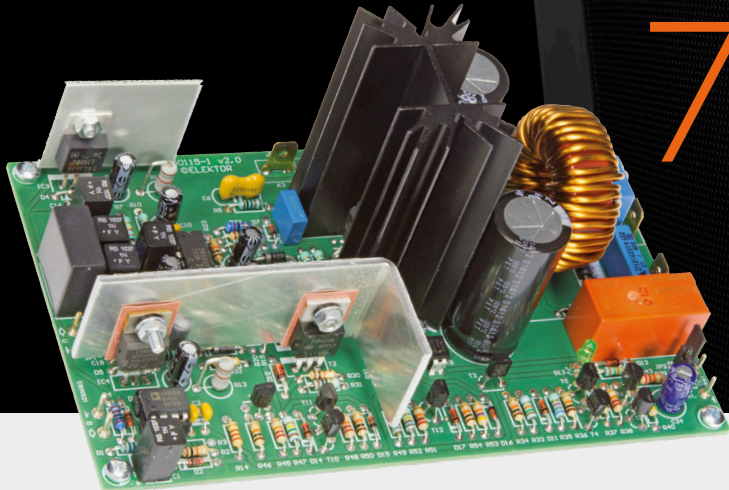
DESIGN

SHARE

- 30 Welcome to the DESIGN section**
- 32 Sand Clock**
This nifty gadget built around an Arduino Uno uses servos and a pantograph mechanism to write the time in a sand bed.
- 40 Super Simple Dice**
A cheap and microcontroller-free project designed especially for beginners!
- 42 Elektor SDR Reloaded (4)**
Can we also use the SDR Shield with just an Arduino and no PC at all? Certainly we can!

COOL POWER with D-WATT

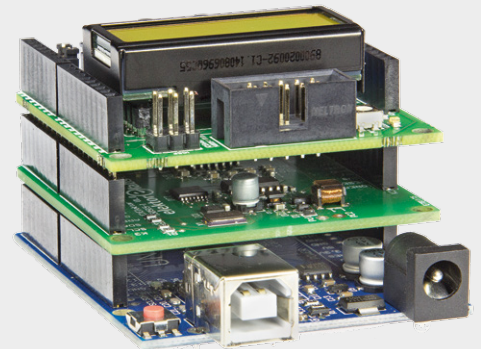
74



ELEKTOR SDR RELOADED: STANDALONE SOLUTIONS

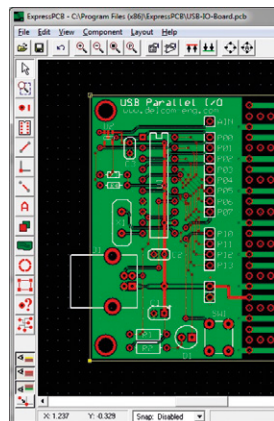
42

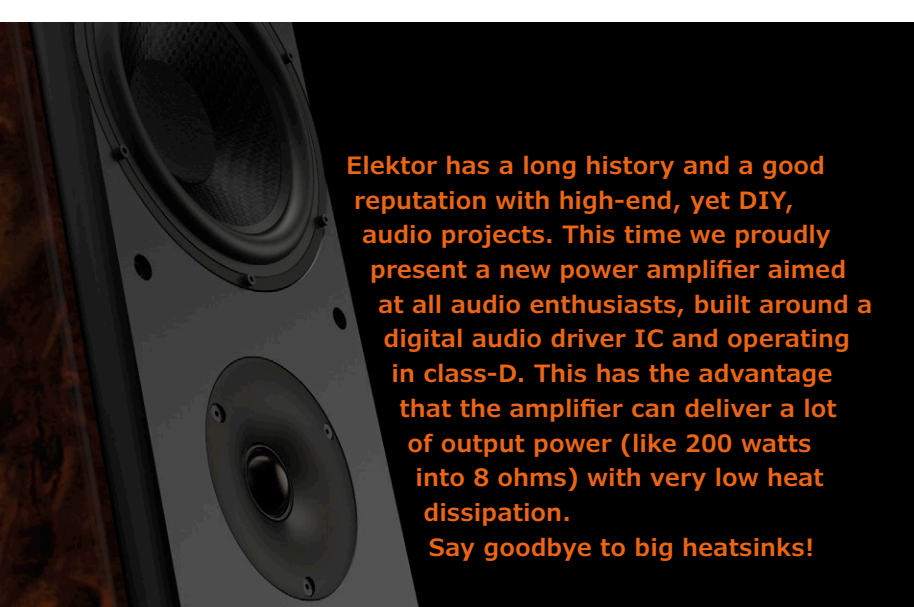
Can we also use the SDR Shield with just an Arduino and no PC at all? Certainly we can. It's entirely feasible (with a few restrictions) because all of the vital control functions are available on the Elektor Extension Shield. The only task that remains then is processing the IQ signals.



- 46 SUPRA 2.0 MC Version**
With some simple changes this design can be adapted for use with moving-coil (MC) cartridges.
- 50 Connect Objects with Genuino 101**
Make any device talk to your mobile.
- 56 BrainBox Arduino**
A sturdy Arduino with screw terminals, an on-board buzzer and a motor-driver IC.
- 60 LEGO® Control Board for the Raspberry Pi**
With 4 motor-control outputs and 16 digital I/O channels.
- 66 Debugging the Arduino Zero & M0 Pro**
Delving deeper into the world of Arduino.
- 71 Analog Random Numbers**
Using an Arduino and a noise source.
- 74 Cool Power with D-Watt**
A high-performance amplifier with lots of output power.

15





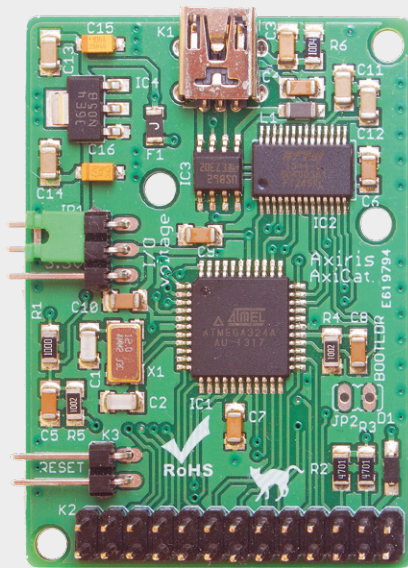
Elektor has a long history and a good reputation with high-end, yet DIY, audio projects. This time we proudly present a new power amplifier aimed at all audio enthusiasts, built around a digital audio driver IC and operating in class-D. This has the advantage that the amplifier can deliver a lot of output power (like 200 watts into 8 ohms) with very low heat dissipation. Say goodbye to big heatsinks!

- 114 Welcome to the SHARE section**
- 115 Simulation with SystemVision**
A free web-based tool.
- 118 Opamp Power Supply Connexions**
Capricious or logical?
- 119 Err-lectronics**
Corrections, Updates and Feedback to published articles.
- 120 The Mendocino Motor**
It spins and sways... by solar power alone.
- 122 Build a Chipamp!**
Great sound with only a few components.
- 124 Elektor Labs Pipeline**
A constant stream of projects and activities in the Labs.
- 125 Retronics: From Verobox to Heavy Metal**
Elektor lab instruments 1980s and 1990s style.

94

AXICAT, A VERSATILE USB DEVELOPMENT TOOL

If you are developing an app for an integrated circuit and you want a fast and easy way to link it to a Linux or Windows PC, we have the answer: AxiCat. This multi-protocol USB adapter can be used for communicating with ICs with an I²C or SPI interface, and you don't have to write any code to use it.



NEXT EDITION

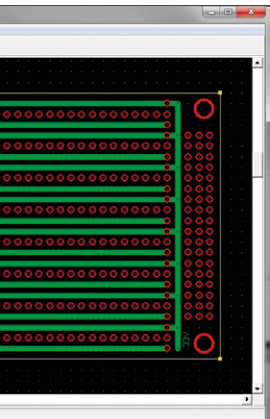
Heating Monitor using ESP8266
This heating monitor logs essential operating parameters, shows them on its built-in display and stores them on a USB stick for analysis later. The data is sent via Wi-Fi, so you check it out from anywhere around the home or via the Internet.

Swiss Pi Extensions
The Elektor 'Swiss Pi' expansion board from the September & October 2016 edition provides lots of useful functions for the popular Raspberry Pi. In this follow-up article we take a closer look at the actual use of the Swiss Server and provide detailed descriptions of demo programs in Python and PHP.

3D Printer Head Temperature Controller
Every type of 3D printing filament requires a different extrusion temperature to obtain the best results. This small device with an OLED display is an accurate extruder head temperature controller controlling the mat temperature as well.

- 85 Sensors make Sense (2)**
Some sensors have dual outputs, one analog and one digital.
- 92 MAXREFDES99# Display Driver Shield**
An easy-to-use 256-pixel solution capable of handling even the most demanding characters.
- 94 AxiCat**
A versatile USB development tool to link an integrated circuit to a Linux or Windows PC.
- 98 LED Stairway Lighting**
One controller for a whole bunch of LEDs.
- 101 Internet Radio with Fluorescent Display (2)**
The firmware for the display and the code for the Raspberry Pi.
- 106 IoT Shield for Arduino**
Lets You Do Your (Internet of) Thing.
- 108 DAB+ Antenna Diplexer**
Add digital reception to your car radio.

CAD for PCBs



Elektor Magazine edition 2 / 2017 covering March and April is published on 10 February 2017. Delivery of printed copies to Elektor Gold Members is subject to transport. Contents and article titles subject to change.



Welcome to the **LEARN** section

SHARE

DESIGN

LEARN



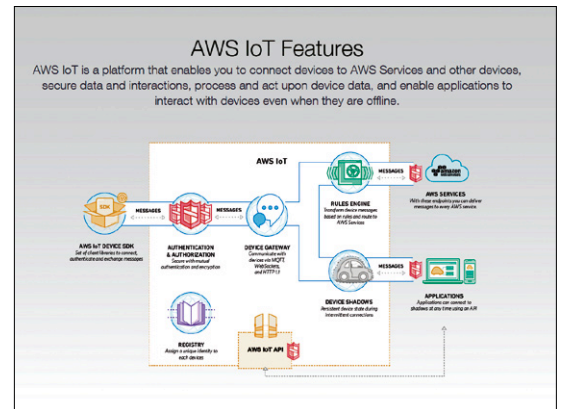
By **Jens Nickel**

We take a trip

The journey from Cologne to **e**lectronica in Munich takes a few hours by train. As usual the on-board Wi-Fi was distinctly flakey and there wasn't a reliable cell phone signal. It's at times like these you really appreciate the benefits of hard copy. There was plenty of time to browse the electronics magazines I had tucked into my bag before I left. A common thread in many of the articles was of course the IoT in general and IoT security in particular, which

are also the two main themes of **e**lectronica this year. I lost count of the number of development systems and boards from different manufacturers available — most were preconfigured to operate with one of the Cloud service providers. One name that always seems to crop up is Amazon Web Services provided by the leading Cloud service provider Amazon. After registration (restricted basic service is free) you can upload data from a sensor boards into the Cloud using a protocol such as MQTT. The board is represented by a virtual version or 'device shadow' in the Cloud where the data is stored. Now even when the sensor board is offline, the user still has access to all its stored data by using their smart device to connect to the shadow device.

My own experiences of developing just such an IoT application are detailed in my blog 'My Journey into the Cloud' which you can read at www.elektormagazine.com by typing the title into the Search box.



So much to see

We certainly didn't have any spare time on our hands at the Elektor stand during the Munich trade show. My first priority was to get better acquainted with the entrants taking part in the Fast Forward Start-up Award and explore their prototypes more closely (I wasn't entirely successful at this, there were just so many interesting things to see, the photo only shows a few of the entrants).

I also didn't get a chance to visit the stands of the major semiconductor companies. Luckily one of our freelance contributors and utter microcontroller fan Viacheslav Gromov was able to do that job for me and you can read his report in the LEARN section.

One I was determined not to miss was the Microchip/Atmel stand. While there I spied an interesting 'Power' debugger and also a kit, which — you guessed it — has been specifically designed for secure access with the Internet of Things (www.atmel.com/tools/at88ckecc-aws-xstk.aspx). On board there is a dedicated Atmel crypto engine preconfigured for generating signer certificates and registering them with — tadaa! — Amazon Web Services. I must admit to suffering a certain level of information overload when discussing the board with the Atmel experts, but there's no shame in admitting that, learning something new is half the fun. I already have a to-do list of topics including (Root-) certification, ECC and the entire login and authentication process required for using the AWS; it's going to be interesting... ◀

(160244)



The Fetron, a Solid-state Tube

Peculiar Parts, the series

It's shiny and tubular, 25 mm tall and 15 mm in diameter, with seven gold-plated pins at one end. It's clearly an active device but is it hollow-state or solid-state? The answer is both: it's a tube and it's solid-state, because it's a Fetron.

By **Andrew Emmerson** (United Kingdom)

Actually it's a tube substitute, entirely solid-state but in the form factor of the tube that it replaces. It uses cascoded JFETs (junction field-effect transistors) to duplicate the transfer characteristics of specific types of vacuum tubes (mainly pentodes but some triodes were also made). Introduced by Teledyne Semiconductor in 1972 and intended for military applications and other 'high utilization' equipment (not consumer electronics), the Fetron was able to extend the service life of expensive tube apparatus, deferring the cost of its replacement. Because Fetrons were entirely solid-state, they consumed less than half the power of vacuum tubes and slashed the cost of air conditioning. The need for frequent adjustment and periodic tube replacement was eliminated and despite their eye-watering price, Fetrons could repay their cost within six months of installation. Teledyne developed the Fetron for the direct plug-in replacement of several types of existing vacuum tubes, but herein lay their weakness. The *connoisseur* of Fetrons, Dr Hugo Holden of Queensland, Australia, explains that marketing Fetrons as tube substitutes was a fatal error. In many applications they were not at all ideal, for example in circuits such as local oscillators that rely on grid current automatic bias. A Fetron malfunctions in this application and draws gate current, he states, causing device saturation and introducing severe harmonics into the output. Nevertheless, a 'tube' that generated no heat, would not drift,

and drew zero filament current was an attractive proposition. Telephone companies took a particular shine to them and the examples seen in **Figure 1** came from New Zealand, where they were used in the launch amplifiers for the sub-sea telephone cables between that country and Australia. Other applications were in high-spec. broadcast equipment and guitar amplifiers (!), whilst Teledyne marketed special tube conversion kits for upgrading vacuum tube voltmeters and oscilloscopes by Hewlett Packard and Tektronix to give these the same stability, low-noise characteristics and reliability as high-quality bipolar transistors (**Figure 2**). When the time came for users to finally replace their equipment with solid-state, the need for Fetrons vanished to leave only a few memories of this ingenious but short-lived device. ◀

(160265)

For further reading

<https://en.wikipedia.org/wiki/Fetron>

<http://hrsasa.asn.au/docs/Fetron.pdf>

www.philipstorr.id.au/radio/eleven/fetron.htm

www.worldphaco.com/uploads/WORLDFETRON.pdf

www.elektormagazine.com/magazine/elektor-201609/39810

Please contribute your Peculiar Parts article,
email neil@gruending.net

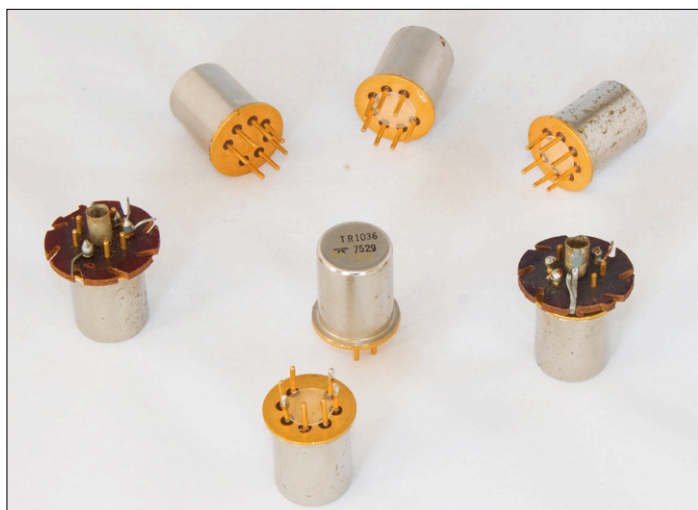


Figure 1. Fetrons from Down Under.



Figure 2. Period product advertising.

Color LEDs

Past, present and future

By **Dr Thomas Scherer** (Germany)

We have known for a hundred years how to get certain inorganic materials to emit light when a current is passed through them, and LEDs themselves have existed for some fifty years. These colorful indicator lamps that have no filament or gas filling have been gradually refined and have become available in more varied and vivid colors. In recent years their efficiency has improved to the point that LED technology is replacing even fluorescent lighting. The most recent developments include LEDs that can replace LCDs.

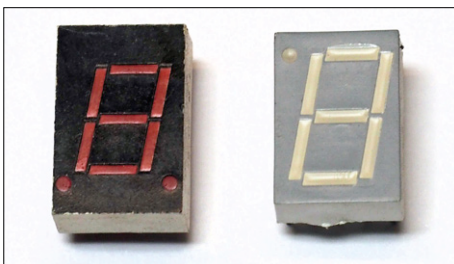


Figure 3. From the antiques collection: a red seven-segment LED display with a character height of 10 mm.



Figure 4. The first mass-market pocket calculators used tiny red seven-segment LED displays. (Source: Erhaka, *Wikimedia Commons*).



After the invention of the crystal rectifier by Karl Ferdinand Braun in 1874 it was another 33 years before Henry Joseph Round discovered that the point of contact between the metal wire and the crystal could be made to emit light, yellow, red or even blue depending on the applied voltage. Round was therefore the original inventor of the LED and the electroluminescent effect in semiconductors that he discovered has been named the 'Round effect' in his honor. And so our story begins.

A slow evolution

The first modern-style LED was made at RCA. It produced invisible infrared light. That was in 1955, and the inventor was called Rubin Braunstein. Texas Instruments (TI) has the honor of being the first mass producer of LEDs, starting in 1962. TI's infrared LED had an efficiency of a mere 1.1%! The first visible red LED was made by General Electric (GE), also in 1962; there was a surprisingly long gap of ten years before the first successful yellow LED was made, again at GE. Through the 1970s red LEDs gradually improved. Efficiencies approached 5% and they started to be used as indicator lights. Devices typically came in round 3 mm or 5 mm plastic packages and used only about 40 mW of power. They had many further advantages over bulbs with glowing tungsten filaments, including

being insensitive to vibration and having a much longer life. Neon lamps, with their working voltages of 70 V to 100 V, also went the way of the vacuum tube: LEDs, operating from voltages of 1.2 V to 4 V (see the text box **Voltage and Color**) worked much better in transistor circuits. Also in the 1970s seven-segment displays began to appear using (red) LED technology: see **Figure 3**. Some readers will be able to remember the first pocket calculators like the one shown in **Figure 4**, which used tiny red displays that often consumed more power than the chip doing the actual calculating. My TI-59 from 1977, one of the first programmable calculators, had many more functions than that example, but still only had the same type of minuscule display. And it took such a long time for this schoolboy to save up for it!

Three different colors of LEDs were available in the 1970s: red, yellow and green. The upper row in **Figure 5** shows (ignoring the blue LED on the right for the moment) some antiques dating from this period. They were in general not particularly bright, the shades of red and yellow light emitted varied from device to device, and the green color was not exactly convincing. **Figure 6** shows the brightness and color variation in green LEDs from that time: all four LEDs are wired in series, with a current of 5 mA flowing through them. The LED at the

Figure 5. A LED miscellany. Top row, from left to right: red, yellow and four different green LEDs of 1970s vintage, compared to a modern blue LED which is so bright that its light saturates the camera sensor. Bottom row: modern green LEDs in 10 mm and 20 mm packages, and RGB, white and infrared LEDs.

Voltage and Color

Inorganic LEDs are based on doped semiconductors, and often contain elements such as aluminum, arsenic, gallium, indium, phosphorus and nitrogen. More recently research has turned to LEDs employing carbon (in the form of diamond), silicon and zinc. The composition, concentrations and construction affect not just the efficiency of the LED, but also have a direct effect on the color of the emitted light. The energy of the photons emitted by a material depends on what is called the bandgap, which is the energy difference between the conduction band and the valence band of the semiconductor in question. The bigger the bandgap the higher the energy of the photon and the shorter the wavelength of the light. The bandgap has a big effect on the threshold voltage, which is the voltage that has to be exceeded before current will flow and before electrons will jump between the conduction and valence bands and so emit photons. The theoretical relationship is shown in **Figure 1**. It is now clear why the forward voltage of a blue LED is around twice as high as that of a red LED, as the frequency of the light is proportional to the photon energy.

Table 1 shows which semiconductor materials are used to produce which colors, along with the corresponding wavelength ranges and the forward voltage you can expect. Now, the actual voltage you might measure across the terminals of a real LED depends of course not just on the bandgap of the materials used but on many other details including the resistance of the materials and bonding wires and the semiconductor junctions. The voltage across the LED will also increase with increasing current. The reason that the transition between current not flowing and current flowing is not perfectly sharp is that in the transition region electrons behave statistically, having a certain probability of overcoming the bandgap which rises with increasing voltage. The spectrum of the light from an LED also moves slightly towards longer wavelengths ('warmer' colors)

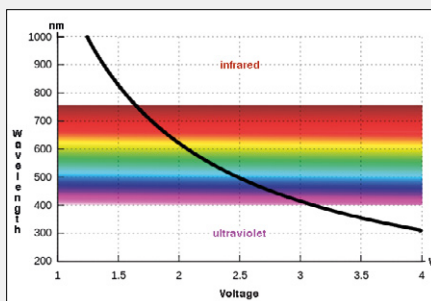


Figure 1. The relationship between an LED's forward voltage (bandgap) and light color. Voltage and wavelength are in inverse relation.

with increasing temperature: this is not a surprise, as the forward voltage drops with increasing temperature. If a particular application requires a stable color, the die temperature of the LED must be kept in mind.

The purity of color of the light from LEDs lies between that of (colored) incandescent bulbs and lasers. In other words, LEDs are not as narrowband as lasers, which emit light of practically a single wavelength, nor as wideband as incandescent bulbs. **Figure 2** shows that, particularly at longer wave-

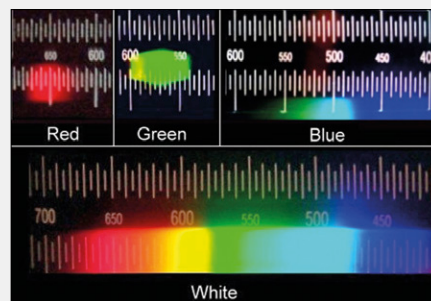


Figure 2. Spectra of various colors of LED. The shorter the wavelength, the more broadband the spectrum. Below, for comparison, the spectrum of sunlight. (Source: Wikimedia Commons).

lengths, LEDs are reasonably narrowband. The spectrum of blue LEDs, on the other hand, is somewhat broader. This characteristic is used in white LEDs that consist of blue LEDs plus a phosphor layer: the result is a more uniform spectrum and hence a more natural (that is, sunlight-like) light. The quality of light from a white LED is characterized by its 'color rendering index', or CRI, R_a . Values are specified relative to sunlight, which has a CRI of 100; white LEDs manage CRI values of between 75 and 95.

Table 1

Color	Wavelength (nm)	Material	Voltage (v)
infrared	over 760	GaAs AlGaAs	less than 1.6
red	610 to 760	AlGaAs GaAsP AlGaInP GaP	1.6 to 1.9
orange	590 to 610	GaAsP AlGaInP GaP	1.8 to 2.2
yellow	570 to 590	GaAsP AlGaInP GaP	2.0 to 2.4
green	500 to 570	InGaN GaN GaP AlGaInP AlGaP	2.2 to 2.7
blue	450 to 500	ZnSe InGaN SiC	2.6 to 3.3
violet	400 to 450	InGaN	3.2 to 3.6
ultraviolet	230 to 400	AlN AlGaN AlGaInN	3.5 to 4.2

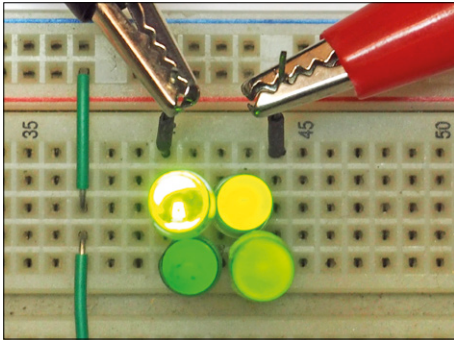


Figure 6. Four different green LEDs: each emits a different light. The LED at the top left is a modern 'superbright' device, which saturates the camera sensor. The other 5 mm LEDs are the samples from Figure 5.

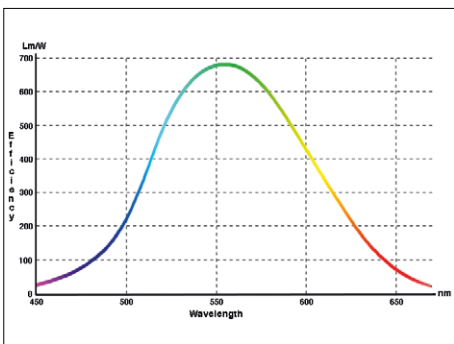


Figure 7. Maximum theoretically possible light output from an LED as a function of color. Note that the lumen is a unit which takes into account the sensitivity of the human eye to different colors.



Figure 8. An LED 'candle lamp'. The LED strips in this E14 bulb mimic traditional filaments. (Source: OSRAM).

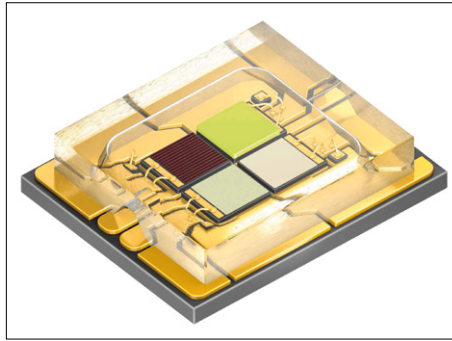


Figure 9. More than an RGB LED: this OSRAM OSTAR LED in an SMD package contains a white LED chip in addition to chips in the primary colors.

upper left is a modern high-efficiency type for comparison purposes: it is so bright that it saturates the sensor on the camera. One of the old-school LEDs is more yellow-green than true green. Red and yellow LEDs suffered from similar weaknesses. Fortunately modern LEDs in 5 mm or SMD packages have beautifully deep colors and are so efficient that, when used as indicators, the current through them must be carefully limited to avoid dazzling the user.

Status quo

There was an even longer gap before LEDs in the previously missing blue arrived. They first came on the market only in the 1990s, produced by Nichia. The development of the blue LED was judged worthy of a Nobel prize for its inventors Shuji Nakamura, Isamu Akasaki and Hiroshi Amano in 2014, a huge honor for such a small device. The award of the prize recognized the fact that high-efficiency blue LEDs are fundamental to making white LEDs, where a phosphor layer is added to the LED which absorbs a portion of the higher-frequency blue light and converts it into yellow light. The combination of this yellow light with the remaining blue light yields white light through additive mixing. This idea was developed by Jürgen Schneider at the Fraunhofer Institute and now forms the basis for almost all LEDs used in lighting applications. Today white LEDs have a slightly greater market share than color LEDs. You can read more about white LEDs in my article 'Let There Be LED' in the January/February 2016 issue of *Elektor* [1].

At the upper right of Figure 5 you can

see a blue LED in a transparent plastic package, operating at 5 mA. It is so bright that it saturates the camera's sensor and so misleadingly appears bluish-white in color. At the bottom left are green LEDs in 10 mm- and 20 mm-diameter packages. The larger of these is a multi-chip LED containing four LED chips connected in series. Next to it on its right is a common-anode RGB LED which therefore has four leads. The three primary-color LED chips in this device can be driven in such a way as to create a mixture of practically any desired color. The transparent LED to its right is a modern white device, and on the far right is an infrared LED. The package of this device is transparent to infrared light.

From the meager levels of the seventies the light output of LEDs began to increase rapidly, especially at the beginning of this millennium, reaching efficiency levels of 100 lm/W. This puts white LEDs on a level where they can compete with fluorescent lamps, matching them for light output while offering the advantages of a longer service life and zero mercury content. In the laboratory, meanwhile, white LEDs have been made with output efficiencies exceeding 300 lm/W, just a few percent short of the theoretical limit of 350 lm/W at a color temperature of 6600 K. **Figure 7** shows how the maximum theoretically possible light output depends on the color of the LED. There is a nice paradox here: although the blue LEDs on which white LEDs are based have an efficiency of no more than 200 lm/W, the white LEDs offer a higher efficiency. The explanation for this is that the way the 'lumen' is defined takes into account the varying sensitivity of the human eye to different colors: green light at a given power looks brighter than other colors. And so, although when generating yellow light using the phosphor a little energy is lost (in fact, converted into heat), the white light looks brighter to us than the blue light directly from the chip under the phosphor would.

It is worth noting, then, that the potential for further refinements in the lab is almost exhausted as there is very little scope for further efficiency gains. There is still progress to be made, however, in converting these prototypes into real products. There is plenty of room for improvement in mass production processes to enable the manufacture of devices matching the performance achieved in the lab at rea-

sonable prices. This will no doubt happen within the next decade, and then we will have at our disposal a wonderfully energy-efficient lighting technology based on modern semiconductors which, thanks to its excellent service life, will also have a relatively small carbon footprint.

Color LEDs

In recent years LEDs have reached such high efficiency and, thanks to mass production, low prices, that they have by and large completely superseded filament lamps. And LED lamps have moved on from the ungainly units seen in the past containing a large number of SMD chips: these days you can buy bulbs whose internal LED structures mimic the filaments of traditional slender candle-shaped filament bulbs (see **Figure 8**). So now it is possible to populate your chandelier with modern energy-efficient bulbs while still remaining faithful to its style. LED lighting is now available in practically any shape, size and color you can imagine: we shall return to this topic later.

Infrared LEDs have of course been used for many years in remote controls, as well as more recently in providing unobtrusive night-time lighting for security cameras or for military purposes. Another interesting application is the use in industry of infrared lamps consisting of hundreds of SMD LEDs with a total output power of over 100 W as a component in projectors to shine a grid of infrared illumination on one part of a production line or on an assembly station. The idea behind this is that, unlike a human operator's eyes, a robot's cameras can easily see the projected grid and hence orientate itself in space and better estimate the position of units on the production line. Humans working alongside the robots or monitoring their operation are not disturbed by the infrared light.

White LEDs are not the only way to generate white light: a more expensive, but equally good, if not better, alternative, is to use RGB LEDs. These consist of LED chips in (at least) the three primary colors mounted in a single package. By passing different currents through the different chips the outputs from the individual LEDs can be mixed at will to produce practically any desired color. **Figure 9** shows an example, an OSTAR SMD LED, which includes not only chips with red (625 nm), green (530 nm) and blue (453 nm) LEDs, but also a white LED. It is designed for

use in specialist lighting applications such as providing precise illumination for surgical operations and other uses where color accuracy is critical. Despite its small size, this SMD device can output over 500 lm of light at its nominal current of 1.4 A. Readers who like to strut their stuff occasionally will already be aware of the widespread usage of LEDs in discos and clubs. Stage spotlights, or 'PAR cans' as they are sometimes called, used to be incandescent lamps rated at several hundred watts equipped with colored filters. Nowadays they have been largely LEDified with all-electronic control (see **Figure 10**). Although the LED versions are not yet quite the equal of the old technology in all respects, the availability of ever brighter LEDs is closing the gap. Service life, robustness, much lower heat generation, reduced risk of injury (stage spotlights are prone to explosion) and instant electronic control all weigh in favor of the LED versions.

Colored LEDs don't have to be used in conjunction with music, of course. At an early stage Philips realized that even Joe and Janet Bloggs would be keen to have configurable mood lighting in their home. Under the 'LivingColors' brand the Dutch electronics giant has been marketing a range of table and wall lights for a few years now (see **Figure 11**). These remotely-controlled LED lamps can be adjusted over a wide range of colors. Each lamp contains a number of color and white LEDs that can produce static mixed colors or flashing patterns, and the lamps can also be dimmed. The success of this range led Philips to produce the 'Hue' range (**Figure 12**). This brand includes not only lamps, but also LED strips and 'normal' colored bulbs for E27 fittings, all remotely controlled using a 'bridge'. A range of lamps with built-in lithium batteries is also available under the 'Hue Go' brand. These lighting systems have been highly profitable for Philips and so doubtless further variants will also appear on the market. If you are looking for a more economical alternative, you will not be disappointed: the DIY sheds have spotted the opportunity and offer shelfloads of inexpensive color LED lighting of all kinds (**Figure 13**), costing just a fraction of the price of the big-brand products. For an even cheaper solution, you can import directly from China using eBay or Alibaba. In this case, of course, you have to reckon with the complications of dealing



Figure 10. Cameo type CLP56RGB05PS PAR stage effect light with 151 colored LEDs and a power rating of around 30 W at 230 V. Not only does the light sport a DMX interface for remote control, it also has a built-in microphone to enable 'light organ' effects.



Figure 11. Philips LivingColors: this lamp from the highly successful range contains remotely-controllable colored and white LEDs in a plastic enclosure.



Figure 12. The Hue series from Philips includes colored LED lamps with E27 fittings and a 'bridge' for remote control



Figure 13. These days you can find a huge selection of LEDs strips, bulbs and effect lamps with or without remote control in any DIY shed.



Figure 14. Google Nexus 6p: the OLED display on this reasonably-priced smartphone has a resolution of 2560x1440 pixels.

with possible customs charges, and of course always bear in mind that cheaper does not necessarily mean better value (see the text box **Feel the Quality**). Naturally ultraviolet LEDs also have their uses. As well as specialist applications there are uses such as exposing the photosensitive lacquer on printed circuit

boards or building 'black lights'. Another application for ultraviolet LEDs is illuminating the water flowing through a transparent pipe, with the aim of reducing the growth of algae in aquaria and swimming pools. Some varnishes can be cured using ultraviolet light, and the LEDs can also be used to test the resistance of products to exposure to ultraviolet light.

OLEDs

Organic LEDs (OLEDs) were discovered much later than inorganic LEDs, and so far cannot compete with them in terms of light output or long-term stability. However, their star is in the ascendant: the advantages of OLEDs are low-cost raw materials and, in principle, lower cost of manufacture, as they do not require such strict 'clean room' conditions. OLEDs can also be made in flat sheets: this is an advantage in many lighting applications, as the sheets do not have to be rigid. Flexible lamps are therefore a possibility. For a long time it has proved difficult to make OLED products outside a laboratory

environment, but that is now changing. OLEDs can very easily be used, for example, to make displays, as printing technology allows the organic light-emitting pixels to be manufactured along with their interconnect wiring. RGB OLED displays are bright and contrasty. Similar approaches to manufacturing displays with semiconductors are much more complicated and decidedly more pricey. OLED displays are increasingly competing with LCD displays and in principle the OLED beats its liquid-crystal counterpart in virtually all departments. There is no backlight or any of its attendant problems, as the OLED pixels are themselves the source of light. Current consumption is also lower, as much of the light from an LCD's backlight is lost in the diffuser and in the polarization filters. Colors that change with viewing angle are a thing of the past with OLEDs. And last but not least the contrast ratio of an OLED display cannot be beaten: when an LED is off, it is off. Blacks really are black. Given all these wondrous advantages you

Feel the Quality

LED lamps, containing a large number of individual LEDs to provide sufficient light output, have been available for some time. Since the mid-1990s car makers have routinely used a string of red LEDs for the 'third' brake light, even on mid-range vehicles: the red LEDs of the time were already bright enough and cheap enough. Customers were happy too, as the light did not need replacing as often over the life of the vehicle as an incandescent version. Brake lights are an important part of vehicle safety, and so not failing after a few hundred operating hours is a significant benefit. All very warm and fluffy, but cracks start to appear in the story when it comes face to face with reality.

More specifically, two phenomena come into play, which together make the situation far from ideal. The LED brake lights do indeed have a huge 'theoretical' service life of up to 50000 hours. But in the vehicle environment that does not represent much of a guarantee, what with vibration, bad weather and high humidity levels (which can lead to corrosion of printed circuit board tracks, for example), and extreme temperature variations. These lead to mechanical stress on the LED packages, chips and bond wires. As a result the quoted service life of 50000 hours is a pipe dream. The car's bodywork can get very hot in sunshine, and the heat dissipated by the LEDs adds to this to generate very high operating temperatures. The consequence is that LEDs in automotive applications do indeed last longer than incandescent bulbs, but not by the large factor that you might have hoped for. In addition to this there is the 'multiple LED effect': even under optimal conditions, when many LEDs are in use it is statistically inevitable that the time for which all LEDs will be working will be reduced.

Correcting for this effect, a lamp consisting of ten LEDs might have a service life of only 15000 hours rather than the hoped-for 50000 hours. And, if in the real-world vehicle environment, the life of an individual LED falls to only 10000 hours, one LED in the ten-LED lamp might easily fail after only 3000 hours. But that is not the end of the story: in vehicle lights that use LEDs designed for a 12 V supply, the LEDs are often wired in series strings of two or three devices each, and so the failure of one LED leads to the loss of light from one or two more. Another problem is that the LEDs in vehicle lights are often hermetically sealed and so are not replaceable. Instead of just replacing a two-dollar bulb the whole lamp assembly has to be taken out and replaced.

These are not mere academic considerations, but reflect my own personal experience: the brake light on my 1996-model Fiat failed in just the way described above after only three years, and a replacement part plus labor was not a cheap proposition. And in case you are thinking 'well, that's Fiats for you' I should say that, despite the alleged reliability of Toyota vehicles I had an LED fail in exactly the same way on my Prius. In this case it was part of the left-hand rear light cluster, which contains a large number of integrated LEDs all firmly fixed in position. At eleven years (in fact, at the same time as the brake disks needed to be replaced) one of the lights failed. Fortunately a third-party replacement part was within my means, but still not cheap. If the LEDs in the headlight on a modern vehicle should fail, you are certainly looking at a tall bill from your garage. Modern technology comes at a price! This disadvantage of multi-LED clusters becomes a significant problem when they are used for domestic lighting. Although humidity and temperature fluctuations play a lesser role in this environment, there is still the question of the quality of the com-

would be forgiven for asking why OLED displays are not more widespread. The problem lies in the extremely demanding requirements of the manufacturing process. The many LEDs in a single display must all be equally bright (within rather narrow limits), and must stay that way over time. The consequence is that as little as two years ago even small-screen TVs were unrealistically expensive. However, towards the end of 2016 LG, the market leader in OLED TVs, introduced a model with a 55-inch diagonal for a reasonably affordable US\$1500. Smartphones with OLED displays have been around for rather longer, and the price difference for small displays compared to LCDs is now relatively small: the moderately-priced Google Nexus 6p, produced by Huawei (**Figure 14**) contains a 2560x1440 pixel panel with some 11 million OLEDs. You might wonder why no iPhone has been produced using an OLED display: the answer is that, as of 2016, there is insufficient manufacturing capacity in the world. Capacity is steadily

being increased, however, and the rumor mill has it that there may be sufficient capacity by the end of 2017 for an OLED panel to appear at least on the higher-end versions of the iPhone 8.

The future

Besides the rumors of an iPhone 8, OLED panels will shortly be found in all kinds of gadgets and appliances that need to display something. It is no exaggeration to say that pretty much every other kind of display technology will soon have had its day. OLEDs will also make inroads into lighting applications: when that starts to become economically feasible, many lighting manufacturers will jump on the bandwagon. Lamps for conventional fittings will remain the domain of semiconductor LEDs, since for the foreseeable future OLEDs do not offer the light density required to provide adequate illumination from a small bulb. Inorganic LEDs will also supersede practically all other lighting technologies except in niche applications with specialist requirements.

This is because total operating costs are much lower than other technologies, not just because of their greater energy efficiency, but because they require less frequent maintenance as a result of their longer service life. Street lighting is one application where soon there will be no alternative to LEDs. LEDs could allow the construction of intelligent car headlights that automatically adjust the shape of the light cone away from oncoming vehicles to avoid dazzling their drivers; perhaps manually dipping and undipping the headlights will no longer be necessary. We should be so lucky!

(160246)

Web Links

- [1] 'Let There Be LED':
www.elektormagazine.com/150577
- [2] www.youtube.com/watch?v=Oj8RIEQH7zA

ponents used. Low-cost LED strings from the Far East used for Christmas decorations or for fun are of course fine, but in many cases it is hardly possible to believe the price of a product on offer. A whole string can often be sold for less than the total cost of the LEDs it contains. How do they do it? The easiest reply to this question is usually 'at the expense of quality'. And aesthetic considerations are no determinant of when an LED becomes an ex-LED [2]; even my Christmas displays, which are of course done in the best possible taste, have on occasion suffered from LED strings (or at least parts of them) going to meet their maker.

Low-cost LED strips with a self-adhesive backing, which are powered indirectly from a low-voltage supply and which are often used for illumination in furniture and similar applications, suffer from the same problem. My personal tribulations involve two LED strips, each 2 m long, stuck to a length of aluminum section: a total of 120 LEDs. My estimate for the time to the first failure had been of the order of 4500 hours, but after just six months and perhaps 600 hours of operation eight LEDs had already kicked the bucket! I desoldered some replacement SMD LEDs

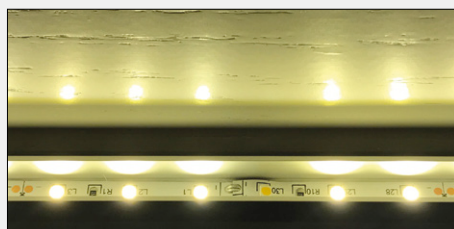
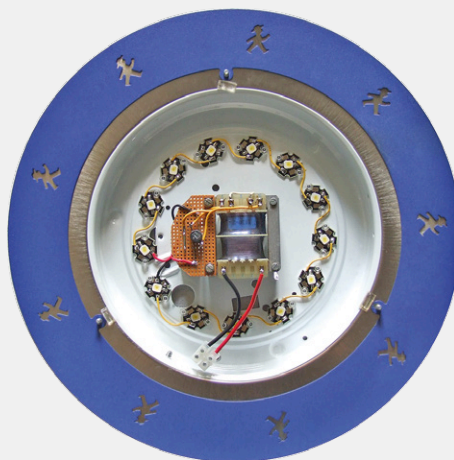


Figure 15. A failed LED strip. It is perhaps partly my fault that LED L30 has gone to join the choir invisible: because of the kink in the strip it was not being properly cooled, and it has failed to a short-circuit rather than an open-circuit. Other LEDs in the same strip, however, seem to have met their demise without such a good excuse.



of the same color from a spare strip and mended the original strips, but after another two months two more LEDs were pushing up the daisies (**Figure 15**), and no doubt this is not the end of my woes. Regular LED lighting can also often have shorter life than expected. LED bulbs have not been around for all that long, but nevertheless I have had four bulbs with E27 fittings and two GU10 spotlights bite the dust. In the case of the E27 bulbs it was the switching supply circuit whose coil proved all too mortal, whereas in the case of the GU10 spotlights the LEDs themselves became history after exposure to excessive temperature. In the case of two LED tubes bought as replacements for fluorescent lights the engineers at LG had put a fuse with too low a rating in the integrated switching power supply circuit: uprating the fuse fixed the problem and the tubes still work today. Also, my home-made LED lamp (**Figure 16**), which uses 13 high-quality LEDs, is still going strong after more than a decade.

Figure 16. The first 'Scherer-brand' home-made LED lamp was made in 2005, and continues to run to this day, after over 10000 hours of operation.

Tips and Tricks

From readers for readers

Here is another neat solution sure to make the life of any electronics enthusiast that much easier.



Programming adapter for USBasp

From Antonello della Pia

I think it's a great idea to provide a forum where we can share ideas with the rest of the electronics community (not forgetting the chance to win £50!). One of my all time favorite microcontrollers is the ATtiny85, which I usually program using a simple and low-cost USBasp programmer.

Sadly though it is not so simple to connect the pins from programmer to the ICSP pins of the ATtiny when it's mounted in a prototyping plug board.

Try this adapter I made to make the job easier. I started with a turned-pin 8-way DIP IC socket and a SIL 6-way male/female header connector; I then separated a six wire-wide strip from an old 9-way ribbon cable (originally used for the serial interface). The six wires supply the signals MISO, MOSI, SCK, RESET, GND and VCC (when you want the

programmer to power the board). These are then soldered to the corresponding pins on the 8-way socket. The pin header is connected to the other end of the cable.

Now, where does this go? The cables cannot pass under the socket or cross so I made up another cable fitted with a female header to make the necessary connections to the USBasp. Check out the

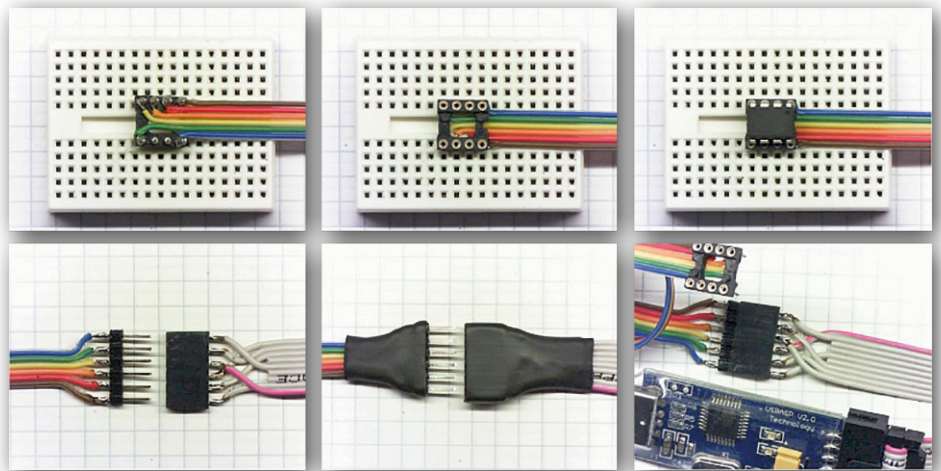


Table 1. Programming pin assignment (ICSP) for the ATtiny85.

Signal	Pin	Pin	Signal
RESET	1	8	VCC
nc	2	7	SCK
nc	3	6	MISO
GND	4	5	MOSI

pin number correspondence between the 8-way socket and the USBasp header (Tables 1 and 2)!

It should be clear from the photo how it all goes together. I also found another use for the adapter: when there's no space or you don't want to fit an ICSP header to a board just for programming, this adapter can be used as an intermediate socket between the ATtiny and its socket on the board for programming purposes.

Table 2. USBasp connector pin assignment.

Signal	Pin	Pin	Signal
MOSI	1	2	VCC
GND	3	4	TXD
RESET	5	6	RXD
SCK	7	8	GND
MISO	9	10	GND

Using the same principle, it should be possible to make a similar sort of adapter for other makes of controller also. ◀

(160277)

Have you come up with an inspired way of solving a really challenging problem? Or found an ingenious but 'alternative' way of using some component or tool? Maybe you've invented a better or simpler way of tackling a task? Do write in – for every tip that we publish, we'll reward you with UKP40 (or local equivalent)!



CAD for PCBs

Free and non-free PCB layout software

By **Harry Baggen** (Elektor Netherlands) and **Dr. Thomas Scherer** (Germany)

Hardly anyone who works with electronics, whether professional or enthusiast, is content to leave their design at the breadboard stage. Particularly with SMD components or relatively complex circuits, this is anyhow difficult, and prototyping boards are not the ideal solution. There is no getting around a real PCB in such cases. For that you need suitable software, and to choose the right software, you need information.

PCBs are essential nowadays, even for very small production runs or one-offs for the lab in a company of any size or on the workbench of a hobbyist, not only because they look nice but also because the stability and reliability of electronic circuits are strongly dependent on how well the circuits are built.

Unfortunately, it costs time and effort to learn how to make PCBs with a CAD program. In addition, the files these programs generate for schematics and PCBs are often specific to the program concerned and difficult or impossible to import into other programs, although this does not apply to files in standard production formats such as Excellon. Different programs also often offer very different features and functions. A schematic editor and layout program are indispensable basic components, but some packages also feature integrated simulation or 3D visualization of the PCBs, which can be helpful for choosing an enclosure. The program resources are also significant, especially the number of components in the libraries. Although you can always create your own components, that takes a lot of time and effort. The CAD packages intimately associated with component distributors score especially well in that area.

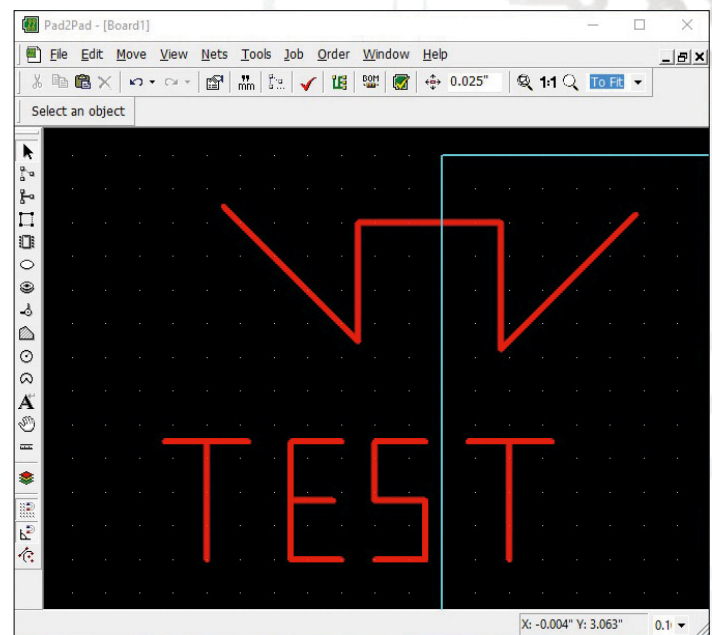
Below we provide an overview of a limited selection of currently available PCB CAD programs, to make it easier for you to choose a particular package or to whet your curiosity.

Pad2Pad (version 1.9.111)

Pad2Pad is a PCB manufacturer which specializes in supplying custom PCBs via the Internet. With their free Pad2Pad software you can lay out PCBs fairly easily and send the data to the manufacturer for PCB production, but unfortunately there is no support for schematic generation.

After starting the program, you first have to enter the PCB attributes, including the number of layers and the anticipated number of boards. Then you start the layout process by placing the components. The program has a fairly large library for this, and if a particular component or pad footprint is missing, you can create it yourself. Unfortunately, you cannot switch between metric and inch units on the fly; this must be specified in advance for virtually every window.

There are also quite a few templates available. You can use them to correctly number a connector in one go or copy a good layout for a board type you already know, such as an Arduino



shield, to your own PCB.

Unfortunately the program crashed while we were trying it out, causing an error report to be sent to the manufacturer. Importing dxf files generated by Eagle (version 6.4) sometimes caused problems. There is clearly room for improvement here. After registering as an Pad2Pad user, you receive an email every day with a link to a tutorial video. We found this helpful. All of the documentation is available online, including the tutorial videos.

The manufacturer is constantly improving the program, which means that you receive updates at regular intervals with new functions and bug fixes. Pad2Pad runs under Windows XP or later.

gEDA

As the name suggests, gEDA is a collection of electronic design automation tools. It is published under a GPL license and is available for Linux (SUSE and Debian) as well as OS X. At first glance it looks very promising, but the program versions we tested (*gschem* 1.8.2 and *PCB* 20140316) have a fair number of small problems. As a result, we often had to use the Help

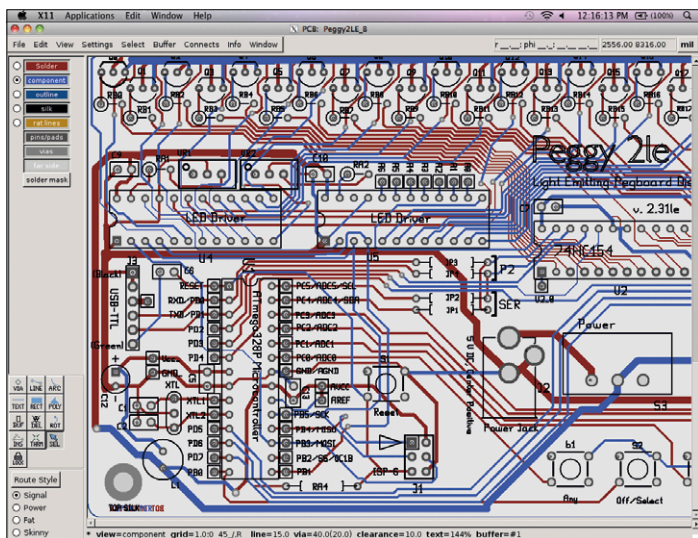


Figure 2. gEDA running under X11 on a Mac.

menu to consult the documentation, the FAQs and the Wiki. However, once we learned how to use it and became familiar with the keyboard shortcuts, we were able to generate schematics quite nicely with the editor. Here we should mention that the suite does not run under the normal GUI on the Mac, but instead under the X11 X-window system, which detracts from more than just the look and feel.

Unfortunately, things go downhill from there. To generate a PCB from the schematic, you first have to manually link each component to its package. That is possible with the schematic editor, but not with the attribute editor. It also helps if you know all the package names by heart, because there is no visual support for this.

After all the packages are linked, you can import the schematic into the PCB editor. There all the components land in a heap. You then create the desired PCB layout by distributing them to where they belong. If you subsequently change a component package in *gschem*, you can use the command *gsch2pcb project* to import the change into the PCB editor. You enter this

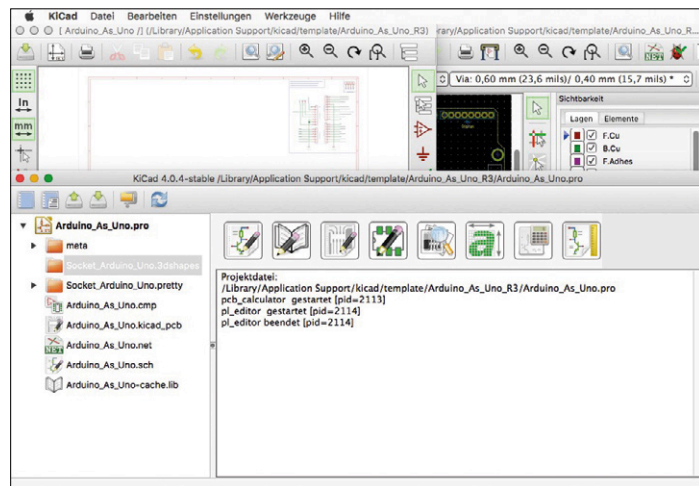


Figure 3. All in one: the KiCad open source package.

command in the Terminal window, with "project" replaced by the name of the actual project.

KiCad

KiCad is also an open source electronic design automation suite, with the relatively mature version number 4.0.4. The high level of user community support for this software project can be seen from the periodic software updates as well as the fact that finished distributions are available not only for Windows and in some cases OS X, but also for the Linux derivatives Ubuntu, Debian, Mint, Arch, Fedora, openSUSE, Snappy and Gentoo. This means that if you opt for KiCad, you can be fairly certain that you will not suddenly find yourself stuck with files that are no longer readable or editable because current software is not available.

The KiCad suite consists of the project manager *kicad*, the applications *eeschema* (schematic) and *pcbnew* (PCB), and the tools *pcb_calculator*, *pl_editor*, *bitmap2component* and *gerbview*. These are stand-alone applications whose data consistency is ensured by the project manager. Along with being mature and very extensive (it takes up more than 1 GB on the hard disk), KiCad allows you to view finished PCBs in 3D because the extensive libraries contain the 3D data of the components. On the Mac the applications (*/Programs/*) and the libraries and templates (*/Library/Kicad/...*) are stored in different places. There's nothing wrong with that, but sometimes it is inconvenient.

As the "professional" amongst open source CAD software, KiCad may not have the latest look and feel and is perhaps not as nicely integrated as packages, but it makes up for that with many good and useful functions, including 3D visualization. The large number of features make this package somewhat unintuitive in use, and it has a steep learning curve. However, the documentation and help files are available in various languages, which makes it easier for users who are not native English speakers.

Fritzing

Fritzing is an open-source program available for Windows, OS X and Linux in 32-bit and 64-bit versions, without any limitations. What sets it apart is that it is an integrated PCB package with a schematic view, a PCB layout view, and — very special — a breadboard view. This means that after you draw your circuit diagram with the program you can first try it out and debug it on a breadboard before you cast it in stone on a PCB. That's especially nice for the maker scene.

As typical with open source software, the Fritzing download is still beta, which means not yet version 1.x. However, the current version (0.9.3b) looks perfectly okay, and community support is in place. There is an online user guide and online help, as well as examples on the Web. The breadboard view is quite realistic. A second special feature is that Fritzing also has a code editor view, so you can write and manage code for microcontroller systems with this package. That makes it truly ideal for makers, Arduino projects and the like. On top of that, the software is localized in virtually all European languages as well as Japanese, Chinese, Russian and Turkish.

Once you have ironed out all the bugs in the circuit, everything

runs as it should on the breadboard and the PCB layout is finished, you can order a board directly from Fritzing Fab, the PCB fabrication service behind Fritzing, without leaving the program environment. However, at 70 eurocents per square centimeter the boards are on the expensive side – a garden-variety Arduino shield would cost you a tidy 29 euros. Of course, you do not have to use this fab service. The software can output PCBs in image, PDF, SVG or Extended Gerber format, and it can export XML and Spice net lists as well as components lists.

CometCAD

Some freeware is good, and some is not. CometCAD unfortunately falls in the latter category. Although this simple combination of a schematic editor and a layout program can do forward annotation and you can make usable PCBs with it, it's not exactly easy.

In terms of appearance the program is rather plain, and software updating is not its strong suite. The current version (1.09) dates from December 2015, and a version specifically adapted to Windows 10 is not available. However, the program ran under Windows 10 without any problems. There are no versions for other operating systems. The free version (L1) is severely limited, with a maximum of two sheets per schematic and a maximum of 50 symbols per sheet, which means less than 50 components. The maximum board size is a measly 102 x 102 mm, with a maximum of 250 pins. What's worse, the maximum library capacity is 2,000 components. All in all, other packages offer a lot more and are more modern. It is unlikely that the free version will encourage users to pay \$67 for the L2 version or \$134 for the L3 version, which allow more components and larger boards.

Osmond PCB

Osmond PCB is similar to CometCAD. There is a limited free version and a payware version. The limitations here are not as severe; the free version is only limited to a maximum of 700 pins if you want to output the board data in a file (Excellon, etc.) or print the layout. There are two significant differences with respect to CometCAD: Osmond PCB is Mac only, which means it is only available for OS X, and there are periodic bug fixes. The current version (1.1.33) dates from August 2016. However, that's the end of the good news. The look and feel of the program have not changed since the original design, which means OS X 10.5. Since then the Mac world has changed a lot, with OS X now at version 10.12. The user interface is simple, and the features are equally simple. One of the biggest shortcomings is the components library, which is a key resource for PCB design. The library contains a total of 130 components, which is pitifully few.

Support for scripting and data output in all major file formats is not enough to make things right. In light of what other packages offer, it is difficult for small software developers to compete, and users have to consider whether they want to put the effort into learning how to use niche software.

EasyEDA

If you are not a fan of big, feature-rich CAD packages from major vendors with long pedigrees, or you only want to make occasional "quick and dirty" PCBs, you might want to take a

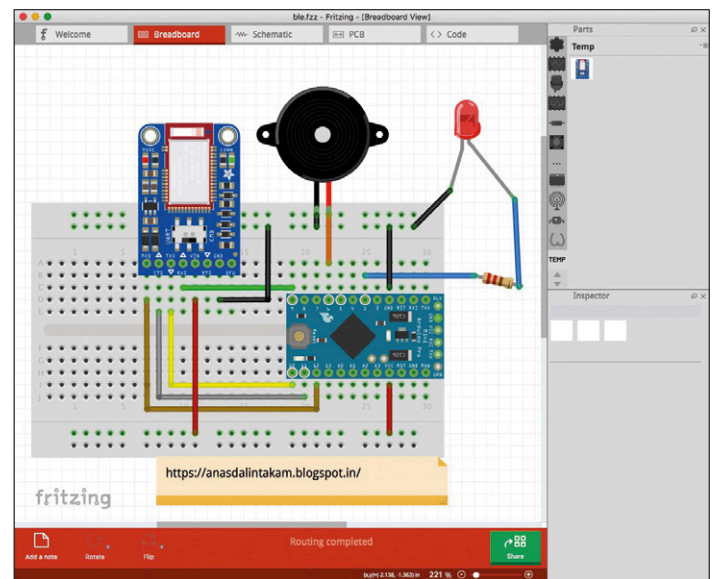


Figure 4. Flashy and trendy for the maker scene: Fritzing.

closer look at EasyEDA. This is an extensive online package which runs in virtually every browser and is obviously cloud-based. It certainly has something to offer – along with the schematic editor and PCB layout tool, it has an integrated mixed-mode simulator that allows you to switch between sche-

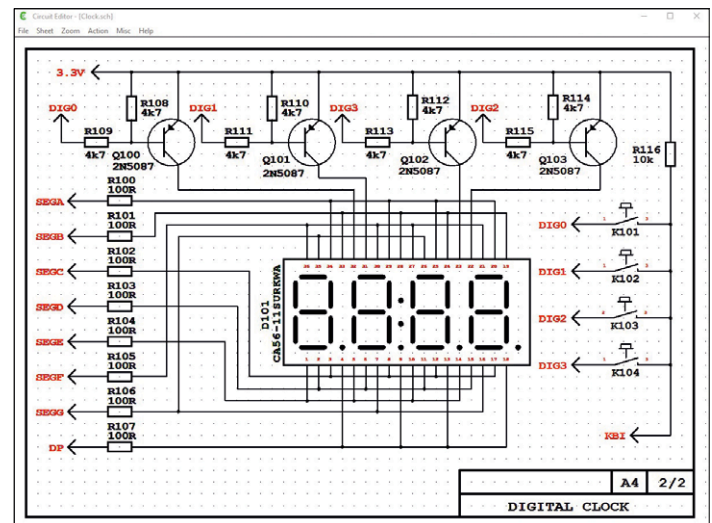


Figure 5. It works, but on the thin side and limited: CometCAD.

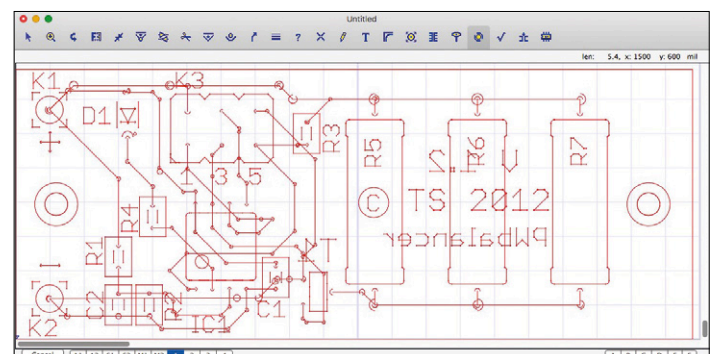
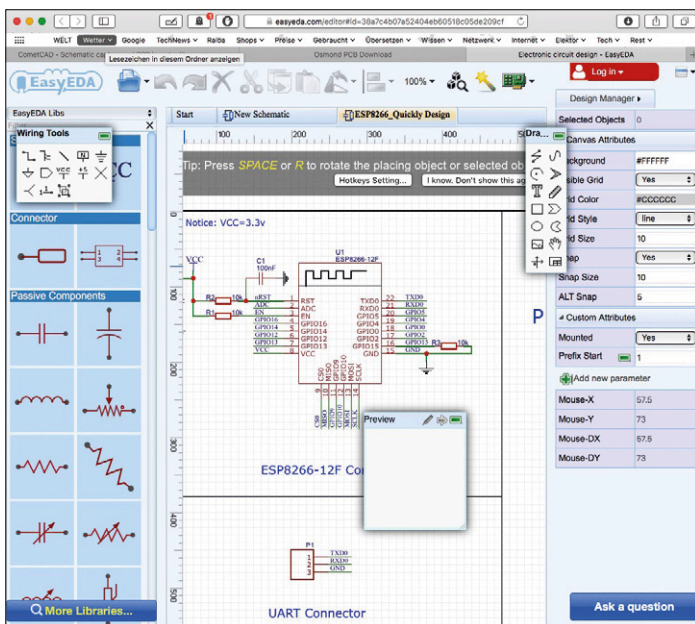


Figure 6. Mac only, and also thin and limited: Osmond PCB.



matic and PCB views within the simulation session. The idea appears to be that modern electronic designers don't bother with breadboarding, since they can simulate the circuit and then go straight to the PCB.

An online placing package that runs in a browser has a number of advantages. You don't have to install it, the software is always up to date, and it is totally independent of any operating system. It is also easy to share schematics or PCB layouts. Of course, you can save your designs on your own computer. It's hard to find any serious disadvantages compared to regular offline software. EasyEDA is easy to use, smooth and modern. The available library has a good selection of connectors and transistors, but it's a bit thin with ICs. However, it supports schematic and PCB file import from the professional packages

Figure 7. As a browser-based package, EasyEDA is platform independent, and it features integrated simulation.

Other options

Along with open source programs and free versions of commercial CAD packages (usually with limitations), there is a third category of CAD software which is fostered by major electronics distributors. Some examples of this are EAGLE from Farnell/element14, DesignSpark PCB from RS Components, and MultiSIM Blue from Mouser Electronics. Informative articles on all of these programs have been published in previous issues of *Elektor*. Their key features are briefly described below.

EAGLE

EAGLE is probably the best known PCB layout program amongst hobbyists. One reason for this is that along with paid versions, it has always been available in completely free versions, mainly limited with respect to board size. Students are especially frequent users of the free versions. EAGLE consists of two modules: one for drawing schematic diagrams and the other for laying out PCBs. They communicate with each other through direct forward and backward annotation. There are two different free versions: Educational and Express. The Educational version is intended for schools and universities and has minimal limitations. It has an autorouter and allows up to 99 sheets per schematic. Boards can have up to six layers, with the maximum size

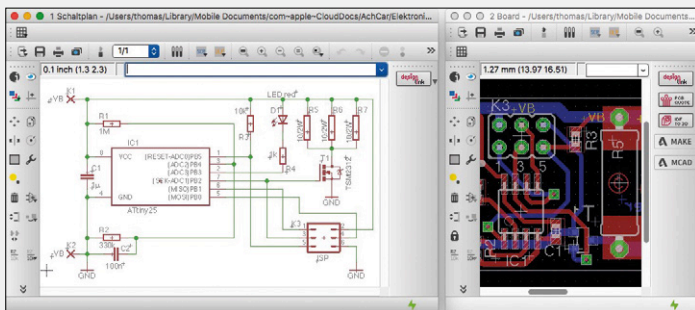


Figure 8. EAGLE with its red and green circuit diagrams, familiar to every electronics adept.

limited to Eurocard format (100 x 160 mm). A prerequisite is a valid .edu email address. The Express version is available to everybody. Although it also has an autorouter, the schematic is limited to two sheets and the maximum board size is half as large (100 x 80 mm). That makes the board size the key difference. Interestingly enough, EAGLE is available not only for Windows, but also for OS X in 32-bit and 64-bit versions.

A control panel provides access to the various modules, including the schematic editor, PCB layout and component libraries. EAGLE is fully sufficient for most purposes. The *Elektor Store* offers several good books on EAGLE, to help you get started quickly, correctly and comprehensively. EAGLE was recently acquired by Autodesk. The new owner has already announced that the free versions will remain available.

MultiSIM Blue

This is a limited version of the expensive professional MultiSIM CAD package from National Instruments. Mouser collaborated with NI to make a free version available. Of course, the free version has several limitations, but fortunately (and for obvious reasons) it comes with a library of more than 100,000 components from the Mouser catalog. Although that falls well short of the full Mouser product portfolio, it is gradually being expanded. You can even put together a components list for a schematic and get the Mouser prices for the components. That is very practical, of course, but if there is a component that Mouser does not carry and you want to add it from another supplier, you have a problem.

The limitations are unfortunately severe: at most six user-generated components, with a maximum of 65 components and one schematic per board. However, the board size is unlimited. The MultiSIM package consists of the MultiSIM program for schematic diagrams and Ultiboard for PCB layout. As a special feature compared to the other packages described in the

Altium Designer and EAGLE. It can also import net lists from LTspice. And last but not least, it can import libraries from KiCad. EasyEDA comes from a Chinese company that also sells PCBs, which can be ordered directly from the program. At about \$17 for a single double-sided board measuring 100 x 100 mm, they are reasonably priced. This online package is certainly worth bookmarking as good backup CAD solution.

What else?

The selection of packages described here is far from complete, but it gives you a good idea of what is currently available in the form of slimmed-down free versions or completely open-source programs with no limitations. Wikipedia has a good tabular overview of electronic design automation (EDA) software, including custom and niche products. If the web links listed below are not enough for you, the Wikipedia overview is a good source of additional leads. ◀

(160176-I)

Web Links

Pad2Pad: www.pad2pad.com/General/Software.html

gEDA: www.geda-project.org

KiCad: <http://kicad-pcb.org>

Fritzing: <http://fritzing.org/home>

CometCAD: www.cometcad.com

Osmond PCB: www.osmondpcb.com

EasyEDA: <https://easyeda.com/editor>

EAGLE: <https://cadsoft.io>

MultiSIM Blue: www.mouser.co.uk/multisimblue

DesignSpark: www.rs-online.com/designspark

CAD overview: https://en.wikipedia.org/wiki/Comparison_of_EDA_software

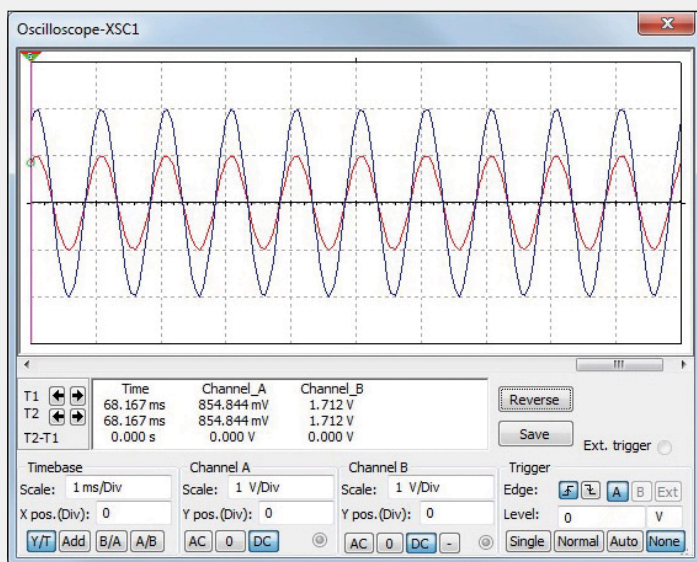


Figure 9. An amplifier simulation in MultiSIM Blue, the slimmed-down professional package from National Instruments for distributor Mouser.

overview, as the name suggests it includes schematic simulation capability.

The user interface and operation of both programs are excellent. However, the package consists of two separate modules. Although there is button in MultiSIM for forward annotation, in the Blue version it is disabled for no apparent reason. The software runs under Windows XP and later versions.

DesignSpark

DesignSpark from RS Components is a version of the Easy-PC PCB CAD program from UK software developer Number One Systems, specifically adapted for RS. Two additional versions have been released in recent years: DesignSpark Electrical for drawing electrical system diagrams, and DesignSpark

Mechanical as a conventional mechanical CAD program. For that reason, the original PCB layout version has been renamed DesignSpark PCB.

DesignSpark PCB has a modern look and feel and runs smoothly. It supports real-time design rule checking and forward/backward annotation between the schematic and the PCB. The program has a reasonable component library with about 80,000 components, and it can automatically output a components list which can be checked for availability and prices through RS. DesignSpark PCB can import EAGLE files, and it can even generate Spice net lists, which can be used to run simulations with LTspice, TINA or other programs. Unlike nearly all of its competitors, this package can visualize PCBs with their components in 3D. As far as we know the program is a full version with no limitations, but it is only available for Windows (7, 8 or 10).

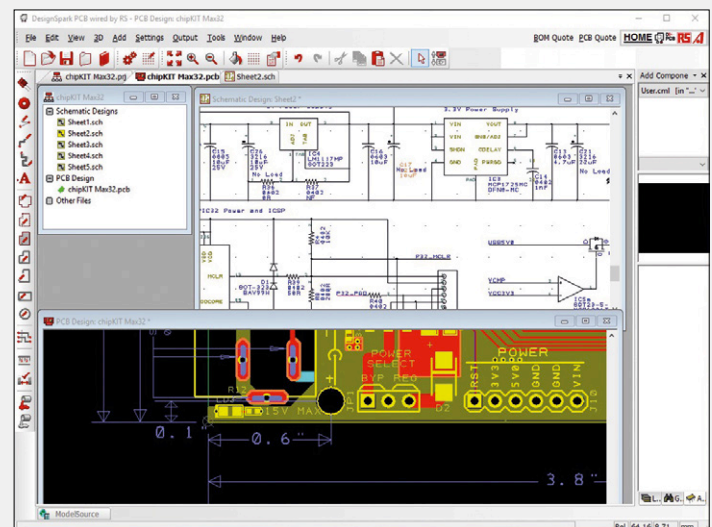
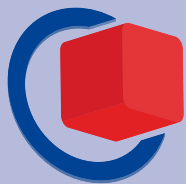


Figure 10. DesignSpark provides 3D visualization of finished PCBs and is free from distributor RS Components.

Get your free one-day ticket today!
Simply enter the following code at
embedded-world.de/voucher:
2ew17P

Nuremberg, Germany
14 – 16.3.2017



embeddedworld

Exhibition & Conference

... it's a smarter world

Get up to speed with the latest
developments in your industry!

embedded world is THE meeting place for the
international embedded community – secure
your crucial knowledge advantage now!

embedded-world.de

Trade fair organizer
NürnbergMesse GmbH
T +49 9 11 86 06-49 12
visitorservice@nuernbergmesse.de

Conference organizer
WEKA FACHMEDIEN GmbH
T +49 89 255 56-13 49
info@embedded-world.de

Media
partners

elektroniknet.de

computer-automation.de

**ENERGIE
& TECHNİK**
Solutions for a Smarter World

**DESIGN &
ELEKTRONIK**
KNOW-HOW FÜR ENTWICKLER

MEDIZIN-und-elektronik.DE

Elektronik
Fachmedium für integrierte Anwender und Entwickler

**Elektronik
automotive**
Fachmedium für professionelle Automobil Elektronik

Markt & Technik
DEUTSCHE VERLAGS-GRUPPE VON KLUWER

**Computer &
AUTOMATION**
Fachmedium der Automatisierungstechnik

MEDIZIN+elektronik
Fachmedium für Elektronik in der Medizintechnik

NÜRNBERG MESSE



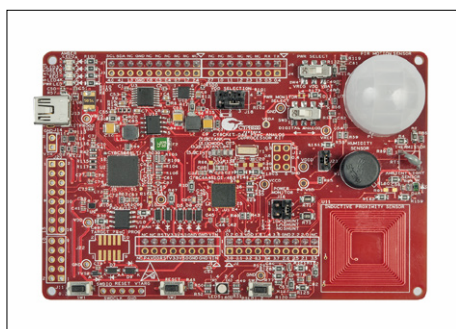
electronica 2016

Overview of what's hot

By **Viacheslav Gromov** (Germany) (readers@gromov.de)

When 2,913 exhibitors and around 73,000 visitors from 88 different countries all get together you can be sure there's lots to talk about and marvel at. From passive components via new types of special connectors and new sensors with pattern-recognition software the range of innovation was exhilarating. Here we present just a tiny selection of some of the intriguing items that caught our eye at the **e**lectronica trade show in Munich, Germany, last November.

Semiconductor design and manufacturing company **Cypress Semiconductor** were showcasing new products in two different areas of interest. Firstly they have a new addition to the established

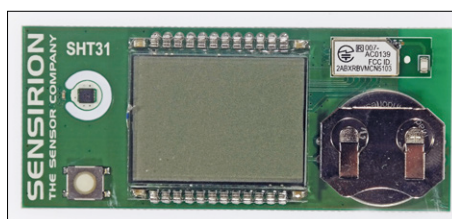


point at the show and Cypress already has much to offer in this area. New solutions for the technology include the EZ-PD CCG3 (48 MHz, 64 KB dual flash), with a 32-bit ARM Cortex M0 core which can fully support a USB-C port including all the interfaces with power delivery. The evaluation kit to go with this is the CY4531 EZ-PD CCG3 Evaluation Kit priced at \$250.

Also from Cypress is the new CY4500 EZ-PD Protocol Analyzer board which looks like a very interesting tool for USB-C developers. The board has a USB-C port and a USB-C receptacle connector allowing it to be inserted, for example between a computer and a peripheral device. It's transparent to the passage of this USB-C data. A third connector type USB micro-B hooks up to a host computer for power and for analysis of the USB data. The analyzer costs around \$200 but looks like an indispensable tool for developers of USB-C applications, it allows you to test programmed communication between two devices as well as the cable and wiring [2].



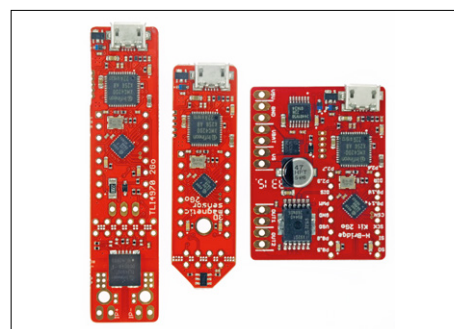
PSoC MCU family called the PSoC Analog Coprocessor (48 MHz, 32 KB Flash). This 32-bit MCU, based on the ARM-Cortex-M0+ core forms an interface between the analog and digital world. On board there are ADCs, DACs, opamps, comparators, analog filter, capacitive-sensing touch interface und much more. The \$49 CY8CKIT-048 PSoC Analog Coprocessor Pioneer Kit has been developed to allow designers to explore the possibilities of this flexible PSoC technology. The board has many analog sensors on-board suitable for applications ranging from temperature measuring to inductive proximity sensing and PIR motion detection [1]. USB-C technology was also a big talking



In addition to their advanced gas flow sensors **Sensirion** were showcasing their small but sophisticated SHT31 Smart Gadget Development Kit priced at €30. This Smart Gadget dev kit is a simple reference design board and development

kit built around the SHT31 humidity and temperature sensor. This new sensor is available in a 2.5 mm square package and communicates via an I²C interface. It samples the temperature and humidity (with 2% or 0.3°C accuracy) at a rate defined by the user. At one measurement per second the sensor draws just 2 μ A. The board shows the measured values which can also be read via Bluetooth on smart gadgets running an app. It also serves as a reference design for designers planning to use the sensor. The board is powered by a button cell and has an LCD screen and an 8-bit MC9S08LL-8CGT MCU, a pushbutton and the Nordic nRF51822 BLE module (**B**luetooth **L**ow **E**nergy). Connections to the sensor, MCU debug pins and the BLE module are easily accessible for development purposes. All information for the hardware and software is also available from GitHub [3].

Infineon certainly had much to show and tell. They have serious numbers of Arduino shields fitted out with their latest ICs. One of their popular small USB-stick sized evaluation boards the 'XMC 2Go' for the ARM based XMC1100 (M0+, 48 MHz, 64 KB) processor has now spawned a family of three similar sized boards using the same MCU plus sensors on board. The SPI-current sensor board 'Current Sensor

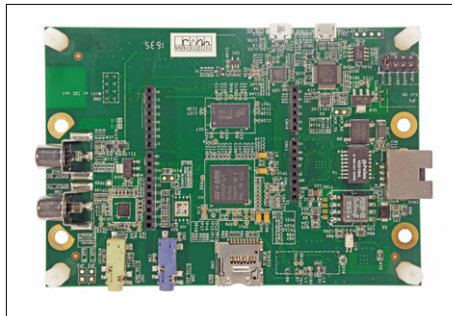




2Go' is fitted with a TLI4970. The I²C 3D magnetic field sensor board '3D Magnetic Sensor 2Go' is fitted with a TLV493D-A1B6 chip while the SPI-H bridge module 'H-Bridge Kit 2Go' uses the IFX9201 chip. In addition to all using the same XMC-based MCU they also have GPIOs and two user LEDs for use. Another feature of the boards is the sensor end can be snapped off leaving solder pads for easy hook-up to any systems. The magnetic field sensor also includes a small joystick so you build a smooth joystick controller. The boards are priced at around €20 [4]. The theme of data security was never far away and Infineon was showing an interesting TPM product (**T**rusted **P**latform **M**odule, an international standard for a secure cryptoprocessor) called the SLB9670, compliant with TPM 1.2/2.0. This module has an SPI interface (also available is the SLB9645 compliant to TPM 1.2 with an I²C interface) and is suitable for integration into existing MCU architectures (or a whole PC) for security. Up to eight 2048-bit keys can be stored internally. These keys never leave the IC and all data encryption and decryption is performed internally, transparent to the system processor. Normally the crypto keys would be stored on something like an SD card but this makes them vulnerable to theft or attack. The IC has been shown working in Linux-based systems; one interesting application is in conjunction with a Raspberry Pi. A special board has been developed for this and will be appearing shortly. These products are also supported by extensive online documentation and shows, explaining for example how you can implement a more secure SSL encrypted link between an RPI and a PC [5].

STMicroelectronics have added to their 32-bit ARM Cortex-M7 family of MCUs with new devices that are not yet in full production. They already have an impressive selection of MCUs and their associated development boards. The latest

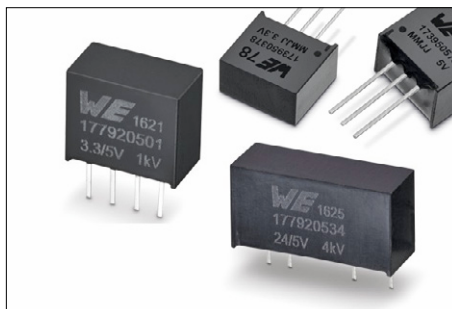
F7 board already in production is the STM32F769 Discovery Kit at \$48 fitted with a STM32F769NIH6 (216 MHz, 2 MB Flash) with impressive graphics capabilities. With its in-built Chrom-ART accelerator (DMA2D) and other peripherals it produces a 720p resolution video output signal with a frame rate of 30 Hz. In addition to the features on the standard



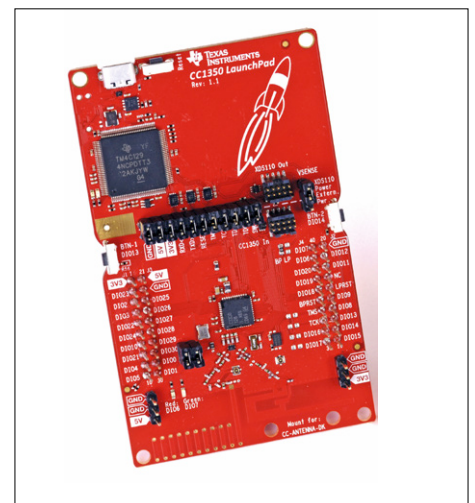
F7-ST Discovery boards the STM32F769 Discovery Kit also gives the possibility to add a TFT touch display, a DVI to HDMI or a WiFi adapter if required for development purposes [6].

Würth Elektronik has expanded their range of integrated power modules with the addition of the MagI³C-FISM (**F**ixed **I**solated **S**IP Module) and MagI³C-FDSM (**F**ixed **S**tep **D**own **R**egulator **S**IP **M**odule) series. The MagI³C-FDSM is a step down regulator with a fixed output voltage level (5 V or 3.3 V at 1 A max.) and a wide input voltage range from 7 to 42 V (boasting an efficiency of up to 93%!). The MagI³C-FISM power modules feature galvanic isolation. The four variants are for use with an input voltage rail at 3.3, 5, 12 or 24 V. All these variants supply a fixed 5 V output at 1 W (with up to 80% efficiency!) and offer an input to output isolation good for 1 kV. They are all simple to use and very robust [7][8].

You certainly don't need to hang around too long waiting for new products from

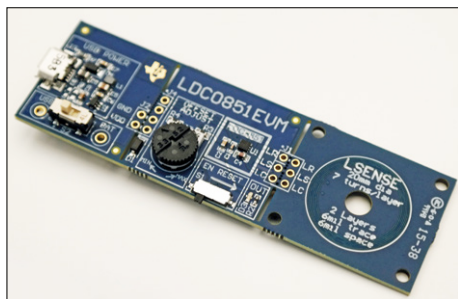


Texas Instruments. The new MCU CC1350 SimpleLink ultra low-power dual band wireless MCU has created a lot of interest. Apart from its 48 MHz 32-bit ARM Cortex M3 core, it has 128 KB flash memory and two RF peripherals. This allows the MCU to communicate in sub 1 GHz bands and also in the 2.4-GHz band compatible with Bluetooth Low Energy 4.2 spec. The sub 1 GHz offers the best possible RF range while the Bluetooth low energy provides the link via apps on BLE enabled smart devices. This combination gives the advantage of intelligent communications within a network. In an IoT or industrial 4.0 context the need for network interoperability is becoming increasingly important. The CC1350 sub 1 GHz (actually 868 MHz in the EU) channel for example would give the range



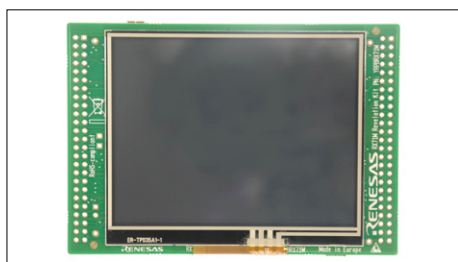
to link actuators and sensors in a local domestic network while the BLE channel can provide the link to the user's smart device. As you would expect TI have a development board for this new wireless microcontroller: the CC1350 LaunchPad, which can be yours for \$29 [9].

For the first time this year I detected a trend away from capacitive and towards inductive proximity sensing technology. Many manufacturers are already supplying MCUs incorporating interfaces supporting this technology. Its advantage is a greater immunity to environmental contaminants such as oil, water and dirt etc. TI has also developed large and small IC solutions for this switch technology where a simple pot is all you need to adjust sensitivity. One of the smaller ICs is the LDC0851; it needs very few external components and uses a detection coil which can be printed on the PCB. A handy



dev board for this chip has the part number LDC0851EVM and costs just €18. The board has a simple integrated inductive touch button and is powered via its micro USB connector or from an on board coin cell battery. The PCB has two snap off sections allowing the sensor and/or the sensor coil to be separated and used for other applications [10].

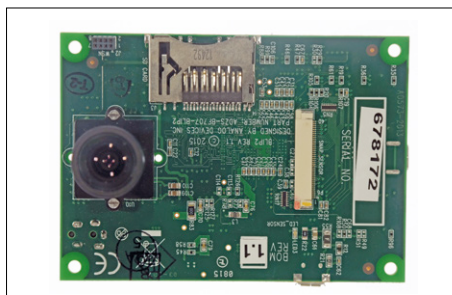
With a series of lectures **Renesas** were plugging their new RPBRX71M Revelation Kit priced at €59 which should prove useful if you are thinking of developing applications using the RX700 family of microcontrollers. The Revelation Kit hardware includes an RX71M MCU with 4 MB Flash, 552 KB RAM and a 320 x 240 QVGA TFT LCD with touch screen (which looked to be around 4"), allowing the user to implement a simple, highly-integrated HMI (Human Machine Interface) solution on a single chip microcontroller. The 32-bit 240 MHz RX71M has a good range of interface capabilities and comprehensive security features including built-in



AES, DES, SHA and RNG for IoT applications. Test and calibration functions for the internal peripherals (e.g. ADC and GPIOs) are also implemented. [11].

Amid all the new innovation in the truly analog world **Analog Devices** were demonstrating the capabilities of their distinctly digital but no less impressive ADZS-BF707-BLIP2 (Blackfin low-power imaging platform) board. This was running some software libraries with multiple functional profiles for applications such as intelligent motion sensing, people counting, vehicle

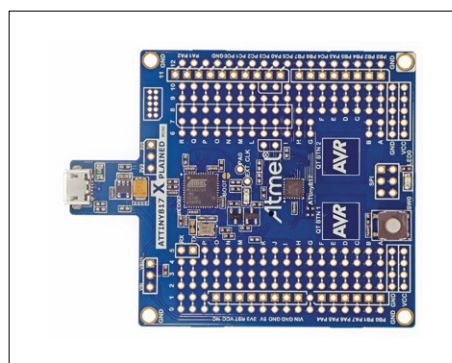
detection, and face detection. This \$200 board runs Analog Devices own 400 MHz 32 bit ADSP-BF707 processor with lots of interfaces including an on-board camera. Also with the kit for application development is the CrossCore Embedded Studio IDE and VOS 3.2.0 software libraries for indoor and outdoor applications plus full user guide and schematics [12].



Last but not least is news for the **Atmel** or should we say **Microchip** community. Good to their promise to maintain and expand 8-bit PIC and AVR support, Microchip had their new 8-bit ATtiny817, 816, 814 and 417 on show. The four devices range in pin count from 14 to 24 pins and 4 KB to 8 KB of Flash. These are the first tinyAVR microcontrollers to feature **Core Independent Peripherals (CIPs)** which include a **Peripheral Touch Controller (PTC)**, an **Event System (ES)**, a **Real Time Clock (RTC)** plus additional user-programmable logic blocks (LUTs). These CIP blocks operate almost independently, freeing up the main processor to take care of more important time critical events. The devices are supported by START, an online tool for graphical

configuration of embedded software projects. The development board for this new family of ATtinys is the USB-powered ATtiny817 Xplained Mini Board (retailing at \$10 (£7)) [13].

The PIC18F family also has a new sub family called the K40. These devices range in pin count from 28 to 64 with a 16 KB to 128 KB flash memory and a clock frequency up to 64 MHz. Highlights include the **Analog to Digital Converter with Computation (ADC²)**, which can go about its business without troubling the CPU. It can for example calculate average values of input signals or compare values (useful for touch applications). The peripherals also provide added security for safety critical applications (CRC / memory scan, windowed watchdog timer and hardware limit timer). Many



members of the K40 family are available in DIP outline so they can be used with the already familiar and universal Curiosity Development Board (priced at €30) [14]. ◀

(160270)

Web Links

- [1] www.cypress.com/products/psoc-analog-coprocessor
- [2] www.cypress.com/products/ez-pd-ccg3-type-c-port-controller-pd
- [3] <https://www.sensirion.com/en/products/humidity-sensors/development-kit/>
- [4] <http://goo.gl/o9hhGd>
- [5] <http://goo.gl/Xu5Lkr>
- [6] www.st.com/en/evaluation-tools/32f769idiscovery.html
- [7] <http://katalog.we-online.de/en/pm/MAGIC-FISM>
- [8] <http://katalog.we-online.de/en/pm/MAGIC-FDSM>
- [9] www.ti.com/tool/launchxl-cc1350
- [10] www.ti.com/tool/ldc0851evm
- [11] www.renesas.com/en-eu/solutions/key-technology/human-interface/rx71m-revelation.html
- [12] [Http://goo.gl/LpdZig](http://goo.gl/LpdZig)
- [13] www.atmel.com/tools/ATTINY817-XMINI.aspx
- [14] www.microchip.com/promo/pic18f67k40

electronica Fast



Who won the € 75,000 media budget

The “electronica Fast Forward Award, the Start-up Platform powered by Elektor”, as it is officially branded, turned out a huge success not only for the three winners of the grand prizes, but for everyone involved. Staged at the electronica 2016 trade show in Munich, Germany, 35 contestants from 15 countries pitched their projects to a jury consisting of industry experts from STMicroelectronics, Conrad, Würth Elektronik and, of course, your own Elektor. After three days of presentations, all showing excellent work, rankings were established in three

categories: Ideas, Prototypes and Start-ups. On the last day of the show, the first place winners of each category went on to compete for the grand prize, a magnificent PR and marketing budget valued at 75,000 euros plus an exhibition stand at electronica 2018 thrown in.

The contest, sponsored by STMicroelectronics (Platinum), Conrad (Gold), Würth Elektronik (Gold) and Trinamic (Bronze), turned out highly interesting with projects covering applications

The results

Ideas

1. Artem Kuchukov (*Germany*) – **Kewazo**, scaffolding assembly robot
2. Jonas Galle (*Belgium*) – **Valcun**, safe low-cost desktop metal 3D printer
3. Michael & Andrey Shustov (*Russia*) – **Baristor**, a barrier resistor

Prototypes

1. Till Nauman & Lara Obst (*Germany*) – **Mowea**, modular wind energy appliances for off-grid areas
2. David Link & Christian Kind (*Germany*) –

nevisQ, intelligent fall prevention and detection

3. Peter Wasilewsky (*Poland*) – **nWatch**, wearable MCU development platform

Start-ups

1. JF Brandon (*USA*) – **BotFactory Squink**, desktop PCB Printer
2. André Kholodov (*Germany*) – **eCozy**, smart connected thermostat
3. Milan Simek (*Czech Republic*) – **Sewio**, precise object tracking and visualisation system

Overall Winners

1. € 75,000 media budget + booth at electronica 2018: **Mowea, modular wind energy appliances for off-grid areas**
2. € 50,000 media budget: **BotFactory Squink, desktop PCB Printer**
3. € 25,000 media budget: **Kewazo, scaffolding assembly robot**

Special Price “Tech for Good”

€ 5,000 euro media budget: Len Williams (*Australia*) – **Every Drop Counts**, water consumption surveillance system

Forward Award 2016



plus a booth at electronica 2018?

as diverse as you can imagine, from IoT security to monitoring systems for the elderly and disabled, from state-of-the-art 3D printers to RGB lights controlled by plants, and from audio systems to precision measurement instruments and scaffolding building robots. Contenders from as far as India, central Russia, the US of A and even Australia, but also from numerous “nearby” European countries were ready to spend a week in Munich because they believe in what they have created. We were honored and proud that we were given the chance to

meet these inspired people and see their work.

Being a contest it was not possible unfortunately to award the Grand Prize to every participant, but none left empty handed. Although some got more than others, all were awarded a lifelong Elektor Hero Membership and we hope to hear from them soon about the progress of their projects. Will you join us next time?

(160278)



The *Mowea modular wind energy appliances for off-grid areas* project was unanimously selected as the Overall Winner of the Fast Forward Award 2016. Till Nauman (second from the right) received the Grand Prize from electronica CEO Falk Senger (right), assisted by our sponsors (from left to right) Shawn Silberhorn (Conrad), Alexander Gerfer (Würth Elektronik) and Jacky Perdrigeat (STMicroelectronics).



JF Brandon from BotFactory came all the way from Long Island City (USA) to present his *Squink desktop PCB printer*. It was definitely worth the effort as he flew home with the Second Prize.



Artem Kuchukov (second from the left) shined a happy smile when he received the Third Prize for the *Kewazo scaffolding assembly robot* project.

Simple Interfacing to Analog and Digital Position Sensors for industrial drive control systems

By **Brian Fortman**, Industrial Drives and Automation Marketing, C2000™ Microcontrollers, Texas Instruments

In industrial drive control system designs, “glue” elements like control and connectivity building blocks pose many challenges such as lengthier development cycles, a larger board area or a higher BOM cost. As a result, developers are often unable to concentrate on differentiating features like enhanced performance, greater precision and improved control loops. Here, some new supportive technologies are discussed to make life easier.



A particular example of this is the task of interfacing microcontrollers (MCUs) to position sensors. These sensors can be linear, angular or multi-axis and typically are used to sense the relative or absolute position of a mechanical system in motion, propelled by a motor. The sensed position is then converted to an analog or digital electrical signal for transmission to the controlling circuit.

Historically, interfacing a position sensor to an MCU could be a time-consuming task that often involved the integration of the communication protocol into a field programmable gate array (FPGA) or the programming of an additional MCU with the decode protocols. In addition, this situation is exacerbated by the fact that there are multiple encoder protocols available, each suited to certain types of functionality and subsystems. The system design team might be forced to develop several protocol-specific FPGAs which would not scale effectively from one application to another. Of course, this type of FPGA imple-

mentation would add cost to the system by increasing the system’s electronic BOM, impacting the necessary board space and requiring lengthy development cycles. Moreover, developers also have to complete extensive compliance testing to certify conformance with industry standards.

This situation begs for a solution that would simplify the interfacing of position sensors to control elements in industrial drive systems and thereby free designers to concentrate on features and functionality that would make their systems truly distinctive, as well as more competitive, in the marketplace.

Integrating position feedback

Building on the C2000™ Delfino™ MCU portfolio, Texas Instruments provides a comprehensive platform for industrial drive and control systems. Starting with the processing capabilities required by sophisticated and precise control systems, the C2000 Delfino F28379D and F28379S MCUs are equipped with

a full complement of on-chip resources, including DesignDRIVE Position Manager technology supporting today's most popular off-the-shelf analog and digital position sensor interfaces. This relieves system designers from many of the more basic, repetitive tasks, saving design time.

TI has extensive expertise with interfacing position sensors to digital controllers. Beginning with standalone interface solutions for resolver-to-digital solutions, such as the TMSRSLVR, TI has continued to add to its position feedback interface support. Expensive resolver-to-digital chipsets have been replaced by C2000 MCU on-chip capabilities, leveraging high-performance analog-to-digital converters (ADCs) and digital-to-analog converters (DACs). Moreover, the powerful trigonometric math processing of C2000 MCUs is particularly well-suited to the additional processing needed to calculate the angle, and extract high-resolution speed information from a resolver's amplitude modulated sinusoidal signals.

Many C2000 MCUs support enhanced quadrature encoder pulse (eQEP) modules that are capable of interfacing with linear or rotary incremental encoders. These encoders count pulses to obtain position (once an index is known), direction and speed information from rotating machines used in high-performance motion and position control systems. In addition, the eQEPs can be employed to interface to pulse train output (PTO) signals generally output by a programmable logic controller (PLC) in industrial automation for motion control.

Also, eQEPs can interface to clockwise/counter clockwise (CW/CCW) signals. CW/CCW signals are typically used in conjunction with stepper or servo drives for controlling motors or other motion-based hardware. The C2000 F28379 MCUs (**Figure 1**) support up to three eQEP modules.

Resolver and QEP capabilities provide fast, efficient and integrated solutions for effectively interfacing position sensors with C2000 Delfino MCUs. The next step has been to extend that support with complementary solutions that would allow the MCU to connect directly to more advanced digital and analog position sensors.

DesignDRIVE Position Manager technology

Available through TI's DesignDRIVE platform, Position Manager technology takes advantage of the on-chip hardware resources of the C2000 Delfino F28379S and F28379D MCUs to interface to the most popular digital and analog position sensors. Already incorporating support for incremental encoders (eQEP), CW/CCW communications and standalone resolver solutions, Position Manager adds solutions for analog position sensing, integrating both resolver excitation and sensing, as well as a SinCos transducer manager.

Unique to C2000 MCUs, Position Manager combines the analog sensor support with the popular digital absolute encoders, EnDat 2.2 and BiSS-C (**Figure 2**), giving system designers a wide range of position sensor types to choose from.

This integrated Position Manager technology offers system designers a real opportunity to accelerate development cycles and reduce BOM costs by eliminating the need for an FPGA to interface a specific encoder to the MCU or by drastically reducing the size of the FPGA that may still be needed for other functions. **Figure 3** demonstrates how Position Manager technology relieves system designers from the burden of developing the high- and low-level software drivers, as well

TMS320F28379D		Temperatures		105C	125C	Q100
Position Manager	EnDat, BiSS, Resolver...	Processing	Processing	Actuation		
Sensing	ADC1: 16-bit, 1.1-MSPS 12-bit, 3.5 MSPS	C28x™ DSP core 200 MHz	C28x™ DSP core 200 MHz	12x ePWM Modules (Type 4) 24x Outputs (16x High-Res)		
ADC2: 16-bit, 1.1-MSPS 12-bit, 3.5 MSPS	FPU	FPU	FPU	Fault Trip Zones		
ADC3: 16-bit, 1.1-MSPS 12-bit, 3.5 MSPS	TMU	TMU	TMU	3x 12-bit DAC		
ADC4: 16-bit, 1.1-MSPS 12-bit, 3.5 MSPS	VCU-II	VCU-II	VCU-II	Connectivity		
8x Windowed Comparators w/ Integrated 12-bit DAC	CLA core 200 MHz	CLA core 200 MHz	CLA core 200 MHz	4x UART		
8x Sigma Delta Channels	FPU	FPU	FPU	2x I2C		
Temperature Sensor	6ch DMA	6ch DMA	6ch DMA	3x SPI		
3x eQEP	Memory	Memory	Memory	2x McBSP		
6x eCAP	Up to 512 KB Flash	Up to 512 KB Flash	Up to 512 KB Flash	2x CAN 2.0		
System Modules	Up to 102 KB SRAM	Up to 102 KB SRAM	Up to 102 KB SRAM	USB 2.0 OTG FS MAC & PHY		
3x 32-bit CPU Timers	2x 128-bit Security Zones				uEP	
NMI Watchdog Timer	Boot ROM				Power & Clocking	
2x 192 Interrupt PIE	2x EMIF				2x 10 MHz OSC	
					Ext OSC Input	
					Debug	
					Real-time JTAG	

Figure 1. A closer look at the C2000™ dual-core F28379D MCU with DesignDRIVE Position Manager technology.

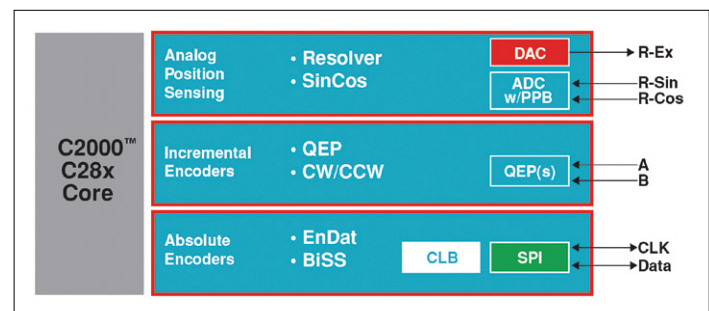


Figure 2. DesignDRIVE Position Manager technology supports the leading analog and digital position sensors.

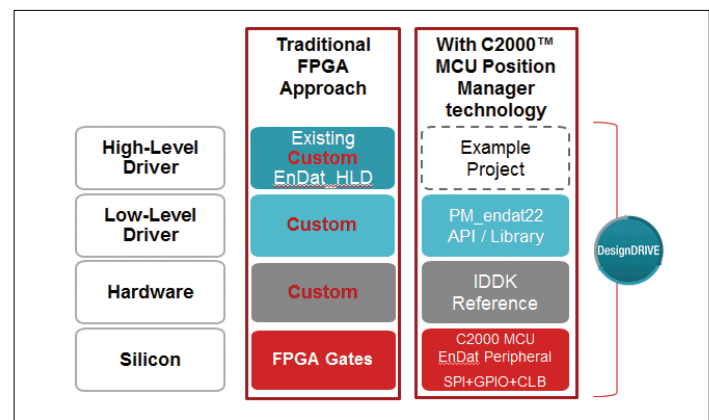


Figure 3. EnDat 2.2 solution example: Stackup vs. FPGA.

as any custom hardware and logic that previously may have been implemented on an external FPGA. In addition, example closed-loop, position-sensor-based control projects downloaded from DesignDRIVE can be modified for integration into cus-

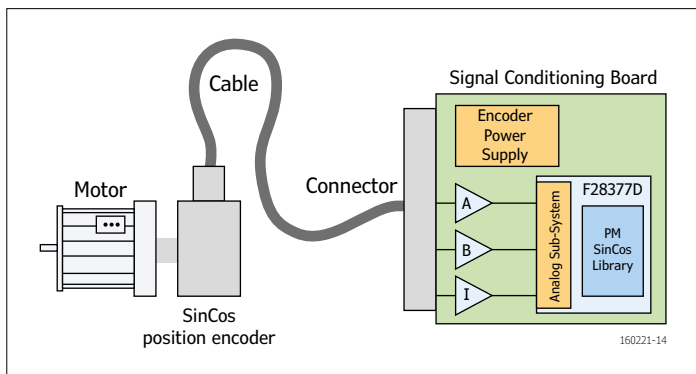


Figure 4. Industrial servo drive with SinCos position encoder interface.

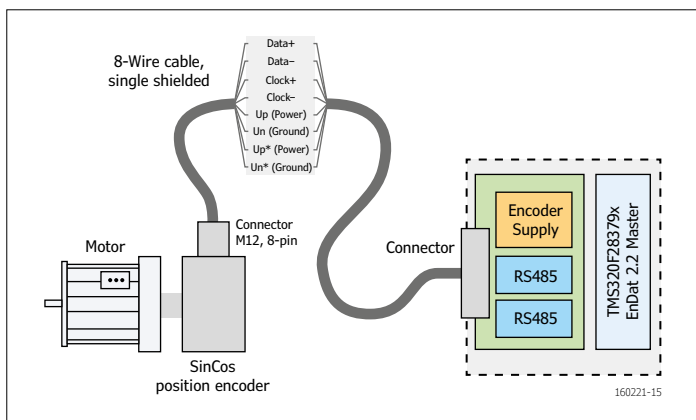


Figure 5. Industrial servo drive with EnDat 2.2 position encoder interface.

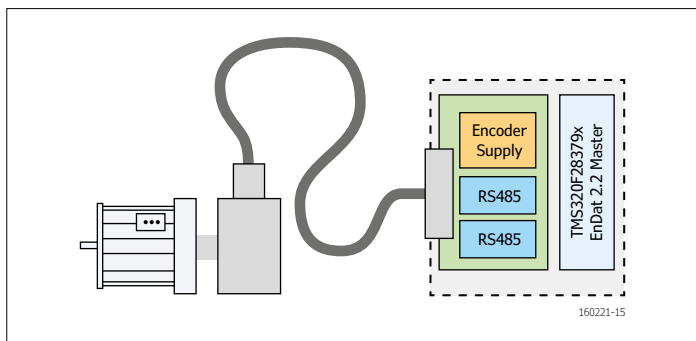


Figure 6. Industrial servo drive with BiSS-C position encoder interface.

tomer projects. The lower system layers are provided on-chip or through reference designs and a ready-to-use library of application programming interface (API) modules. In addition to reducing development time, Position Manager technology also decreases the compliance and interoperability testing that system manufacturers have undertaken in the past. The Position Manager technology is fully tested across a variety of sensors. Please see the user's guides for details on the testing results. Moreover, future revisions and updates to the applicable standards will also be supported by Position Manager technology.

New position sensor interfacing capabilities

TI has been able to expand its position sensor interface solutions with enhanced capabilities and performance. The following are several recent additions to TI's capabilities through DesignDRIVE Position Manager technology.

SinCos

SinCos is a feedback methodology which is incorporated into encoder interfaces like Hiperface® as well as other proprietary interfaces. These so-called sinusoidal absolute encoders typically offer much higher position and speed resolutions than do resolver or incremental encoders. In conventional quadrature encoders, angle information is obtained by counting the edges of a pair of quadrature pulses.

Angular resolution is fixed by the number of pulses per mechanical revolution. However, in SinCos transducers, precision of the angular measurement is increased by computing the angle between edges using the relationship between a pair of sine and cosine outputs from the sensor. Effectively, an interpolation between edges is made to obtain a 'fine' angle. The fine angle is computed using an arctangent of the two sinusoidal inputs. For this computation to be valid, both inputs must be sampled simultaneously.

Typically, several thousand electrical revolutions of the sinusoidal signals occur during each mechanical revolution of the encoder shaft.

The internal analog sub-system of the F28379 Delfino MCUs is ideal for interfacing to SinCos transducers. The presence of multiple ADCs, which can be triggered from the same source, allows simultaneous measurements of both input channels. In addition, the F28379 MCUs include a native ARCTAN instruction as part of the Trigonometry Math Unit (TMU) which means the angle calculation can be done in as little as 70 nanoseconds! Another consideration is the high motor shaft speed state (Figure 4). In this case, there is no longer a need for precise angle information and the measurement algorithm only needs to count the number of complete sinusoidal revolutions to determine a "coarse" angle measurement. Typically, this is done using a pair of analog comparators which compare the incoming sinusoids with a threshold representing the zero crossing point. The comparator outputs correspond to the sign of each sinusoid and the resulting digital signals are similar to those produced by a quadrature encoder. On the F28379 MCUs, there are up to eight pairs of analog comparators, each with its own programmable threshold voltage. These allow the quadrature pulses to be generated which are then fed internally to one of the on-chip quadrature encoder peripheral (QEP) modules for coarse angle and speed measurements.

EnDat

EnDat is a digital bi-directional four-wire interface developed by the German company, HEIDENHAIN. A sensor with an EnDat encoder can communicate position values, transmit and update information stored in the encoder, or save the information. Data is sent along with clock signals. The C2000 MCU can select the type of data the encoder will transmit, including position values, parameters, diagnostics and others.

Position Manager technology interfaces the C2000 F28379 MCU directly to the EnDat encoder (Figure 5). The only components external to the MCU are two RS-485 transceivers and the

encoder power supply circuit. The EnDat Master is implemented using the C2000 MCU's configurable logic block, where the communication protocol is handled.

Position Manager technology has been tested against a range of rotary, linear and multi-turn encoders from HEIDENHAIN and across resolutions from 13 bits to 35 bits at distances of 70 meters or more.

BiSS-C

The open source BiSS (bi-directional/serial/synchronous) digital interface is based on a real-time communications protocol. The original specification was developed by iC-Haus GmbH of Germany.

BiSS-continuous mode (BiSS-C) is employed in industrial applications. The specification has its roots in the Synchronous Serial Interface (SSI). The BiSS-C interface consists of two uni-directional or bi-directional lines for the clock and data.

As with all interfaces supported by Position Manager technology, a BiSS-C master running on a C2000 F28379 MCU can connect directly to a BiSS-C encoder slave on a position sensor (**Figure 6**). The interface transmits position values and additional information directly from the encoder to the MCU.

The MCU is able to read and write directly to the encoder's internal memory. TI's Position Manager technology includes a feature-rich BiSS-C library of capabilities, which system developers can readily draw on for their development projects. For example, clock frequencies of 8 MHz are supported on cables up to 100 meters long. In addition, the C2000 MCU BiSS inter-

face can be adjusted to feature improved control of modular functions and timing by transmitting position information from encoders every control cycle.

Industrial drive control systems-on-chip

Powerful and programmable MCUs like TI's C2000 Defino F28379 MCUs represent the next step toward industrial drive control systems-on-chip (SoC). They empower more effective and efficient system architectures by eliminating the need for an external FPGA for ancillary processing requirements or by reducing the size of the FPGA significantly.

By enabling a direct connection between a C2000 MCU and a position sensor, Position Manager technology frees developers from the more mundane tasks of device connectivity so they can focus on the features and capabilities that will make their system solutions truly distinctive in the marketplace with significant competitive advantages.

(160221)

The platform bar, C2000 and Defino are trademarks of Texas Instruments. All other trademarks are the property of their respective owners.

Advertisement

4power!

#redCUBE
*WE speed up
the future*



REDCUBE Terminals are the most reliable high-power contacts on the PCB level. Low contact resistance guarantees minimum self-heating. Four different designs cover all leading processing technologies and offer a wide range of applications.

www.we-online.com/redcube

- Flexibility in processing and connection technologies
- Highest current ratings up to 500 A
- Board-to-Board and Wire-to-Board solutions
- Extremely low self-heating
- Robust mechanical connection





Welcome to the **DESIGN** section

By **Clemens Valens**, Elektor Labs

SHARE

DESIGN

LEARN



Heinrich Rudolf Hertz (1857 – 1894)

The German city of Hamburg has fathered two famous concepts: the hamburger sandwich better known as The Burger, and the hertz, the SI unit of frequency. Both date back to the second half of the nineteenth century, but it is the second concept that retains our attention in this bimonthly column because of the importance of frequency (symbol: f) in electronics. So who was the man who invented frequency?

Heinrich Rudolf Hertz was born in a wealthy family, his father was a barrister and at some point even Senator of Hamburg. Heinrich quickly showed a keen interest in science and languages and went to good universities. He obtained his PhD in Berlin where he studied under Hermann von Helmholtz and Gustav Kirchhoff, two scientific heavyweights well known to electronics engineers. Heinrich

did not invent frequency, of course. He started to work on electromagnetism when Helmholtz suggested that he had a shot at proving Maxwell's theory. Although initially he did not think it was feasible he returned to the subject years later when he accidentally invented the tools that made it possible. With Maxwell's theory proven he let it rest as he did not think that there would be much use for it.

Not a one-trick pony, Heinrich also laid the foundations for the current field of contact mechanics when he solved the contact problem of two elastic bodies with curved surfaces. With his observation that a charged object loses its charge faster when illuminated by ultraviolet radiation he helped to establish the photoelectric effect. His experiments even place him at the beginning of X-rays. That Heinrich lived at the edge of knowledge was proven for the last time when he died from a rare disease that was only discovered officially some 40 years later.

Although he died too young, Heinrich did manage to pass some of his genius on to his offspring, allowing his youngest daughter Mathilde to become a world-renowned biologist and psychologist. His brother's son, Gustav Ludwig, won the Nobel Prize for Physics in 1925. Comparing photos of Heinrich and his clever nephew, the resemblance is so striking that one wonders that maybe his nephew's daddy wasn't his nephew's daddy, but his nephew's daddy didn't know?

The SI unit for frequency became the hertz (Hz) in 1960 when it replaced the cycle per second (cps). One hertz is defined as 1/second or s^{-1} , exactly as the becquerel (Bq), however one hertz means one event per second with the events spaced exactly one second apart, whereas one becquerel means one event per second on average. Where Joseph Henry has a mountain range named after him, Heinrich has a crater on the dark side of the moon that carries his name. Searching the Internet I did manage to find a recipe for a Hertz burger, nicely combining the two great nineteenth century Hamburger concepts into one.

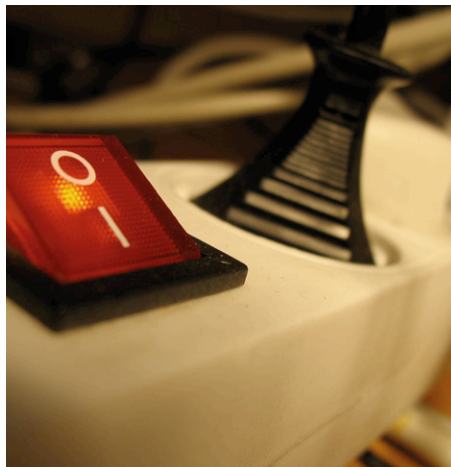
(160247)



Smart Power Bar

By **Hans-Norbert Gerbig** (Germany)

You can make a smart power bar (also called 'block'; 'strip') with one single semiconductor component: a triac!



It works like this: the full AC grid voltage (120 V or 230 V) is applied to connection A1 of the triac and the master equipment live feed connects to the gate (G) lead. Live supply to the slave equipment comes from the A2 lead on the triac (**Figure 1**).

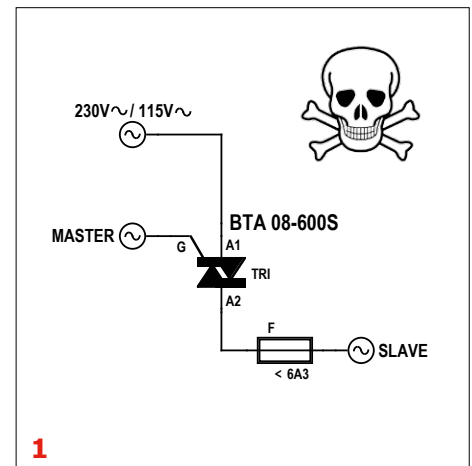
When the master equipment is switched on, its supply voltage flows between A1 and G. The triac conducts and provides a live supply to all the slave equipment from its A2 pin. When the master equipment is turned off the gate current drops to zero, the triac turns off and power to the slave equipment also turns off.

It also functions if the gate and A1 connections are switched so that the AC input connects to the gate and A1 provides power to the master socket. In this configuration however combined power to the master and slave will now pass through the gate instead of A1, see Labs comment below.

Figure 2 shows how the triac is wired to a 4-outlet power bar to provide three slave outlets. The triac can be soldered to a small piece of stripboard together with a three-way terminal strip. In the power bar cut the Live and Neutral between the first outlet socket and the remaining three sockets. The triac 'module' can then be wired according to the diagram using short lengths of wire.

For protection it's recommended to connect a slow-blow fuse in series with the slave outlets. As the circuit connects

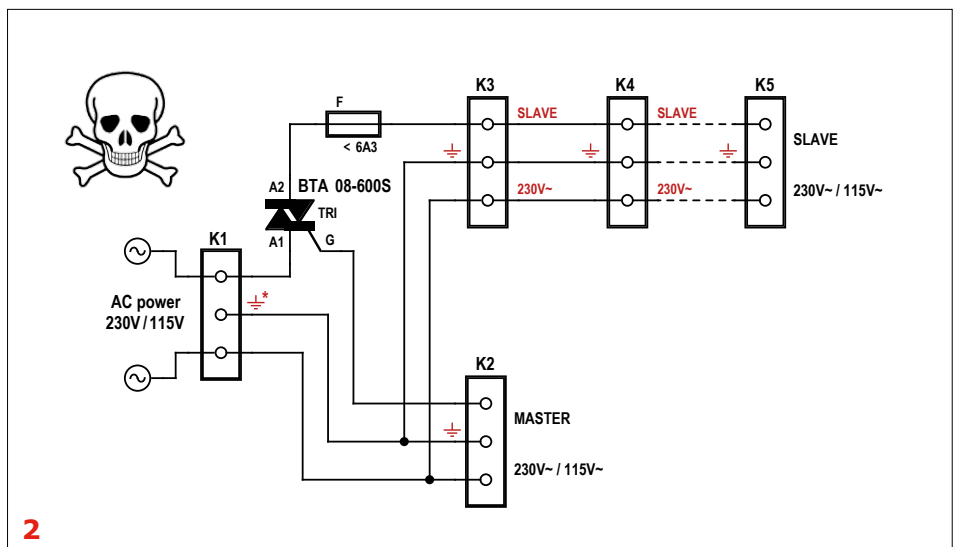
directly to the AC grid with no galvanic isolation, it is imperative to ensure that the circuit is fully enclosed to avoid any possibility of accidental contact with any part of the circuit during use. The Protective Earth (US: Ground) lead should remain intact to all the sockets.



1

The type of triac used here depends on the amount of current drawn by the master and slaves. A BTA08-6008 (600 V/8 A) triac requires a very small gate current and low trigger voltage. ◀

(130419)



2

Elektor Labs feedback:

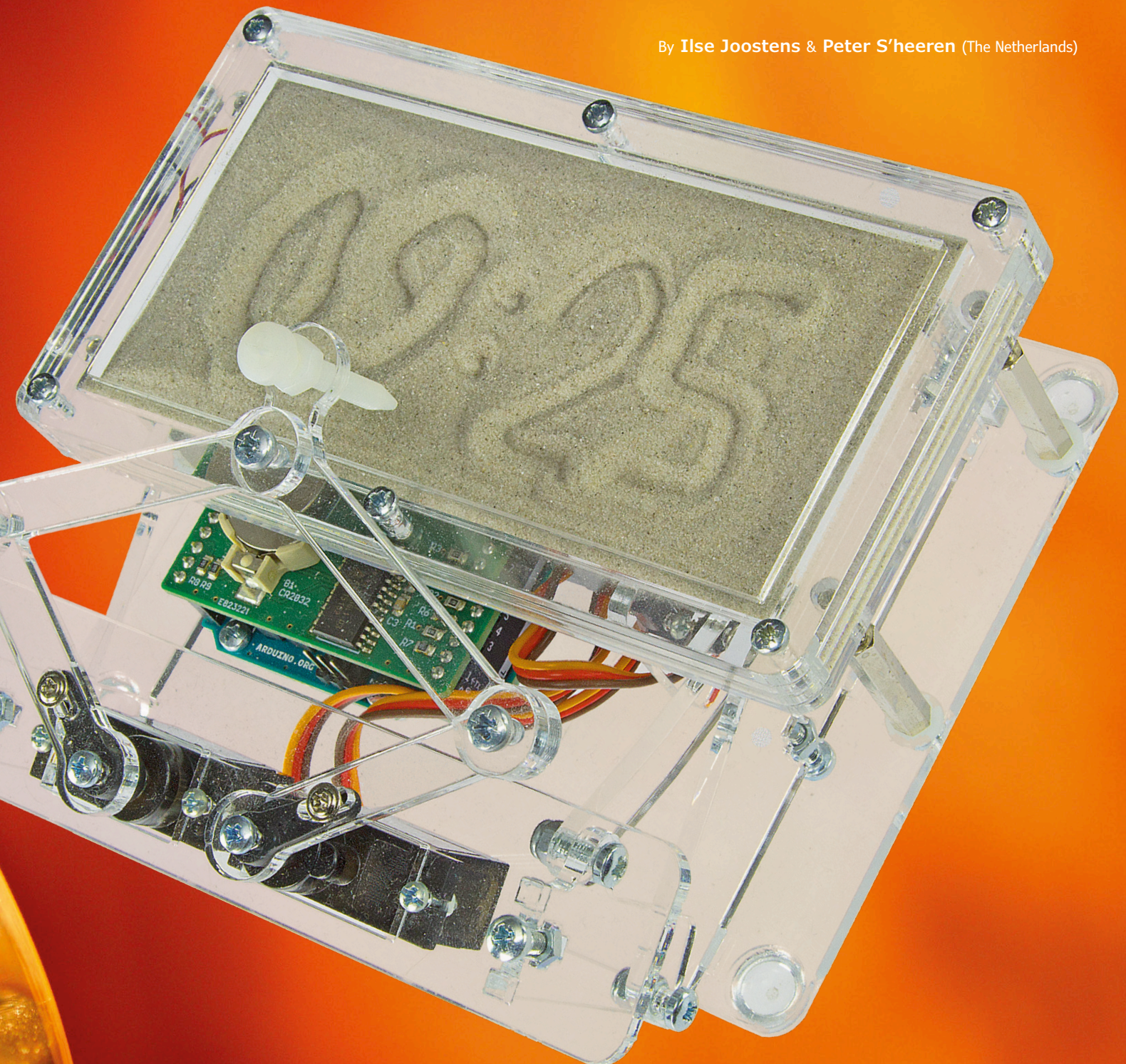
This is a neat trick we haven't seen before. It does raise the question whether a triac can be safely used in this way. Looking at a typical triac die layout the gate region is relatively small compared to the A1 and A2 connections. You can't expect the gate to pass the same amount of current. Just how much current the gate can handle is unclear. Most of the triac data sheets we looked at from many different manufacturers didn't specify a maximum value. One data sheet however gave a maximum gate current of 500 mA. Bearing this in mind, be aware that the master equipment must not draw too much current.



Sand Clock

A real eye-catcher

By Ilse Joostens & Peter S'heeren (The Netherlands)



This nifty gadget built around an Arduino Uno uses some servos and a pantograph mechanism to write the time in a sand bed. After a configurable time interval the sand is smoothed out by a pair of vibration motors and the cycle starts over again. Along with the automatic time mode, the clock software has a command mode which allows you to control the pen using simple commands.



Figure 1. The kit contains all the parts (including the AC adapter) necessary to build the Sand Clock. However, you have to program the Arduino yourself using the free download software.

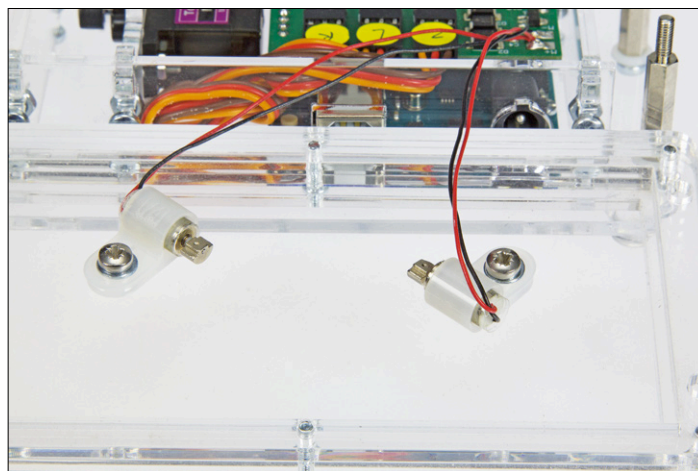


Figure 2. The positioning of the two vibration motors beneath the sand bed is critical for shaking the sand surface smooth.

The original idea for this project came from one of our friends at the German magazine *Make*, who in turn was inspired by the plotter clock built by FabLab in Nuremberg (Germany). During the first hardware team meeting with the folks at *Make*, the authors selected a number of projects that appeared to be suitable for adaptation and modification by Elektor Labs to produce DIY kits for retailing. The first project tackled by the Elektor Labs is the Sand Clock described here.

We started by modifying the mechanical components to make them suitable for a kit, and easy to assemble. Then we designed an Arduino shield for the additional electronics, and finally we rewrote the original software to make it easier to understand and easier to use, and to add more features. All of that together resulted in the project described here.

A kit with all the parts, including an Arduino Uno and a fully assembled shield, is available in the Elektor Store (see **Figure 1**). If you have access to a laser cutting machine and are not afraid of a bit of SMD soldering, all of the CAD files and Gerber files are available as free downloads [1].

Most of the mechanical parts of the Sand Clock are made from 3-mm extruded PMMA (acrylic) sheet, also known by the brand names Perspex and Plexiglas. Along with being readily available, this material can be cut very easily with a laser cutting machine, resulting in nicely finished edges. The thermoplastic properties of the material are an added bonus.

Here we use three model servos to drive the clock mechanism, the same as in the original plotter clock. The torque necessary for this is very small, so the requirements on the servos are fairly relaxed. Due to the simple drive shaft mounting arrangement for this application we opted for servos with metal gears, which also keeps the mechanical play reasonably small.

The sand bed is shaken smooth by two 6-mm vibration motors mounted at a 45° angle below the sand bed and secured with plastic cable clamps.

Pantograph mechanism

The key mechanical components of the Sand Clock are the two servos (left and right) and the acrylic arms, which position the drawing pen. This arrangement is called a pantograph due to its similarity to pantograph drawing tools of a bygone

age. Another relative from a mechanical perspective is the Autopen signing machine, which uses a pantograph mechanism to duplicate handwritten signatures on paper. These machines are still used by high-ranking politicians and public figures. They are also sometimes used in fund raising campaigns because an Autopen signature looks more authentic than a printed signature. Many US presidents have been well-known Autopen users, including Barak Obama, who used the machine to sign a number of laws while visiting France or on vacation in Hawaii. The **“Inverse Kinematics” inset** gives an overview of the equations which form the basis for the operation of the Sand Clock, and which were also used to produce the software for the project.

Vibrating sand bed

The most difficult part of the project was not the dense mathematics described in the inset, but instead the design of the sand bed. We tried out a lot of prototypes before arriving at a satisfactory solution. The vibration motors are supposed to shake the sand, but not the entire clock, so the energy should be confined to the sand bed as much as possible. For this purpose the sand bed is mounted loosely

on four supports and is able to move to a certain extent in the horizontal plane. The biggest challenges were to get the sand smooth enough within about five seconds to allow the time to be clearly written in the sand, and to keep the sand from piling up in specific locations over the course of time. After experimenting with different configurations and a variety of vibration motors, it became clear that the desired results could not be achieved with a single vibration motor. In the end we obtained good results with a pair of 6-mm vibration motors. The positioning of the motors is crucial, and they have to be oriented at a 45° angle to the edge of the sandbox base plate (see **Figure 2**).

The type of sand is also important. The flow characteristics of sand depend on the grain size (from coarse to fine) and the shape of the sand grains (sharp or round). We obtained good results with fine white sand and a grain size of 0.1 to 0.3 mm. Due to the layered structure of the sandbox, we do not recommend the use of extremely fine sand. The sandbox has a rim on all four sides with a slight overhang to prevent the sand from spilling over the edge.

Electronics

The main electronic module of the Sand Clock is an Arduino Uno, which provides enough processing power for this application despite the many equations shown in the inset. There are also some other components necessary for powering the servos, driving the vibration motors and keeping the time correct when the power is disconnected (see the schematic diagram in **Figure 3**).

To make building the Sand Clock as easy

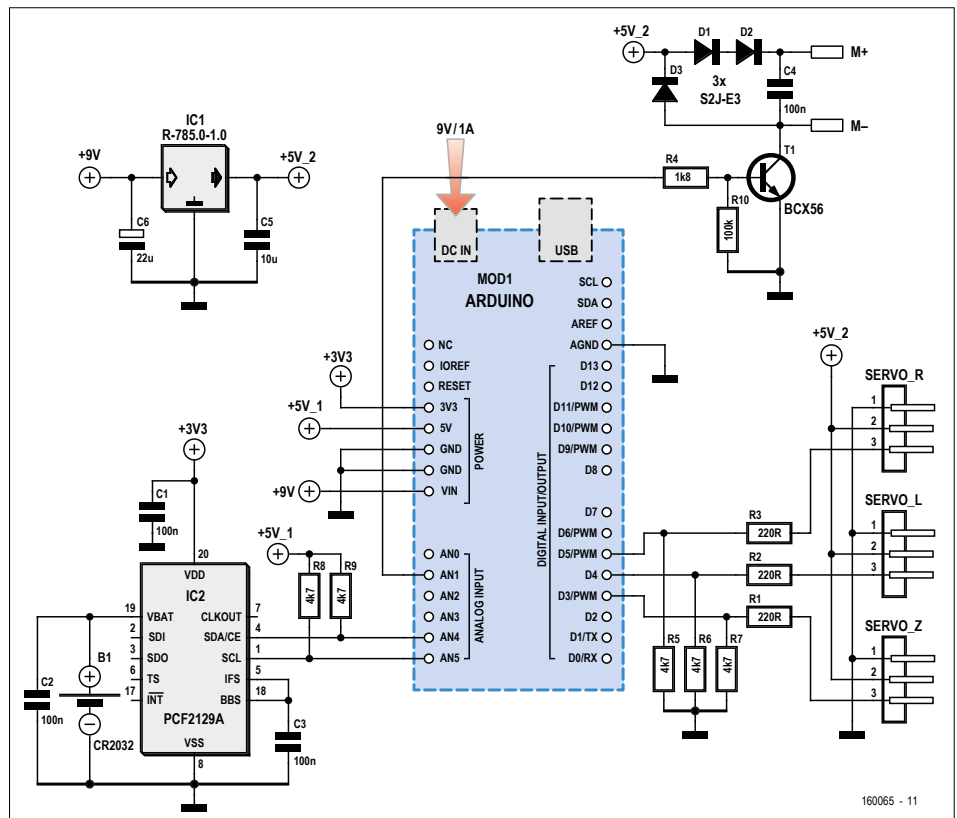


Figure 3. Schematic diagram of the shield designed specifically for the Sand Clock. It contains a DC/DC converter for powering the servos, a real-time clock IC, a driver for the vibration motors and connectors for the servos.

▶ You can also control the Sand Clock manually using commands

as possible, we designed an Arduino shield for the additional electronics. The shield PCB (**Figure 4**) is mainly populated with SMDs. To avoid potential soldering difficulties, the board in the kit is supplied pre-assembled. Only the connectors and

the DC/DC converter still have to be soldered on the board.

The clock is powered by a standard switch-mode AC adapter with an output voltage of 9 to 12 V. The vibration motors and servos draw a fair amount of current,

Advertisement



HAMMOND
MANUFACTURING®

Enclosures and Platforms for Pi and Arduino

www.hammondmfg.com/1593HAM.htm

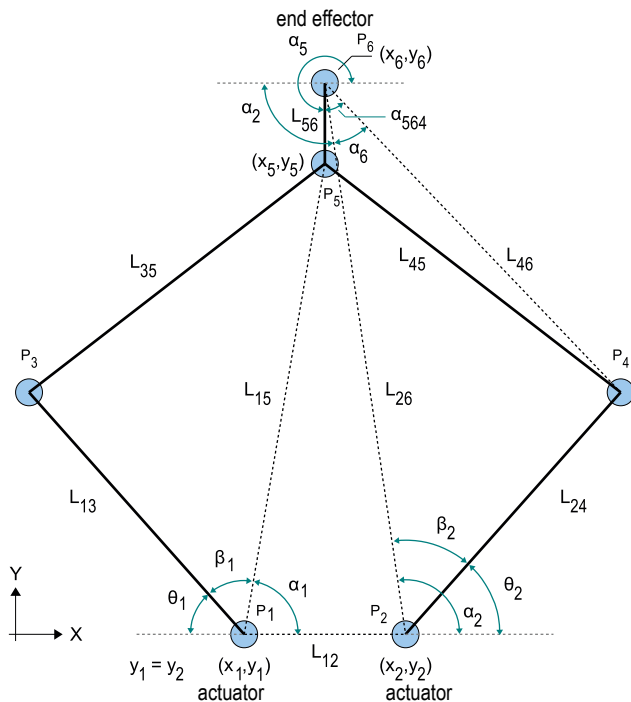
www.hammondmfg.com/1593HAMEGG.htm

01256 812812

sales@hammondmfg.eu



Inverse Kinematics



A 'kinematic chain' in robotics is a mathematical model of a mechanical system consisting of a set of rigid bodies and joints. The motions of kinetic chains can in turn be modelled using mathematical equations.

Forward kinematics means that the configuration of the kinematic chain is calculated based on the positions of the joints. Inverse kinematics is the opposite. It takes the desired configuration of the kinematic chain as the starting point and calculates the joint positions necessary to achieve that configuration, usually with the objective of determining the position and trajectory of an end effector or manipulator.

Outside the domain of robotics, inverse kinematics is also used in 3D animations and computer games. The motions of the human body and of animals can also be modelled this way.

When we apply inverse kinematics to the sand clock, the kinematic chain consists of four rigid bodies and five pivot joints, as shown in the accompanying **illustration**. Two of the five pivot joints have an actuator, which in this case is a servo. The end effector here is the pen.

The objective of inverse kinematics for the Sand Clock is to determine the angles θ_1 and θ_2 of the actuators (the positions of the two servos) for a given pen position (x_6, y_6) . Since the pivot joints have only one degree of freedom and there are only two actuators, the mathematical equations are fairly simple and can be handled by an Arduino module based on an ATmega328p microcontroller.

To simplify the calculations, the left and right servos are assigned the same Y position: $y_1 = y_2$. The diagram shows the various positions, lengths and angles. Some of them are constant, and others are variable.

Constant: (x_1, y_1) , (x_2, y_2) , L_{13} , L_{24} , L_{35} , L_{45} , L_{46} , L_{56} , α_{564}

Variable: (x_5, y_5) , (x_6, y_6) , L_{15} , L_{26} , L_{46} , α_1 , β_1 , θ_1 , α_2 , β_2 , θ_2 , α_5 , α_6

First we determine the angle θ_2 . We derive this angle from α_2 and β_2 . α_2 is the angle between the X axis and the line passing through P_2 (right servo) and P_6 (pen):

$$\alpha_2 = \arctan 2 \left(\frac{y_6 - y_2}{x_6 - x_2} \right)$$

We use the sides of triangle P_2 - P_4 - P_6 to calculate angle β_2 . Length L_{46} is known. We calculate length L_{26} as follows:

$$L_{26} = \sqrt{(x_6 - x_2)^2 + (y_6 - y_2)^2}$$

Now we can determine β_2 :

$$\beta_2 = \arccos \left(\frac{L_{26}^2 + L_{24}^2 - L_{46}^2}{2 \cdot L_{26} \cdot L_{24}} \right)$$

θ_2 is thus:

$$\Theta_2 = \alpha_2 - \beta_2$$

We use the triangle P_1 - P_3 - P_5 to calculate angle θ_1 . The position of P_5 (x_5, y_5) is a variable which first has to be calculated. The triangle P_4 - P_5 - P_6 is a rigid body with constant sides and angles.

In order to calculate (x_5, y_5) , we assume that triangle P_4 - P_5 - P_6 with side L_{56} rotates about P_6 through an angle α_5 as indicated in the diagram.

First we determine the angle α_6 :

$$\alpha_6 = \arccos \left(\frac{L_{26}^2 + L_{46}^2 - L_{24}^2}{2 \cdot L_{26} \cdot L_{46}} \right)$$

We already know angle α_2 . Now we can determine angle α_5 :

$$\alpha_5 = \pi + \alpha_2 + \alpha_6 - \alpha_{564}$$

The position of P_5 is then given by:

$$\begin{aligned} x_5 &= x_6 + L_{56} \cdot \cos(\alpha_5) \\ y_5 &= y_6 + L_{56} \cdot \sin(\alpha_5) \end{aligned}$$

Now we return to angle θ_1 . We derive this angle from α_1 and β_1 . α_1 is the angle between the X axis and the line passing through P_1 (left servo) and P_5 :

$$\alpha_1 = \arctan 2 \left(\frac{y_5 - y_1}{x_5 - x_1} \right)$$

We use the sides of triangle P_1 - P_3 - P_5 to calculate angle β_1 . Length L_{35} is known. We calculate length L_{15} as follows:

$$L_{15} = \sqrt{(x_5 - x_1)^2 + (y_5 - y_1)^2}$$

β_1 is then:

$$\beta_1 = \arccos \left(\frac{L_{15}^2 + L_{13}^2 - L_{35}^2}{2 \cdot L_{15} \cdot L_{13}} \right)$$

Finally there is θ_1 :

$$\Theta_1 = \pi - \beta_1 - \alpha_1$$

These calculations are implemented in the sketch in the function `pen_calc`.

so it is not a good idea to use the 5 V supply voltage from the Arduino board. The 5-V voltage regulator on the Arduino Uno is a low-drop linear regulator with a maximum rated load current of 800 mA. That does not provide much margin, and due to the low efficiency things get pretty warm while the time is being written. With a USB connection between a computer and the clock, there is also a risk that the computer's USB supply voltage will collapse if you forget to plug in the AC adapter.

For the above reasons, we decided to provide a separate power supply for the motors, isolated from the supply for the Arduino. For this we use a switch-mode DC/DC converter in the form of a 78xx replacement with three leads (IC1), which is connected directly to the input voltage of the Arduino board. It provides an output voltage of 5 V at 1 A maximum with an efficiency of over 90%, with an input voltage in the range of approximately 9 to 12 V.

The vibration motors are driven by an NPN transistor (T1) under control of an Arduino I/O pin. The voltage on the vibration motors is reduced somewhat by diodes D1 and D2 in combination with the saturation voltage of the transistor (V_{CE-sat}). This is necessary because the operating voltage of the motors (according to the data sheet) is only 3 V.

The control signals for the servos are supplied directly by three I/O pins of the Arduino board. Three connectors are mounted on the shield for the three servos. Resistors R1, R2 and R3 are there to protect the Arduino if one of the servo cables is accidentally plugged in with the wrong orientation. Resistors R5, R6 and R7 keep the levels on the data lines to the servos stable when the Arduino micro-controller is in sleep mode.

A real-time clock IC is included to maintain the correct time, so the clock does not have to be set again every time it is powered up. For this we chose an PCF2129A (IC2), which has a built-in crystal. This IC is inexpensive and very precise, with internal temperature compensation and an accuracy of 3 ppm. A CR2032 button cell serves as a backup battery with a lifetime of about ten years. The RTC IC is not suitable for a 5-V supply voltage, but fortunately the Arduino board provides a 3.3-V supply voltage on the shield connectors. Equally fortunate is the fact that the I²C pins of the IC are

5-V tolerant, so they can simply be connected to the I²C interface of the Arduino without any need for level shifting.

Arduino sketch

The sketch for the Sand Clock is designed for the Arduino Uno, and it also works with the Elektor Uno R4. The sketch includes all the functions necessary for calibration of the Sand Clock after assembly (described in detail in the assembly instructions, which can be downloaded

together with the sketch [2]), as well as the clock functions which write the time in the sand.

We also made several internal functions of the sketch accessible via commands, so that you can move the pen under manual control, draw figures in the sand, and switch the vibration motors on and off. For this purpose the sketch has two modes: autonomous and command. In autonomous mode the Sand Clock oper-

Component List

Resistors

R1,R2,R3 = 220 Ω , SMD 0805
R4 = 1.8k Ω , SMD 0805
R5-R9 = 4.7k Ω , SMD 0805
R10 = 100k Ω , SMD 0805

Capacitors

C1-C4 = 100nF, SMD 0805 MLCC
C5 = 10 μ F 10V, SMD 1206 MLCC
C6 = 22 μ F 16V

Semiconductors

D1,D2,D3 = S2J-E3
T1 = BCX56
IC1 = SIP3 5V/1A DC/DC converter (Würth Elektronik # 173 010 578)
IC2 = PCF2129A

Miscellaneous

K1 = set of SIL pinheaders, 0.1" pitch (1 pc. 10-pin, 2 pc. 8-pin, 1 pc. 6-pin)
SERVO_Z,SERVO_L,SERVO_R = 3-pin right-angled pinheader, 0.1" pitch
B1 = CR2032 button cell with holder
Arduino UNO R3 or equivalent

Mechanical parts:

6 pcs. bolt, M2x10, Pozidrive/Phillips
6 pcs. bolt, M2.5x8, zinc-plated steel, Pozidrive DIN 7985A
6 pcs. bolt, M2.5x12, zinc-plated steel, Pozidrive DIN 7985A

2 pcs. bolt, M3x6, zinc-plated steel, Pozidrive DIN 7985A
2 pcs. bolt, M3x8, zinc-plated steel, Pozidrive DIN 7985A
15 pcs. bolt, M3x10, zinc-plated steel, Pozidrive DIN 7985A
1 pc. bolt M4x30 plastic, Phillips (sharpen end with pencil sharpener)
6 pcs. nut, M2, zinc-plated steel, DIN 934
7 pcs. nut, M3, steel, DIN 934
3 pcs. locking nut, M3, zinc-plated steel, DIN 985
1 pc. nut, M4, polyamide
2 pcs. washer, M3, zinc-plated steel, DIN 125A
4 pcs. washer, M3, plastic, DIN 125A
4 pcs. standoff, 3mm high, polyamide, for M3
4 pcs. spacer, 25mm, M/F M3, nickel-plated brass (min. 33mm total height, e.g. TME TFM-M3X25/DR213)
2 pcs. cable clip, polyamide, Panduit type CCS25-S10-C
4 pcs. self-adhesive rubber foot
3 pcs. micro servo, Tower Pro MG90S or MG90 with metal cogs
2 pcs. vibration motor, round, 6mm diam., type VM6ZK273 (JPR Electronics # 450-007)
Extruded (XT) PMMA, 3mm, clear, laser-cut
Fine-grain white sand (see text)
Nourishment coloring additive (optional)

A DIY kit including all parts and the SMD-preassembled shield is available through the Elektor Store.

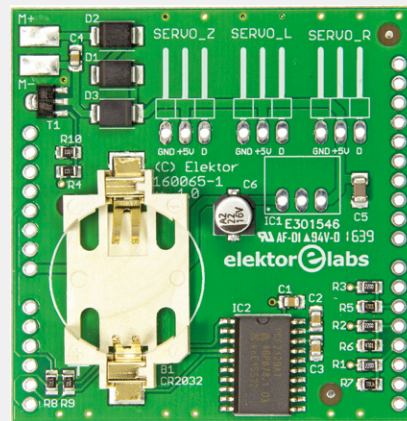
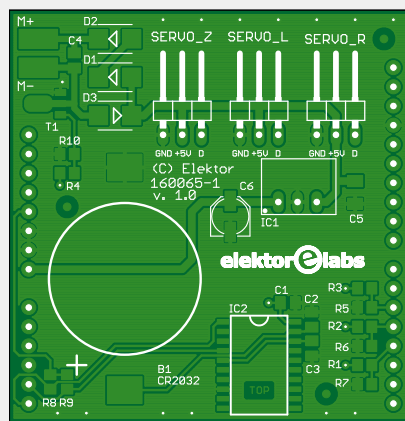


Figure 4. The shield PCB is mainly populated with SMDs, which are pre-assembled on the board in the kit.

ates independently. In command mode the sketch executes the commands you send to the sand clock.

After uploading the sketch to the Arduino, you can use the serial monitor of the Arduino IDE to send commands to the sand clock. Select 9600 baud and something other than “No line ending” in the adjacent selection box. The commands consist of letters (e.g. `svd`), optionally followed by one or more parameters (for example, `ps -20.55 +55.8`). Press the Enter key to send a command. The Arduino returns a response for most of the commands. All available commands are listed at the start of the sketch. For example, you can use the command `pa` to draw an arc.

Now let’s get back to the original subject. During the assembly process you run through the calibration procedure. This procedure requires you to enter a series of commands, in some cases with accompanying parameters. Once all the settings have been made (see the command `sed`), they are written to the EEPROM (command `sew`). During start-up, the sketch checks the validity of the EEPROM content. If the EEPROM content is valid, it determines the start-up mode of the clock. In most cases you will probably set it to autonomous mode. If the content is invalid, the clock starts up in command mode, the same as the first time after you upload the sketch.

Note that the EEPROM of the microcontroller is not cleared when the sketch is uploaded. This means that you do not have to enter the settings again (and save them with the `sew` command) every time you upload the sketch. If necessary, you can use the command `sec` to erase the EEPROM.

In autonomous mode the Sand Clock periodically draws the current time in the sand. This drawing action consists of a number of pen motions in the horizontal plane, as well as up and down motion. The sketch uses the current X/Y pen position as the starting point for all types of motion: drawing a straight line, tracing an arc, drawing a character, or repositioning the pen.

The function `pen_set` moves the pen to the stated X/Y position. First the coordinates are converted to the angular

positions of the left and right servos, expressed in radians. This calculation is performed by the function `pen_calc` and can either succeed or fail. The calculation fails when the stated coordinates cannot be reached by the pen. If the calculation succeeds, the resulting servo positions are converted from radians to PWM pulse widths for the left and right servos.

The sketch uses floating point operations to calculate the inverse kinematic functions. The floating-point variables have a width of 32 bits. The standard used here is IEEE 754. All of these properties are determined by the AVR GCC compiler used to compile the sketch.

All positions and distances are expressed in millimeters. These values correspond to the real world. The sketch includes definitions of the physical dimensions of the mechanical parts of the clock. Angles are expressed in radians.

The diagram in **Figure 5** shows the positions (in millimeters) of the left and right servos with respect to the sand bed, as defined in the sketch.

During drawing in the sand the pen is moved in steps with a maximum step size of 0.25 mm. Despite the many calculations, the processor is more than fast enough, so the code waits 5 ms between successive changes in pen position during drawing to prevent the pen from being pushed through the sand too quickly.

You can interrupt autonomous mode at any time by sending an end of line (EOL) character over the serial interface. Press the Enter button in the serial monitor to switch the sketch to command mode. Bear in mind that the transition to command mode may take a while when the clock is busy writing in the sand. You can return to autonomous mode by sending the command `ma`.

Construction

In this article we only provide a brief description of how to build the Sand Clock. If you want to build the project, please consult the assembly instructions [2] for a more extensive description with many detailed photos.

Start by mounting the connectors and the DC/DC converter on the Arduino shield board with the pre-assembled SMD components. Then plug the shield onto the Arduino board and connect the servos. Pay attention to the orientation of the

connectors. Then connect the AC adapter to the Arduino board and connect the USB port of the Arduino to a computer on which the Arduino IDE is installed. Compile the sketch [2] and load it into the Arduino.

After the sketch is loaded and the Arduino is reset, the servos are automatically moved to their midpoint positions (pulse width 1500 ms). Disconnect the Arduino from the computer and disconnect power to the Arduino. Then remove the shield and disconnect the servos so they can be mounted on the clock. If you have a servo tester, you can use it instead to set the servos to the midpoint position. Now it’s time to start the mechanical assembly. Attach four rubber feet to the bottom of the base plate and mount the four supports for the sandbox. Next assemble the base frame with the lift servo, and then mount this assembly along with the Arduino and the shield on the base plate.

Now it’s time for the left and right servos of the pantograph mechanism. Connect all the servos to the shield again and assemble the arms and pen of the pantograph. When you assemble the arms of the pantograph, thread the screws directly into the acrylic plastic. That works quite well thanks to the previously mentioned thermoplastic properties of this material: the heat generated by friction when the screws are threaded in effectively causes the plastic to flow around the threads. This mounting technique minimizes the mechanical play of the arms. However, the mounting hole for the pen does have to be threaded. If you cut your own parts using a laser cutting machine, you have to tap that hole with an M4 thread tap. Mount the sandbox, but leave the vibration motors aside for the time being. Now it’s time for the calibration. Connect an AC adapter to the Arduino board and plug it in. Then connect the Arduino board to the computer through a USB cable. Now you can use the serial monitor of the Arduino IDE to send commands that set the servo positions. To calibrate the servos, you have to determine the positions where they are exactly vertical and horizontal. These positions are then stored in the EEPROM. The next task is to calibrate the lift servo. Put the sandbox in place and position the lift servo so that the tip of the pen hovers a few millimeters above

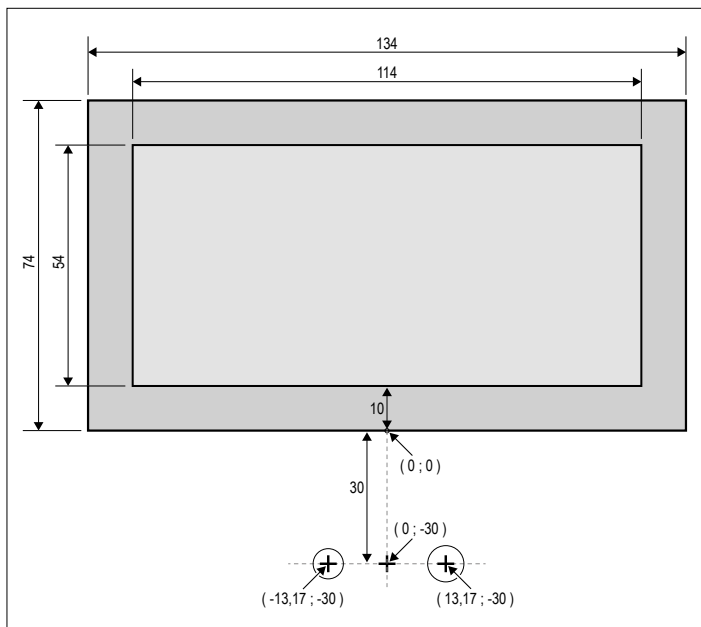


Figure 5. The positions (in millimeters) of the left and right servos relative to the sand bed, as defined in the sketch.

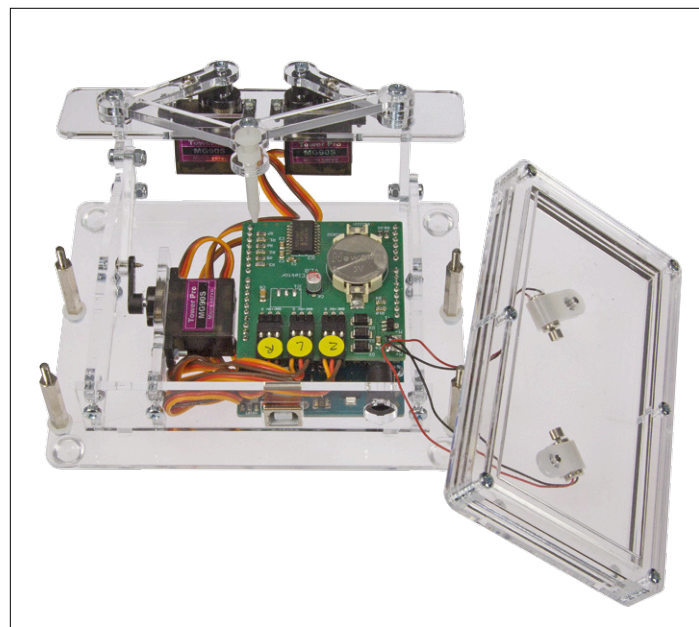


Figure 6. The structure of the Sand Clock is clearly visible here.

the bottom of the box, regardless of the X/Y position. You can also adjust the distance by turning the pen higher or lower in the mount. Once the pen is in the right position, tighten the plastic M4 nut to hold it in place. Next you have to set the middle position and the highest position of the lift servo. These are not critical, and usually you can simply take the values from the assembly instructions.

Write the settings to the EEPROM and then set the clock to the right time. Now you can do a dry run with the clock (without sand) to check its operation. If that all goes well, you can set the clock to autonomous mode so that it automatically writes the time at fixed intervals, and then disconnect it from the computer.

Now disconnect the Arduino power source and mount the vibration motors underneath the sandbox. Twist the matching leads of the two motor cables together and solder the ends to the solder pads on the shield. Be careful not to touch the acrylic parts with the hot soldering iron. Then place the sandbox on the supports, taking care that the vibration motors do not touch any leads and are free to turn.

Finally, pour the sand into the sandbox. That completes the construction of the sand clock.

If you wish, you can give the sand a special tint. To do so, put the sand in a tin with a lid and add a few drops of food color. Then shake the closed tin until the sand has a uniform color. Repeat if necessary until you obtain the desired color. Let the sand dry completely before pouring it into the sandbox. ◀

(160065-1)

Web Links

[1] www.elektormagazine.com/160065

[2] www.elektor.com/sandclock-160065-71

Advertisement

USB Add USB to your next project.
It's easier than you might think!

DLP-USB1232H: USB 2.0 UART/FIFO

HIGH-SPEED
480Mb/s



Only \$28.95!

- Multipurpose: 7 interfaces
- Royalty-free, robust USB drivers
- No in-depth knowledge of USB required
- Standard 18-pin DIP interface; 0.6x1.26-inch footprint

DLP-IO8-G

8-Channel Data Acquisition



Only \$29.95!

- 8 I/Os: Digital I/O
- Analog In
- Temperature
- USB Port Powered
- Single-Byte Commands

DLP-IOR4

4-Channel Relay Cable

DLP-TH1b

Temp/Humidity Cable

DLP-RFID1

HF RFID Reader/Writer

DLP-FPGA

USB-to-Xilinx FPGA Module

DLP
Design

www.dlpdesign.com

Super Simple Dice

Cheap and microcontroller-free!

By **Roy Aarts** (Elektor Labs)

This simple electronic dice was designed specifically for beginners in electronics. The board has a spacious layout, with only a handful of components, which are all through-hole mounted. There is no microcontroller in this circuit; all the hard work is carried out by two CMOS ICs.

For several decades the dice has been a willing subject for a small electronics circuit. Over the years, Elektor has already designed and published a large number of circuits for a dice. It often became a challenge to use as few components as possible, but this sometimes resulted in some small flaws, such as a less than realistic representation of the spots. (The number of LEDs lit up did give a useable result, but the layout of the LEDs was not the same as that of the spots on a real dice). We've even managed to design a dice using a single IC from the 4000 series, complete with a touch-sensitive control. The only 'disadvantage' was the need for a 9-V battery (fairly large and not cheap).

The aim of the design published here was to make a simple circuit with LEDs, which didn't have any 'difficult' components, and which was easy to construct. A battery holder with a CR2032 button cell is mounted on the board, which makes the circuit easy to carry around. This configuration also makes it very easy to replace the battery. This is a very neat kit to make people interested in electronics, even if they've never soldered before. With a bit of help, anybody could build this!

The circuit

The complete circuit is shown in **Figure 1**. This consists of an oscillator (IC2) and a decade counter (IC1), which has 7 LEDs connected to its outputs.

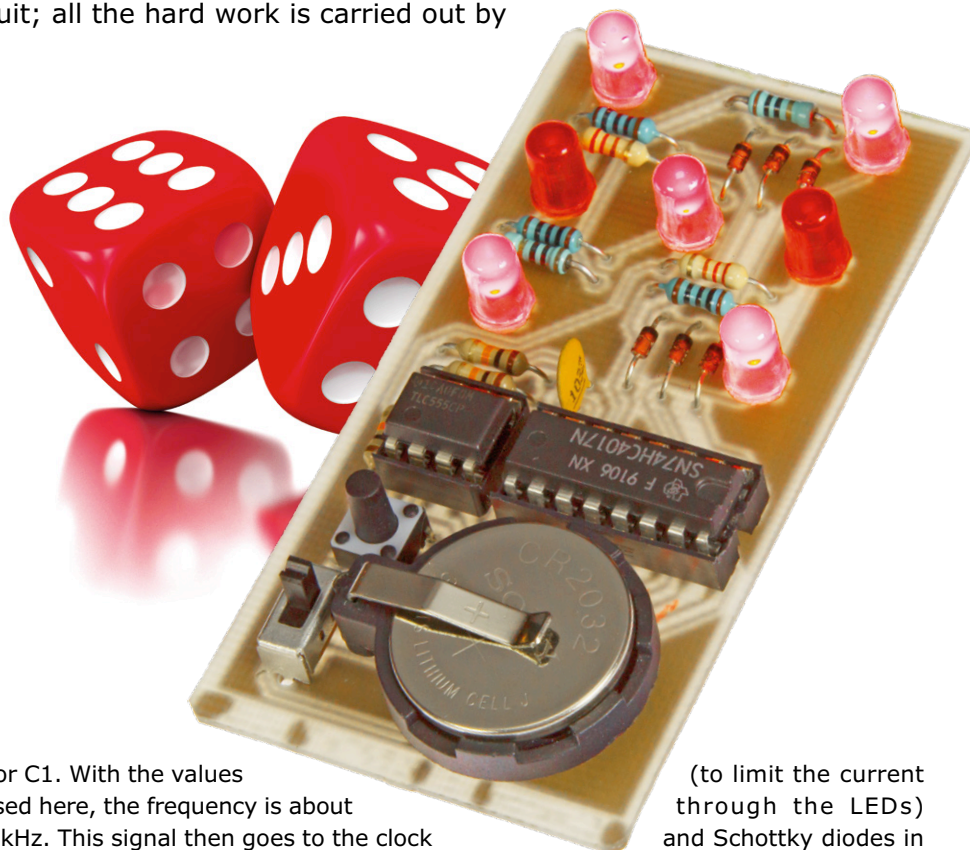
The well-known 555 timer was selected for IC1. It is configured here as an astable multivibrator by connecting pins 2 and 6 (Threshold and Trigger) together. The oscillator frequency is determined by the values of resistors R1 and R2, and capac-

itor C1. With the values used here, the frequency is about 5 kHz. This signal then goes to the clock input (pin 14) of decade counter IC1, which is a 74HC4017. By connecting the Q6 output (pin 5) to the Reset input (pin 15), the counter can only count from 1 to 6, when it will receive a reset pulse. This is exactly what we need for our dice. The pulses at the clock input are only counted by IC1 when its enable input is low. When a pushbutton and a pull-up resistor are connected to this pin, we can make IC2 count for as long as S1 is pressed. When the pushbutton is released, one of the Q1-Q5 and CO outputs will remain at a logic high level. Because the 555 operates at such a high frequency, it's not possible to manipulate the selected output with S1, so the result is completely random!

The seven high-efficiency LEDs (LED1-LED7) are of course arranged in the same pattern as that found on a real dice. They have been connected via series resistors

(to limit the current through the LEDs) and Schottky diodes in a certain configuration to the six outputs (Q1-Q5 and CO) of IC1. The diodes prevent short circuits between an active output and any non-active outputs that drive the same LEDs. Since the carry-out output of the IC (pin 12) is high after a reset and when outputs Q0 to Q4 are active, the diode matrix produces the following numbers: pin 2 high: 3, pin 4 high: 4, pin 7 high: 5, pin 10 high: 6, pin 1 high: 1, pin 12 high: 2.

Since the supply voltage to the circuit is so low (3 V), the diodes used are Schottky types in order to keep the voltage drop across the diodes as small as possible. A lot of experimenting with the values of the series resistors took place in the prototype, because the outputs of a 74HC4017 can only supply a small current at that supply voltage, and the output voltage drops at greater currents. The values of



the series resistors have been adapted to take account of the number of LEDs that the outputs have to drive at the same time, making the brightness of all the LEDs roughly equal.

A 3 V lithium button cell has been used to power the circuit. It's been mounted in a holder on the board to make it easy to replace. The dice can be turned on and off with the help of slide switch S2. Since the current through the LEDs is only several milliamps, a button cell should last several hours.

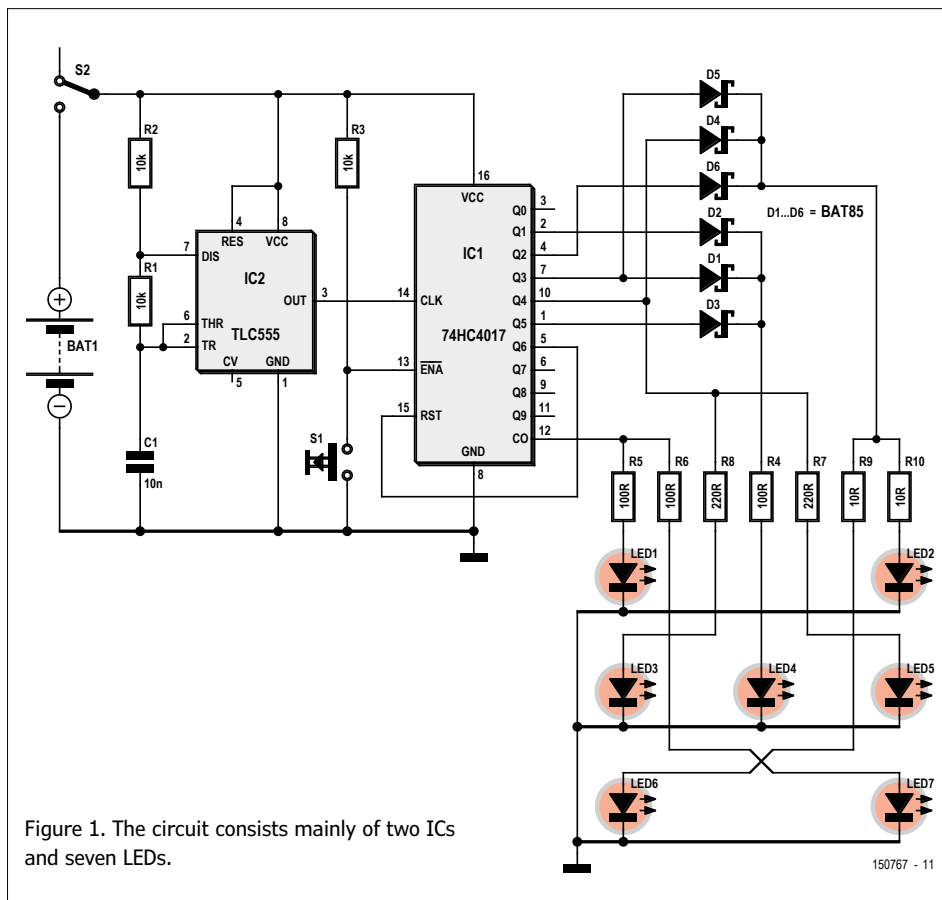
Construction

The printed circuit board shown in **Figure 2** has a spacious layout, with slightly larger tracks and pads than usual, to make it easier to construct for people with less experience in soldering. For the enthusiast, an attractively priced kit consisting of the board and all the components can be obtained from the Elektor shop.

The construction of the dice isn't difficult, although a lot of things may not be clear to beginners in electronics. We won't go into detailed descriptions here (such as color codes for resistors), but we would suggest that those with little experience of electronics and soldering should construct the circuit with the help of somebody more knowledgeable.

In any case, start with the mounting of resistors, followed by the diodes (the band on one end of the diode corresponds to the line (cathode) on the PCB), and the capacitor. Next, the two IC sockets should be soldered onto the board, followed by the LEDs. The cathode (again indicated by a line on the PCB) is denoted by the shorter of the two leads or by the flat side on the casing. Finally it's the turn of the switch and the pushbutton, followed by the battery holder. Now put the two ICs in their sockets, making sure that the indentation on each IC corresponds to that shown on the board. And that's it! You should now check the board to make sure that every component has been mounted correctly and is in the right place. Once you're satisfied that everything is ok, put the button cell in its holder, switch the circuit on, and press on the pushbutton. You'll now be shown a number between 1 and 6 on the LEDs. Congratulations, your (first?) circuit works! ◀

(150767)



▶ With a bit of help, anybody could build this!

Component List

Resistors

Default: 0.25W, 5%

R1,R2,R3 = 10kΩ

R4,R5,R6 = 100Ω

R7,R8 = 220Ω

R9,R10 = 10Ω

Capacitors

C1 = 10nF, 0.1" pitch

Semiconductors

D1,D2,D3,D4,D5,D6 = BAT85 Schottky diode

LED1-LED7 = LED, high-efficiency, red, 5mm

IC1 = 74HC4017

IC2 = TLC555

Miscellaneous

S1 = pushbutton, PCB mount, 6x6 mm

S2 = slide switch with changeover contact,

PCB mount, e.g. C&K OS102011MS2QN1C)

BATT1 = CR2032 battery holder, PCB mount

(e.g. Multicomp CH25-2032LF)

CR2032 button cell

PCB # 150767-1 or

Kit of parts including PCB and all components:

Elektor Store # 150767-71

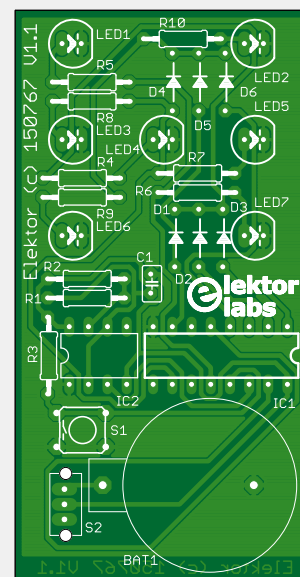


Figure 2. Everything's on this board, including the battery and the on/off switch.

Elektor SDR Reloaded (4)

Standalone solutions

By **Burkhard Kainka** (Germany)

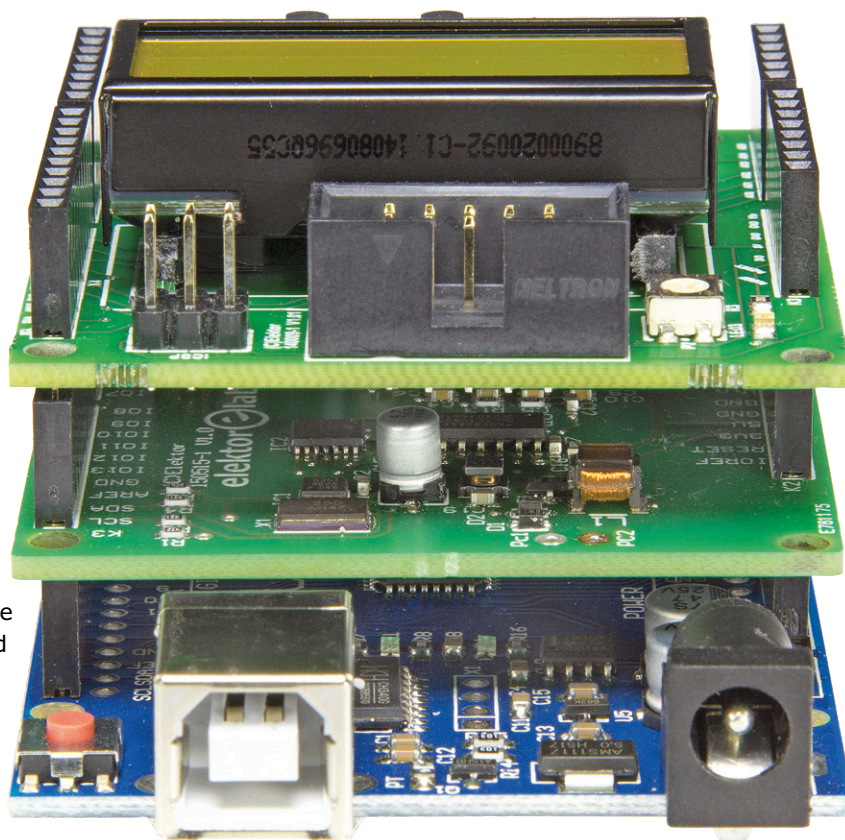
Can we also use the SDR Shield with just an Arduino and no PC at all? Certainly we can. It's entirely feasible (with a few restrictions) because all of the vital control functions are available on the Elektor Extension Shield. The only task that remains then is processing the IQ signals.

Your simplest option is to mix down the signals and listen to them directly. Even headphones plugged directly into the audio jack on the SDR Shield will produce something audible, albeit a little bit on the quiet side. The receiver is then operating as a direct mixer. All we need to do then is make a suitable control for the fine tuning.

Tuning with the Arduino

On the Elektor Extension Shield (no. 140009-91, published in July & August 2014; [1]) we have two push-buttons and a pot. We can use these to make a convenient control setup. S1 reduces the frequency, S2 raises it and with the pot we can adjust the increment size between 20 Hz and 100 kHz. On the upper line the LCD indicates the current frequency and below this the increment we have set (**Figure 1**). We can now tune to a far finer degree than the 1-kHz steps we had available previously. With this control we can not only change between frequency bands but also manage fine tuning for CW or SSB with ease. The frequency specified initially on start-up is 7000 kHz, enabling you to go immediately to the 40-m band and search for CW and SSB stations.

For compatibility reasons the new software for the Arduino/



SDR Shield/Extension Shield combo (ready for free download at [2]) still supports serial inputting of frequencies, so you can still carry out tuning from your PC. You can see how a frequency set up on a PC is displayed in kHz on the LCD (**Figure 2**). However, as soon as one of the two pushbuttons on the Shield is activated, the display changes to Hertz and indicates also the precise VFO frequency without the 12-kHz offset (see **Listing 1**).

You can proceed on the PC and initiate an SDR program next. After that it's your choice entirely whether you drive the receiver



Figure 1. Display on the Elektor Extension Shield: 7001.040 kHz, adjustable in 20 Hz steps.

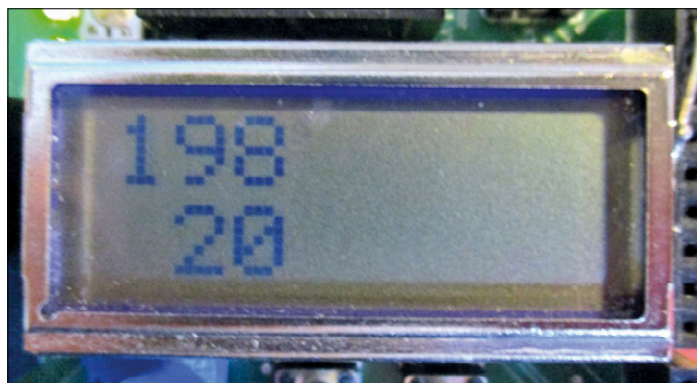


Figure 2. 198 kHz, set up using the PC.

Listing 1. Excerpts from the standalone software.

```

void loop(void)
{
  pot = analogRead(A3);
  fstep = 1000000;
  if (pot > 100) fstep = 1000000;
  if (pot > 200) fstep = 500000;
  if (pot > 300) fstep = 100000;
  if (pot > 400) fstep = 20000;
  if (pot > 500) fstep = 5000;
  if (pot > 600) fstep = 1000;
  if (pot > 700) fstep = 200;
  if (pot > 800) fstep = 100;
  if (pot > 900) fstep = 20;

  if (fstep != fstepOld){
    lcd.setCursor(1, 1);
    lcd.print(fstep);
  }

  lcd.print(" ");
  fstepOld = fstep;

  if (Serial.available()) {
    freq = Serial.parseInt();
    if (freq > 0){
      lcd.setCursor(0, 0);
      freqHz = freq * 1000 - 12000;
      lcd.print(freq);
      lcd.print(" ");
      setfreq (freqHz);
    }
  }
  if (digitalRead(A0) == 0){
    freqHz = freqHz - fstep;
    lcd.setCursor(0, 0);

    //freq=freqHz/1000;
    lcd.print(freqHz);
    setfreq (freqHz);
    lcd.print(" ");
    delay (200);
  }
  if (digitalRead(A1) == 0){
    freqHz = freqHz + fstep;
    lcd.setCursor(0, 0);
    //freq=freqHz/1000;
    lcd.print(freqHz);
    setfreq (freqHz);
    lcd.print(" ");
    delay (200);
  }
}

```

using the PC or with the controls on the Shield. But we can also do this without involving a PC at all. The simplest way is to use the SDR Shield as a direct mixer.

Direct mixer

A direct mixer transposes an RF signal down to the audio range in one step. Of course the Shield is actually intended for an IF of 12 kHz, but the IF stages provide sufficiently wide bandwidth to allow trouble-free operation down in the lower ranges up to 3 kHz. The only thing else that we could ask for is some extra amplification.

For that reason we built a small stereo headphone amplifier with presettable volume control (**Figure 3**). Stereo provides a touch of special magic in combination with a basic direct mixer. The two signals *I* and *Q* are in fact phase-shifted by 90 degrees. With a little practice, the brain can make plenty out of this combination. A bit of phase difference can in fact be decisive, even in our spatial hearing. When you listen to (ideally many different) CW signals for long enough, you then start to develop a spatial impression that palpably reinforces selective hearing. Many users declare that they can even differentiate signals below and above the VFO frequency, thus eliminating the key disadvantage of direct mixers: there is no image suppression, you see.

We have now turned the SDR Shield into a veritable deluxe direct mixer, with a stable VFO across the whole range up to 30 MHz plus fine tuning, and on top of this, a stereo IQ output. The best thing is that the PC remains switched off and cannot cause any interference. Of course a receiver with no ALC whatsoever is a particular blessing on CW and SSB, as the noise level is not increased during pauses. Normally you often have to adjust the volume control when a very loud station comes in. The small headphone amplifier is constructed using the twin op-amp LM358. You could of course employ fewer components by using a special audio IC but probably everybody has

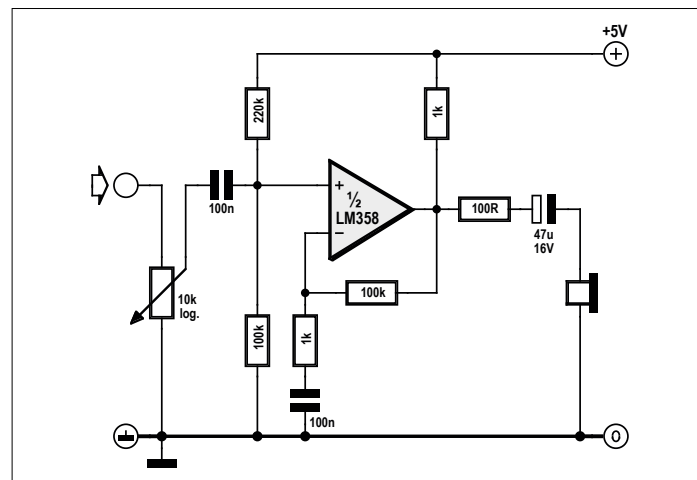


Figure 3. One channel of the headphone amplifier.

an op-amp in their component stores. The LM358 is prone to transient distortion if driven at high level into a relatively

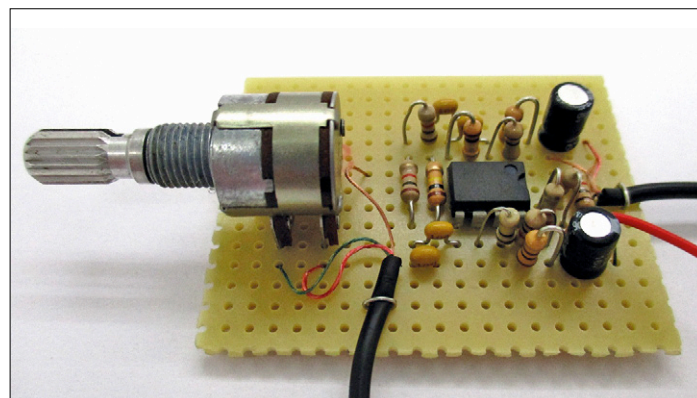


Figure 4. The experimental headphone amplifier on a scrap of perfboard.

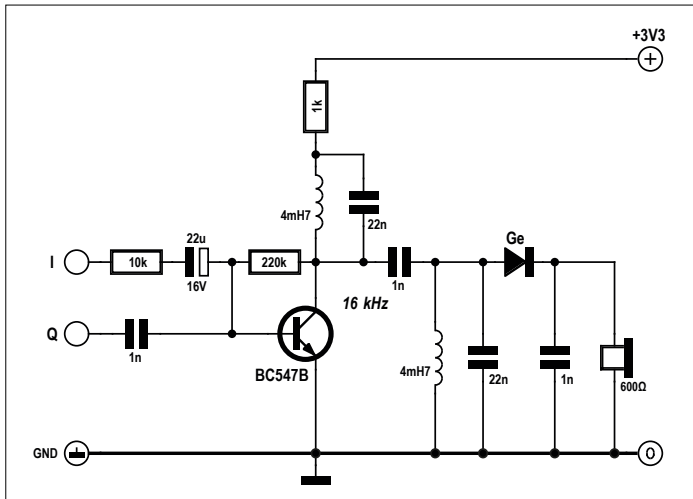


Figure 5. A diode detector.

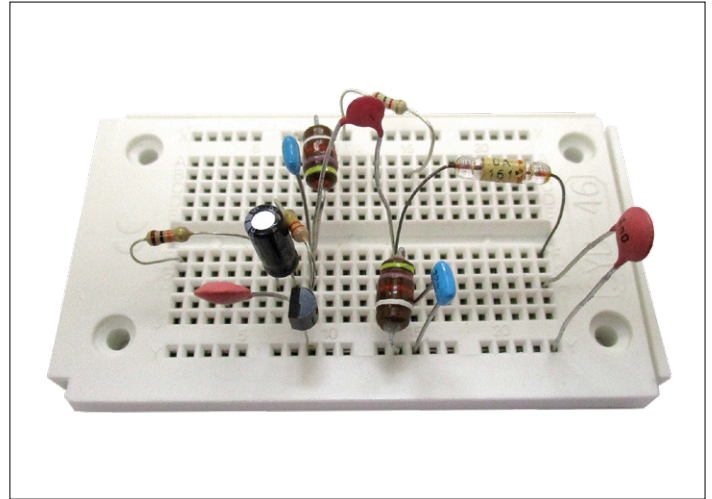


Figure 6. Diode detector constructed on a breadboard.

low-impedance load. This problem is mitigated by providing the 1-k Ω resistor from the output to the supply rail, because at smaller modulation levels only the lower part of the push-pull output is active. In series with the headphones we also have a 100- Ω resistor. On the one hand this reduces the work of the op-amp and on the other it eliminates any risk of damaging your hearing. Limiting comes into play for the amplifier in time to ensure that the maximum power to the headphones remains within bounds at all times. Incidentally, the LM358 can be driven almost down to GND level but not quite all the way up to the power rail. Consequently the open circuit voltage is about a third of the operating voltage.

The amplifier can raise the signal level by up to 40 dB. This is more than you need in most situations, so normally the volume control is throttled well back. Receiving CW stations is truly a pleasure with this setup. Even SSB comes in well, because the frequency can be set so conveniently and precisely. You can listen for hours like this.

But what's it like when listening to AM? Actually this is not the right job for a direct mixer. But does it work nevertheless?

It does, with some limitations. You must set the VFO on the zero-beat position, i.e. as precisely as possible on the exact carrier frequency of the transmitter. Then you can hear the modulation clearly, admittedly with an additional beat overlaid at most times. Phase-locked tuning is not possible, you see. Nevertheless, receiving AM stations is certainly feasible. However, it will work a lot better with the AM IQ demodulator to be described next.

The IQ detector

Under normal circumstances the SDR software processes a 12-kHz IQ signal. The question now arises whether you cannot also 'solidify' signal processing into hardware format. Our goal is now to create an AM demodulator for IQ signals. This turns out to be simpler than expected.

The first attempt (**Figures 5 and 6**) was based on an IF filter with two fixed inductances. An operating frequency of 16 kHz was determined more or less by chance, because 22 nF capacitors were available to hand. With a coupling capacitor of 1 nF a good band filter with suitable bandwidth was the result.

The two signals *I* and *Q* need to be mixed together with a

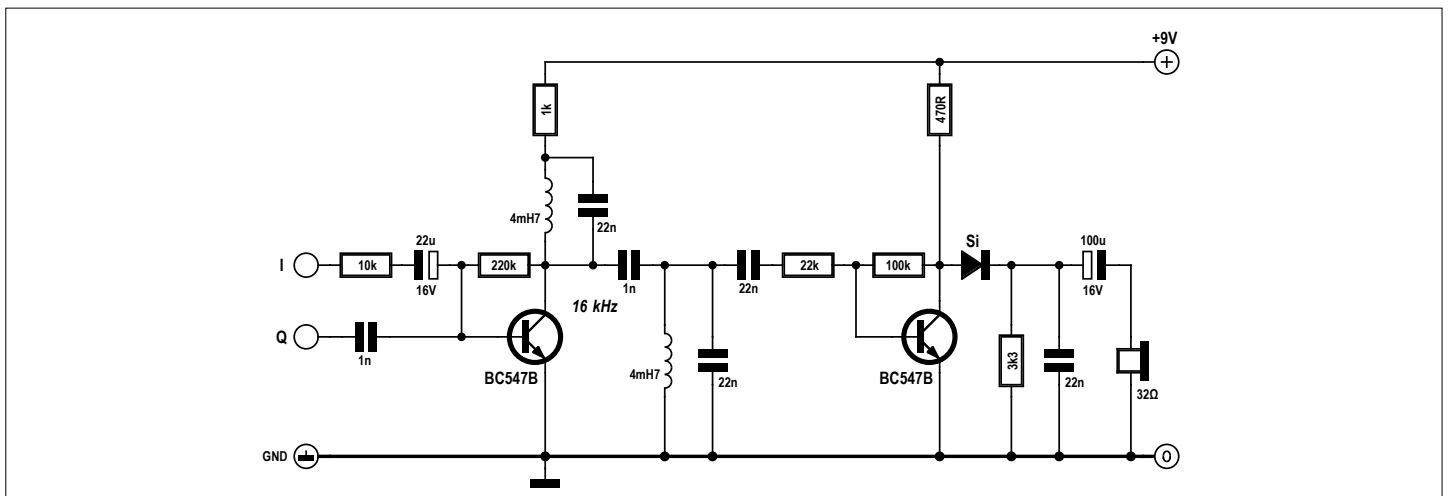


Figure 7. Expansion with an IF stage.

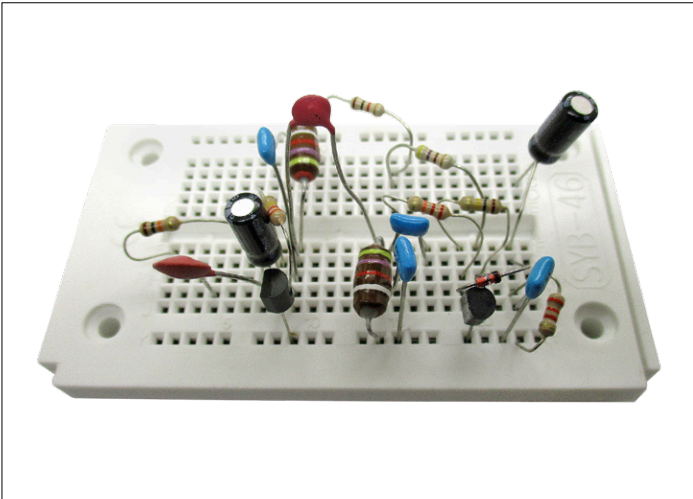


Figure 8. The complete circuit from Figure 7 on a breadboard.

90-degree phase shift. At 16 kHz this works out pretty accurately, with the 1-nF capacitor having a capacitive impedance of 10 k Ω here. At this operating frequency the *Q* channel is shifted by 90 degrees, whereas the *I* channel passes through without any phase shift. As expected, signals are amplified at +16 kHz and canceled out at -16 kHz. Effective image suppression is the result. Our first signal processing task is now completed.

The second task is signal filtering, which is what the band filter is for. This always worked fine in old tube radios and still does here, only at a lower frequency. After the filter we have the AM demodulator, in the form of a diode detector. To underscore the historical continuity an OA61 more than 50 years old is employed (you can also use a 1N60A or something similar of course). Beyond this come some high-impedance headphones or low-impedance 'phones with a matching transformer. Already we can hear something. The sound is very pleasant and easy on the ear, which is probably thanks to the band filter in the main. The first attempt worked well only for the strongest signals and even then it produced no great volume level. For that reason we added a further IF amplifier stage (**Figure 7**). Now the AM rectifier has some bias, which is why a silicon diode also works here. The result is good sensitivity and plenty of volume using 32-ohm headphones. Naturally the AM detector is also suitable for connecting to a loudspeaker amplifier.

To a greater or lesser extent, the experiments just described for standalone SDR are all simple compromises. Of course the perfect solution would provide genuinely digital signal processing with a DSP. But could we maybe manage this using an Arduino? That would be a major achievement indeed. If it's at all feasible, it's likely to be an Elektor reader who would take on this task. Have fun experimenting with your SDR Shield!

(160165)

Web Links

- [1] www.elektormagazine.com/140009
 [2] www.elektormagazine.com/160165

mouser.com

The Newest Products for Your Newest Designs®

The widest
selection
of the
newest
products.

Over 4 Million
products from over
600 manufacturers.



MOUSER
ELECTRONICS

Authorised distributor of semiconductors and
electronic components for design engineers.

SUPRA 2.0 MC Version

Getting the most out of a tiny signal

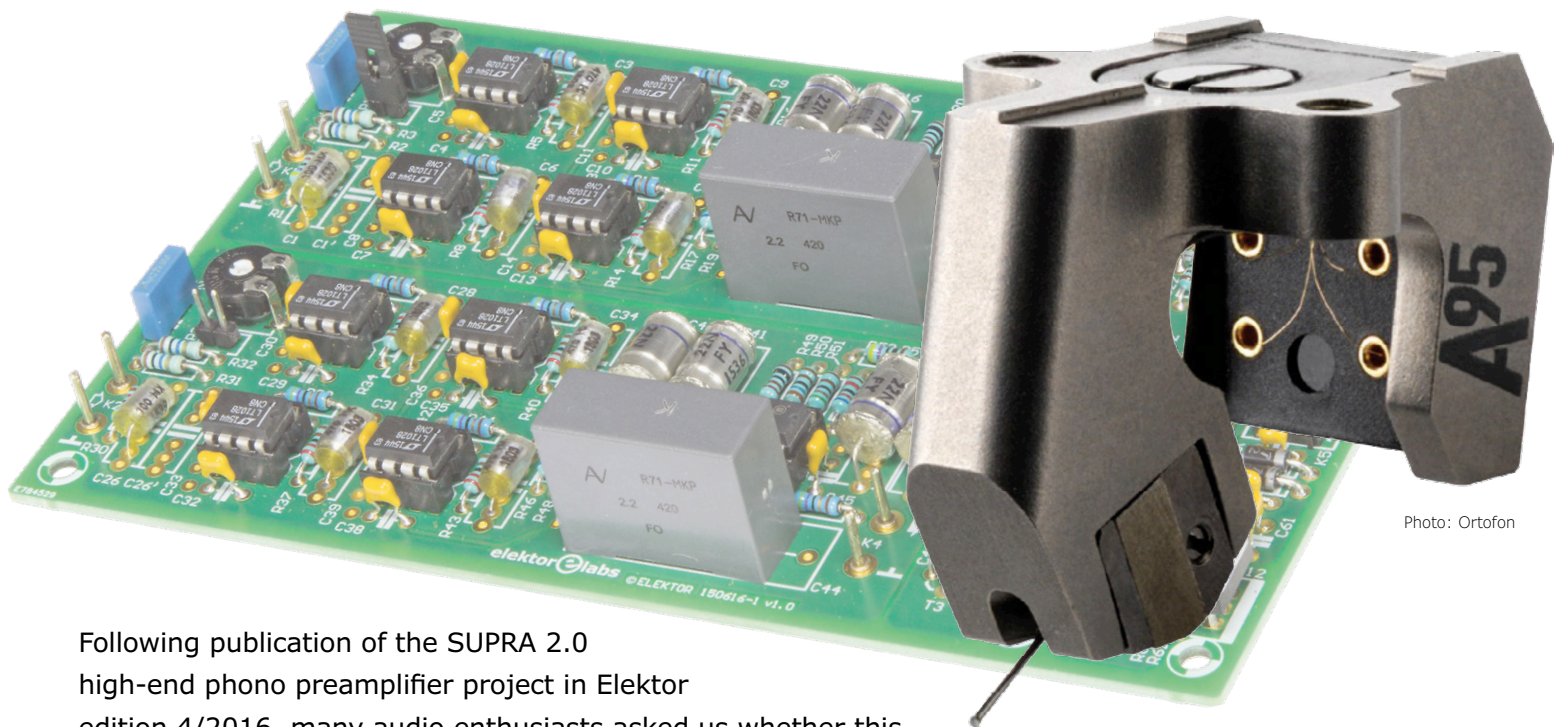


Photo: Ortofon

Following publication of the SUPRA 2.0 high-end phono preamplifier project in Elektor edition 4/2016, many audio enthusiasts asked us whether this design could be adapted for use with moving-coil (MC) cartridges. A reasonable request, because the noise reduction obtained by wiring several opamps in parallel is particularly important with low input signal levels. Here we show you the necessary changes, which are actually fairly simple.

By **Ton Giesberts** (Elektor Labs)

The SUPRA 2.0 MM/MD phono preamplifier [1] has proven to be especially pop-

ular amongst audio enthusiasts. Many of them have taken the effort build this preamp, which offers truly superior performance. There were even quite a few purchasers for the parts kit, despite its

steep price due to the high-quality components contained in the kit.

However, a lot of vinyl fans were holding back and hoping for the option of connecting a moving-coil (MC) cartridge to the SUPRA 2.0 preamp. MC cartridges are generally better than their MM or MD counterparts, but they require a special preamplifier with additional gain because their output signal level is about 0.2 to 0.5 mV. The original SUPRA was actually designed for MC cartridges, but for the SUPRA 2.0 we decided to initially focus on MM/MD cartridges and leave MC for a bit later. Since then we have modified a SUPRA 2.0 in the lab and increased the gain, and of course we checked it out on the test bench.

Specifications & Measurements

Measurements made with an input signal level of 0.3 mV at 1 kHz (source impedance 5 Ω).

Values measured with a laboratory power supply and supply voltages of ± 14 V.

- Input impedance: 100 Ω
- Output level with 0.3 mV input: 270 mV
- SNR
 - Linear: >73 dB (BW = 22 Hz to 22 kHz)
 - A weighted: >79 dB (BW = 22 Hz to 22 kHz)
- THD+N (1 kHz): <0.02% (only noise, BW = 80 kHz)
- THD at 1 kHz: <0.0007% (from averaged FFT measurement)
- THD at 10 kHz: <0.0002% (from averaged FFT measurement)
- Deviation from RIAA curve: <0.1 dB (50 Hz to 10 kHz)
 - <1 dB (20 Hz)
 - <0.15 dB (20 kHz)

The changes

As previously mentioned, the changes are fairly simple. If you already have a fully assembled SUPRA 2.0, you only need to change nine resistors in each

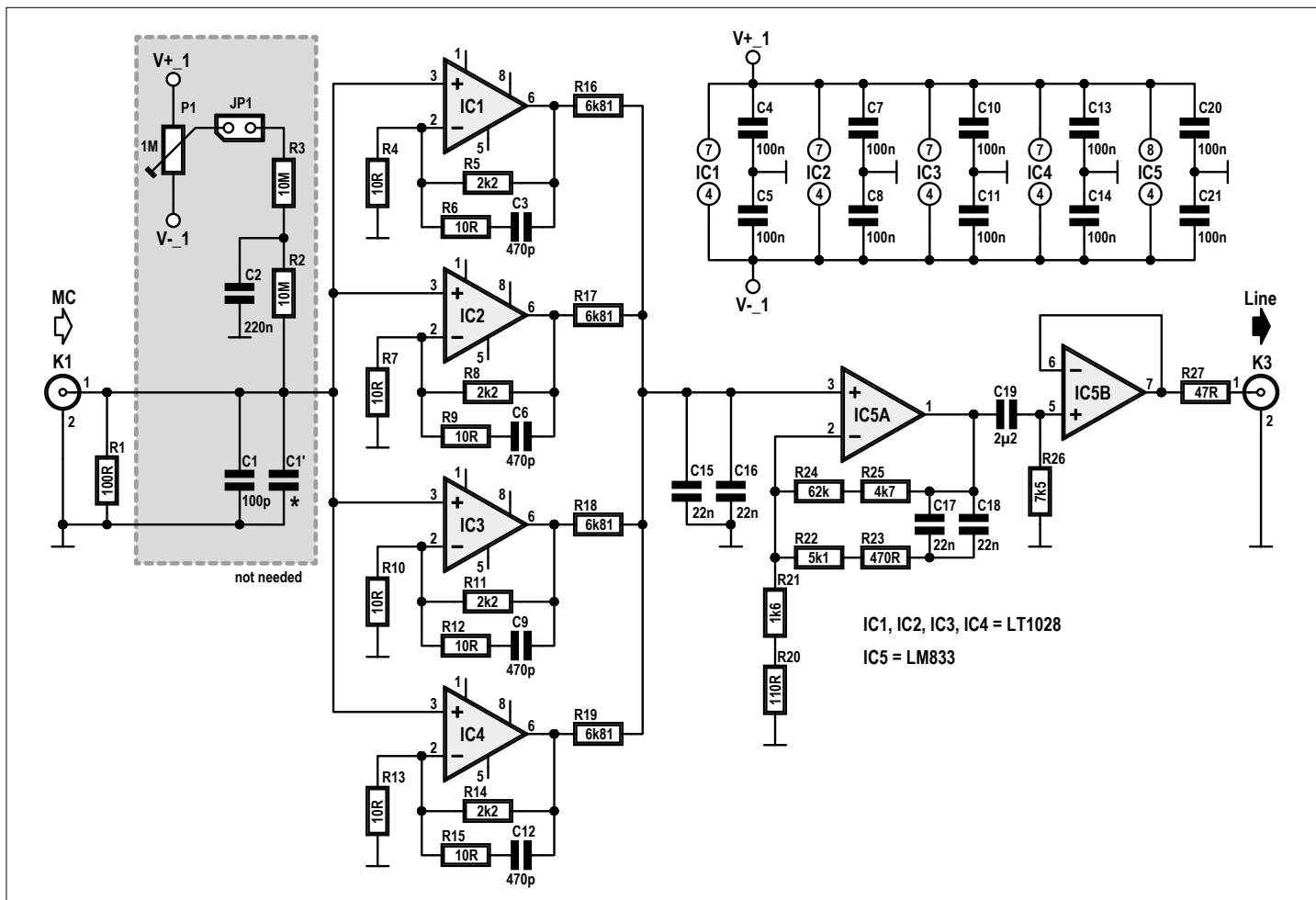


Figure 1. Schematic diagram of one channel of the SUPRA 2.0 with altered resistor values for use with MC cartridges.

channel. The portion of the schematic where changes are necessary is shown in **Figure 1**.

Most MC cartridges are specified for a termination impedance of about 100 Ω. To achieve that, the resistors which determine the input impedance of the preamplifier (R1 and R30) must be replaced by 100-Ω resistors. However, if your cartridge is specified for a different input impedance (for example, 47 Ω), then you should use that value for R1 and R30.

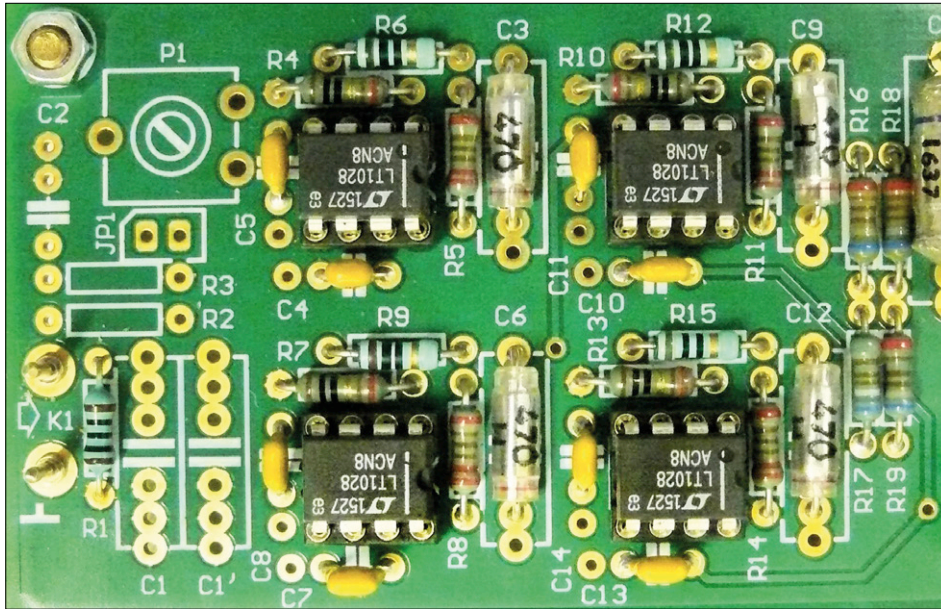
Capacitors C1 and C26 determine the input capacitance of each channel of the preamplifier. With MM and MD cartridges you can influence the high-frequency characteristic with this capacitance, but with MC cartridges it has little effect because their internal resistance is only a few ohms and their inductance is negligible due to the small number of turns in the coil. You can remove C1 and C26 from the board, but there is no

harm in leaving them where they are. Of course, if you are building the SUPRA 2.0 from scratch you can simply omit these capacitors.

Next, all sixteen 47 Ω resistors in the negative feedback networks of IC1–IC4 and IC6–IC9 (R4, R6, R7, R9, R10, R12, R13, R15, R33, R35, R36, R38, R39, R41, R42, R44) must be replaced by 10-Ω resistors. That puts the overall amplification factor of the preamplifier at 900, with the four opamps wired in parallel accounting for the lion's share (amplification factor 221). To determine the necessary amplification factor, we took the average output signal level of 26 MC cartridges, which yielded a signal level of slightly more than 0.3 mV. As the line input of most audio amplifiers has a sensitivity of 100 or 200 mV, we thought it prudent to choose a level that is not much higher, namely 250 mV, to avoid making the amplification factor greater than necessary.

Another reason for reducing the nominal output signal level compared to the MM/MD version of the SUPRA 2.0 (468 mV with a 2.5 mV input signal) is to keep the bandwidth of the four parallel opamps large enough despite the fairly high amplification factor of the MC version. The open-loop bandwidth of the LT1028 opamps is 70 MHz. With an amplification factor of 221 the bandwidth is a bit more than 300 kHz. The 470 pF capacitor in the feedback network limits the bandwidth of the input stage to 150 kHz. The resulting effective bandwidth is approximately 115 kHz.

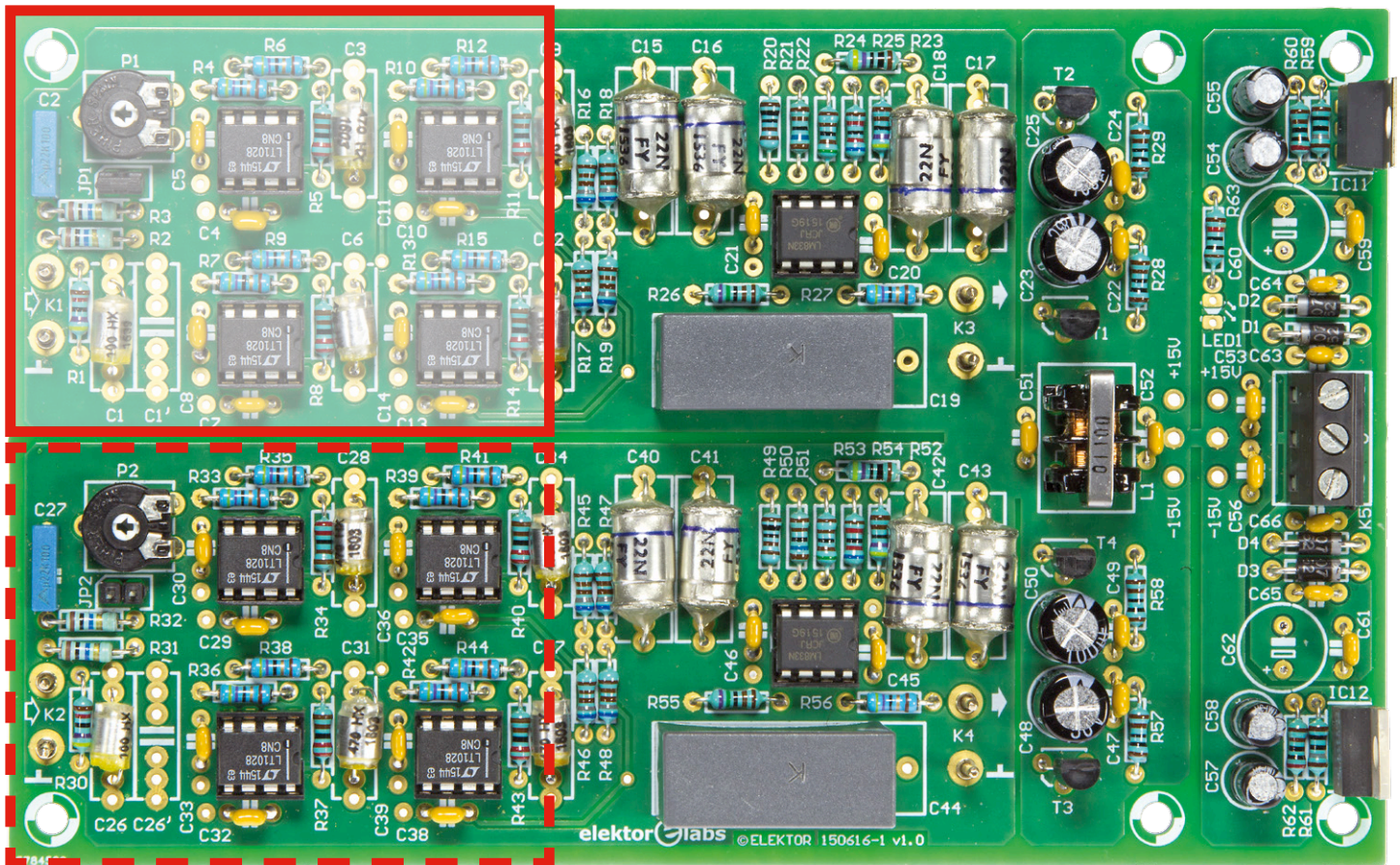
If we increased the gain of the input stage even more, there would be a noticeable effect on the signal amplitude at 20 kHz. If more gain is needed, it would be better to increase the gain of the second stage. However, adjusting the gain of that stage is not straightforward because it incorporates part of the RIAA correction. It



Summary of changes w.r.t. original SUPRA 2.0

- R1,R30 = 100 Ω / 1% (depending on actual MC cartridge)
- R4,R6,R7,R9,R10,R12,R13,R15, R33,R35,R36,R38,R39,R41,R42, R44 = 10 Ω / 1%
- C1,C26 = Not necessary, but can be left in place if already present
- P1/R2/R3/C2/JP1 and P2/R31/R32/C27/JP2 = Not necessary, but can be left in place if already present

Figure 2. The PCB area with the modified components.



would therefore be necessary to change all the resistors in that stage.

There are also several compensation circuits on the SUPRA 2.0 PCB for each channel to allow the input bias currents

of the four opamps to be nulled if necessary. Due to the low input impedance, these networks (P1/R2/R3/C2/JP1 and P2/R31/R32/C27/JP2) are not necessary in the MC version. However, if they are already fitted on the PCB you can simply

leave them in place.

For the MC version of the SUPRA 2.0 we opted for the type A version of the LT1028. It costs a lot more than the standard version (which is already expensive), but it has lower input offset and

bias currents. The highest output offset measured with the type A opamps in our prototype was just 7 mV with an amplification factor of 221.

You should bear in mind that the sum of the four offset voltages will be amplified by a factor of 40 in the second stage. If the offset voltage at the output of IC5A or IC10A is more than a few hundred millivolts with the standard LT1028 opamps, you can consider replacing them with type A versions. Although that is fairly costly (you need eight in total), you should remember that a high offset voltage at the output of IC5A or IC10A will cause more current-induced noise in the feedback networks. In our prototype the offset voltage at the output of IC5A or

IC10A was approximately 100 mV. We hope you enjoy listening to your vinyl collection with the SUPRA 2.0 MC pre-amplifier feeding the signal from your top-end record player and MC cartridge to the rest of your system.

(160263-1)

Web Link

[1] www.elektormagazine.com/150616

Several characteristic curves are shown below to back up the specifications.

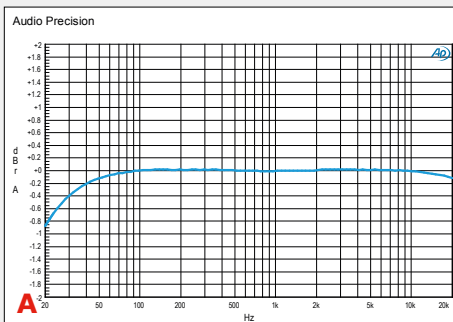


Chart A shows the output amplitude with the input driven by a sinusoidal signal corrected according to the RIAA standard. At 20 Hz the deviation is a bit less than -0.9 dB. At 20 kHz the deviation is -0.1 dB. A slight rise in amplitude can be seen at 138 Hz and 3.3 kHz, but it is negligibly small (0.022 dB). That is a remarkably good result when you consider that all of the components responsible for the RIAA correction have a tolerance of 1%.

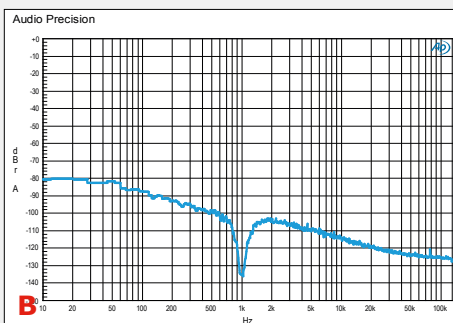


Chart B shows an FFT plot with a 1-kHz input signal. The displayed spectrum runs from 10 Hz to 130 kHz, with the fundamental frequency at 1 kHz suppressed. To give a clearer impression, the curve is the result of averaging sixteen measurements. No harmonics are visible on this plot, from which it can be concluded that the total harmonic distortion is less than 0.0007%.

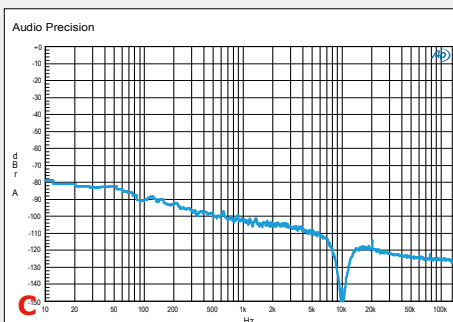
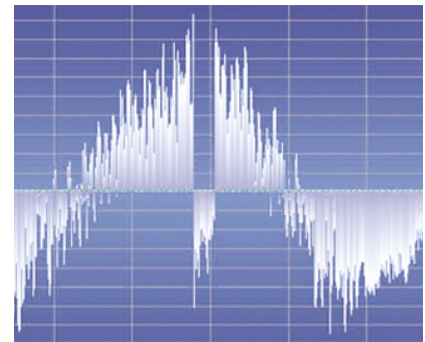


Chart C shows an FFT measurement similar to that of chart B, but with an input frequency of 10 kHz. Here again the THD is very low — only the second harmonic is visible at a level of -114 dB, corresponding to 0.0002% THD.

Noise considerations



The thermal noise of the four 6.81-k Ω summing resistors at the LT1028 outputs has virtually no effect on the signal to noise ratio of the opamps. The noise in each feedback network (2.2 k Ω in parallel with 10 Ω) is 0.41 nV/ $\sqrt{\text{Hz}}$. The input noise of each LT1028 is 1 nV/ $\sqrt{\text{Hz}}$. If the MC cartridge has an internal resistance of 5 Ω , its noise contribution is 1.04 nV/ $\sqrt{\text{Hz}}$. From all that it can be concluded that the total input noise of one opamp is 1.12 nV/ $\sqrt{\text{Hz}}$, or 1.08 nV/ $\sqrt{\text{Hz}}$ with the input shorted.

At the opamp output the noise will be 221 times greater, which is 247 nV/ $\sqrt{\text{Hz}}$. If we add the noise of the 6.81 k Ω resistor, the output noise is 247.2 nV/ $\sqrt{\text{Hz}}$. The total noise at the summing junction is half this value, which makes it 123.6 nV/ $\sqrt{\text{Hz}}$. This figure does not take the current noise into account.

The noise level of the MC input stage is 3.7 times that of the MD input stage (with the input shorted). However, the amplification factor is 4.62 times greater in the MC version. If we recalculate the measured noise of the MD stage with the input shorted (88 dB) for a 270-mV output signal level, the signal to noise ratio is 4.84 dB lower at 83.15 dB. The calculated difference in noise level between the MC and MD versions is 11.4 dB with the input shorted.

From these calculations we can predict that the signal to noise ratio of the SUPRA 2.0 MC version will be about 72 dB. That is very close to the actual measured value (see the specifications).

Connect Objects with Genuino 101

Make any device talk to your mobile

By Clemens Valens (Elektor Labs)

The Internet of Things (IoT) and Connected Objects have been on everybody's lips for a couple of years now; industry analysts predict billions of connected devices in the near future and the market will be worth tens of billions of dollars. Impressive numbers, big money, wouldn't it be nice to get a share of that? It would, so here's how to jump the bandwagon: connect an object.

When we mention connected devices and IoT, we usually mean wireless connected devices sending and receiving data from the cloud. Wireless in this case can be anything really, but Wi-Fi and Bluetooth are today's most commonly used technologies. Bluetooth 4.0, Low Energy or BLE, is very well suited for connecting (ultra) low-power devices like sensors and wearables to a more powerful device like a smartphone or a tablet which in turn functions as an access point to the cloud.

Intel's Genuino 101 board (called Arduino 101 in the USA) (**Figure 1**), an Arduino compatible board fully supported by the popular and free Arduino development tools, is equipped with BLE, making it a great platform for creating your own connected objects. Powered by an Intel Curie module — integrating a 32-bit Quark microcontroller, 384 KB flash memory, 80 KB SRAM, BLE, a six-axis gyroscope-accelerometer combo, and battery charging circuitry — the board features Arduino Shield compatible extension connectors giving access to up to twenty digital input/outputs and six analog inputs.

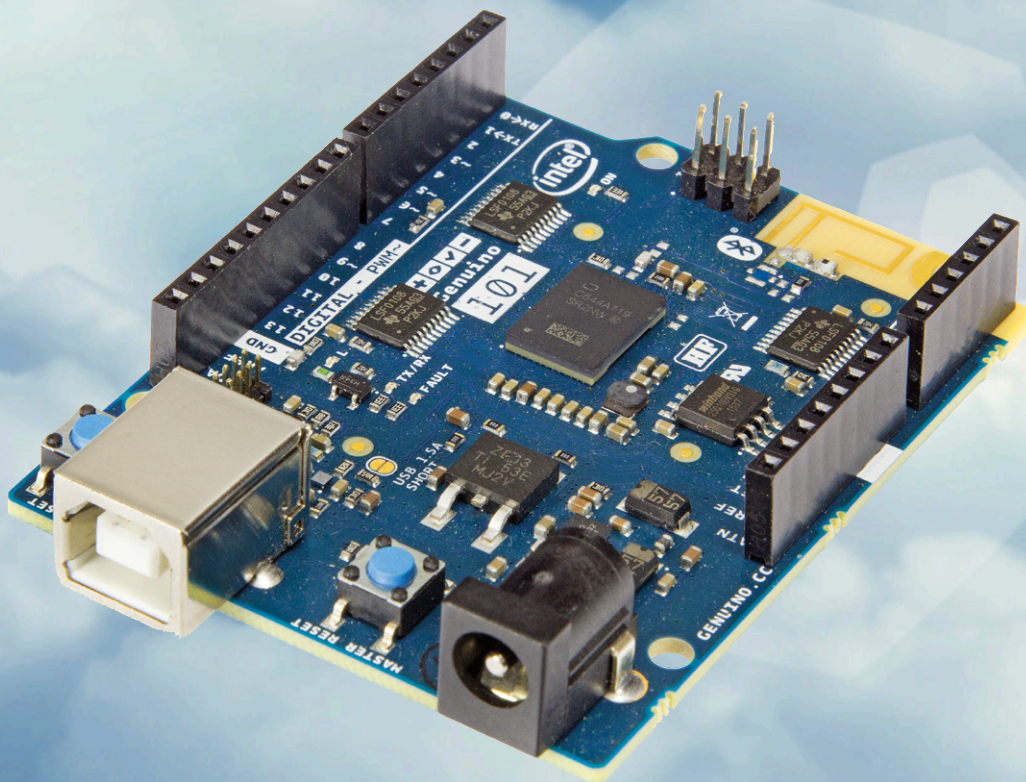
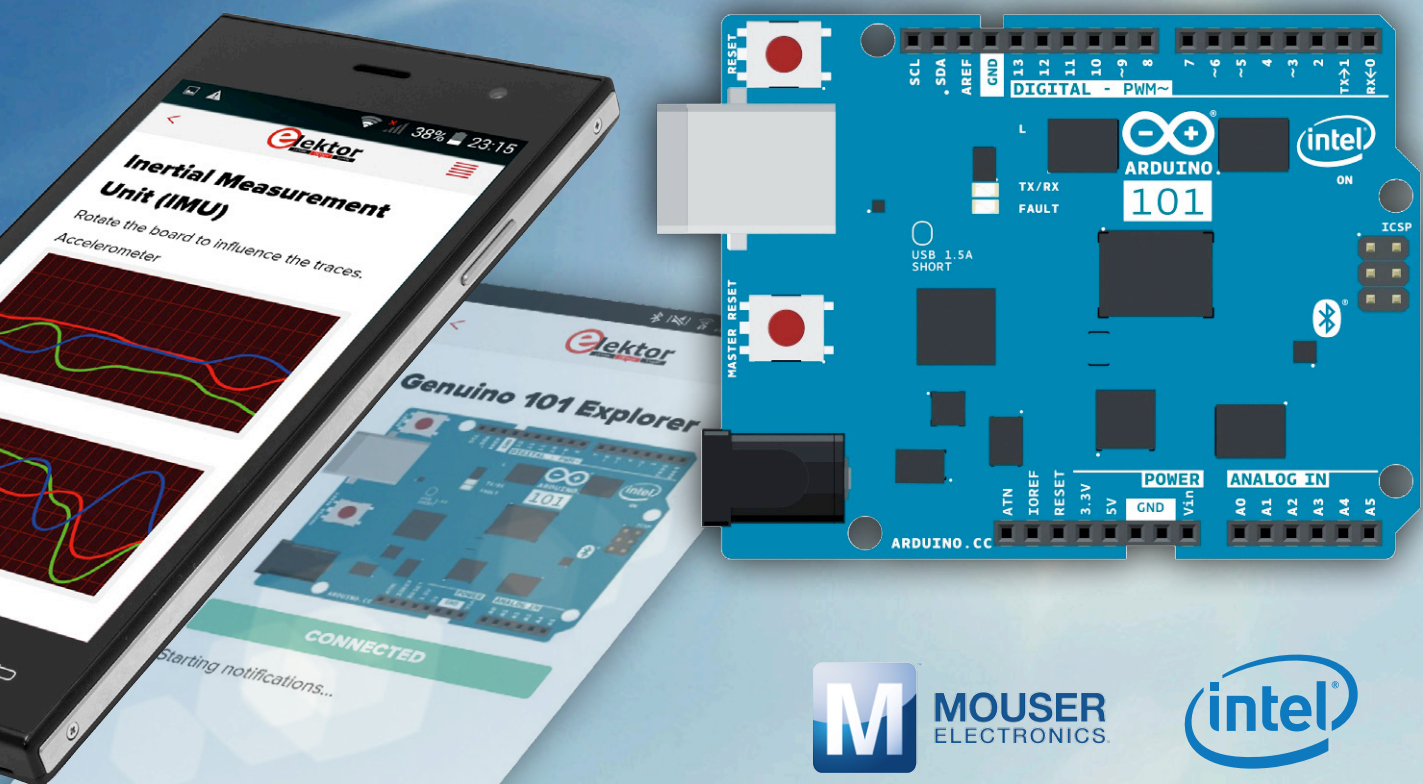


Figure 1. The Intel Arduino/Genuino 101 board has a 6-axis inertial measurement unit and Bluetooth Low Energy capabilities.



This article sets out to teach connecting the '101' to a mobile, BLE-capable device running Android 4.4 (KitKat) or higher. Officially BLE support is available since Android 4.3, but because its reliability leaves a lot to desire, it's best to stay away from it.

First we will add an extension board (a Shield in Arduinospeak) with a BME280 sensor to capture temperature, air pressure and relative humidity [1]. Furthermore, we are going to capture data from the 101's inertial measurement unit (IMU, the gyro-accelerometer combo, that is) and display it in pleasant graphs on the mobile device [2], and you will be able to read the 101's analog inputs and control its digital outputs from the mobile device.

In order to connect the Genuino 101 to an Android device over BLE an App is required. Since we design the connected object to our own specifications, it is also our job to create this App. Several ways lead to Android Apps, but some are easier than others. Being newbie App develop-

ers, we prefer an easy approach, and a good tool for us kind of people is Evothings. The idea behind this tool is simple: provide a basic App framework to which the developer can add his own code using JavaScript, HTML and CSS, standard, well-known webpage development languages.

Actually, designing an App in Evothings amounts to creating a tiny website and then displaying it in the framework, the Viewer, running on the mobile device. Because in Evothings the App is nothing more than a simple website, it can be changed on the fly without lengthy compilation steps. All you do is write and modify code; as soon as you save it, Evothings will update its viewer. Instant result on your mobile — isn't that neat? No risk of crashing your mobile either, the viewer catches your mistakes. Once you are completely satisfied with your app, you can create a 'real' App for it and distribute it through the Google Play Store.

Sounds good, doesn't it? So how do we go about it?

Prerequisites

- Genuino 101;
- BME280 shield;
- PC (Windows, Linux or OSX) with at least one free USB port, a plain text editor and, optionally, an image editor (I used Windows 7 Family, SP1, 64-bit);
- Mobile device, BLE capable, running Android 4.4 or higher (I used a Samsung Galaxy J5 running Android 5.1.1);
- Internet access for PC and mobile device.

Assemble the hardware

Putting together the hardware is quite simple as it consists of plugging the BME280 shield on the 101 (**Figure 2**). If you want to get fancy, you can add your own extension board with some resistors, LEDs and trimmers to play with the digital and analog I/O functionality, but this is optional.

If you do, take care to avoid pin conflicts with the BME280 shield. Digital pins 0 to 6 and analog inputs 0 to 3 are free.

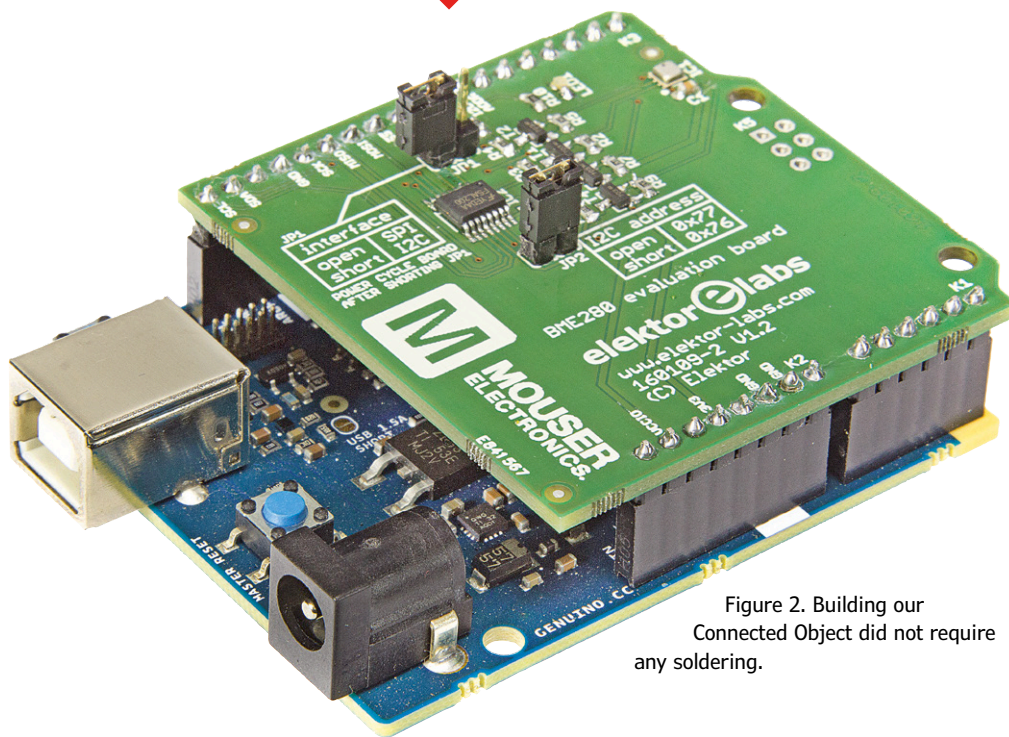


Figure 2. Building our Connected Object did not require any soldering.

sketch in the 101, wait a few seconds and open the Arduino IDE Serial Monitor (Tools → Serial Monitor). After a while you should see values appear for temperature, air pressure and relative humidity. Make sure the shield is working correctly before continuing.

Download the Genuino 101 Explorer sketch from [1] and upload it to the 101. After a few seconds the Serial Monitor will show some data indicating that the 101 is waiting for a Bluetooth connection (**Figure 3**).

Set up Evthings

Evthings Studio is also called Evthings Workbench — just launch the executable whatever its name is. Open the New tab and enter “Genuino-101-Explorer” as App folder name. Before clicking Create, make a note of the Parent folder where the App will be saved. Close Evthings Workbench. With a file manager navigate to the App folder you just created and delete everything inside. From [1] download the Genuino 101 Explorer Evthings project and unpack it in this folder. You should now have some folders and files including index.htm and app.js. Those two are the important ones. Relaunch Evthings Workbench and open the MyApps tab. You should see an entry labeled: Arduino/Genuino 101 Explorer.

On your mobile launch the Evthings Viewer App (**Figure 4**). It will open and request a connect key. This key is generated by the Evthings Workbench: click the Connect tab and then the Get key. Enter the key in the viewer and tap Connect. For this to work, both the PC and the mobile must be connected to the Internet. When the connection is successful the viewer will tell you what to do next. What it means to say is that you have to open the MyApps tab in the workbench and click on the Run button of the Arduino/Genuino 101 Explorer app. Do this and observe the Loading icon appear in the viewer, followed by the App opening a few seconds later.

If the Internet connection on both your mobile and PC is good and stable, the connection between the viewer and the workbench will persist. However, if you are seeing bizarre loading errors you may try to reconnect the two with a new key.

Try it out

If you had disconnected the Genuino 101 from the PC, now is the time to recon-

Install the software

Softwarewise there is rather a lot to install and set up before you can create and publish your App in the Google Play store, but we will start unpretentiously:

- PC: Download and install Arduino IDE (www.arduino.cc, I used version 1.6.10)
- PC: Download and install Evthings Studio (https://evthings.com/, I used version 2.0.0)
- Mobile: Download and install Evthings Viewer (Google Play Store, I used App version 1.4.1)

Genuino 101

Do not connect the board to the PC yet, you must install some drivers first. Launch the Arduino IDE and open the Boards Manager (Tools → Boards → Boards Manager...). In the top-left corner choose Arduino Certified, this will give you a shorter list in which you can easily find the Intel Curie Boards. Click on it, and then click on the Install button. Depending on your Internet connection the installation may take a while. When done, close the Manager, then go back to the Tools → Boards menu and scroll down through the list until you find the Arduino/Genuino 101. Select it.

Connect the board to the PC. A serial port should become available; select it in the Arduino IDE (Tools → Port).

If you run into trouble with the installation of the 101, please consult the online installation guide at <https://www.arduino.cc/en/Guide/Arduino101>.

Download the BME280 sketch from [1]. The QR code on the back of the BME280 shield contains the URL (<http://bit.ly/2aN-NDq7>) that will get you there. Load the

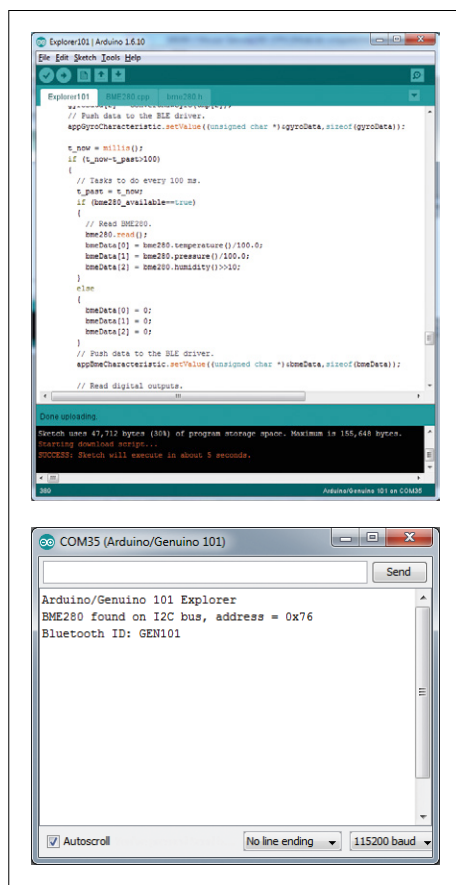


Figure 3. Sketch uploaded successfully and board waiting for a Bluetooth connection!

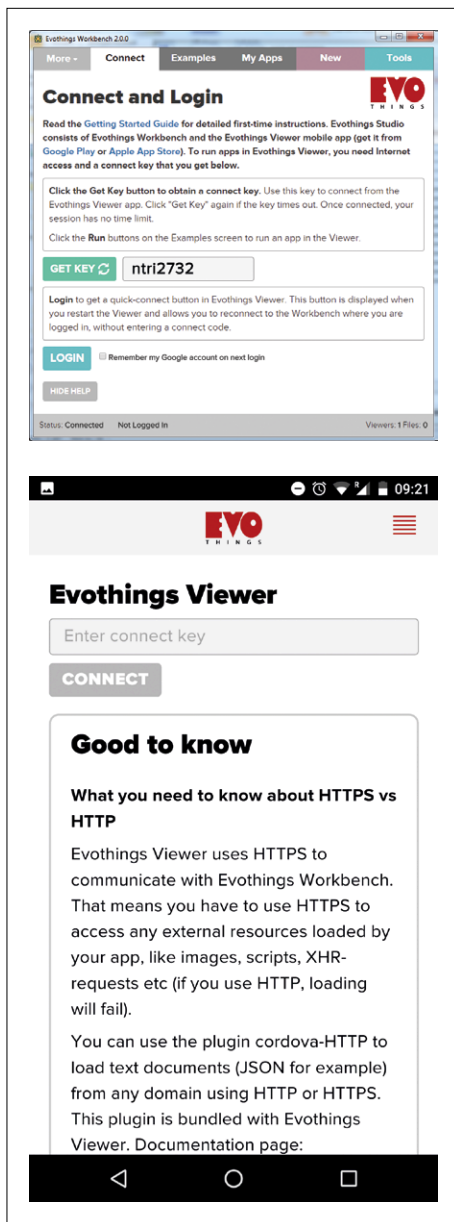


Figure 4. Evthings Workbench on the PC and its Viewer on the mobile device.

nect it (or power it from a supply). If it was connected already, press the Reset button (not the Master Reset button). Give it ten seconds or so to get going, use the Serial Monitor to be sure that it is waiting for a connection, then tap the red Connect button in the Evthings viewer on your mobile.

If Bluetooth was not yet activated on your mobile, it will ask you for authorization to activate it now. Accept this and observe the status messages that appear below the connect button. If all goes well, the button turns green after a few seconds. Tap the menu button in the upper-right corner and select a data view (**Figure 5**).

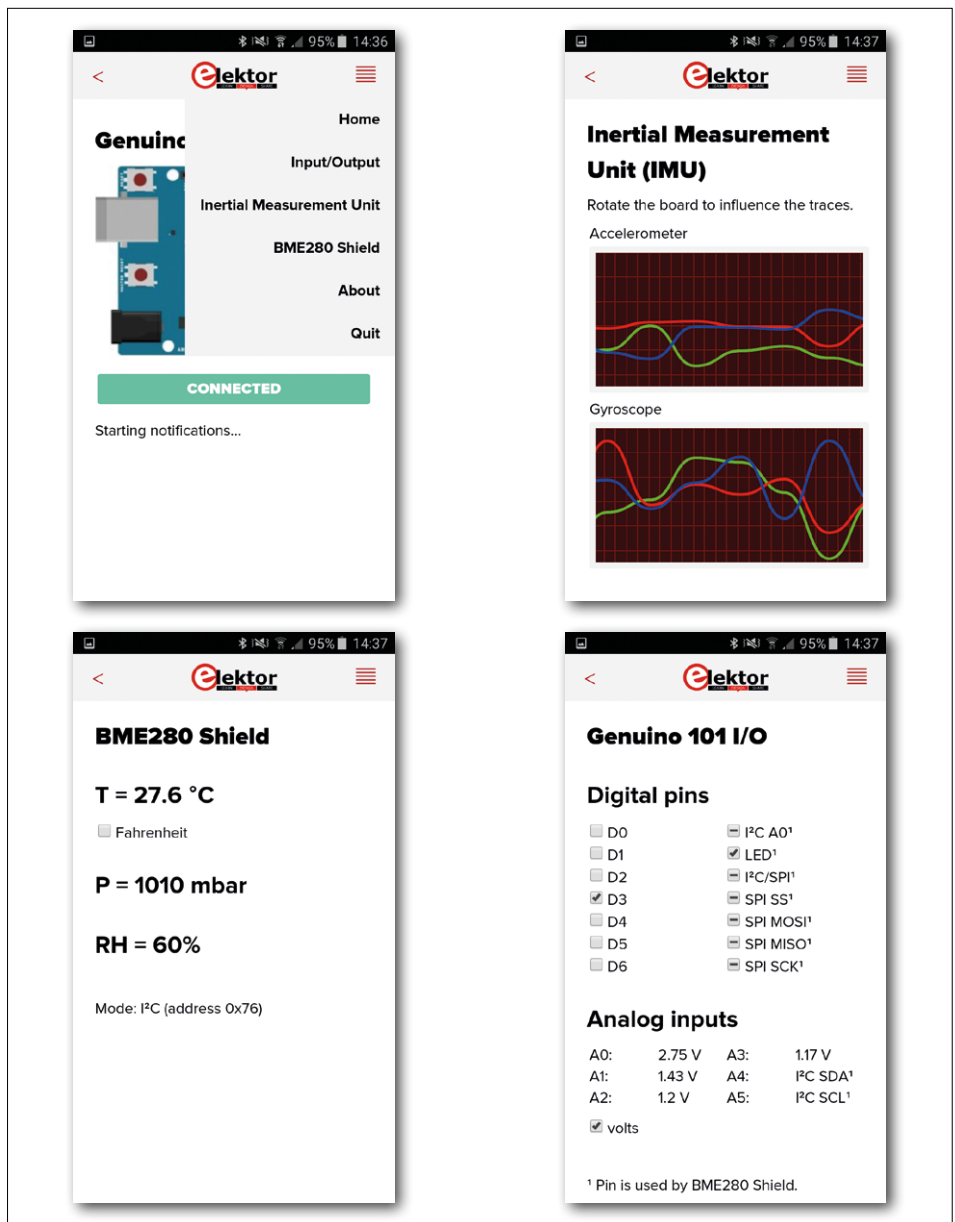


Figure 5. A few screenshots from the App showing the different data views.

Regarding the Sketch

Now that you have a working environment it is time to investigate the different components that make up the system. Let's start with the Genuino 101 sketch. This sketch is based on the CurieBLE and CurieIMU examples (Arduino IDE → File → Examples) to which I have added the BME280 driver. In this article we will concentrate on the BLE side of things, we take the hardware interfacing for granted. The basic idea is that BLE uses services that are identified by unique IDs. A number of such services are defined by the Bluetooth SIG (www.bluetooth.com), but since our combined IMU weather service is a not existing one, we must create a

new unique ID for it. You do this by generating a Universal Unique Identifier or UUID (with an online UUID generator, for instance). Then, for each kind of information that must be received or sent, a characteristic must be added to the service as an attribute, each characteristic with its own unique ID. Characteristics can be read-only, write-only or read-write and they may generate notifications.

In theory a characteristic can convey up to 512 bytes, but the Genuino 101 BLE library limits this to 20. Our application produces three 4-byte floating point values for the accelerometer (i.e. 12 bytes in total), the same for the gyro and for the BME280; the analogue data fits in

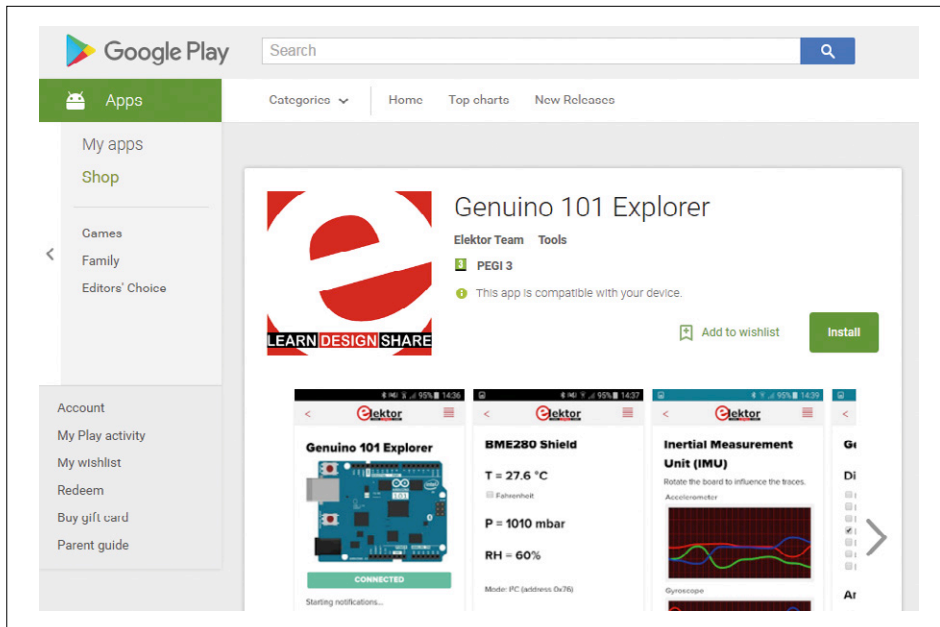


Figure 6. Our own App in the Google Play Store — hey we're in (IoT) business!

six 2-byte values (again 12 bytes) and the digital I/O data fits in two bytes. This totals to 50 bytes which could fit in three characteristics, but in order to keep things simple by avoiding complicated data multiplexing schemes we simply define a characteristic for each data type.

Once the characteristics defined and initialized, and added as attributes to the service, the BLE peripheral can be started. As long as there are no connections nothing much will happen, but once connected, we can start putting data in outbound characteristics and read data from inbound characteristics. That is all there is to it really, the library handles the difficult stuff for us.

The data from the IMU is streamed as fast as possible to the mobile; the other data is sent every 100 milliseconds.

Reading data from a characteristic can be done in two ways, either by polling it using the written() function or event driven by registering a listener. Both ways are illustrated for the digital I/O pins characteristic in the Genuino 101 Explorer sketch.

The service can be given a (local) name, "GEN101" in our case, but care must be taken to keep it short. The reason is that it is put in the space that is left in the advertising buffer and that is not a lot. If the local name does not fit, the library will make up its own name, breaking the automatic detection capability of the App in the process.

About the App

On the App side things are quite similar. The App's bread-and-butter code lives in the JavaScript file app.js. The user interface (UI) is defined by the HTML file index.htm and the style sheet ui\css\evothings-app.css determines just how smart the HTML elements will look. Like Genuino 101, Evothings features a nice BLE library that takes care of the Bluetooth magic, making it easy to use in our App. The unique IDs created for the Genuino 101 sketch must be known by the App in order to access the service and its characteristics. Since the characteristics come in asynchronously the App activates reception notifications for them to process the data as it arrives.

Except for the analog and digital I/O that is displayed immediately, the rest of the data is stored first before being shown to the user. Interval timers are set up for the graphs and the BME280 data. The graphs are updated rather slowly, so rapid board shaking and noise are suppressed, giving them a really smooth appearance. This smoothness is also due to the library used to create the graphs: Smoothie Charts (<http://smoothiecharts.org/>).

Communication between the UI in HTML and the App in JavaScript is bidirectional. From app.js up to index.htm we make extensive use of the getElementById() function. For this to work the HTML element to modify must have an id. The innerHTML property lets you rewrite the text inside the element (HTML code

is allowed too so you can dynamically rewrite the page), and with the get and setAttribute methods you can query and change the element's attributes, like changing its color or size.

The HTML code in return can call functions in app.js when the user clicks or taps a button or menu item, or toggles a checkbox by declaring a JavaScript function for the element's onclick attribute, for instance onclick="app.onStartButton()". The JavaScript and HTML code can naturally connect to the Internet — this is shown on the App's 'About' page, where some basic links allow the user to open web pages. With a bit more code you can connect to online data storage services, Twitter and what not and publish your data on a www scale.

Turn it into an App

When you think that your App is ready for transforming into a real Android App you must compile it. Note that it is not only possible to compile the App for Android, but for IOS too if you prefer (but not on Windows). Compiling is easy enough if you have the Apache Cordova toolchain at your disposition (<https://cordova.apache.org/>), but installing it requires a huge amount of software:

- Node.js (I used v4.4.7);
- NPM (I used v2.15.8);
- Git (I used version 2.6.3.windows.1);
- Cordova (I used v6.3.1);
- Java JDK (I used jdk1.8.0_101_x64);
- Android SDK (I used v25.1.7 + API 24 + API 23 + API 19).

Luckily, detailed instructions on how to do this can be found online at <https://evothings.com/doc/build/build-overview.html#Install> so I won't waste any space on it here.

Supposing you have Cordova working, here's the recipe for building an App from your Evothings project.

First you need to create a Cordova project:

Create a folder to work in and open a command line interface (CLI) here.

In this folder enter the command "`cordova create [my_project] [com.elektor.labs.my_app] [my_app_name]`" where you replace everything inside square brackets by names of your own devising. Note that the dotted name is supposed to be a reverse URL (you can make one up).

In the CLI, cd into the project folder you just created: “cd [my_project]” Delete everything inside the subfolder ‘www’, but do not delete this subfolder. Copy your Evothings project into the folder ‘www’, i.e. all the files and subfolders that accompany app.js and index.htm. In the CLI, enter the command “cordova plugin add cordova-plugin-ble” to add the BLE library to your project. Repeat this for any other plugin that you may require.

In the CLI, enter the command “cordova platform add android”. For IOS use “cordova platform add ios”. You can add more than one platform.

Now you're all set to build the project from the CLI with these commands:

```
cordova build android
cordova build ios (not on Windows)
```

Normally the build should succeed without warnings and errors and the result can be found in: platforms/android/build/outputs/apk. To test it on your mobile device you have to copy the APK file on it and install it through the mobile's file manager. This requires your device to allow Unknown Sources (in the security settings).

To publish your App in the Google Play Store you must recompile it in Release mode and sign it. For this it is necessary to first create a key (one line command):

```
keytool -genkey -v -keystore
[Name].keystore -alias [Alias]
-keyalg RSA -keysize 2048 -validity
10000
```

Where [Name] and [Alias] are for you to choose. Fill in the requested information and make sure to write it down

somewhere for future reference. Then run (again, one line command, be careful with the double dashes):

```
cordova build android --release --
--keystore="[Path]\[Name].keystore"
--storePassword=[Password]
--alias=[Alias]
```

Where [Name] and [Alias] are the same as before, [Path] is the path to the new keystore file, and [Password] the one you were asked by keytool to create. A tiny pop-up window will ask you for your password once more, enter it and click OK. The result is an .APK file named “android-release”.

If you haven't one already open a Google Play Store publisher account (one-time \$25 fee), upload the APK file, go through the forms, publish, and you are in business (**Figure 6**)!

Pitfalls

Your Android App may (will?) not work as expected from what you saw in the viewer. The most likely reason is that you did not fix all the problems that were silently handled by the Evothings viewer. To find out what's wrong, use the JavaScript console in the Evothings workbench, on the Tools tab, and run your App in the viewer. Everything that's Unreferenced and Unhandled etc. must be fixed.

Keep in mind that timeouts may be too short for real-life situations. Also keep in mind that things work differently when the mobile is connected to the object. The Bluetooth stack may be blocking something. Maybe you should close the connection first before you can do what you want to do?

What if your App doesn't look as expected? Open the file index.htm in a browser on your computer, it should look

the same as in the viewer (but much bigger). If not, try to find the reason and fix it.

A final word on BLE. Handling a BLE connection properly is a bit tricky as it is almost impossible to completely exit an App. Android keeps it running in the background where it may remain (partly) connected. To get out of this kind of loopholes kill the App in the mobile's application manager and, just to be sure, restart the connected object.

Conclusion

This article showed how to connect a Genuino 101 board to a Bluetooth Low Energy (BLE) capable (mobile) device like a smartphone or tablet by using open source and free tools only. The board was extended with a BME280 shield to capture temperature, air pressure and relative humidity data. After defining a custom BLE service and some custom characteristics any kind of data could be exchanged between the two connected devices. This way typical IoT data like accelerometer, gyro and weather information was transferred to the mobile device. A user interface (UI) running on the mobile device as a native App allowed displaying the data and user interaction, and provided the Internet capabilities needed for transferring data to and from the cloud. Finally, we published the App in the Google Play Store for our future users to enjoy.

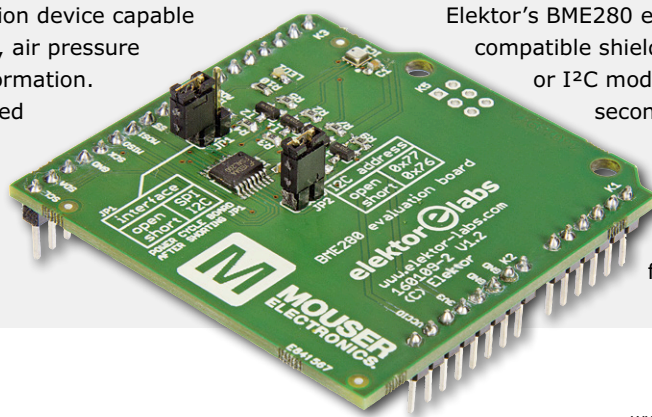
(160109-2)

Web Links

- [1] www.elektormagazine.com/labs/bme280-evaluation-board-160109-2
- [2] www.hackster.io/gov/imu-to-you-ae53e1

BME280 Evaluation Board

The Bosch Sensortec BME280 sensor is a high-performance, high-precision device capable of capturing temperature, air pressure and relative humidity information. Although primarily targeted at mobile devices for face/hand proximity detection and indoor navigation, it also makes a very capable weather station. The device can



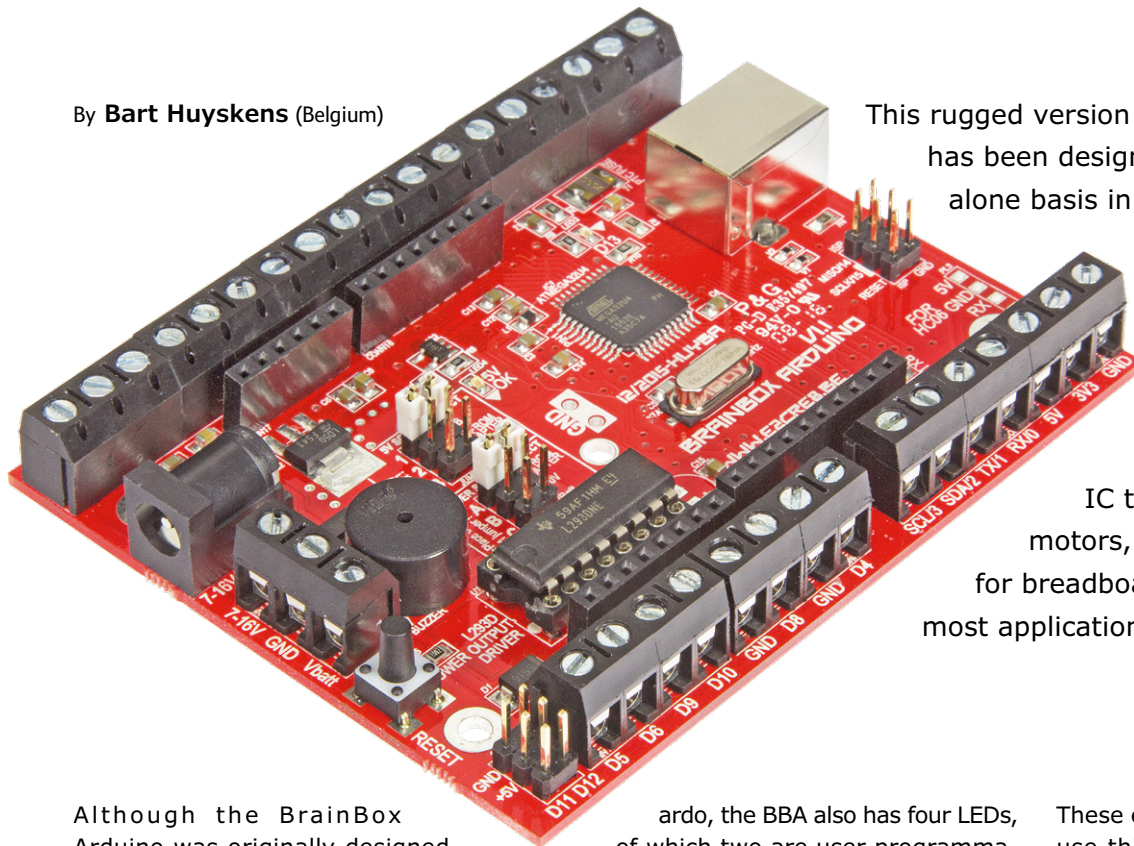
operate on either an SPI bus or an I²C bus.

Elektor's BME280 evaluation board (# 160109-2, an Arduino compatible shield) permits the use of the sensor in SPI or I²C mode, simply by pulling or fitting a jumper. A second jumper permits selecting between two I²C addresses. The selected mode and I²C address can be detected by the host, as well as set by suitable programming. A user LED is available for visual feedback.

BrainBox Arduino

A 'tough' Arduino with screw terminals

By **Bart Huyskens** (Belgium)



This rugged version of the Arduino Leonardo has been designed to be used on a stand-alone basis in various projects and in the educational sector. Thanks to its sturdy screw terminals, a number of different power supply options, an on-board buzzer and a motor-driver IC that connects directly to motors, there is usually no need for breadboards, extra ICs or shields in most applications.

Although the BrainBox Arduino was originally designed for use in the educational sector, the board is so versatile that it lends itself for use in many other electronics projects. Anybody can program this BrainBox Arduino (shortened to BBA for the rest of this article) in next to no time, using their preferred programming language, since the example programs have all been developed for five different development environments.

The BBA can be obtained fully assembled, complete with a programmed bootloader, from the Elektor Store, which means you don't need any soldering experience to get started with this board. Furthermore, there is also an educational kit available that contains all the hardware, software and courseware to build the robot buggy shown in the inset, and to control it with a dedicated app.

Extras

The BBA is built around a powerful Leonardo processor, the ATmega32U4, which runs at 16 MHz. This processor can be programmed directly via the USB port. Just as with the standard Arduino Leon-

ardo, the BBA also has four LEDs, of which two are user programmable. The differences are mainly due to the extra features of the BBA, compared to the standard Arduino boards:

A buzzer with a wide frequency response. This is ideal for trying out all those ring tones you've created.

A double H-bridge with four power outputs, which can source up to 600 mA per pin. You can use this to directly drive 4 DC motors in half-bridge mode, or two DC motors in full-bridge mode. These power outputs can also be used to drive power LEDs, to generate heat using power resistors or to drive stepper motors, to give a few examples. The user can select one of the following power sources for the four power outputs, using a set of jumpers: 5 V, mains adapter, battery, or an external supply. The voltage to the power outputs must be between 4.5 V and 36 V. The power source for the processor is selected via a jumper from one of the following: USB, mains adapter or battery. Two connectors for servo motors. Extra connectors for an HC06 Bluetooth module, RS232 and I²C.

These extra features make it possible to use this board in a range of projects, without the need for spaghetti circuits on breadboards or extra Arduino shields.

Circuit diagram

The complete circuit diagram for the BBA is shown in **Figure 1**. At first sight this appears rather complex. In view of the educational background of the BrainBox Arduino, it would be better if beginners looked at the connection diagram in **Figure 3**, which clearly shows all the connections on the board with their name, as well as the functions of the main components on the board.

For those of you who would like to know more about the 'real' circuit, we'll take a look at the components shown in the schematic. At the heart of the BBA is an ATmega32U4. This low-power 8-bit RISC-based microcontroller has 32 KB of flash memory, a 2.5 KB SRAM and a 1 KB EEPROM. It has a built-in USB 2.0 full-speed interface, a 12-channel 10-bit A/D converter and a JTAG interface for on-chip debugging. The maximum clock frequency is 16 MHz, which is the speed of

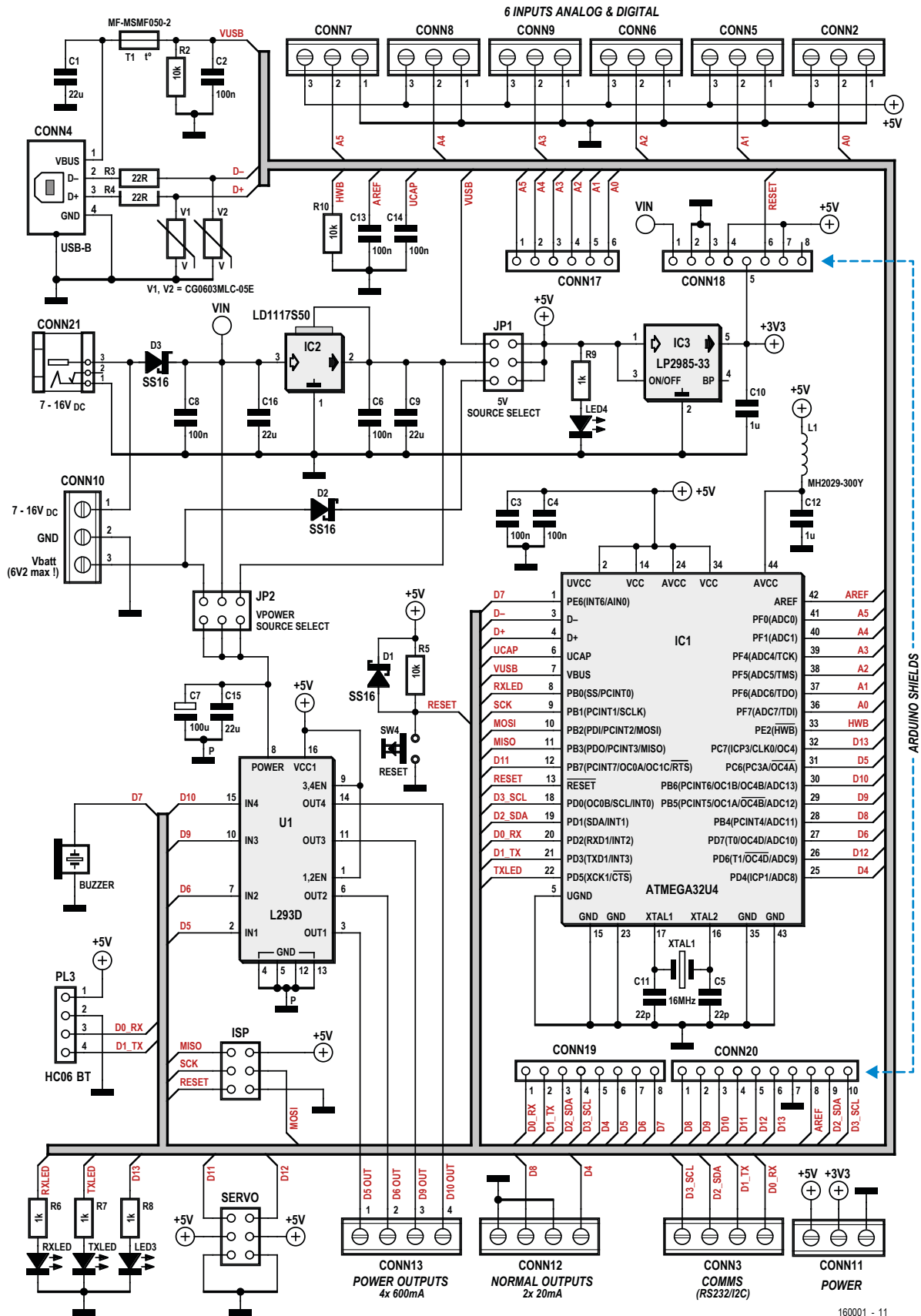


Figure 1. The circuit diagram of the Arduino, with an Atmega32U4 at its heart.



Young people program their robots and design electronic circuits for the Brainbox Arduino

the crystal (XTAL1) used with this board. The USB connector (CONN4) is connected directly to the controller via protection resistors R3 and R4. The supply to this connector is fed via a resettable fuse (T1), which is connected to JUMPER1 for the selection of the power supply source for the board.

The Motor driver (IC U1), an L293D, contains a double H-bridge that drives the four power outputs on CONN13, with a maximum current of 600 mA per pin. The electronics inside the IC is powered by voltage regulator IC2; the power drivers in the IC have a separate power connection, where the source can be selected via JUMPER2.

There are also four LEDs, of which two are used as indicators for the serial communications via CONN3 (RXLED is also programmable), a power indicator LED and a freely programmable LED (LED3).

A buzzer is connected to pin 1 and a reset button is connected to pin 13 of the controller.

There is an extensive power supply section in order to cater for a wide range of supply options. A mains adapter with an output voltage between 7 V and 16 V can be connected directly to CONN21. Alternatively, the 3-way screw terminal (CONN10) can be used to connect an external power supply (also 7–16 V) or a battery pack with a maximum voltage of 6.2 V. The 7–16 V DC voltage is fed via reverse-polarity protection diode D3 to a 5 V voltage regulator (IC2), which provides power to the microcontroller and motor driver IC (U1). This is followed by a 3.3-V voltage regulator (IC3), with its output made available on the shield connectors and on the Power screw terminals (CONN11).

Pin headers JUMPER1 and JUMPER2 are used to select one of the different power supply options. JUMPER1 is used to select the power source for the board: USB connector, mains adapter or battery pack. JUMPER2 is used to select the power source for the motors that are connected to the motor driver IC (U1). Here you have a choice of the regulated 5 V from IC2, the mains adapter, or the battery pack.

The rest consists of various connectors and screw terminals. CONN17 to CONN20 are available for the connection of Arduino shields. There are also six groups of 3-way screw terminals for analog and digital inputs (CONN2, CONN5 to CONN9), each with its own 5-V power supply. These are all grouped on one side of the board. On the other side are the screw terminals for the motors (CONN13), two digital outputs (CONN12), serial and

Software options

Apart from the Arduino IDE, there are several other programming environments that can be used to easily program the BBA. An important feature of this BBA project is that in many instances the example programs have been written for all of the five programming environments listed below. This way you can program the BBA using your preferred programming environment.

Arduino IDE with libraries

The original Arduino IDE is a simple C compiler with a large number of libraries. On the one hand, these libraries and the unique pin numbering make it possible to write fairly complex programs without needing much knowledge about the microcontroller. On the other hand, this means that you hardly learn anything about the microcontroller itself.

Arduino IDE without libraries

What many people don't know is that the Arduino IDE can also be used to program the Arduino using the real register names and pin numbers. Users will first have to familiarize themselves with the microcontroller before they can start writing programs for it. It's also possible to use a combination of both systems, with and without libraries.

Flowcode (7) for AVR

From an educational point of view, Flowcode is without a doubt the best way to get started with embedded programming. The flowcharts make it very easy to see the program structure, and the clever simulator lets you try out the code before you send it

to the hardware. Even though this is a graphical language which is amply provided with libraries that make programming easier, it remains the case that Flowcode is much closer to real 'embedded programming' than the Arduino IDE, for example. Flowcode is therefore the ideal first step on the road to Embedded C.

Atmel Studio 7 with GCC compiler

Atmel Studio is the professional development environment for Atmel controllers, and GCC is probably the most-used free C compiler for Atmel AVR controllers. You can quickly program the BBA from within this environment. If you want to, you can still use the extensive Arduino libraries in this environment, to make the programming of certain functions easier.

Snap 4 Arduino

With S4A it becomes possible to program the BBA with the immensely popular 'Scratch puzzle blocks'. S4A uses a fairly stable connection to the BBA, which requires that the 'Firmata' program is loaded first. From then on, all instructions on the screen are carried out live on the hardware of the connected BBA – a bit like an emulator, really.

The beta version can translate most basic programs into real code, which could be loaded at a later stage using the Arduino IDE, making it possible to run some programs independently. It looks like a promising start for S4A, which is an ideal way to impart some of our electronics enthusiasm to the younger generation.

I²C communications (CONN3), and finally connector CONN11, where all the supply voltages are made available. Then there is a 6-way pin header (SERVO) for driving two servo motors, a 6-way ISP-connector (ISP) for directly programming the micro-controller and a 4-way connector (PL3), which accepts an HC06 Bluetooth module. We haven't shown a parts list, since the BBA is supplied ready-made and pre-programmed because of its educational nature.

Overview

The layout of the Brainbox Arduino has deliberately followed the IPO principle: Input – Processing – Output. At the top are all the connections for sensors, the processing is done by the powerful ATmega32U4 processor, and at the bottom are all the connections for actuators. The sensors and actuators are connected directly via sturdy 5 mm screw terminals. The sensor inputs can all be used as either analog or digital inputs. Each input has its own GND and 5 V connections. There is also a set of typical Arduino connectors for the connection of Arduino shields.

The BBA has a wide range of power supply options, which makes it suitable for use in many applications.

On the right-hand side of the board are the programming options. The BBA is supplied with a pre-programmed Arduino Leonardo bootloader as standard. You can therefore immediately program the BBA via the USB port. If you decide to program the ATmega32U4 processor without a bootloader, you can do this via the 6-way ISP header.

At the bottom-left are two connectors for servo motors and 4 power outputs with PWM capabilities. These are driven by a double H-bridge, which can supply up to 600 mA per pin.

At the bottom-right are the two most popular communication ports, I²C and RS232. These can be used to connect a huge range of sensors, actuators and LCDs to the BBA. The 4-way connector at the bottom-right corner is used to directly connect a popular HC06 Bluetooth module. Example programs for both the BBA and the APPINVENTOR help you on your way to use your own apps on a smartphone to communicate with your BBA. You should give it a try — you'll be amazed how quickly you can get your app up and running.

Connections

The colored pin numbers in the diagram shown here correspond to the names given to the pins in the Arduino IDE and in Snap4Arduino. The gray pin numbers are the real pin names as shown in the datasheet, and are used in AVRStudio and Flowcode.

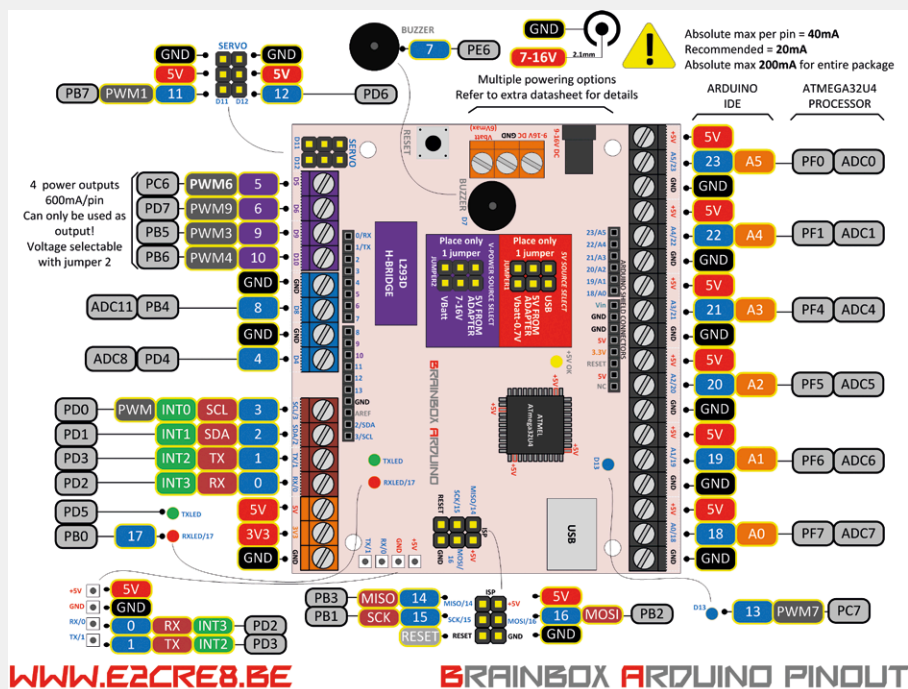


Figure 2. Connection details for the board, with brief descriptions.

(Dutch) teaching material and further information is available from [1]. ◀
(160001)

Web Link

[1] <http://e2cre8.be/> (in Dutch only)

The Author

Bart Huyskens is an electronics and ICT teacher in Belgium. In addition, he organizes workshops and develops hardware, software and teaching material to encourage inquisitive and creative STEM education.

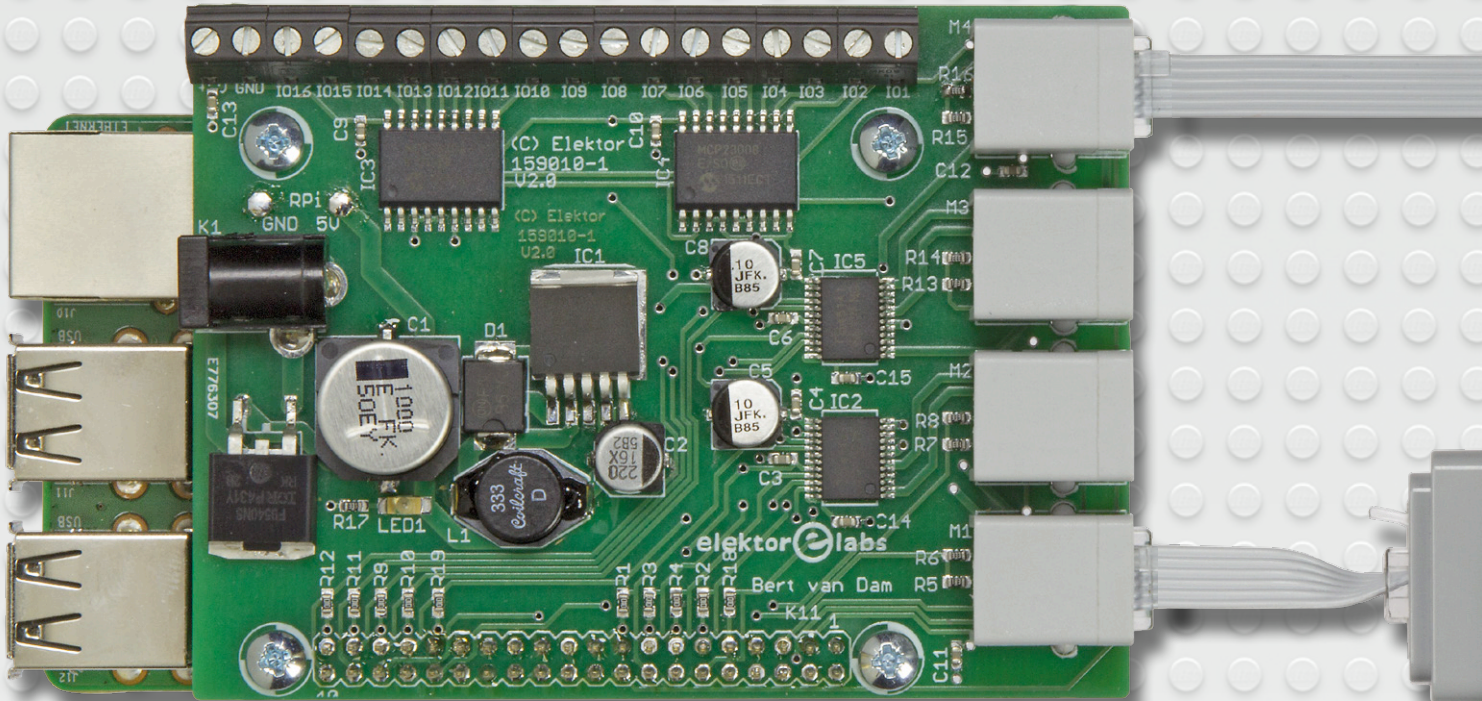
Complete BrainBox Arduino kit

The BrainBox Arduino kit retailed by Elektor comes complete with all the required sensors, actuators, wires, mechanical parts, teaching material and example programs to build the robot buggy shown here, and control it via a dedicated app.

But don't stop there, give your creativity a free reign and use the BBA to create your own projects.



LEGO® Control Board for the Raspberry Pi



This Raspberry Pi HAT puts at your disposal 4 motor-control outputs for LEGO EV3 motors and 16 buffered I/O connections that can be used in combination with a powerful Raspberry Pi. Programming is possible in languages such as C and Python. As an application example we show how this board and a few LEGO parts can be used to build a 'useless box'.

The board described here combines the flexibility of LEGO with regards to

mechanical construction and the flexibility of the Raspberry Pi with regards to software, Internet communications and a wide range of potential sensors.

for the Raspberry Pi, you therefore only need to connect a 9-V (line) power supply to this board (recommended rating 5 A; positive to center pin). From this the motors, the board itself and the Raspberry Pi are powered. Mobile applications can use batteries, because the board uses an efficient buck converter.

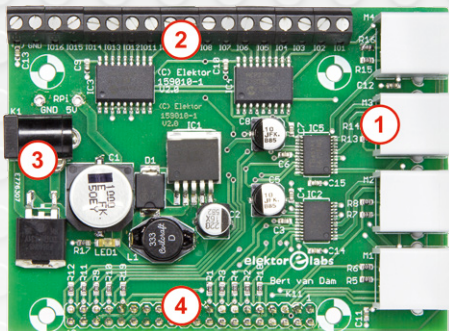


Figure 1. The connections to the LEGO control board. 1) 4 LEGO motor connections; 2) 16 I/O connections with screw terminals; 3) power supply socket, 9 V; 4) feedthrough of the Raspberry Pi I/O pins.

When you attach this board to a Raspberry Pi 2 or 3, you will have at your disposal 4 motor-control outputs for EV3 motors (or compatible types), which you can use to animate your LEGO models. There are also 16 buffered digital inputs and outputs present, to which you can connect sensors, LEDs, etc. You cannot, however, connect LEGO sensors (in that case you would be better served using a standard EV3 controller), but you can connect all sorts of other sensors such as infrared, ultrasonic, tilt and switches. You are limited only by your imagination. The board also contains a power supply

Since the Raspberry Pi header is continued through the LEGO control board, you can also connect SPI and I²C devices and sensors. You can therefore combine all the projects and sensors from the book "Raspberry Pi – Explore the RPi in 45 Electronic Projects" with your LEGO models!

Through the Raspberry Pi itself you can communicate via the Internet, with your

With 4 motor-control outputs and 16 digital I/O channels

By Bert van Dam (Netherlands)



choice of a wired or Wi-Fi connection. If you connect a USB/4G dongle to your Raspberry Pi then you can even commu-

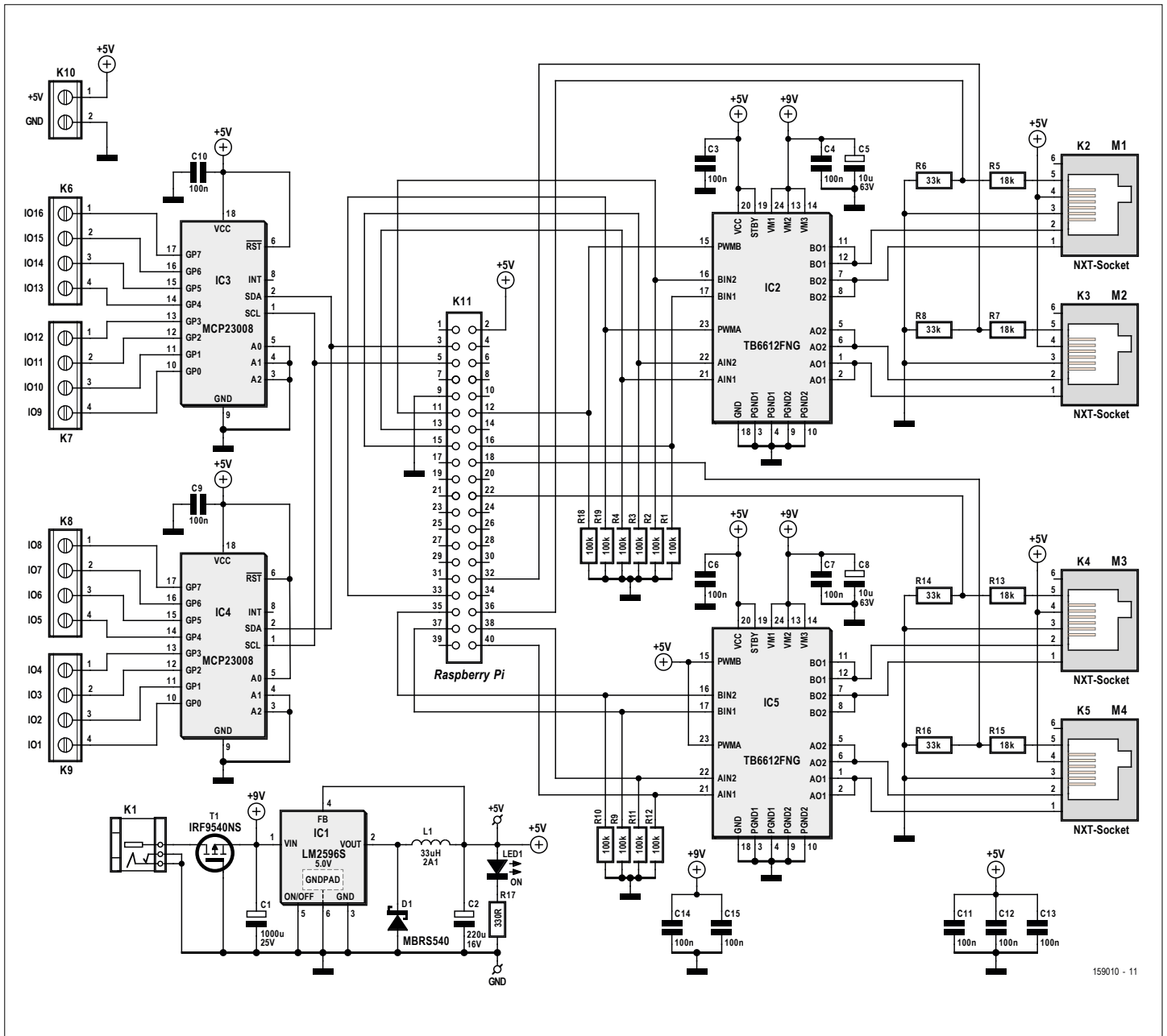
nicate with your model when it is away from your home.

With this board you can therefore make

models where you combine LEGO motors and various types of sensors with the power of the Raspberry Pi!

Table 1. Strong and weak points of the LEGO EV3 and Raspberry Pi.

<p>LEGO — strong</p> <ul style="list-style-type: none"> • The mechanical construction, which allows you to do anything you like. • The motors, although for serious applications you would need servos and hydraulics. 	<p>LEGO — weak</p> <ul style="list-style-type: none"> • The number of connections for sensors. A reasonably serious robot would need 4 and that is only the start. • Video processing and all the other things for which LEGO have not come up with a sensor.
<p>Raspberry Pi — strong</p> <ul style="list-style-type: none"> • Openness of the system; in combination was the easy-to-learn language Python much is possible. For the enthusiast C is an even more powerful option. • Many connectors and via I/O-expanders offers the possibility of 'unlimited' expansion. 	<p>Raspberry Pi — weak</p> <ul style="list-style-type: none"> • Connection pins are mechanically weak and not electrically protected. • Current consumption is on the high side.



159010 - 11

Figure 2. The circuit of the control board comprises 2 motor driver ICs, 2 I/O expanders and a step-down voltage regulator.

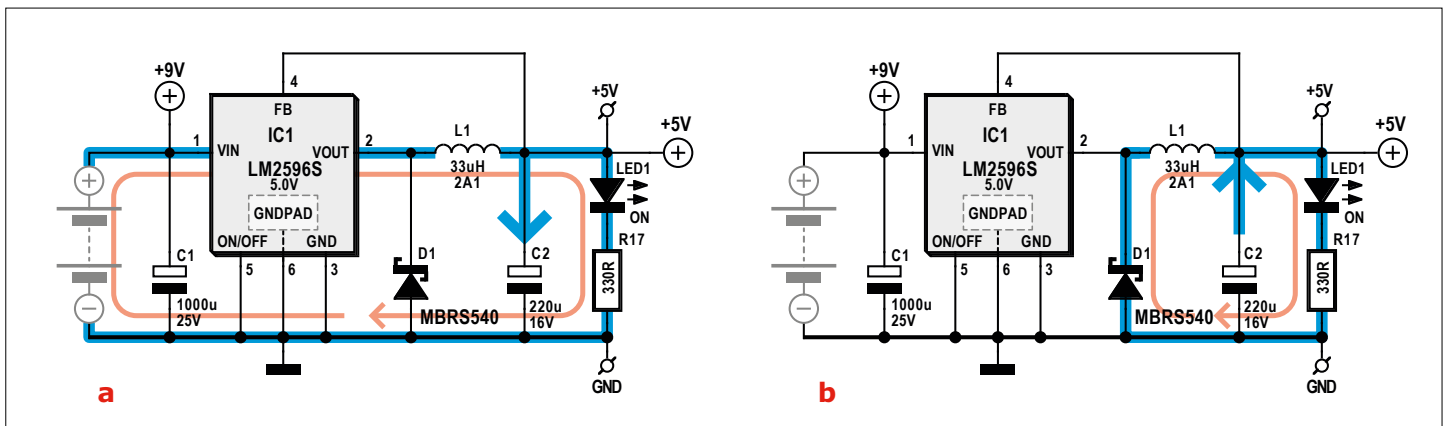


Figure 3. Buck converter: a) cranking up the flywheel b) coasting of the flywheel.

Design considerations

The LEGO control board is certainly not a copy of the EV3 controller, but it does combine the strong points of LEGO with those of the Raspberry Pi. **Table 1** lists the strong and weak points of each. The conclusion is that this board needed to have the following characteristics (and it therefore was given them):

- 4 LEGO EV3 motor control connections (for LEGO 45502, 45503 or compatible), with all 4 of provided with tacho feedback signals and 2 provided with PWM outputs;
- 16 buffered I/O pins (expandable thanks to the I²C connection);
- energy efficient power supply for the Raspberry Pi (using buck-converter);
- free Python 2 software library;
- feed-through of the Raspberry Pi header (you therefore can add further connections yourself, such as protected I/O or analog I/O).

Design result

The complete schematic for the LEGO control board is shown in **Figure 1**. The most relevant parts in this circuit are the two motor-driver ICs type TB6612FNG from Toshiba (which each can drive two motors with a maximum output current of 1.2 A) and two 8-bit I/O-expanders type MCP23008 from Microchip with a high-speed I²C interface and SPI interface. We now take a look at a few interesting details of the circuit.

Power supply

In order to enable everything to be powered a single source, the Raspberry Pi is powered from the control board. The board itself is powered from a 9-V DC power source. This power source can either be batteries or a substantial line power supply (recommend rating 5 A). To go from 9 to 5 V (the voltage required by the Raspberry Pi) an energy-efficient method needs to be used to prevent the batteries from draining too quickly. The choice therefore fell on a buck converter based on the step-down regulator LM2596S-5.0V, a so-called Simple Switcher Power Converter from Texas Instruments. This operates at a frequency of 150 kHz and can supply 3 A. The buck converter functions as follows (see **Figure 3**). The loop formed by the diode (D1), the inductor (L1) and the capacitor (C2) act like a kind of electronic

flywheel. The LM2596 begins by 'turning up' the voltage to 'full'. This gets the flywheel started. This does not happen immediately, because the capacitor first needs to be charged via the inductor. The inductor itself builds up a magnetic field. The diode does not conduct at this time since the current returns via the LM2596 itself.

When the voltage at the output threatens to become too high, the LM2596 switches its output off. The magnetic field in the inductor will now collapse, which causes the voltage across it to reverse and the diode will now conduct. This creates a current that circulates via the load and the diode (the path via the LM2596 is no longer possible because this is switched 'off'). The capacitor also discharges. When the voltage at the output threatens to become too low, the LM2596 switches on again.

The LM2596 therefore only switches on and off. The effect of this is that the efficiency is very high and the LM2596 therefore barely heats up. Because of this continuous on and off switching, the value of C1 is also important and for both C1 and C2 low-ESR types need to be used.

I/O

The MCP23008 I/O extenders IC3 and IC4 are controlled via I²C and ensure that the I/O connections of the Raspberry Pi do not need to be used directly. This design has several advantages:

- The I/O connections operate at 5 V
- The I/O pins have a greater current capability than the Raspberry Pi pins
- Should something nevertheless go wrong, the damage will generally be limited to the I/O extender chip

The maximum current that the MCP23008 can supply is 25 mA, with a maximum per IC of 125 mA. Note that one IC can dissipate a maximum power of no more than 700 mW. This applies to each MCP23008, so if you need a lot of current it would be best to spread the loads across both MCP23008s and not connect everything to a single IC.

Because the Raspberry Pi headers are continued through to the top of the control board you can add additional I²C components yourself, provided that they have different addresses. The LEGO control

board uses addresses 0x20 and 0x21. You can, for example, add further I/O-extendors, D/A converters for generating analog signals, or A/D converters to sense analog signals.

Motors

The two TB6621FNG ICs (IC2 and IC5) each contain two H-bridges to allow the control of four motors in total. The current consumption of the LEGO motors is approximately:

	EV3 small	EV3 large
free running	85 mA	80 mA
slowly forward	500 mA	1000 mA
stalled	780 mA	1800 mA

An EV3 motor can be stalled for only a short amount of time, because the protection built into the motor will switch it off. It is nevertheless always a good idea to prevent stalling the motor, because should the protection not work for some reason the motor could burn out.

The TB6621FNG can typically handle 1.2 A (per channel, so per motor) with a peak of 3 A; this is therefore more than sufficient.

If you intend to make a software library for the motor controllers, then take into account that the PWM pins have to be high in order for the TB6621FNG to operate. When you apply a PWM signal then it is always correct, independent of the direction of rotation.

Circuit board

In **Figure 4** you can see the circuit board that was designed for this LEGO control board. The dimensions and connector positions are obviously selected such that the board will fit trouble-free on a Raspberry Pi model 2 or 3 (using a 40-way I/O connector). Since most of the components are in SMD packages, you can also purchase a completely built version of the board from Elektor. Of course, it is also possible to do everything yourself, the circuit board layout is available from [1]. Note when assembling the board that the 40-way connector K11 is fitted on the bottom side of the circuit board and is soldered from the top side. The four NXT sockets for the motors are quite rare, you will have to search on the Internet for a suitable supplier.

Component List

Resistors

R1,R2,R3,R4,R9,R10,R11,R12,R18,R19 = 100kΩ 1%, 0.1W, case 0603
 R5,R7,R13,R15 = 18kΩ 1%, 0.1W, case 0603
 R6,R8,R14,R16 = 33kΩ 1%, 0.1W, case 0603
 R17 = 330Ω, 0.1W, case 0603

Capacitors

C1 = 1000μF, 25V, aluminum, d 12.5mm, h 13.5mm, ELPP-CP-125-135
 C2 = 220μF, 16V, aluminum, d 6mm, h 7.7mm, ELPP-CP-063-066
 C3,C4,C6,C7,C9,C10,C11,C12,C13,C14,C15 = 100nF 10%, 16V, X7R, case 0603
 C5,C8 = 10μF, 63V, case D

Inductors

L1 = 33μH, 2.1A (e.g. Coilcraft DO3316)

Semiconductors

D1 = MBRS540, 40V 5A, ELPP-DO-214AB
 LED1 = LED, red, case 1206

T1 = IRF9540NSPBF, P-channel MOSFET, D2-PAK
 IC1 = LM2596S-5.0, DC/DC buck converter, TO-263-5
 IC2,IC5 = TB6612, motor driver, 24-SSOP (Digikey # TB6612FNGC8ELCT-ND)
 IC3,IC4 = MCP23008-E/SO, I/O expander, SOIC-18

Miscellaneous

K1 = DC adapter connector, 12V 3A, 1.95mm center pin
 K2,K3,K4,K5 = NXT socket
 K6,K7,K8,K9 = 4-way PCB screw terminal block, 3.81mm pitch
 K10 = 2-way PCB screw terminal block, 3.81mm pitch
 K11 = 40-pin (2x20) GPIO stacking header for RPi, extra tall
 PCB # 159010-1

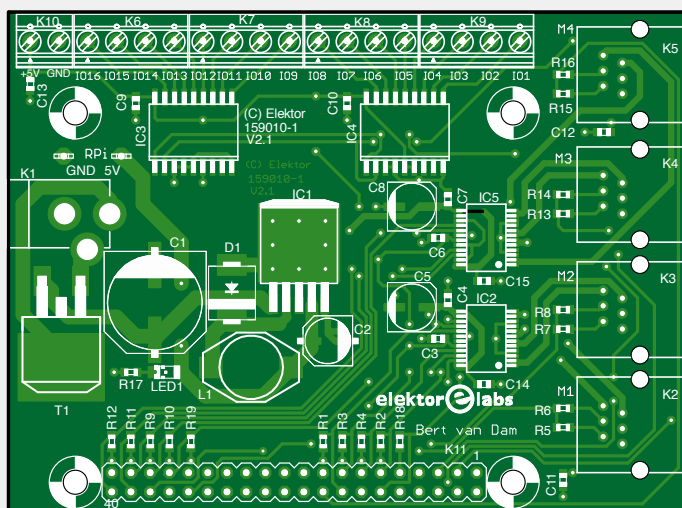


Figure 4. The layout of the circuit board. The 40-way connector K11 is mounted on the bottom of the circuit board.

workings; making the box we leave up to you. You can make this easily using cardboard or wood. With the download for this article you will find a short video of this project. Both the pictures accompanying this article as well as the video show only the mechanical part, without the box. This way you can get a clear view of its operation.

What do you need?

- The SD card from the book 'Raspberry Pi – Explore the RPi in 45 Electronic Projects';
- The free download that goes with this article (contains the library and the source code for the demo project);
- A LEGO motor (type 45502 'large' motor) with connecting cable and a few LEGO parts (you can build the demo project using the LEGO Mindstorms EV3 box).
- A Raspberry Pi model 2 or 3;
- The LEGO control board;
- Two resistors: 1 kΩ and 10 kΩ, 0.25W;
- An (easily operated) small in-line switch (see **Figure 4**).

The hardware

We use a 'large' LEGO motor type 45502 which can be found in, for example, the LEGO Mindstorms EV3 box and the parts that are shown in **Figure 5**. In the download you will find a short video where you can see how you assemble these parts into the motorized finger, which is shown on the right in the figure. You now attach the in-line switch on top of the assembly in such a way that the black beam can push the switch to its off position.

Demo project: The Useless Box

The concept of this 'Useless Box' is a box with a switch on top. When you oper-

ate the switch, a 'finger' appears from the box which turns the switch off again. For this project we only build the inner

This switch is connected with two resistors (see **Figure 6**) to I/O pin 16. This could also be accomplished with only one resis-

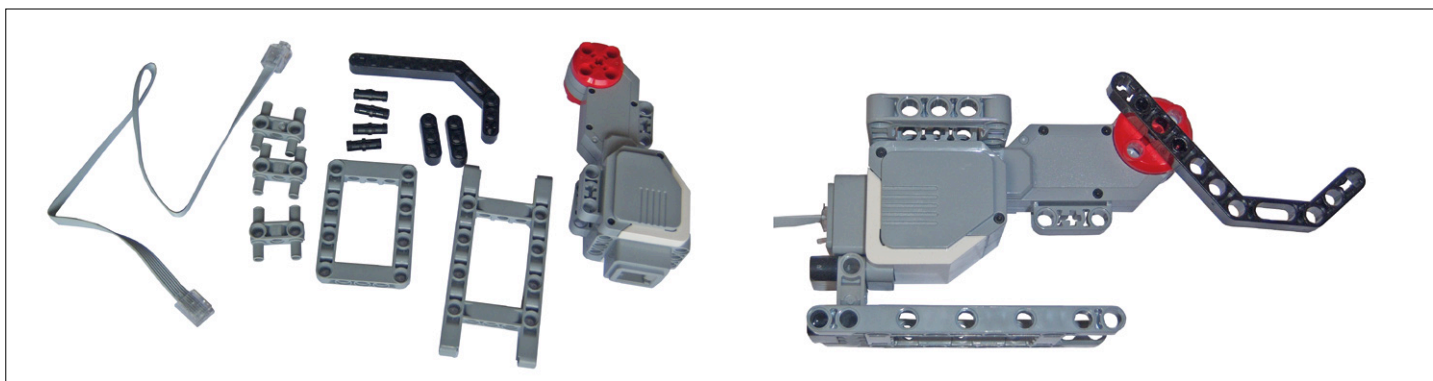


Figure 5. Parts and the assembled model of the motorized finger.

tor (the 10 k Ω one), but the additional resistor serves as protection. If you now inadvertently make the input into an output and set its level low then nothing bad will happen. Without the additional 1-k Ω resistor the I/O port could be damaged.

The software

The software is surprisingly simple. (Note: the library expects Python-2 programs!) You can find more information about the library and the instructions in the manual, which is a free download accompanying this project [1]. The very first thing we do is load the rpirobot library and execute it with the instruction 'execfile'. The motor outputs of the LEGO control board are now automatically configured correctly and the PWM threads are running. We then make pin 16 of the I/O on the control board an input, because this is where the switch is connected to.

```
execfile('rpirobot.lib')
```

```
# Switch on io 16, input
iodir(16,1)
```

```
print "Useless Box Program running"
```

We use a try/except construct (see **Listing 1a**), so that the program waits until the switch is turned on by the user or the user has quit the program using Ctrl-C. Once this happens, the program waits half a second to give the user timer to remove their finger and then activates the LEGO finger with a PWM value of 60 (60% of the maximum speed). We wait in a loop because of the threads that are running. Since nothing happens in the loop this is not actually helpful, so we introduce a short sleep instruction. This instruction takes 0.000001 seconds (1 μ s), and will

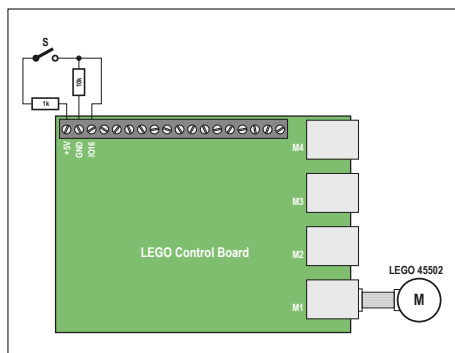


Figure 6. Connection diagram for the switch with the two resistors.

not noticeably slow down the program. As soon as the switch is off again, the program will turn on the brake in the motor. The next step (**Listing 1b**) is to return the LEGO finger at a slower speed (PWM 30, 30% of the maximum speed) to its rest position. For this case there is no switch at the end of the movement, so the program now uses the tacho. The program counts the tacho changes and once there are 100 (28% of a circle) the motor stops with the float instruction. The LEGO finger is now about one centimeter above the surface.

In the except-construct (**Listing 1c**) any errors or a Ctrl-C instruction from the user are caught. The program gives a brief abort instruction to the Raspberry Pi library and then terminates itself as well. Without this construct it is obviously still possible to exit the program using Ctrl-C, but then the library continues running and that is clearly not the intention.

The steps

1. Set the LEGO finger in the rest position, that means as far as possible from the switch (against the bottom surface for example).
2. Start the Raspberry Pi and wait until the booting process has completed.
3. Copy the test folder from the download to the Raspberry Pi (using WinSCP, for example).
4. Return to the test folder with the instruction 'cd test'.
5. Now issue the instruction 'python uselessbox.py'.
6. Wait until the text 'Useless Box Program running' appears on the Raspberry Pi and turn the switch on (and get your fingers out the way!). The LEGO finger will now return the switch to its off position.

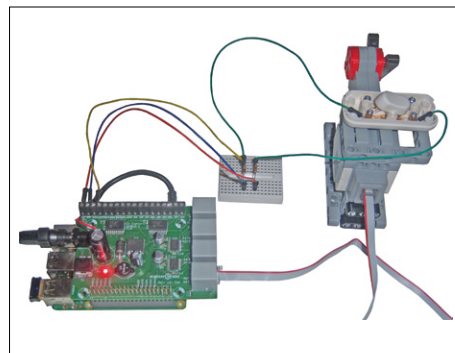


Figure 7. The complete model.

7. Repeat step 6 as many times as you like and then stop the program using Ctrl-C.

This is only a simple application example. The LEGO control board can of course be used to realize much more complicated applications with its four motor control connections and 16 I/O pins. ◀

(150597)

Web Links

[1] www.elektor.com/150597

Listing 1a.

```
try:
    while 1:
        # wait for the user
        # to push switch on
        if ioread(16):
            # push switch off
            time.sleep(0.5)
            reverse(1)
            pwm(1,60)
            while ioread(16):
                time.sleep(0.000001)
            brake(1)
```

Listing 1b.

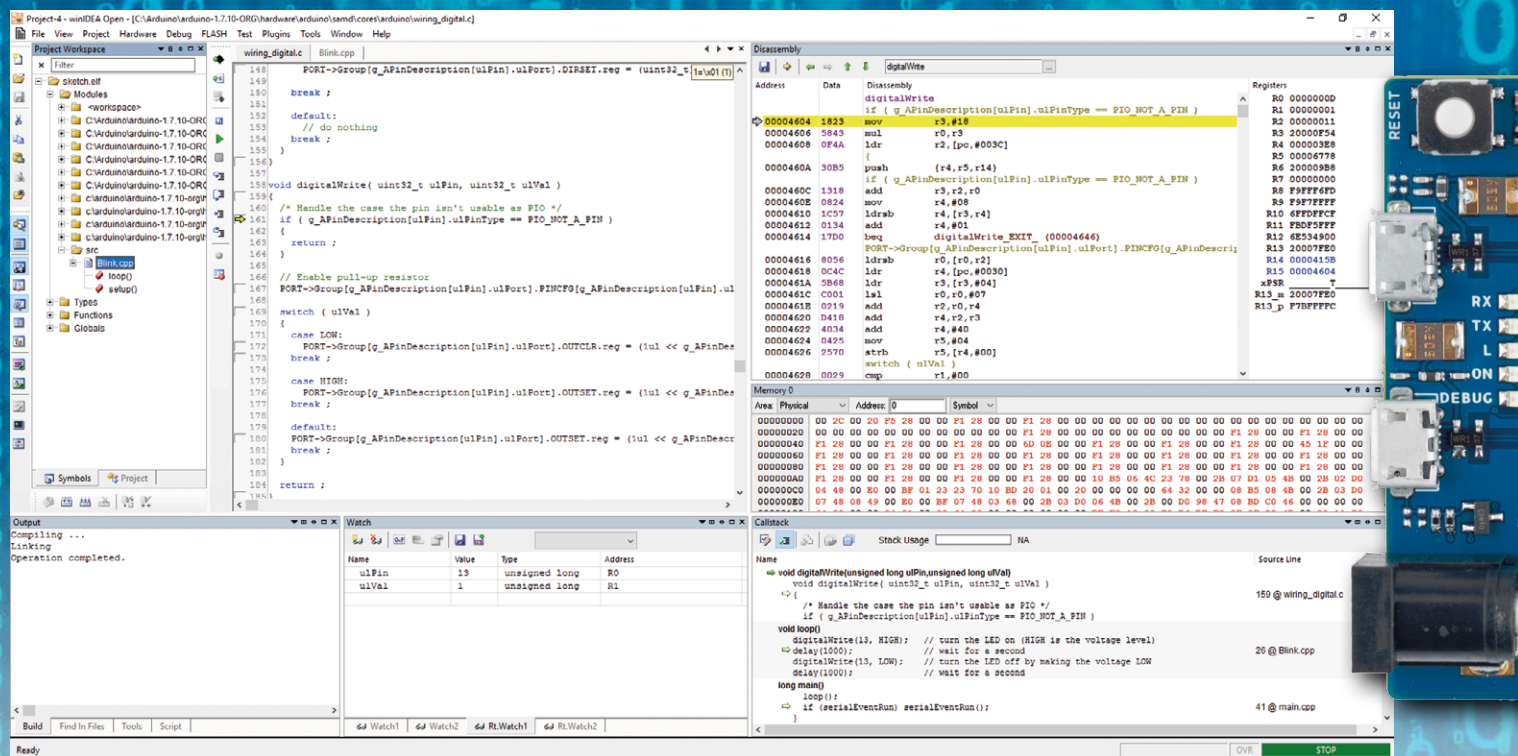
```
# retract lever
forward(1)
pwm(1,30)
counter1=0
lastpin1=tacho(1)
while counter1<100:
    time.sleep(0.000001)
    if tacho(1)!=lastpin1:
        counter1+=1
        lastpin1=tacho(1)
# float motor
float(1)
```

Listing 1c.

```
except:
    # user aborts program with
    # Ctrl-C
    print "Program aborting"
    abort()
    print "Done"
```

Debugging the Arduino Zero & M0 Pro

Delving deeper into the world of Arduino



By **Stuart Cording** (iSystem, Germany)

Boards like the Arduino M0 Pro and DUE provide, respectively, the same pin-out as their 8-bit Arduino Uno and Mega cousins, but with the advantage of significantly more SRAM, flash memory and CPU clock speed. Perhaps the only disadvantage or challenge with these boards is the 3.3-volt restriction on the input and output pins. However, these advantages, combined with support for many of the available shields on the market and the established base of libraries, make the 'upgrade' simple and fast.

The biggest challenge remaining for the advanced hobbyist or professional using Arduino for rapid prototyping is the limited Integrated Development Environment (IDE). For entry into the world of Arduino, the Arduino IDE is perfect with its simplicity and clarity. Once a sketch becomes slightly more involved, however, and failures in the code cannot be found with a blinking LED or serial message output alone, the sheer power of a professional IDE is highly alluring.

Debug (as) a Pro, Pay Nothing

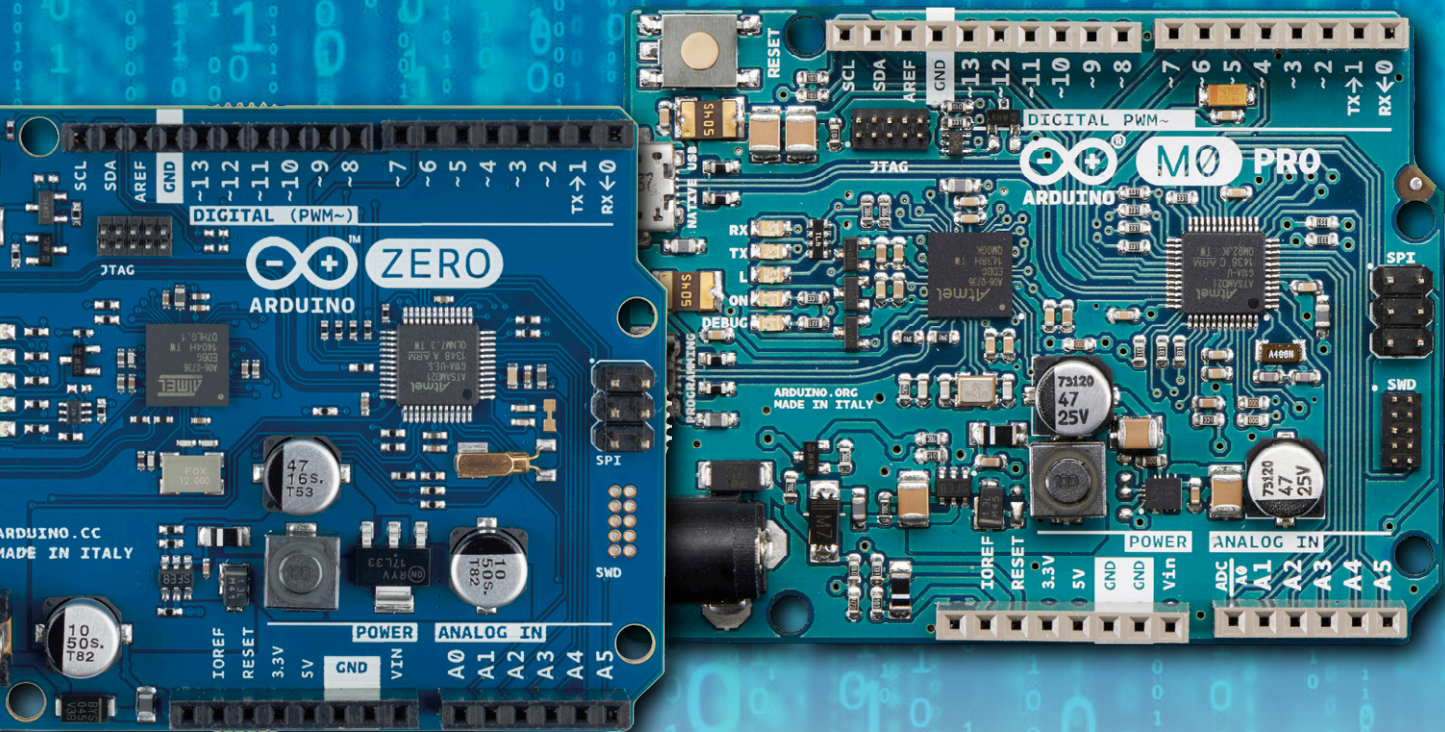
The Arduino M0 Pro (from Arduino.org) as well as the Arduino/Genuino Zero (from Arduino.cc), based on the SAM D21G from

Atmel (now Microchip), are kitted out with a slightly different programming interface compared to previous boards. The device behind the USB "programming" connector is also an Embedded Debugger (EDBG) which can be used by many different development environments as an interface to not only program the board but additionally to interrogate the internal workings of the MCU at the heart of the system. With an appropriate IDE, it is then possible to debug not only your sketch, but the entire Arduino core code and libraries.

One such IDE is iSystem's winIDEA Open, the free version of their IDE targeted at MCUs that use ARM's Cortex-M processor technology. As a professional tool it is easy to get started, as, at the most basic level, all that is needed is the output file from a sketch built in the classic Arduino IDE. However, as we will discover, the build manager can also be invoked to speed up the compilation of sketches. Additionally, the integrated test environment testIDEA can be used to check whether any bugs have scurried into our project during development.

There are many details and nuances behind configuring and using such a tool. In order to provide both an overview for those simply interested in this concept as well as detailed instructions for those who would like to go hands on, the overview in this article is supplemented by a series of tutorials and code available online [1].

With the introduction of ARM Cortex-M-based 32-bit microcontrollers to the Arduino portfolio, the maker has some very powerful hardware in their hands. For entry into the world of Arduino, the simplicity and clarity of the Arduino code editor is perfect. However, once a sketch becomes more involved, the power of a professional development environment becomes highly alluring.



My First Project

When starting a new project for a SAM D21 MCU in other IDEs, such as Atmel Studio which has been covered by Elektor before [2][3], you really need to start from a template and have to include many pre-defined library source code files. The winIDEA Open IDE is different in that it forms the premise that the output of the compiler toolchain, for example the ELF file containing the binary code of your Arduino sketch, is the most important file of your project. From this single file, the IDE can find all of your source code and the core Arduino files and libraries needed to debug the application code. Thus the simplest method to debug an Arduino sketch is to create and build it in the Arduino IDE and then import and download the resulting ELF file into the Arduino M0 Pro using winIDEA Open. In the first three projects of Tutorial 1 [1], the classic 'Blink' sketch is created in the standard Arduino IDE. Due to slight differences between the Arduino M0 Pro and Arduino/Genuino Zero, winIDEA Open needs to be configured slightly differently. However, for those who just want to get started, a preconfigured workspace is provided allowing the developer to simply open the appropriate workspace and download the ELF file that the Arduino IDE has created.

Once the code is in the MCU, it is then possible to start exploring the internal workings of the Arduino core code. In the "Project

Workspace" window (Figure 1) all the files that belong to the project, along with all the used functions, are listed in a tree structure just like in Windows' File Explorer. So, if you have ever wondered what code lies behind the `delay()` function, simply expand the "Functions" folder, scroll down until you find `delay(unsigned long ms)` and double-click. In the editor window, the function will be displayed. If you are interested to see when the function is called, you can set a break-point on a line of code where a grey box is also shown in the editor's gutter (the grey region in the text editor to the left of the line numbers). Simply right-click with the mouse and select "Set Breakpoint". Once the code is restarted, the MCU will halt when it reaches this breakpoint, allowing you to analyse the content of variables, registers and memory.

Note that, despite the Arduino environment's apparent simplicity, the code in its core software makes good use of many of the advanced tricks the C programming language has to offer. As such, some 'symbols' (names of variables and functions) may appear listed in the Project Workspace window but may not actually be associated with any code of significance in the software.

Speeding Up the Workflow

One of the issues with the method discussed thus far is that the original [Blink.ino](#) cannot be debugged properly. The issue

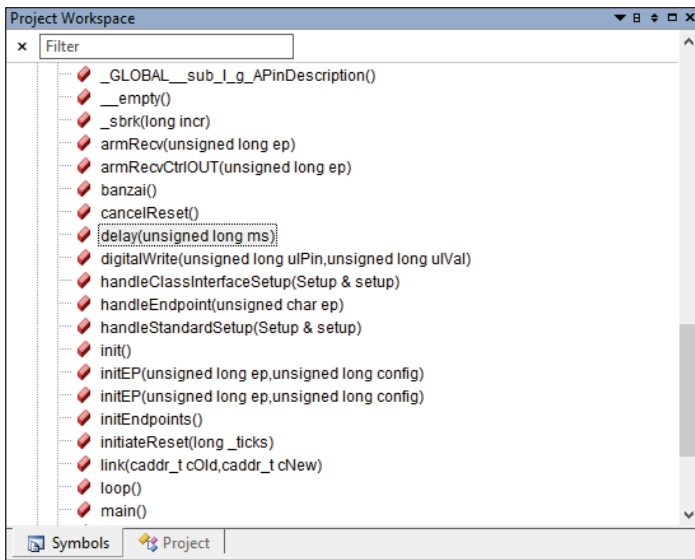


Figure 1. The 'Functions' view shows all the functions included in the sketch, even those belonging to the Arduino core and library code.

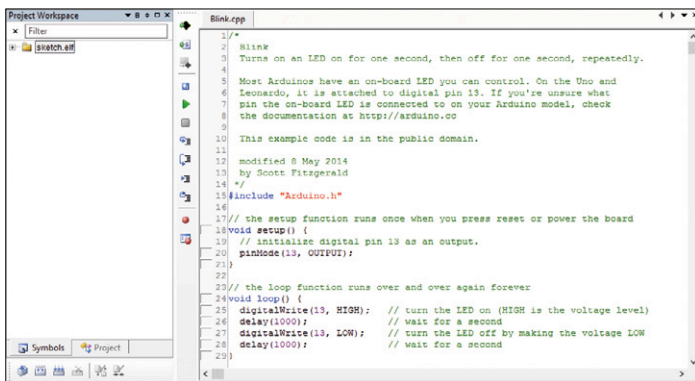


Figure 2. Blink.cpp in the winIDEA Open IDE.

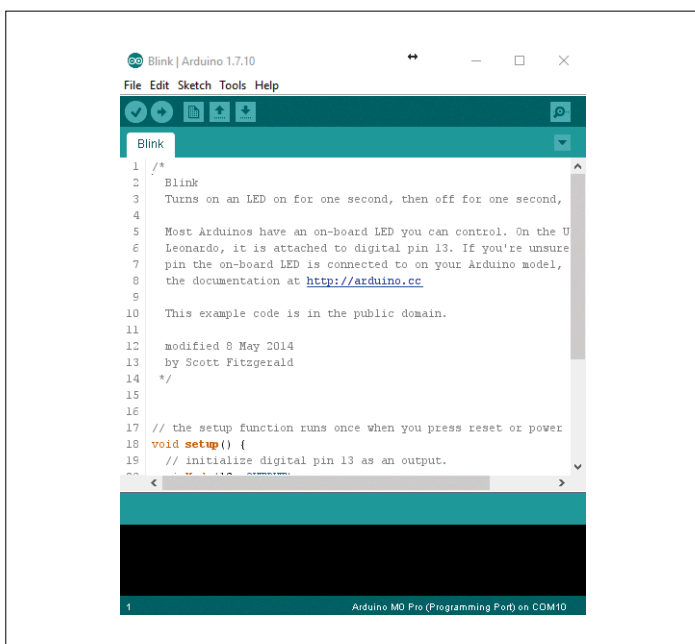


Figure 3. The Blink sketch in the Arduino IDE.

relates back to how the Arduino IDEs compile the sketch. For the beginner, this is really no problem, but when wanting to go deeper into the code, it would be nice to be able to build a sketch outside of the Arduino IDE.

There is also another benefit to be gained from this. One of the most frustrating things about the standard Arduino IDEs is that every compilation rebuilds the entire project from scratch. Again, for smaller sketches this is no big deal. However, as soon as a couple of libraries are included and you have several source files, rebuilding and programming the board starts to become quite a laboured activity. Here again, a professional IDE can help by providing the means to build the project outside of the IDE.

In order to make the build process more intuitive and faster, Tutorial 2 [1] uses a Makefile to build our sketch. This requires a few changes to the creation of your sketch as follows:

- Your sketch will need to be stored in a file named `<SKETCH NAME>.cpp` and stored in a folder named 'src'.

Your sketch will need the line of code `#include "Arduino.h"` (see **Figure 2**) added to the top of it (before the `setup()` function call).

In addition, you will need the make utility which can be installed as part of MinGW. Detailed instructions on how to do this can be found in Tutorial 2, available from [1].

Now, instead of building the sketch in the Arduino IDE (**Figure 3**) and then simply programming and debugging the code in the winIDEA Open IDE, the whole process can be undertaken in winIDEA Open.

As a result, we can now debug our sketch. The project workspace works in a manner similar to that of Windows Explorer, allowing the elements listed to be "expanded" to show their contents in the same way folders can be expanded. In the Project Workspace simply expand the elements listed using the plus symbol to the left of each element in the following order: sketch.elf → Modules → src → Blink.cpp (**Figure 4**). Finally, double-click on `loop()` or `setup()` and the editor will open the source code at the respective line of code where the function is implemented. As we see in the gutter, each line of code has a grey box associated with it allowing us to set a breakpoint in the code execution if we wish. Simply right-click on the desired line of code and select "Set Breakpoint".

Tips & Tricks

Now is probably a good time to note the limitations of the debug features of the SAM D21. In total, only three breakpoints can be set at any one time so, at some point, you will get a window pop up explaining that all breakpoint resources are used up (**Figure 5**). The easiest method to work around this is to disable a current breakpoint (right-click on a line with a breakpoint and select 'Disable Breakpoint') and then set the new breakpoint at the location of your choosing. It is also possible to 'Clear Breakpoint', but disabling a breakpoint has the advantage of leaving a red marker in the gutter of the code editor as a reminder of where the breakpoint previously was. This is especially helpful as you jump back and forth in the code. In the event that you are interested in the efficiency of the code generated by the Arduino tool chain (which incidentally is GCC), we can also open a disassembly window from the menu View

→ Disassembly. If we click inside this window (**Figure 6**), any further single-stepping of the code will be executed instruction by instruction, instead of source code line by source code line as is the case in the editor window. In this way it is also possible to see how the CPU registers are used during the calling of functions to pass parameters, amongst other things.

Beyond Debugging

Typically, when developing an application, specific functionality will be broken out into separate functions. For small projects, where perhaps a single developer is responsible, it is relatively easy to keep track of what is working, what isn't and where a bug might be when the code fails. However, for larger, multi-programmer projects, it can help to have a battery of tests available that, when executed, ensure that the code is still operating as originally specified. For this purpose, winIDEA includes the Original Binary Code (OBC) unit testing tool testIDEA. Here we will use it to develop some tests for a function that evaluates an imaginary input value and returns a new time delay for our `delay()` function call.

The algorithm implemented in the code is quite simple (**Figure 7**). If the passed value is less than 50, it returns 150. If the passed value is between 50 and 99, it returns 1000. For all other values, it returns 1750. Having developed the code, it would make sense to develop a short test that could be executed to ensure that it still works, even if someone else makes changes and introduces a bug into the code.

Before writing tests for a function, it is worthwhile initially considering how you would go about testing it by writing the steps down on a piece of paper or making a small table of input and expected output values. In the **Table 1** we have focused on input values at the extremes of possible inputs as well as values that are close to the boundaries defined in the algorithm (50 and 100).

In order to develop our tests, from the menu bar, we select Test → Launch testIDEA. Upon starting testIDEA, the tool suggests creating a 'test specification file'. This file is where our tests will be stored and the file extension is 'iyaml'. Here we explain how to create a test:

- From the menu, select Test → New Test...
- Now we need to define the function to be tested. Before we can do this, we have to refresh the link to our winIDEA

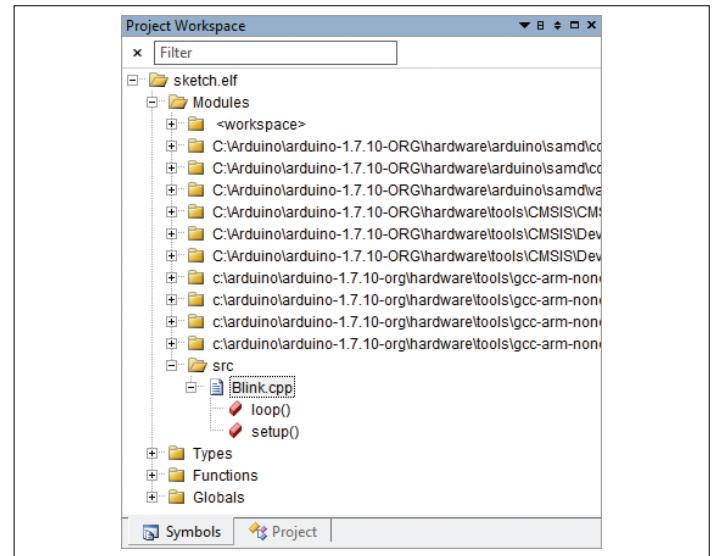


Figure 4. The sketch executable can be explored as if it was a folder tree.

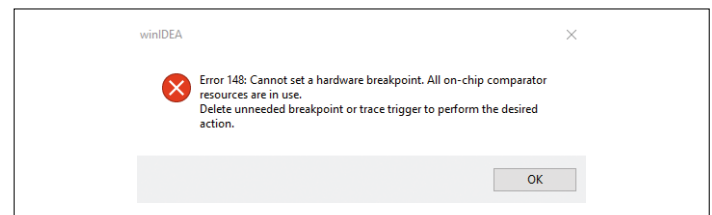
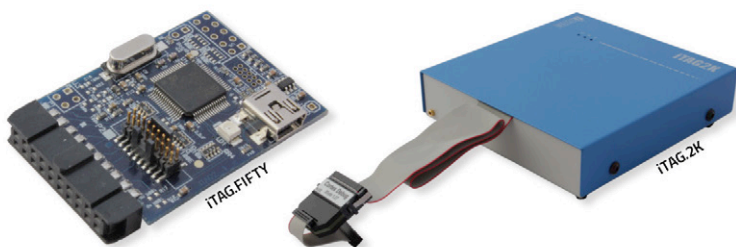


Figure 5. Message when no more breakpoints are available.

project by clicking the refresh symbol. Once complete we can select our function to be tested, `evaluateNumber()`, from the drop-down list.

- In the Parameters box we enter the value to be passed to the function. For test number 1, this is 0.
- Next we enter the Expected Result, first selecting the 'Default expression' radio button. In the field we enter 150, which is the expected response entered in our table.
- Click OK and the first test is complete.

Advertisement



iSYSTEM Enabling Safer Embedded Systems

- Debugging of Cortex®-M MCUs
- Original Binary Code (OBC) Testing
- Automated Test Support with Python API
- Integrates into Jenkins Continuous Integration (CI) tools

www.isystem.com/cortex-m

Figure 6. Disassembly view for the Blink sketch.

Table 1. Input values to exercise our algorithm and the expected output values.

Test Number	Input Value	Expected Response
1	0	150
2	1	150
3	48	150
4	49	150
5	50	1000
6	51	1000
7	98	1000
8	99	1000
9	100	1750
10	101	1750
11	150	1750
12	200	1750
13	255	1750

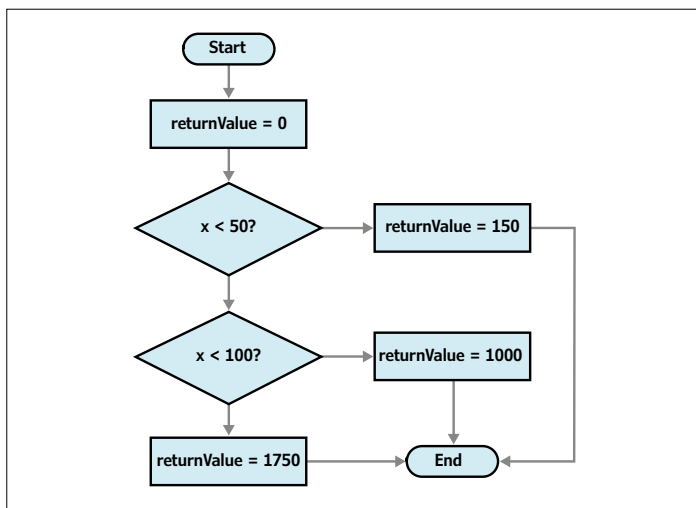


Figure 7. Flow diagram of a simple algorithm.

The uniqueness of OBC testing is that the testing itself all occurs on the MCU target in use and not, like in some other unit testing tools, in a simulation of the MCU. When we now execute this test, the test will be actually executed on the MCU target after downloading the code into the MCU's flash memory. This is performed by selecting Test → Run All Tests. Hopefully, upon executing the tests, the result will be "OK", indicated by a small green checkbox next to the test in the Outline panel. In order to distinguish all the different tests from one another, we can add some meta data to our test. In the Form panel, select 'Meta' and add a Test ID such as 'Test_1'. We can now go on to add the remaining tests to our testing plan. Upon completion, we can run all the tests which should result in a pass in each case. Of course, the real value is in using the tool to find any bugs that have slipped in whilst we weren't looking. Let us assume that a colleague on the team misunderstands the specification and decides that all the less-than ('<') comparisons should actually be less-than-or-equal ('<=') comparisons and changes them in your code. The first time you become aware of the issue is when the project as a whole does not work as expected. To trial this potential mistake, replace the '<' with '<=' at lines 20 and 22 in the `Blink.cpp` source code (Tutorial 3, Project 5 [1]). In order to see where the bug may have occurred, simply rebuild the application in winIDEA, open up the test specification in testIDEA and run all tests. You should see that tests 5 and 9 now fail and, working from that result, it should be possible to determine the cause of the error.

Sounding off

The Cortex-M-based Arduino boards have enormous potential, especially in the area of rapid prototyping. STMicroelectronics recently announced a Cortex-M4 STM32-based board in the form of the STAR Otto. Such boards will likely be used for complex prototyping with Ethernet and Wi-Fi shields. A professional debugging environment, such as winIDEA Open, will certainly provide relief when searching for why a sketch is not quite working as expected. ◀

(160228)

Web Links

[1] www.elektormagazine.com/160228

[2] www.elektormagazine.com/130392

[2] www.elektormagazine.com/140037

The SAMD21 is just one in the range of SAM D ARM Cortex-M0+ based, 32-bit microcontrollers from Atmel. These devices offer up to 256 kB of flash memory and 32 kB of SRAM. Additionally, the devices boast a wide range of interfaces, including full-speed USB, a real-time counter for implementing a clock, USART, SPI and I²C interfaces, and a 12-bit ADC. Optimal energy efficiency is also covered with the power consumption lying at 70 μA/MHz. The family also features an automotive grade version of the MCU and some devices can be used to implement a capacitive touch interface, making use of Atmel algorithms to implement buttons, sliders and wheel user interfaces.

www.atmel.com/products/microcontrollers/arm/sam-d.aspx

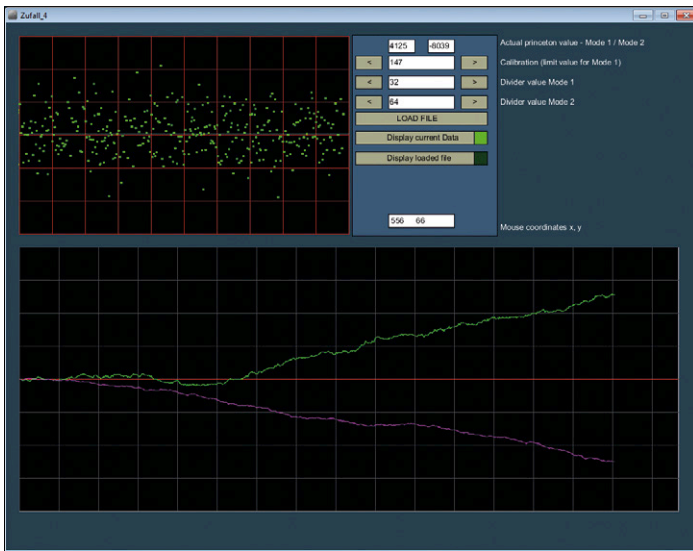


Figure 3. User interface presented by the program, which automatically starts collecting data when it is launched.

an Arduino Uno is used in this case. There are two connections: first, the signal connection, which is taken to analog input A0 of the Arduino; and second, the ground connection. The noise source can be powered from any suitable external 2 x 15 VDC power supply. The Arduino is powered from the PC over the USB cable.

Software

As already mentioned, two separate pieces of software are needed in this project: one running on the Arduino and one on the PC [1]. The program running on the PC is written in Processing. The Processing programming environment (which so far I have only used under Windows XP and Windows 7) can be downloaded from the Internet free of charge [2]. A Processing file must always be placed in a directory that has the same name as the program file itself.

It is very important to ensure that you enter the correct COM port number in the setup part of the Processing code so that the Arduino is recognized by Windows. You can find the correct number using the Device Manager. For example:

```
serport = new Serial(this, "COM3", 115200);
```

Unfortunately it can happen that the port number changes by itself while you are experimenting with the system.

After using the Arduino IDE to flash the sketch into the Arduino, it can be connected to the PC and then the Processing program can be launched. After a brief delay the signal should appear on the screen in the upper graph window (see **Figure 3**, above), and it should be symmetric about the horizontal center line. If it is shifted or if the amplitude is not correct, P1 and P2 must be adjusted. Below we will describe how you can fine-tune these settings using the software.

Operation

Upper graph window

The points shown in the upper graph window represent the random sample values received from the noise source. Every second approximately thirty of these values are received from the Arduino. Filling the window takes about ten seconds, corresponding to about 300 samples per pass that are available for further processing.

Lower graph window

Two curves slowly build up in the lower window, one green and one purple. Each time the upper window is filled each of these curves is extended by one pixel. The curves are produced from the random sample values as follows.

Purple curve (mode 2)

The program checks whether the current sample value of the noise signal is even or odd, and respectively increments or decrements a variable. Then the value of this variable is plotted.

Green curve (mode 1)

If a sample is above the horizontal center line a variable is incremented; if it is below the center line, it is decremented. It is not easy to set the offset potentiometer in the hardware sufficiently accurately, and in any case the offset will vary slightly with temperature, and so a calibration process is required. The calibration buttons help with this.

Calibration buttons

These are the two buttons towards the top of the display, which can be used to make vertical adjustments to the horizontal line in the upper graph window that divides the positive samples from the negative ones. Depending on this setting, the green curve in the lower graph window will either run horizontally or will trend up or down. The offset potentiometer in the noise source should be adjusted so that the green curve runs as horizontally as possible when the threshold position is set to 150 using the buttons. In this case the random points displayed in the upper graph window should appear to be perfectly symmetric with respect to the center line. The threshold value of 150 appears among the variable declarations in the Processing code, as follows:

```
int limit = 150 ;
```

If a different value suits your hardware better, you can modify the code.

Divider buttons

The next two buttons down allow for the expansion and compression of the curves in the Y direction, to allow them to be analyzed in greater detail. The two numbers shown at the top of the display are the most recent values plotted on the curves.

Files and 'LOAD FILE'

Normally every thirty minutes a file is created storing the data represented by the curves in the lower graph. These files are saved in the same directory as the Processing program. The file name is determined when the program is started and consists of the time and the date when the program was started. The time interval between files being saved can be adjusted by modifying the variable declarations in the code: details on

Spooky action at a distance

Since the end of the 1990s a research group at Princeton University has been conducting a study, called the Global Consciousness Project, or GCP, that collects and evaluates data generated by seventy random number generators located all around the world.

In theory the distributions of numbers from these generators should all be identical. For example, if the plot of noise sample values is shifted so that they are symmetrical about the zero line then occurrences of positive and negative values will balance when analyzed over a sufficiently long time period. The consequence of this is that the plot obtained by accumulating a fixed number of these values should wobble about the zero line; if the shift is not exactly correct then the plot will drift at an approximately constant rate in one direction or the other. In rare cases the researchers at Princeton observed significant disturbances to this delicate

balance when an event of global significance occurred that had an emotional effect on a large fraction of human beings: the events of 11 September 2001, the death of Princess Diana, or the Madrid attacks. In all these cases the readings collected showed deviations from their expected properties.

Further information on the background of this research, as well as criticism of it, can be found at the following sites:

<http://global-mind.org>

<http://global-mind.org/results.html#alldata>

<http://global-mind.org/control.distribution.html>

<http://noosphere.princeton.edu/story.html>

<http://subroutine.jimdo.com> (author's website, partly in English, mostly in German)

how to do this are given in the program.

Clicking on 'LOAD FILE' brings up a file selection dialog where you can navigate to any previously-saved file and reload it.

Display current data / Display loaded file

The data plotted in the lower graph can be changed by clicking on the two buttons 'Display current data' or 'Display loaded file'. When the plot is changed to 'Display current data' it is not updated until the plot in the upper graph reaches its right-hand edge.

The figures below the buttons show the coordinates of the mouse pointer when the mouse button is pressed. This is a useful tool to have when adding new elements to the user interface design.

Interpretation

Of course the plotted curves are open to various interpretations. When you launch the program you will notice that, although there are many deviations of small magnitude, the overall trend of each curve is firmly in one direction. The tiny jagged deviations are a natural consequence of the random nature of the noise generator; they can also be caused by temperature changes. However, if the curve should suddenly show a clear change in direction from its previous average trend, perhaps something is afoot! Perhaps you might want to take a quick look at your favorite news website to see what is going on...

On the other hand, you might prefer to spend your time modifying the processing program to display and analyze other types of data more obviously influenced by the outside world, such as readings from a temperature sensor!

(150831)

Web Links

[1] www.elektormagazine.com/150831

[2] <https://processing.org/download/>

Listing 1. Arduino sketch (excerpt).

```
void setup()
{
  Serial.begin(115200);
}

void loop() // Endlosschleife
{
  int sensorValue = analogRead(A0);
  Serial.println(sensorValue);
}
```

Notes on the Processing code

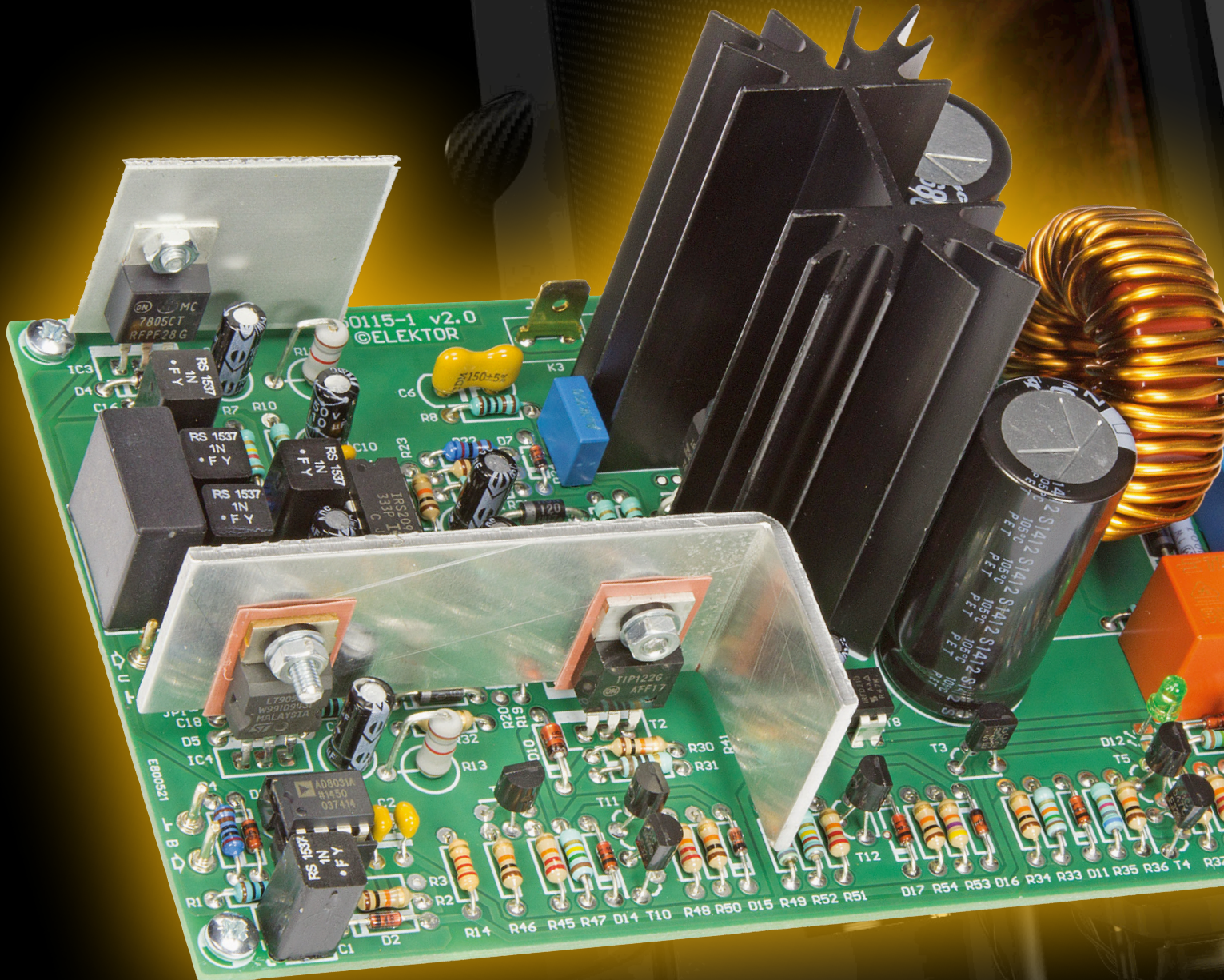
The code might at first sight seem rather complicated. This is because Processing does not have a built-in way to construct graphical user interfaces for programs. All buttons and text labels, and all the mouse events that connect together to form such an interface, must be written as 'normal' source code. However, the result works well and the payoff for the complexity of the code is that graphics handling in Processing is astonishingly quick. I have added thorough comments

to the source code wherever required. Readers interested in programming in Processing will find many interesting ideas in the listing, such as how buttons and text labels are programmed, how to display the contents of an array as a plot in a graphics window, store them to a file and read them back again, how to determine the time and how to create a file selection dialog.

Cool Power

A high-performance amplifier with lots of output power

By Ton Giesberts (Elektor Labs)



with D-Watt

Elektor has a long history and a good reputation with high-end yet DIY, audio projects. This time we proudly present a new power amplifier aimed at all audio enthusiasts, built around a digital audio driver IC and operating in class-D. This has the advantage that the amplifier can deliver a lot of output power (like 200 watts into 8 ohms) with very low heat dissipation. Say goodbye to big heatsinks!

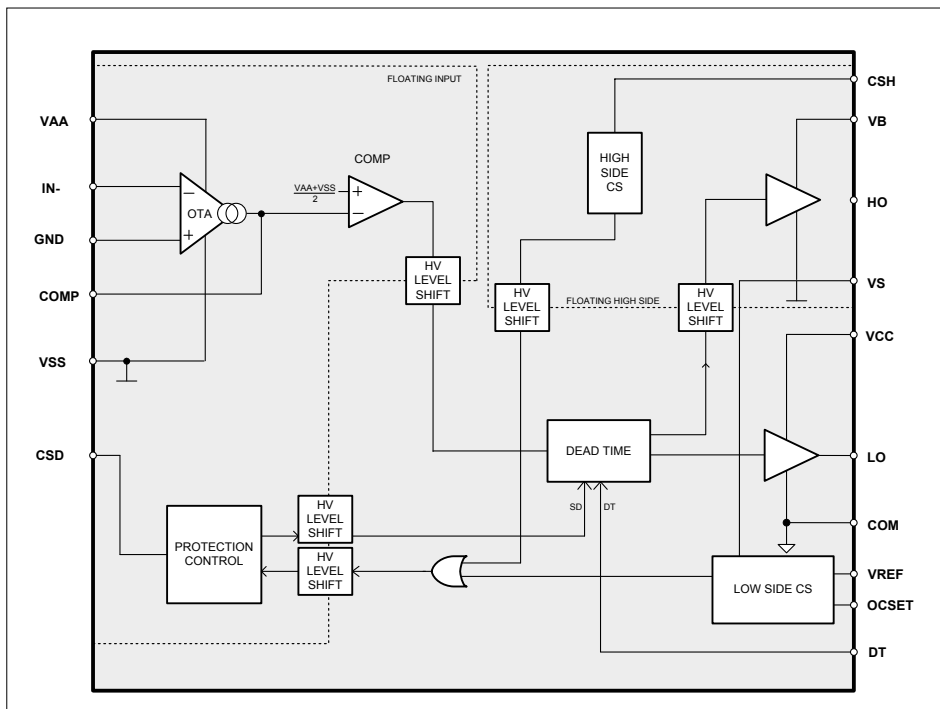


Figure 1. Block diagram of the IRS2092 digital driver IC used in this amplifier.

It has been nearly three years since we presented our last power amplifier project in Elektor — the *Q-Watt Audio Power Amplifier* [1]. That was a standard class AB power amplifier with high output power and low distortion. Since then a large number of audio enthusiasts have built their own copy of that amplifier.

With the design presented here, we are taking the digital route for a change. In a class-D power amplifier, the output transistors operate in switching mode with a very high switching frequency. Pulse width modulation converts the amplitude of the analog input signal into a specific ratio of positive and negative pulse widths in the output signal. A filter at the output blocks the high-frequency switching signal and passes the amplified audio signal to the loudspeaker. The main advantage of this type of digital power amplifier is its high efficiency, which allows the heatsinks to be much smaller than with a comparable conventional design.

The key components in this amplifier are a driver IC in a 16-pin package and a dual power MOSFET. Together with extensive protection circuitry, they are mounted on a single PCB, so you only have to add a suitable power supply — either regulated or unregulated. There are no SMDs in the design, so building the board is very easy and you can experiment with dif-

ferent components or component values if you wish.

Design choices

It is possible to build a class-D amplifier entirely from discrete components, but that does not make for a simple design. Fortunately, there are several semiconductor manufacturers who make dedicated ICs for class-D amplifiers. A large variety of ICs are available for low power amplifiers, but the choice is fairly limited if you are looking for high power combined with high quality. We found an outstanding solution in the form of an IC designed by International Rectifier (now Infineon): the IRS2092. This 16-pin IC goes by the name “Protected Digital Audio Amplifier”, and it integrates nearly everything you need to build a relatively simple high-performance class-D amplifier. You only have to add some power transistors.

The IC contains gate drivers that can deliver high currents for the connected power MOSFETs, a PWM modulator with built-in oscillator, bidirectional current protection and programmable dead time for driving the MOSFETs, and it can operate with switching frequencies up to 800 kHz (see the block diagram in **Figure 1**). Along with the datasheet [2] and a wealth of application notes, Infineon offers a lot of reference designs for the

IRS2092. So why did we develop our own design when someone else has already done it for us? The answer, as you might guess, is that a practical design turns out to be good deal more elaborate than a simple copy of the reference design.

Amplifier

With an IC designed for a specific application, it’s a good idea to start with the standard application circuit from the manufacturer. That certainly holds true for the IRS2092. A detailed description of this IC can be found in application note AN-1138 [3].

Our version is shown in **Figure 2**, where the configuration around the driver stage IC2 and the dual MOSFET T1 is virtually “by the book”. Transistor T1 is specifically designed for this sort of application, with two n-channel MOSFETs housed in a single package. The IRFI4020H-117P is a member of the Infineon family of digital audio MOSFETs and has a rated $U_{DS\ max}$ of 200 V, $I_{D\ cont}$ of 9.1 A and $R_{DS(on)}$ of 80 m Ω . It is a perfect choice for power amplifiers with output power of 200 to 300 W into 8 Ω . Thanks to the single package, this dual MOSFET is easy to mount on the circuit board and on a heatsink.

The feedback network between the output and the input consists of R7 and R8, along with capacitor C6 to suppress RF signals (in combination with R8). The gain is determined by the ratio of $R7 + R8$ to $R5 + R6$. The input impedance of this inverting amplifier is determined by R5 and R6 and is fairly low (about 3.3 k Ω). For that reason we added a buffer stage in the form of opamp IC1A (an AD8031ANZ, which is a fast rail-to-rail opamp). You can choose which of the two inputs you want to use by fitting a jumper in the appropriate position on header JP1. The input impedance of the opamp stage is 11 k Ω . It is designed with a gain of 1, but if you wish you can increase the gain by fitting a resistor in the R3 position ($A = R4/R3 + 1$). The supply voltage for the opamp is provided by the ± 5 V voltage regulators, which also supply the auxiliary voltages for IC2 (see below).

Figure 2. Complete schematic diagram of the D-Watt amplifier. The amplifier circuitry is shown in the top half, with the protection circuitry in the bottom half.

Virtually all of the components around IC2 have specific functions. The majority of the resistors connected to the IC determine the two current limit levels, the dead time, the bootstrap drive voltage for the high-side MOSFET, and the feedback from the output.

IC2 requires two auxiliary voltages (+5 V and -5 V). Instead of deriving them directly from the main supply voltages using Zener diodes and series resistors, we use separate voltage regulators (IC3 and IC4) to keep the auxiliary voltages stable even when the amplifier is powered from an unregulated power supply. There are Zener diodes (D4 and D5) at the inputs of the voltage regulator ICs to keep their input voltages at a safe level (the maximum allowable supply voltage is ± 70 V) and reduce the dissipation of these ICs. To prevent HF switching noise from affecting the voltage regulators,

additional decoupling at the auxiliary supply voltage pins of IC2 (VAA and VSS) is provided by networks in series with the supply lines (R10/C10/C11 and R11/C12/C13). Diode D3 protects the IRS2092 if the negative supply voltage drops out. The voltage on the negative supply voltage pin VCC (which is used to drive the low-side MOSFET) is +12 V with respect to the main negative supply voltage -VP. This pin draws somewhat more current than the auxiliary supply voltage pins, so a stabilization circuit consisting of voltage regulator IC5 in series with an external Darlington power transistor (T2) is provided here. In this arrangement the transistor handles the additional power dissipation. The VCC pin of IC2 is locally decoupled by C20 and R15.

The voltage on the positive supply voltage pin of IC2 (VB), which floats rela-

tive to ground, is used to drive the high-side MOSFET. This pin is connected to a standard bootstrap circuit consisting of R16, D6 and C21, along with R21 which is necessary to charge C21 before the PWM oscillator starts running. Resistor R16 limits the peak current.

The dead time (the time interval when both MOSFETs are cut off to prevent large peak currents through both devices at the same time) can be set with voltage divider R19/R20 to one of four discrete values: 105, 65, 40 or 25 ns. Here we selected the minimum dead time setting of 25 ns by omitting R20. Note that increasing the dead time causes a significant increase in THD in the upper frequency range. You should bear in mind that MOSFETs generally need about twice as much time (or longer) to switch off than to switch on. Resistors R25 and R26 limit the charge and discharge currents to and from the gates of the power MOSFETs. The short-circuit current ratings of the driver outputs are 1 A for sourcing and 1.2 A for sinking. With a value of 18 Ω for R25 and R26, the peak charge and discharge currents are limited to about 0.5 A.

Voltage divider R17/R18 determines the low-side current limit. Here the on-resistance ($R_{DS(on)}$) of the MOSFETs is used for current sensing. The positive temperature coefficient of $R_{DS(on)}$ has the advantage that the current limit level decreases when the temperature of the MOSFETs increases. The IRS2092 has an internal 5.1 V reference voltage (VREF on pin 7) for precise setting of the current limit. Unfortunately, the drain-source resistance of MOSFETs has a fairly high tolerance. With a gate-source voltage of 10 V, the $R_{DS(on)}$ of the MOSFETs used here is 80 m Ω typical and 100 m Ω maximum (at 25°C). At 125°C, $R_{DS(on)}$ rises to more than 175 m Ω . In our design we set the current limit voltage to 1.56 V, which puts the current limit at 19.5 A (typical) at a junction temperature of 25°C. At 125°C the limit level drops to approximately 8.7 A. If the output current is too high, the IRS2092 shuts down and the LO and HO outputs are set low to protect the MOSFETs.

The high-side current limit is set using the CSH input (pin 16). This pin has a fixed threshold voltage of 1.2 V relative

Measured performance

Measured with the 6/8 Ω version and ± 60 V supply voltages (Hypex SMPS400A400), BW = 22 kHz, R7 = 100 k Ω

• Input sensitivity:	1.3 V (189 W / 8 Ω , THD+N = 1%) 1.25 V (174 W / 8 Ω , THD+N = 0.1%)
• Input impedance:	3.33 k Ω / 11 k Ω
• Continuous output power:	174 W into 8 Ω (THD+N = 0.1%, 1 kHz) 189 W into 8 Ω (THD+N = 1%, 1 kHz) 224 W into 6 Ω (THD+N = 0.1%, 1 kHz) 246 W into 6 Ω (THD+N = 1%, 1 kHz)
• Power bandwidth:	5 Hz (-3 dB) to 20 kHz (-1 dB)
• Slew rate:	15 V/ μ s (8 Ω)
• Rise time:	4.4 μ s (8 Ω)
• Signal to noise ratio:	>77 dB (linear, BW = 22 Hz to 22 kHz) >80 dBA (reference 1 W / 8 Ω)
• Total harmonic distortion plus noise:	0.014% (1 kHz, 1 W / 8 Ω) 0.004% (1 kHz, 50 W / 8 Ω) 0.024% (1 kHz, 1 W / 6 Ω) 0.0046% (1 kHz, 50 W / 6 Ω)
• Intermodulation distortion:	0.013% (1 W / 8 Ω) 0.023% (50 W / 8 Ω) 0.013% (1 W / 6 Ω) 0.017% (50 W / 6 Ω)
• Dynamic IM distortion:	0.037% (1 W / 8 Ω) 0.009% (50 W / 8 Ω) 0.043% (1 W / 6 Ω) 0.008% (50 W / 6 Ω)
• Damping factor:	174 (1 kHz, 8 Ω)
• Efficiency at full power:	93% with 8 Ω load 94% with 6 Ω load (THD+N = 0.1%)
• DC protection:	+3 V / -4 V
• DC output offset:	<0.3 mV
• Switch-on delay:	6 s

▶ The main advantage of a digital power amplifier is its high efficiency

to VS (pin 13). For high-side current limiting, a blocking diode (D7) is necessary in addition to the voltage divider in order to protect the CSH pin against high voltage when the high side is switched off. The 0.6 V forward voltage drop over D7 increases the drain-source voltage of the high-side MOSFET by 0.6 V. Resistor R24 provides a bias voltage for D7 when the high-side MOSFET is switched on. Voltage divider R22/R23 reduces the voltage over the high-side MOSFET and D7. All this means that if you want to set the same limit level here as for the low side, you have to take D7 into account. That yields a value of 10 k Ω for R23 and 8 k Ω (adjusted to 8.2 k Ω) for R22. See the description of this amplifier on the Elektor Labs website [4] for the calculation. The PWM signal at the output of the MOSFETs is filtered by a second-order filter consisting of toroidal inductor L1 and capacitor C29. It is followed by a Boucherot network (R28/C30), which provides an output load for the amplifier at high frequencies.

Capacitors C22–C28 decouple the supply voltages. As with all class-D amplifiers, there is a risk of power transfer from the output filter to the power supply due to the “bus pumping” effect. That mainly occurs at high signal levels and low frequencies with low load impedance. With a regulated power supply (in particular a switching power supply), this can destabilize the feedback loop. The simplest way to suppress a potential increase in the supply voltage is to connect large capacitors in parallel with the supply lines. You should start with a 10,000 μ F, 100 V electrolytic capacitor on each supply line. Be careful with this, because most switching power supplies have difficulties with large capacitive loads (check the manufacturer’s specifications).

The supply voltages for the amplifier are ± 60 V for the 6 Ω / 8 Ω version, with 100 k Ω as the recommended value for R7. The maximum is ± 70 V, but at that level the distortion increases because the PWM component of the feedback signal is stronger. If you want to use the

amplifier with a 4 Ω load, it will frequently go into current limiting with ± 60 V supply voltages. In that case it is better to modify the amplifier and use different supply voltages, as described in the “4 Ω Configuration” inset.

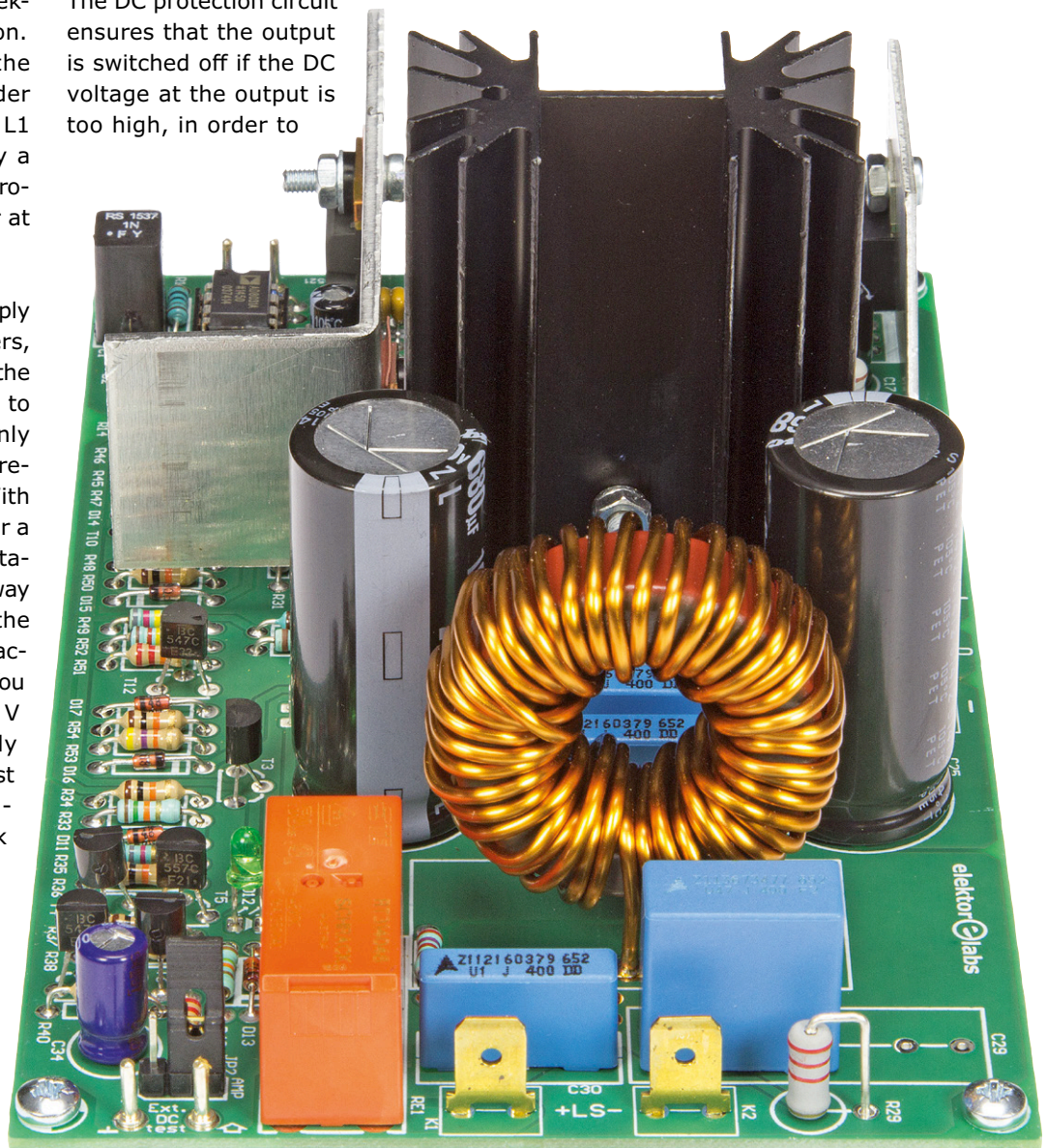
Protection

To supplement the built-in protection of the IRS2092, we added a number of external protection circuits: overvoltage protection for both supply voltages, undervoltage protection (also for both supply voltages), and DC protection with switch-on delay. Each of these protection circuits can de-energize the output relay and shut down the IRS2092.

The DC protection circuit ensures that the output is switched off if the DC voltage at the output is too high, in order to

protect the connected loudspeaker. The DC protection circuit reacts at a DC level of a few volts. A relatively high value was chosen for this because there is always some DC voltage present at the output when no load is connected and the IRS2092 is shut down. Apparently there is a leakage current from the VS pin when the IRS2092 is in shutdown mode.

The DC protection circuit consists of R38, R39, R40, C34, T6 and T7. Voltage divider R39/R40 determines the trip levels. With the indicated resistor values, they are +3 V and -4 V. Although the selected levels are somewhat higher than usual, they should not be harmful to the loud-



Component List

Resistors

Default: 5%, 0.25W

R1,R8 = 1k Ω
 R2,R23,R24,R30,R34,R36,R37,R46,R50,R54 = 10k Ω
 R3,R20 = not mounted
 R4 = 2k Ω 1%
 R5 = 330 Ω
 R6,R18 = 3k Ω 1%
 R7,R52 = 120k Ω
 R9 = 470 Ω
 R10,R11,R15,R31 = 10 Ω
 R12,R13 = 1.8k Ω , 2W
 R14 = 2.2k Ω
 R16 = 4.7 Ω
 R17,R35 = 6.8k Ω
 R19 = 5.6k Ω
 R21 = 33k Ω
 R22 = 8.2k Ω
 R25,R26 = 18 Ω
 R27 = 1 Ω , 1 W
 R28 = 10 Ω 1W
 R29 = 2.2k Ω , 2W
 R32,R40 = 4.7k Ω
 R33,R43 = 15k Ω
 R38,R39,R45,R48,R51 = 22k Ω
 R41 = 1M Ω
 R42 = 10M Ω
 R44 = 2.7k Ω
 R47 = 150k Ω
 R49 = 820k Ω
 R53 = 47k Ω
 P1 = 1k Ω trimpot, top screw adjustment

Capacitors

C1,C5,C7,C8,C9 = 1nF 1%, 63V, polystyrene, 7.18mm pitch (LCR Components type EXFS/HR 1000pF \pm 1%)
 C2,C3,C11,C13,C16,C18,C31 = 100nF 10%, 50V, X7R, 0.2" pitch
 C4 = 10 μ F 10%, 63V, MKT, 15mm pitch
 C6 = 150pF 5%, 500V, silver mica, 5.9mm pitch (Cornell Dubilier type CD15FD151J03F)
 C10,C12,C14,C20,C32 = 10 μ F 20%, 50V,

5x11 mm, 2mm pitch
 C15 = 10nF 10% 100V, X7R, 0.2" pitch
 C17,C19,C33 = 1 μ F 20%, 50V, 5x11 mm, 2mm pitch
 C21 = 22 μ F 20%, 50V, 5x11 mm, 2mm pitch
 C22,C24 = 100nF 10%, 250V, MKT, 4x10 mm, 7.5mm pitch (TDK type B32520C3104K000)
 C23,C25 = 680 μ F 20%, 100V, 28m Ω @ 2.57A, 18x40 mm, 7.5mm pitch (Rubycon type 100ZL680MEFC18X40)
 C26 = 100nF 10%, 200V, X7R, 0.2" pitch (Kemet type C330C104K2R5TA)
 C27,C28,C30 = 100nF 5%, 400V, polypropylene, 15mm pitch (TDK type B32652A4104J000)
 C29 = 470nF 5%, 400V, polypropylene, 15mm pitch (TDK type B32652A4474J000)
 C34 = 100 μ F 20%, 10V, bipolar, 2.5mm or 3.5mm pitch, diam. 8mm max.
 C35 = 2.2 μ F, 50V, 5x11 mm, 2mm pitch

Inductor

L1 = 22 μ H, Micrometals type T130-2 ring core, max. 20x40mm. Wire: 200cm of 1.5mm/AWG15 enameled copper

Semiconductors

D1,D2,D11,D13,D14,D17 = 1N4148
 D3,D8,D9 = 1N4007
 D4,D5,D10 = BZX85C18 SB00018/E1, 18V/1.3W zener diode
 D15,D16 = BZX55C68-TR, 68V/0.5W zener diode (alternatively, NXP type BZX79-C68)
 D6 = MUR120G (200V, 1A)
 D7 = BAV21 (250V, 250mA)
 D12 = LED, green, 3mm
 T1 = IRFD210H-117P
 T2 = TIP122G
 T3,T4,T6,T7,T9,T10,T11,T12 = BC547C
 T5 = BC557C
 T8 = IRFD210PBF
 IC1 = AD8031ANZ
 IC2 = IRS2092PBF
 IC3 = MC7805CTG

IC4 = MC7905ACTG

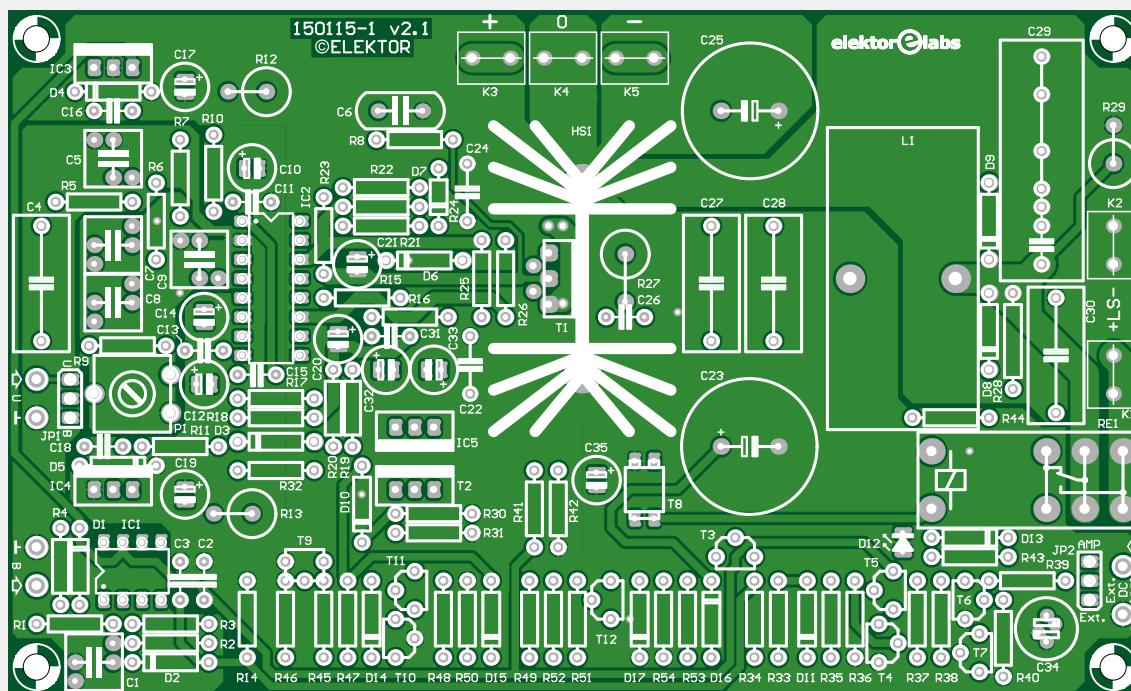
IC5 = MC7812CTG

Miscellaneous

K1,K2,K3,K4,K5 = Faston spade connector, PCB mount, dim. 6.35x0.81 mm, 0.2" pitch
 JP1,JP2 = 3-pin pinheader, 0.1" pitch, with jumpers
 RE1 = relay, PCB mount, SPCO, 16A, 48V/5.52k Ω (TE Connectivity/Schrack type RT314048)
 HS1 = heatsink, PCB mount, 2.6K/W, mounting pins spaced 1 inch (Aavid Thermalloy type 530002B02500G)
 Heatsink for IC4/IC5/T2: 27x90 mm aluminum sheet, 2mm thick
 Heatsink for IC3: 30 x 30 mm aluminum sheet, 1mm or 2mm thick
 Isolation materials for T2,IC3,IC4,IC5 (TO-220 thermal pad + TO-220 bush)
 6 pcs solder pin, 1.3mm diam.
 1 pc DIP-16 IC socket
 1 pc DIP-8 IC socket
 Kit of parts incl. PCB 150115-1 (mono), excl. supply: Elektor Store # 150115-71
 PCB # 150115-1, available separately and optionally

Ancillaries for Hypex SMPS400A400 PSU connection

1 pc JST VHR series 5-way straight case for PCB connector, 3.96mm pitch (RS Components # 820-1188, set of 5 pcs)
 1 x JST VHR series 3-way straight case for PCB connector, 3.96mm pitch (RS Components # 820-1175, set of 5 pcs)
 5 x JST cable set with crimp connectors at both sides (RS Components # 820-1135, set of 5 pcs)
 3 pcs isolated Faston connector, female, 0.25" pitch



speaker system. When T6 or T7 starts conducting due to excessive DC voltage, it drives T5 into conduction, which in turn switches on T4. That transistor discharges capacitor C35, cutting off MOSFET T8 and causing the relay to drop out. At the same time, the IRS2092 is shut down by discharging C14 on the CSD pin through T3, which acts as a level adapter. Diode D11 near T5 is included to prevent T3 from being constantly driven in to conduction through R35.

The relay is driven by a small MOSFET so that the value of capacitor C35 (which together with R41 provides the switch-on delay) can be kept small. That also allows C35 to discharge faster through T4. Header JP2 is provided for testing the DC protection circuit. For normal operation the jumper is placed in the "Amp." position here. If you want to try a different value for R40 in order to change the DC trip level, put the jumper in the "Ext. DC Test" position so you can use an external adjustable DC voltage source.

The maximum continuous drain current of the IRFI4020H-117P dual MOSFET used here is 9.1 A with a case temperature of 25°C. At a temperature of 100°C it is only 5.7 A. For this reason, we added supplementary overvoltage protection to limit the power dissipation at relatively high supply voltages. This amplifier is designed for maximum supply voltages of ± 70 V. The overvoltage protection is dimensioned to react at about ± 75 V. The main components here are the two Zener diodes D15 and D16. Depending on the values of series resistors R52 and R53, transistor T12 starts to conduct when the positive or negative voltage is too high. It drives transistor T5 into conduction (the same as for DC protection), which causes the relay to be de-energized through T4 and T8. Diode D17 is necessary for detection of positive overvoltage. Without D17 the voltage on the emitter of T12 would rise in an overvoltage situation, but that would not cause the transistor to start conducting.

Undervoltage detection is a bit more complicated, and it additionally ensures that

Figure 3. The entire amplifier, including the protection circuitry and all necessary heatsinks, is located on a single PCB. An external heatsink for the output transistors is not necessary.



Figure 4. Two of the heatsinks are home-made from aluminum sheet metal with a thickness of 1 mm or 2 mm, respectively. The small one is for IC3, and the large one is for IC4, IC5 and T2.

the relay is de-energized immediately when the amplifier is switched off in order to prevent popping sounds from the loudspeaker. Here two transistors are used for voltage detection, instead of two Zener diodes. Transistors T10 and T11 both conduct when the two supply voltages are high enough, causing T9 to be cut off. Otherwise T9 is driven into conduction and the relay is de-energized through T5, T4 and T8 as previously described. Diode D14 prevents T11 from conducting when T10 is fully conducting. In that situation the voltage on the collector of T10 is about -0.6 V, and without D14 the emitter of T11 would also be at that level, so it would also start conducting at a very low supply voltage level. That would impair the operation of the undervoltage protection circuit. The detection levels are determined by voltage divider R49/R50 for the positive supply voltage and voltage divider R47/R46 for the negative supply voltage. The values of R49 and R47 also depend on the base current of T11 and the emitter current of T10. With the values shown on the schematic diagram, the undervoltage protection will react when the supply voltages drop below about ± 40 V.

Construction: three heatsinks and a coil

Figure 3 shows the printed circuit board for the D-Watt amplifier. Although it may seem fairly large, all necessary heatsinks are also located on the board, so there is no need to add a bulky external heatsink. Furthermore, all the components are leaded types, so building the board is straightforward and you can easily

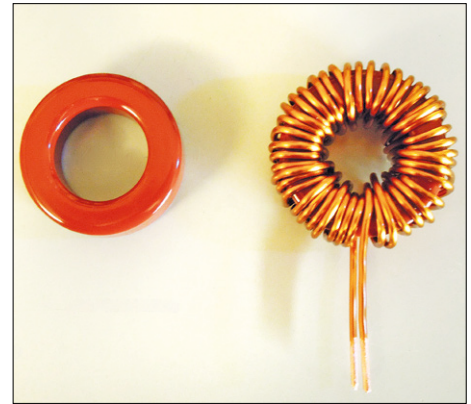


Figure 5. To keep the parasitic capacitance as low as possible, 45 turns of copper wire have to be wound on the core in a specific manner.

experiment with various modifications if you wish.

The bare PCB is available in the Elektor Store, but you can also buy a semi kit (order number 150115-71) containing the PCB and all necessary components, including the heatsinks (note that you will have to drill the holes in the heatsinks yourself).

Not much needs to be said about mounting most of the components; they should not present any difficulties if you have a reasonable level of soldering experience. Do not mount T1 right away.

Use good-quality sockets for IC1 and IC2, preferably with turned pins. It is important to use capacitors of the best possible quality for the PWM modulator circuitry and HF decoupling of the audio input signal (C1, C5, C7, C8 and C9). For that reason we provided lots of room for these components on the PCB, so you can decide for yourself whether you want to mount polystyrene, polypropylene or ordinary polyester types. For HF decoupling of the feedback network (C6), we specifically recommend using a high-quality silver mica capacitor (150 pF / 500 V). Using a different type here is not advisable. For coupling capacitor C4 at the input we used a standard MKT type. Polypropylene is not desirable here due to the large component size, which would cause the capacitor to pick up too much noise (including switching noise).

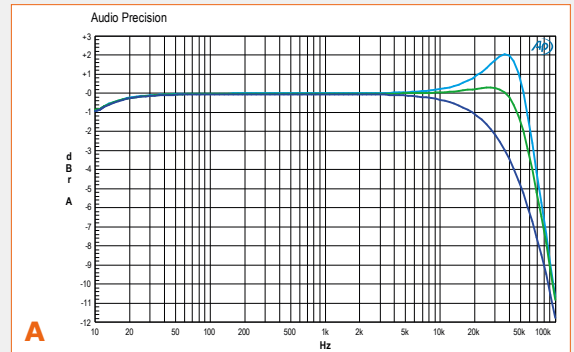
Voltage regulator IC3 only needs a small heatsink, which you can make yourself from a piece of aluminum sheet

Some measured characteristic curves

Test equipment: Audio Precision System Two Cascade Plus 2722 Dual Domain (Elektor Audio Lab)

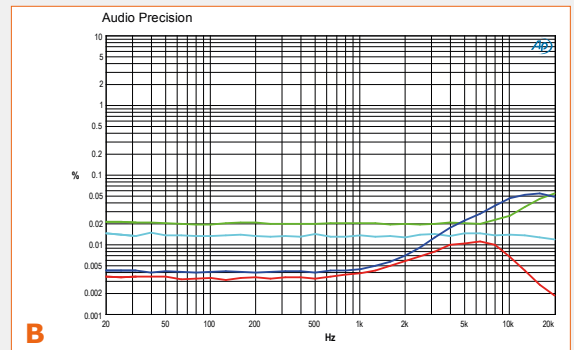
Plot A

This shows the measured amplitude versus frequency with load impedances of 4, 6 and 8 Ω . In the 20 kHz region the curves show that the dimensioning of the output filter is a good compromise for the different impedances. The deviation from 10 to 20 kHz is approximately ± 1 dB. With 8 Ω (cyan) the resonant frequency of the output filter is clearly visible. With 6 Ω (green) the curve is nearly flat, and with 4 Ω (blue) the corner frequency is somewhat lower.



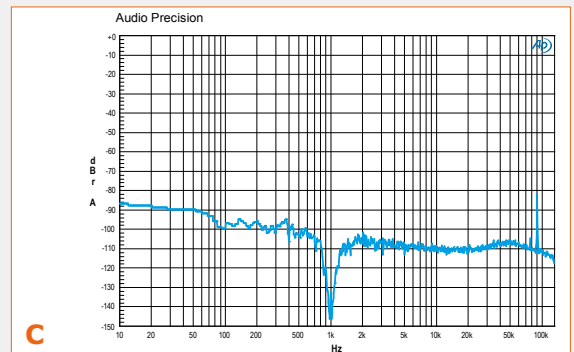
Plot B

THD+N at 1 W and 50 W with an 8 Ω load and measurement bandwidths of 22 kHz and 80 kHz, respectively. Measurements with large bandwidth are usually not shown in data sheets for class-D power amplifiers because the effect of the switching frequency increases with greater bandwidth. However, the amplifier does produce output signals above 22 kHz, so showing what happens above that frequency is only fair, even if you cannot hear it. The red curve shows the THD plus noise at 50 W, measured with a bandwidth of 22 kHz. The blue curve was measured at the same power level but with a measurement bandwidth of 80 kHz. The cyan curve shows the THD plus noise at 1 W with a bandwidth of 22 kHz, and the yellow curve show the THD plus noise at 1 W with a bandwidth of 80 kHz. All measurements were made using a ninth-order elliptical low-pass filter with a corner frequency of 200 kHz ahead of the analyzer input to block residual PWM noise from the amplifier.



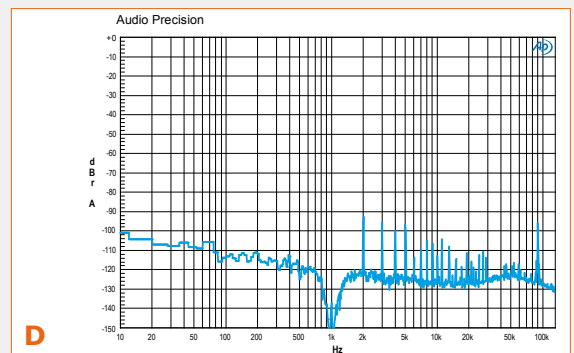
Plot C

FFT of a 1 kHz signal at 1 W into 8 Ω . Hardly any harmonics are visible, although the second harmonic can be seen in the noise of you look closely. The THD+N is 0.014% and consists almost entirely of noise. In this FFT plot and the next one, the switching frequency of the Hypex power supply is clearly visible at 90 kHz (according to the manufacturer's datasheet, it ranges from 80 to 120 kHz), but the level is so low (-82 dB) that it is not a cause for concern.



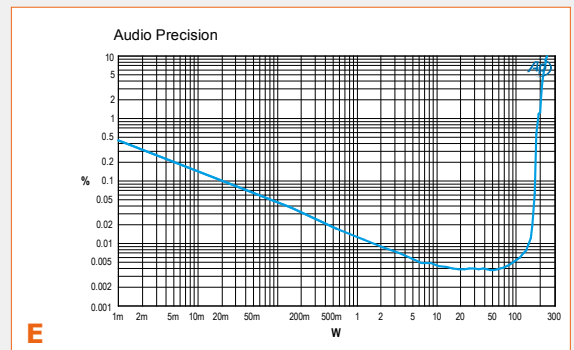
Plot D

The same FFT measurement as plot C, but with a power level of 50 W into 8 Ω . Here some harmonics are visible. The THD+N is 0.004%, and the switching noise from the power supply has dropped to an insignificant level of -96 dB.



Plot E

THD+N versus output power (1 kHz / 8 Ω , BW = 22 kHz). The measurement bandwidth was reduced here to improve the visibility of the distortion curve. The distortion gradually increases above approximately 50 W. The amplifier can comfortably deliver 200 W with the nominal supply voltages.



with a thickness of 1 mm. An area of 30 x 30 mm is sufficient. The metal mounting tab of the IC is connected to ground, so the IC does not need to be insulated from the heatsink. Mount the heatsink plate on the voltage regulator so it is a few millimeters above the surface of the board. Also make sure that the mounting hole in the corner of the board remains accessible (see **Figure 4**). IC4, IC5 and T2 are all mounted on an aluminum plate 2 mm thick with dimensions of 27 x 90 mm, with one end bent at a right angle 65 mm from the end (see Figure 4 again). To avoid contact with other components, mount the heatsink on these three components so it is at least 5 mm above the surface of the board. All three components must be **insulated**, and two of them (IC5 and T2) are mounted opposite each other and secured by the same screw. The two holes in the plate are spaced

First check that the leads are properly positioned in the holes without twisting, and then tighten the screw firmly. After this, solder the MOSFET leads to the PCB.

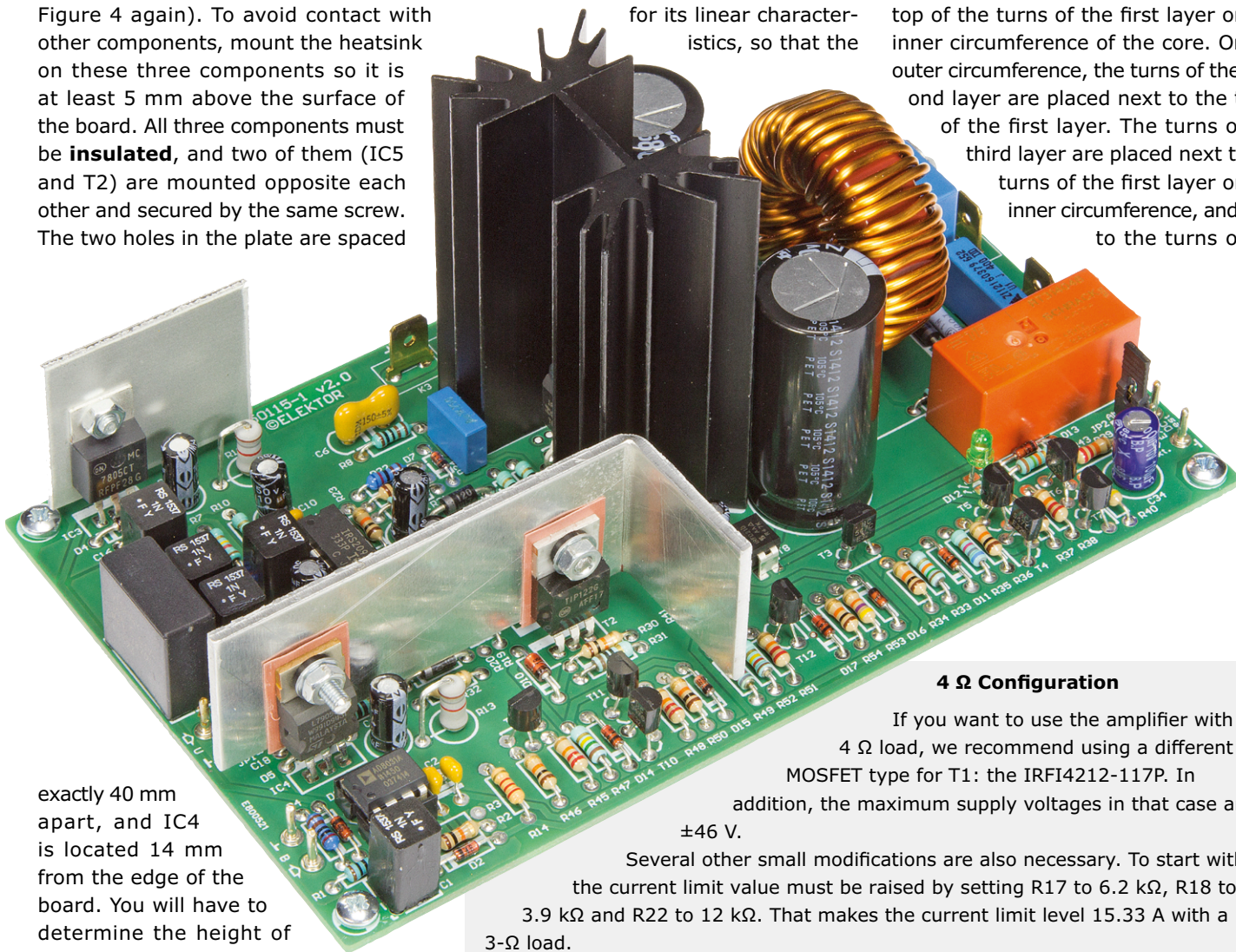
Output inductor

The most difficult component in this project (in terms of construction) is the inductor of the output filter. You have to wind it by hand on a toroidal core with a diameter of 33 mm (1.3 inch) using 1.5 mm enameled copper wire (15 AWG). The core is an iron powder type from Micro-

metals. We chose this core for its linear characteristics, so that the

filter does not cause much distortion. The downside of this core material is its low permeability, which means a lot of turns have to be wrapped on the core to obtain the necessary inductance.

You need 2 meters (approx. 7 feet) of 1.5 mm enameled copper wire (15 AWG) for the winding, and you have to wind 45 turns on the core. They will not fit in a single layer on the inner circumference of the core, so the coil must be wound in a special way to keep the parasitic capacitance low. The turns of the second layer of the winding are placed on top of the turns of the first layer on the inner circumference of the core. On the outer circumference, the turns of the second layer are placed next to the turns of the first layer. The turns of the third layer are placed next to the turns of the first layer on the inner circumference, and next to the turns of the



exactly 40 mm apart, and IC4 is located 14 mm from the edge of the board. You will have to determine the height of these two holes by empirical measurement.

Next we have the heatsink for T1. The package of this dual MOSFET is completely composed of plastic, so it does not need to be insulated. First mount the heatsink on the PCB, then apply a thin coat of thermal grease to the rear of T1 and insert its leads into the corresponding holes in the board. Secure T1 to the heatsink with a screw, plain washer, split washer and nut (screw length 12 mm).

4 Ω Configuration

If you want to use the amplifier with a 4 Ω load, we recommend using a different MOSFET type for T1: the IRFI4212-117P. In addition, the maximum supply voltages in that case are ± 46 V.

Several other small modifications are also necessary. To start with, the current limit value must be raised by setting R17 to 6.2 kΩ, R18 to 3.9 kΩ and R22 to 12 kΩ. That makes the current limit level 15.33 A with a 3-Ω load.

Now the 48 V relay can be connected directly to the power source, with R44 replaced by a wire link. The feedback network must also be modified by reducing the value of R7 to 75 kΩ.

At continuous full output power with a 3-Ω load, current limit protection will trip after a few seconds because the temperature of the main heatsink will rise too much, despite the high efficiency. With this configuration it can therefore be a good idea to make a larger heatsink yourself (no higher version is available for the type specified in the components list).

Extensive measurement data for the 4-Ω configuration is available on the Elektor Labs website [4]. Among other things, it shows that the amplifier can deliver 243 W into 4 Ω with 1% distortion (and even 297 W into 3 Ω).

second layer on the outer circumference. Repeat this procedure for the fourth layer. After you have wound 45 turns on the core this way, the ends of the copper wire will be located opposite each other on either side of the core. You may have to improvise or make a few adjustments to get all 45 turns on the core this way, but hopefully you understand what we mean. Have a look at the picture of the finished coil in **Figure 5**. We should warn you that 1.5 mm copper wire (15 AWG) is fairly stiff and not easy to bend. Work carefully and wind the wire as tightly as possible against the core, but do not use any metallic tools due to the risk of damaging the insulation of the wire.

When you are done, you will have a coil with an inductance of 22 μH and a resistance of about 20 m Ω . Now you can mount it vertically on the

PCB and solder it in place. That completes the assembly of the circuit board.

Power supply and enclosure

Next you have to fit the board and its power supply into a suitable enclosure. Although it is possible to use a conventional power supply with a hefty power transformer, bridge rectifier and filter capacitors, that doesn't really fit with a modern class-D amplifier. Furthermore, the measured performance of the D-Watt Amp is significantly better with a regulated power supply. For that reason, we used switching power supplies from the Dutch company Hypex in our prototype. They are specifically designed for use with class-D power amplifiers. We chose the SMPS400A400, which can supply 400 W at 20 Hz with nominal output voltages of $\pm 64\text{ V}$ (the mains voltage in our lab is a bit lower than

usual, so we measured output voltages of $\pm 60\text{ V}$). With this power supply the amplifier is suitable for loads from 6 to 8 Ω . For lower load impedances it is advisable to modify the amplifier (see the inset), as otherwise the protection circuit will trip occasionally during high-power passages.

In **Figure 6** you can see how we fitted two amplifier boards and two Hypex power supplies into a standard enclosure. There is not much wiring in the enclosure because everything is located on a single PCB.

We hope you enjoy building and using this cool high-power amplifier. More detailed information about the amplifier, along with even more measurement data, is available on the Elektor Labs website [4]. ◀

(150115-I)

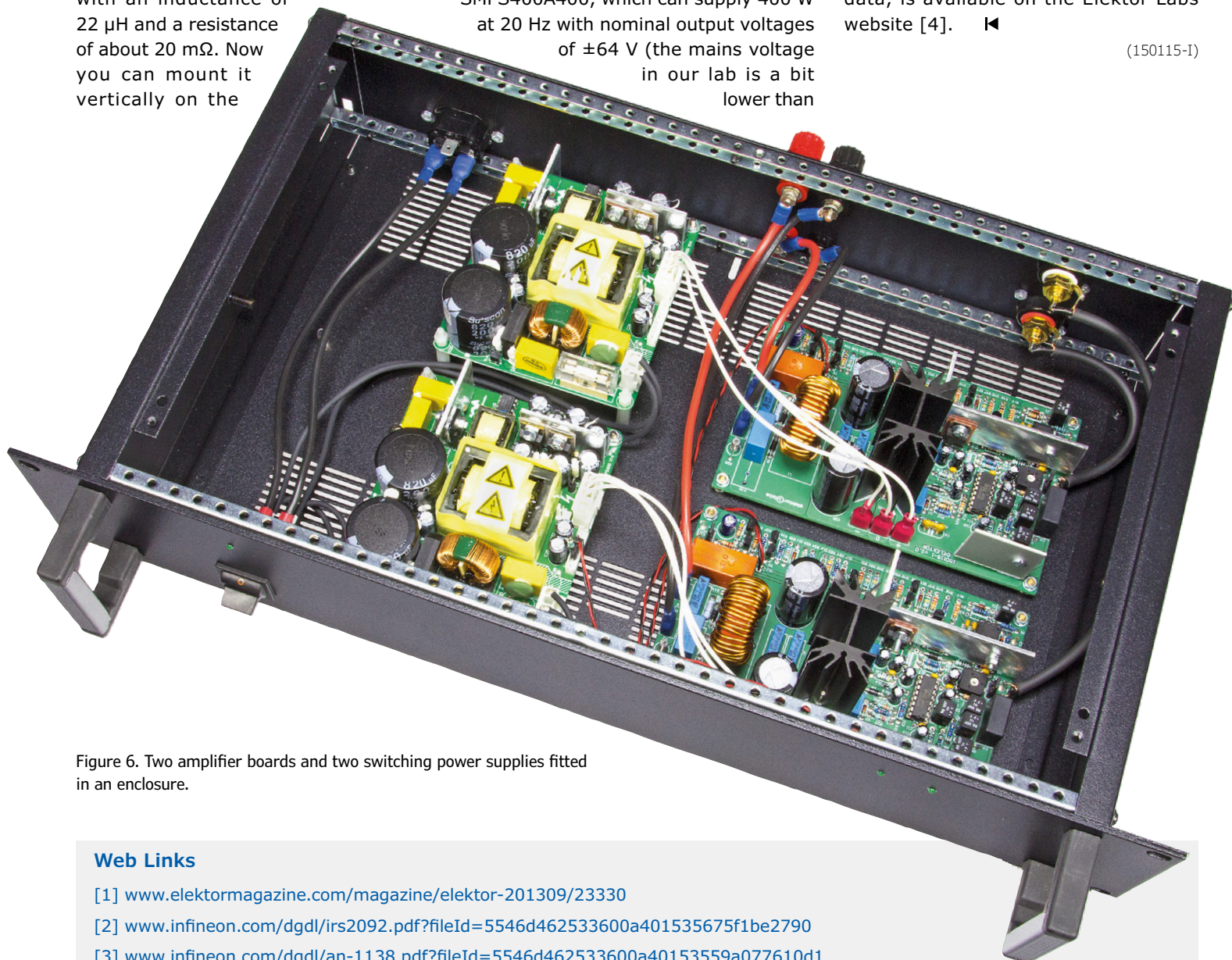
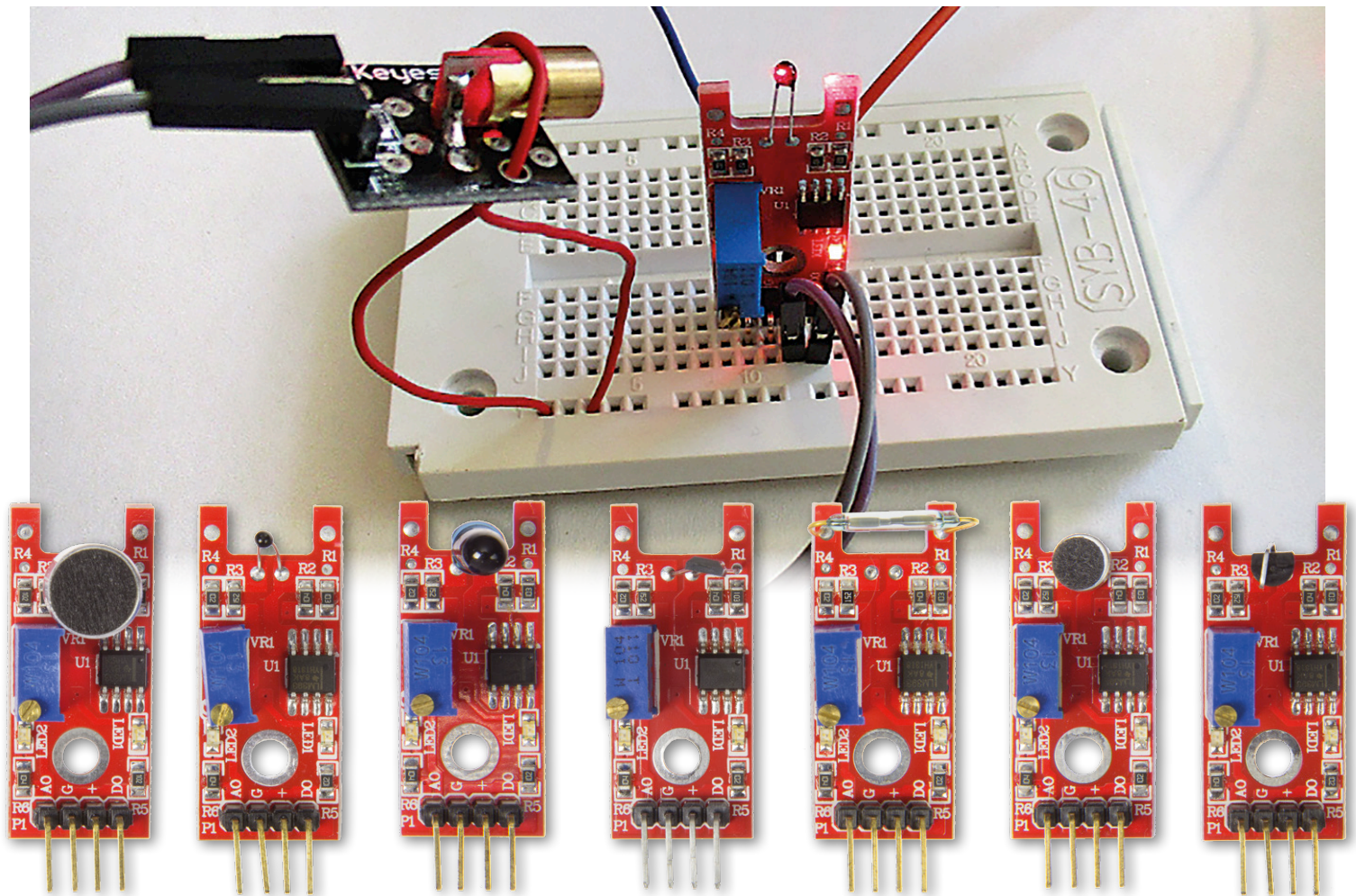


Figure 6. Two amplifier boards and two switching power supplies fitted in an enclosure.

Web Links

- [1] www.elektormagazine.com/magazine/elektor-201309/23330
- [2] www.infineon.com/dgdl/irs2092.pdf?fileId=5546d462533600a401535675f1be2790
- [3] www.infineon.com/dgdl/an-1138.pdf?fileId=5546d462533600a40153559a077610d1
- [4] www.elektormagazine.com/labs/200w-class-d-audio-power-amplifier-150511



Sensors make Sense (2)

For Arduino and more

Sensors are either analog or digital. Analog values are read in via an A-D input, whereas for digital signals a simple Port connection frequently adequate. In some cases, however, sensors have dual outputs, one analog and one digital.

By **Burkhard Kainka** (Germany)

A glance at the sensor combo kit (available from Elektor [1]) reveals seven superficially identical PCBs equipped with differing sensors. All of them have one analog output AO and one digital output DO. As you know, to create a digital signal from an analog one, we use a comparator. There is one of these included on each of these boards, or more precisely an LM393 dual differential comparator.

Sensors equipped with comparators

When we examine the circuit of the board in **Figure 1** more closely, its function becomes clear. Once more we find the

actual sensor in a voltage divider that this time is preset with a 25-turn precision trimmer pot. A second voltage divider made up from two 100 k Ω resistors provides a reference voltage of 2.5 V for the comparator. The temperature sensor (*Digital Temp*) can now be adjusted so that the sensor voltage at the desired temperature will likewise be exactly 2.5 V. A rising temperature at the NTC sensor makes the comparator switch on the digital output DO, whilst a falling temperature switches it off. The second comparator connected downstream operates only the status LED. Thanks to these LEDs we can test all seven sensors without any need for software.

The circuit does not include a bypass capacitor (see **inset**), meaning that any variations or spikes on the supply voltage

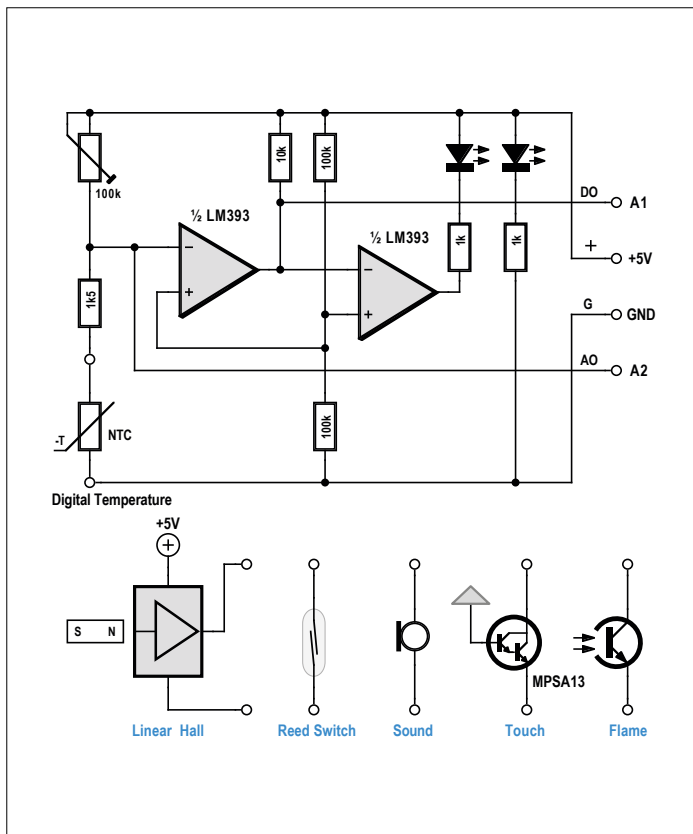


Figure 1. Sensors with comparators.

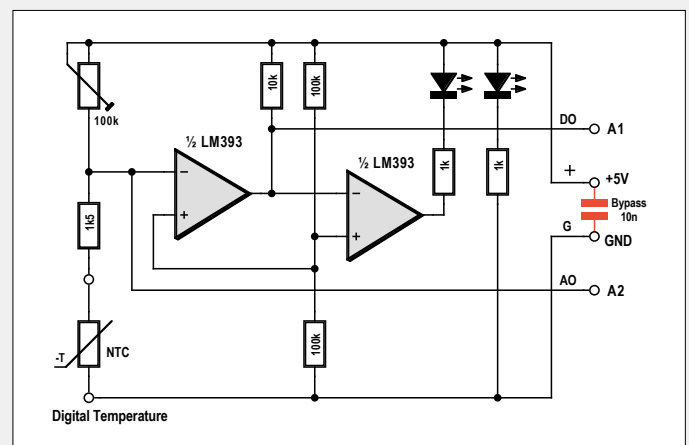
Bypass capacitors

In many circuits we find capacitors wired between the supply voltage (V_{CC}) and ground (GND). They improve the stability of the circuitry and help prevent radio interference. Whenever electronic circuitry uses a lengthy cable to the power supply it's not just the resistance of the wires in the cable that comes into play but also the inductivity. A piece of twin-conductor cable 1 m long can have an inductance of around $0.5 \mu\text{H}$, according to the gauge (thickness) of the wires and the distance between them. From this arises an inductive resistance of 3Ω at 1 MHz or 30Ω at 10 MHz. If your circuit has a varying current load, effectively you have an alternating current in the supply lead and with this some voltage drop. Typical problems with voltage drop of this kind are susceptibility to radio interference and vulnerability to failures in the passive elements of the circuit. The long (from a radio perspective) cable can also act as an antenna. Then you have radio-frequency (RF) signals radiating that potentially may exceed the permitted limits. Conversely pulses of interference may induce brief fluctuations in the supply voltage that might interfere with the correct functioning of a circuit.

DC motors will produce a varying load on the current source and can lead to radio interference. This is why capacitors are connected direct to the supply terminals and are generally called anti-interference or suppressor capacitors. The DC motor in an RC model cannot function without these, as otherwise it would interfere with its own receiver onboard. Microcontrollers like the Arduino also require a capacitor

may affect the operation. The length of the wiring and other random factors can also have some influence. In actual fact with the temperature sensor and the heat rising, we find a narrow region in which the output oscillates, which is possibly caused by parasitic capacitance on the PCB. If we fit a small capacitor between the input and the second comparator output, we can see an oscillator in action. This is easy to recognize with the oscilloscope. The status LED on the sensor PCB shows it as well. When things warm up slowly, it lights up initially at half brightness (state of oscillation) and only after this does it reach full brilliance (stable condition). If the software behaves strangely, bear this point in mind.

The other sensors using this circuit behave in more or less the same way, even if the changes in the measured value are usually not so gradual. Most comparable with the NTC sensor is the phototransistor in the flame sensor. The dark housing allows the longer wavelengths to pass through preferentially, in order for flames to be detected. The Hall sensor differs in that it expects an operating voltage on its third pin. The fact that the reed switch is connected as an analog sensor may surprise you. But there are two out-of-phase outputs. When a magnet is brought closer, one output goes on and the other goes off. Sound and touch behave somewhat differently; normally they deliver a squarewave signal with rapid transitions. The two sound sensors employ one large and one small electret microphone. With the pot adjusted correctly, half waves of loud sound signals appear at the output as square waves. This must be



between GND and V_{CC} , as otherwise they would not pass their test for electromagnetic compatibility. In this case they are usually called decoupling capacitors, because any radio-frequency currents can be diverted or decoupled by taking a shortcut through this capacitor mostly commonly of 100 nF . The older expression 'blocking capacitor' used for these components has fallen out of fashion.

Our sensors are all definitely not particularly susceptible to interference problems of this kind. If you do wish to employ a cable longer than 12 inches (30 cms), you should connect a bypass capacitor of 100 nF direct to the sensor between the supply terminals (see [image](#)).

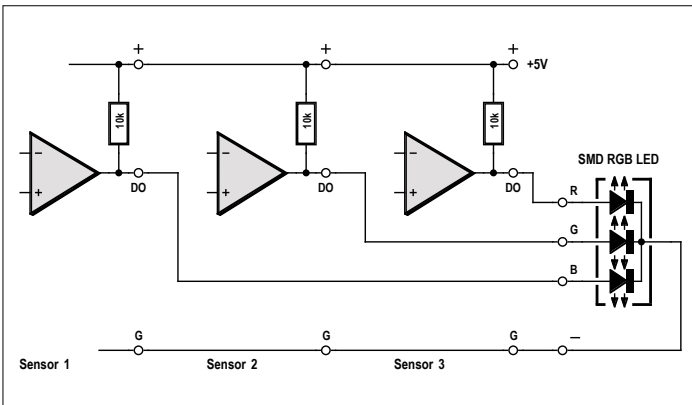


Figure 2. Direct connection to RGB LED.

taken into account during your evaluation testing. In the same way the touch sensor generally provides a 50 Hz squarewave signal (60 Hz in North America) when activated.

The comparator circuit delivers a sharp switchover point, without any hysteresis. This means that with a slow change of temperature, a region can occur in which the output flutters a bit. In a program you can take account of this and poll the digital output only at longer intervals. Or you could use the analog output signal AO direct from the voltage divider and evaluate this, something that we'll describe in greater detail below. Unlike the analog NTC sensor already described, you will then have the advantage that a desired temperature can be set externally using the pot.

It's also interesting to note that the digital output of this board can control an actuator directly, without any assistance from a microcontroller. Output DO is connected to the input of the relay PCB. That's it — the temperature controller or thermostat is complete. Admittedly without hysteresis, which in some applications could be problematic. If you feel like connecting one of the LEDs directly to the comparator, you need to bear in mind that a comparator has an open collector output. Therefore it switches down actively, so that in the High state only the pull-up of 10 kΩ supplies current, meaning that the LED does not illuminate very brightly. Nevertheless, applications are conceivable in which three separate sensors directly drive the three colors of an RGB LED (**Figure 2**), not with great intensity but clearly visible nevertheless. The resulting color mixture of pink perhaps then stands for "hot and loud".

If you wish to control laser lights directly, you'll need to hook these up between DO and +5 V (**Figure 3**). On/Off reverses the situation, but you can now switch larger currents up to about 20 mA. With this arrangement an interesting experiment can be carried out: opto-thermal feedback. The laser is pointed precisely towards the NTC sensor (see header photograph). The switching temperature is then set exactly at the change-over point using the potentiometer. Once the correct position is found, the laser starts to flash. Here's how: when switched on, it warms the sensor slightly. The digital output switches on, which turns off the laser because of the inverted connection. This gives the sensor the chance to cool down again until the comparator flips and the sequence begins once more.

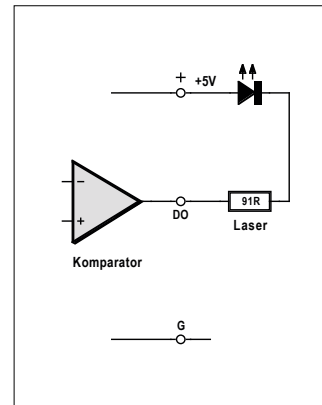


Figure 3. The laser on the comparator output.

Software-based Schmitt trigger

When you have a transition region a comparator can generate highly volatile output states. A Schmitt trigger helps mitigate this by displacing the transition points that set the switch-on and switch-off points. An example is switching on at 25 degrees Celsius and off at 20 C, as discussed regarding the NTC sensor (*Analog Temp*) in first part of this series.

All the sensors on the comparator board also have an analog output, as you know. There is nothing to stop you making a comparison in the software, and in so doing, building in some appropriate hysteresis. Because all the sensors are adjusted using the trimpot to a switching point of 2.5 V, the same software can be used for the differing sensors.

Listing 1. A comparator with hysteresis (Comparator.bas).

```
Do
  D = Getadc(2)
  If D > 514 Then Portb.2 = 0
  If D > 514 Then Portb.5 = 1
  If D < 510 Then Portb.2 = 1
  If D < 510 Then Portb.5 = 0
  ...
  Waitms 500
Loop
```

Listing 2. The Arduino comparator.

```
//Comparator AD1
...
void loop() {
  value = analogRead(sensorPin);
  if (value > 514) {
    digitalWrite (output1, 1);
    digitalWrite (output2, 0);
  }
  if (value < 510) {
    digitalWrite (output1, 0);
    digitalWrite (output2, 1);
  }
  Serial.println(value);
  lcd.setCursor(0, 0);
  lcd.print(value);
  lcd.print (" ");
  delay(50);
}
```

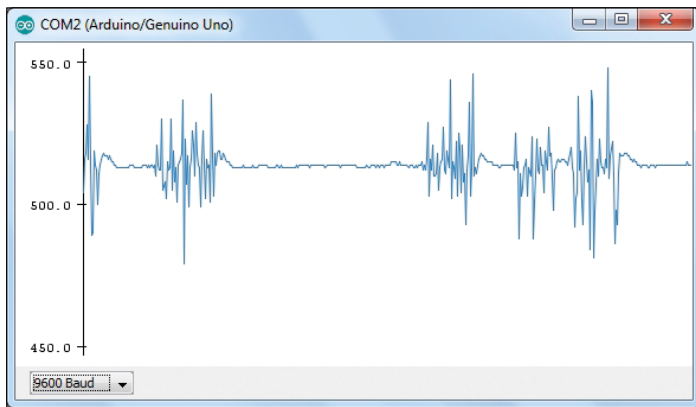


Figure 4. Microphone signals.

In the software the two outputs B.2 (LED 2 on the Elektor Extension Shield, see Part 1 [2]) and B.5 (LED on the Arduino board) are used as outputs. There's a specific reason for switching them in antiphase. You might connect the dual-color LED to both outputs. Or you might hook up actuators of this kind between both outputs, which in reality still require a series (dropper) resistor but in this situation would at least see two internal resistances in series. This class of actuators also includes the SMD RGB LED and the infrared LED. The color-changing LED discussed later on belongs in this category too, but it includes a reverse-voltage protection diode, which will not tolerate the inverted voltage level cheerfully. In this case you are better off using a genuine 'real' dropper resistor.

The sample Bascom code in **Listing 1** evaluates the raw data

from the A-D converter (as always, all of the code samples can be downloaded from the Elektor Magazine website [3]). The center of the measurement range represents a value of 512. The switching threshold points in this example lie at 509 and 515. With around 5 mV per A-D step we then have a hysteresis of 30 mV. The program also controls LED2 on the Extension Shield. This means you can compare the switchover with that on the sensor board yourself. The software offers two slightly displaced switchover points as opposed to the sharply defined switching point of the sensor-comparator. The complete program additionally displays the analog sensor voltage, both on the LCD of the Shield and on the terminal screen. This helps you find the correct trimmer setting.

The Arduino version of the program (**Listing 2**) differs little from the BASIC example. The analog signals can be displayed without additional overheads using a serial plotter. **Figure 4** shows signals from the microphone sensor. This program can be used again for all seven sensors on the red comparator PCB.

Polling contact sensors

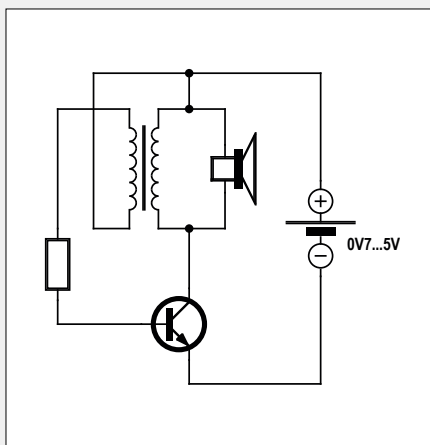
The touch sensor does not produce a slowly varying signal, but rather a (normally) squarewave signal of 50 or 60 Hz. Were you to use this for controlling a relay, the result would be wavering, noisy and inelegant. However, the signal can be improved in software (**Listing 3** and **Listing 4**). The digital output in this case is connected to AD1. Yes, this is an analog input but you can also use it as a digital input Port. The result of polling it will be 1 or 0 and can be copied direct to an output Port. Here we chose B2 (Arduino pin 10), because it can also control the LED2 on the Shield. Additionally B5 is also controlled, as that's

Oscillators

One of Murphy's Laws states that when you construct an oscillator it never oscillates, whereas when you make an amplifier it oscillates all the time. There's something in this observation, which is why it's always worth looking closely when your circuit turns into an oscillator and 'starts to hoot' as people say. An oscillator consists fundamentally of an amplifier and some matching feedback from the output to the input. The signal fed back must moreover have the correct phase relationship. If the voltage at the input rises directly, the same should happen at the output, only correspondingly amplified. Then it's sufficient to feed back a part of the output signal to the input using a capacitor — and now you have an oscillator.

An amplifier of this kind can be constructed with two transistors, in which both of them rotate the phase by 180 degrees. Or you can use an op-amp or an integrated loudspeaker amplifier.

A single amplifier stage with one transistor in grounded emitter mode turns the phase through 180 degrees. Rising



input voltage will make the output voltage drop. All the same, you can build an oscillator with only one transistor. All you need do is ensure that the phase is reversed accordingly. You can do this, for instance, using several capacitors and resistors (phase-shift oscillator). Or use a transformer for the feedback (Meissner oscillator). If the oscillator still observes Murphy's Law and doesn't oscillate, all you normally need to do is reverse one of the two windings to get the phase right. So it's likely that the small buzzer is constructed in this way (**image**). The 'loudspeaker' has two windings, which at

the same time make a transformer.

If you construct an amplifier with multiple stages and higher overall gain, it may actually be not that easy to prevent self-oscillation. Signals from the output can sneak back to the input via the supply voltage, which you may be able to avoid using a fairly large bypass capacitor. Or some small capacity may exist between the output and input wiring. In extreme cases the only remedy that can still solve this problem is a physical screening (shielding) plate.

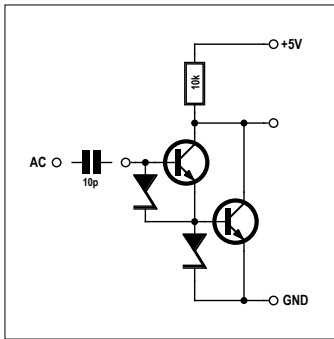


Figure 5. Contact sensor with Darlington transistor.

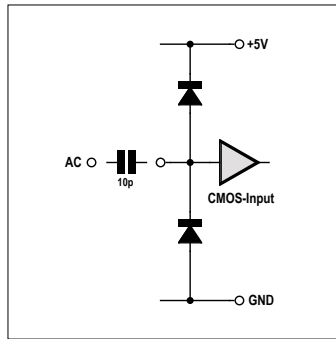


Figure 6. Input Port used as contact sensor.

where the Arduino-internal LED is located. Externally you can also hook up another LED or the laser light.

The program consists of a simple loop that has a waiting time of 21 ms built in. If you now tap the touch sensor, the output flashes. What actually happens is that touching it applies a 50 Hz interference signal to the base of the Darlington transistor. Therefore we get a 50 Hz squarewave signal on the digital output, i.e. a cycle duration of 20 ms. If we now sample this signal using a slightly different frequency, you obtain a significantly smaller frequency. The cycle duration 21 ms indicates a sampling frequency of around 48 Hz. The difference 50 Hz – 48 Hz = 2 Hz appears in antiphase at outputs B2 and B5.

It may not be self-evident that we can control a transistor in this manner. Usually we specify a base current and determine the bias point in this way. With small signals, things would not work without base current. The person touching the sensor is seen as one plate of a capacitor, with the second plate formed by all the wires and cables located in the environment. A base current would charge this capacitor so far negative that the transistor would block completely. It works only because every bipolar transistor contains a zener diode in the range 7 V to 10 V between base and emitter (see **Figure 5**). The idle state mains hum voltage of a person is generally significantly higher than 10 V_{pp}. Therefore an alternating current flows, with its positive half-wave driving the transistor.

You might expect that on the collector of the Darlington transistor we would also find an approximately equal squarewave signal. So let's also try coupling the analog output AO to the digital input A1. That's right; now the output flashes when the base is touched. However, the difference is that this time the output in idle mode is 'on', because the comparator on the sensor board inverts the signal.

It even works without the sensor board. At input A1 we have just a piece of insulated wire connected (**Figure 6**). If you then touch the outside of this insulation, it still flashes! This happens because the input of the controller has extremely high impedance. The wire, insulation and finger form a small coupling capacitor of a few picofarads in value. If an AC voltage is applied to the input, it is limited to the input voltage range by the internal protection diodes. The open input does admittedly have a disadvantage compared to the correct touch sensor. The idle state is not unambiguous, you see, which makes evaluation tricky.

Processing switching signals

Up to now all we have done with our touch sensors is achieve flashing on the output. In reality we'd expect a contact switch to do rather more, namely switching something on. This is entirely feasible if we write our program somewhat differently. To manage this, all we need do is make sure that the switching-on process takes precedence and switching-off takes place after a short delay. Our *Touch* programs (**Listing 5** and **Listing 6**) additionally employ a delay period of 10 μs and consequently interrogate the Port very frequently. Once a High state

Listing 3. Input and output Port in Bascom.

```
Dim D As Boolean
Config Portb = Output

Do
  Portb.2 = Pinc.1
  D = Pinc.1 Xor 1
  Portb.5 = D
  Waitms 21
Loop
```

Listing 4. Input and output Port using Arduino-C.

```
//Touch1 A2 > 10, 13
#include <LiquidCrystal.h>
int input = A2;
int output1 = 10;
int output2 = 13;

void setup() {
  pinMode(output1, OUTPUT);
  pinMode(output2, OUTPUT);
}

void loop() {
  digitalWrite (output1, digitalRead(input));
  digitalWrite (output2, 1-digitalRead(input));
  delay(21);
}
```

Listing 5. Contact sensor and touch switch (Touch2.bas).

```
Dim T As Word

Config Portb = Output

Do
  If Pinc.1 = 1 Then T = 50000
  'If Pinc.1 = 0 Then T = 50000
  If T > 0 Then T = T - 1
  If T > 0 Then Portb.2 = 1 Else Portb.2 = 0
  If T > 0 Then Portb.5 = 0 Else Portb.5 = 1
  Waitus 10
Loop
```

is detected, the program sets counter T to a value of 50000 and switches on the output. The counter decrements slowly and only after about 500 ms without any further pulse does the output switch off. Any further impulse that occurs will reset the counter High again and the timeout period is extended accordingly. Now everything functions as desired. The output goes on immediately if someone touches the sensor and goes off again, not when the contact ceases but after some delay. The function corresponds to a retriggerable monostable multivibrator. Now our touch sensor, with its Darlington transistor, is working the way we want it to. The squarewave signal on the digital output has been transformed into an unambiguous switching signal that will work even in environments without mains hum. Short pulses caused by static charge are sufficient to trigger the output.

There's a special reason why we chose the loop delay of 10 μ s to be much smaller than necessary for a 50 Hz signal. The format of the program makes it equally suitable for use with the

Listing 6. Touch switch in Arduino-C.

```
void loop() {
  if (digitalRead(input) == 1) timeout = 50000;
  //if (digitalRead(input) == 0) timeout = 50000;
  if (timeout > 0) timeout = timeout -1;
  if (timeout > 0) {
    digitalWrite (output1 , 1);
    digitalWrite (output2 , 0);
  }
  else {
    digitalWrite (output1 , 0);
    digitalWrite (output2 , 1);
  }
  delayMicroseconds(10);
}
```

Listing 7. Additional serial output.

```
void loop() {
  if (digitalRead(input) == 1) timeout = 50;
  if (timeout > 0) timeout = timeout -1;
  if (timeout > 0) {
    digitalWrite (output1 , 1);
    digitalWrite (output2 , 0);
  }
  else {
    digitalWrite (output1 , 0);
    digitalWrite (output2 , 1);
  }
  Serial.
    println(analogRead(sensorPin)+100*digitalRead
      (output1));
  delay (20);
}
```

two sound sensors in the Sensor Kit, with which, according to the sound frequency being sampled, up to 10 kHz or more can appear on the output. Correct adjustment of the trimpot will give you a very usable speech switch that will also react to loud whistling, hooting or other noises. Even gentle tapping your finger on the microphone will set off the switch. The sensitivity depends very much on the precise setting of the pot. You need to keep in mind that this type of microphone normally delivers less than 1 mV. This means you must maintain the sensor voltage to the switching point within a few millivolts of accuracy in order for sound signals to make the comparator change level correctly.

Shock sensor

Certain other digital sensors, such as the *Tap Module*, are appropriate for this kind of sampling. This contains a spring that can close a contact when disturbed. Two very brief pulses are produced in this process. You need to be aware that the module has a pull-up resistor, meaning that the output voltage is +5 V when not operated and the pulses are 0 V. In this situation the sampling process needs to invert the input signal, which is marked in the software with a commented line. Alternatively in this case you might also connect +5 V and GND transposed and leave the software unaltered, as the polarity is of no interest to the switch and resistor.

The *Shock Sensor* was tested in the same way, in which an internal contact is closed for a short duration each time it is disturbed. The sensitivity of the Shock Sensor is greater than that of the Tap Module. The *Tilt Switch* (ball switch) can also be sampled in this manner. At the right degree of inclination a small ball rolls downwards and closes two contacts. If you shake the sensor rapidly to and fro, you can hear the ball rattling and watch the program triggering the output.

In point of fact all switching sensors can be polled or sampled meaningfully using this method, as well as press buttons (*Button*), reed switches (*Mini Switch*) and the reed switch on the comparator board. Using this type of time delay we can achieve effective debouncing of the switches. Just about every mechanical switch (apart from mercury switches, which are now banned in many territories, however, on account of the poisonous mercury in them) bounces momentarily one or more times when operated, generating several impulses to begin with. The software turns this into a single, elongated pulse. If you alter the period somewhat you can make a time switch, for instance to illuminate a staircase at night. The light can then be activated, according to the type of sensor, by loud speech, touch, tapping or advancing a magnet. Even the digital output of the temperature sensor could be polled in this way. This would simultaneously eliminate the problem of fluttering transitional states between Low and High.

With a minor alteration (**Listing 7**) the function of a comparator sensor could also be made clearer using the serial monitor. For this we additionally make use of the analog signal at AD2, while the digital output signal of the comparator carries on with the task of triggering the actual switching process. To display the output signal with only one channel, we raise the output in a High state by 100. You can then observe the original analog

signal as if raised on a pedestal. **Figure 7** shows the result for the sound sensor. You can see clearly that a sound signal must first exceed a certain level to switch the output state. At the end of all of the signals the software enables the laid down delay period, after which the output drops back down. **Figure 8** portrays the work of the temperature sensor. For this the program is slowed down to an extent by a delay of 100 ms. During the measurement time-window the sensor was twice warmed by touching with a finger. You can spot clearly the inversion performed by the comparator: a falling voltage on the actual sensor switches on the comparator's output. Equally easy to recognize is a brief phase of oscillation in the region around the switchover point. The software suppresses these oscillations effectively.

Buzzers and other actuators

At output B2 we can again attach the relay board. But there are yet more actuators that we can also put to use. Among these belongs the *buzzer*, a small active device that is provided initially with some protective foil. At 5 V the current consumption amounts to 25 mA. The buzzer can therefore be attached direct to a Port. The polarity is admittedly not completely unambiguous and depends on how the buzzer is soldered onto the board. On the sample device the signal and minus connections were transposed in fact and the S-pin need to be connected to GND. A test using the lab power supply also indicated that the buzzer begins to work already at 0.7 V and at a different frequency if the sound opening is blocked. Without even opening the housing, our experts were able to figure what lay inside: a free-running oscillator with a bipolar silicon transistor with the normal threshold voltage of 0.5 V to 0.7 V (naturally it insists on having the correct polarity of supply voltage). At the same time take care that you do not confuse the active buzzer with the passive one, which is more comparable to a small 16- Ω loudspeaker.

Another interesting actuator is the color-changing LED (*Color Flash*), in actual fact a three-color LED package with a built-in controller. This automatic LED produces a color transition between red, green and blue. This is another case requiring a dropper resistor, since color-change LEDs of this kind are in fact designed for 3 V. Incidentally, although there is indeed a resistor of 10 k Ω on the board, it is connected in parallel (**Figure 9**). For 5 V a dropper resistor of 100 Ω would work best. Here again, however, the compromise presented in the previous article works again: you can connect the LED between two Port pins, in order to achieve a degree of current limitation using the internal resistance of the Ports. Pin 12 (B4) is recommended for the opposite pole.

Incidentally, the internal controller of the color-change LED, like most ICs, includes an inverse polarity-protection diode on its supply voltage connections and therefore reacts sulkily if you reverse the supply connections. Without the dropper resistor a reversed power connection could lead to destruction. This is vitally important: GND lies in the middle of the three connections. By the way, just for fun, you can also connect the passive mini-speaker in series. Then you can not only see the switching process but also hear it. ◀

(160173)

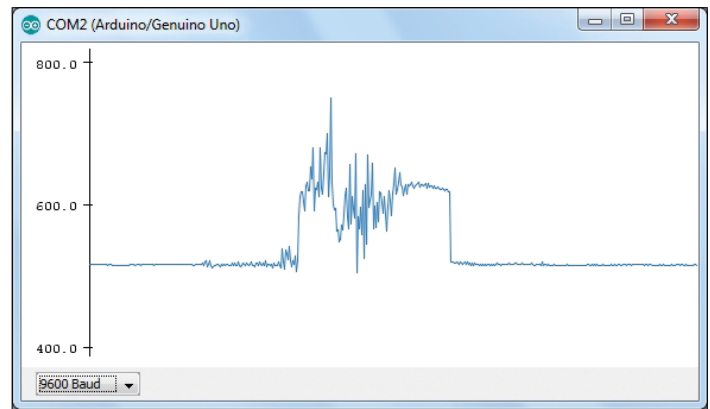


Figure 7. Switching using the sound sensor.

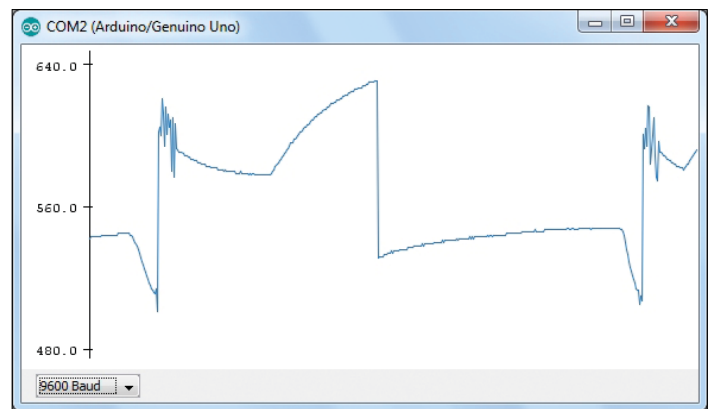


Figure 8. Temperature sensor in action.

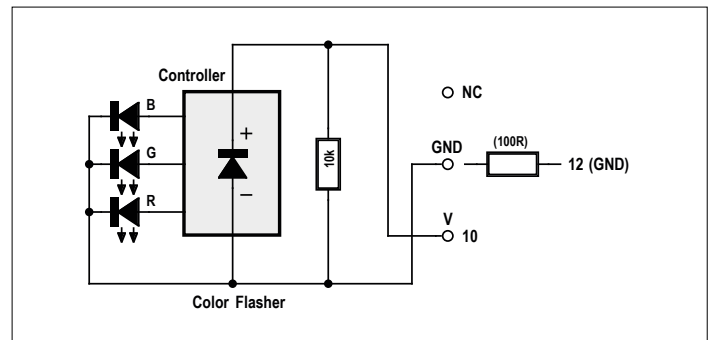


Figure 9. Color-changing LED.

Web Links

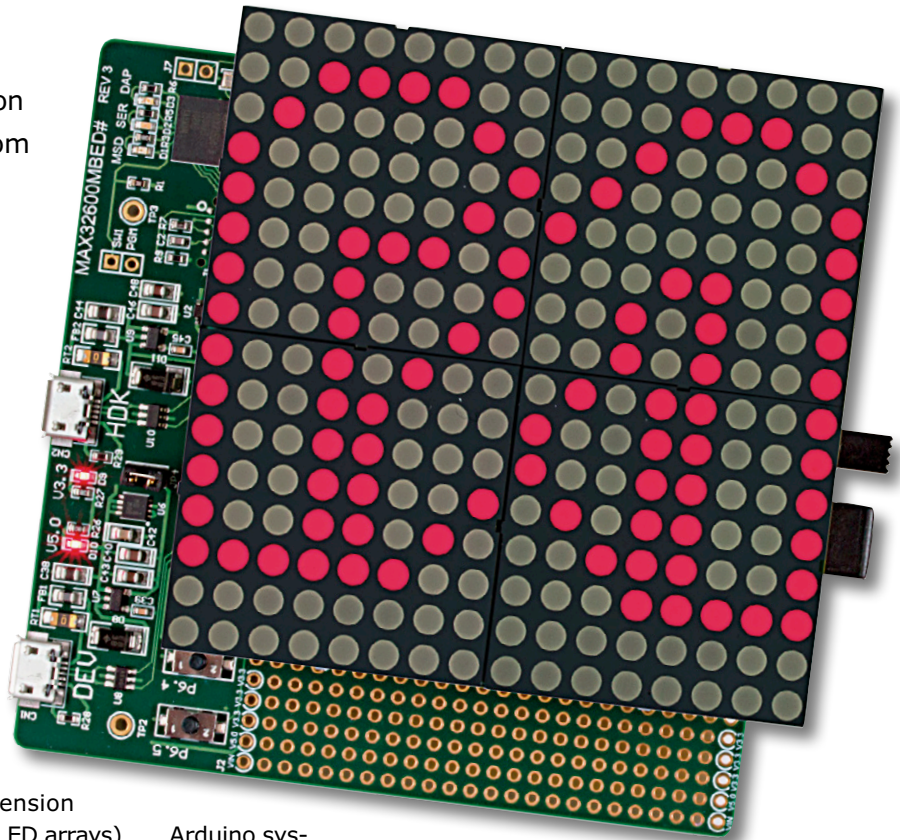
- [1] www.elektor.com/arduino-sensor-kit
- [2] www.elektormagazine.com/160152
- [3] www.elektormagazine.com/160173

MAXREFDES99# Display Driver Shield

256 LEDs at your command

By **Clemens Valens** (Elektor Labs)

The American Standard Code for Information Interchange, first published in 1963 and from then on known as ASCII, was based on the English alphabet whose characters can be displayed nicely with as few as 5x7 pixels or even less. Today, electronic signs must show anything from Arabic to Zulu and more pixels per character are required. The MAXREFDES99# is an easy-to-use 256-pixel solution capable of handling even the most demanding characters.



The Shield...

... RD99 for friends, is an Arduino compatible extension board sporting a 16x16 LED array (actually four 8x8 LED arrays) and the necessary drivers for it. The 7.7 x 7.7 cm (3 x 3") square board is intended for signage applications and its 256 LEDs probably allow displaying any international alphanumeric character that you can think of. It can also display scrolling messages in a 5x7 font, or some other font. What exactly it shows is up to you.

The shield is supported by open-source software libraries (C++) for both the Arduino and ARM mbed rapid development platforms. The libraries not only take care of the low-level communication with the display, they also offer high-level functions for displaying all printable ASCII characters from 'space' (32) to '~' (126) and for scrolling text messages.

The shield must be powered from 5 V_{DCr} supplied by either the host system or the on-board power supply, depending on the position of switch SW1. The input voltage for the on-board regulator is limited by the maximum voltage allowed on the V_{IN} pin (connector H3, pin 8) of the host system or by the voltage rating of C1 (25 V), whichever is lowest. For an Arduino Uno R3 host this is 20 V_{DCr} although it is recommended limiting it at 12 V_{DCr}. The kit comes with a 9 V_{DC} 1.3 A wall-mount power adapter with American-style pins, which is a bit of a shame since it accepts anything from 100 to 240 V_{AC} as input. The RD99 monitors the IOREF pin (H3, pin 2), the pin that, in

Arduino systems, indicates the I/O voltage level of the host system. A level shifter automatically adapts to this voltage making the shield suitable for systems with I/O signal levels as low as 1.2 volts.

Communication with the host system requires only three signals: clock (rising edge on 'D13', H4, pin 5), data ('D11', H4, pin 7) and a load signal (rising edge on 'D10', H4, pin 8). These signals are compatible with 16-bit SPI busses where the Slave Select (SS) signal can be used as the load signal. Add to these signals a reset line (active low, H3, pin 3), the IOREF voltage and power, and you know that the all singing, all dancing interface needs eight connections.

The MAX7219

The RD99 has four MAX7219 LED display drivers, one for each 8x8 LED matrix. Each part is capable of driving up to eight 7-segment displays (with decimal dot) or up to 64 individual LEDs. The MAX7219 includes a BCD code-B decoder, multiplex scan circuitry, segment and digit drivers, and an 8x8 bit static RAM to store the value for each LED. A serial interface allows programming the device's registers and, of course, activating or deactivating individual LEDs. Individual digits (or groups of eight LEDs) may be addressed and updated without rewriting

the entire display. The integrated BCD decoder may save some work upstream but can be deactivated when it is not needed. The brightness of all LEDs is controlled by a single external resistor that sets the current flowing through them; four of these control the four 8x8 LED matrices. Display brightness can also be set digitally (PWM) by using the intensity register. A convenient 4-wire serial interface permits easy connection to a microcontroller. The serial protocol is very similar to 16-bit SPI. If the microcontroller's SPI peripheral is not capable of controlling the MAX7219 (because it is 8-bit only) then it is easy enough to bit-bang the data into the device as there are no strict timing constraints. The only thing to keep in mind is that data always passes through the chip's internal shift register no matter what the state of the LOAD pin is. A rising edge on this pin will load the data from the shift register into internal memory and update the display, so be careful to load at the right moment.

Using It With Arduino

In what follows it is supposed that a working Arduino development environment is available, including an Arduino Uno R3 board. Before plugging the RD99 on the Uno, check if the shield is working. To do this, connect the power adapter and put SW1 in the position closest to the power input socket (this will set it to external power). A power indicator LED should light up (irrespective of the position of SW1). Disconnect the power and plug the RD99 on the Arduino Uno (without changing the position of SW1), reconnect the power adapter and connect the Uno to your computer.

From [2], from the Design Resources tab, download the firmware for the Arduino Platform. Unpack this file to a folder where you can find it, then copy the folder `MAX7219` into the folder `libraries` of the Arduino Sketchbook location (to find it, from the Arduino menu select File -> Preferences). If the `libraries` folder does not exist, create it first.

Launch the Arduino IDE and go to File -> Examples -> MAX7219 and open `MAXREFDES99_example`. If you don't see this example, then the library is not installed correctly or the Arduino IDE was running while you copied the library. If it was then close and restart the IDE. Click the Upload button to load the program into the Uno. For me uploading only worked when the shield was powered from the power adapter.

When uploading is finished, open the Arduino IDE's Serial Monitor and set the baud rate to 115,200 baud. A menu will appear. If it doesn't, press the Reset button on the Uno. Type '6' in the Send box and click the Send button. At first you may think that nothing is going to happen, but after about 5 seconds your command is suddenly acknowledged, the display comes alive and you can admire a magnificent demo.

When the demo has terminated the attentive observer will notice

As an authorised distributor for Maxim, Mouser carries more Maxim Integrated products than any other distributor. If that weren't enough, we're adding new products daily so you always have access to the newest products from Maxim. Plus, Mouser backs it all with same-day shipping, fast and accurate order deliveries combined with quality service and support.

Find the newest Maxim products on Mouser.com

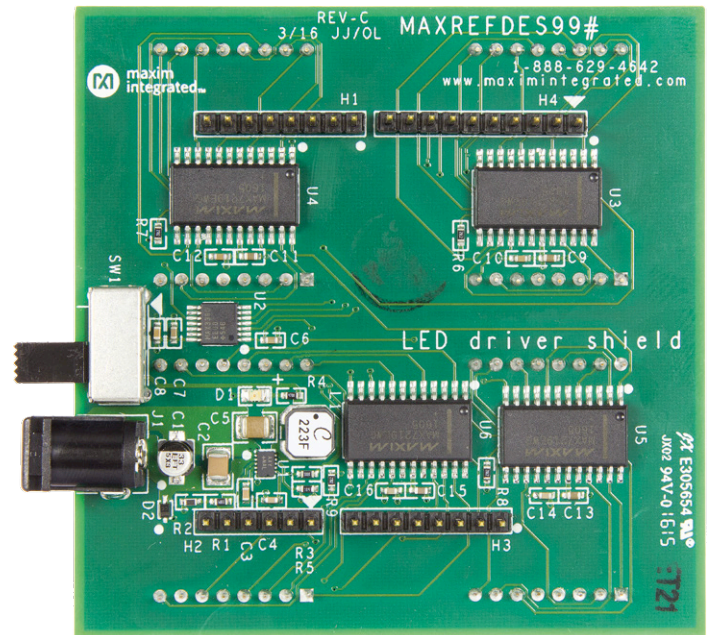


Figure 1. Backside of the shield

a problem with the last character in the lower right corner, the 'm'. It seems as if one pixel has jumped out of line, transforming the 'm' into a strange character. Inspection of the character 'm' in the file `maxrefdes99.cpp` (already open in the Arduino IDE) shows why (line 130): the sequence `"0x58, 0x44"` should read `"0x78, 0x04"`. (Can't figure out why? Write out all the numbers as binary sequences where a '1' is a dot and a '0' is not.) Apply the correction, upload the program and run demo number 6 once more. All should be well now.

A Simple Animation

There is a second example named `MAX7219_example`. Open it, upload it to the board, see what it does, and study the source code. It is rather simple, but it shows clearly how to light individual pixels and where they are located on the grid. Inspired by this example I created a little full-screen animation that you can download from [3]. The animation consists of a sequence of 11 frames displayed one after the other with an inter-frame delay of 25 ms. The last frame is shown for 1.5 seconds, then the animation starts again. A second sketch shows a way of scrolling large custom characters. Videos of these sketches are available on [3].

Having studied the provided examples you should now be ready to create signage applications to your own specifications. Have fun! ◀

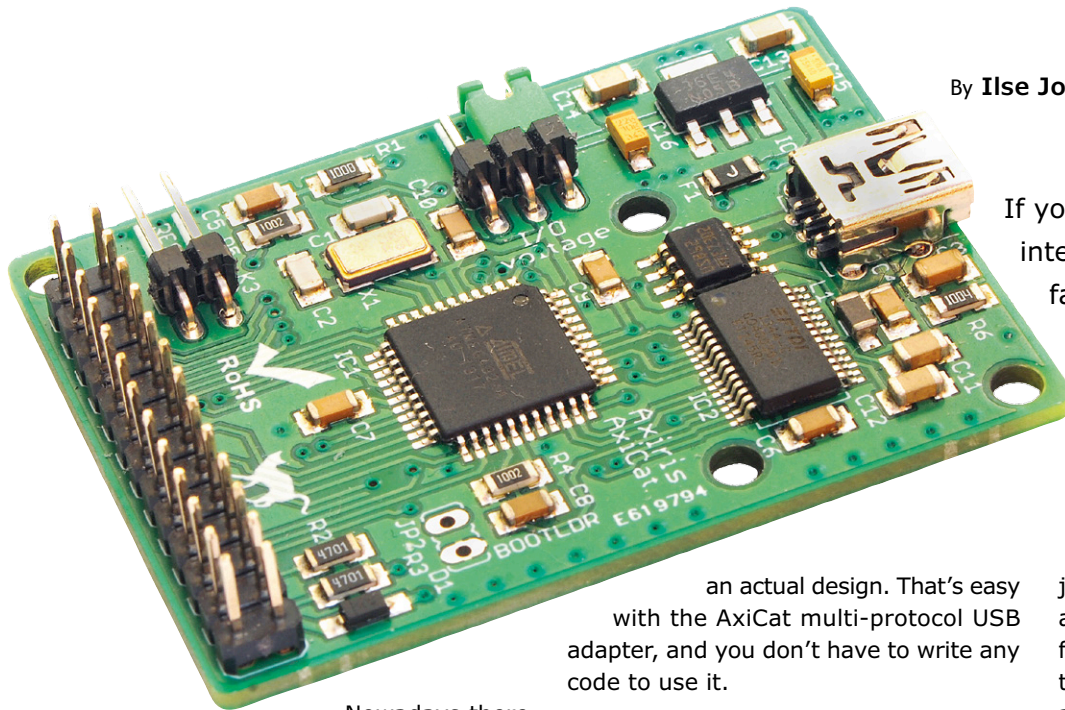
(160267)

Web Links

- [1] www.mouser.com/new/maxim-integrated/maxim-maxrefdes99
- [2] www.maximintegrated.com/MAXREFDES99
- [3] www.elektormagazine.com/labs/maxrefdes99-led-shield-experiments-160267

AxiCat

A versatile USB development tool



By **Ilse Joostens** and **Peter S'heeren** (Belgium)

If you are developing an app for an integrated circuit and you want a fast and easy way to link it to a Linux or Windows PC, we have the answer: AxiCat.

Nowadays there are thousands of useful ICs available to electronics hobbyists and professionals. A lot of them only come in SMD packages, which are not very convenient for hobbyists, but don't worry: many well-known (and less well-known) vendors who focus on the hobby sector offer a broad spectrum of breakout boards. Most of these ICs have an I²C or SPI interface, and it would be handy if you could quickly connect them to a Linux or Windows PC to check out their functions before you start using them in

an actual design. That's easy with the AxiCat multi-protocol USB adapter, and you don't have to write any code to use it.

Schematic and structure

The key component of the circuit, as shown in **Figure 1**, is an Atmel ATmega164A microcontroller (IC1). You can also mount other members of the same microcontroller family, such as the ATmega324A, 644A and 1284. The only difference is the amount of internal memory. The AxiCat normally comes with an ATmega164A or an ATmega324A, depending on availability.

The microcontroller is powered from 5 V or 3.3 V, depending on the position of

jumper JP1. For that reason we opted for a 12 MHz clock crystal instead of a higher frequency, to ensure that the microcontroller remains within its safe operating area with a 3.3 V supply voltage.

The USB connection is handled by IC2, an FT245RL from FTDI. Unlike the better known FT232RL, this device does not have a serial interface, but instead an internal FIFO with an 8-bit data bus. The advantage of this is that with Full Speed USB and a 12 MHz clock frequency, you can achieve a respectable data rate between the USB bus and the microcontroller (IC1).

Thanks to the underlying layers of the USB communication protocol between the driver in the host system and the FT245RL, the data arrives in the FIFO of the FT245RL without any bit errors. This means that the data read by the microcontroller is error-free, and the data output by the microcontroller also arrives in the host system free of errors, eliminating the need for error correction in the communication protocol. That also pays dividends in terms of speed. IC3, a USB6B1, protects the bus lines against ESD, making the board significantly more robust. The 3.3 V supply voltage is provided by IC4, an LM1117-3.3 low-drop voltage regulator. This voltage is available on GPIO connector K2, and along with the 5 V supply voltage from the USB port it can be used as desired to power the microcontroller and the I/O portion of IC2.

Features

- Based on a powerful ATmega164A microcontroller clocked at 12 MHz
- Compatible with Windows and Linux
- Selectable 3.3 V and 5 V logic levels
- 17 bidirectional I/O lines with configurable pull-up
- I²C bus with pull-up resistors, master and slave mode; maximum speed 400 kHz
- SPI master with four slave select lines; maximum speed 6 MHz
- Two independent serial ports
- 1-Wire master with strong pull-up option and built-in accelerators for enumeration
- Powered over the USB bus
- Asynchronous design – all interfaces can be used simultaneously
- Can be used as an in-system programmer for AVR microcontrollers
- Boot loader for easy firmware upgrade and loading your own software
- GPIO connector pinout compatible with Raspberry Pi A and B

Polyfuse F1, with a rated value of 350 mA, limits the total current consumption to prevent overloading of the supply voltage lines of the USB port. Specifically, this means you should keep the total current drawn from the GPIO connector (I/O lines, 5 V supply voltage and 3.3 V supply voltage) below 350 mA.

Diode D1 in the 5 V supply line to pins 2 and 4 of GPIO connector K2 is there to prevent extension cards that supply

power to the Raspberry Pi from also feeding power to the AxiCat and through the USB port to the computer.

GPIO connector K2 is a 26-pin header. This connector and its pinout (see **Figure 2**) were not chosen randomly; they largely correspond to the GPIO connector on the early Raspberry Pi models (A and B). The main advantage of this is that a large number of extension cards and HATs for the Raspberry Pi can also be

connected to the AxiCat. However, extension cards for the newer Raspberry Pi models with additional I/O lines on the 40-pin connector (A+, B+, 2 and 3) are not compatible.

Finally there is jumper/connector K3, which can be used to manually reset the AxiCat. If jumper JP2 is kept closed at the same time, the internal boot loader is activated. This allows you to send your own firmware to the AxiCat.

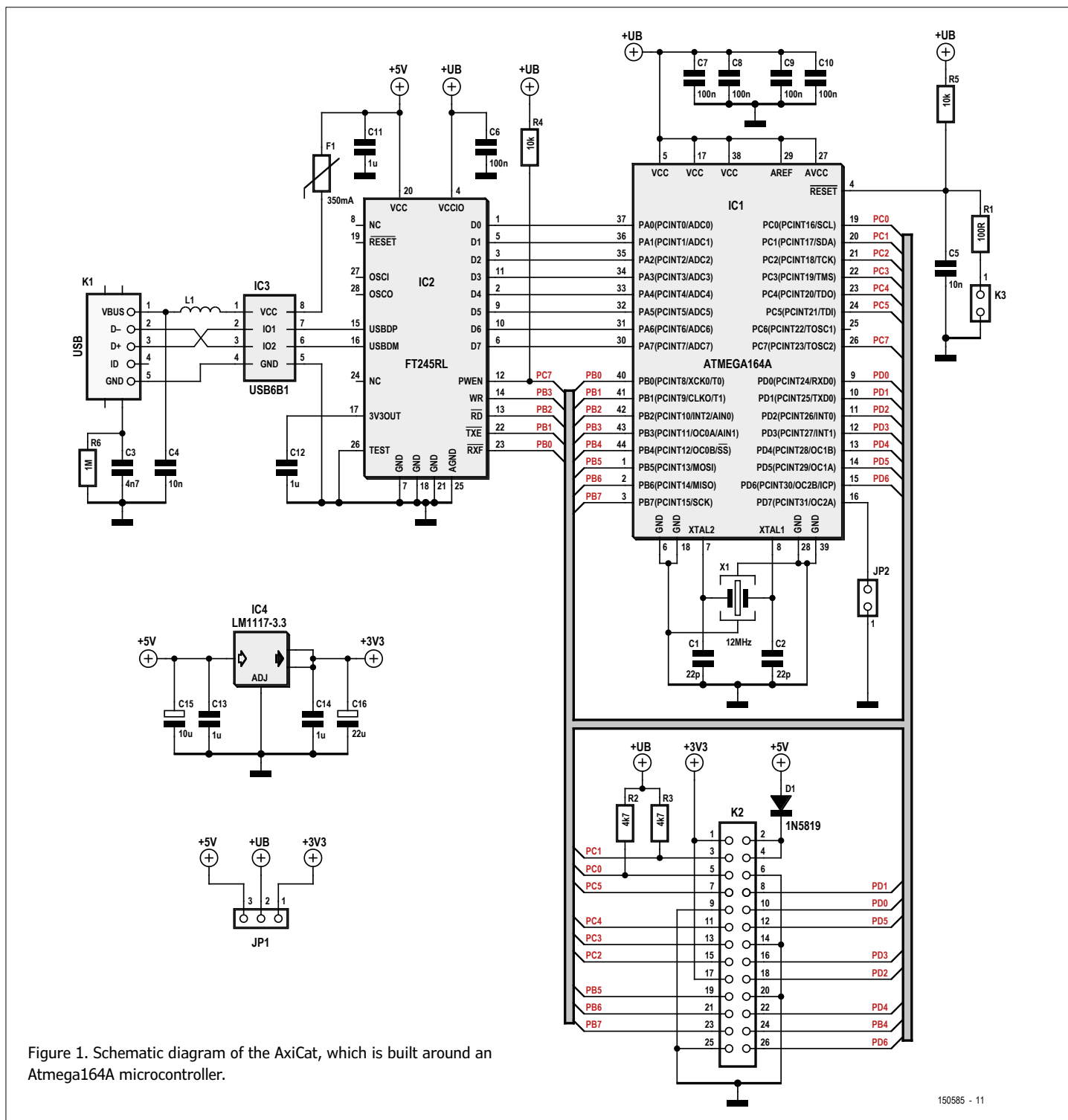


Figure 1. Schematic diagram of the AxiCat, which is built around an Atmega164A microcontroller.

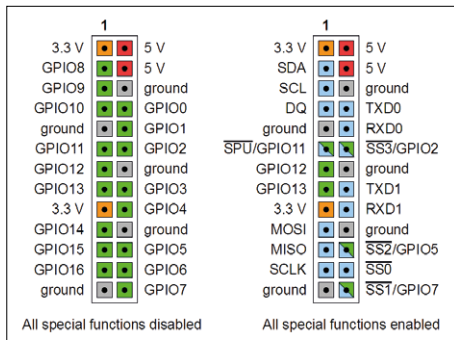


Figure 2. The header pinout is designed to be compatible with Raspberry Pi models A and B. This allows a large number of Raspberry Pi HATs to be connected directly to the AxiCat.

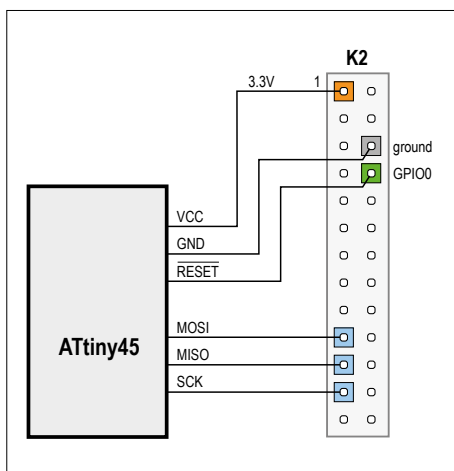


Figure 3. Using the AxiCat as a programmer connected to an ATtiny45.

Serial data protocol

The serial data protocol consists of byte packets which flow between the host and the AxiCat in full duplex mode. The host sends command packets, and the AxiCat sends response packets. The file *axi-cat.h*, included in the download package from [1], contains the definitions in the C language.

There are two types of commands: synchronous and asynchronous. They differ in how the AxiCat executes them after they are received. Synchronous commands are executed immediately. If a response package is defined for the command, the AxiCat returns it to the host immediately. Among other things, synchronous commands are used to read status data, modify settings, and access GPIO lines.

Some commands take a relatively long time to execute. These are commands

that involve bus transactions, such as an SPI transfer or a 1-Wire enumeration. To avoid blocking of the incoming flow of data packets, the AxiCat executes long-duration commands asynchronously, which means in the background. With asynchronous commands, the AxiCat waits until the command has been executed before it returns a response to the host.

Asynchronous commands are stored in working buffers, which are available for the following components: I²C master, I²C slave Tx, I²C slave Rx, SPI master, 1-Wire master, UART #0 Tx, UART #0 Rx, UART #1 Tx, and UART #1 Rx. Note that the UART working buffers do not retain any commands; they only send data bytes (Tx) or receive data bytes (Rx). The working buffers enable the AxiCat to perform bus transactions on several different interfaces in parallel. The effective size of the working buffers can be configured in the firmware and is primarily determined by how much RAM is available in the microcontroller.

There are also autonomous response packets, which are called unsolicited responses. For example, there is no command for reading received data bytes from a UART. This means that the AxiCat automatically sends a response packet when a UART receives data bytes.

Firmware

The main program of the AxiCat is *AxiCat Main*, which is what makes everything tick. The serial data protocol is implemented here. The main program is written in C; we used WinAVR 20100110 for this. WinAVR is a Windows version of GCC for AVR 8-bit microprocessors and AVR Libc. During the development of the AxiCat we frequently checked the generated assembler code, and in many cases we rewrote the source code to make the machine code more efficient and compact. That was necessary for two reasons: firstly because fast iteration of the main loop is crucial for the performance of the AxiCat, and secondly because the main program has to fit in the 16 KB of flash memory in the ATmega164A.

We opted to use interrupts as little as possible, because the GCC compiler often generates interrupt routines that push a lot of registers onto the stack and later retrieve them, which degrades the effi-

ciency of the interrupts. Instead of interrupt routines, we use non-blocking tasks integrated into the main loop. *TIMER1*, which acts as a millisecond counter, is the only hardware function that works with an interrupt routine.

We chose the size of the working buffers based on the available RAM and the code generated by the compiler. The main program uses working buffers with a size of 2^n bytes and a maximum size of 128 bytes, which results in very efficient and compact code for an AVR 8-bit microcontroller.

The main program supports USB remote wakeup, so the AxiCat can wake the host computer when it is in sleep mode. Of course, this is only possible if the operating system of the host also supports USB remote wakeup.

AxiCat Bootloader is executed when the AxiCat starts up. The boot loader first checks the state of jumper JP2. If the jumper is open, the boot loader jumps to the main program. If the jumper is closed, the boot loader remains active and you can use the boot loader program *aximcubldr* to download a new main program from the host computer to the AxiCat.

Software

The *AxiCat Server* program provides the functionality of the AxiCat via network ports and standard I/O in the form of a communication protocol. The server can accept multiple client connections, allowing several clients to work with the AxiCat concurrently. The server acts as a multiplexer between several clients and a single AxiCat, which creates a lot of flexibility.

The communication protocol consists of commands and corresponding responses. A client sends a command, and the server returns a response once the command has been executed. The communication protocol is coded in ASCII format. The syntax is designed to make the communication protocol easy to use both manually and in software.

The server supports a large number of commands for working with the AxiCat. Some examples are:

Configure GPIO pin 1 as an output, set the output low and read the state:

```
Command  iod 1 out
Response iod ok
```



```

Command iow 1 0
Response iow ok
Command ior 1
Response ior 01 1 0 out

```

PCF2129A real time clock: activate I²C master, write register index 0, read registers 0–9:

```

Command ime
Response ime ok
Command imw 81 0
Response imw 081 00001 ack
Command imr 81 10
Response imr 081 08h 16h 04h 30h
          22h 16h 06h 00h 08h 16h nack

```

Activate SPI master, transfer four bytes and use slave select #2:

```

Command sme
Response sme ok
Command smt 2 0FFh 0FFh 10100b 5
Response smt 2 000 255 032 005

```

Many responses from the server contain numbers. The format of these numbers is fully configurable – you can choose from binary, decimal and hexadecimal, with or without leading zeros.

The communication protocol is fully asynchronous, which means that a client does not have to wait for the response to a command before sending the next command. This allows clients to send several commands in direct succession, which dramatically increases communication efficiency. Each command can optionally be provided with an identification number, which is sent back with the response. For each of the two UARTs, the server offers the option of transferring data bytes directly between the UART and a network port. This option allows you to use a terminal emulator program such as PuTTY to read and write data directly through the UART.

The *AxiCat Application Layer* module provides a C API on top of the serial data protocol. The C API supports the full functionality of the serial protocol, including asynchronous transfers, so that programmers can manage the various interfaces of the AxiCat optimally and operate them in parallel. The source code of the AxiCat server provides good insight into how to use the C API. This module is used in a variety of programs, including *AxiCat Server*, *AxiCat AVR ISP*, *I/O Card Explorer*, and *1-Wire Automation Server*.

AxiCat AVR ISP converts the AxiCat into an in-system programmer for 8-bit AVR microcontrollers. It can program flash memory, EEPROM and fuses, and it can read Intel hex files and raw files. *AxiCat AVR ISP* works with the SPI bus (but without slave select) and a reset line. You connect the SPI bus between the AxiCat and the microcontroller, and any desired GPIO pin to the reset pin of the microcontroller. An example is shown in **Figure 3**.

The *AxiCat Command Line* tool converts simple text commands into byte packets and sends them directly to the AxiCat. This tool is mainly useful for developing and debugging the main program for the AxiCat. You can use it to send invalid packets to the AxiCat, which is a good way to test the robustness of the main program. It is also a handy tool for anyone who wants to study the serial data protocol.

Summary

We mainly use the AxiCat to test our own extension boards for the Raspberry Pi (Swiss Pi and PiWire+) and to install firmware on these boards. We also use the AxiCat in all of our software development, because writing and debugging C code is a lot faster and easier on a PC than on a Raspberry Pi. At the end of the development process, we compile and test the software on a real Raspberry Pi.

The AxiCat is available from the Elektor Store as a fully assembled module [2]. You can find a lot more background information and all necessary software at [3]. ◀

(150585-1)

Web Links

- [1] Software download:
www.elektormagazine.com/150585
- [2] AxiCat in the Elektor Store:
www.elektor.com/150585
- [3] More information and software:
www.axiris.eu/en/index.php/i-o-cards/axicat

Component List

Resistors

Default: SMD 1206
R1 = 100Ω
R2,R3 = 4.7kΩ
R4,R5 = 10kΩ
R6 = 1MΩ

Capacitors

Default: SMD 1206
C1,C2 = 22pF
C3 = 4.7nF
C4,C5 = 10nF
C6–C10 = 100nF
C11–C14 = 1μF
C15 = 10μF 10V, tantalum, case A
C16 = 22μF 10V, tantalum, case A

Inductors

L1 = ferrite core, 3A (Farnell #1653393)

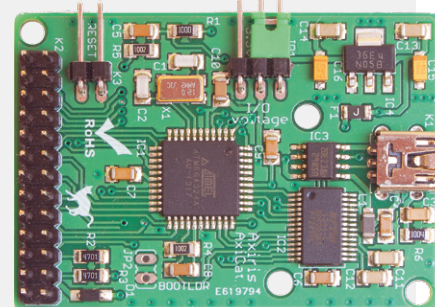
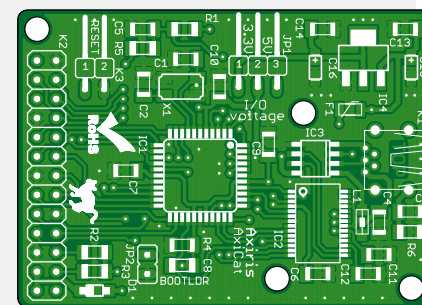
Semiconductors

D1 = 1N5819, SOD123
IC1 = ATmega164A-AU, programmed*
IC2 = FT245RL
IC3 = USB6B1
IC4 = LM1117IMP-3.3

Diversen:

F1 = polyfuse 350mA, 1206
JP1 = 3-pin pinheader, right angled, 0.1" pitch with jumper
K1 = mini-USB connector, type B
K2 = 26-pin (2x13) pinheader, 0.1" pitch
K3 = 2-pin pinheader, right angled, 0.1" pitch with jumper
X1 = 12MHz quartz crystal

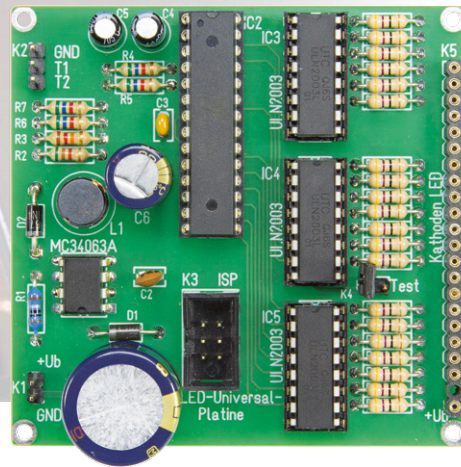
*see text





LED Stairway Lighting

One controller for a whole bunch of LEDs



By **Peter Grundmann** (Germany)

If the *Stairway to Heaven* was fitted with lights I guess they would look something like this... Each step is softly illuminated in sequence by a microcontroller, not just fascinating but energy efficient as well!

A friend was telling me how he had been impressed by a cool stairway lighting system he had seen in a high-tech house. A pushbutton at the bottom of the staircase starts a lighting sequence where each step beginning from the bottom is gradually lit up until the topmost step is reached. At the top another pushbutton turns off the stair lighting, again in a sequence which begins from the bottom all the way up to the top step. Pressing the top pushbutton again begins the lighting sequence but this time from the top to the bottom. Hey, that sounds neat but after some research I found the cost of a commercially available controller unit — at few hundred euros seemed a bit excessive. I have been working with Atmel microcontrollers for some time now and wondered how simple it would be to build my own version.

First thoughts

To work out what we would need, we can firstly assume that a domestic stair-

case is seldom likely to have more than 20 steps. For inputs we need a pushbutton at the top and at the bottom of the stairway to turn the lighting on and off. This number of I/O lines could easily be taken care of by an ATmega8 controller. Without the need of a serial port or external crystal all the I/O pins are free to be used as required.

Each step can be lit by a string of LEDs and where necessary two strings wired in parallel to give more light. A standard microcontroller I/O will not be able to supply the voltage or current required for each string so we will need some output driver chips to drive the LEDs.

This job can be performed by the venerable ULN2003 chip which comes in a DIP package and contains eight Darlington-transistor drivers able to switch up to 50 V at 500 mA, much more than we need in this application. The inputs to each buffer include a series resistor so they

can be connected directly to the microcontroller I/O pins. Looking at the schematic in **Figure 1** the only other items besides the controller and output buffer chips are the two pushbuttons connected via connector K2 and the power supply. The circuit does not contain any dangerous high voltage levels because it is powered by a mains adapter unit. The adapter should be able to supply an output voltage in the range of 12 to 20 V DC and can, for example be salvaged from a disused router. The processor runs with a supply of 4 V so a simple linear voltage regulator would need to dissipate considerable power. Despite the fact that for operation the controller only draws a few milliamps, each active output adds around another milliamp so the power dissipated by a smaller outline L-type linear regulator would quickly exceed its limit. We would therefore need to use a larger TO220-outline linear regulator or a more efficient switched-mode regulator

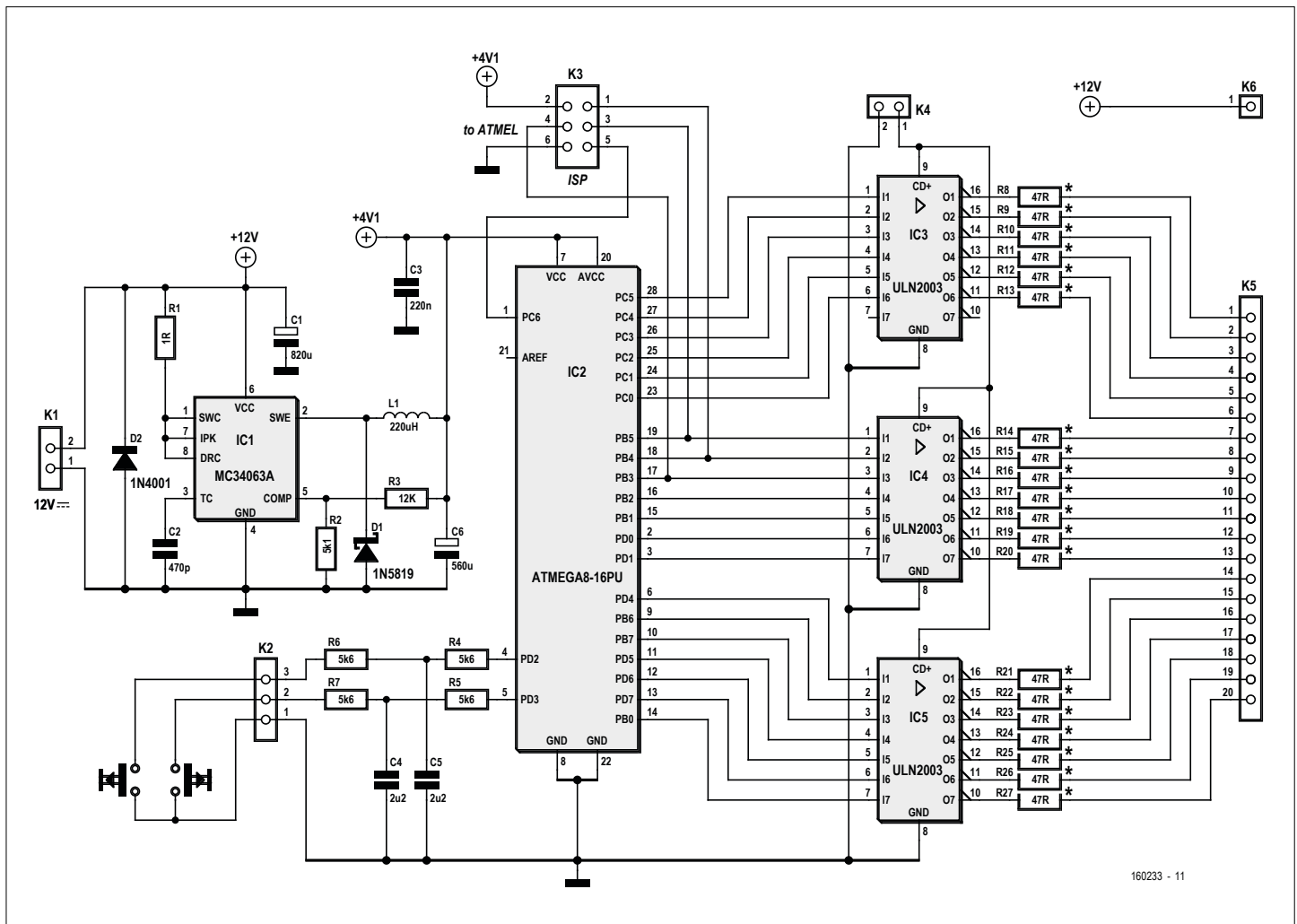


Figure 1. The Atmel controller provides the muscle.

like the MC34063A which only requires a few additional external components.

Circuit building

The circuit was first tested using some perfboard (a.k.a. stripboard/Veroboard). The driver chips use a DIP package so they were really quick and easy to wire up. The step-down converter chip was also quickly configured according to information from the datasheet and takes up little space on the board. The connections necessary for programming the controller were more difficult to arrange, a double-sided board would certainly make the job easier. This PCB contains a few components not essential to the board's function; D2 is included to prevent any damage if the input supply voltage is connected the wrong way round. The push-buttons to activate the lighting sequence are likely to have long cable runs so they are fitted with an RC filter network to suppress any unwanted interference pickup

in the wires. All of the outputs driving the LEDs are fitted with series resistors to limit maximum current to safe values. The ULN2003 driver chip has in built protection diodes at each output which can be used to suppress voltage spikes when the chip is used to switch inductive loads. In this application we connect the common cathode pin of the three driver chips together. When this connection is linked to ground via a jumper it switches all the LEDs on together — that's really useful during installation to test the LED wiring independent of the controller program.

The Program in Bascom

The program has been written in Bascom (free download from [1]). I have purchased the full version of the compiler without the code size restrictions of the free demo version. The controller also has more than enough memory for this application so no attempts were made at code optimization. The code would need

to be modified to make it work with the demo version of the compiler.

The port pins have been individually declared and are referenced via their alias names so that porting to a different controller or any changes to the I/O pin assignments will be made easier.

Generating the pulse width modulated (PWM) signals to control the LED brightness was a bit more challenging. The built-in PWM functions are only available for use on a few port pins so they are not of much help in this application. We can of course do the necessary work 'long hand': A variable Pwm_help is used and continuously increments from 1 to 254 and rolls over to 1. A byte variable for each LED and step level defines the value of brightness. When this value is smaller than Pwm_help the LED is off and when it is greater the LED is switched on. A brightness value of 0 corresponds to the LED being completely dark and values from 1 to 254 define the mark/space

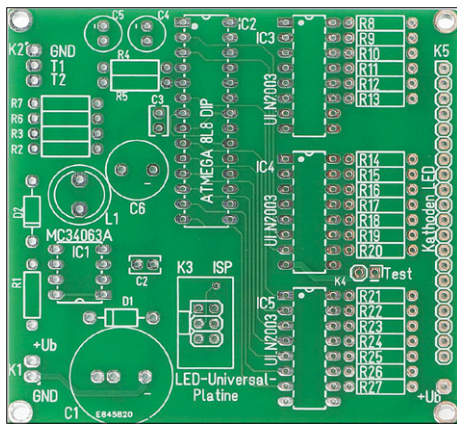


Figure 2. The uncluttered PCB layout makes it easy to fit the components.

ratio increasing until at 255 it is continuously on. It only remains for this evaluation process to be linked to an interrupt routine which is called at a reasonable frequency. In this case a reasonable frequency is one that's high enough so that the eye cannot detect the LED switching. When a pushbutton at the bottom of the stairway is pressed the illumination level of the individual steps is increased from the bottom step to the top step (also in the other direction for descending) with a delay between each step. This is simply controlled using the value of a few bits and effective over four steps. You can alter the speed change value by changing the values of the variables Delay1 and Delay2.

Energy efficiency

Any lighting system such as this should be energy efficient. Powered from a 12 V supply the switch regulator alone draws around 2.5 mA, with the controller plugged into its socket the current increases by a further 3 mA. Tests using different versions of the ATmega8 showed

no significant difference in power consumption. Experimenting with different sleep modes in the controller did not produce significant savings in the 36 mW consumed by the board. With the operating mode triggered, current to the board rises to just above 14 mA and then you need to add the current taken by the LEDs to this figure. Here the resistors in series with the LEDs produce the biggest losses. This equates to an energy loss of 2.5 watts. In normal operation the LEDs will not be in continuous operation so these power losses can be considered acceptable.

Assembly

The PCB layout is shown in **Figure 2** and is available as a Target3001! file for download from the Elektor Magazine project page [1]. Fitting all the components to the board should be fairly simple on this well designed board. All the components are standard and stocked by many suppliers but it will first be necessary to work out the best value and power rating for the LED series resistors. The 47- Ω 0.25-W values specified in the parts list are designed for use with nine LEDs per step. The nine LEDs consist of three 'strings' of LEDs wired in parallel (each string is made up of three LEDs wired in series). It will be useful if all the LEDs come from the same manufacturing batch so that their characteristics will be similar and produce more or less equivalent light output.

To work out the value of the LED series resistors I first used information from the LED data sheet. After some experimentation with the LEDs however, it was clear that their forward voltage drop was consistently lower than the value stated. Some experimentation may be necessary to arrive at the best value for these resistors.

You can also use continuous, adhesive LED strips cut to length. This type of LED lighting strip is specified for operation at 12 V. The strip consists of groups of three LEDs in series together with a series resistor and will not require any additional board-mounted series resistor. The 47 Ω resistors can, in this case be replaced with short lengths of wire. The Darlingtons in the ULN driver chip have a saturation voltage in the range of 0.8 to 1 V, you will need to factor this value into any calculation of the LED drive current and supply voltage to achieve the desired brightness.

How you hook up all the LEDs is up to you, for the purposes of experimentation you can use a connector but when the unit is installed its simpler to just solder the wires onto the board pads.

Why stop there?

The PCB can of course be used for other projects; everything that you need to control LEDs is on board. According to the data sheet the voltage regulator can accept an input voltage up to 40 V. The voltage rating of reservoir capacitor C1 will however need to be increased correspondingly. If you find your style is getting cramped by the relatively small 8-K flash memory in the ATmega8 you could substitute the more powerful, pin-compatible ATmega328.

(160233)

Web Link

- [1] www.elektormagazine.com/160233
(Bascom source code and Target PCB files)

The Author

Ever since he was young Peter Grundmann (b. 1956) has been fascinated by electronics. His first practical experiences were connected with his interest in model trains. Much later he graduated in business studies and now works as a software development engineer.

He has always been drawn to hobby electronics and besides analog and digital projects he is now mainly occupied with microcontroller project design and programming. Along the way he has developed specialized lighting effects and electromechanical actuators for model train installations. These have also been incorporated into commercial products.

Peter is always happy to hear of your own experiences in this field of activity. You can contact him at: peter.grundmann@meier-modellbau.de.



Internet Radio with Fluorescent Display (2)

Raspberry Pi plus ATmega plus software

By **Michael Busser** (Germany)

The plan is to make a kitchen radio using a good, old-fashioned, tried-and-tested fluorescent display, but with the added twist of an Internet connection. In the first installment we looked at its functions, the hardware and how we drive the display. But, without software, the hardware is nothing: the ATmega in the display module needs firmware, and code is also needed to turn the Raspberry Pi into an Internet radio.

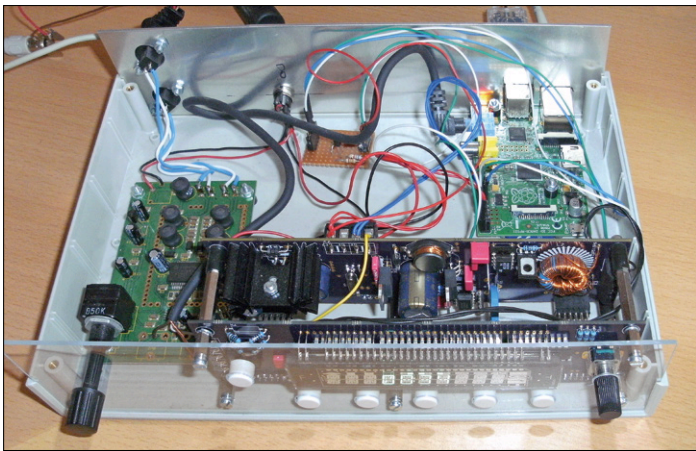


Figure 1. All the boards fitted into a clamshell enclosure and wired together.



Figure 2. Lid on, firmware in: the kitchen Internet radio in operation. The electronics is visible through the acrylic front panel.

If, with a little help from the circuit diagrams, board layouts and other documentation you have built the VFD module from the first installment of this series [1], you may have discovered that (even assuming you have put everything together correctly) not a lot happens when you power it up. Even with all the printed circuit boards installed in the box and wired together (**Figure 1**), and with the lid closed (**Figure 2**) the display will still not light up as shown in the photographs. The reason behind this is that the microcontroller in the display module needs firmware, so that it can understand the control commands sent to it. This firmware and, last but not least, the code for the Raspberry Pi, forms the topic of this second and final installment.

Communication protocol

If two computer systems want to exchange data packets with one another it is necessary to agree beforehand on how the packets are to be interpreted: a protocol has to be defined. The homebrew protocol used here was also designed for use in a home automation system using a two-wire bus, and so has a few features that we will not be needing in this case. Nevertheless, its structure follows the same lines as other similar protocols. The firmware for the ATmega in the display unit was written entirely within the AtmelStudio development environment (see screenshot in **Figure 3**).

Table 1 shows the general structure of a data packet. A message of type *TDataBuf* is passed to the transport layer. The

Table 1. Data packet structure.

STX <i>start of transmission</i>	Message Container	ETX <i>end of transmission</i>
0x02	data	0x03

Table 2. The components of a message.

STX	SRC	DST	MT	MODE	Payload (0 to 15 bytes)	CS	ETX
-----	-----	-----	----	------	-------------------------	----	-----

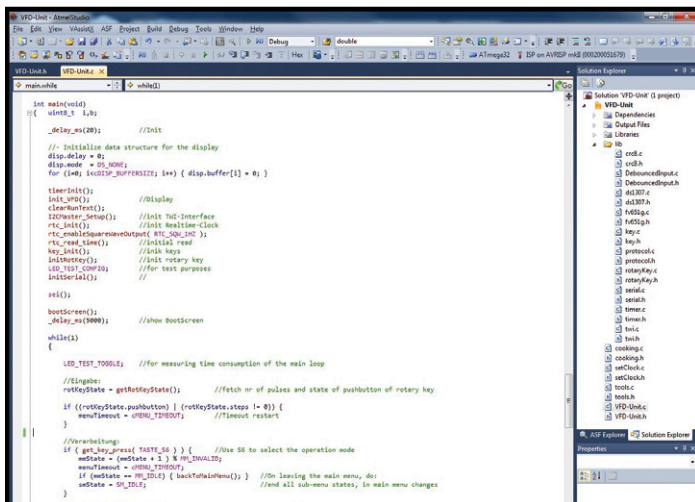


Figure 3. Screenshot of the AtmelStudio development environment showing the ATmega firmware for the display module.

transport layer starts by sending an STX character to mark the beginning of a transmission; an ETX character is sent at the end of each transmission. The STX and ETX characters should not appear within the message itself: if they do, they should be escaped by prefixing them with the data link escape (DLE, 0x10) character. The receiver removes any escape character from the data stream and appends the immediately following character to the partially-assembled message. If it is desired to transmit the escape character itself, it is necessary to send

Table 3. Message codes.	
Field	Meaning
SRC	Source address: 0 to 255 except for the values 2 and 3.
DST	Destination address: 0 to 255 except for the values 2 and 3.
MT	Message type. The following are defined: 0x10 StatusReq 0x11 StatusResp 0x12 DisplayReq 0x13 DisplayResp 0x14 Key_Req 0x15 Key_Resp 0x16 Time_Req 0x17 Time_Resp Note that codes 0x02 and 0x03 may not be used; other codes are available for future expansion.
MODE	Bit Meaning 0...3: Length of payload in bytes: 0 to 15 4: 0/1 = checksum absent/ checksum present 5: 0/1 = acknowledgement with ACK not required/acknowledgement required 6: 0 (reserved) 7: always 1 Note: values 0x02 and 0x03 may not be used. To ensure this bit 7 is specified to always be 1.
CS	Message checksum.

two DLE characters. The following functions provide the software interface (see *serial.c* and *serial.h*):

```
void initSerial(void);
void sendMsg(TDataBuf msg);
uint8_t peekMsg(TDataBuf *msg);
```

The function *initSerial()* should be called once when the main program starts up. It initializes the UART and enables various interrupts. The baud rate can be adjusted by modifying the definition

```
#define UART_BAUD_RATE 9600UL
```

in *serial.h*. The other serial port parameters are eight data bits, no parity, and one stop bit ('8N1'). The function *sendMsg()* takes a message of type *TDataBuf*, copies it to the internal transmit buffer (if it is free) and then returns immediately to the caller. Transmission and reception proper all happen under interrupt control in the background. As soon as a valid message is received it is stored in one of the two receive buffers. A call to the function *peekMsg()* will then return a '1' and the valid message will be written to *msg*.

The escape mechanism is handled transparently within these functions. There is one transmit buffer, but two receive buffers that are used alternately: this allows one message to be processed while the next one is being received.

The code module is structured independently of the format of the messages being sent, which is defined in *protocol.c* and *protocol.h*. Two-wire bus systems such as RS-485 require an extra control signal which switches the hardware between transmit and receive modes. The default mode is 'receive', transmit mode being enabled only for the duration of a message transmission. This is the purpose of the two definitions

```
#define SENDER_OFF
#define SENDER_ON
```

in *serial.h*. The receiver runs continuously, and so will pick up any message that is sent: this can be used to detect a bus collision. The only function we need in *protocol.h* is *ownAdr()*, which returns our own address on the bus.

Table 2 shows the individual components of a packet and **Table 3** describes their meanings. **Listing 1** shows the structure of a message as a C 'struct'; the 'union' is a construct which describes how the *TDatagramm* structure and the *TBuf* byte array can overlay one another at the same memory addresses. This allows easy reading and writing of the message as a simple array of bytes, which makes the receive and transmit code more straightforward.

ATmega communication

The following message types are defined.

```
#define MT_STATUS_REQ 0x10
#define MT_STATUS_RESP MT_STATUS_REQ + 1
```

```
#define MT_DISPLAY_REQ 0x12
#define MT_DISPLAY_RESPMT_DISPLAY_
REQ + 1

#define MT_KEY_REQ 0x14
#define MT_KEY_RESP MT_KEY_REQ + 1

#define MT_TIME_REQ 0x16
#define MT_TIME_RESP MT_TIME_REQ +
1
```

The *DISPLAY_REQ* message allows the Raspberry Pi to display something on the VFD. *KEY_REQ* informs the Pi when a button has been pressed on the display module. *TIME_REQ* lets the ATmega obtain the current time from the Pi, which in turn obtains the time over the Internet using NTP.

The ATmega can send and receive messages over its serial interface in a specified format using simple calls to the relevant functions in *protocol.c* (**Listing 2**). Received messages are processed in a loop in the main program (**Listing 3**), which continuously checks whether a new message is available and then decides how to handle it.

Let us look at an example of how these messages are used. Every 60 minutes the ATmega sends a message of type *MT_TIME_REQ* to the Raspberry Pi by calling the function *sendMsgTimeReq()*:

```
if (timerFlags.flags.bMin) {
    doNTP_Sync--;
    if (doNTP_Sync == 0) {
        doNTP_Sync = NTP_SYNC;
        sendMsgTimeReq(); // fetch
current time from Raspberry Pi
    }
    timerClearMin();
}
```

The Raspberry Pi replies with a message of type *MT_TIME_RESP*, which is then processed in the receive loop in the main program.

The source code is too long to reproduce here in its entirety, but it can be found in the file *Software_2.zip* in the download available at [1].

Raspberry Pi software

We use version dated 09.09.2014 of the 'wheezy-raspbian' operating system on the Raspberry Pi. Detailed instructions for obtaining an image, copying it to an SD card, and carrying out basic config-

Listing 1. Message structure.

```
typedef struct {           //construction of a message
    uint8_t  src;         //source address (sender)
    uint8_t  dst;         //destination address (receiver)
    uint8_t  mt;         //message type
    TMode     mode;       //mode byte, describes structure of message
    TPayload  Payload;    //usage depends on application level
    uint8_t  cs;         //checksum, e.g. crc-8
} TDatagramm;

typedef union {
    TBuf      Buf;        //access via byte array
    TDatagramm Datagramm; //access via structure
} TDataBuf;
```

Listing 2. Example message function.

```
void sendMsgTimeReq(void) {
    TDataBuf msg;

    msgClear( &msg );
    msg.Datagramm.src = ownAdr();
    msg.Datagramm.dst = masterAdr();
    msg.Datagramm.mt = MT_TIME_REQ;
    msg.Datagramm.mode.bAckReq = NO_ACK_REQUIRED;
    msg.Datagramm.mode.bCRC = NO_CHECKSUM; //WITH_CHECKSUM;
    msg.Datagramm.mode.count = 0; //don't count header / trailer bytes
    msgPrepare( &msg );
    sendMsg( msg );
}
```

Listing 3. Processing a message.

```
if (peekMsg(&recMsg) == 1) { //if message received: evaluate
    switch (recMsg.Datagramm.mt) {
        case MT_STATUS_REQ:    break;
        case MT_STATUS_RESP:   break;
        case MT_DISPLAY_REQ:   i=0;
            b=1;
            for (i=0; i < cDISP_BUFFERSIZE; i++) {
                if (recMsg.Datagramm.Payload[i] == 0) {b = 0;}
                if (b == 1) {
                    disp.buffer[i] = recMsg.Datagramm.Payload[i];
                } else {
                    disp.buffer[i] = 0;
                }
            }
            disp.delay          = cDISP_SHOWTEXT;
            disp.mode          = DS_TEXT;
            showText();
            break;
        case MT_DISPLAY_RESP:   break;
        case MT_KEY_REQ:        break;
        case MT_KEY_RESP:       break;
        case MT_TIME_REQ:       break;
        case MT_TIME_RESP:      setTimeOnMsgReq( &recMsg );    break;
    }
}
```

Listing 4. The *handleMsg()* function.

```

void handleMsg(int fd, TDataBuf *msg) {
    if (msg->Datagramm.mode.bCRC) {
        syslog( LOG_NOTICE, "checksum ignored");
    }
    if (msg->Datagramm.dst == masterAdr()) {
        switch (msg->Datagramm.mt) {
            case MT_STATUS_REQ:    onStatusReq(fd, msg);    break;
            case MT_DISPLAY_REQ:   onDisplayReq(fd, msg);   break;
            case MT_KEY_REQ:       onKeyReq(fd, msg);       break;
            case MT_TIME_REQ:      onTimeReq(fd, msg);      break;
            default:               syslog(LOG_ERR, "unknown message dropped");
                                dumpMsg(msg);
        }
    } else {
        syslog( LOG_NOTICE, "message not for me, ignoring it");
    }
}

```

Listing 5. The *onTimeReq()* function.

```

void onTimeReq(int fd, TDataBuf *msg) {
    TDataBuf reply;
    msgClear(&reply);
    reply.Datagramm.src = masterAdr();
    reply.Datagramm.dst = msg->Datagramm.src;
    getTimeResp(&reply);
    //dumpMsg(&reply);
    frame_Send(fd, &(reply.Buf[0]), reply.Datagramm.mode.count + MSG_
HEADER_LEN, reply.Datagramm.cs );
    syslog( LOG_NOTICE, "onTimeReq");
}

```

uration tasks can be found at [2].

At [3] there is a guide to configuring the Raspberry Pi as an Internet radio using *mpd* ('Music Player Daemon') and *mpc* ('Music Player Client'), and there are many other similar guides available. A search for 'Raspberry Pi mpd' will magically lead

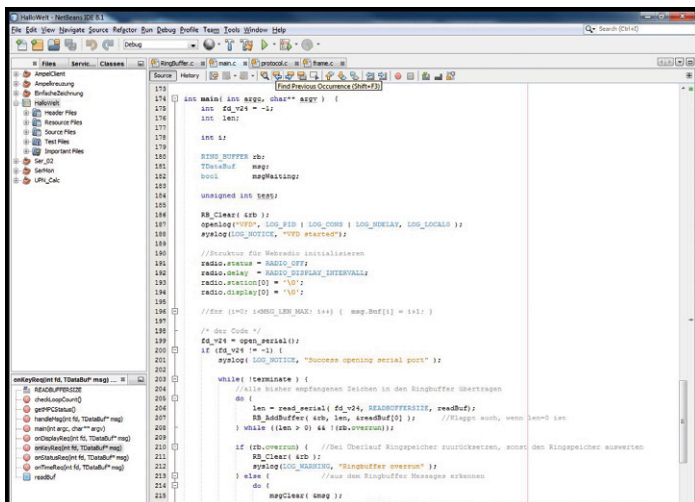


Figure 4. Screenshot of the NetBeans IDE. With the software shown we can turn the Raspberry Pi into an Internet radio.

you to everything you need to know. A development environment is required to get our first C program for the Raspberry Pi up and running (although in principle the command line can also be used). My environment of choice is NetBeans, which being written in Java also runs on my desktop computer. **Figure 4** shows a screenshot of the NetBeans IDE in action. A step-by-step guide to installing the NetBeans environment on the Raspberry Pi can be found at [4].

Once these hurdles are overcome, it is possible to write C programs for the Raspberry Pi conveniently under Windows, and then compile and debug them on the Pi itself.

Fortunately C code written for the ATmega will run without significant changes on the Raspberry Pi, as long as you avoid using any of the special functions and modules which it does not support.

The program running on the Raspberry Pi copies messages to a syslog server, which helps with debugging. The syslog server must be configured on the Raspberry Pi. The function *dumpMsg()* takes one message and outputs it to the syslog in human-readable form. Inside the function *main()* in the Raspberry Pi program we read all the characters that have arrived on the serial port since we last looked and store them in a circular buffer:

```

do {
    len = read_serial(fd_v24, READBUFFERSIZE, readBuf);
    RB_AddBuffer(&rb, len, &readBuf[0]);
    //works even if len=0
} while ((len > 0) && !(rb.overrun));

```

The function *frame_checkFrame()* is called at regular intervals. It checks whether there is a complete message in the circular buffer. If so, it returns '1', and if not, it returns '0'. The parameters *startindex* and *length* give the starting offset and length of the message.

A call to the function *frame_get()* returns the corresponding message in *msg*. This is then processed by the function *handleMsg()* (see **Listing 4**). So if, for example, a message of type *MT_TIME_RESP* is received, the function *handleMsg()* will pass it on to the function *onTimeReq()* (see **Listing 5**). Here a reply to the request message is constructed and then immediately sent back to the sender using the function *frame_Send()*. The NTP client must be configured on the Raspberry Pi. System time is determined by a call to the function *localtime()* inside the function *getTimeResp()*. This function is also responsible for

assembling the reply message in the *TDataBuf* data structure.

Internet radio

The Internet radio functionality itself is implemented using the *mpc* music player client. This allows you to construct playlists from Internet radio stations: the details are set out at [5]. Also, the command *mpc help* is available from the command line. Our specially-written C program controls *mpc* by passing it the necessary options via a call to *system()*, in just the same way as it would be used from the command line. The excerpt from the function *onKeyReq()* shown in **Listing 6** is also worth a look. This function is called whenever a button on the display module is pressed.

If *mpc* is called without parameters then the first line of its output is the current station and the name of the currently-playing program. The function *getMPCStatus()* obtains this information at regular intervals and then sends the station name to the ATmega using an *MT_DISPLAY_REQ* message. The current station name then replaces the current date on the display.

Each radio stream must be stored in a playlist before it can be used by *mpd*. The playlists are stored in the directory */var/lib/mpd/playlists*.

Start-up

As it stands the program on the Raspberry Pi needs to be started by hand, as development is not yet complete. One slight niggle with using NetBeans is that it creates very long directory paths. If you follow the guide to installing NetBeans as the root user exactly, executable code ends up in the directory

```
/root/.netbeans/remote/192.168.1.24/mib2-Windows-
x86_64/D/Projekte/RaspberryPi/RPi/HalloWelt/dist/
Debug/GNU-Linux
```

The part of the path printed in bold changes according to the environment, including the IP address of the Windows PC,

Listing 6. Excerpt from the function *onKeyReq()*.

```
void onKeyReq(int fd, TDataBuf *msg) {
    uint8_t keys = msg->Datagramm.Payload[0];    // define variable / check button
    if (keys & KEY_S0) {syslog( LOG_NOTICE, "onKeyReq: S0");}
    if (keys & KEY_S1) {terminate = true; syslog( LOG_NOTICE, "onKeyReq: S1");}
    if (keys & KEY_S2) {syslog( LOG_NOTICE, "onKeyReq: S2");}
    if (keys & KEY_S3) {syslog( LOG_NOTICE, "onKeyReq: S3");}
    if (keys & KEY_S4) {                                //next station
        if (radio.status == RADIO_ON) {
            system("mpc next");}
        syslog( LOG_NOTICE, "onKeyReq: S4");
    }
    if (keys & KEY_S5) {                                //radio on / off
        if (radio.status == RADIO_OFF) {                //if radio off
            system("mpc play");                          //play last station
            radio.status = RADIO_ON;
        } else                                          //radio off
            system("mpc stop");
            radio.status = RADIO_OFF;
        }
        syslog(LOG_NOTICE, "onKeyReq: S5");
    }
}
```

its hostname (here 'mib2') and the path to the project under Windows (here 'D:/Projekte/...').

However, it is easy to make this program execute automatically when the Raspberry Pi is booted up. One description of how to do this can be found at [6]. If the program is started from a shell, it remains dependent on this shell; if the shell is closed, the program will also terminate. A simple '&' character after the end of the command avoids this:

```
./hallowelt &
```

This makes the program execute in the background, with the shell generating a job number and a process ID. These are needed if you wish to use a command such as

```
fg %1
```

to bring the process back into the foreground.

The program is now ready to run on the Raspberry Pi. Extensions are possible and indeed ideas for extensions are welcomed: you can import the program directly into NetBeans from the file *HalloWelt.zip*. We hope you have fun experimenting! ◀

(160207)

Internet Links

- [1] Software download and first installment: <https://www.elektormagazine.com/150720>
- [2] Raspberry Pi operating system installation: <https://www.raspberrypi.org/documentation/installation/installing-images/>
- [3] Raspberry Pi as an Internet radio: <https://www.youtube.com/watch?v=jf3M1RVpQbM> (audio in German)
- [4] NetBeans for the Raspberry Pi: <http://bit.ly/2aBZ14A>
- [5] Internet radio tutorial: <https://www.youtube.com/watch?v=jf3M1RVpQbM> (audio in German)
- [6] Automatically run a program on boot: <http://pi.bek.no/autostartProgramOnBoot/>

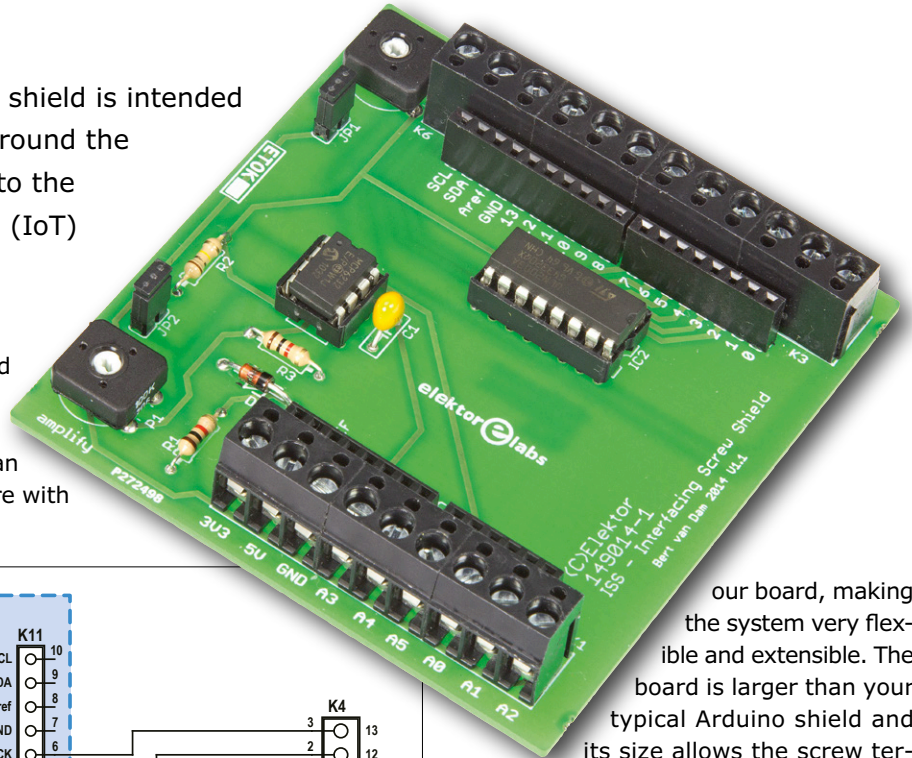
IoT Shield for Arduino

Lets You Do Your (Internet of) Thing

By Clemens Valens (Elektor Labs)

This handy Arduino Uno R3 compatible shield is intended for simple control applications in and around the house. When the system is connected to the Internet, the label “Internet of Things” (IoT) becomes applicable.

Wires to and from the shield [1] can be connected with a screwdriver – no soldering is required. The circuit on the board proper does not use all of the Arduino signals, and other shields can be stacked on it as long as they do not interfere with



our board, making the system very flexible and extensible. The board is larger than your typical Arduino shield and its size allows the screw terminals and trimmers to remain accessible even when another shield is plugged into it.

The board offers several expedient functions:

- a variable voltage source for biasing or powering sensors;
- an analogue input with adjustable gain for small signals;
- a current and voltage limited input for detecting a high voltage;
- six power transistor outputs for switching for instance lamps or relays.

Let's walk through the shield's functions step by step helped by **Figure 1**.

Variable voltage source

Trimpot P2 provides a variable voltage between 0 and 5 V. If jumper JP1 is shorted this voltage is available on the Arduino's analog input A3. In this case it is also available at K1, pin 3, and can be used to control, bias or even power an external (low-current) device like a sensor. When JP1 is left open P2 has no function and this pin can be used as an analog input or a digital input/output.

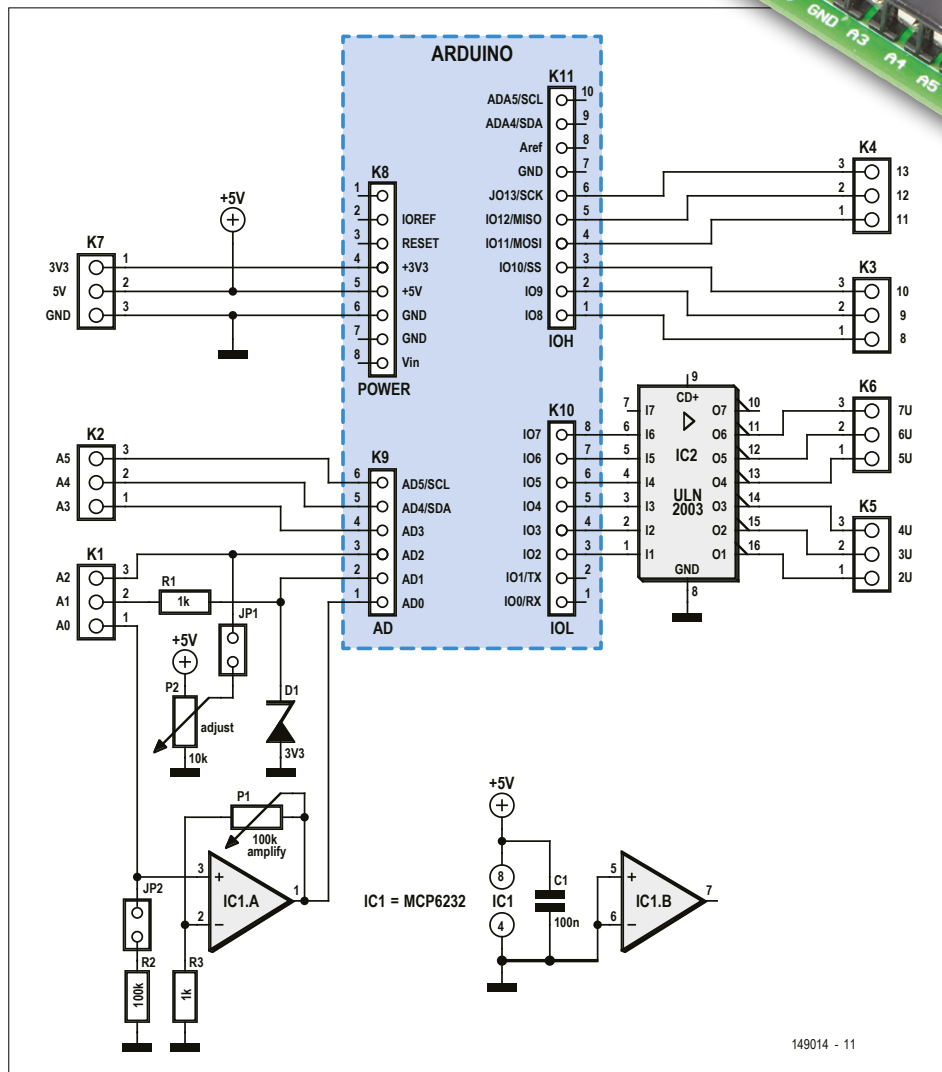


Figure 1. Just the facts, ma'am. A straightforward design without fancy tricks and hard-to-get components.

Small signals

IC1A, powered from the 5-V line, is a rail-to-rail input-output operational amplifier (opamp) meaning that both its input as its output cover the range of 0 to 5 V (within a few millivolts). This output is connected to the Arduino analog input A0 only, hence its signal is not available on a screw terminal block. The opamp is wired as a non-inverting amplifier with its gain is controlled by P1. The maximum gain is about 100, the minimum gain, one. The result is that the input signal on K1, pin 1 can either be amplified or it can simply be buffered. Although the input impedance of the input is very high, it can be fixed at 100 k Ω by shorting JP2. Doing this is useful because it avoids the opamp struggling with noise, hum and other interference when its input is left open. If, on the other hand, the source connected to this input has a very high output impedance, it may be better to open JP2 to prevent squashing such a sensitive signal.

Detect high voltages

K1, pin 2 is a voltage and current limited analog input. Zener diode D1 will limit high DC voltages to 3.3 V maximum — a safe level for Arduino boards. But be careful, negative (AC) voltages are not allowed, not on this input and, by the way, not on any other input. R1 provides current limiting for both the input and D1. Its value is a little high, meaning that when you connect the input to, say, about 5 V, the voltage will be limited to around 3 V, but with an input voltage of 12 V or 15 V you will measure 3.3 V here. The purpose of this input is to detect voltages (much) higher than the Arduino can accept, for instance the output voltage of a DC adaptor or the state of a switch connected to a high voltage. Although in theory this input can easily withstand voltages up to 100 V, **never ever connect it to the AC line voltage!**

Powerful outputs

Part IC2, a ULN2003A, has a long-standing reputation, and a good one, which is why it was used here. In it are seven powerful Darlington transistors that can be controlled digitally. Although the device has seven channels, only six were used on this shield. Each transistor can switch 500 mA and can stand up to 50 V. These transistors are very good to switch something connected to a power supply (50 V

Component List

Resistors

All 5%, 0.25 W
 R1, R3 = 1k Ω
 R2 = 100k Ω
 P1 = 100k Ω , trimpot, horizontal
 P2 = 10k Ω , trimpot, horizontal

Capacitors

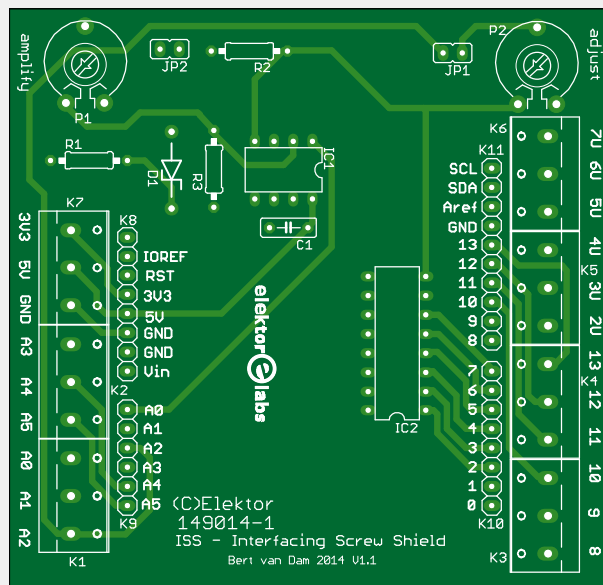
C1 = 100nF, 0.2" pitch

Semiconductors

D1 = BZX85C, 3.3 V, 1.3 W Zener diode
 IC1 = MCP6232
 IC2 = ULN2003

Miscellaneous

JP1, JP2 = pinheader, 2-way, 0.1" mm pitch
 2 pcs jumper for JP1 and JP2
 K1, K2, K3, K4, K5, K6, K7 = 3-way PCB screw terminal block, 5 mm pitch
 K8, K10 = 8-way stackable pinheader socket, SIL, 0.1" mm pitch
 K11 = 10-way stackable pin socket, SIL, 0.1" mm pitch



IC socket, DIP-8 for IC1
 IC socket, DIP-16 for IC2
 PCB # 149014-1 v1.1

max), a relay, a lamp or a motor for instance, to 0 V as they can handle relatively large currents. An Arduino output cannot do this without the help of such a driver circuit. In case you must switch more current than one channel can handle you can put two or more of these outputs in parallel. If you want to switch inductive loads (like a solenoid, relay or motor) you should add a flyback diode in parallel with the load (cathode to the power supply, anode to the output of IC2) as the chip's internal flyback diodes are not used on this board.

Some remarks

Power to the shield is supplied by the Arduino that carries it; do not power the shield through K7 when it is plugged onto an Arduino. This connector is intended for powering devices connected to the shield only, not to power the Arduino. The 3.3-V output is the Arduino's 3.3 V output which is not very powerful, so treat it with loving care. IC1 and IC2 are mounted in sockets so they can be replaced easily in case of a mishap.

Most of the Arduino extension signals have been brought out to sturdy screw terminals to allow easy connection of wires and cables to the board. Most, but not all: the serial port pins (pins 0 & 1,

V_{IN} , Reset, IO_{REF} and A_{REF} are only accessible on the shield connectors. ◀

(160169)

Web Links

- [1] www.elektor.com/interfacing-screw-shield-149014-91
 [2] www.elektormagazine.com/160169

Great examples of how the shield described in this article can be used for all sorts of applications, IoT or not, can be found in the book *IoT Get U Going* (Elektor Store SKU # 17460).

www.elektor.com/iot-get-u-going



DAB+ Antenna Diplexer

Add digital reception to your car radio



If you feel like replacing the VHF-FM radio in your car with something more modern offering DAB+ reception too, you'll discover that combination receivers have two separate antenna inputs. That's because two individual antennas are required. But this RF combiner makes it possible to connect both inputs to one standard car antenna.

By **Alfred Rosenkränzer** (Germany)

I began to pay attention to this remarkable type of filter when I considered replacing the feeble VHF radio in our car with a more modern set including DAB+ reception. Fortunately our car was old enough to be still fitted with a DIN slot (standardized on many European automobiles), making the change-out relatively problem-free; DIN-compliant radios are still available. A DAB+ car radio does of course have two antenna inputs, one for VHF (FM) and a separate one for DAB+. The one for VHF connects to a normal car antenna, whilst some DAB+ radios

are provided with a glass-mounting antenna in case the car is not pre-equipped for digital radio (which was our situation). This front windshield (windscreen) antenna is not an advantage visually of course and moreover it requires making a hole in the dashboard for the antenna cable. I wanted to avoid unpleasant discussions with my wife in any case.

If you wish to use an off-the-shelf antenna, you must first establish whether it will receive signals at all in the DAB+ region; this is because some active antennas have a built-in lowpass filter. The simplest way to find out is to connect it to a spectrum analyzer (**Figure 1**), otherwise test it with a DAB+ receiver. As is well known, the VHF (FM) band covers from 88 to 108 MHz, whilst DAB in Band III goes from 174 up to 230 MHz. L-Band in the region of 1.5 GHz (used for DAB in some territories) is irrelevant for broadcast reception in vehicles.

The available signal needs to be split for the two antenna inputs. In the simplest case you just connect two cables in parallel. This of course upsets the impedance matching, leading to signal reflection and in some cases to (partial) loss of signal. A passive splitter (consisting of three resistors) avoids this mismatch for sure but brings with it additional attenuation, which you seldom want. An RF transformer is another option for sharing the signal.

First trials

You could, however, achieve the desired separation also by combining a lowpass and a highpass filter with suitable frequency characteristics.

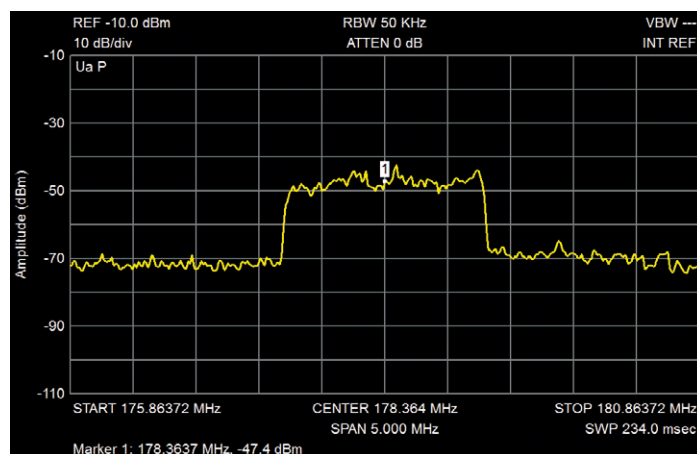


Figure 1. Image of the DAB+ spectrum at 178 MHz, Channel 5c.

To test this out I began by measuring two commercial filters. **Figure 2** shows the frequency responses of the filters, each measured separately with a network analyzer. The frequencies illustrated are not appropriate for separating VHF and DAB+ but both filters are commendably flat regarding the passband region.

If you connect both filters in parallel using a T-adaptor, there is a strong mutual influence on the frequency response (**Figure 3**). After this I experimented further dividing the signal with a resistive power splitter and additional 6 dB attenuators after the power splitter. Although the mutual influence is reduced, it's still not elegant. And there is additional attenuation of 3 or 6 dB. The low-pass filter also shows a falling frequency response. This is not an optimal solution then.

Diplexer software

A diplexer could offer a solution here. This employs a combination of lowpass and highpass filters (or bandpass filter/band rejection filter). In other words a frequency divider for radio frequency use.

Both filters are matched to each other in such a way that they can (and must) be driven in parallel by the source. For calculation there is the 'Diplexer Design' software by James L. Tonne [1]. This program allows you to select between lowpass/highpass and bandpass/bandstop variants. The following description covers only the lowpass/highpass version.

On the Design page of this program (**Figure 4**) you can define the characteristics of the diplexer. The 'Order' is a measurement of the complexity, and hence outlay (the number of components), and the abruptness of the filtering achievable. 'Crossover freq' indicates the point of intersection of the two filters. In this VHF/DAB+ example it is roughly in the middle between the highest VHF frequency (108 MHz) and the lowest DAB+ frequency (174 MHz). The crossover frequency can be fine-tuned to the extent to make the attenuation of the unwanted band in each filter more or less equal. You can also attempt to replace 'awkward' (i.e. not made commercially) coil values calculated with readily available ones.

The lowpass in the upper section begins always with a series inductance. When calculating an individual filter you can select whether you want to start with the series inductance (T-structure) or a shunt capacitor (Pi-structure). Here this is not an option. Correspondingly the highpass always begins with a series capacitor.

'Passband ripple' indicates the amplitude of the periodic variation in the passband region, as in a Chebyshev filter. The greater the ripple, the sharper the filter and vice versa.

'System Z' indicates the impedance, in our example 50 ohms. When you press the 'Plot' button, you reach the next screen-grab (**Figure 5**).

You can change the order, ripple and crossover frequency again at this stage and observe the effects directly. You can set the presentation style using 'Plot Options' and at 'Markers' superimpose markers.

Next we use Simetrix to simulate the diplexer we designed and compare it with a Chebyshev lowpass filter.

A practical design

Following this work, I developed a PCB using EAGLE. Awkward coil and capacitor values were handled using available values

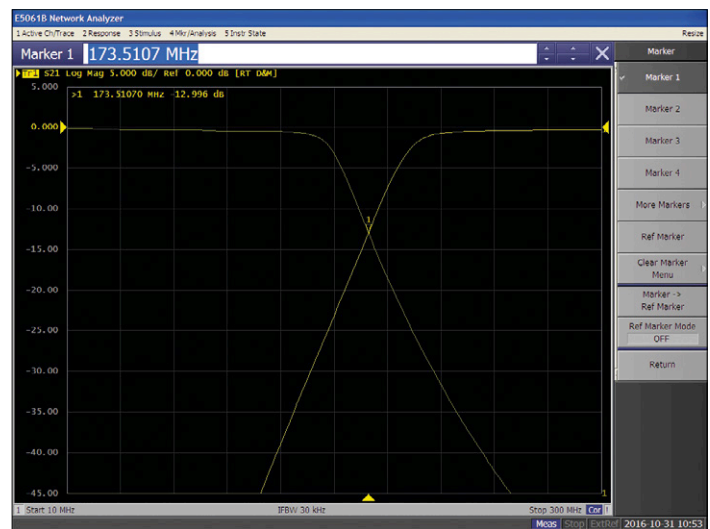


Figure 2. Lowpass filter SLP-150+ and highpass SHP-250+ by Mini-Circuits, each swept independently with a network analyzer.

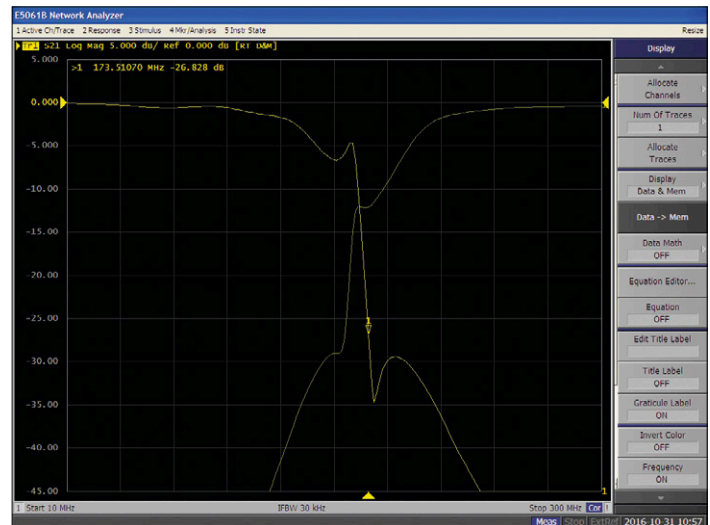


Figure 3. Both filters are connected in parallel here by T-adaptor. This creates a strong mutual influence on the frequency response.

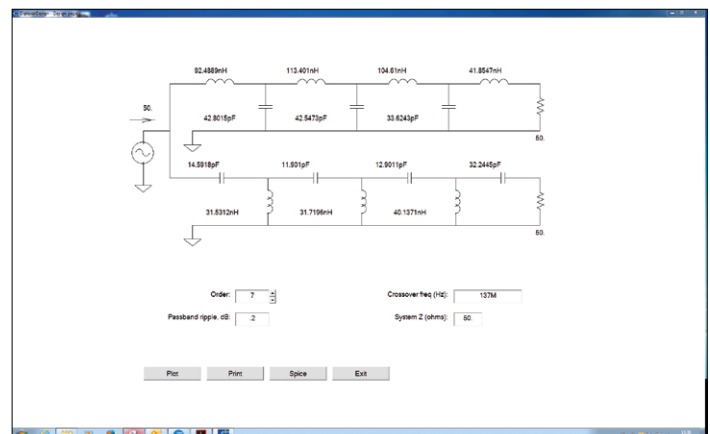


Figure 4. Design page of the Diplexer Design program.

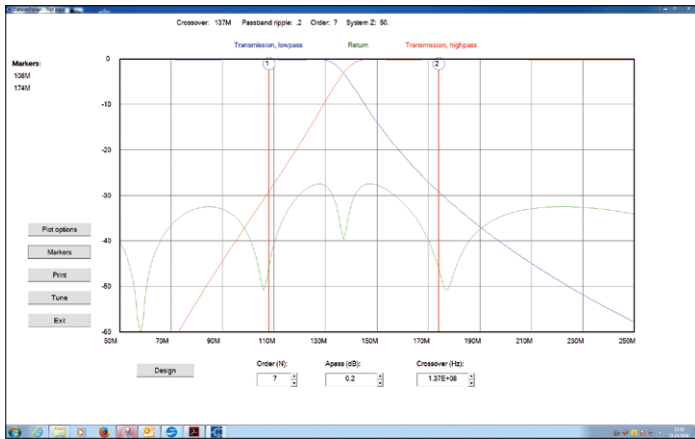


Figure 5. Plot page of the Diplexer Design program.

wired in parallel and series. It is possible to connect capacitors in parallel and install them side by side without any problems. Connecting coils in series is another matter. The two magnetic fields can interact, creating a transformer (albeit a very bad one), and the effective inductance value is not the desired sum of the two values. Accordingly the coils were offset by 90 degrees and, as far as the SMD layout would permit, separated from one another by distance.

The maximum order is 7. By omitting the capacitors and coils (substituting wire links) you can, however, also achieve every deeper order. The coils are available in 0603 packages.

The schematic of the practical circuit is shown in **Figure 6**, with the associated PCB in **Figure 7**. This PCB fits in a small plastic box and can be screwed in place (**Figure 8**). The input and output cables are soldered direct to the PCB. On its upper side there is an area of groundplane between the two filters; the underside is entirely groundplane.

The resulting measurements in **Figure 9** corroborate this design within the scope of the component tolerances. At approximately 45 dB attenuation the lowpass goes into 'saturation'. Although definitely undesirable, that's not functionally significant.

Active version

As the transmit power of DAB+ transmitters is rather modest in comparison with analog VHF stations, I searched for an appropriate preamplifier and struck rich with the MAX2630 from Maxim.

As shown in the datasheet [2], it comes in a 4-pin SAT143 package. The only extra circuitry, besides the bypass capacitor for the supply voltage, is one capacitor each at the input and output. Since it fits perfectly on a scrap of perfboard, no PCB had to be developed for testing. Its gain is around 15 dB and the noise figure is 4. The measured frequency response

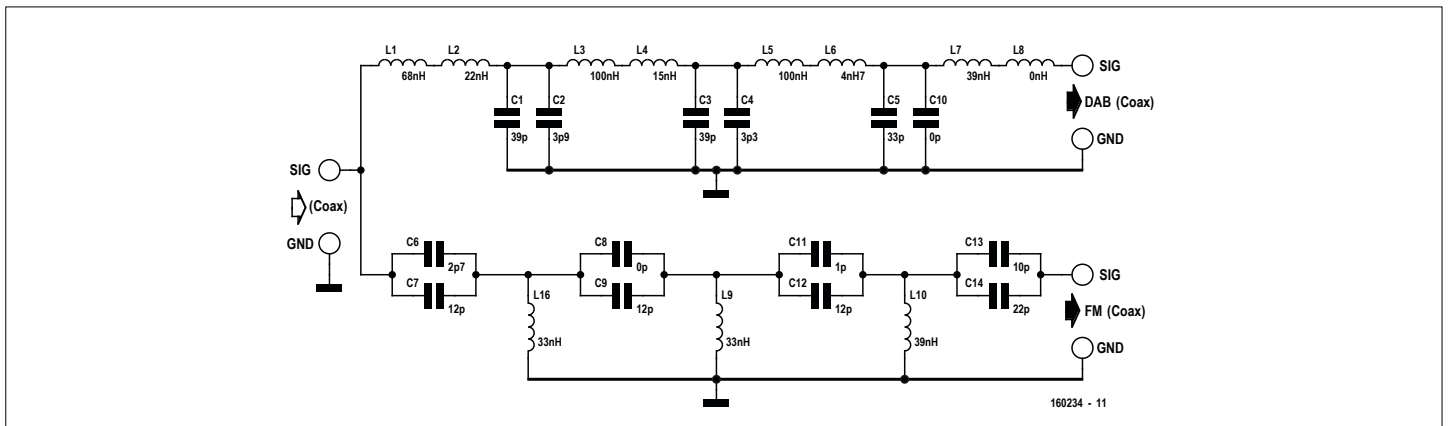


Figure 6. Schematic of the passive frequency splitter for FM/DAB+.

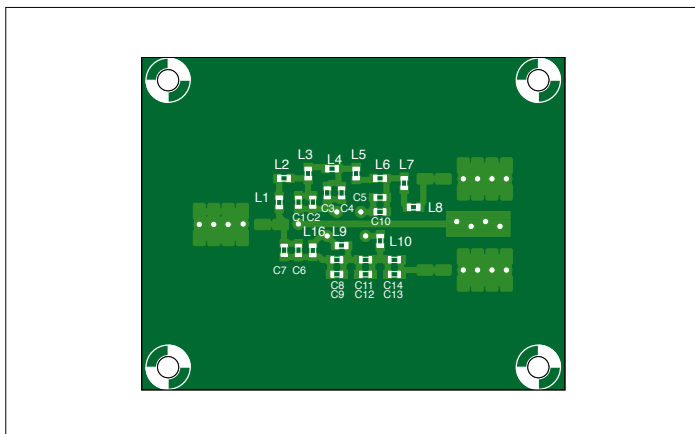


Figure 7. EAGLE layout for the circuit in Figure 6.

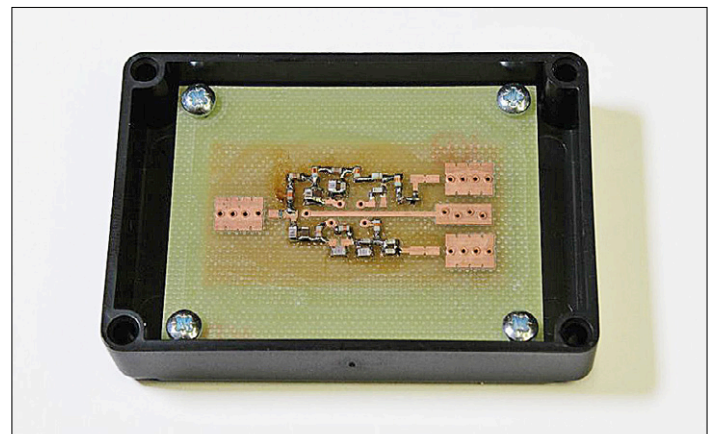


Figure 8. Prototype of the diplexer in its plastic housing.

runs in an almost straight line up to 250 MHz (and evidently carries on much further).

Encouraged by these results I developed an active diplexer: see **Figure 10**. The input and the two outputs are each fitted with a MAX2630.

The amplifier on the input probably makes sense only if a passive antenna is installed in the car. With the higher level from an active antenna, this could be too much of a good thing. You can then you can bridge the input and output of the amplifier with a wire link or an SMD zero-ohm resistor. The same applies on the FM output if the receiver is being overdriven. However, the amplifier on the DAB+ output probably does make sense in most cases.

The power is supplied by a 3.3-V voltage regulator, with each amplifier drawing about 7 mA. The series diode at the input prevents damage from incorrect polarity; the two resistors should reduce the thermal power dissipated by the regulator. The 12 V can be tapped off the car radio at the output provided for powering an active antenna. The PCB of the active diplexer (**Figure 11**) fits in a diecast aluminum box from Hammond (type 1550WQ).

(160234)

Web Links

- [1] www.tonnesoftware.com/diplexer.html
- [2] <https://datasheets.maximintegrated.com/en/ds/MAX2630-MAX2633.pdf>

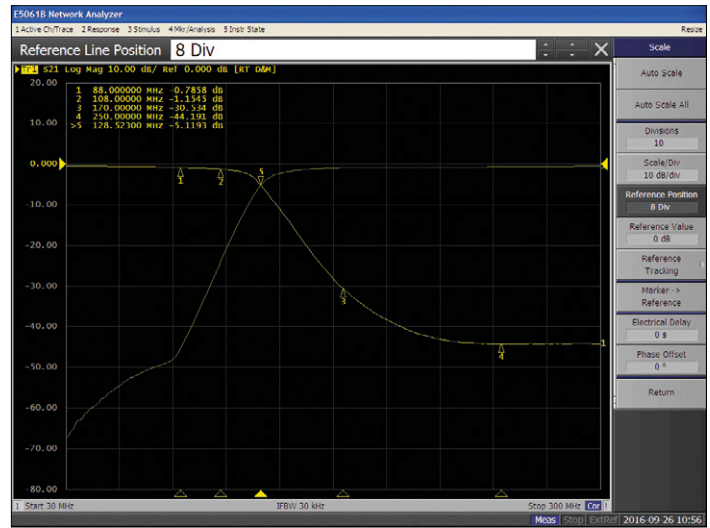


Figure 9. Frequency curve of the diplexer with markers.

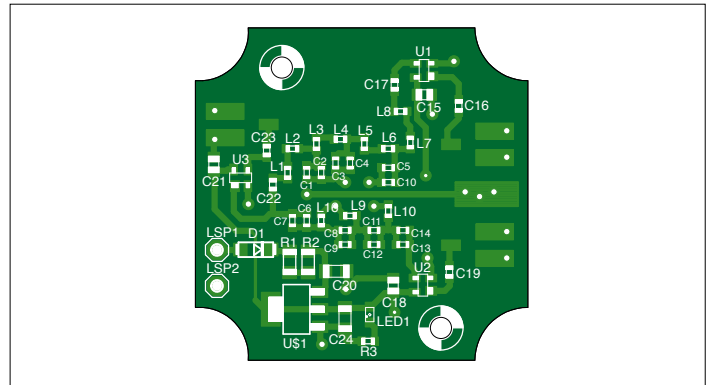


Figure 11. PCB layout for the active diplexer.

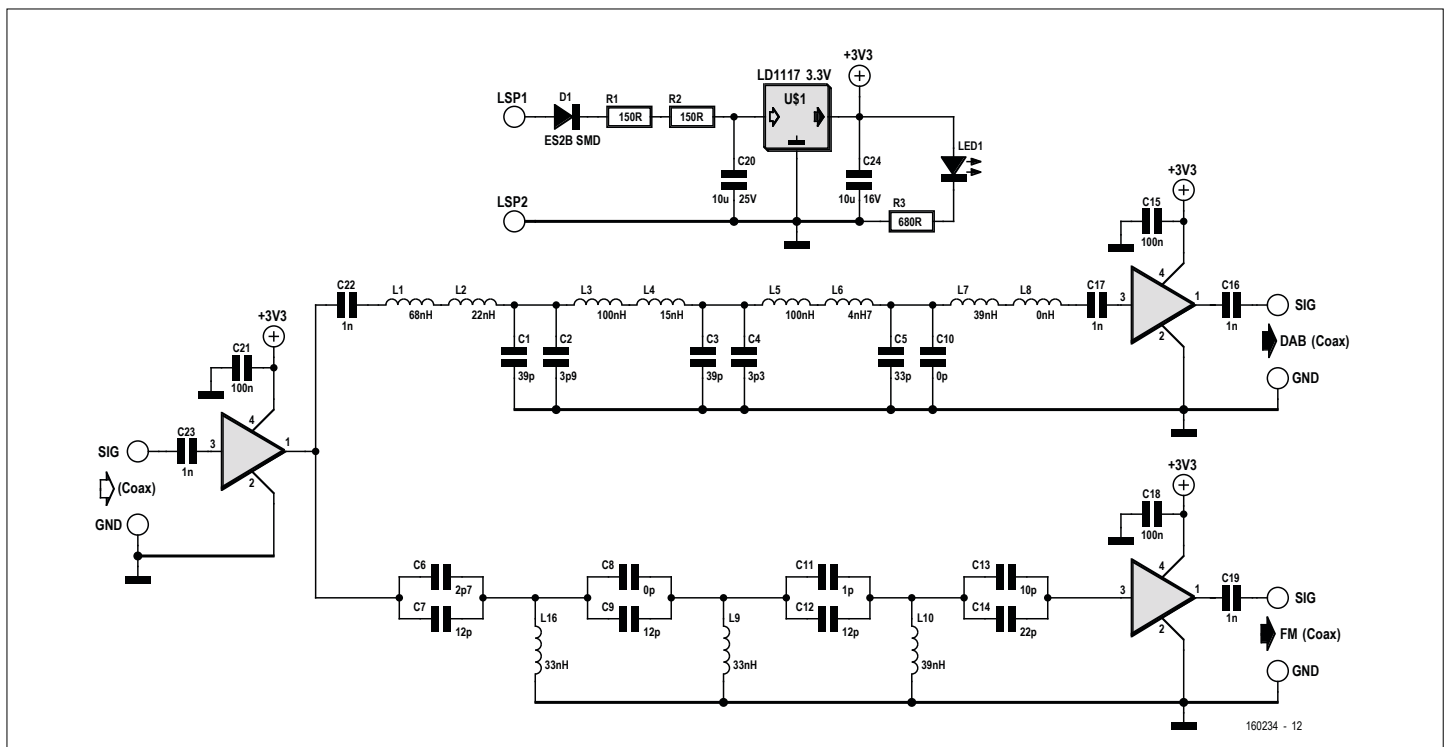


Figure 10. Schematic of the active diplexer.



welcome in your **ONLINE STORE**

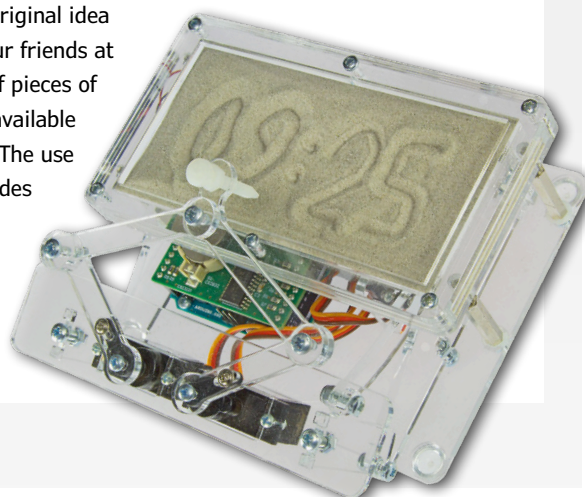
EDITOR'S CHOICE



Built around an Arduino Uno, this cool gadget displays the current time of the day, not by showing digits or changing the relative position of a pair of hour and minute hands, but by plotting four digits into a layer of sand. After an adjustable delay two vibration motors flatten out the sand layer before the write cycle begins again. Three miniature servo motors control the arms of a pantograph assembly. Mechanical parts of the Sand Clock (a refined project based on an original idea from our friends at

Make Magazine) mainly consist of pieces of acrylic sheet, a material readily available and well suited for laser cutting. The use of metal gear servo motors provides high accuracy with acceptable slack.

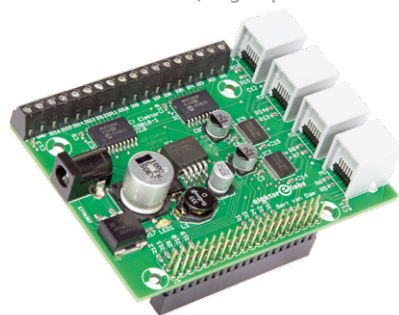
Denis Meyer
Elektor Editor



www.elektor.com/sandclock

Elektor Bestsellers

1. LEGO Control Board for Raspberry Pi
www.elektor.com/lego-rpi-board



2. Red Pitaya for Test and Measurement
www.elektor.com/red-pitaya-book

3. Elektor Uno R4
www.elektor.com/elektor-uno-r4

4. CAN Projects with ARM and Arduino
www.elektor.com/can-projects

5. BBC micro:bit
www.elektor.com/microbit

6. Mooshimeter
www.elektor.com/mooshimeter

BBC micro:bit - 35 Touch Develop & microPython Projects



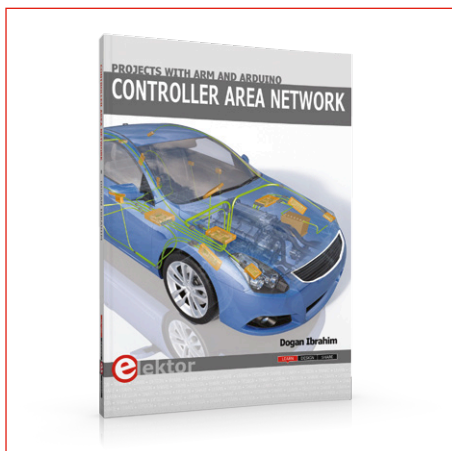
NEW

This book is about the use of the BBC micro:bit computer in practical projects. The BBC micro:bit computer can be programmed using several different programming languages. This book makes a brief introduction to the Touch Develop programming language and the microPython programming language. It then gives 35 example working and tested projects using these language. Complete program listings are given for each project.

member price: £20.95 • €22.45 • US \$25.00

www.elektor.com/microbit-book

CAN Projects with ARM and Arduino

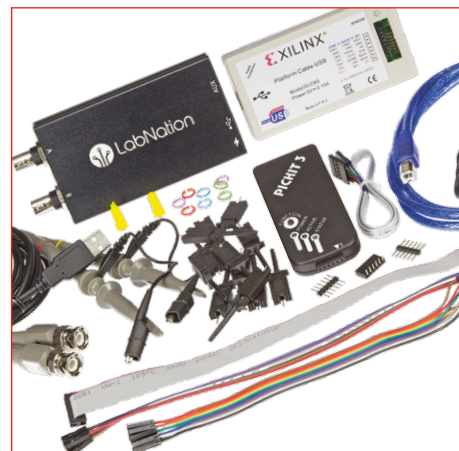


This book details the use of the ARM Cortex-M family of processors and the Arduino Uno in practical CAN bus based projects. Inside, it gives a detailed introduction to the architecture of the Cortex-M family whilst providing examples of popular hardware and software development kits. Using these kits helps to simplify the embedded design cycle considerably and makes it easier to develop, debug, and test a CAN bus based project.

member price: £28.95 • €33.75 • US \$38.00

www.elektor.com/can-projects

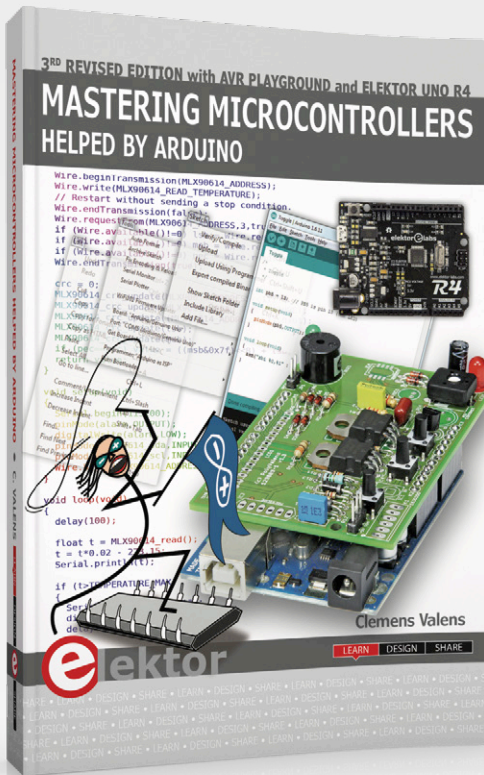
SmartScope Maker Kit



The new SmartScope Maker Kit (available exclusively from Elektor) contains a version of the SmartScope that's specially prepared for FPGA development, with ready assembled internal headers and additional FPGA pins that permit reading and writing. The remainder of the kit contents, including 2 programmers and 2 probes, is also highly worthwhile.

member price: £242.95 • €269.95 • US \$300.00

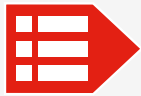
www.elektor.com/smartscope-maker-kit



Mastering Microcontrollers Helped by Arduino

**3RD revised edition
with AVR Playground and Elektor Uno R4**

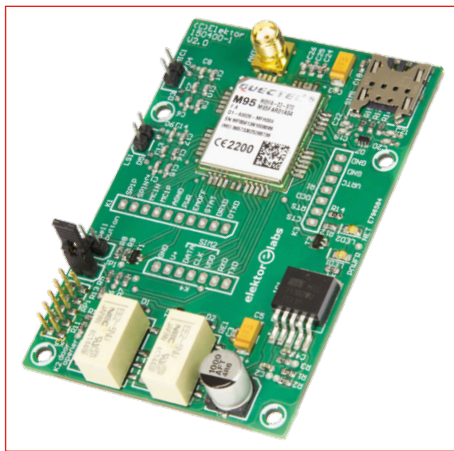
This third, extended and revised edition of Mastering Microcontrollers Helped by Arduino will not only familiarize you with the world of Arduino but it will also teach you how to program microcontrollers in general. In this book theory is put into practice on an Arduino board using the Arduino programming environment. Having completed this fun and playful course, you will be able to program any microcontroller, tackling and mastering I/O, memory, interrupts, communication (serial, I²C, SPI, 1-wire, SMBus), A/D converter, and more. This book contains two new chapters: AVR Playground and Elektor Uno R4.



MEMBER PRICE: £32.95 • €36.50 • US \$41.00

www.elektor.com/mastering-microcontrollers-3

Door Spy with Raspberry Pi



This GSM HAT-board for the Raspberry Pi can be used in combination with a Pi camera to build a hi-tech doorbell. When a visitor rings your doorbell, the camera takes a picture of him/her and sends it in an MMS-message to your cell phone. You can start a telephone conversation via the built-in intercom, and you can even send an SMS to open your front door.



£52.95 • €58.46 • US \$65.00

www.elektor.com/rpi-door-spy

Red Pitaya for Test & Measurement



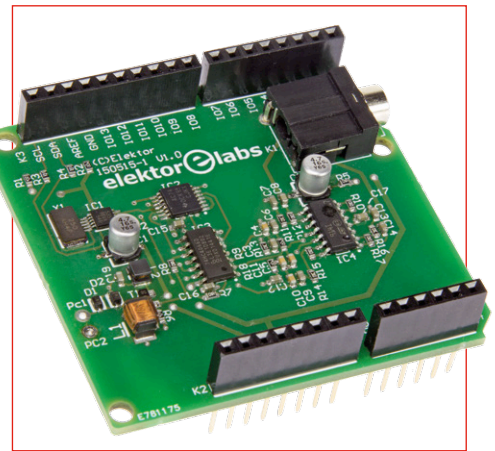
The Red Pitaya is a credit card-sized, open-source test and measurement board that can be used to replace most measurement instruments used in electronics laboratories. This book aims to teach the principles and applications of basic electronics by carrying out real experiments using the Red Pitaya. Many fun and interesting experiments are included. The book also makes an introduction to visual programming environment.



member price: £26.95 • €31.45 • US \$36.00

www.elektor.com/red-pitaya-book

Elektor SDR Reloaded



This Arduino shield is a remake of our famous SDR project published in 2007. A Software Defined Radio is a universal tool in RF technology circles, one that can also be put to use for making measurements. The characteristics of the receiver are defined in software, which now gives us the opportunity to use an Arduino Shield as a front-end.



member price: £98.95 • €26.96 • US \$144.00

www.elektor.com/sdr-reloaded



Welcome to the **SHARE** section



By **Thijs Beckers** (Elektor Netherlands)

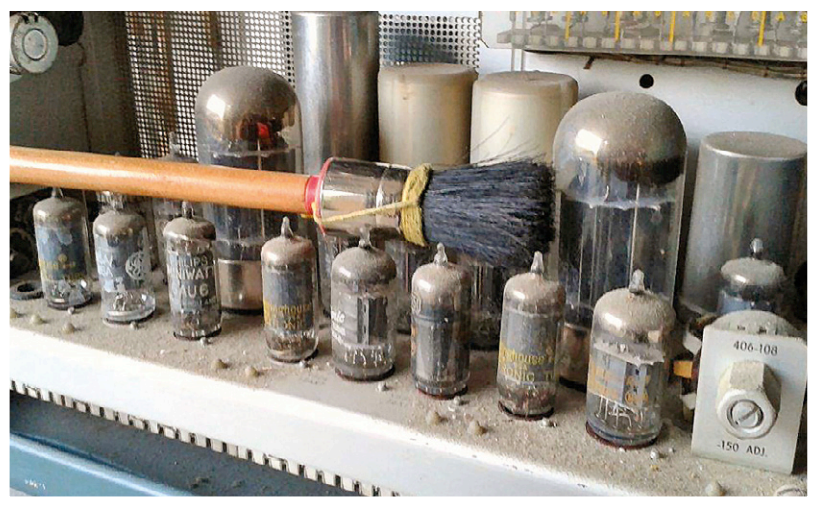
The following remarkable story was shared with me by one of the designers in the lab. He had problems at home with his hot-water heater. The following was happening: whenever warm tap water was required, there was a brief period of hot water, followed by a long period of cold water and finally the water became warm again and then stayed warm... Since this was a rented heater, there was no need to think about this too long and a repairman was called. The fault was reasonably quickly diagnosed. The water heater detected much too late that warm water was required and as a result was very slow to turn on. The first small quantity of warm water came from a smaller heater. Anyway, a new part needed to be ordered. But until then — and here it comes — the temporary solution: plug the AC line plug the other way around in the wall socket! (in Holland, earthed AC outlets are not polarized, *Ed.*)

This was more than a little strange, of course, because what is the difference with AC, you would think. The repairman explained that the problem *probably* was related to the ionization of the part, but he wasn't able to provide more details. We can imagine that ionization could indeed have an influence on the operation of the temperature or the flow sensor, but that turning the plug around would have any effect we find at least a little suspicious. This leads to the conjecture that somewhere in the electronics in the heater, part of a circuit has been economized away during the development. But that doesn't mean we are not interested in knowing what is happening exactly! If you know what is going on, write to us at editor@elektor.com and we will share it with the Elektor community. We are very curious.

Now something completely different. As the permanent *Retronics* editor, colleague Jan Buiting frequently comes into contact with tubes. Old tubes. *Such-thick-layers-of-dust-that-the-type-number-is-unreadable* tubes. So, what do you do? Exactly! Clean them. Only in that past the (white) ink used wasn't the best, so if you don't take care you will clean the number off right along with the dust!

Jan has by now developed a knack for this, which I would like to share with you:

Firstly: **never ever** wipe a cloth over the printing. You are sure to rub off the printing! Wipe only the dust off the other parts of the tube, this is usually without any problems. Now take a new, unused paint brush with long, soft bristles and use this to brush **carefully** back and forth



across the printing, taking very good care that you do not damage the lettering. It could easily take 50 back and forth sweeps before you see any results. Don't be tempted to use a toothbrush or similar for this; these are much too stiff and possibly damage the printing. A **soft** paint brush works the best. If necessary you can then clean the remainder of the tube with a damp cloth, optionally with a small amount of glass cleaner, but stay away from the printing. And done! Don't wipe the brush across the damp tube and the printing, otherwise there is the risk that you rub the latter off after all. ◀

(160252)

Simulation with SystemVision

A free web-based tool

Circuit simulation is a valuable tool for finding errors in circuit designs at an early stage. There are lots of Spice-based programs available for this purpose, but they are usually fairly expensive. Free versions are also available, but most of them are rather limited. Mentor Graphics recently launched SystemVision Cloud as a free browser-based simulation tool. We took this program out for a spin.

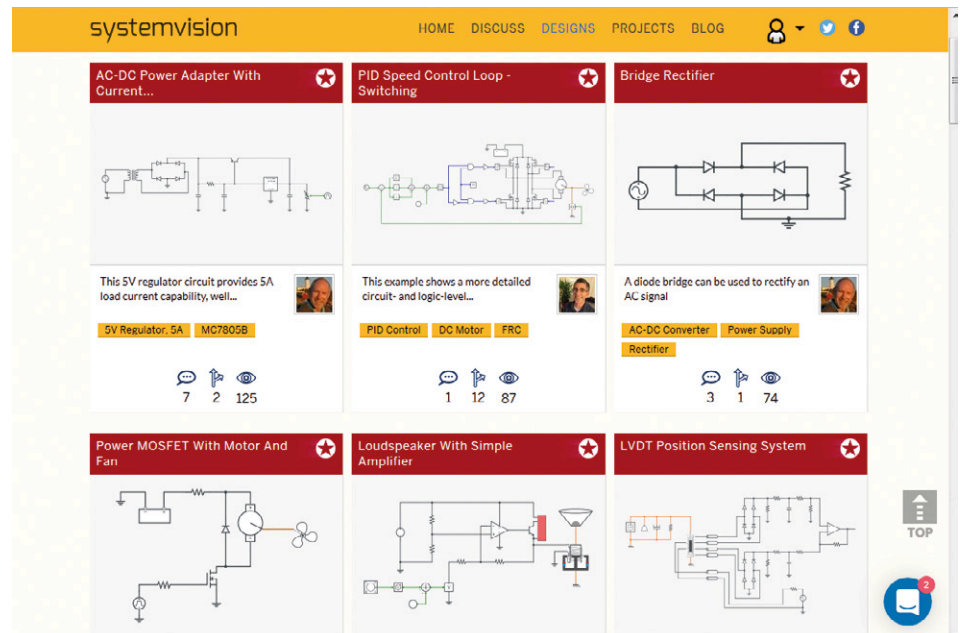
By **Tam Hanna** (Slovakia)

It's common knowledge that the earlier you find an error, the less it costs to correct. Computer verification of circuit designs is especially suitable for circuits without programmable components.

SystemVision from Mentor Graphics is a comprehensive simulation program which has been around for a good while already, but it is not free. SystemVision Cloud (**Figure 1**) was launched a while ago as a free version with basic functionality. The main difference between the free and full versions is that components generated with SystemVision Cloud are automatically shared with the community.

Getting started

For our needs, that should be enough to get started. Go to the website **www.systemvision.com** and click the *Log In* button to open the pop-up login window. At the time of publication of this article, SystemVision accepts not only traditional user accounts, but also accounts for users



who register for the service through their Facebook, Microsoft, LinkedIn, Twitter or Google accounts. The actual login process depends on which network you use. With

Twitter, for example, you see a pop-up window on the screen where you have to enter your Twitter address to give SystemVision access to your account.

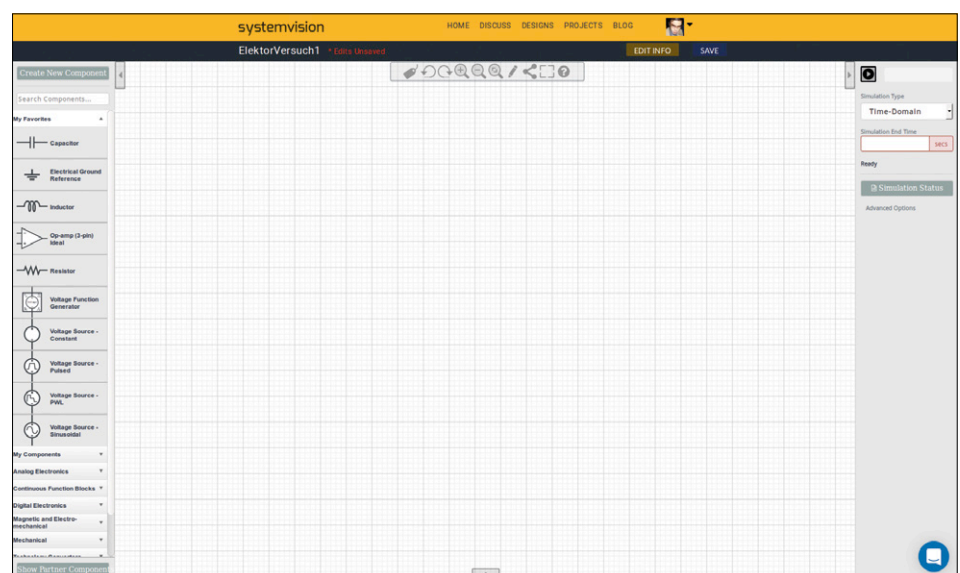


Figure 1. The SystemVision workspace.

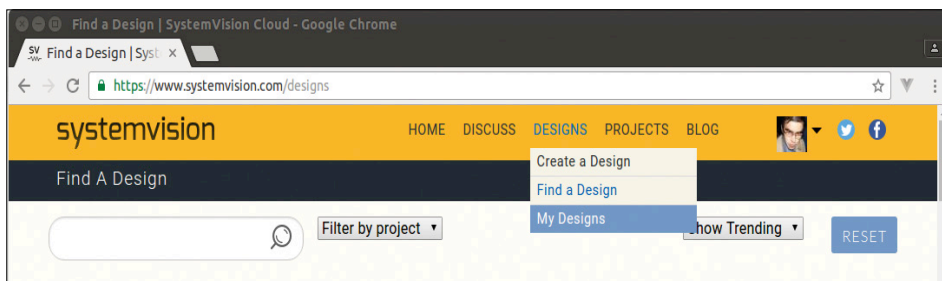


Figure 2. SystemVision makes extensive use of context menus.

Due to the strong social media focus of SystemVision, creating a new project is not entirely straightforward. Click on the yellow taskbar at the top of the screen and select the option *Designs* → *Create*

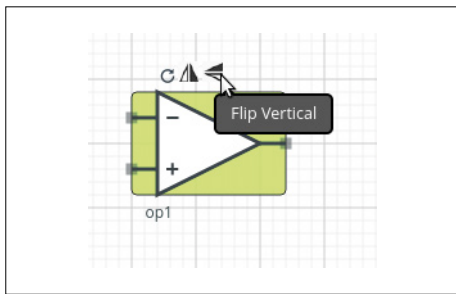


Figure 3. Components can be flipped and rotated as desired.

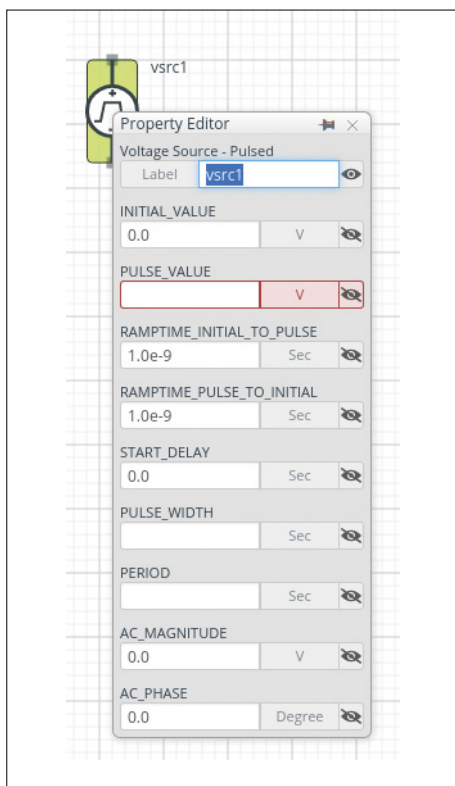


Figure 4. The window for entering component attributes is hidden away.

a *Design* (Figure 2).

SystemVision allows you to choose an existing design as the next step. If you want to start with a completely blank sheet, click *My Designs* instead. In the window that appears next, click the *Create* function. In response, SystemVision opens the Design view, which in turn fetches the wizard for opening a new design on the screen.

The program takes a while to load. Depending on your system configuration, it may go faster with Google Chrome or Firefox than with the traditional Internet Explorer. When you generate a design from scratch, you have to enter the title and select the visibility. If you select “Show only to me” in the *Visibility* field, you create a private design.

The user interface of SystemVision is fairly simple. On the left side there is a list of the various components available in SystemVision. They can be positioned on the drawing sheet by drag and drop. Options for flipping and rotation are shown in grey for currently selected components (Figure 3). You can use these options to adjust component orientations as necessary in your circuit.

You can wire components together by dragging and dropping lines between their terminals. To do so, position the mouse cursor on a component terminal, where it changes to a crosshair. Then drag a line to the destination terminal and drop it there to complete the connection.

As a first example, we decided to draw a simple RC circuit which demonstrates the familiar gradual voltage rise after the power is switched on.

First you drag and drop the necessary components onto the drawing sheet and

connect them in the usual way. Now comes the question of how you assign individual component values in SystemVision. The answer is that you click the label, which opens the dialog window shown in Figure 4. There you enter the desired parameter values and then click somewhere outside the dialog window to save the entered values.

Next you enhance the circuit as shown in Figure 5 to create the conditions necessary for proper simulation of the RC network. Here it should be mentioned that SystemVision basically works with units of ohms, farads and so on. You can change the order of magnitude by entering the following letters or letter combinations after the numerical values as appropriate:

- f
- p
- n
- u
- m
- K
- Meg
- G
- T

At this point alert readers are probably wondering why we chose a pulsed voltage source for this circuit instead of a constant voltage source. This seemingly illogical choice is due to a minor bug: if you power an RC network from a constant voltage source, you disable part of the simulation environment, and as a result you only see constant voltages instead of characteristic waveforms.

Super simulation

Before you can start the simulation, you have to declare a ground symbol. That’s because the simulation engine behind SystemVision only works when it has been assigned a reference potential. Then you click the *Simulation End Time* box on the *Emulation* toolbar at the right side of the screen and enter “5” for a simulation interval of five seconds. Then you click the arrow symbol at the top to start a time domain simulation. SystemVision can also do simulations in the frequency domain, but there is not enough space here to talk about that.

After the simulation run has finished on the Mentor Graphics computer, the program opens the results window shown in Figure 6, where the various quantities generated during the simulation are

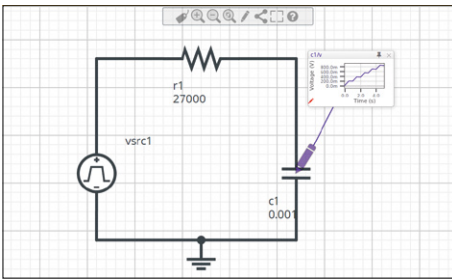


Figure 5. Our RC circuit is ready for simulation.

listed. Any errors in the circuit model are indicated by red bubbles on the screen, and the error messages are generally easy to understand.

made in the browser window when you click the *Save* button located on the previously mentioned yellow toolbar.

Social media and more

Although the simulation functions of SystemVision are very attractive, its main feature is the ability to share designs with your friends and colleagues. When you click the *Share Design* button, your design is released for sharing and SystemVision displays a URL (for example, <http://sysvis.io/smKBn9>) which you can share with other people.

The free basic version of SystemVision allows up to five private designs. If you need more you have to buy the profes-

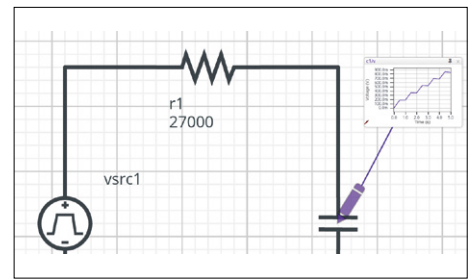


Figure 7. You can use the pencil symbol to create a watcher.

immense functional scope which can only be touched on in a single article such as this.

▶ The main feature is the ability to share designs with your friends and colleagues

If you want to save yourself the trouble of constantly digging through the results folder, you can use the pencil symbol to create a watcher as illustrated in **Figure 7**. A watcher is a component that hovers above the circuit and displays the results of the current simulation run.

At the time of publication of this article, SystemVision had an annoying bug in the simulation system: when you launch a second simulation, an additional folder is created in the simulation window. For this reason it is advisable to delete diagrams and simulation components that are no longer needed, by right-clicking them and selecting *Delete* in the context menu. Another minor irritation is how changes are saved: SystemVision only saves changes to the circuit diagram

sional version, but you can also stay below the five-design limit by deleting designs that are no longer needed or releasing them for sharing. Unfortunately, clicking the *Upgrade Account* button only leads to an email address. No price information can be found on the Internet.

If you are looking for inspiration, you can browse through the circuits provided by other designers under the *Find a Design* heading. That is especially helpful when you want to learn more about a particular component or you are looking for tips on how to set up effective simulations.

Summary

Although SystemVision appears a bit complicated at first glance mainly due to the cloud-based user interface, it offers an

If you are interested in circuit simulation, you should definitely check out this program. Particularly with regard to usability, it is miles ahead of PSpice. ◀

(160203-1)

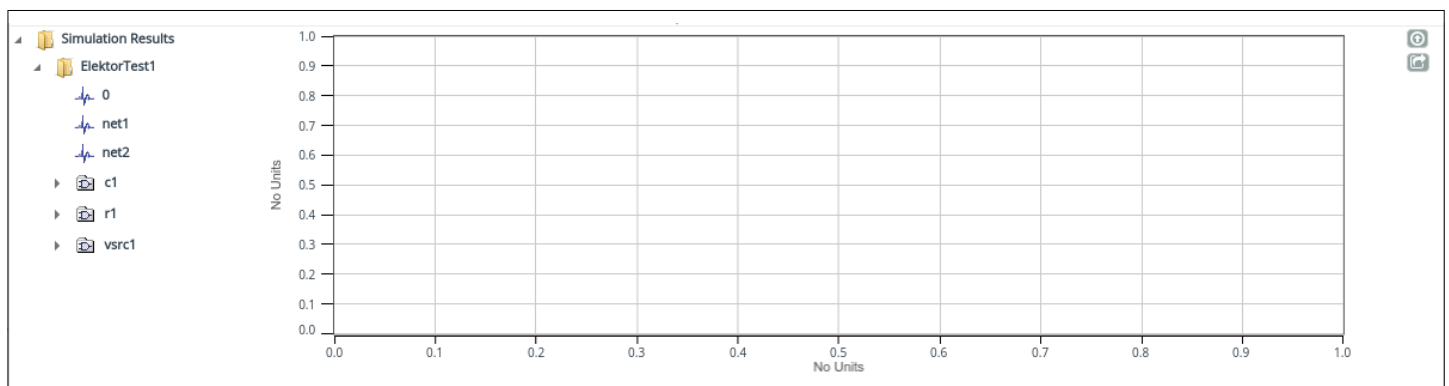


Figure 6. The simulation results have arrived.

Opamp Power Supply Connexions Capricious or logical?

What can go wrong when fitting a component on a circuit board? All sorts of things of course, but in this case the cause turned out to be really strange: The power supply connections of the IC were in the wrong place!

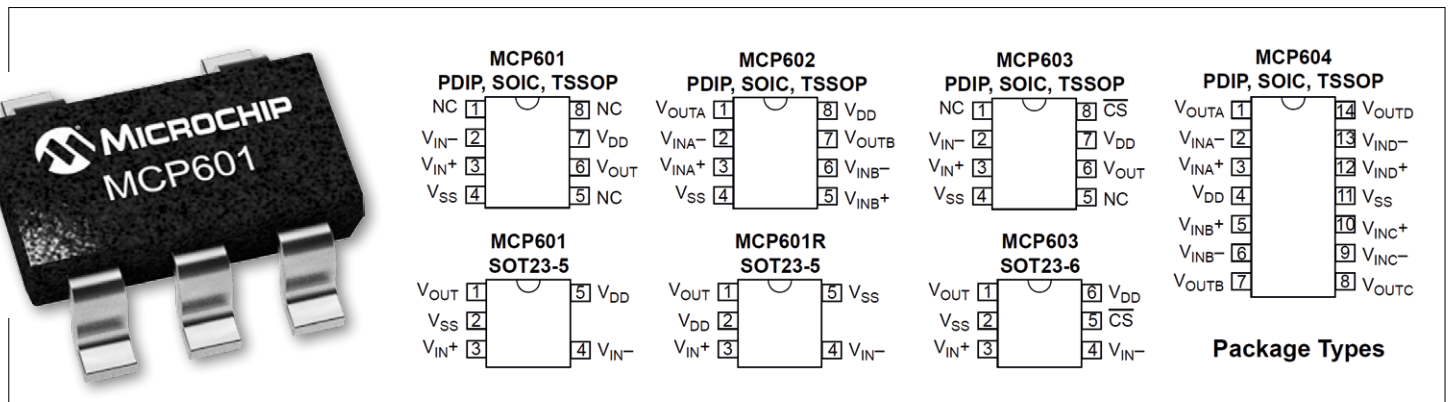
By **Luc Lemmens** (Elektor Labs)

During the testing of the first prototype of the *Tabula Tangibles* (a project that is a collaboration between Elektor Labs and Aachen Technical University), there was something seriously amiss with the power supply: the output of the voltage regulator appeared to be short-circuited. The first thought is a soldering problem. An SMD board that's built by hand could, for example, have a solder

to not function properly, but one of those was very remarkable.

The true and tested method of finding such faults is very simple: Remove the components one by one from the circuit board and measure the resistance across the power supply after each one, until the culprit has been found. Fortunately there are not that many components on this board and I also had the good fortune that after unsoldering the first IC the short-circuit disappeared! This was

turer of the device in hand in order to be 100 percent sure of the correct pin-out. With ICs, the suffix after the part number is usually an indication of the specifications or the type of package, but that an *R* on an IC means that the supply connections are reversed, that's new to me! I can't think of any plausible explanation why, as a manufacturer, you would produce both these variants; the datasheet doesn't mention a single word about this either.



bridge between the pins of an IC, or something. This is generally not a big deal and is easily fixed with some solder wick. However, a thorough visual inspection of the board didn't reveal any problems. An additional check of the schematic did not bring anything suspicious to light either. Neither did another check of the footprints and the connections of the components. Nevertheless there was a suspiciously low resistance between the output of the voltage regulator and ground, there had to be 'something' happening here. In the end there appeared to be two things that caused the circuit

a single opamp type MCP601 from Microchip in a SOT23-5 package, so back to studying the datasheet once more and what turns out to be the case: there are two variants of this IC in this particular package — the 601 and the 601R, with the difference that the latter has the power supply connections V_{DD} and V_{SS} the other way around than planned on my PCB...

I have come across many surprising things in electronics. For example, consider the BF254 or BS170P FETs, where you need to know the actual manufac-

As already mentioned, there was something else wrong (incorrect step-down converter for the desired voltage range), so that is why I haven't investigated this phenomenon any further at the moment. But after I promised the Editor to write a *From the Labs* article about it, I found it so improbable that I measured it again, just to be doubly sure. But it is really so: there are two variants. Whoever understands the logic behind it may elucidate us. ◀

(160257)



New Precise Nixie Clock

Elektor 3/2016, p. 56 (150189)

UPDATE. A case for this unit includes six LEDs and three resistors to give cool background illumination effects. A circuit diagram to assist in wiring the lights is given on page 4 of the manual which can be downloaded from:

www.elektor.com/acrylic-glass-case-for-six-digit-nixie-clock-150189-72

Err-lectronics

Corrections, Updates and Feedback to published articles



LEDs replace the fluorescent tube

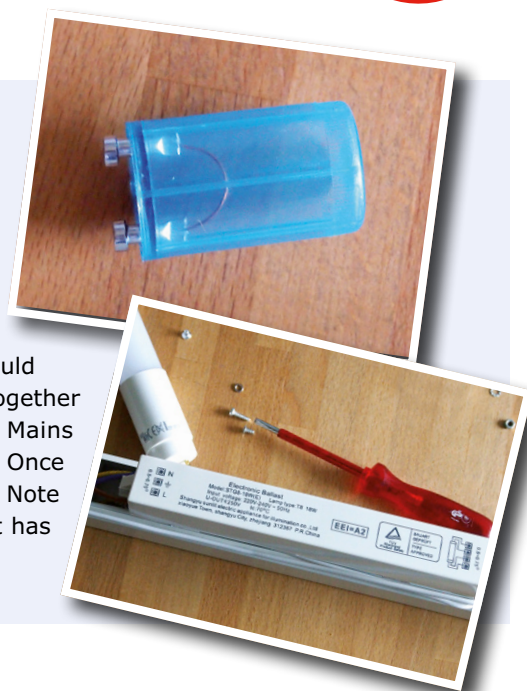
Elektor 1/2016, p.123 (Err-lectronics)

FEEDBACK. Further to a letter in Elektor 3/2016:

I finally got round to testing two replacement LED strip lights or tubes in a bathroom cabinet fitting which had two fluorescent tubes connected in tandem (in series). My conclusion is that LED tubes are not suitable as a direct replacement in this series-connected application.

Even with the two LED tubes wired in series without a ballast they will light but one or both tubes will have reduced light output. A competent electrician could relatively easily rewire the light fitting so that both LED tubes are connected together in parallel and both receive full mains voltage when the light is switched on. Mains voltage is lethal so it's important to ensure all necessary precautions are taken. Once the unit has been rewired the ballast and pseudo starter can be dispensed with. Note that conventional fluorescent tubes can no longer be used in the fitting once it has been rewired for use with LED tubes.

Robert Schrott



ELF Receiver

Elektor 10/2014, p. 8 (140035)

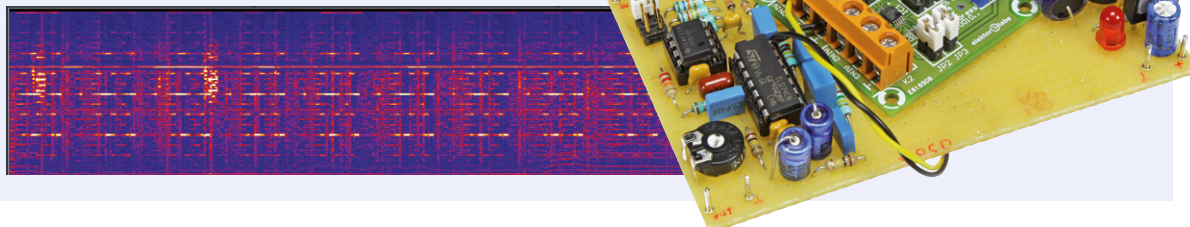
FEEDBACK. In addition to my participation in lectures at the University of Jena I am also actively investigating low frequency (electromagnetic) vibrations. So far the frequency range of my investigations has been between 3 Hz and 1 MHz.

I am also interested in lower frequency signals below 1 Hz. I went on to build the Elektor ELF receiver and the resulting investigations are very interesting. I was surprised to detect signals in the frequency range from 0.01 to 0.02 Hz. I use a wire antenna 8 cm to 17 m in length so these signals are more likely to be electrical rather than magnetic in nature. So far I have been unable to find any literature describing these signals. Also I have not had any satisfactory explanation as to their origin at the University of Jena.

Maybe geological vibrations of the earth are the key but I don't know the mechanism involved where (mechanical) spherical or toroidal vibrations could induce these electrical signals. The frequency range is maybe about right?

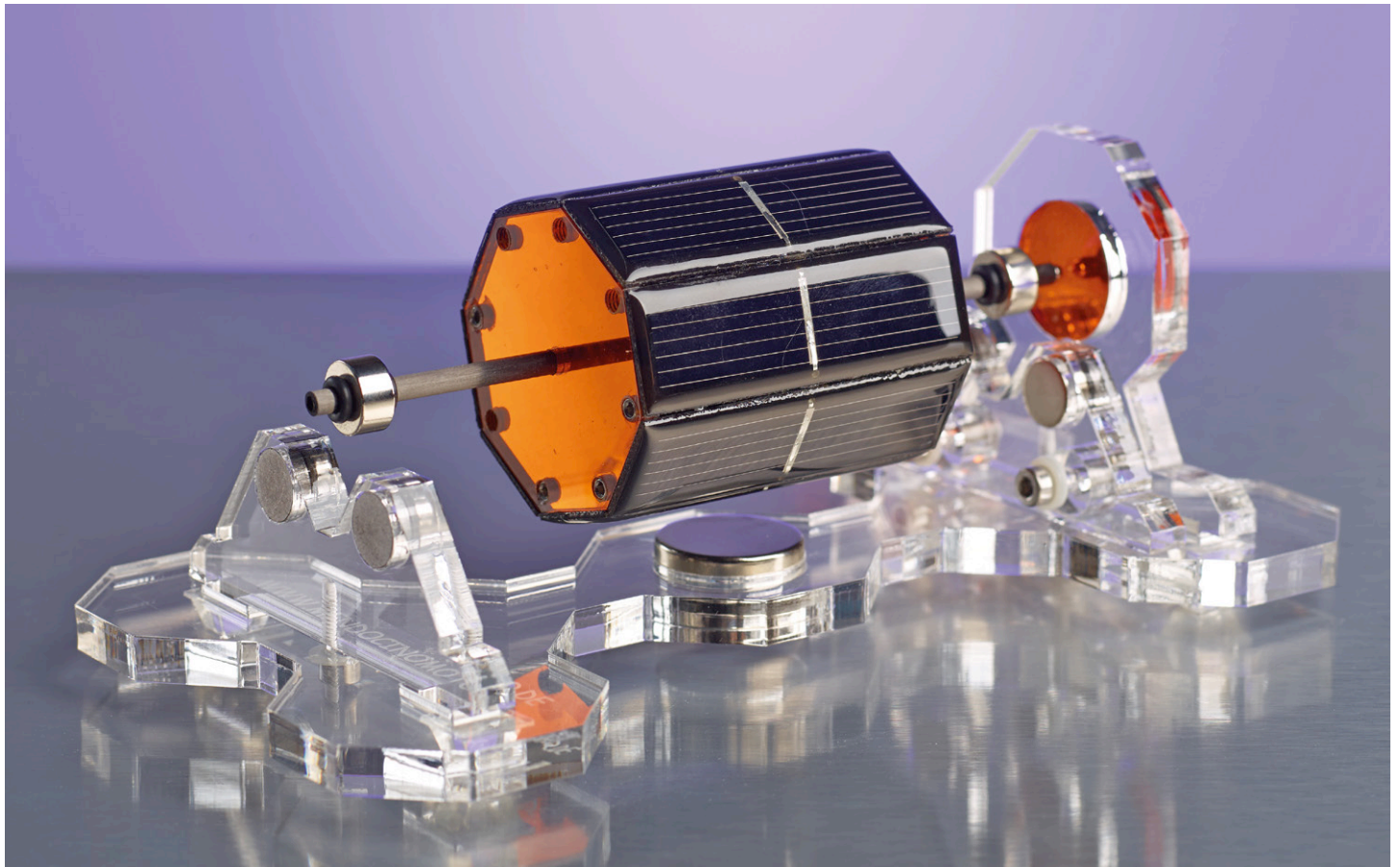
I wonder if any other reader has had a similar experience.

Walter Koch



The Mendocino Motor

It spins and sways by solar power alone



By **Manfred Klose** (Germany)

At first sight it looks like something that came out of a wizard's studio — hovering in mid-air and spinning on its own, apparently without any source of energy. The first time you see a Mendocino motor you can't help being fascinated. The operating principle is really very simple, a well-designed and neatly-finished model adds to the intrigue and makes the effect even more magical.

A Mendocino motor is a type of electric motor powered by energy produced by solar cells. It uses a rotor suspended on low-friction magnetic bearings. The motor was invented by Larry Spring who comes from Mendocino County in California.

The motor consists of a rotor and frame. The rotor is an eight-sided cylinder with an axle passing through its centre. Circular magnets are fitted at each end of the axle with the same magnetization direction as magnets fitted to the frame (**Figures 1 and 2**). This arrangement acts as the rotor bearings while the magnets exert a repelling force on one another to

support the rotor in mid air. They are also slightly offset to keep the rotor in suspension. According to Earnshaw's theorem it is not possible to achieve static levitation using any combination of fixed magnets and electric charges. To make this bearing arrangement stable, one end of the axle is terminated with a ball bearing which is pushed against a plate to form a low-friction contact. This prevents the rotor being pushed out of its position in the frame by the magnets.

To illustrate the motor principle we will take a closer look at the X-8 type of motor which was built by the author and is now

available to order from the Elektor Store. The frame of this motor is made from transparent acrylic sheet. A powerful neodymium magnet is positioned centrally beneath the rotor. Its magnetic field permeates the rotor coils above it. A small current flows through the coil when light falls on the illuminated solar cell. The Lorentz energy produced in the coil gives the rotor a pulse to make it rotate. The arrangement of coils and solar cells on the rotor ensures that pulses of energy will keep the rotor spinning, so long as the light intensity falling on the cells is sufficient.

The Lorentz force is very weak so the low-friction magnetic bearing is necessary in the design. Because the forces involved are so small it's important to reduce any static imbalance in the rotor also so that the mass inertia can be overcome and the rotor can start turning on its own when the light level is sufficient.

Construction of the Mendocino motor

Altogether there are eight solar cells arranged edge to edge to form the rotor. This gives the rotor an octagonal cross section. One cell will be illuminated while its opposite number will be in the shade. The two cells are wired in series which would seem to present a short circuit to the voltage generated (**Figure 3**). In practice however it doesn't cause a problem because one of the cells will be in the shade whilst its partner will be in full sun. Between the cell facing upwards and the cell facing downwards there is a difference in the incident light and also a difference in the current generated. The cell facing the light produces a voltage while the cell in the shade acts as a relatively high impedance. Current therefore flows through the coil. After the rotor has rotated through 180° the coil has turned around so that the energy pulse supplied by the opposite cell (which now faces the light) is of the correct polarity to reinforce the rotor's spin.

For this motor we use 65 x 20 mm sized solar cells which have a maximum output voltage of 0.5 V. The magnetic field strength produced by a coil can be calculated as the product of the number of turns and the current flowing in the coil. To get a high value of current flowing it's necessary to ensure that the coil does not have a high value of resistance. Each coil on our motor is made up of 60 turns of 0.3 mm diameter (AWG 28) wire. The coil winding has a DC resistance of around 5 ohms.

Current in the coil is given by the voltage divided by the resistance:

$$I = V/R = 0.5/5 = 100 \text{ mA.}$$

The solar cell specified here can supply almost 300 mA in full sunlight, yielding some reserve in the system. Solar cells show a steep drop in output voltage and current as the incident light level decreases. The use of a relatively high number of small narrow solar cells for the



Figure 1. The magnets on the axle are suspended by magnets fixed in the frame.

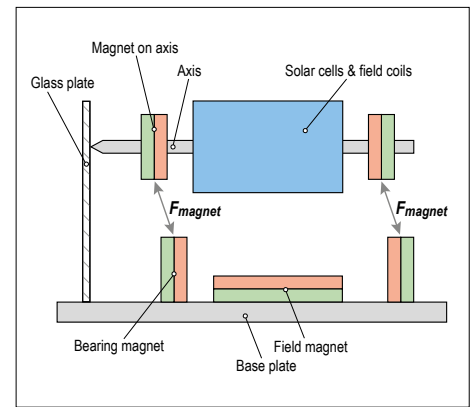


Figure 2. Sketch showing the Mendocino motor bearing arrangement.



The motor spins from the glow of a large tea light.

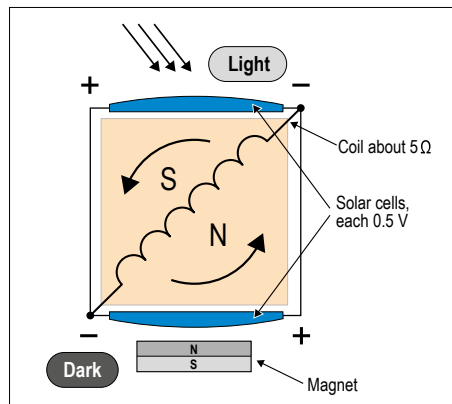


Figure 3. Two cells are connected in series which should produce a short-circuit condition but only one of the two cells will have light falling on it.

rotor produces a sequence of drive pulses which are close together, making rotation smoother and improving the motor's 'willingness' to start turning.

The coils in the rotor assembly fit underneath the solar cells (**Figure 4**) and are the same width as the drive magnet in the frame. They are positioned directly behind the solar cells and are therefore positioned optimally in the magnetic field. In contrast to other motor designs where the coils are positioned parallel to the cells and are visible, the wire length in this design is reduced which has a beneficial influence on current flow.

The ideal number of solar cells is eight. The cell which generated the preceding rotational pulse will have turned 45° away

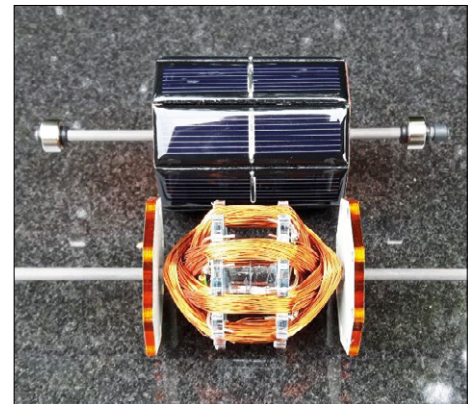


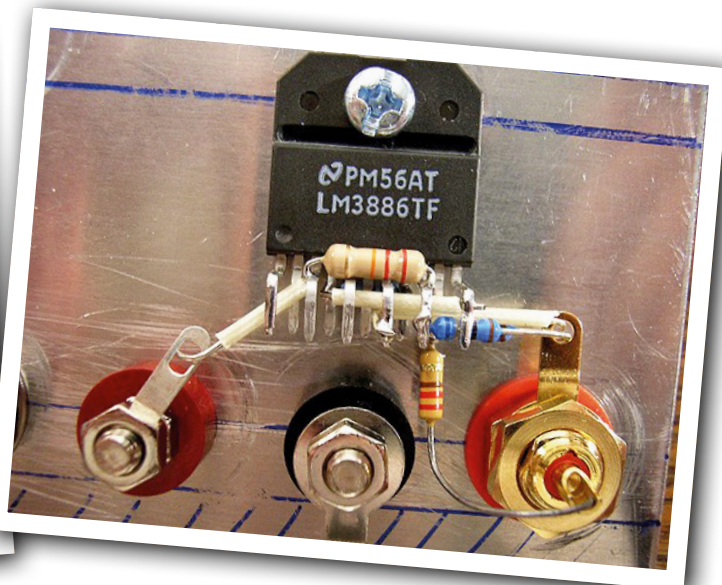
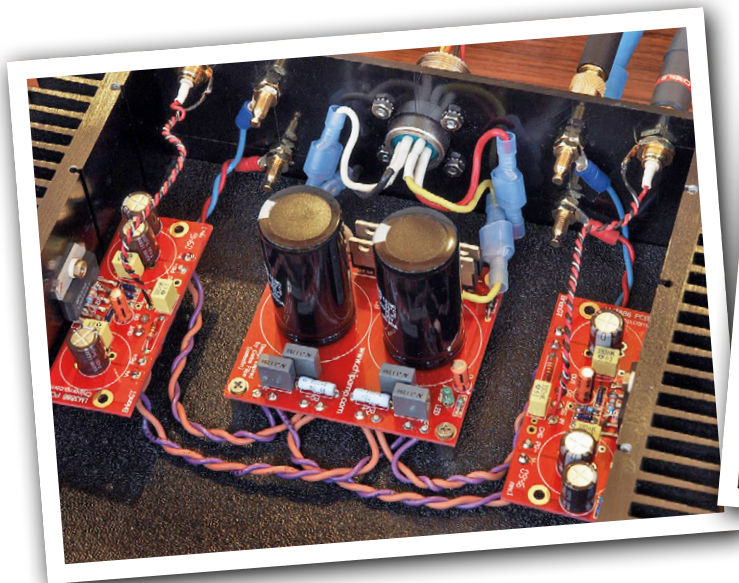
Figure 4. The coils are arranged inside the assembly of solar cells.

from the incident sunlight by the time the next pulse is provided by the following cell. The rapid decrease of energy from the preceding cell means that the energy it produces will not be sufficient to stop the rotor. Eight pulses per revolution ensure that the motor will spin even with relatively low light levels. In full sun you can expect it to get up to around 1400 rpm. What's more impressive however is that the glow from an ordinary tea light (with a flame height of around 2 cm) is enough to get the motor turning.

(160227)

Web Link

[www.elektor.com/
search?o=mendochino&q=mendocino](http://www.elektor.com/search?o=mendochino&q=mendocino)



Build a Chipamp!

Great sound with only a few components

The building and tweaking of audio equipment is something that occupies many enthusiasts. It is very satisfying and nothing beats listening to music on equipment that you have built yourself, even partially. In this installment of WebScouting we'll show you that you don't need a complex design, or a large number of components, to produce a fine sounding power amplifier.

By **Harry Baggen** (Elektor Labs)

It's certain that not all audiophiles are techies. Most of them aspire to get as faithful a sound reproduction as possible, and technical details don't interest them. They're perfectly happy to just listen and trust their ears. This has caused many strange products to appear in the audio world, with their effectiveness questioned by the techies, such as wooden supports for loudspeaker cables...

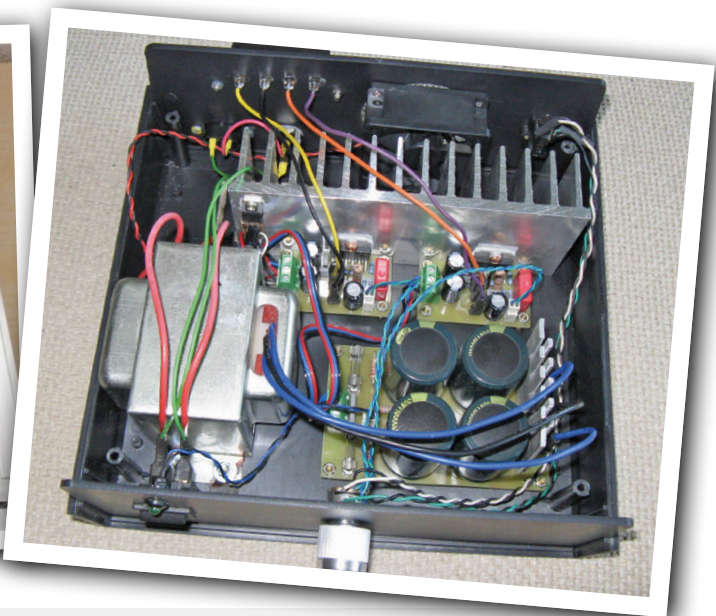
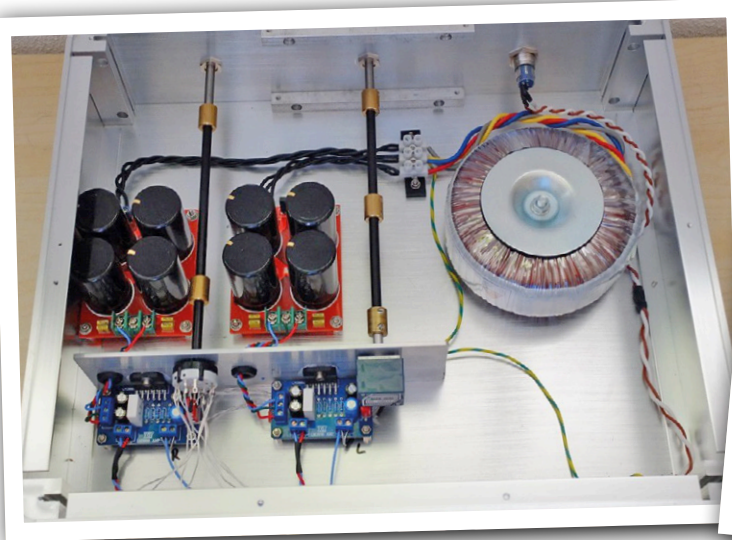
In contrast, when you're choosing components for an audio amplifier you can certainly work out why they would sound different. The choice of the amplifier configuration also plays a big role, of course. When I was searching the Internet I noticed that there were many web sites that dealt with so-called chipamps or gainclones. These are simple audio power amplifiers that use a power amp module. Most of these were ICs from the *Overture* series made by National Semiconductor, which has since been taken over by Texas Instruments.

The popularity of these amplifiers, which were really designed for general audio use and not specifically for Hi-Fi (let alone high-end) equipment, began in 1999 with the introduction of the 4706 (Gaincard) amplifier made by the Japanese manufacturer 47 Laboratory [1]. The aim of 47 Labs was to create the best possible sound reproduction by using as few components as possible in the audio signal path. This amplifier consisted of just nine components per channel, which had an LM3875 at its heart (an IC costing a few euros). In various reviews the 4706, which cost several thousand dollars, was praised to high heaven. As a result, several other well-known audio amplifier manufacturers started to use this IC in their amplifiers.

This has encouraged many hobbyists to experiment with the ICs from the *Overture* series. There are even several specialist web shops that sell only components and PCBs for these ICs. Chinese manufacturers have also jumped on the bandwagon, and offer a fully assem-

bled board for less than \$10. It uses the LM3886 [2], which is the most powerful IC from the *Overture* series that can output almost 70 W into 4 Ω . They also supply the required power supply boards very cheaply. At those prices you are beginning to question the quality of the components they've used...

The advantage of such a power amplifier is that it's very easy to experiment with different types of components, and of different qualities. All you need to add is a transformer, a bridge rectifier and several electrolytics, and you'll get some music. You can then find out what the sound is like when you use different types of electrolytics, or when you connect many smaller electrolytics in parallel, or when a different input capacitor is used, etc. A good description of several experiments using this LM3886 chipamp can be found on the (Dutch) website of Dick van de Merwe [3]. The website of DIY Audio Projects has several projects based on chipamps, such as the DIY LM3875 (which they call 'The Beast') [4] and the



DIY LM3886 Chip Amplifier (Gainclone) Kit [5].

It's even possible to build this circuit without the use of a PCB. An extensive set of photos that show you how it's done is shown on Mick Feuerbacher's website, Dogbreath [6].

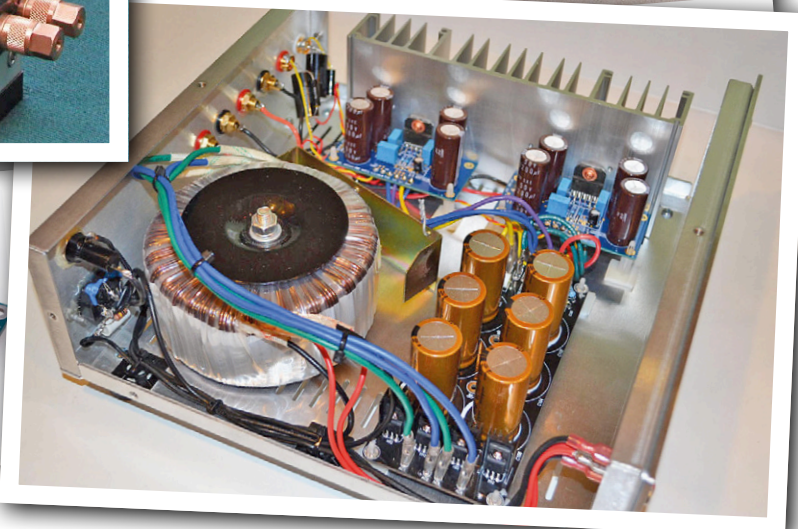
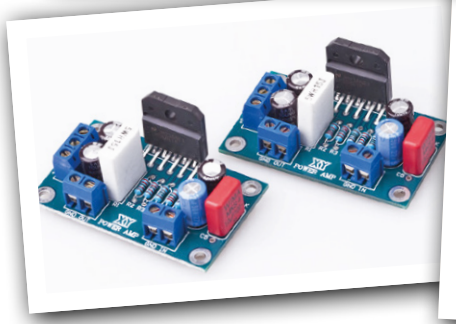
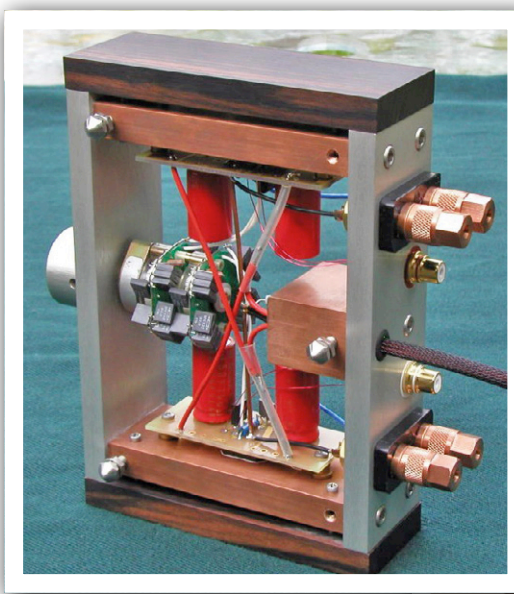
Of course, Elektor got in on the act as well, publishing a design for the LM3886 in the Bumper Summer Issue of 1998 [7], when the IC had only just appeared on the market. It even included a PCB layout!

There are many more similar designs to be found on the Internet, why not have a look? The bottom line is that all these designs use a single IC along with a few passive components. It's hard to believe really. Oh well, you can't argue about taste and sound quality! ◀

(160261)

Web Links

- [1] www.sakurasystems.com/reference.html
- [2] www.ti.com/product/lm3886
- [3] www.audio-creative.nl/audio-creative/diy-lm3886-chipamp/
- [4] <http://diyaudioprojects.com/Chip/Beast/>
- [5] http://diyaudioprojects.com/Chip/LM3886_CA/LM3886_CA.htm
- [6] <http://dogbreath.de/>
- [7] <https://www.elektormagazine.com/magazine/elektor-199807/34116>



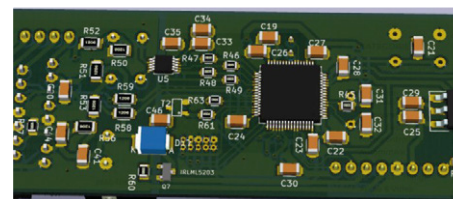
Elektor Labs Pipeline



The New Year has begun, and of course the (stale) list of resolutions compiled a few years back got dug up once more. Since these resolutions are never fulfilled anyway, why not replace some of them with things that are feasible, like doing some electronics projects? Here are a few suggestions to get you started.

Build a LOUD sound player

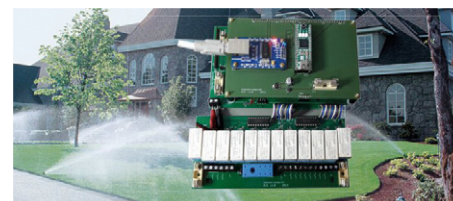
Searching for a powerful gong or audio notifier? The "Card Sound" board presented here consists of a TDA7266 audio amplifier featuring two 7-watt outputs, a CS4344 AD converter, an STM32F401 ARM Cortex-M4 microcontroller, and a slot for a micro SD card. The goal is to play different (high-quality) sounds stored on an SD card with enough volume to wake up even the deepest sleeper.



<https://goo.gl/yct0r6>

Let a Raspberry Pi water your lawn

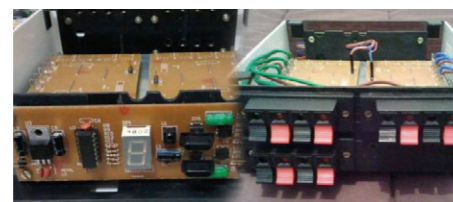
A little worked up because of precious time lost on watering his lawn when he could have been working on his electronics projects, the author started thinking of ways to automate this tedious job. A first prototype, a 12-channel sprinkler timer/controller, built with an intelligent PIC-based, touch-capable graphic display worked well, but sadly could not be controlled remotely. The second version is technologically more advanced with a PIC32, Ethernet, Windows 10 IoT, and a Raspberry Pi.



<https://goo.gl/JwND75>

Remote controlled speaker switch box

Some people sure would like to listen to a 5+1 surround audio system, but don't want to give up their stereo system. Sadly, both systems in one room would require eight speakers, which is too much for many, and also not necessary. Since both systems have speakers for the left and right channels the idea emerged to build a switch box which allows configuring the system as 5+1 or as stereo while only requiring the 5+1 system's speakers.



<https://goo.gl/wh6Jos>

Intelli-Cell smart battery analyzer

Here is a very smart and reprogrammable rechargeable battery analyzing instrument with a special mounting system to access the terminals of any cell phone battery (including Apple), as well as any other type of battery of any composition or voltage from 1.25 V up to 12 V. It has automatic polarity sensing, heat sensing both on the mounting plate and on the external cable accessory. It can be programmed easily for any composition (NiCd, NiMH, Li-Ion, Li-Fe, Lead Acid) and any amp-hour rating from 400 mA to 5000 mA.

The mounting plate includes fully adjustable pogo contacts with slides and elevator to access any battery contacts, no matter how difficult. The external cable has alligator clips and a sensing thermistor to reach larger batteries or for using the Apple adapter. ◀



<https://goo.gl/ar15x7>

(160253)

From Verobox to Heavy Metal

Elektor lab instruments 1980s and 1990s style

Roughly between 1984 and 1988, Elektor published several pieces of DIY laboratory instruments all housed in two-color Verobox, hard plastic cases with aluminum front and rear panels. A second gulf of benchtop instruments for many more applications was launched in 1989 and



marked by solid, metal cases and again a sort-of uniform 'look and feel' to them. I found a couple of both series in the Retronics attic and decided to take the family picture "in the box, 15 years on".

By **Jan Buiting**, Editor-in-Chief

Back in 1995, with time on hands before my return flight, I was given a private tour of the RadioSpares laboratories in Corby, Northants. That's right, it's today's headquarters of RS Components. It was at a time when the first CD-ROM version appeared of the *RS Catalog*, complete with a search function, which was a sensation although it proved unable to find 'BC547'. One function of the RS lab staff in Corby was to supply verified engineering data on components and systems, for use in the RS Datasheet Collection RS customers could subscribe to. All on paper of course and in luscious, dark red binders complete with tab sheets. To my amazement all five lab staff in the Components section had a full complement of Elektor test equipment published from about 1984 onwards as DIY projects. The instruments were easy to recognize from their Verobox housings with the distinctive grey and off-white color scheme! The techies at RS were very satisfied with the equipment and said it all worked okay once calibrated against "the unaffordable gear at head office". While it's tempting to think of one "Elek-

tor measurement series" in fact there were several. This article is my attempt at describing two, suitably identified by the boxes the Elektor lab workers agreed to use. Did they?

Product engineering

If you are among the countless readers who started out as an electronics hobbyist and now wield spreadsheets, \$K budgets and BOMs "for the industry" you will remember that the sheer gusto produced by *your* working circuit was often so great as to quench any desire to build the PCB into a case decent enough to show outside your shack. And indeed many Elektor projects published over the years are "done, finished, ready" when the autorouter says *zero errors* and the prototype board works. The fact that these days an increasing number of projects are microcontroller based does not seem to encourage any of the programmers out there to have a go at giving the project a professional look (and feel). They hate nuts and bolts.

A case for the case

Back in the 1980s when a team of Elektor lab workers and editors outlined their

plans for a series of affordable, high quality, DIY measurement apparatus, simply stopping at the PCB was not an option. Like true industrial designers, they started with the enclosure and the ergonomics of the controls, before even going to the library and opening the RS Components Datasheet Collection to find an opamp or a connector. Some designers argued that the prescribed enclosure was a restrictive factor to their creative genius but eventually followed suit. One size fits all — these instruments must have the Elektor look and be instantly recognizable!

In the beginning was Vero

The first more or less coordinated series was to be housed in the two-tone (two-color) "Verobox" sized 205 (w) x 137 (d) x 75 (h) mm. As an example for your amusement and *déjà vu* experience, I have pictured the **Elektor Pulse Generator** in **Figure 1**. The case consists of an off-white cover and a greyish bottom, both made from ABS (hard plastic). The front and rear panels are 1.5-mm mm thick aluminum sheet. I remember we sold ready-made screen-printed self-adhesive foils for sticking on the front panel (which you had to drill first). The word-



Figure 1. To kick off in ABS: Elektor Pulse Generator, mid 1980s. Sorry about the missing cap on the 0.1–1.0 Repetition Time variable control.

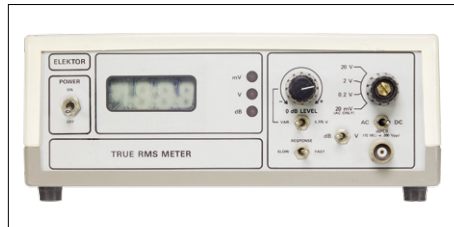


Figure 3. A rare bird and seldom used too as nobody cares any longer about the exact expression of analog voltages: True RMS Meter, again mid-1980s.



Figure 5. Over to Telet metal cases with the H_{FE} Tester from 1990.

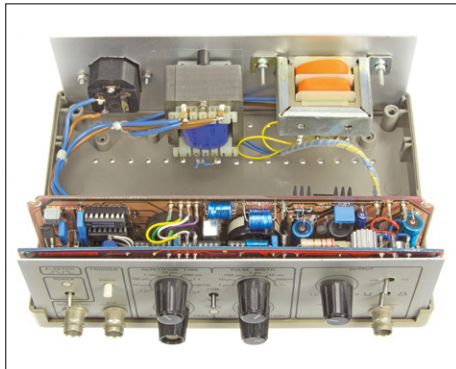


Figure 2. A look inside the Pulse Generator; note the vertically mounted boards and the two transformers rightly deferred to the rear panel.

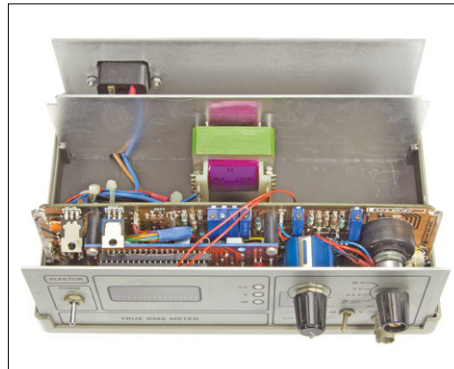


Figure 4. Inside the True RMS Meter. Note the range switch connected directly to the PCB.

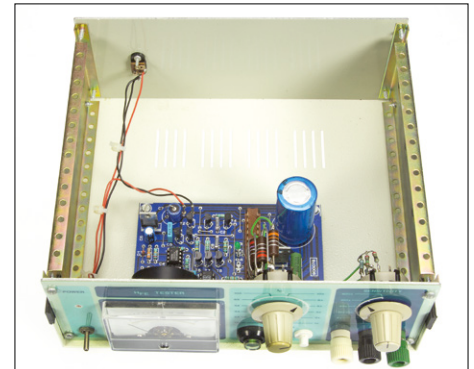


Figure 6. Plenty of space inside that Telet case!

ings and symbols used on the foil were fiercely debated between the four language departments within Elektor, and fortunately it was agreed to use a combination of (minimal) English and “fits-u-all” symbols. The name of the instrument remained a challenge though as you can see in Figure 1 (top left hand corner). The slots formed by pairs of protrusions at the short sides of the bottom half of the Verobox were used to secure printed circuit boards vertically. As shown in **Figure 2**, the slots allowed front panel controls such as potentiometers and switches to (a) assist in securing the “controls” board to the front panel and (b) to be connected directly to the PCB tracks rather than with flying wires as in “hobbyist” projects in EPE *et al.* LEDs and other indicators could shine through transparent clearances in the front panel foil. Everything to do with AC line voltage and power transformer’ing was allocated to the rear panel. Note that the molded screw bases in the bottom half of the Verobox were not used — all PCB orientation was to be vertical. And note those ventilation holes carefully drilled in the top and bottom sections by Jan Visser and/or Mr. Feron.

Another fine specimen of the Verobox T & M series was the **Elektor True RMS Meter I** stumbled on in the Retronics attic (**Figure 3**). Sorry about that LCD which went fuzzy over the years, I will send a complaint to Sharp. In this instrument the power transformer was mounted on a separate aluminum sheet, possibly because of its weight (**Figure 4**). Sadly I do not have the pinnacle of the Elektor Verobox T & M series in my collection, the **Elektor XR2206 Function Generator**. Rest assured though that it looks just like the other two instruments I was able to rescue from the skip back in 2006. That Function Generator is still seen today on auction sites and it’s a veritable classic.

From memory I recall several other instruments in the Verobox series, like a wattmeter and a multimeter but I do not have them.

That Verobox was expensive and Elektor got many complaints especially from super thrifty Hollanders. Some advertisers were thrilled though with the T & M series and started to supply kits as well as the infamous “elektorical” parts. In the UK, Maplin was the main supplier. In Holland, DIL Elektronica ruled the roost, in Ger-

many, Geist, while C-I Electronics served most non-European areas with mail order only. The upshot: Elektor achieved a massive success and these instruments are now legendary.

Italian metal is better

Like just about any supplier of test and measurement instruments, Elektor worked to innovate and refresh the product line and improve brand awareness. By about 1988, the Verobox got a bit long in the tooth and a new “standard enclosure” was sought by the Elektor team to house yet another glut of DIY instruments for use in the home lab. Still using the adagio: Affordable — Quality — Professional. Rumor has it that parts distributor Texim Electronics, a fantastic source of elektorical parts and located in a murky part of Holland called *Achterhoek* (“back corner”), “kindly advised” metal cases produced by Telet, Italy. As opposed to the Verobox, these Telet cases were (a) all-metal and (b) available in different sizes. They were accepted as the new standard.

Figure 5 shows a random example of Elektor’s “new” T & M series, the H_{FE} Tester, sized 198 (w) x 180 (d) x 80 (h) mm. Note the correct use of _{sub}scripts

on the front panel — a feat for Patrick Welders on the graphics & drawing staff! Also watch those two shades of Arduino green (some say: turquoise) used for the front panel foil. I hope the colors come out true on the photos.

Within the Telet series of instruments, boards were allowed to be fitted horizontally on standoffs, and controls secured directly to the front panel for wiring to the board (**Figure 6**). Note that in the case of the H_{FE} Tester the enclosure is *pretty vacant* as no internal PSU was incorporated and the instrument ran off a DC power adapter (well done guys, Safety First!).

What other Telet instruments from the Retronics collection?

- An AC line **Wattmeter** with LCD readout, switchable $\times 1$ and $\times 10$ ranges and the AC line out socket right on the front panel (**Figure 7**) (Telet 198 \times 132 \times 80 mm) [2].
- The **Watt-Hour Meter** from 1993 in the largest Telet case Elektor used in their series (Telet 297 \times 180 \times 80 mm) (**Figure 8**) [3]. Advanced for its time, at least in DIY land, it had a V24 serial interface to capture readings as well as do the calibration. The V24 connector is a 9-pin sub-D located on the back panel of the instrument.
- The **TV Pattern Generator** which I still use today to check vintage oscilloscopes for their trigger and delay capabilities (**Figure 9**) (Telet 247 \times 180 \times 80 mm). Here the LEDs are seated in holes and they protrude slightly from the front panel. This instrument is AC line powered.
- Finally, an **Inductance Meter** feigning to have nanohenry ranges (**Figure 10**). Admittedly, in the lowest nH range this instrument is beaten hands-down by our more recent 500 ppm LCR Meter.

Two other Telet-style projects are not shown here but should be mentioned also: a simple **15 V, 500 mA power supply** and the **Advanced LCR Meter** [4]. The latter I have available as a kit originally purchased from DIL Elektronika by a deceased Elektor reader. I still have to assemble this fantastic design by Hans Bonekamp, and pitch it against our recent 500 ppm LCR Meter which costs 2.5 dB more.

Not forgotten

With the exception of the True RMS Meter,

EST[®] 2004

www.elektor.tv



Retronics is a monthly section covering vintage electronics including legendary Elektor designs.

Contributions, suggestions and requests are welcome; please telegraph editor@elektor.com

all of the instruments discussed here actually work and are part of the *Retronics Classic Repair* initiative I started here at Elektor House. While I realize their joint functionality can be run in twenty or so clock cycles of a Red Pitaya board, or by a 15-year old running an Arduino or Raspberry Pi with suitable extension boards, these retro instruments are fun to use even occasionally. For example, the Watt Meter is permanently inserted in the AC line supply to alert me to boatanchors consuming more than 1,000 watts or so from the electricity rationed to Retronics. In stark contrast with the latest embed-

ded technology projects, the articles published in *Elektor Magazine* on these humble instruments not only explain their assembly and practical use, but also delve into the measurement principles, and engineering issues spotted and vanquished elegantly in Elektor fashion. Add to that, the effort the lab team put into metalworking and product engineering. If you have a Verobox'ed or Telet-cased instrument built from an Elektor article I did not mention here, please let me know as I am sure the list presented here is not exhaustive. I am in the **EST[®] 2004 box**. ◀

(160197)



Figure 7. The very plain, almost Spartan looking Watt Meter from 1991.

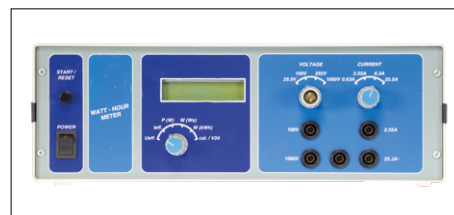


Figure 8. One of the largest instruments built into a Telet case was the Watt-Hour Meter from 1993.

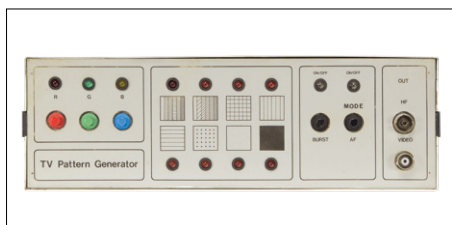


Figure 9. Still in frequent use at Retronics Labs: the TV Pattern Generator, albeit not to test TV sets or monitors any longer.

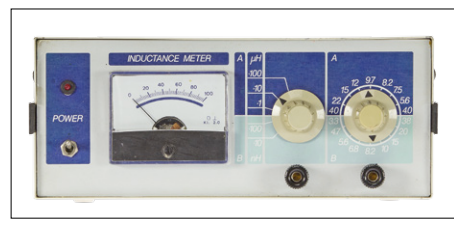


Figure 10. Inductance Meter. I fail to see why the tiniest of tiny center dot (interpunct) was used instead of a plain 'x' sign in front of the range markings.

Web Links

Note: original articles from 198x and 199x are available as pdf files on Elektor's 1980-1989 and 1990-1999 DVDs respectively.

- [1] H_{FE} Tester (900078), Elektor Electronics September 1990, www.elektormagazine.com/magazine/elektor-199009/32219
- [2] Watt Meter (910011) Elektor Electronics April 1991, www.elektormagazine.com/magazine/elektor-199104/32348
- [3] Watt-Hour Meter, Elektor Electronics February & March 1993, www.elektormagazine.com/magazine/elektor-199302/32797
- [4] Advanced LCR Meter, Elektor Electronics April 1997, www.elektormagazine.com/magazine/elektor-199704/33817

Compiled by **Robert van der Zwan**

Fast Forward Award 2016: wind and... water

The winner of the 2016 **e**lectronica Fast Forward Award sure knows how to exploit wind energy. But what about exploiting



water with great accuracy? This is the idea behind a novel water meter with a level of precision rarely seen today. The FFA Jury considered the idea of Australian FFA participant Lenn Williams to be so unique that it created the new 'Tech for Good' award right

there and then. Lenn (photo; with glasses) flew back to his home country with a 5,000 euro check to spend on PR and marketing, and for advocating to the entire water sector that it and its customers can get a fair(er) share.

Do not take our word for it

Do not take our word for it. Falk Senger, the Managing Director at Messe München, made it very clear he was more than happy with the way the **e**lectronica Fast Forward Award brought all those e-innovations to the fore. Concluding the FFA 2016 awards ceremony, he could not help noticing that both the quality and the diversity of the entries were outstanding. The Jury of the FFA, 'powered by Elektor', could not concur more.



electronica 2016 in numbers

Let's have a short flashback on the **e**lectronica 2016 trade show. A total of 2,913 companies from more than 50 countries presented their solutions to approximately 73,000 visitors from more than 80 countries. The main topics were security and automotive. 99 percent of the visitors rated the Munich show as 'good' or 'excellent'. Combined with a 7 percent increase in the numbers of exhibitors since the last **e**lectronica in 2014, Falk Senger, Managing Director at Messe München, thinks exhibitors and organizers did "an impressive job".



Director of **e**lectronica München Angela Marten and Publisher of Elektor Don Akkermans signed an agreement to continue the collaboration on the **e**lectronica Fast Forward Awards, the Start-up Platform powered by Elektor. The FFA will return to **e**lectronica in 2018 and be kept alive through 2017.

Low on Sunshine? Why Not Switch to the

The **e**lectronica Fast Forward Award winners can step forward now! Two of them are from Germany, one from the USA. It sure was an exciting Friday morning at the **e**lectronica 2016 trade show in Munich. The excitement was in the air until the very last moment – and naturally so!

The **e**lectronica Fast Forward Award ceremony kicked off at 10 am sharp at the Elektor Booth, the vast booth at the East entry that captured the imagination of grown-ups and youngsters alike. The Elektor booth was all about innovation, bringing together 35 inventions from 16 countries. The international Jury, coordinated by Elektor and also consisting of renowned companies like STMicroelectronics, Würth Elektronik, Conrad and Trinamic, had a tough time deciding which three entries should win in the set categories Prototype, Start-up and Idea.

After some deliberations on Friday morning, based on the 35 presentations during the first three days of **e**lectronica 2016, the Jury reached its conclusion around 10:20: Mowea from Berlin topped in the category Prototype, BotFactory from New York was first in the category Start-up and Kewazo from Garching (Germany) took pole position in the category Idea.

All three were asked to respond to salient questions from both members of the Jury and the public.

At 11:00, it was decision time. FFA 2016 Jury Manager Clemens Valens announced the winner of the third prize; Kewazo with its (yes...) scaffolding building robot. Artem Kuchukov became the happy owner of a check worth 25,000 euros to be spent on PR and marketing.

As Clemens rightly put it: "calling out number 2 is, by implication, also calling out number 1". Number 2 was BotFactory with its PCB prototype desktop printer (a complete pick & place machine included!). CEO J.F. Brandon was the lucky owner of a check worth 50,000 euro.



PEOPLE NEWS • Tanja Pohlen can take credit for organizing the **e**lectronica Fast Forward Award, as trade show, Tessel Renzenbrink and Jens Nickel kept online readers updated on everything they needed anchorman Jan Buiting and cameraman Patrick Wielders helped a few camera-shy people to speak up Sales, Client, and Member staffers Julia Grotenrath, Chantalle Reuling and Raoul Morreau spoke to

Weakest of Winds?



Number 1 was... Mowea with its micro wind turbines, allowing households or small companies to combine solar energy with wind energy, and so create a stable power supply despite being off-grid. The prize for CEO Till Naumann and his team: a 75,000 euro check and an exhibition stand at **e**lectronica 2018 included!

Photo: Dr. Till Naumann and Lara Obst of Mowea presenting their 'pico wind' turbine.

can Angela Marten from **e**lectronica 2016 • During the to know from Munich • While in Munich, Elektor TV and share bits and bytes of extra information • Elektor countless people at the Elektor Booth

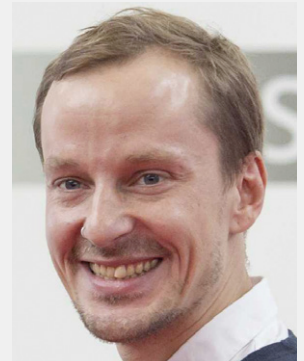
EXPERT PROFILE

Elektor works closely together with more than 1,000 experts and authors for the publication of books, articles, DVDs, webinars and live events. In each installment of Elektor World News we put one of them in the limelight.

Name: **Dr. Till Naumann**

Education:
PhD, Technical University Berlin

Professional interests: **I love wind and aerodynamics, maybe because my father was a stunt pilot**



Who is Till Naumann?

I am an engineer from Berlin. I wrote my PhD thesis at the Technical University Berlin about the aerodynamics of wind turbines. I'd like to see 'pico wind' made affordable, using my knowledge about big wind turbines. I met my partner Andreas [Amberger, Ed.] at the TU Berlin. We developed a modular wind energy system based on 10-kW nominal power.

Where you surprised to win the **electronica FFA?**

Our team was very surprised! We came from Berlin by train, slipped in one of the cheapest hotels and put all our energy in the pitch in front of the Jury – after almost three days of no sleep. We made it and we are very happy indeed!

Do you have a plan how to spend your 75 K euros or is it too early to tell?

Being a start-up, we need marketing in every direction you can imagine: corporate design, a better looking homepage, testimonial videos of our prototypes and of our field tests, content for B2B partners, presentation opportunities at relevant trade shows and especially community building, on Facebook and other media. We are planning a crowdfunding campaign, tentatively at the end of 2017.

What do you hope to accomplish within the next five years?

We aim to build a fast growing, but also solid business that is able to deliver clean, affordable off-grid energy solutions. We want to take part in the decarbonization of energy production.

What made you start developing your wind energy plan?

I love wind and aerodynamics. Because my father was a stunt pilot, I spent a lot of time in the air as a small kid. Anyhow, Andreas and I decided to take up our passion for 'small wind' when we noticed a strong interest in small, renewable energy solutions worldwide — but especially in countries that lack a proper energy infrastructure.

What will be the most key electronics development in the not too distant future?

Andreas is an expert in everything to do with electronics and software. We want to make our turbines 'smart' so that we can collect data about energy production and customer behavior. We also want to integrate our turbines into micro grids. In short there is an awful lot we want to do!

chatdoku a mind-teaser for elektornics enthusiasts

Contributed by **Claude Ghyselen** (France)

This New Year, our traditional Hexadoku grid is being replaced with a Catdoku grid (the outline is shaped like a cat's head).

The rules for filling in the grid and the characters allowed are the same as for the normal Hexadoku grid.

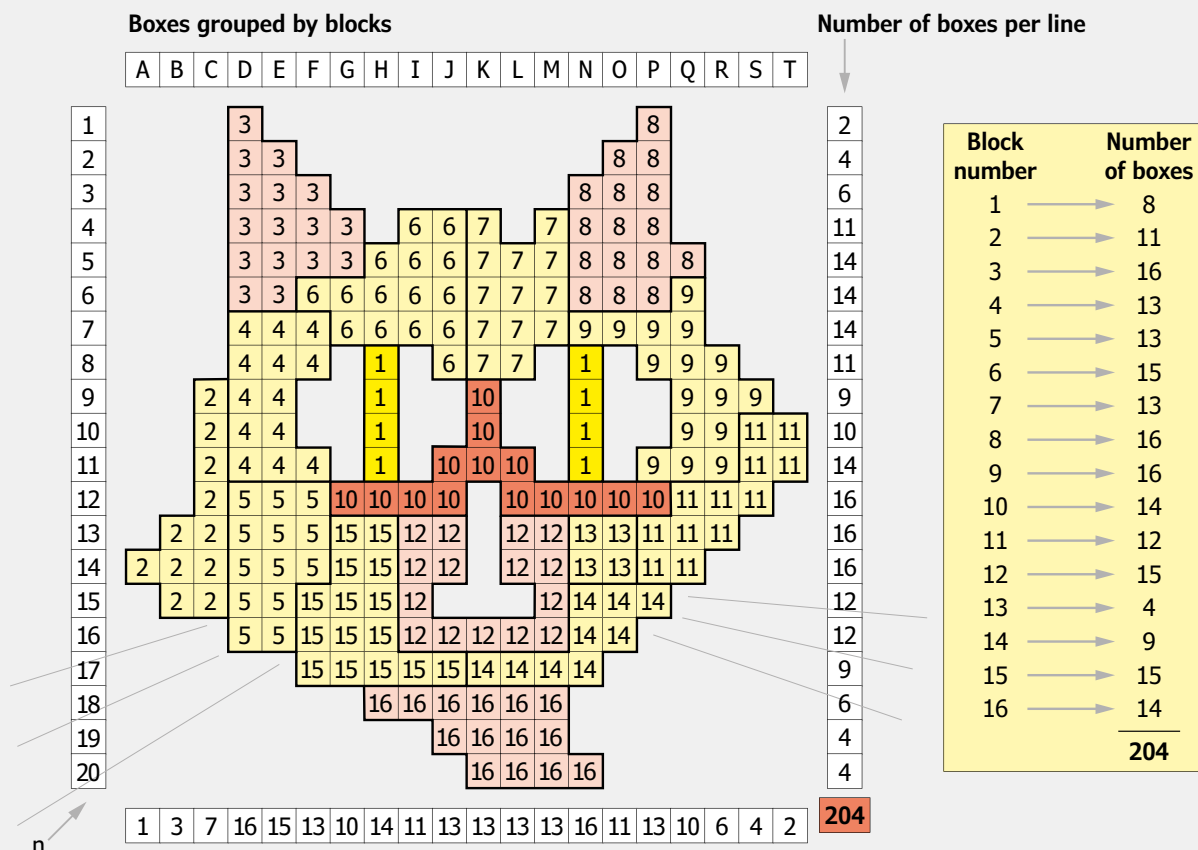
Reminder: a Hexadoku grid uses figures from the hexadecimal system, i.e. from 0 to F. You should fill it in so that all the hexadecimal figures from 0 through F (0-9 and A-F inclusive) are used **once and only once** in each row, column, and block (identified by a thicker line).

The Catdoku differs from the Hexadoku in the following ways:

- The grid has 16 blocks, but 20 lines (1-20) and 20 columns (A-T) (see grid below).
- Five areas are empty (white).
- The blocks are not square or rectangular, but are more like pieces of a jigsaw (referred to as a hexadoku jigsaw).
- Block #1 (pupils of the eyes, in bright yellow) is made up of two separate parts, one each for the left and right eyes.

- The number of boxes in each block depends on its shape and position, but is never more than 16. So this grid can still be described as a Hexadoku.
- The total number of active boxes is 204, compared to 256 for a Hexadoku.

As certain blocks contain fewer than 16 boxes, it's not possible to use all the hexadecimal characters in these ones. This is just a little extra difficulty to bear in mind when solving it. Certain figures have already been entered in the grid to define the starting situation. In order to enter, there's no need to send us the whole grid, all you need do is send us the sequence of six figures in the grey boxes.



A B C D E F G H I J K L M N O P Q R S T

1																			9			
2			8													0	F					
3			B												7		1					
4			1	6		3				0	4				A	5						
5			D	5	9				B							4			2			
6			0					E	3	7				F	B							
7								6	F	8	B	3				9						
8			5	E	B					0					D				C			
9			2	6	D										4				E	B		
10			6	C	2										1					4	B	
11			F			A								8		3		5	1		D	E
12									B	1						7	0		8	9	2	
13			0												1		E	C		A		
14			D	9																	5	
15			4	3																		
16			9	F					B	0						7				D		
17									F	9	1	3	6									
18																						
19										6						9	A	4				
20																						

Solve Catdoku and win!

Correct solutions received from the entire Elektor readership automatically enter a prize draw for three Elektor Book Vouchers worth **\$70.00 / £40.00 / €50.00 each**, which should encourage all Elektor readers to participate.

Participate!

Ultimately **February 1, 2017**, 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 installment 5/2016 (November & December) is: **6182F**.

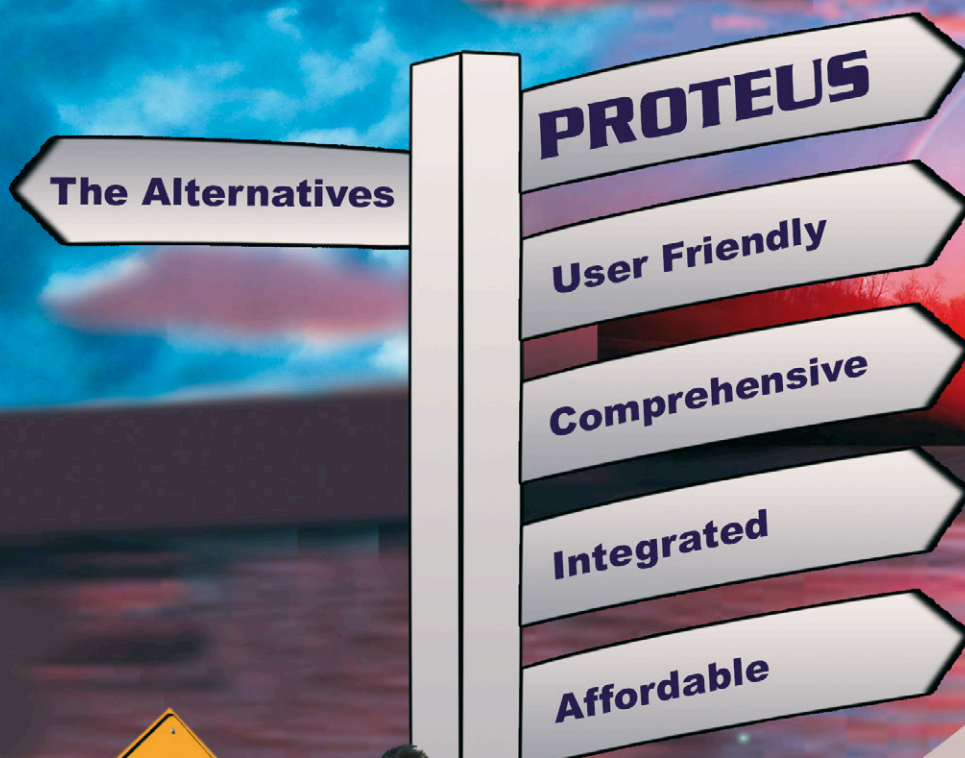
The €50 / £40 / \$70 book vouchers have been awarded to:
Eugene Stemple (USA); Michael Staffler (Germany);
Dirk Dreesen (Belgium).

Congratulations everyone!

The competition is not open to employees of Elektor International Media, its subsidiaries, licensees and/or associated publishing houses.

3	6	4	5	A	7	B	F	8	D	E	0	2	9	C	1							
D	E	1	7	C	0	2	8	F	9	3	5	A	B	4	6							
A	F	0	2	9	D	1	5	C	B	4	6	8	E	3	7							
9	8	B	C	3	4	6	E	A	1	7	2	5	D	F	0							
E	C	3	6	D	8	4	7	0	2	F	9	B	A	1	5							
4	1	2	8	B	5	0	3	D	6	A	E	C	F	7	9							
5	7	9	0	F	A	C	2	3	8	B	1	4	6	D	E							
B	A	F	D	E	1	9	6	4	C	5	7	3	0	8	2							
C	0	E	B	2	9	7	4	6	A	8	3	D	1	5	F							
1	2	A	F	5	B	3	0	7	4	9	D	6	8	E	C							
6	3	8	4	1	E	D	C	B	5	0	F	9	7	2	A							
7	5	D	9	6	F	8	A	1	E	2	C	0	3	B	4							
F	9	5	E	0	C	A	1	2	3	D	8	7	4	6	B							
0	B	6	1	8	2	F	D	9	7	C	4	E	5	A	3							
2	D	7	3	4	6	E	B	5	0	1	A	F	C	9	8							
8	4	C	A	7	3	5	9	E	F	6	B	1	2	0	D							

DO YOU WANT THE BEST ELECTRONICS DESIGN SOFTWARE



FEATURES

- Schematic Capture
- PCB Layout
- Gridless Autorouting
- 3D Visualization
- M-CAD Integration
- SPICE Simulation
- MCU Co-simulation
- Built in IDE
- Visual Programming

NOW INCLUDES:

Serpentine Routing, Layer Stackup Manager and Assembly Variants.

labcenter  www.labcenter.com

Electronics

Toll Free: +1 866 499 8184