£9.95 | €10.95 | $12.95

elektor MAG
design › share › sell

576

EDITION 6/2020
NOVEMBER/DECEMBER 2020
ELEKTORMAGAZINE.COM

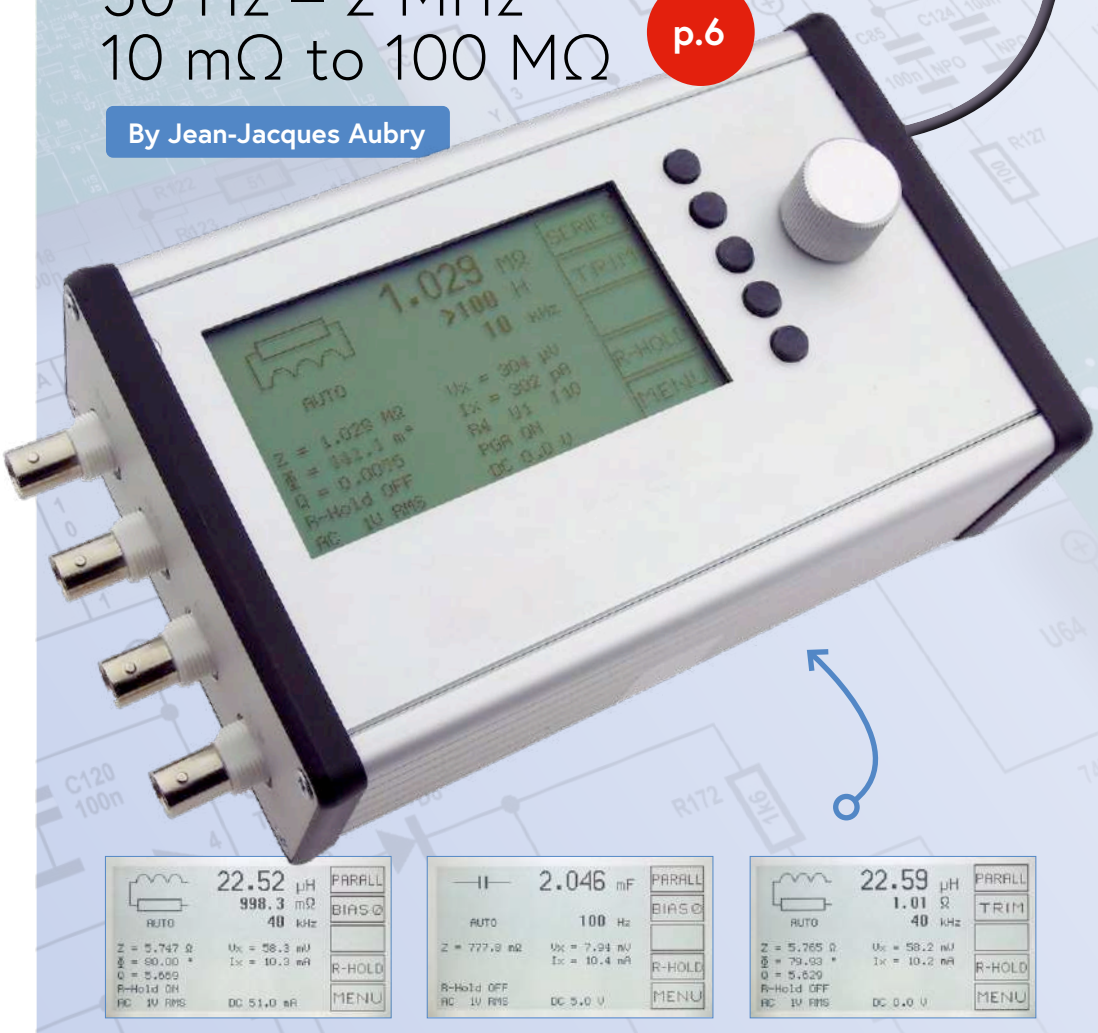
New LCR Meter

50 Hz – 2 MHz
10 mΩ to 100 MΩ

p.6

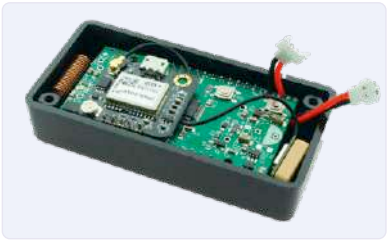
Back this project on Elektor LABS

By Jean-Jacques Aubry



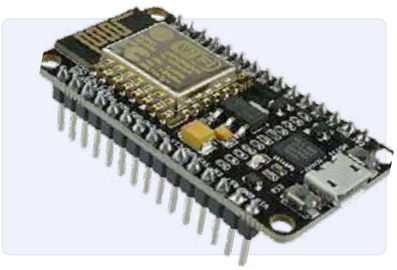
In this edition

and much more!



LoRa GPS Tracker
With Open Hardware and Software
p.98

Cyrob Review: Rigol DG2072 Generator
Electronics YouTuber Philippe Demerliac About Pros and Cons
p.62



More DIY Home Automation
Integrate Your Doorbell in Home Assistant with ESPHome
p.20

00504
9 770268 451173

Join the Elektor Community

Take out a elektor GOLD membership!

- The Elektor web archive from 1974!
- 6x Elektor magazine (Print)
- 9x Digital (PDF) including Elektor Industry (EN) magazine
- A 10% discount in our web shop and exclusive offers
- Elektor's annual DVD-ROM
- An online Elektor LABs account, with access to more than 1000 Gerber files and a direct line of communication with our experts!
- Bring a project to publication or even sell it in our shop

Also available

The Digital elektor GREEN membership!

- Access to Elektor's web archive
- 10% discount in our web shop
- 6x Elektor magazine (PDF)
- Exclusive offers
- Access to more than 1000 Gerber files

www.elektor.com/member

elektor
design > share > sell

COLOPHON

Elektor Magazine,
English edition

Edition 6/2020
Volume 46, No. 504
November & December 2020

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

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

Advertising & Sponsoring:
Margriet Debeij
Phone: +49 170 5505 396
E-mail: margriet.debeij@elektor.com

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

EDITORIAL

Jens Nickel

International Editor-in-Chief, Elektor Magazine

Build and Learn

The title project for this issue is an LCR meter. The same author had a similar project in Elektor in 2013 — the six-part series also became a great commercial success at the time. Of course, time does not stand still. Powerful measurement devices are now also available for entry-level budgets, as can be seen, for example, with many oscilloscopes in the Elektor store. And the proportion of those readers who want to recreate large Elektor projects 1:1 may have shifted over the years. We therefore wanted to let our readers decide for themselves whether we should bring a kit to the market. If you are interested in purchasing a kit, you can indicate your (non-binding) interest at www.elektor.com/lcr. Even before the actual project articles were published (two parts in this and the next issue), there were already more than 100 interested parties. The target of 150 "supporters" (who will receive a discount of €100 euros if the project is successful) could already be achieved with this edition.

For our editorial team, which is currently thinking about content for next year, the question arises again and again how we can best support the development of electronics. As I've said, in purely financial terms, it has become even less profitable in recent years to build devices and modules yourself. Another example is our second big project in this issue — a GPS tracker that can transmit its position via LoRa networks such as TheThingsNetwork. Such modules can now be bought for a handful of euros — DIY cannot keep up here. Nevertheless, I have been a strong supporter of the project. Because when you set up our project, you get a completely transparent system. We also offer the CAD files so that you can expand and "mod" our project as you wish. And we accompany the project with articles that will teach you a lot about setting up your own LoRa project. Of course, those of you who have neither the time nor inclination to heat up the soldering iron or take the breadboard out of the drawer can also take something with them. But if you take the time to do it, you will find out what terms such as "Network Session Key" and "Device Address" mean.

It is precisely this mixture of theory and practice that sets Elektor apart, and we will remain true to it in 2021!

The Team

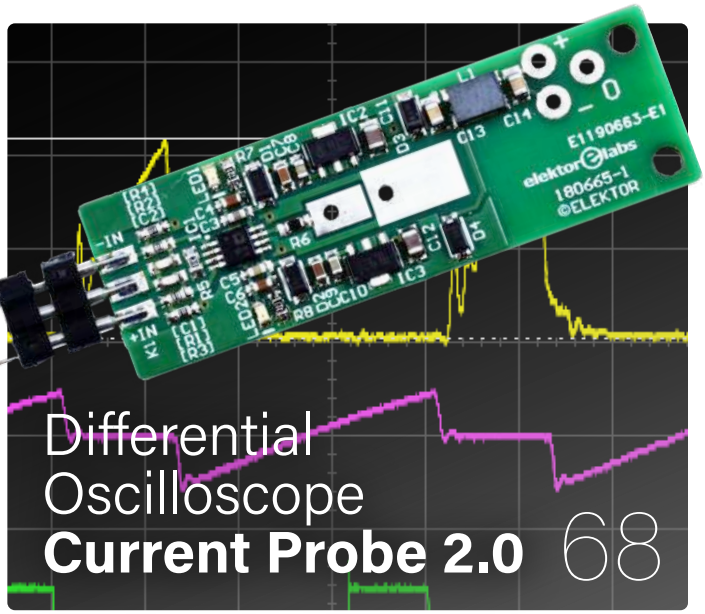
THIS EDITION

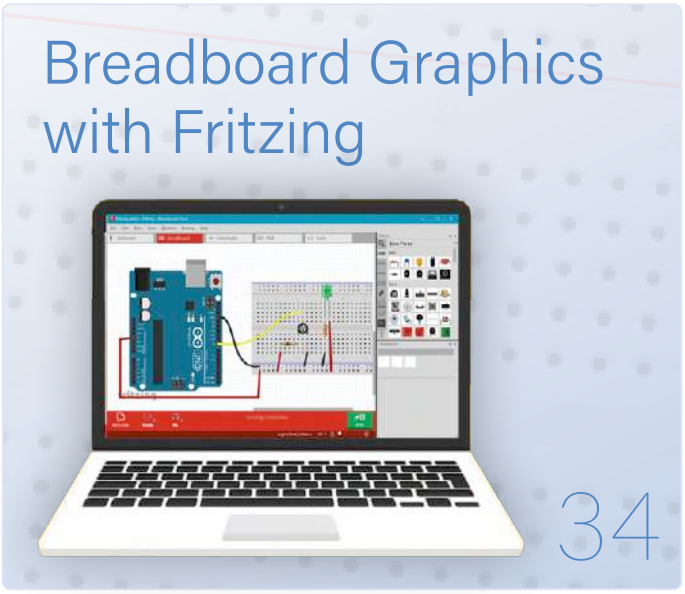
Volume 48 – Edition 6/2020
No. 504 – November/December 2020

Regulars

Features

Projects





New LCR Meter
50 Hz – 2 MHz
6

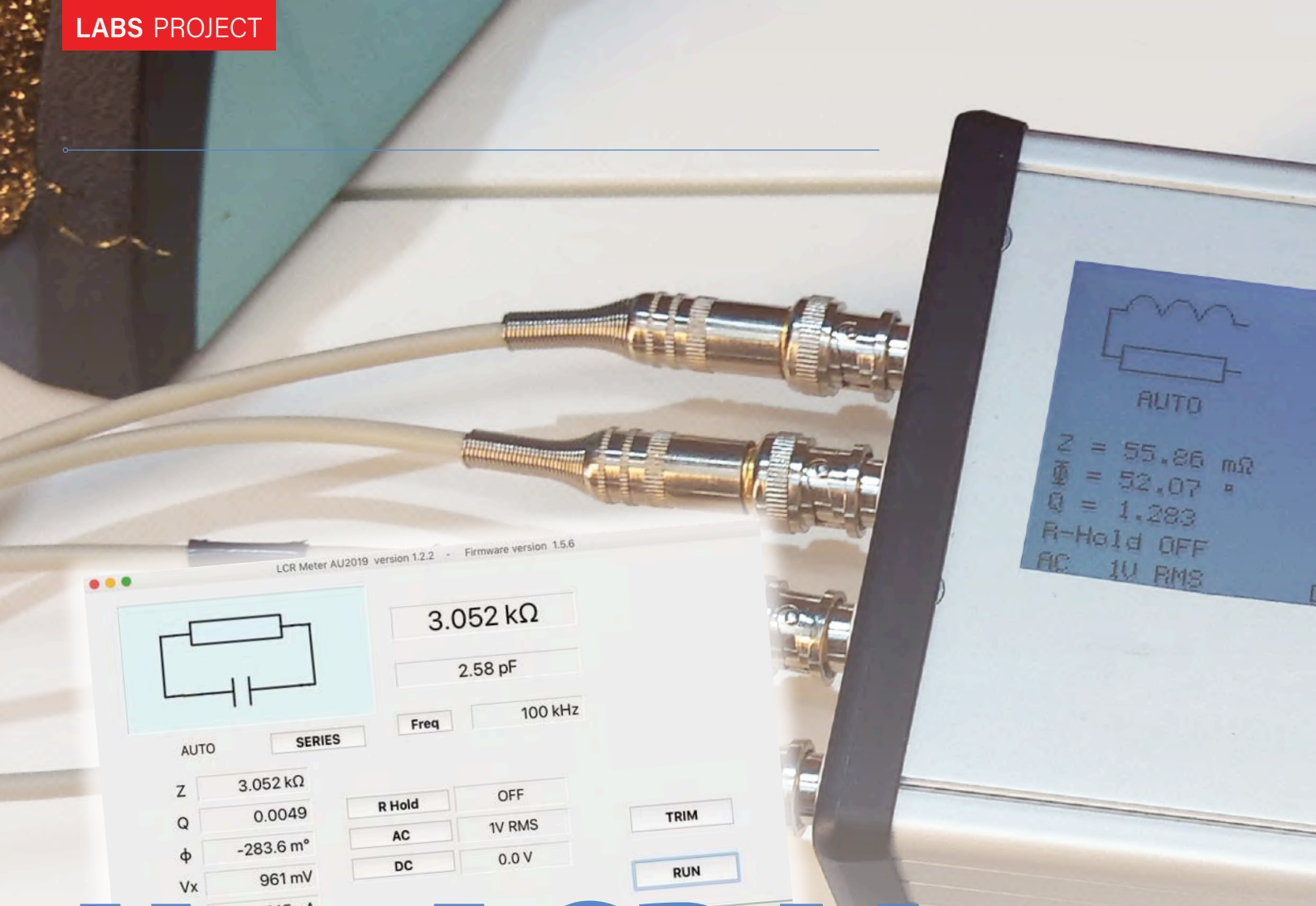
Next Edition

Elektor Magazine Edition 1/2021 (January/February)
As usual, we'll have an exciting mix of projects, circuits, fundamentals and tips and tricks for electronic engineers.

From the contents:
- 32-Bit I2S Sine Wave Generator
- DIY Thermo-Cam
- Power Analyzer
- Bluetooth Beacons in Practice
- Homelab Project: Particulate Matter Sensor
- Home Automation with Home Assistant (3)
- FFT on the Maixduino
- Object-Oriented Programming for Beginners

And much more!

Elektor Magazine edition 1/2021 covering January/February is published around the 02 January 2021. Arrival of printed copies with Elektor Gold Members is subject to transport. Contents and article titles subject to change.



New LCR Meter 50 Hz - 2 MHz

An automatic impedance measuring bridge measures the resistance, capacitance and inductance of components with an impedance of 10 mΩ to 100 MΩ

By **Jean-Jacques Aubry** (France)

More than seven years ago, Elektor published my 500 ppm LCR meter. Since then I have developed a fully new LCR meter, sacrificing on extreme accuracy for the sake of extended functionality with: test frequency from 50 Hz to 2 MHz, four possible test voltages (100 mV, 200 mV, 500 mV and 1Vrms), and additional DC polarization up to 5 V for capacitors and 50 mA for inductors.

PROJECT DECODER

Tags
Measuring instrument, Kit

Level
entry level – intermediate level – expert level

Time
1 hour

Tools
Standard homelab tools
Adjustment tool for trimmers

Cost
799€ (kickstarter: 699€)
www.elektor.com/lcr

This is by no means a remake of my project published by Elektor in 2013. Neither an update nor an evolution, it is a fully redesigned, completely different instrument. In this new project, particular attention is paid to its ease of implementation (calibration) and use. A rotary encoder is used to navigate through the menu and to change the frequency.

General information

This LCR meter is an automatic impedance measuring bridge. It measures the resistance, capacitance and inductance of components with an impedance of 10 mΩ to 100 MΩ, and performs measurements at a frequency in the range 50 Hz to 2 MHz.

Two configurations are possible:

- Base unit (main board + display extension). Operates in standalone mode (without PC) with an external 5V power supply via the USB connector (which can also be connected to a computer for power supply). This is the recommended configuration.
- Main board. Works only when connected via USB to a computer running the user program. This program, developed from Qt libraries [1], has been tested under Windows 7 and MacOSX.

A Bluetooth extension for connection with a smartphone used as GUI is in progress.

Measurement principle

The impedance (Z) is an important parameter to characterize passive electronic components (resistors, capacitors, inductors). It is a complex number which can be represented by a real part (R) and an imaginary part (X) such that $Z = R + jX$, or in polar form by the modulus of its impedance and the phase shift between voltage and current:

$$|Z| \angle \theta.$$

To determine impedance it is therefore necessary to measure at least two values (in magnitude and in phase), generally the voltage at the terminals of the component and the current flowing through it.

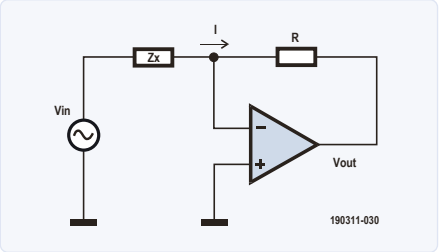


Figure 1: The self-balanced bridge method uses a simple operational amplifier for the current/voltage converter.

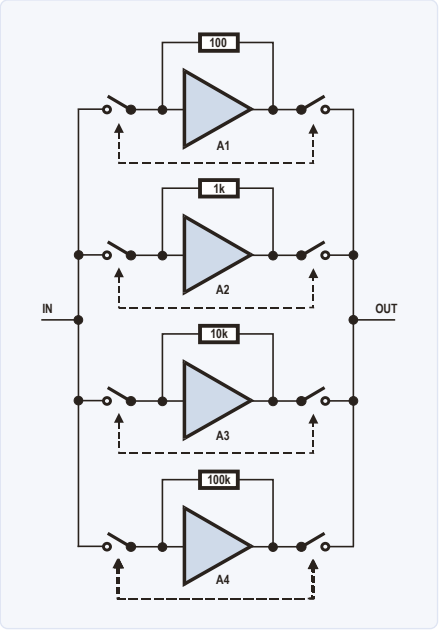


Figure 2: Four I-V stages, one for each measuring range.

The LCR meter AU2019 uses, like its predecessor, the self-balanced bridge method with the use of a simple operational amplifier for the current/voltage converter (I-V converter, see **Figure 1**).
This simple method provides good measurement accuracy at a reasonable cost. Its main disadvantage is a frequency range limited in the high frequencies by the performance of the operational amplifier used.
To obtain a wide impedance measurement range (from a few tens of mΩ to more than 100 MΩ) it is necessary to switch the precision resistor (R) used in the I-V converter. Unfortunately, common analog integrated switches (such as 74HC4052) introduce parasitic elements (mainly capacitance) which will limit the performance at high frequencies. This is why most similar instruments have their frequency range limited to 100 (or 200) kHz. It is however possible, by an original design and the choice of high-performance components, to push the high frequency to 2 MHz without excessively higher costs, while maintaining the simplicity of the design.
The chosen solution, shown in **Figure 2,** is not to switch the four measuring resistors (and thus reducing parasitic capacitances to a minimum) but to have four Amplifier + Resistor pairs, each pair selected to match with the impedance to be measured.
The operational amplifiers used (AD8099 from Analog Devices) have a cut-off frequency of approximately 200 MHz at an output voltage of 2 V_{p-p} and have a control input to inhibit (mute) the output, exactly what we need for this application. The switches are PhotoMOS solid-state relays manufactured by Panasonic with a very low product (ON resistance x output capacitance).
Another important point is the choice of the method of generating the test frequency. Nowadays it is easy and inexpensive to use Direct Digital Synthesis (DDS) components for this task, with the advantage that any frequency in the 50 Hz / 2 MHz range can be generated. It is, moreover, easy to generate a signal of the same frequency for the synchronous detector, but with variable relative phase, thanks to a second DDS circuit that is synchronized to the first.

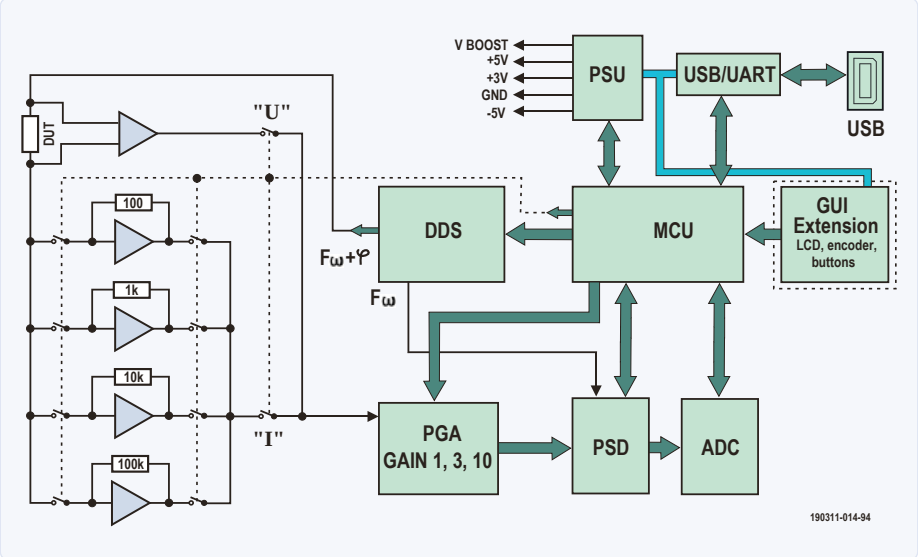


Figure 3: The various parts of the LCR meter and how they are interconnected.

Block diagram

The block diagram in **Figure 3** illustrates in which sections the hardware of the LCR meter is organized and how these sections are interrelated. Every section will be discussed later in this article.
The user interface is designed on a separate PCB, with control elements reduced to a strict minimum:

- 240 x 128 dots LCD graphic display.
- Five push buttons.
- A rotary encoder with auxiliary contact.

Although I don't recommend to do so, you may omit this extension board and control the LCR meter by connecting it to a computer via the USB interface, using a suitable PC program. Currently my PC program does not offer the same features as my standalone version. I still think a stand-alone measuring device is always the best choice. Physical buttons are easier and faster to operate! No update is required every time the PC operating system is being upgraded.
The LCR meter is powered via a Mini-USB connector. You can use a smartphone charger, an external 5V power bank (for smartphone) or connect it to a computer (this last connection will also allow a firmware update).

Main Board circuit description

Analog input section (Figure 4)

The measurement is done with a 5-Terminal configuration [2] to minimize the influence of the test cables. J4 (**H**igh **D**rive) and J7 (**L**ow **D**rive) connectors power the DUT (Device Under Test) while the J5 (**H**igh **S**ense) and J6 (**L**ow **S**ense) connectors allow the voltage to be measured as close as possible to the device. If a bias voltage is applied, the positive terminals are at J4/J5.
The PhotoMOS solid state relays U54 to U57, which connect the operational amplifiers U9 to U12, having an ON resistance of about 1Ω, the voltage must be measured at the DUT terminals and not relative to the ground. This

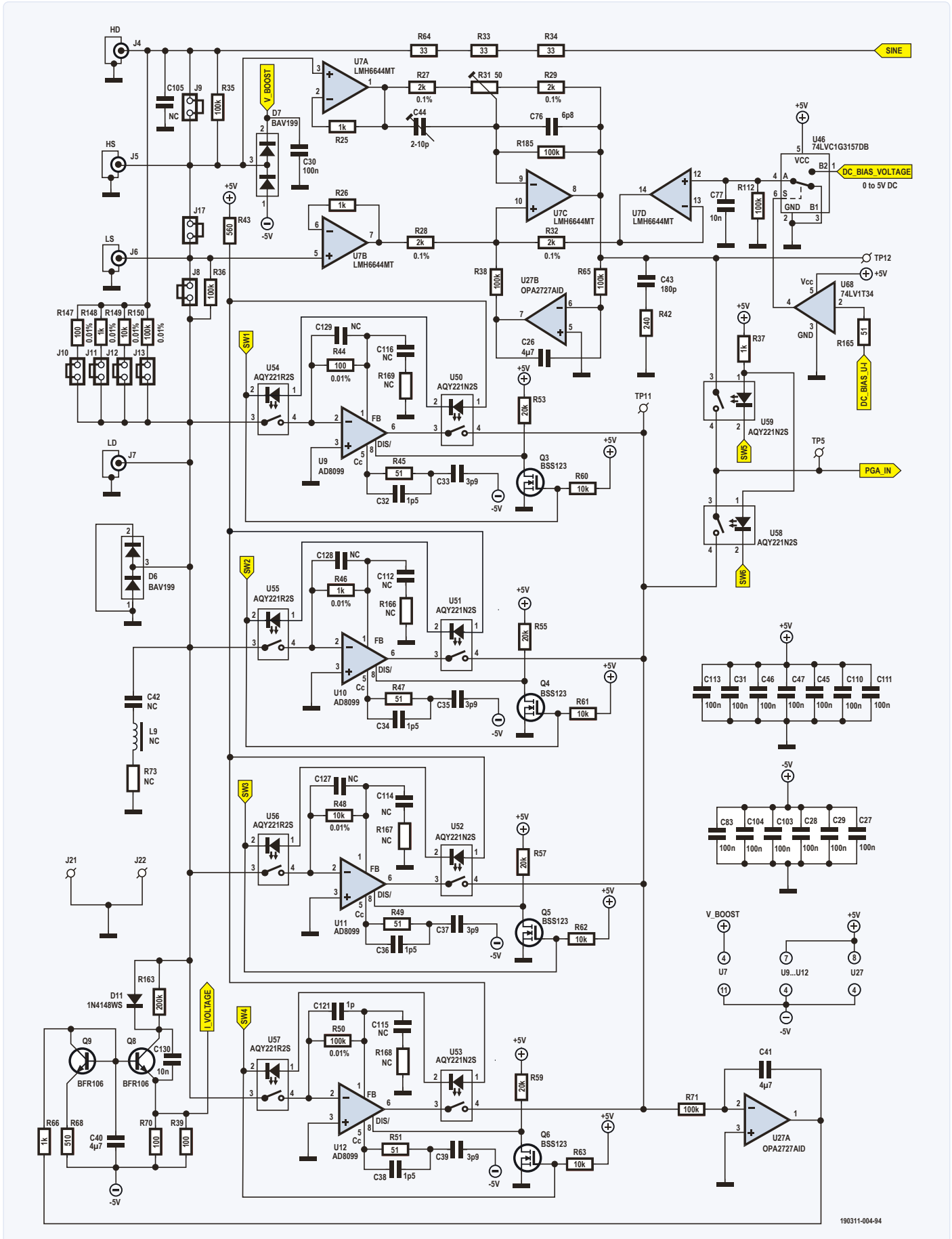


Figure 4: The analog input stages. Four impedance ranges can be selected by the microcontroller.

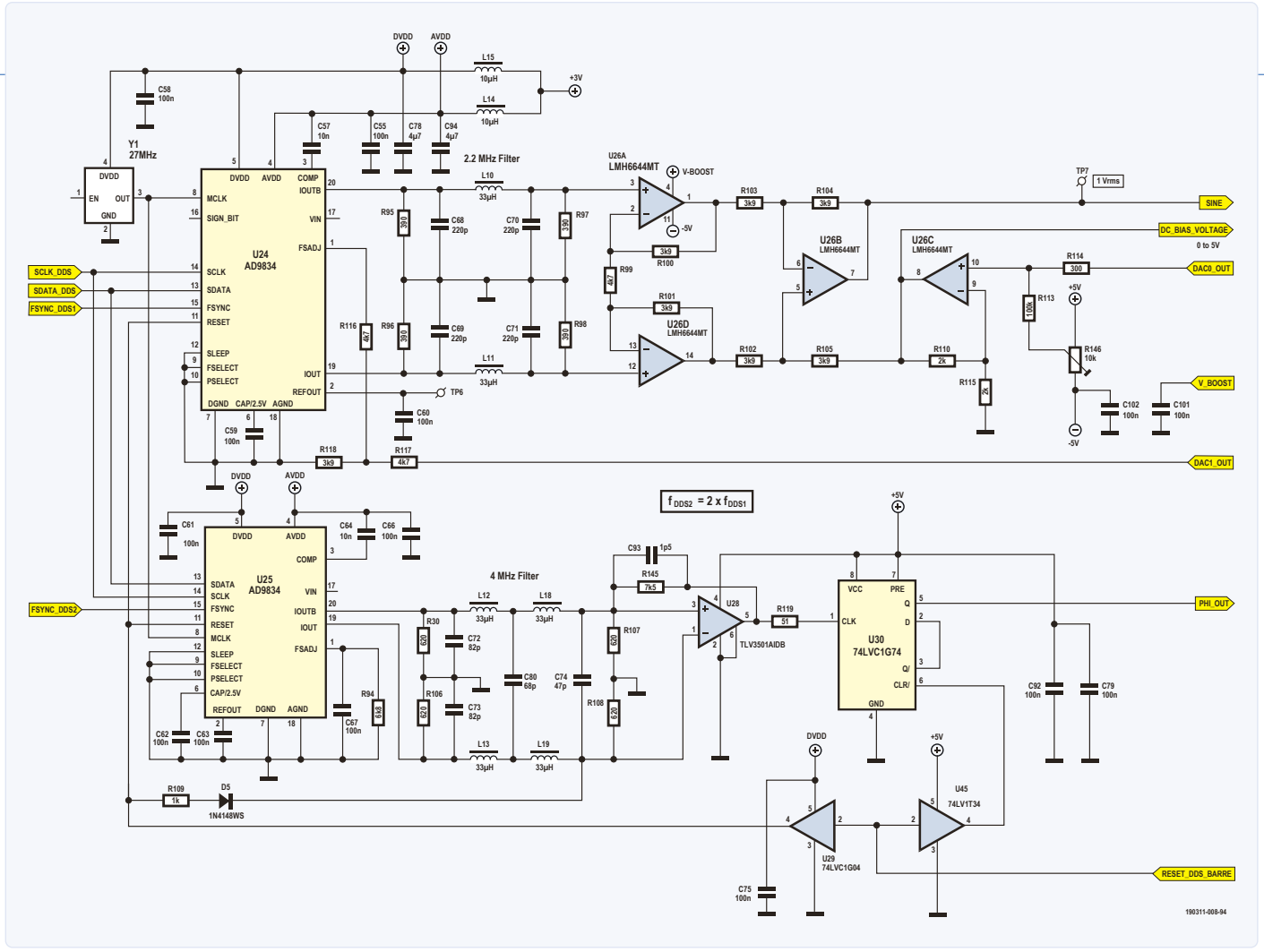


Figure 5: Two DDS-ICs generate the test signals.

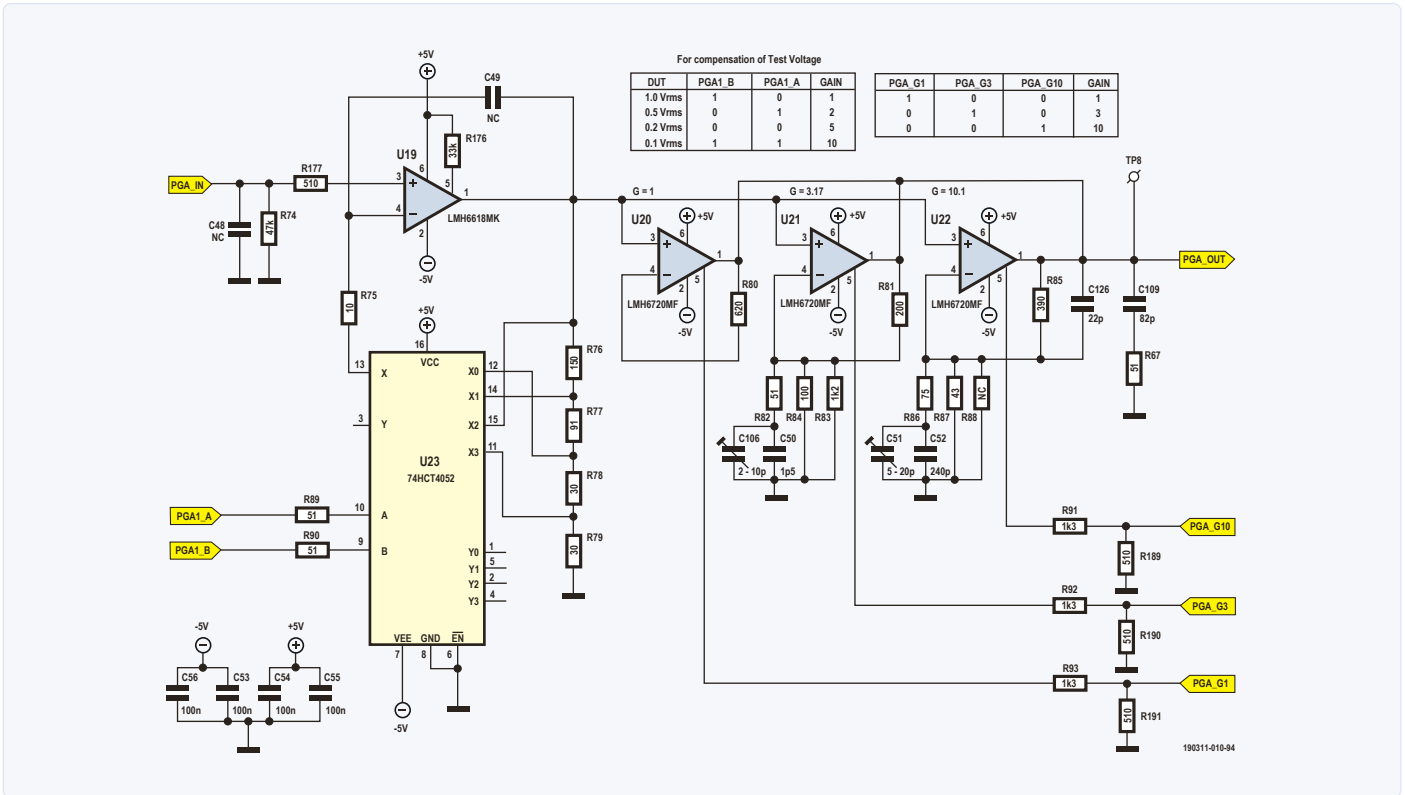


Figure 6: Programmable Gain Amplifier with frequency compensation.

is the task of the differential amplifier built around U7A, U7B, U7C. The common mode rejection ratio (CMRR) must be high throughout the frequency range of the LCR meter. An adjustment is therefore necessary, first by R31 for low frequencies (10 kHz) and C44 for high frequencies (1 MHz).
The integrator built around U27B allows automatic offset adjustment.
Depending on the measurement range, one of the operational amplifiers U9 to U12 is selected. Its DIS/ pin must be at logic high level and the associated PhotoMOS relays (input by U54 to U57 and output by U50 to U53) must be switched ON by one of the signals SW1 to SW4 from the MCU.
Signals SW5 (U59 control) and SW6 (U58 control) are used to switch either the U7C output voltage (DUT voltage) or the I-V converter output voltage (DUT current image) to the measurement circuit.
The application of DC bias on the DUT is made at the sine generator by a shift of its offset from 0 to 5 V. Since a 100 Ω resistor is put into series with the output of this generator, it will also correspond to a current of 0 to 50 mA if the DUT (inductive) has a low series resistance. For a voltage on a capacitor, U7D allows pre-compensation at the voltage differential amplifier. For a current through an inductance, it is the direct current source formed around Q8 and Q9 that absorbs it. The integrator built with U27A ensures a DC voltage output of the I-V converter close to 0 V.
To make calibrations, precision resistors identical to those used for the I-V converters can be connected with a jumper (J10 to J13).

Sine wave generator (Figure 5)

The generator is built around integrated circuit U24, an AD9834 DDS circuit from Analog Devices. Its 27 MHz clock is supplied by oscillator Y1.
The output (two signals in opposite phase) is filtered and amplified by U26A, U26B and U26D. Amplifier U26C allows a voltage shift to be applied. Without the latter, the trimmer R146 adjusts the offset of the sine signal (measured on TP7).
The phase detector used requires a perfectly square signal, at exactly the same frequency as the generator, but the relative phase can be varied. This is achieved by the U25 circuit (also a DDS AD9834) operating at twice the frequency and followed by a fast comparator (U28) which drives a divider by 2 (U30 flipflop).

PGA (Programmable Gain Amplifier) (Figure 6)

A first amplifier stage, consisting of U19 and U23, with a selectable gain of 1, 2, 5 or 10, compensates for the reduction of the useful signal when the amplitude of the test signal decreases. Since its gain is not changed during a measurement cycle, its frequency response is not very important.
The next stage is either U20 with a gain of 1, U21 with a gain of 3 (3.17 more exactly), or U22 with a gain of 10 (10.1). U21 is compensated in frequency by the setting of C108, and U22 by C51. The calibration procedure will take into account the actual response of this PGA, for each of the frequencies used.

PSD (Phase-Sensitive Detector) (Figure 7)

This circuit is built around the two analog switches U41 and U42 which allow to connect a tank capacitor to the input signal for half a period, and to an integration capacitor during the other half period. The two switches operate in phase opposition in order to obtain a double-value differential voltage, usable for the analog-to-digital converter U43.
The relative phase of the switching signal relative to the generator sine wave makes it possible to make measurements of the phase or quadrature components of the input signal of this PSD.
The time constant for the tank capacitor is adjusted according to the measurement frequency: eight values are possible by selecting, thanks to U70, one of the resistors R121, R122, etc. A DC offset of 2 V is applied to the input

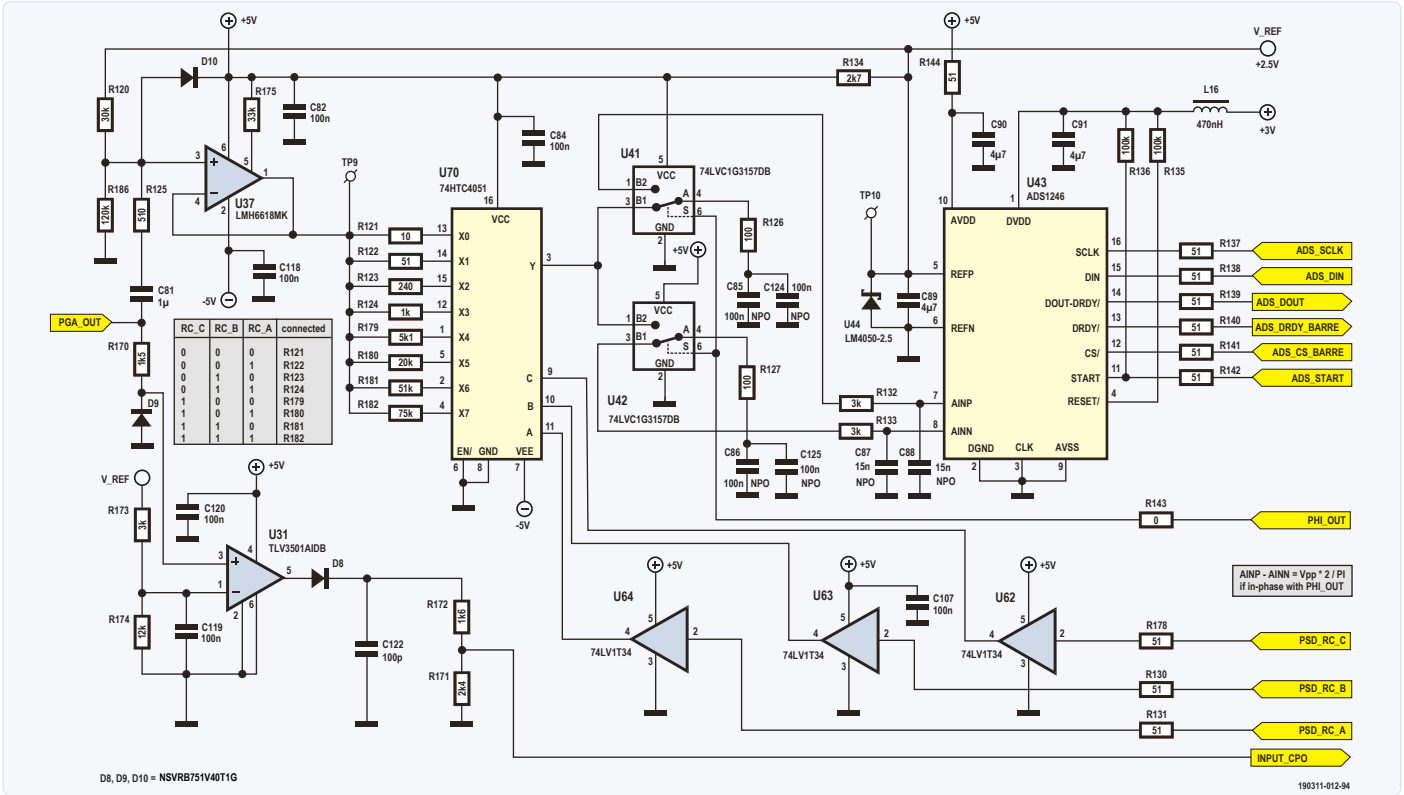


Figure 7: The Phase-Senstive Detector stage.

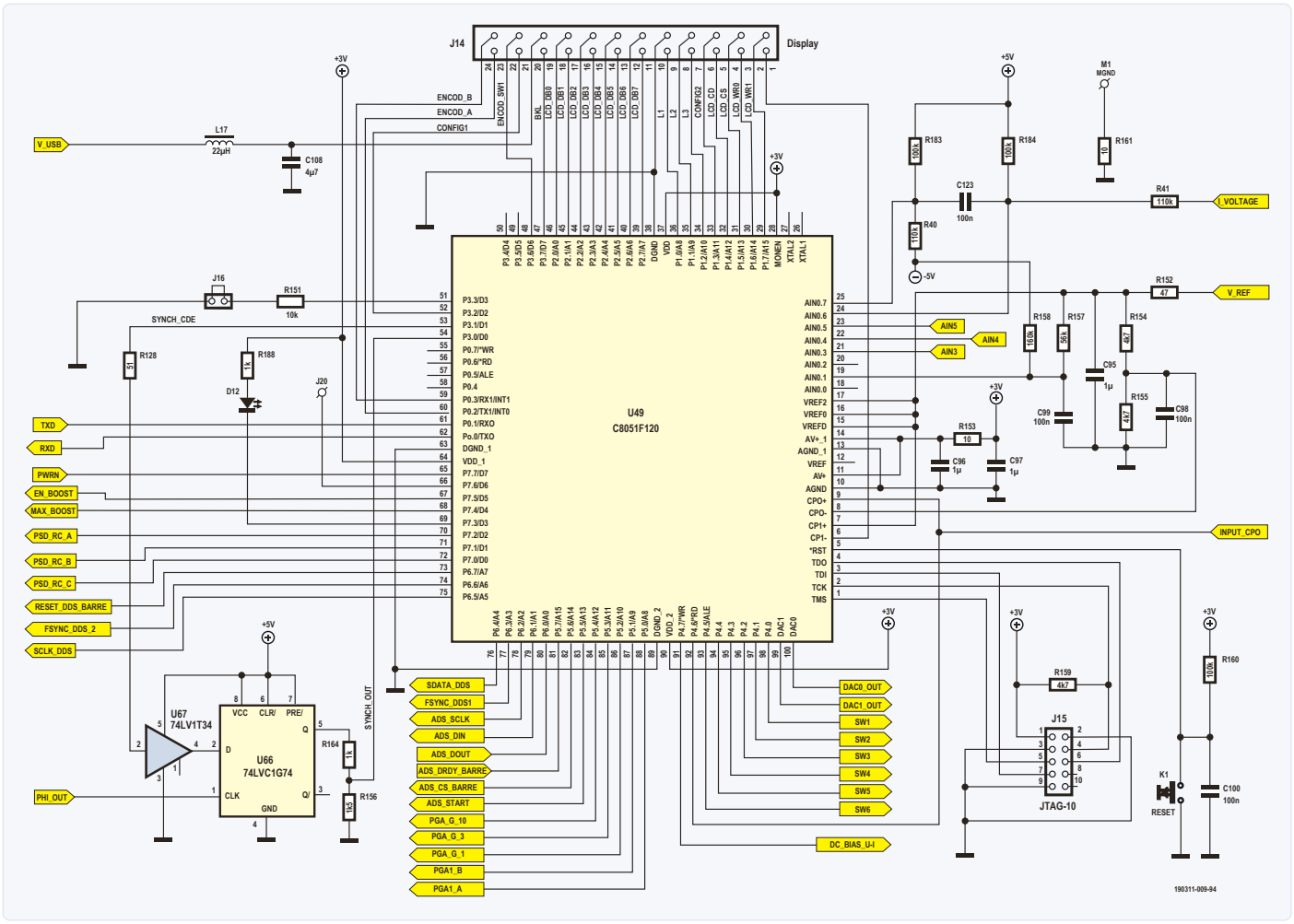


Figure 8: A Silicon Labs MCU is the heart of this LCR meter.

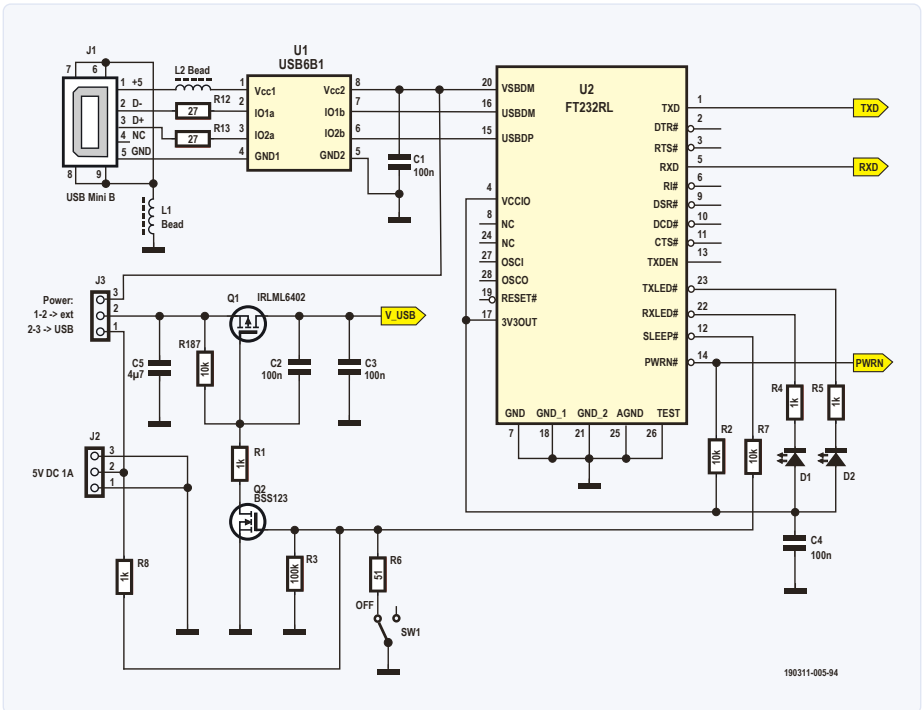


Figure 9: The USB/UART is used for communication and software updates.

of U37 so that the signal remains at maximum within the constant resistance range of the analog switches (linearity problem of this PSD). When searching for optimal measurement conditions (choice of I-V converter and PGA gain) the U31 Fast Comparator detects any signal overflow above 2 V peak, and send this information to the MCU (start of an interrupt routine).

MCU (Microcontroller Unit) (Figure 8)

The brain of the device is a C8051F120 MCU from Silicon Labs. Why is this relatively old type of circuit used, when there are more modern MCUs available, for example with ARM architecture?

- It is sufficient for the built-in peripheral circuits and its programmable Flash memory (64 K + 64 K).
- The best microcontroller is the one we know well!
- I already have the development tools!

Its internal oscillator + PLL generate a clock frequency of 73.5 MHz. The J15 connector

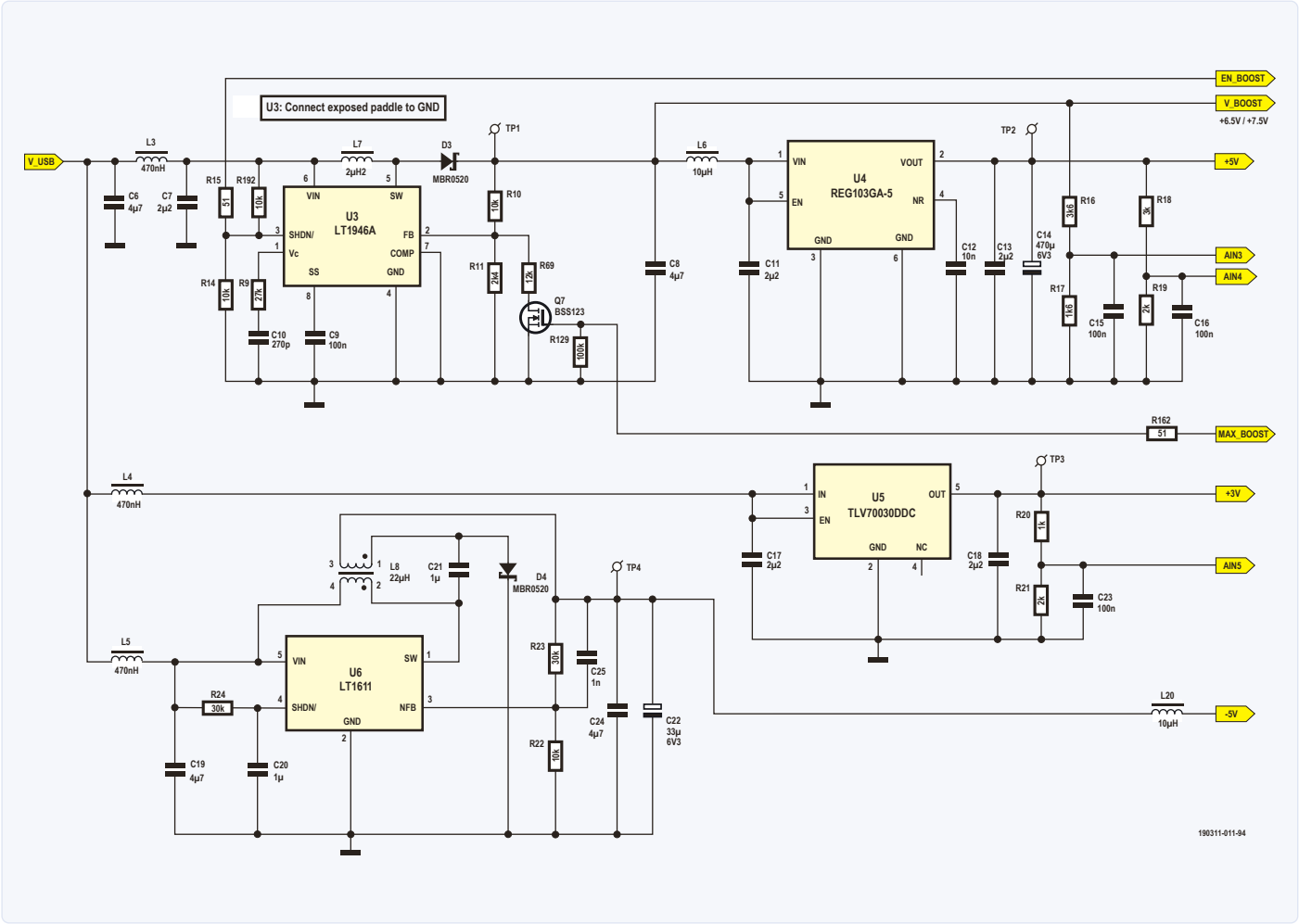


Figure 10: Four voltage regulators are used on the main PCB.

(JTAG) is used to connect a USB Debug Adapter from Silicon Labs (e.g. RS part no. 757-0297) to flash the Bootloader into the EEPROM. The J14 connector connects the Display extension board. LED D12 is used for debugging during the development of the firmware (for example when pushing a button or when the encoder is operated) and during normal operation to signal certain error conditions:

- No connected interface on J14 and no communication via USB: permanent flashing ½ s ON and ½ s OFF.
- Power test error: In addition to the message on the error number, permanent flashing ½ s ON and 1 s OFF.

Circuits U66 and U67 allow synchronization with the sine wave signal for the DUT voltage and current measurements.

A jumper on the J16 connector informs the Bootloader of an unconditional firmware update request.

The K1 push button (RESET) is optional (it is only used during the development phase).

USB controller and supply voltages (Figure 9)

An FT232RL (U2) from FTDI serves as a USB/UART interface with the MCU. Normally, the power supply is made via USB connector J1, but it is also possible to power the circuit via connector J2 and by putting a jumper between 1 and 2 of J3 instead of 2 and 3.

Four voltages are required to power all circuits (**Figure 10**):

- A V_BOOST voltage of +6.5 V or +7.5 V (depending on the control level MAX_BOOST) provided by the step-up regulator U3 combined with L7 and D3.
- A voltage of +5V provided by the linear regulator U4.
- A voltage of +3V provided by the linear regulator U5.
- A voltage of -5V provided by inverter regulator U6 combined with L8 and D4.

All these voltages are verified by the program when powering up using the 12-bit ADC and multiplexer built in the MCU.

Display board (Figure 11)

A flat cable on J1 connects this board to the main board. Five push buttons K1 to K5 and the rotary encoder switch SW1 are switched in matrix to use only three port lines on the MCU for decoding. The rotary encoder uses two more port lines.

RC circuits allow for a first anti-bounce filtering, simplifying software filtering.

Diodes D12 and D13, by triggering an interrupt routine at the MCU, inform the internal program that a button is pressed.

The graphic display U1 is powered by +3 V (U2 linear regulator) from the filtered USB supply voltage. The backlight is controlled by transistor Q1, and transistor Q2 allows, when the device is shut-down, a rapid discharge of the internally generated V_{LCD} voltage, avoiding unpleasant visual effects.

This Display board is identified, during boot,

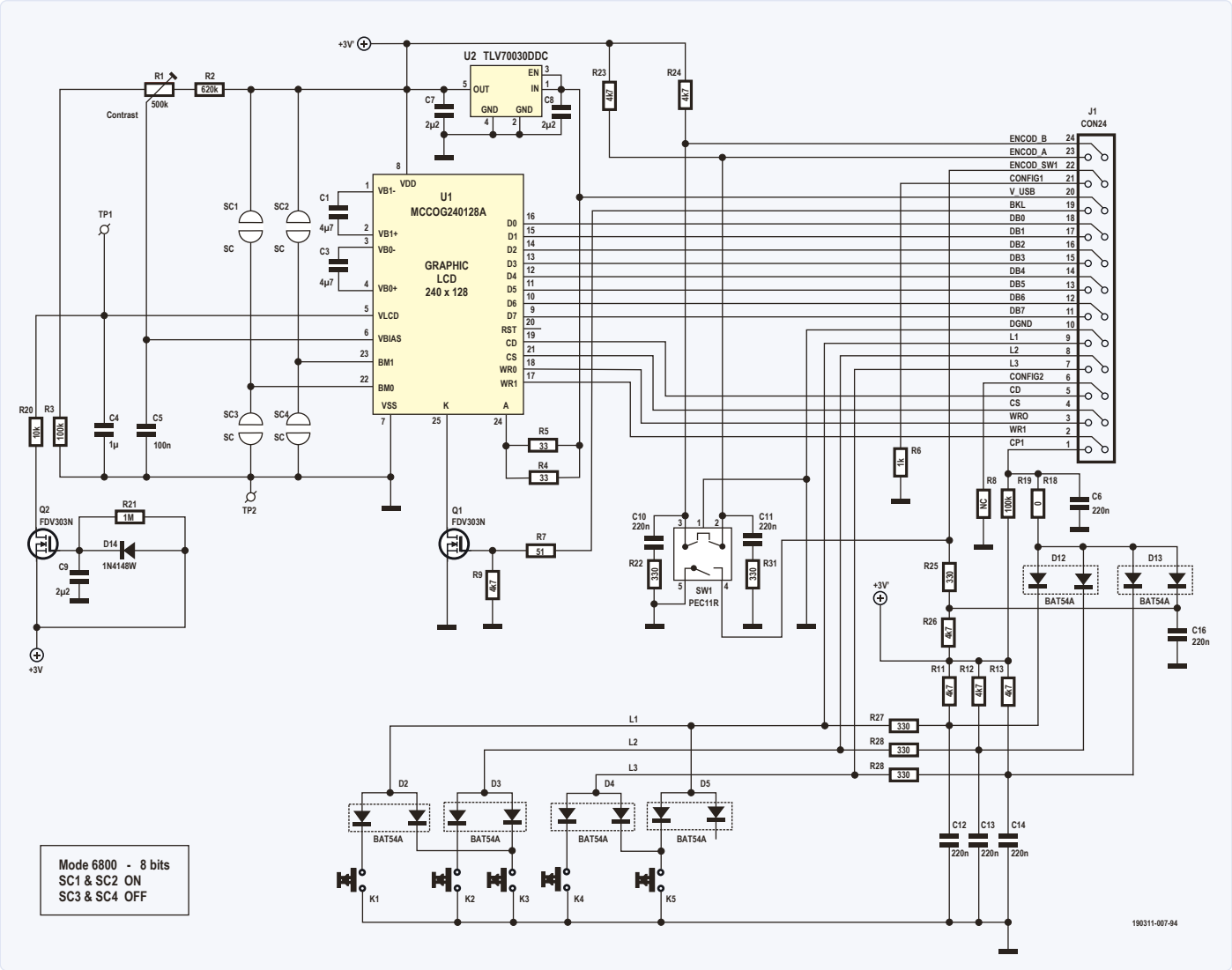


Figure 11: The board behind the front panel with buttons, LCD and rotary encoder.

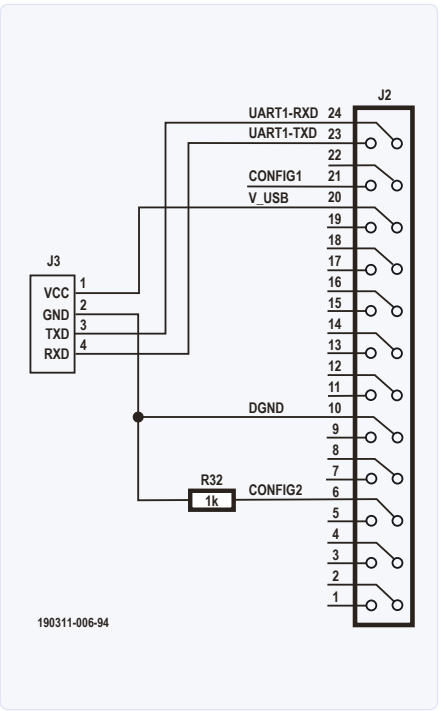


Figure 12: A Bluetooth interface can be used to control the LCR meter.

by the presence of a resistance to ground (R7 in series with R9) on pin 19 (BKL) of J1.

Bluetooth Low Energy interface (Figure 12)

An extension board with a Bluetooth Low Energy (BLE) module is under development. It is identified by pin 6 (CONFIG2) of J1 with a 1 kΩ resistor to ground.

Calibration in software (Figure 13)

The self-balancing bridge technique using a simple operational amplifier (A2) has a significant disadvantage at high frequencies: the virtual ground (see **Figure 13**, V3 voltage) is not maintained at zero because of the high frequency limitations of this amplifier.
Most simple computations of compensation of the parasitic elements (OPEN/SHORT compensation) which assume that the voltage V3 is a virtual ground are therefore no longer applicable to high frequencies and more sophisticated methods of compensation, such as the one called OPEN/SHORT/LOAD compensation must be used.
For those interested in the theory on this subject: read the article "Test Fixture Compensation Techniques in Impedance Analysis" in [4].
The calibration of the device will be done in several steps:

- For each predefined frequency and for PGA (A3) gains of 3 and 10.
- LOAD, for each of the 4 ranges with Zx = R_{ref}, for each predefined frequency.

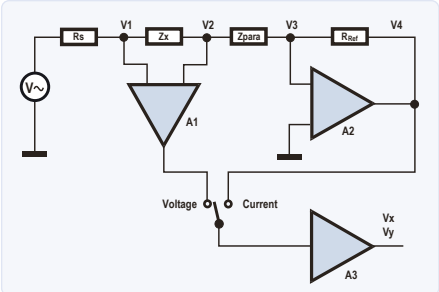


Figure 13: Simplified diagram of the measurement circuitry.

- OPEN TRIM (without connected component), for each predefined frequency.
- SHORT TRIM (input short), for each predefined frequency.

The stored values will be used to calculate the corrections to be made to the raw measurement values.
After adjusting the measured DUT values to account for the gain (complex value) of A3, both for the "CURRENT" and the "VOLTAGE" measurement, the final value will be calculated according to the formula:

$$Zx = Rref \frac{(Zxm - Zshort)(Zopen - Zstdm)}{(Zstdm - Zshort)(Zopen - Zxm)}$$

All these values are complex:
- Zxm is the measured value
- Zstdm is the measured value of R_{ref} during the LOAD phase
- Zshort is the value measured during the SHORT TRIM phase
- Zopen is the value measured during the OPEN TRIM phase

During use, it is possible to repeat the two TRIM operations ([TRIM] button), at the current frequency, if for example the probe has changed. When selecting a user frequency, the corrections will be made by interpolation from the two nearby predefined frequencies.

PCBs (Figures 14a, b & c)

The PCB layouts are available for download on the Elektor website [3]. Given the complexity of these boards – the main PCB has four copper layers! – they are not suitable for self-etching. Self-assembly is definitely only recommended for seasoned and (especially!) very experienced soldering professionals, people with the right skills and tools. With patience and a very steady hand, a magnifying glass and/or (stereo) microscope the job can be done with a very fine soldering iron, but this is not ideal for the parts with an 'exposed pad', a soldering joint that is not directly accessible with the soldering iron. We highly recommend to use solder paste with a suitable dispenser or stencils and a hot-air soldering tool or reflow-oven instead. If you have any doubts that you can finish this job, DON'T EVEN START.
The Gerbers files for the PCBs (for those who want to order the PCBs themselves) are available for download too, as well as the drill and mill drawings for the Hammond enclosure we used for this LCR meter [3].
At this very moment there is a 'project pledge' running on the Elektor Labs page https://www.elektormagazine.com/labs/remake-lcr-meter with already 59% of targeted 150 pledgers, interested to buy pre-assembled PCBs for this project (**Figure 14d**). Don't miss your chance to join in!

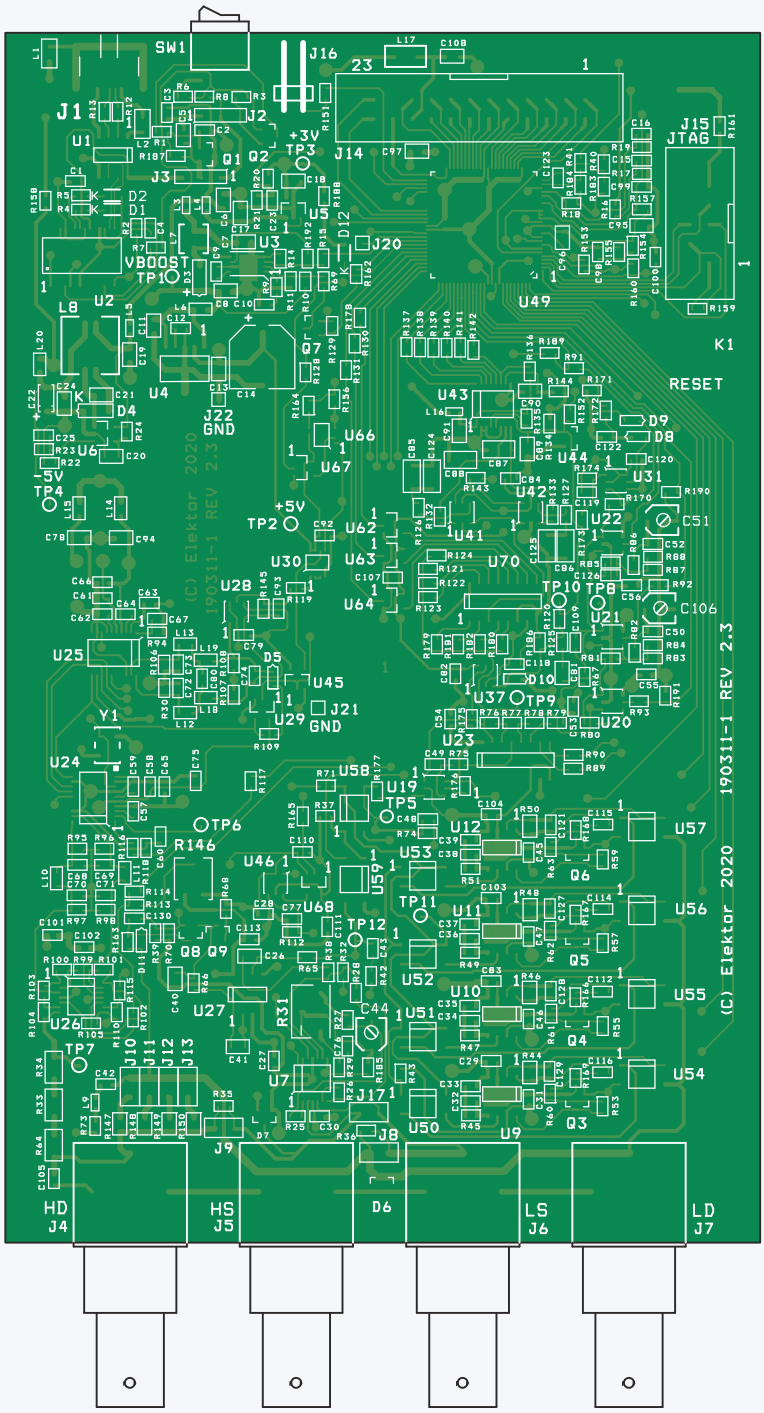


Figure 14a: PCB layout of the main board, all parts are mounted on the top side.

Accessories

The device is powered (5 VDC) by the Mini-USB type B connector. So, you need a USB cable compatible with the device side Mini-USB type B (Male) connector and probably USB type A (Male) on the power side (that of a smartphone is suitable).
As the measurement is performed by a 5-Terminal configuration [2] to minimize the influence of the measurement cables, there are 4 BNC connectors: from left to right, in front: HD, HS, and LS, LD (H for high, L for low, D for drive and S for sense). Special probes must be used for this type of measurement.
A Kelvin Clip for LCR Meter with 4 BNC Plug Test Wires Cable (**Figure 15a**) is the most affordable solution (about 10€ on eBay). It is recommended to shorten the cables as

COMPONENT LIST MAIN PCB 190311-1 V2.3

Resistors

(SMD 0603 1% 100ppm unless otherwise stated)

R1,R4,R5,R8,R20,R25,R26,R37,R66,R109,R124, R164,R188 = 1k
R2,R7,R10,R14,R22,R60,R61,R62,R63,R151, R187,R192 = 10k
R3,R35,R36,R38,R65,R71,R112,R113,R129,R135, R136,R160,R183,R184,R185 = 100k
R6,R15,R45,R47,R49,R51,R67,R82,R89,R90,R119, R122,R128,R130,R131,R137,R138,R139,R140,R141, R142,R144,R162,R165,R178 = 51Ω
R9 = 27k
R11,R171 = 2.4k
R12,R13 = 27Ω
R16 = 3.6k
R17,R172 = 1.6k
R18,R132,R133,R173 = 3k
R19,R21,R110,R115 = 2k
R23,R24,R120 = 30k
R27,R28,R29,R32 = 2k 0.1% 25ppm
R30,R80,R106,R107,R108 = 620Ω
R31 = 50Ω trimmer (Bourns 3266W)
R33,R34,R64 = 33Ω 100ppm 1206
R39,R70,R84,R126,R127 = 100Ω
R40,R41 = 110k
R42,R123 = 240Ω
R43 = 560Ω
R44,R147 = 100Ω 0.01% 50ppm 0805
R46,R148 = 1k 0.01% 10ppm 0805
R48,R149 = 10k 0.01% 10ppm 0805
R50,R150 = 100k 0.01% 10ppm 0805
R53,R55,R57,R59,R180 = 20k
R68,R125,R177,R189,R190,R191 = 510Ω
R69,R174 = 12k
R73,R88,R166,R167,R168,R169 = NC
R74 = 47k
R75,R121,R153,R161 = 10Ω
R76 = 150Ω
R77 = 91Ω
R78,R79 = 30Ω
R81 = 200Ω
R143 = 0Ω
R83 = 1.2k
R85,R95,R96,R97,R98 = 390Ω
R86 = 75Ω
R87 = 43Ω
R91,R92,R93 = 1.3k
R94 = 6.8k
R99,R116,R117,R154,R155,R159 = 4.7k
R100,R101,R102,R103,R104,R105,R118 = 3.9k
R114 = 300Ω
R134 = 2.7k
R145 = 7.5k
R146 = 10k trimmer (Vishay TS53YJ)
R152 = 47Ω
R156,R170 = 1.5k
R157 = 56k
R158 = 160k
R163 = 200k
R175,R176 = 33k
R179 = 5.1k
R181 = 51k
R182 = 75k
R186 = 120k

Inductors

L1,L2 = Bead Murata BLM21AG102SN1D
L3,L4,L5,L16 = 470nH TDK MLZ1608DR47DT
L6,L14,L15,L20 = 10µH TDK MLZ2012M100WT
L7 = 2.2µH Coilcraft 1008PS-222KLB
L8 = 22µH Cooper Bussmann DRQ73
L9 = NC
L10,L11,L12,L13,L18,L19 = 33µH TDK MLZ2012M330WT
L17 = 22µH Taiyo Yuden CBC3225T220MR

Capacitors

(0603 unless otherwise stated)

C1,C2,C3,C4,C9,C15,C16,C23,C27,C28,C29,C30, C31,C45,C46,C47,C53,C54,C55,C56,C58,C59, C60,C61,C62,C63,C65,C66,C67,C75,C79,C82, C83,C84,C92,C98,C99,C100,C101,C102,C103, C104,C107,C110,C111,C113,C118,C119,C120, C123 = 100nF 50V 10% X7R
C5,C6,C8,C19,C24,C26,C40,C41,C78,C89,C90,C91, C94,C108 = 4.7µF 10V 10% X7R 0805
C7,C11,C13,C17,C18 = 2.2µF 10V 10% X7R 0805
C10 = 270pF 50V 5% NPO
C12,C57,C64,C77, C130 = 10nF 50V 10% X7R
C14 = 470µF 6.3V EEEFP0J471AP PANASONIC (Aluminium Electrolytic)
C20,C21,C81,C95,C96,C97 = 1µF 25V 10% X7R 0805
C22 = 33µF 6.3V CASE-A (Solid Tantalum)
C25 = 1nF 50V 5% NPO 0603
C32,C34,C36,C38,C50,C93 = 1.5pF 50V 0.25pf NPO
C33,C35,C37,C39 = 3.9pF 50V 0.25pF NPO
C42,C48,C49,C105,C112,C114,C115,C116,C127, C128,C129 = NC
C43 = 180pF 50V 5% NPO
C44,C106 = 2-10pF trimmer (Knowless Voltronics JZ100)
C51 = 4.5-20pF trimmer (Knowless Voltronics JZ200)
C52 = 240pF 50V 5% NPO
C68,C69,C70,C71 = 220pF 50V 5% NPO
C72,C73,C109 = 82pF 50V 5% NPO
C74 = 47pF 50V 5% NPO
C76 = 6.8pF 50V 0.25pF NPO
C80 = 68pF 50V 5% NPO
C85,C86,C124,C125 = 100nF 50V 5% NPO 1206
C87,C88 = 15nF 50V 5% NPO 1206
C121 = 1pF 50V 0,25pF NPO 0603
C122 = 100pF 50V 5% NPO 0603
C126 = 22pF 50V 5% NPO 0603
C112, C114, C115, C116, C127, C128, C19 = NC

Semiconductors

D1,D2,D12 = red LED, low power 0805
D3,D4 = Schottky diode MBR0520
D5,D11 = diode 1N4148WS
D6,D7 = dual diode BAV199 LT1G
D8,D9,D10 = Schottky diode NSVRB751V40T1G
Q1 = P-channel MOSFET IRLML6402
Q2,Q3,Q4,Q5,Q6,Q7 = N-channel MOSFET BSS123
Q8,Q9 = RF NPN BFR106
U1 = USB6B1 ESD protection
U2 = FT232RL USB to UART
U3 = LT1946A DC/DC switching regulator
U4 = REG103GA-5 linear regulator 5V 500mA
U5 = TLV70030DDCR linear regulator 3V 200mA
U6 = LM2611AMF Buck converter
U7,U26 = LMH6644MT quad OPAMP
U9,U10,U11,U12 = AD8099ARDZ OPAMP
U19,U37 = LMH6618MK OPAMP SOT23_6
U20,U21,U22 = LMH6720MF OPAMP SOT23_6
U23 = 74HCT4052D analog multiplexer
U24,U25 = AD9834BRUZ DDS IC
U27 = OPA2727AID precision amplifier
U28,U31 = TLV3501AIDB analog comparator
U29 = 74LVC1G04DBVT inverter
U30,U66 = 774LVC1GDCTR flip-flop
U41,U42,U46 = 74LVC1G3157DBVR analog switch
U43 = ADS1246IPW 24-bit ADC
U44 = LM4050CIM3-2.5 voltage reference 2.5V
U45,U62,U63,U64,U67,U68 = 74LV1T34DBVR buffer
U49 = 8-bit MCU C8051F120-GQ
U50,U51,U52,U53,U58,U59 = AQY221N2S Solid State relay
U54,U55,U56,U57 = AQY221R2S Solid State relay
U70 = 74HCT4051D analog multiplexer

Miscellaneous

J1 = USB_B_Mini (TE Connectivity 1734035-2)
J2,J3 = 3-way SIL 2.54mm
J4,J5,J6,J7 = BNC PCB-mount (TE Connectivity 1-1337543-0)
J8,J9,J17,J10,J11,J12,J13,J16 = 2-way-90° SIL 2.54mm
J14 = 24-way boxheader
J15 = 10-way boxheader
J20,J21,J22 = test point
K1 = Optional 6mm push button (Omron BF3-1020)
SW1 = toggle switch Nidec Copal AS1D-5M-10-Z
Y1 = 27 MHz oscillator Epson X1G0044510005 SG5032CAN 27 MHZ TJGA
PCB 190311-1 VER 2.3

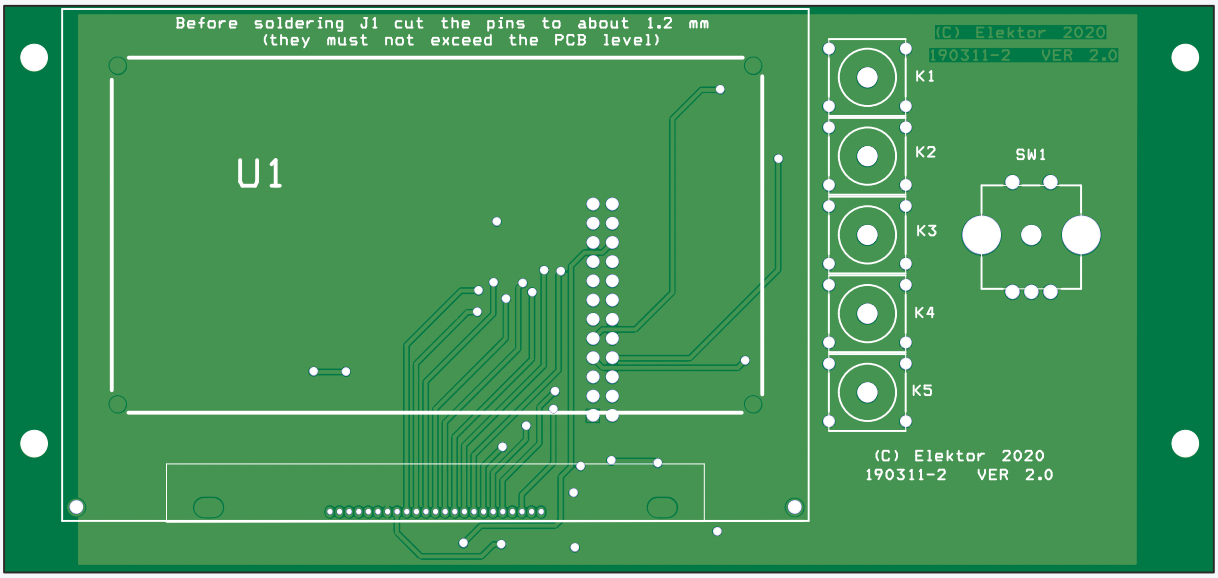
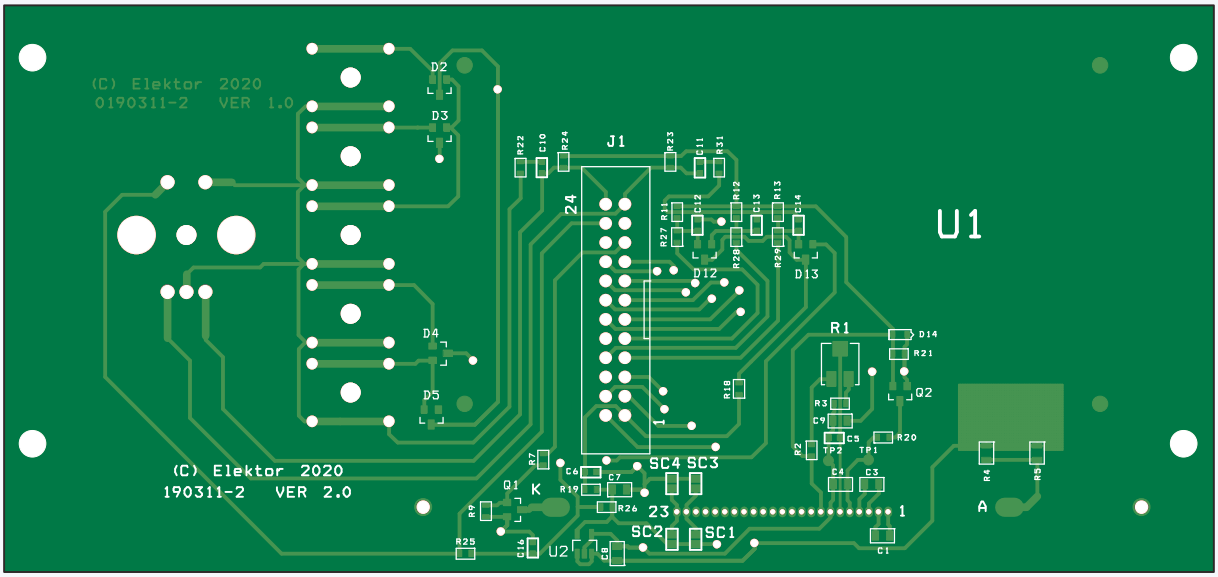


Figure 14b: PCB layout of the front panel board, bottom side. ↑

Figure 14c: PCB layout of the front panel board, top side. ↓



much as possible. The TH26001A 4 terminal test fixture (**Figure 15b**) is a more expensive (about 55€ on eBay), but has the advantage of very short and sturdy connection between DUT and LCR meter.

Accuracy

The accuracy obtained depends mainly on

- the type of component tested (L, C or R) and its value,
- the test frequency which must be appropriate for this component.
- the test device (Kelvin clips, test mixture…), again depending on the test frequency.
- the care taken in the calibration (to be redone If the temperature has changed significantly).

It is the impedance Z of the component which conditions the measuring range (choice of I/U converter) and the activation (or not) of the PGA.

The accuracy will be maximum (0.1%) for 50Ω < Z < 330 kΩ and depending on the range for frequencies < 100 kHz.

For extreme values it should be around 1%.

For frequencies > 1 MHz the accuracy degrades a little, up to 3%.

Figure 14d: Main and frontpanel PCBs connected for testing

COMPONENT LIST DISPLAY PCB 190311-2 V2.0 & ENCLOSURE

Resistors
(SMD 0603 unless otherwise stated)
R1 = 500k
R2 = 620k
R3,R19 = 100k
R4,R5 = 33Ω 0805
R7 = 51Ω
R6,R8 = NC
R9,R11,R12,R13,R23,R24,R26 = 4.7k
R18 = 0Ω
R20 = 10k
R21 = 1M
R22,R25,R27,R28,R29,R31 = 330Ω

Capacitors
C1,C3 = 4.7uF 10V 10% X7R 0805
C4 = 1uF 25V 10% X7R 0805
C5 = 100nF 50V 10% X7R 0603
C6,C10,C11,C12,C13,C14,C16 = 220nF 50V 10% X7R 0603
C7,C8,C9 = 2.2uF 10V 10% X7R 0805

Semiconductors
D2,D3,D4,D5,D12,D13 = dual diode BAT54A SOT23
D14 = diode 1N4148WS SOD323
Q1,Q2 = N-channel digital MOSFET FDV303N SOT23
U1 = Display MCCOG240128A6W-FPTLW (MIDAS)
U2 = TLV70030DDCR linear regulator 3V 200mA SOT23-5

Miscellaneous
J1 = 24-way boxheader (i.e. T821 series Amphenol)
SW1 = Rotary encoder PEC11R-4115F-S0018 (BOURNS)
PCB 190311-2 VER 2.0

Enclosure: HAMMOND 1455N1601
4 x Standoff hex female/female M3 7 mm
1 x Standoff hex female/female M3 6 mm
1 x Flat washer steel 3 mm
5 x Machine screw, countersink head M3 16 mm
5 x Washer M3
5 x Nut M3

2 x IDC connector socket 24 contacts (AMPHENOL T81214A101CEU)
24-way flatcable 15 cm
Rotary knob aluminium (e.g. MENTOR 507.61)
4 x Jumper 2.54mm gold plated

Figure 15a: Kelvin cables, available for a reasonable price.

Figure 15b: TH26001A 4 terminal test fixture, more expensive.

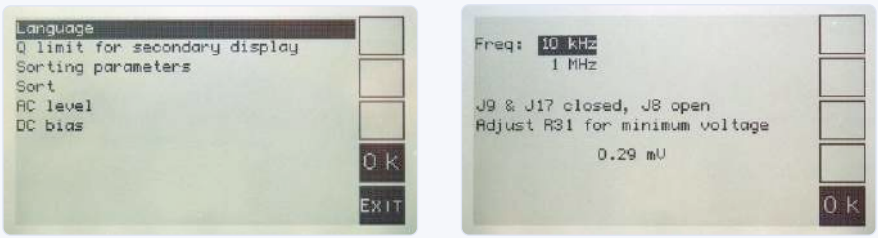


Figure 16: Two screenshots of the Calibration Menu, the many subtle options of which we will discover in the 2nd part of this article. The simplicity of its calibration is one of the major assets of this device.

Temporary Conclusion

In the second part of this article, to be published in the next issue of Elektor Magazine, the main topics will be:

- Stand-alone mode and PC mode (in this display-less configuration, you communicate with the device via the screen on your PC).
- Calibration, first use with a blank microcontroller, and software updates.
- Resistance, capacitance and inductance measurement operations.
- And of course the description of the configuration and calibration menus (**figure 16**).The six screenshots below (**figure 17 & 18**) give an overview of some typical measuring situations.
- We will also answer the questions that this first part will certainly raise.

Our readers can already download the documentation (consisting of two manuals) and all files related to the project (Gerbers included).

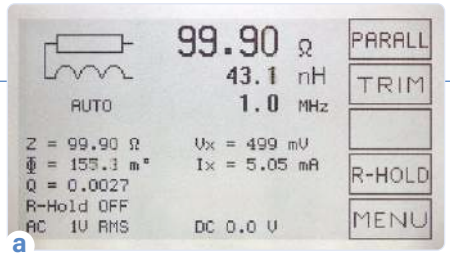


Figure 17a: Measuring a low-value resistor (100Ω), the parasitic inductance (43.1nH) of which is revealed at high frequency!

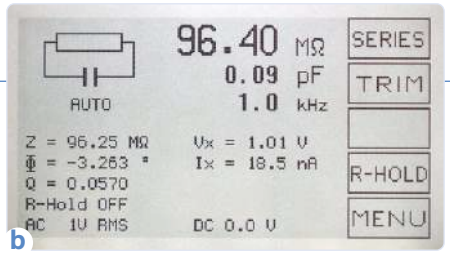


Figure 17b: Measuring of a high value resistor (100MΩ) the parasitic capacitance (0.09pF) of which is revealed at low frequencies!

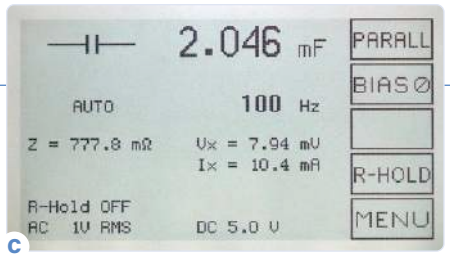


Figure 17c: Measuring a large capacitor at low frequency and 5V DC bias.

Figure 18a: Measuring an inductor with bias current

Figure 18b: Measuring an inductor at higher frequency.

Figure 18c: Measuring an inductor without bias current.

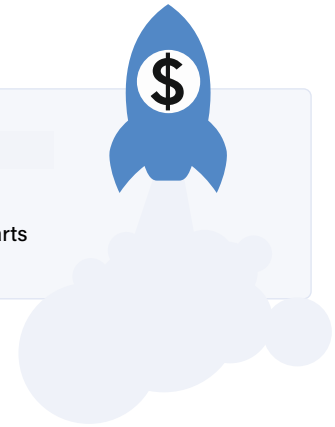
Comparative test

In the next issue, we will present the comparative measurements made with the new AU2019 LCR meter and reference instruments such as HP 4263B-1 4263B-2, Hameg HM8118, Keysight (Agilent) U1732A).

190311-01

RELATED PROJECTS

- **Elektor "Kickstarter" Project:**
  Kit including Main Board + Display Extension Board + all parts
  www.elektor.com/lcr



WEB LINKS

[1] **Qt**: https://www.qt.io/

[2] **The Impedance Measurement Handbook (2003 – Agilent Technologies Co. Ltd) § 3.1.4**: https://wiki.epfl.ch/carplat/documents/rcl_agilent.pdf

[3] **Full download (Gerber, drill&mill, BOM, ...)**: www.elektormagazine.com/190311-01

[4] **Test Fixture Compensation Techniques in Impedance Analysis**: http://www.eejournal.ktu.lt/index.php/elt/article/view/11206/5939

[5] **Operation Manual (latest version)**: www.elektormagazine.com/190311-01

[6] **User Manual (latest version)**: www.elektormagazine.com/190311-01

[7] **New LCR-meter page on Elektor Labs**: www.elektormagazine.com/labs/remake-lcr-meter

[8] **LCR-1- 2013-03-US2013030121.pdf – March 2013 p.12**: www.elektor-magazine.com/110758

[9] **LCR-2- 2013-04-US2013040221.pdf – April 2013 p.22**: www.elektor-magazine.com/130022

[10] **LCR-3- 2013-05-US2013050181.pdf – May 2013 p.18**: www.elektor-magazine.com/130093

[11] **LCR-4- 2013-06-US2013060641.pdf – June 2013 p.64**: www.elektor-magazine.com/110758

[12] **LCR-5- 2013-07-US2013070841.pdf – July-August 2013 p.84**: www.elektor-magazine.com/130174

[13] **LCR-6- 2013-11-US2013110221.pdf – November 2013 p.22**: www.elektor-magazine.com/130307

Questions or Comments?
Do you have questions or comments about his article? Email the author at jjacques.aubry@free.fr.

Contributors

Idea, Design, Text and Illustrations: **Jean-Jacques Aubry**

Schematics: **Kurt Diedrich**
Editors: **Luc Lemmens, Denis Meyer**
Layout: **Giel Dols**

Comparative measurements : **Alfred Rosenkränzer**

From **Ding-Dong Door Chime** to **IoT Doorbell**

Integrate your doorbell in Home Assistant with ESPHome

Contributors
Idea, Design, Text and Illustrations: **Clemens Valens**
Schematic: **Kurt Diedrich**
Editor: **C.J. Abate**
Layout: **Giel Dols**

In a previous article [1], I presented the open-source home automation projects ESPHome and Home Assistant. I also explained how to install them and how to get started. This article shows how I connected the doorbell of our home to the system. The main goal was to detect doorbell ring events in Home Assistant so that they can be relayed to places where the doorbell itself cannot be heard.

Figure 1: How to turn a basic door chime into a complex connected doorbell system.

The pushbutton of our electromechanical Ding-Dong door chime was replaced many years ago by a wireless door carillon featuring multiple ringtones. It came with two remote stations which greatly increased the range of the doorbell. Unfortunately, everything is powered from batteries, and when these wear out, the carillon becomes unreliable. Also, it tends to confuse visitors as they do not hear anything when they press the button. At our door the postman always rings twice.

As I had recently begun experimenting with home automation, it was a good opportunity to update the doorbell and integrate it into the system. To avoid running cables, I opted for a Wi-Fi connection between the doorbell and the home automation controller.

The old chime itself and its connecting wires, including the power supply, had remained in place. This turned out to be rather practical as the chime offered enough space inside to fit a home-brew wireless doorbell interface in and to power it as well.

Not wanting to ditch the working wireless door carillon, I decided to use its pushbutton for my new connected doorbell. This way I ended up with a three-generation Wi-Fi-connected door chime/carillon that plays one of several digital ringtones and produces a mechanical Ding-Dong sound at the same time while sending notifications to the cloud (or do some other useful action), see **Figure 1**. Audible feedback for the visitor has been restored as well.

Build a doorbell interface

For the Wi-Fi doorbell interface, I chose an ESP8266-based NodeMCU board (**Figure 2**) because it has an analogue input that I wanted

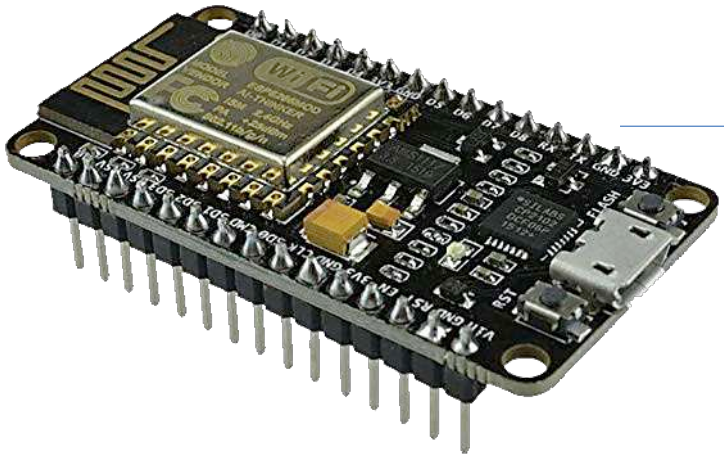


Figure 2: The NodeMCU module has a voltage divider on its analogue input, which divides the input voltage by 3.2.

to use (more on this below). Also, it does not need a USB-to-serial converter, so, besides a soldering iron, no special tools are needed for this project.

As mentioned above, the button of the wireless door carillon is connected to the doorbell interface using the old, existing wires. Because the pushbutton is outside and exposed to the elements, I protected the connection with series resistors, a filter capacitor, and clamping diodes.

The door chime's solenoid needs about one ampere when powered from 12 volts (AC). This is way too much for the carillon's tiny tactile pushbutton (such contacts usually can handle only up to 50 mA or so), but with a relay or power transistor it can be done. I used a small 5-volt relay I had lying around controlled by the MCU through a transistor.

The complete schematic is shown in **Figure 3**.

Monitor a battery voltage

The wire to the carillon's pushbutton connects directly to the cold side of the pushbutton's pull-up resistor. This means that when the pushbutton is not being pressed down, the wire carries the voltage of the battery mounted inside the pushbutton's enclosure. I therefore connected the wire not only to a GPIO pin of the MCU but also to its analogue input. This allows Home Assistant to keep an eye on the wireless doorbell's battery level as well as relaying doorbell ring messages. This is practical because the wireless range of the carillon is proportional to the battery voltage. When the voltage becomes too low, the remote station(s) can no longer be reached, and the system becomes unreliable. Note that it is, of course, possible to use the

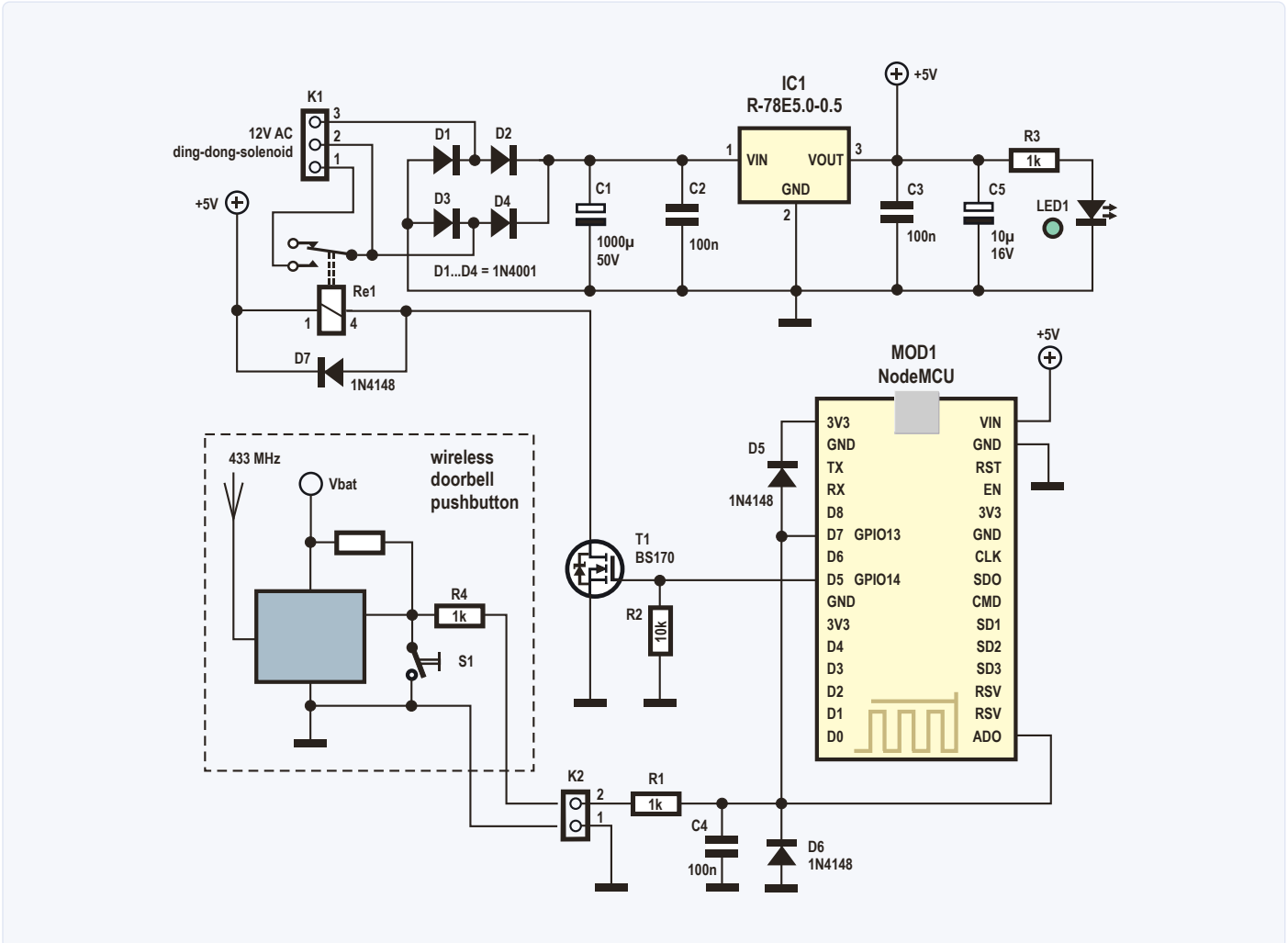


Figure 3: The doorbell interface needs a clean power supply. The pins connected to the pushbutton require some protection as the latter is placed outside where it is exposed to the elements.

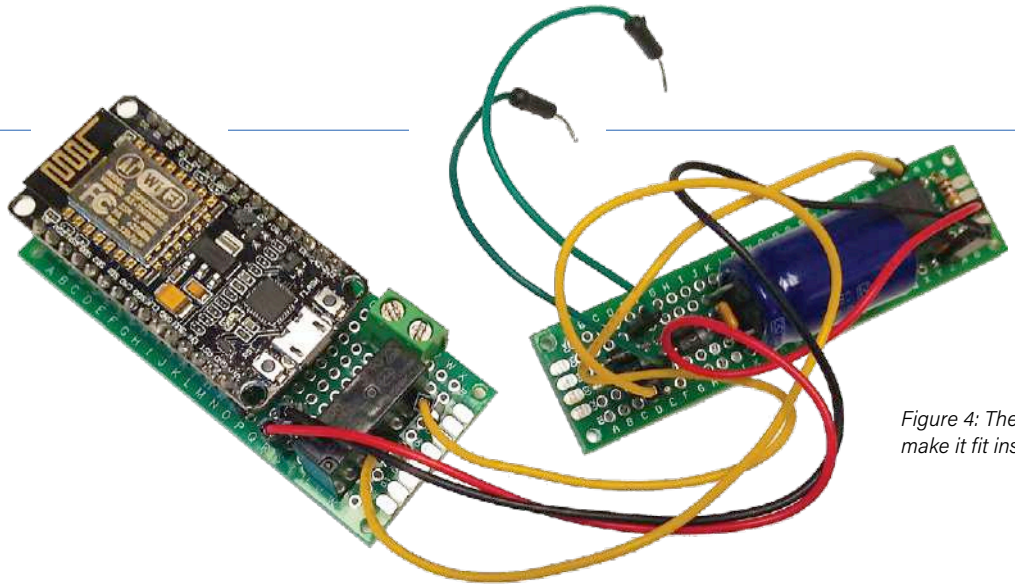


Figure 4: The doorbell interface was built on two boards to make it fit inside the old door chime.

A SHORT REMINDER

ESPHome is open-source firmware for an ESP8266- or ESP32-based remote device like a sensor or a switch (and much more).
https://esphome.io/

Home Assistant is an open-source home automation controller project that interconnects remote devices like sensors and switches (and much more) with actuators and services adn the like, allowing them to be controlled by complex rules.
https://www.home-assistant.io/

See [1] for more information.

analog input for detecting button presses as well as for monitoring the battery level, but this complicates the firmware. By adding a GPIO pin that was left over otherwise, this complication disappears.

Power supply

The power supply required some attention as the old chime only has a 12-volt transformer while I needed 5-volt DC for the relay and the NodeMCU board.
The chime's solenoid is quite a heavy load for the transformer, and it makes the transformer's output voltage sag when activated. A large reservoir capacitor filters out the dips caused by the solenoid. Without it, the NodeMCU module reboots at every button press.
The 5-volt regulator is a small 7805-compatible switch-mode regulator which helps to avoid heat dissipation problems.
A power indicator was added too because the NodeMCU board doesn't have one and it turned out to be practical to have.

By spreading out the doorbell interface circuit over two prototyping boards – one for the power supply, the other for the rest – I managed to fit it all snugly inside the old chime's enclosure (**Figure 4**).

ESPHome firmware configuration

The software for the doorbell interface makes use of ESPHome. The advantage of doing so is clear: about 30 minutes of programming (or should I say typing and editing?) is enough to come up with a fully functional device featuring Over-the-Air (OTA) programming, its own web interface and perfect integration into Home Assistant. What's more, it can be modified at will whenever you feel like it with your smartphone (or tablet or computer).
Programming is not done in C, C++ or Python, but in YAML. YAML is not a programming language, but a markup language that only provides rules to format text. ESPHome takes a YAML script or configuration file as input and transforms it into C/C++ code which is then added to and compiled with the rest of the library. The script specifies the parts from ESPHome you want to include (Wi-Fi settings, web interface or not, things like that), how and which kinds of sensors and actuators are connected to MCU pins, and what to do with them. Most common things are predefined, which makes things pretty easy and quick to set up.
YAML may take some time to get used to, but it is not complicated as long as you respect indentation.

Binding the relay to the pushbutton

The relay that controls the chime's solenoid is defined as an internal 'switch' connected to GPIO14. Specifying it as 'internal' makes it invisible for Home Assistant, which avoids cluttering your system.

```
switch:
  - platform: gpio
    pin: GPIO14
    id: ding_dong_relay
    internal: true
```

The switch is given an ID so it can be used in automations (see below).
The carillon's pushbutton is a 'binary_sensor' on pin GPIO13. As it is active low, we inform ESPHome about it with the 'inverted' keyword.

```
binary_sensor:
  - platform: gpio
    name: "my doorbell
  pushbutton"
    pin:
      number: GPIO13
      inverted: true
    on_press:
      then:
        - switch.turn_on:
```

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

```
ding_dong_relay
      - delay: 200ms
      - switch.turn_off:
ding_dong_relay
```

When a pushbutton is pressed, ESPHome generates an 'on_press' event which it sends to Home Assistant. However, these events can also be used as signals inside the YAML file to trigger one or more actions. This is called 'automation' and constitutes a powerful feature of ESPHome.

Here we use the 'on_press' event to produce a 200 ms pulse on the relay output. The relay's ID is specified here to accomplish this. Now the chime will ding-dong when somebody presses the button, even when the network is down. In short, this restores the original chime as it was installed more than 40 years ago, but instead of just a simple, direct pushbutton connection, we inserted several thousand transistors.

Keep an eye on the battery level

The carillon's battery is defined as a 'sensor' on A0. Its type ('platform') is 'adc' since it is a voltage that must be converted into a digital value. The NodeMCU module divides the input voltage on A0 by 3.2. In ESPHome, a 'filter' allows modifying a sensor's output value, and a multiplier is considered a filter. Therefore, if we multiply the sensor's value by 3.2, we get the original value back.

```
sensor:
   - platform: adc
     name: "doorbell battery
   level"
     pin: A0
   filters:
   - multiply: 3.2
```

Specifying lights

The NodeMCU module has a Flash button on GPIO0 and an LED on GPIO16 that can be used too. The Flash button can be a test button; the LED might provide a visual indication that someone has rung the doorbell and

PROGRAMMING ESPHOME ON A VIRGIN DEVICE

Because a new (virgin) ESP8266 or ESP32 device is not yet ESPHome compatible, the first time it must be programmed over the serial port. Depending on the device, you may have to install a USB-to-serial-port driver on the computer on which you do your ESPHome development (i.e., the computer running Home Assistant) to make this work. The NodeMCU boards with a Silabs (Silicon Laboratories) CP2102 USB chip that I used worked straight out of the box on a Raspberry Pi.

After connecting the device to the computer running Home Assistant, open the ESPHome Dashboard either from the sidebar (if you enabled this option) or by clicking 'Open Web UI' from the add-on card in the Supervisor Dashboard view. Check that the serial port is available from the drop-down list in the upper-right corner that defaults to 'OTA (Over-The-Air)'. If it doesn't, restart the ESPHome add-on (by going back to the Supervisor Dashboard). That should do the trick.

The next step is to click the pinkish '+' button on the ESPHome Dashboard. This will open a wizard that will guide you through the first part. Note that I have never specified an access password for any device, but that might be stupid of me.

As upload port, select the serial port to which the device is connected.

RELATED PRODUCTS

- **NodeMCU ESP8266 microcontroller board**
  www.elektor.com/nodemcu-microcontroller-board-with-esp8266-and-lua
- **Book: IoT Home Hacks with ESP8266**
  www.elektor.com/iot-home-hacks-with-esp8266

Home Assistant saw the event. Therefore, I defined GPIO0 as a binary sensor (added to the binary-sensor section).

```
light:
   - platform: binary
     name: "LED1"
     output: led1

 output:
   - platform: gpio
     id: led1
     pin:
       number: GPIO16
       inverted: true
```

Mainly to illustrate the concept, I defined a 'light' connected to an output called 'led1.' This will create a 'light entity' in Home Assistant, which offers other possibilities than a switch does. Also, it has a different icon. The LED itself (on GPIO16) is defined as an output with ID 'led1.' Note that it is active low ('inverted' is 'true').
Now, when Home Assistant sends a message to light 'LED1' to switch it on or off, the light will forward the message to the output with the corresponding ID, which is 'led1.'
The complete YAML configuration file for ESPHome can be downloaded from [2]. Note that you will have to change the SSID and password to suite your Wi-Fi network. You may also want to change the name of the device.

Use the ESPHome add-on for Home Assistant

As I have the ESPHome add-on installed in Home Assistant, I wrote the YAML script in the add-on. That way, the device's YAML file is already in the system, making OTA updates later easy. The first time, however, at least with a virgin ESP8266 system, you must use the serial port to program the MCU since OTA programming has not been activated yet (see inset).

Automation in Home Assistant

The LED on GPIO16 was only declared in ESPHome's YAML file as being available, we did not specify an automation rule for it, and so it will not do anything without intervention from Home Assistant. The idea was to make it light up when someone rings the bell. This can be done with a simple automation in Home Assistant. All it must do is wait for a button press event and switch on the LED when one comes along.
There are several ways in Home Assistant to open the automation editor. One way is to click 'Configuration' in the menu on the left followed by a click on 'Automations' on the right. Create a new automation by clicking the round orange '+' button in the lower right corner.
You can now enter a phrase to let Home Assistant try to convert it into the desired automation. However, trying to come up with a suitable phrase is not easy and then having to fix the crippled automation rule created by Home Assistant is a waste of time, so here we click 'Skip' instead.
Enter a name for the automation, and, optionally, a description. Leave the mode on 'Single (default).'

Triggers and actions

There are several ways to specify a trigger event that will start the automation. To me, in this case the most intuitive way is to choose a 'Device' as 'Trigger type.' In the 'Device' list, look for the name that you specified in its YAML file. Similarly, search the 'Trigger' list for the name that you gave the pushbutton.
There are no conditions, so skip that part.
Actions are similar to triggers, and, like triggers, there is more than one way to obtain the result you are looking for. One way is to choose 'Device' as 'Action type' and select your device from the 'Device' list. From the 'Action' list select 'Turn on LED1' (if you named the LED differently, use that name instead).
Save your work by clicking the round orange floppy disk icon in the lower right corner.
Check if the automation works by ringing the doorbell. When you do this, and if it worked, you will notice that you have no means to switch the LED off again.

Add a light to the user interface

To remedy this, we will add a so-called card to the Home Assistant overview page. This will not only allow for switching the LED off (by clicking on the card), it also provides a remote view of the LED's state, so you don't have walk all the way over to the doorbell to see if it is on or not.
Click 'Overview' in the menu on the left. Click the three dots in the upper right corner ('Open Lovelace UI menu') and select 'Configure UI.' Now click the round orange '+' button in the lower right corner (in Home Assistant you mouse around a lot). Look for the 'Light' card and click on it. Search the 'Entity' list for the LED and select it. Enter a name too as that will make it easier to identify the light in the overview. For now, it may be your only light, but when your home automation system evolves there will probably be more and so a good name will be helpful.
Save the card and leave UI-configuration-mode by clicking the cross in the top left corner ('Close').
Clicking the card should now allow you to toggle the state of the LED.

Sounding off

If you followed the steps presented in this article (and corrected any mistakes I made, and adjusted it to the latest versions of the ever-evolving ESPHome and Home Assistant projects), you will now have a Wi-Fi-connected doorbell that you can automate in Home Assistant. You can make it do much more (useful) things than lighting an LED, but I'll leave that up to you. This article showed the basic principles to get started, and now it is up to you to improve things and adapt it to your needs and desires.

200089-01

WEB LINKS

[1] **Home Automation Made Easy**: www.elektormagazine.com/200019-01
[2] **How-To: Integrate Your Doorbell in Home Assistant Using ESPHome**: www.elektormagazine.com/labs/esphome-doorbell

Battery Management

What To Be Aware of When Using (Lithium) Batteries

By **Dr. Thomas Scherer** (Germany)

With modern batteries based on the alkali metal lithium, the storage of electrical energy has become more convenient but, at the same time, more complicated. You can store more energy per unit volume and they have hardly any self-discharge. However, when charging a multi-cell battery pack you need to be careful with their 'management'!

So, what is it about lithium batteries that, unlike other common battery technologies, results in this demand for electronic battery management? This article takes a look at that question and shows how it is dealt with.

Non-lithium cell technologies

Lead acid

The beauty of the old lead-acid batteries is that they can be safely charged by simply applying a DC voltage to the battery pack. Each cell requires a voltage source in the range of 2.15 to 2.35 V to charge. A 6-cell 12 V battery therefore needs a DC supply in the range of 12.9 to 14.1 V. As the cell voltage rises during charging, the current through the battery decreases as it reaches its Maximum Charge Voltage (MCV).

This type of battery is very tolerant to overcharging; at worse it may vent some hydrogen. If one of the cells in a battery pack has a little less capacity than the others, you can still charge the remaining cells to the appropriate MCV. They can tolerate a certain amount of abuse provided the cell voltage is not allowed to fall too low. They are so robust that a standard 44 Ah car battery can last for years without any maintenance while being fed from a 55 A alternator and relatively crude regulator…that's pretty impressive.

A lead-acid cell should not be discharged below 1.8 V which means a car battery will be damaged if its voltage falls below 10.8 V. The self discharge rate is about 5% per month, so it's important to keep a motorbike battery charged if it's not used during the winter months.

Nickel cadmium

Round-cell NiCd batteries were once a popular rechargeable replacement for the zinc-carbon batteries used to power consumer electronics devices. In the meantime, production of this type of battery has been banned in the EU for more than 10 years because

Figure 1: Typical NiMH cells in AA format offer low self-discharge.

Figure 2: The Prius II battery — 28 packs, each with 6 cells = 201.6 V nominal voltage.

of the high toxicity and environmental pollution caused by the heavy-metal cadmium used in this type of cell. NiCd batteries have fallen out of favour and production is unlikely to be resurrected.
Out of interest, NiCd batteries are able to handle high currents and tolerate many charging cycles. They were a popular choice for radio-controlled cars and the first cordless power tools. The nominal cell or mid-point voltage of a NiCd is 1.2 V. The method used for charging is similar to that of the lead acid battery but they have an MCV of 1.5 to 1.57 V per cell (depending on the current). Deep discharge below 0.85 V per cell damages the battery, even though it is generally much more robust than a NiMH or lithium battery. Self-discharge in this type of battery amounts to something like 20% per month. A series connection of these cells when charging is not an issue — therefore hardly any battery management is required.

Nickel-metal hydride

The small NiMH cells available in AA format as shown in **Figure 1** have largely replaced NiCd batteries as a non-toxic alternative. They were the standard choice to power cordless tools due to their low price and easy handling. Larger capacity cells are also available and are used in the Toyota Prius which was the first mass-produced hybrid vehicle. Models II and III of the Prius carry a hefty NiMh battery pack weighing almost a hundredweight.
This pack has a capacity of 6.5 Ah at 200 V. Charge and discharge characteristics are carefully monitored and utilization of partial battery capacity helps promote a long lifetime and supports a high number of recharge cycles. **Figure 2** shows the internals of the battery pack from my own Prius II. Lithium cells are now the preferred energy storage medium for most plug-in and newer hybrid models now that early teething problems associated with the technology have apparently been resolved.
The self-discharge of NiMH batteries is (usually) comparable to that of NiCd types. However, there are variants like the battery in **Figure 1** that have a much lower self-discharge of around 1.5% per month but at the expense of the energy density. These batteries are always preferable for use with gadgets that are used infrequently, as deeply discharged NiMH batteries become unusable.

Lithium

Lithium-based batteries are increasingly replacing all other types of batteries in most mobile and cordless-device applications. They can supply high levels of power and have good energy density. They also have an extremely low self-discharge rate of around 5% per annum. However, as we know there is no such thing as a free lunch and this is reflected in Li-ion (lithium-ion) batteries' high sensitivity to overcharging and deep discharge conditions. Not only can this type of abuse cause them to permanently lose capacity, some have been known to rupture or even burst into flames. These types of cells also suffer at low temperature as their available capacity decreases. Full capacity returns when the cell warms up, but this is of little consolation to drivers of electric vehicles during winter. The properties of Li-ion batteries also degrade at high temperatures. Due to their sensitivity to overcharging, caution is advised when connecting them in series to achieve higher voltages. Additional charging electronics are required here in the form of a 'charge balancer' that prevents individual cells with a slightly lower capacity from being overcharged when other cells in the pack are not yet fully charged. To ensure a long lifetime and optimal recharge cycling it should be made clear that Li-ion batteries always require a battery management system or BMS.
What makes things even more complicated is that there is not just one type of lithium battery. Depending on the technology and internal chemistry used, the possible number of cycles, the mid-point or Nominal Cell Voltage (NCV), the Maximum Charge Voltage (MCV), the End of Discharge Voltage (EODV), the maximum discharge current and other parameters will vary. The BMS must be tailored to work with these parameters and, of course, monitor current and cell temperature conditions in each specific case.
Lithium-ion battery packs used to power electric vehicles have sophisticated power management systems to control the battery environment and ensure a long service life. For more basic applications, such as providing power for cordless tools and other domestic appliances, the need for such strict monitoring is not usually necessary. Some mobile smart devices and smartphones typically use a single battery and, therefore, require no charge balancing at all. In such cases the charger will only need to monitor the cell voltage to detect the EODV as well as the maximum charge current.
The smaller Li-ion batteries fall into two main categories that differ mainly in the material the positive electrodes are made from. Some use cobalt, cobalt dioxide, manganese and/or nickel, and aluminum. The most widely used positive electrode material is $LiCoO_2$, which is a layered oxide. Fortunately, the nominal cell voltage of all these types is around 3.6 to 3.7 V, the MCV is 4.2 V, and the EODV is 2.5 V (exception: with $LiNixCoyAlzO_2$ the EODV is 3.0 V). Even though the maximum output current and service life may differ across the range of diverse electrode materials they can, in principle, all be managed by the same management ICs.
One exception to this are lithium iron phosphate ($LiFePO_4$) cells

that have the following characteristics: NCV = 3.2 V, MCV = 3.65 V and EODV = 2.5 V. This type of battery offers a high number of charge cycles and a slightly lower energy density. For multi-cell applications you will need appropriate charge balancers for this type of cell. The semiconductor industry naturally offers integrated solutions for both categories to cover various battery pack sizes and load ratings - some ICs are even programmable, switchable for certain parameters, and may even have an interface for a host microcontroller.

Li-ion batteries are available in many different designs. The LiPo (lithium polymer) types are mainly built into mobile devices and laptops because they can be made in a relatively flat package. The term polymer primarily refers to the fact that the LiPos are simply welded into a plastic film pouch instead of a solid housing – therefore, any kinks or punctures to the plastic enclosure must be avoided. There are various round cells, including the particularly widespread type 18650 (ø = 18 mm, length = 65 mm) which, astonishingly, is installed by the thousand as a pack in Tesla's electric cars, with the spaces between the round cells used for temperature management. **Figure 3** shows an 18650 cell made in China with an incredible specification: 7,800 mAh is a long way from fitting into the 18650 format. The best cells available today come in at around half this capacity in real terms. Finally, there are the 'prismatic cells', the name of which merely indicates that their housing is not round but cuboid. Such cells were used in the German 'Hotzenblitz' mini car manufactured in the 1990s. In **Figure 4** you can see that each cell is equipped with electronics to monitor its condition.

Figure 3: Typical lithium cell in 18650 format with extremely exaggerated capacity specification. Its integrated electronics help prevent deep discharge and overcharging.

Figure 4: Part of a LiFePo battery in the German 'Hotzenblitz' two-seater car with battery management electronics (Image: Joes-Wiki, CCASA 3.0 DE [2]).

A delicate balancing act

The term 'battery management' is all encompassing and describes systems that monitor and control recharging, state of discharge, and provide battery protection. Lead-acid, NiCd, and NiMH batteries are fairly tolerant to abuse so their battery management scheme is usually only concerned with details such as MCV, and EODV to control charging and sometimes to limit output current. The situation is more complex when a battery pack is made up of multiple, series-connected lithium cells. Here the BMS must implement a charge balancer.

The main reason for this is that lithium batteries are sensitive to overcharging. Even though mass production techniques ensure uniformity of production, it is inevitable that no two cells will be absolutely identical. In particular, each will have a slightly different energy storage capacity. This means there will always be one weak link in the chain of cells that has the lowest energy storage capacity and becomes fully charged before any other cell in the pack. Given their sensitivity to over-charging, a balancer circuit is essential to cut off charge to this cell before damage can occur. Cell performance also degrades over time so that differences become exaggerated.

A balancer in its simplest form for a two-cell battery pack is shown in **Figure 5** monitoring the voltages of two cells. If one of the two cells reaches the MCV, an appropriately small load or shunt is placed parallel to the cell so that the charging current is diverted to the cell that is still charging. If there are more cells, the balancer is scaled accordingly. **Figure 6** shows a small BMS board for a four (small) cell pack, available from eBay priced from €1 to €10. In **Figure 7** you can see how much effort went into balancing the 48 V LiFePo battery of my Segway clone. Each cell is monitored by its own microcontroller.

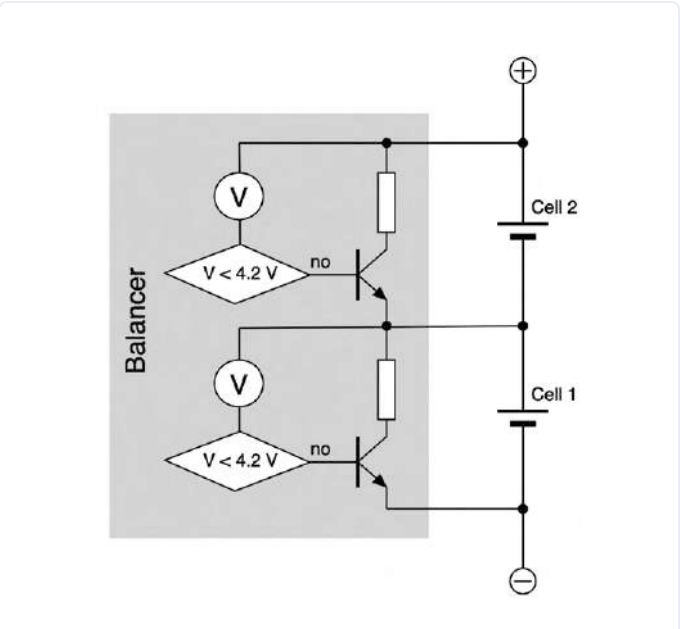


Figure 5: Schematic of a simple balancer for two cells.

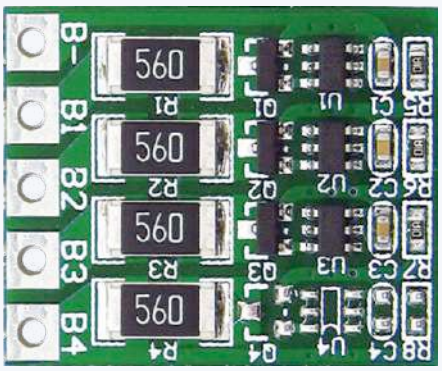


Figure 6: Such balancer boards can be found on eBay in multiple variants and designs at a low price.

Figure 7: LiFePo battery with balancer from the author's Segway clone.

You may think that the shunts look a little underrated for the level of current the batteries can deliver but, in practice, they only need to channel the charging current past the cell to be protected and this is significantly lower than the maximum charging current that can occur. The reason is that Li-ion batteries are initially charged at a nominal current until they reach a state of charge of 50 to 80%. From then on they charge with a small, usually steadily decreasing, current. With LiCoO2 cells the charge current is quite low above 3.95 V. Since the differences between healthy cells are not that great, it is not too troublesome to dissipate this significantly lower current level. In practice, shunts to handle around 1 to 5% of the maximum charge current are almost always sufficient. If it is not enough, it is usually an indicator that a particular cell is no longer usable.

Depending on the current level, energy is dissipated by the balancer circuit, which is quite wasteful. There are special balancing circuits available that can make use of the excess energy by feeding it back into the overall system via a step-up switching regulator (per cell!).

Another important criterion is cost: balancers use electronic components and ICs that are expensive in high voltage versions. Modern BMS ICs are typically available for packs of 2 to 12 cells. With 12 cells you already have an operating voltage of around 50 V. If you want more cells in a battery pack, you simply scale the BMS electronics. With the 800 V systems of modern electric cars, the pack can consist of 20 modules using 12s BMS ICs in series (12s denotes a battery pack containing 12 cells connected in series). The individual subsystems communicate with one another via a (electrically isolated) bus and then to a central controller.

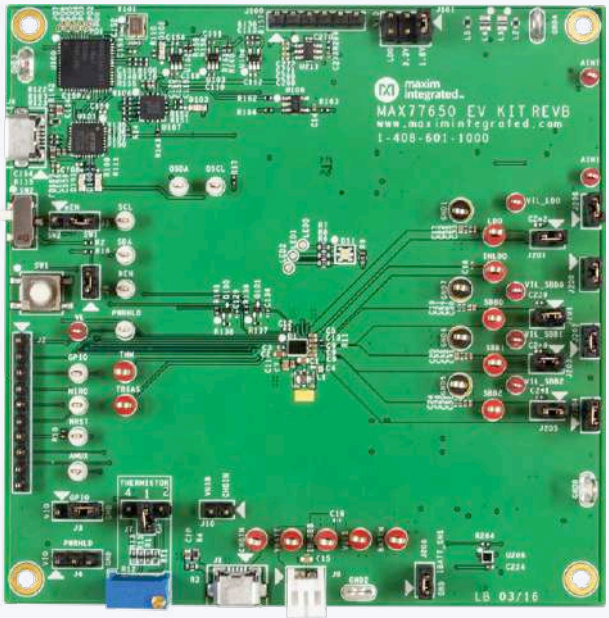


Figure 8: Evaluation kit for the MAX77650/MAX77651 BMS ICs (Photo: Maxim Integrated).

ICs

There are already a wide range of BMS solutions from various IC manufacturers available, too many to provide a comparative overview in the space available here. Instead we will take a look at a specific IC and associated circuits as an example.

In addition to balancing, a BMS for lithium batteries also has the task of protecting the battery from undervoltage and overcurrent. Sometimes the charging circuit is also included in the system. The use of a microcontroller makes perfect sense for such complex tasks. Modern BMSs therefore sometimes have a small integrated SoC or an interface to hook one up. BMS ICs are available from manufacturers supplying analog ICs or switching regulators, as well as being provided by companies such as Microchip, Maxim (see **Figure 8**) or Renesas. The example covered here comes from the latter, which is also well-known for its microcontrollers.

Figure 9 shows the internal block diagram of the BMS IC ISL94212. Along the left side you can see the connections for ground (at the bottom) and positive operating or battery voltage (at the top) and the connections for a total of twelve lithium cells. The voltages pass through the 'input buffer/level shifter' via two multiplexers (VC MUX and MUX) to an A/D converter. There are also reference voltage generators, blocks for measuring temperature, monitoring of the EODV and, of course, the most important thing, the fully digital block 'Control Logic and Communications'.

Figure 10 shows a basic application circuit with two controller ICs and 24 cells. A host microcontroller is shown that communicates with the chip via an SPI interface. Due to the high voltages of such module chains, the serial interfaces are not coupled directly, instead being linked via small capacitors.

The IC not only offers monitoring of many cells but also the required balancing. **Figure 11** shows the basic connection of one load resistor per cell (far left) via a small power MOSFET that is controlled via the associated IC output CBX. In this way, balancing can also be implemented for very high currents. The details of the many IC functions can be found in the data sheet [1].

Balancing in practice

If you are dealing with a lithium battery, you basically have a choice of one of three strategies. For simple applications, you can simply buy a small, inexpensive and fully assembled circuit board from a Chinese producer such as the one in **Figure 5**. If you are planning a larger and/or more expensive battery implementation, you can choose from a vast range of ready-made BMS solutions from the big-name manufacturers. Last but not least, it makes sense to tailor your solution for the battery package in question using one of the wide range of ICs available. In some cases, however, you can and must write your own code for monitoring and controlling the battery if you really want to do it accurately.

180350-04

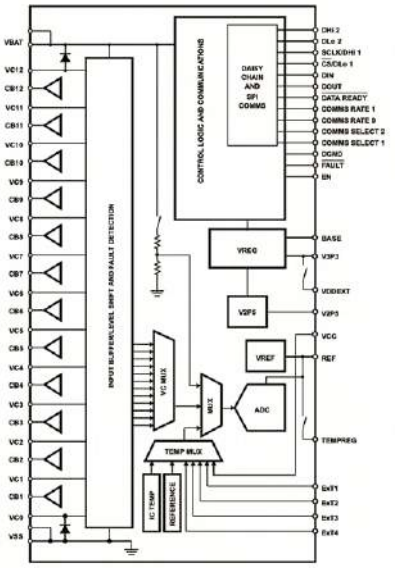


Figure 9: Block diagram of the BMS-IC ISL94212 (Photo: Renesas).

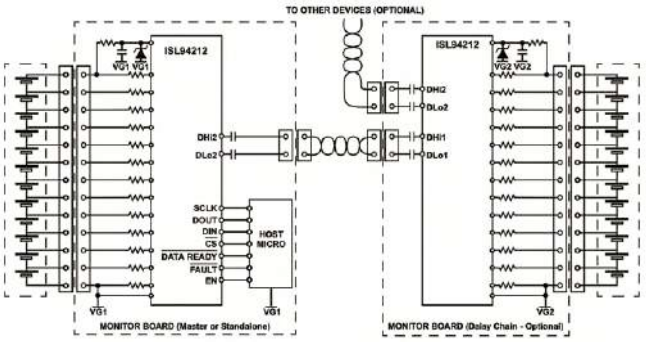


Figure 10: Basic configuration, scaling up the BMS to manage more than 12 cells with the ISL94212 (Graphic: Renesas)

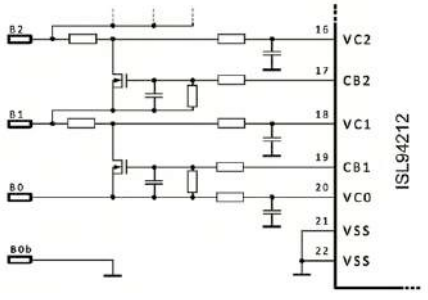


Figure 11: In the case of the ISL94212, external MOSFETS perform the balancing.

Contributors

Text and Images: **Dr. Thomas Scherer**
Translator: **Martin Cooke**
Editor: **Stuart Cording**
Layout: **Giel Dols**

Questions or Comments?

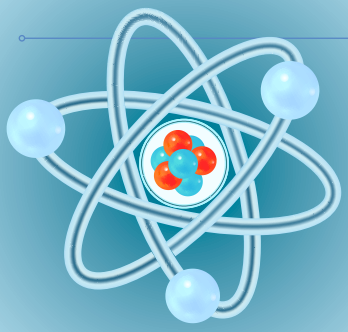
Do you have questions or comments about this article? Then email Elektor at editor@elektor.com.

RELATED PRODUCTS

- **USB Solar Panel Battery Regulator Charge Intelligent Controller (12/24 V, 10 A)**
  www.elektor.com/19126
- **OWON XDM2041 Multimeter (55000 Counts)**
  www.elektor.com/owon-xdm2041-multimeter-55000-counts
- **RS232 data logger (140126-41)**
  www.elektor.com/rs232-data-logger-and-spy-tool-140126-41

WEB LINKS

[1] **Datasheet for Renesas ISL94212**: www.renesas.com/en-us/www/doc/datasheet/isl94212.pdf
[2] **CCASA 3.0 DE**: https://creativecommons.org/licenses/by-sa/3.0/de/deed.en



Starting Out in Electronics (5)

Easier Than Imagined!

In the previous installment (in the September/October issue of your favourite magazine) we racked our brains over the H-circuit and saw that we'd better not (in most cases) ignore the resistance of a conductor in the real world. Now it is time to busy ourselves with alternating voltages.

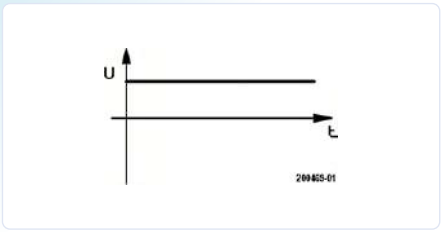


Figure 1: DC voltage.

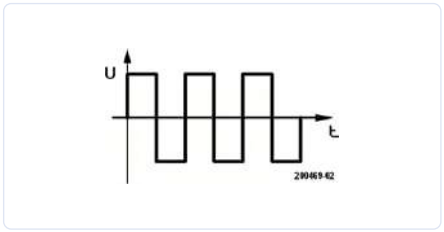


Figure 2: Alternating voltage (here in the shape of a square wave).

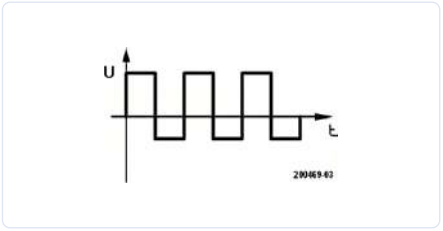


Figure 3: An AC voltage superimposed on a DC voltage.

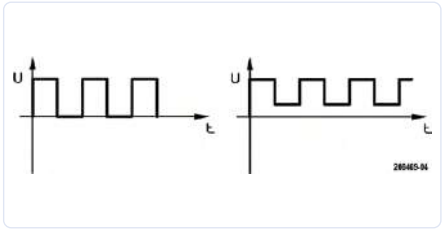


Figure 4: Pulsating DC voltage.

Up until now, in this series of articles, we have consistently and silently assumed that all the examples and calculations were about *DC voltages*. However, things really become interesting when we start looking at alternating voltages; an audio amplifier, for example, amplifies alternating voltage.

As is well-known, batteries generate a voltage with a polarity that does not change – this is a direct or DC voltage. The *magnitude* of this voltage changes (becomes lower) as the battery becomes exhausted, but the *polarity* doesn't change. In **Figure 1** we have drawn the textbook example of a DC voltage.

However, with an alternating voltage the polarity changes (more or less) periodically. Generally, the magnitude of the voltage varies as well (see later on), but with the alternating voltage shown in **Figure 2** (a square wave, and yes, that is really an alternating voltage!) the magnitude doesn't change; only the polarity does. It will be clear that the average value of the voltage (calculated over a long enough time period) of the alternating voltage in **Figure 2** is equal to zero; in this case we speak of a pure alternating voltage.

It is also easily imaginable that the average value of such a voltage is not equal to zero but its polarity, nevertheless, changes continuously. In such cases we have an alternating voltage that is superimposed on a direct voltage (that means, the voltage is the sum of a pure AC signal and a DC voltage). This is sketched in **Figure 3**.

And finally, in **Figure 4,** we have sketched a similar situation but with an important difference. Here we have a voltage that changes in magnitude periodically, but the polarity always remains the same. In this case we speak of a pulsating DC voltage.

Period and frequency

The amount of time it takes for an AC voltage to go through its *complete* cycle is called the period *T* of the AC voltage. This is expressed in seconds. The reciprocal of the period is the frequency, which is expressed in hertz (Hz):

$$f = \frac{1}{T}$$

In Europe, the mains frequency is 50 Hz (on the other side of the great pond it is 60 Hz). The period of the mains AC voltage is therefore:

$$T = \frac{1}{f} = \frac{1}{50\,Hz} = 0.02\,s = 20\,ms$$

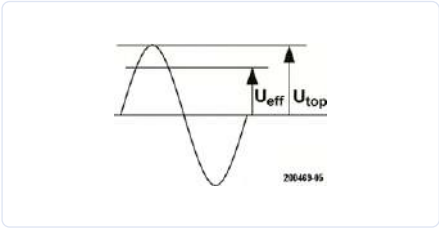


Figure 5: One complete period of a sinusoidal AC voltage.

Sinusoidal AC voltage

The AC voltage that we deal with the most is the sinusoidal AC voltage, as sketched in **Figure 5**. The voltage as a function of time is described by this expression:

$$U(t) = U_{top} \cdot \sin\left(\frac{2 \cdot \pi \cdot t}{T}\right) =$$
$$U_{top} \cdot \sin(2 \cdot \pi \cdot f \cdot t) = U_{top} \cdot \sin(\omega \cdot t)$$

The highest value that is reached by an AC voltage is called the peak value – and that will probably not surprise you. You will also not be surprised to learn that an AC value that reaches its peak value only some of the time is less 'effective' (that is, it develops less power) than a DC voltage with a size of the peak value. In order that DC and AC voltages can still be compared to each other, the concept of 'effective value' or 'RMS value' is introduced:

'The effective (RMS) value of an AC voltage is the value of a DC voltage that produces the same amount of heat in an ohmic resistor (i.e. a common resistor) as the AC voltage in question.'

An incandescent lamp, for example, lights up to the exact same brightness when powered from a DC voltage of 12 V or from an AC voltage of 12 V_{RMS}. Note: when the value of an AC voltage is given, this is always assumed to be the RMS value unless something else is explicitly stated.

Now, at this point we could torment (and/or bore) you with a lot of 'advanced' mathematics, but this falls outside the scope of this article series. You will have to take our word for it that the relationship between the peak value and the RMS value of a sinusoidal AC voltage is:

$$U_{top} = U_{eff} \cdot \sqrt{2}$$

And with that we can calculate something that you have perhaps wanted to know for a long time: A power outlet (in Europe) supplies an AC voltage of 230 V, and we now know that this is the RMS value, but what is the magnitude of the peak value of this AC voltage?

$$U_{top} = U_{eff} \cdot \sqrt{2} = 230V \cdot 1.41 = 324.3V$$

Crest factor

Now, let's cover one final concept before we leave (for a while) the theory behind us and focus more on the practical side of electronics. The ratio between the peak value and the RMS value of an AC voltage is often called the *crest factor* (also sometimes called amplitude factor or peak factor):

$$CF = \frac{U_{top}}{U_{eff}}$$

For a square wave (**Figure 2**) the crest factor is 1, for a sine wave it is √2 (as we observed above) and for a triangle wave the crest factor is equal to √3. An interesting fact is that the crest factor of a noise signal approaches infinity (the more 'random' the noise, the higher the crest factor). It is not all that likely that this knowledge is useful to you, but you never know...

Between theory and practice are...

Schematics and lists! You don't build an electronic circuit by placing a box of components on your workbench and then joining them together haphazardly. Quite the reverse: you think about what you want to build, and then sketch a schematic on a piece of paper of what you think the final circuit should look like. You will try this out on a breadboard or similar and, after a number of improvements, this results in the definitive schematic.

Based on this schematic you design a circuit board plus a layout of where the components in question are to be fitted (the component overlay). Or you build the circuit on perforated prototyping board, in which case a simple sketch of the wiring is usually sufficient.

A parts list is generally also useful – even if it is only used as the shopping list for acquiring the necessary components.

We won't explore this dimension of the electronics 'subject' any further; when you leaf through the pages of this issue of Elektor (or any other issue), you will see plenty of examples of schematics, circuit boards, component lists and so on. If you want to get a good impression of what it takes to realise a (commercial) electronics product, we refer you to the excellent series of articles 'From idea to product' from the pen of Clemens Valens, the first episode of which appeared in the 2019 September/October issue of Elektor. Next time we will cover the physical components that you will deal with in practice!

200469-03

The magazine article series "Starting Out in Electronics" is based on the book *Basic Electronics for Beginners* by Michael Ebner, published by Elektor.

Contributors
Idea and illustrations: **Michel Ebner**
Text: **Eric Bogers**
Editing: **Stuart Cording**
Layout: **Giel Dols**

Questions or comments?
Do you have any questions or comments prompted by this article? Send an email to the author or to the editor of Elektor via editor@elektor.com.

Small Circuits Revival

From the Elektor suggestions box

Compiled by **Eric Bogers** (Elektor)

Here are, once more, a few little circuits to keep you soldering and to sharpen your experimentation skills when you don't feel like beginning something really complicated.

idea: Michael A. Shustov (Russia)

Pseudo-thyristor

A conventional thyristor operates as a switch in a rather primitive manner: it switches on as soon as the control voltage is (briefly) applied, but to turn it off again, the power supply voltage has to be switched off.

The pseudo-thyristor in **Figure 1** operates in a completely different way. It is switched on by briefly connecting the input to the positive supply voltage; switching off is performed by connecting that same input briefly to ground. It is not difficult to understand how it works. When the power supply is turned on, transistor Q4 is off; Q1 and Q3 will conduct and the actual switch Q2 is off: the load (here in the form of lamp EL1) is switched off. LED D1 indicates that the circuit is ready.

When the input of the circuit is set briefly high (pressing of SB1, a push button with momentary contact), Q4 will start to conduct. As a consequence the base of Q3 is connected to ground: this transistor now turns off. The result is that Q2 will turn on and switch on the load. This situation continues when SB1 is released.

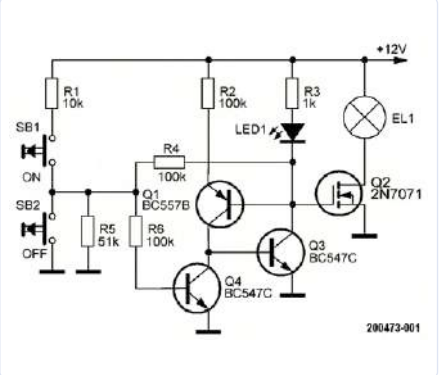


Figure 1: Using just a few transistors a pseudo-thyristor can be built that has better characteristics than a 'real' thyristor.

With a brief press of SB2 the circuit returns to its initial state and the load is turned off again. When SB1 and SB2 are pushed simultaneously, SB2 has precedence and the load will be switched off.

idea: Roel Arits (Netherlands)

Stunning Jingle Bells

Christmas is an exciting time — especially when there are packages under the Christmas tree ("That little package at the front left — are they the keys to my new Mercedes?"). And, with this circuit, we increase the tension a little — up to 30 kV to be precise.

We use that 30 kV to generate sparks and this alone already looks spectacular in the dark. However, in this case those sparks also play a melody: either 'Jingle Bells' or 'Mission Impossible.' The surprisingly simple schematic is shown in **Figure 2**.

At the heart of the circuit is an Arduino Pro Mini. It generates a PWM signal with a fixed duty cycle of 10% (and is therefore actually a normal square wave). This square wave is converted to 12 V and buffered. Then, a high-voltage transformer turns that into a voltage of around 30 kV — high enough to generate beautiful sparks.

An ordinary square wave doesn't make music on its own — that's why the software in the Arduino (which can be downloaded from [6]) varies the frequency at the tempo of the two pre-programmed melodies. The result can be seen and heard in a video by the author [1].

The construction of the circuit is not critical; only the transformer requires some tinkering. The primary winding consists of 20 turns of 0.7 mm copper lacquer wire around the end of two adjacent ferrite rods that have a diameter of 10 mm and a length of 80 mm. These two rods fit snugly in the opening of a ready-to-use 17 kV coil that serves as the secondary (see **Figure 3** and [2]).

Note: after the online publication of this circuit, it turned out that the coil used by the author is no longer available. An alternative is [3], which is also already supplied with a primary winding, so you do not have to make it yourself. Another option is [4], but then you will still have to wind the primary. Finally, [5] is another option, but in that case the drive voltage of the primary may have to be lowered. And, in all cases, you will have to determine what the optimal duty cycle is for obtaining the most powerful sparks without also saturating the core.

And yes, the circuit also operates as a primitive spark transmitter and as such is possibly illegal, but as long as you don't use it continuously you will probably not invite any complaints. With a small AM receiver you could find out at what distance any noise is still noticeable — you have been warned!

From the project page [6] you can download a larger version of **Figure 2**

idea: Elektor Lab

Lie detector

Real lie detectors use various sensors attached to a person's body to measure parameters such as blood pressure, heart rate, breathing, skin resistance and temperature. The detector records these values on a long paper strip and, from this data, it can be deduced (allegedly) whether the person in question has answered the questions truthfully or not. Incidentally, the use of lie detectors is controversial in many countries

Here is a simple circuit that measures the changes in skin resistance and can be used in a playful way as a lie detector.

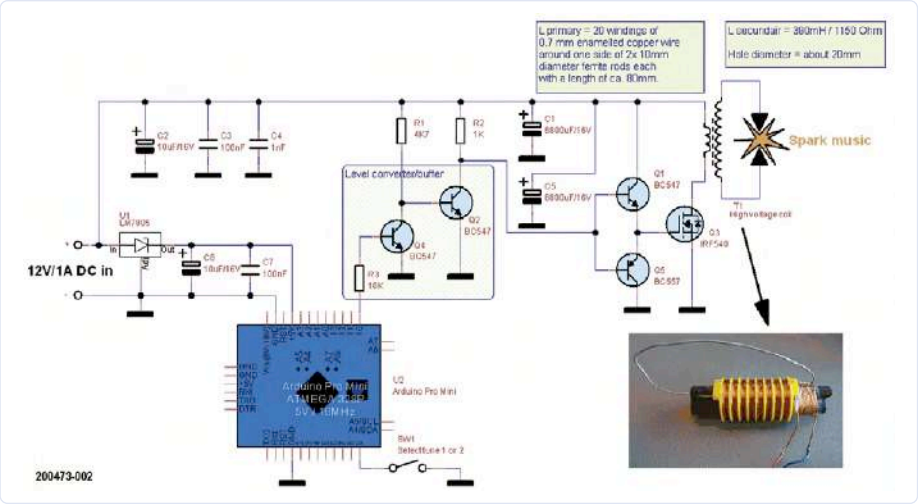


Figure 2: The heart of the musical spark generator is formed by an Arduino Mini Pro.

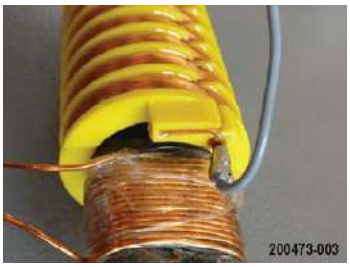


Figure 3: The primary winding is tightly wound around two ferrite rods that fit exactly into a ready-made HV coil.

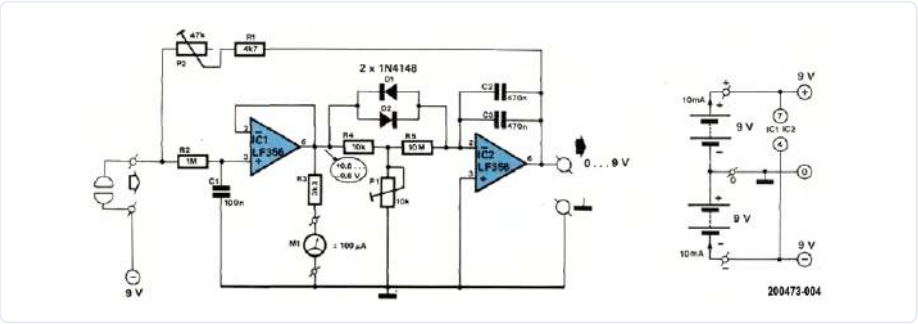


Figure 4: A simple lie detector that indicates changes in skin resistance.

Questions or Comments?
Do you have questions or comments about this article? Then email Elektor at editor@elektor.com.

An ordinary ohmmeter is not useful for our purposes — after all, we don't want to measure the absolute value of the skin resistance, only its change. A circuit that makes this possible is shown in **Figure 4**.

On the left side of the schematic we see two electrodes that are (for example) attached to the back of the hand and the corresponding forearm. The 'lower' electrode has a negative voltage of 9 V. Through the skin (resistance) and R2, this voltage is applied to the extremely high input impedance of the non-inverting input of IC1. This input draws almost no current; the current flowing over or through the skin therefore has to go through P2 and R1 to IC2. This opamp controls its output voltage in such a way that the 'upper' electrode is always at 0 V.

R5 and the parallel circuit of C2 and C3 provide a certain delay. Large voltage differences are quickly eliminated by the pair of diodes; with small voltage differences it can take up to 30 seconds before the voltage of the upper electrode is back to 0 V. In this way we obtain a clear result on the moving-coil meter M1.

R1 and C2 form a low-pass filter to keep out any detected (hum) disturbances. The measuring instrument must be of a type with the zero point at the middle of the scale.

Regarding the adjustment we can be brief: attach both electrodes (how you make them is up to you) to the test subject's skin and connect a multimeter to the circuit's output (range: 10 V_{DC}). Then turn P2 until the meter indicates a voltage of between 2 and 6 V (wait until the reading is stable). With that, the detector is adjusted to the skin type of the subject. When we now lightly press the electrodes, M1 should show a result. If that is okay, the detector is ready for use.

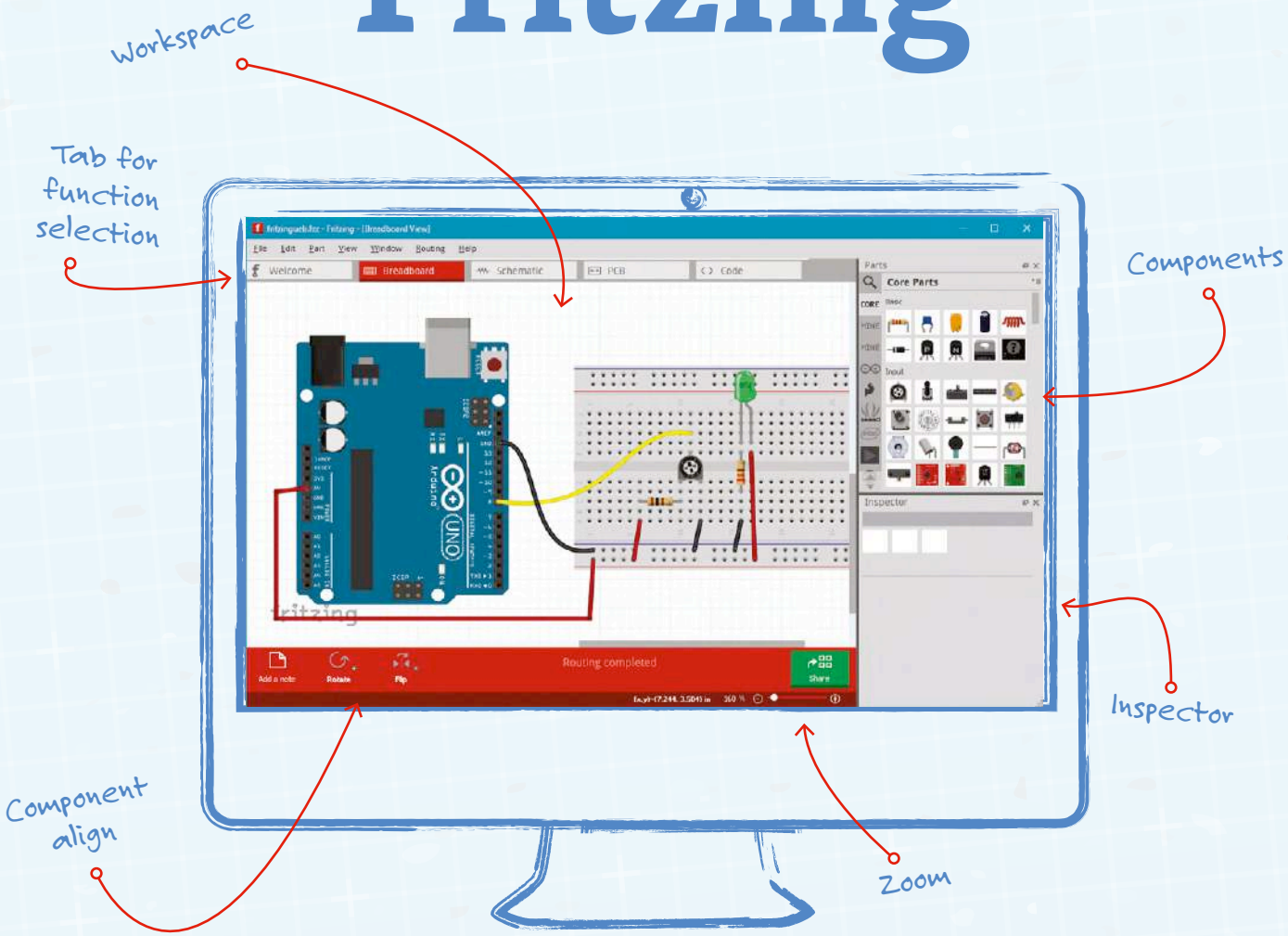
Warning: This circuit may only be powered using two 9 V batteries. Do not use a mains power supply under any circumstances!

200473-02

WEB LINKS

[1] **Video Jingle Bells**: www.youtube.com/watch?v=NGSjcOffbuI&feature=youtu.be
[2] **Original HV coil**: https://bit.ly/3lRr8R7
[3] **Alternative HV coil #1**: https://bit.ly/35crVGo
[4] **Alternative HV coil #2**: https://bit.ly/2FfsCU4
[5] **Alternative HV coil #3**: https://amzn.to/3jPcAiX
[6] **Project page for this article**: www.elektormagazine.com/200473-02

Breadboard Graphics with Fritzing



By **Florian Schäffer** (Germany)

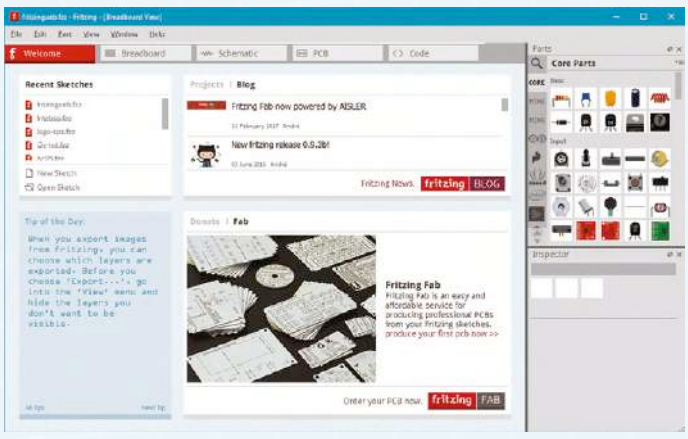
The path from a circuit diagram (see the previous issue) to a first prototype built on a breadboard is not always smooth: the abstract shapes of the schematic must be turned into real components and connections. With Fritzing you can create realistic-looking layouts and thereby simplify subsequent construction.

In principle it is a good idea to take a photograph of a physical circuit's construction. However, getting lighting and composition right takes a measure of skill and the result is often not good enough to use as a reference for copying. With Fritzing, on the other hand, you can create breadboard images that are both easy to understand and attractive. The components are graphically reduced to their essentials, making the required connections clearer. In any case, before working with any physical parts, it is always well worth laying out the circuit on a virtual drawing board to help determine the optimal component placement.

Alternative software

Since its initial release, Fritzing has been very popular as it is free of charge, simple to use, and produces pleasing results. The program also offers additional functions including the production of circuit diagrams, printed circuit boards and source code, but these functions are considerably less useful. In the next issue we will get acquainted with Tinkercad Circuits, which will let us simulate circuits as well as produce beautiful images of assembled designs. The most important difference between the two pieces of software is that, in Fritzing, any user can create new graphical components to add items missing from the selection of built-in objects. This is not possible in Tinkercad Circuits.

Although the program is available for free, since the end of 2019 the download has been hidden behind a request for payment. To get around this (entirely legally) you can search for and install an older version from the Internet (and when you first run it you will be prompted to update it) or you can download the current version, 0.9.4, directly using the link *https://fritzing.org/download/api/1.0/update/344*.
After installing it, launch Fritzing. It will initially search for updates to the program and its component libraries. The language for the user interface is selected automatically based on the system language configuration, and so for most readers this will default to English. You will then be taken to the welcome screen. Here, across the top of the window, you will see a row of tabs corresponding to the various available functions. We will just be using the 'Breadboard' function, so click on that button.

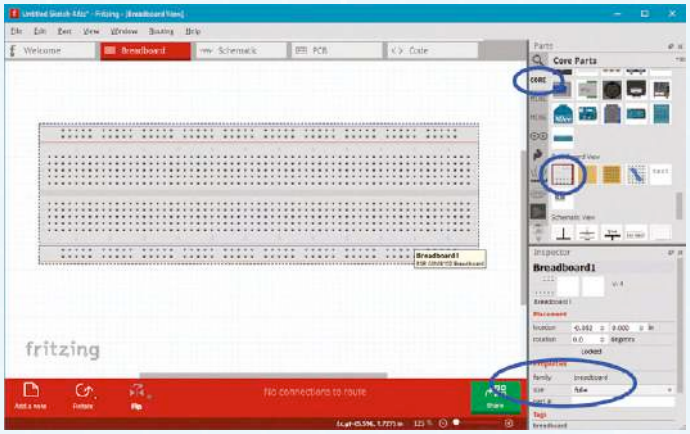


Fritzing's welcome screen. Click on 'Breadboard' to switch to that view.

Export formats

Fritzing can write a variety of file formats. When you use *File/Save* it saves files in its native format with a *.fzz* extension. These files can be loaded back in later to work on them further. On the other hand, using *File/Export* gives the choice of other formats that can be used for various purposes. Among the most useful of these are the image formats allowing for export as a PNG and creating a bitmap image that is ideal for use on websites. Unlike the JPG format, saving images as a PNG is 'lossless.' The SVG vector format is most appropriate when the image is subsequently to be scaled or further modified using a graphics program. However, each graphics editing program tends to interpret SVGs in its own idiosyncratic way, which can lead to different results with different programs. Even whole components can disappear because of errors in Fritzing's component library. For this reason it is actually better to initially export to a PDF file as there are many vector image editing programs capable of handling this file type.

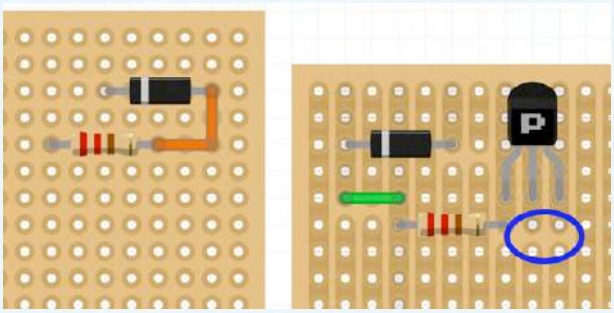
Choose a breadboard



Fritzing always starts up showing a large breadboard, but the size can be adjusted and it can even be deleted entirely. The mouse wheel is used to zoom the view in and out and panning is possible using the middle mouse button.
When a component is selected by clicking on it, its attributes will always be displayed in the 'Inspector' on the right. If you select the default large breadboard you can use the *Size* attribute to choose a different type of breadboard as desired. If you wish to add a (further) breadboard to your design, you will find it in the *Parts* list in the 'Core' bin. New components can be dragged directly into the working area.

Circuit board view

A relatively little-known feature of Fritzing is that it is also ideal for laying out prototyping boards (both for the pad-type and strip-type). You can use this feature to determine optimal part placement before soldering and to work out where strips will need to be cut or wire links will need to be added. Alongside the symbol for a breadboard in the component list there are two brown circuit board symbols. The first of these represents a prototyping board that only has solder pads; the second represents stripboards. You can use the Inspector to modify the type of board, as well as its size, as required.

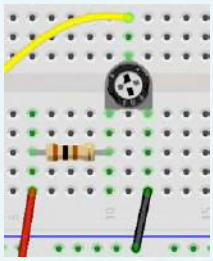


A pad-type prototyping board and a stripboard. In the latter case the conductors can be cut.

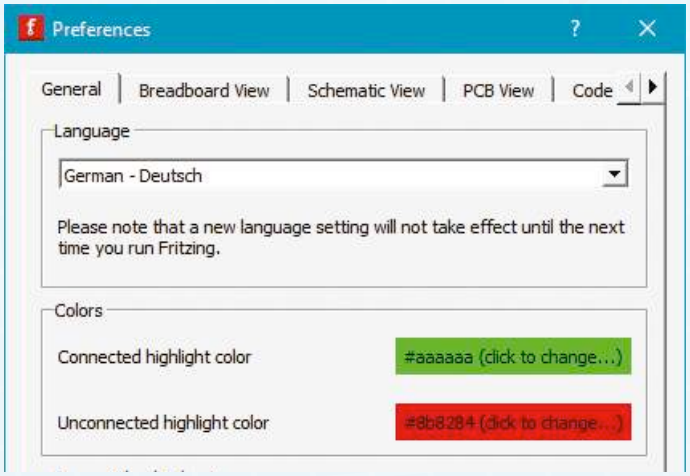
The best feature when using stripboard is that it is possible to 'cut' the conductors (and subsequently join them back up again). To do this it is necessary to click between two holes. You may also need to zoom in a long way in order to hit the spot accurately. In practice, of course, these cuts are made using a manual spot-face cutter or a small drill that is somewhat larger than the width of the conductor.

Practical settings

In order to make it easier to see errors, connections that are made through the spring contacts of the breadboard are highlighted in green. However, this colour is perhaps too lurid for use in documentation.



Connections made via the sprung contacts in the breadboard are shown in green.



Window showing the settings that can be adjusted to customise Fritzing.

Under *Edit/Settings* it is possible to modify the behaviour of the program to some extent. In the *General* tab you can adjust the colours: click on the green area representing the 'Connected highlight colour' and then select a custom colour, such as a medium grey, which is less hard on the eyes.

Using grey as the highlight colour makes the connection indication much less conspicuous.

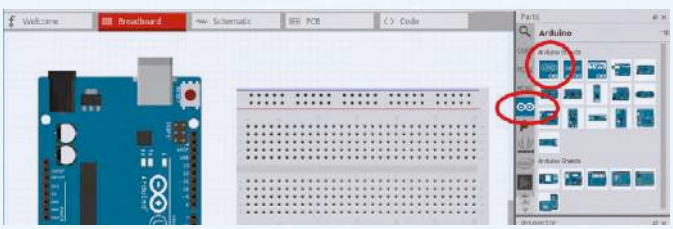
Normally wired connections are shown as straight lines with sharp corners. However, for many connections curved wires are more suitable, especially in cases where the line cannot be laid out with only right-angle turns, or where this would result in a convoluted path. Under the *Breadboard view* tab in the settings screen you can enable the use of curved wires and pins, giving you the freedom to create straight or curved wires at will.

Working example

For a first experiment we can try to reconstruct the image shown at the beginning of this article (the exact function of the circuit is not important for now).

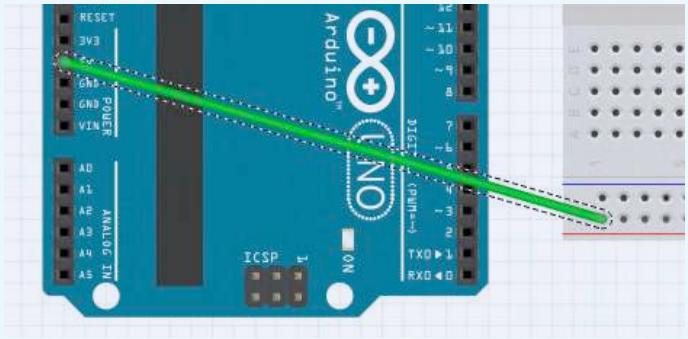
1. Select the breadboard and then, in the Inspector, adjust its type so that it looks like the version in the picture.

2. Under 'Parts' select the collection with the Arduino logo. When all the components are loaded, drag the first symbol ('Uno') to the working area so that an Arduino Uno board now sits next to the breadboard on its left.



3. While the Arduino board is selected you can drag it around and, using the tool on the bottom edge of the window, rotate it. Do this until the USB socket is pointing toward the top of the screen.

4. You can start to draw a connecting wire by clicking on a contact point. This can be the end of the pin of a component, a contact point on the breadboard, or a pin or socket on a circuit board such as the Arduino Uno. Draw a connection from the 5 V point on the Arduino to the red power distribution bus on the breadboard.

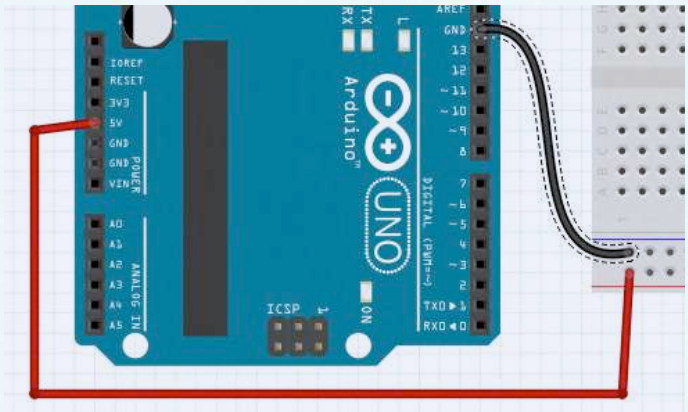


5. You can choose any colour you want for the line. The default colour is blue. If you click on the line you can change its colour in the Inspector.

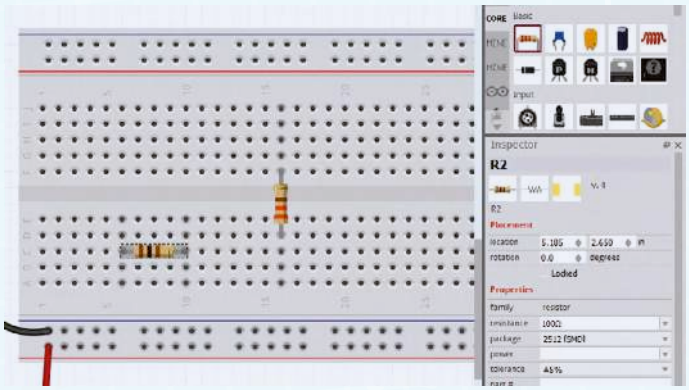
6. If you wish to add a node to the line in order to change the routing of the connection, click twice on any point along the line. A node will be added that you can now drag around. You can add as many nodes as you need to the line. A right-click on a node will bring up a context menu that allows you to modify its various properties.

7. Draw a direct connection from the right-hand GND point on the Arduino to the blue power distribution bus on the breadboard and change its colour to black.

8. If you wish to turn a straight connection into a curved one, grab the line at a point along its length and drag it in any direction. This will create a Bézier curve. You can use a maximum of two such points to affect the path of the connection.



9. Most component types can be placed in the same way. For each type of component there are, in general, different settings in the Inspector, for example affecting its colour, value or size. The most frequently-used components can be found in the 'Core' bin; the other bins offer components such as modules from manufacturers of Arduino accessories. If you cannot immediately find the part you need, you can click on the magnifying glass and enter a search term.



10. Add a resistor to the working area and drag it to the desired position. In the Inspector you can choose between a carbon film resistor with three bands, or a metal film resistor with four bands. You can also choose a value (or enter a custom value that does not appear on the list) and a tolerance. The colours of the bands on the resistor will change to reflect these settings.
Next, add the trim potentiometer and the LED. When placing the LED, make sure that the side with the bent leg (the positive or anode connection) lies to the right. It is possible to approximate the effect of mirroring a component using the rotate button at the bottom edge on the window. You can choose a wavelength for the LED, but the only effect is to change its colour in the graphic.

11. All that remains to be done is to add the final connections on the breadboard and to the Arduino.

This article originally appeared in German in a special edition of Elektor offering an introduction to electronics based around the Arduino platform.

200225-B-02

Analogue Filter Design (Part 2)

Active Filters

By **Alfred Rosenkränzer** (Germany)

In this second part in our series of articles on the design of analogue filters, we will look at active filters. The inductors that are used in passive filters are replaced by active building blocks — typically using operational amplifiers. This has a number of advantages, which will become clear soon.

Electronics engineers are generally not that enthusiastic about inductors. This is not a surprise since the properties of real-world inductors are hardly ideal. Because they are wound from wire, they also have a resistance that cannot be neglected. This resistance depends on the cross-sectional area and the length of the wire that is used. In addition, there are parasitic properties, such as inter-winding capacitance, that can lead to undesirable resonances. Furthermore, low frequencies require large values of inductance that result in physically large inductors, making them expensive and susceptible to external magnetic fields. In certain circumstances they could even be the source of undesirable stray fields themselves. Generally, large inductors are now only used in the passive crossover networks of loudspeakers (**Figure 1**).

Active filters

Inductors only make a very rare appearance in active filters, and then only in their smaller form factors. What makes a filter 'active' is the use of active building blocks; semiconductors in the form of transistors (seldom) or operational amplifiers (almost always). Without inductors, active filters can implement multi-stage filters that are close to an ideal filter, since each of stage has little influence on the others. In the simplest case, a transistor in a collector circuit acting as a buffer could be sufficient. However, not only is the design with op-amps easier, but gains > 1 can

Figure 1: Loudspeaker crossovers are passive filters with big coils. Here is the model HW3/120NG from Visaton. (Image: Visation)

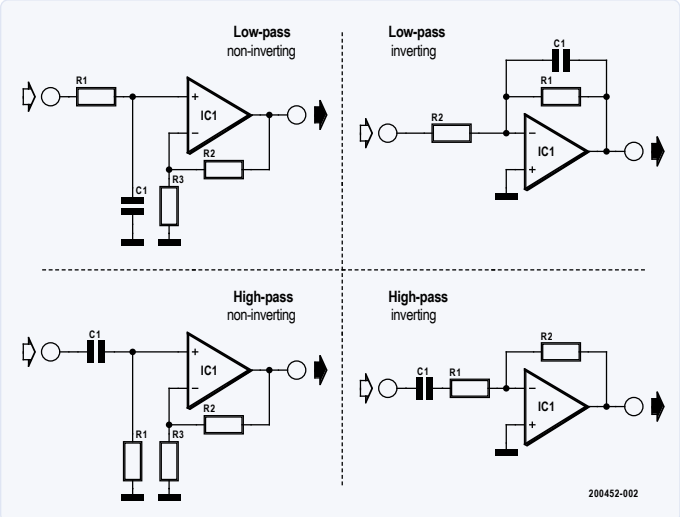


Figure 2: Basic schematics of high- and low-pass first-order filters in inverting and non-inverting variants.

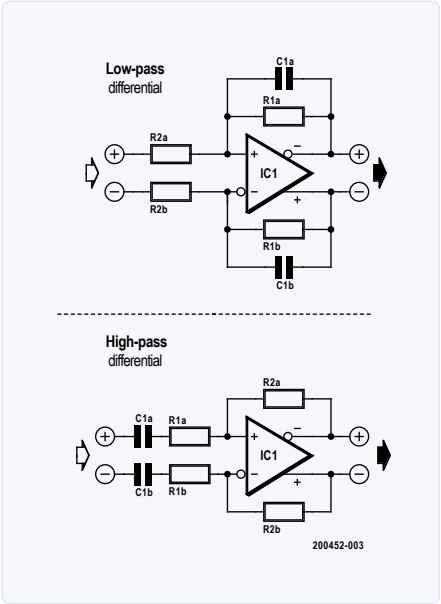


Figure 3: Basic schematics for first-order differential high- and low-pass filters.

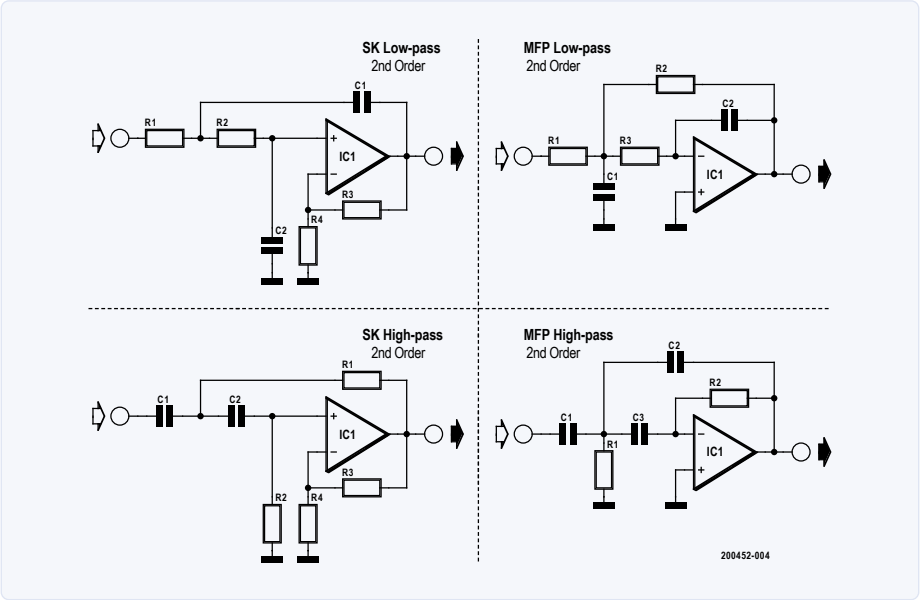


Figure 4: Basic schematics of second-order high- and low-pass Sallen-Key (SK) and Multiple Feedback (MFB) filters.

also be easily realised when the application requires it.

Filter types

Just as with the passive filters, active filters can also be built as low-pass, high-pass, all-pass, band-pass as well as band-stop variants. These filter types were already described in Part 1 [1], so to prevent duplication here you are encouraged to refer back to that article.

Approximations

Just as with the passive filters, active filters can also be designed with Bessel, Butterworth, Chebyshev (and other) characteristics. The realisation of Cauer filters and the inverse-Chebyshev with their notches in the stop band are not impossible in principle, but typically are not implemented due to the high cost and their limited use.

Instead, taking the example of a frequency cross-over in the audio range, one type of filter that is frequently used is the Siegfried Linkwitz, named after its inventor. It enables a straight sum frequency response of the low-, mid- and high-pass branches of a cross-over network. A 4th order Linkwitz filter consists of two 2nd-order Butterworth filters connected in series — this is not the same as a 4th-order Butterworth filter! In a Linkwitz filter the frequency cut-off is also defined at the -6 dB point, not at the more usual -3 dB point. Although a flat amplitude response can be obtained, the group delay is anything but flat. As a result there is significant impulse distortion, especially with square-wave signals.

Structure and order

Figure 2 shows the basic circuit topologies for low-pass and high-pass filters of the first order. The two filters on the left consist of an RC low- or high-pass followed by an op-amp as a buffer in a non-inverting circuit. The two resistors R2 and R3 define the gain in the traditional way. The filter function is not influenced by the gain. In both cases, the input impedance is defined by the combination of R and C. The non-inverting input of the op-amp has a very high input impedance and can therefore be neglected. The output of the op-amp is also the output of the filter and, therefore, has a very low impedance. Any subsequent filter stage can therefore be connected directly to this output without affecting the filter behaviour.

The two filters on the right are based around an inverting op-amp. Here the gain and cut-off frequency cannot be set independently of one another. The cut-off frequency depends on R1 and its gain on the ratio of the two resistors. The filter tool that I used cannot calculate the inverting variant. This would also be redundant since both the low-pass and the high-pass filters behave the same and the cut-off frequency is always given by $fg = 1 / (2 * \pi * R1 * C1)$.

As already mentioned, building a first-order band-pass filter or band-stop filter is not a simple task. A kind of band-pass filter could be realised with a low-pass and a high-pass filter in series (or with these filters in the opposite order). However, since their roll-off slopes are not very steep, and their cut-off frequencies are close together, they are of little use for implementing narrow-band filters. The advantage of this approach, however, is that you can set the lower and upper cut-off frequencies independently of each other. But, be careful: if $fg_{HP} > fg_{LP}$, then there unfortunately is no bandstop filter, but only a broadband attenuator with a 'sloping' frequency response.

Differential filters

Fully differential filters amplifiers are increasingly being used in audio applications and they are also used in high frequency applications of several hundred MHz. You can also use these to build active filters, as **Figure 3** shows. However, this only works for one topology. If you are interested you could use the LME48724 differential op-amp for IC1. There are two different topologies for active second-order filters: the Sallen-Key-Filter (SK), named after its inventors, and an arrangement known as

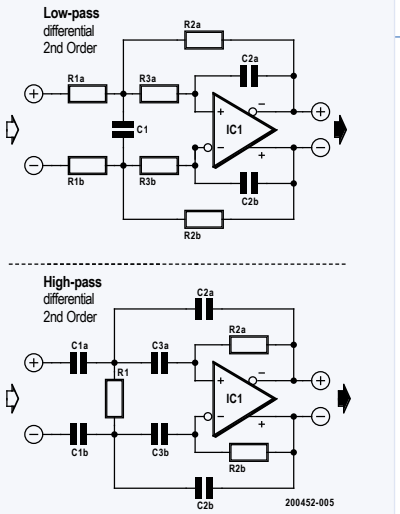


Figure 5: Basic schematics of second-order differential high- and low-pass filters. These are only possible as MFB types.

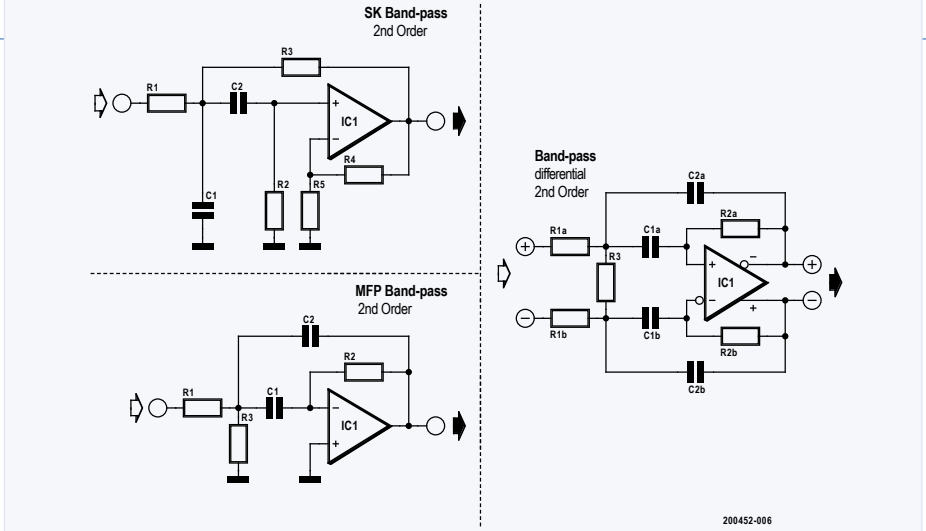


Figure 6: Basic schematics for second-order band-pass filters.

'Multiple Feedback' (MFB). **Figure 4** shows both topologies in their low- and high-pass implementation. In contrast to the SK filter, an MFB filter inverts the signal. The gain in the pass band is determined by the ratio of R3 and R4 for the SK high- and low-pass filters. In the MFB low-pass filter, the gain in the pass-band is set by the ratio of R1 and R2 but, in the MFB high-pass, the gain depends on the ratio of C1 and C2.
Second-order differential filters can only be implemented using the MFB topology (see **Figure 5**). The basic circuit of a band-pass filter can be seen in **Figure 6**. The differential band-pass filter is of the MFB type.

Higher order band-pass filters

Fourth-order band-pass filters can be designed directly as a single circuit or as a combination of low- and high-pass. The latter again has the advantage that you can set both filter edges independently.
You can even combine high- and low-pass filters of different orders in a series arrangement. This is suitable, for example, for the filters in a bat detector: The relatively loud 'low' frequencies can be suppressed with a 4th-order low-pass. A 2nd-order high-pass is then sufficient to block frequencies >80 kHz.

Band-stop filters

For a band-stop filter of the second-order, in addition to the SK and MFB (**Figure 7**), there is also the Bainter topology. The SK implementation requires only a single

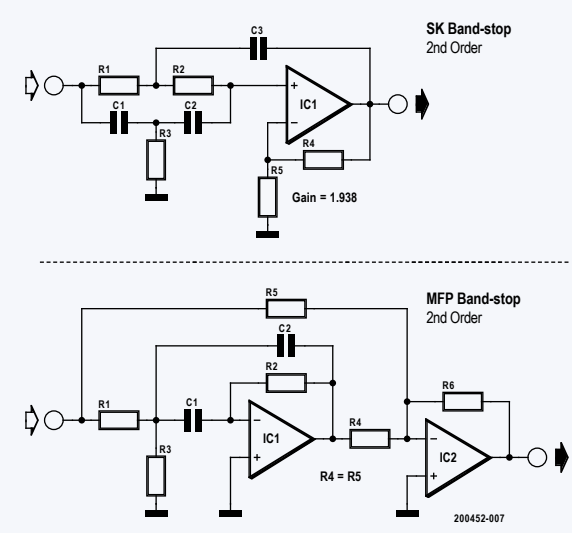


Figure 7: Basic schematics for second-order band-stop filters.

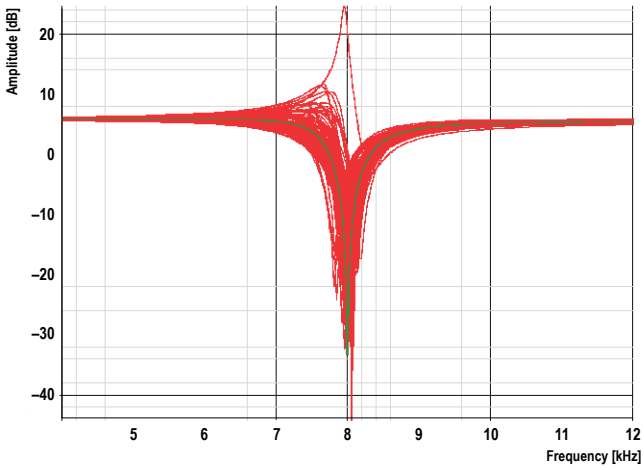


Figure 8: Simulated frequency response of a second-order SK band-stop filter. The green curve is the ideal characteristic. The cluster of red curves were created using the Monte-Carlo simulation of component tolerances.

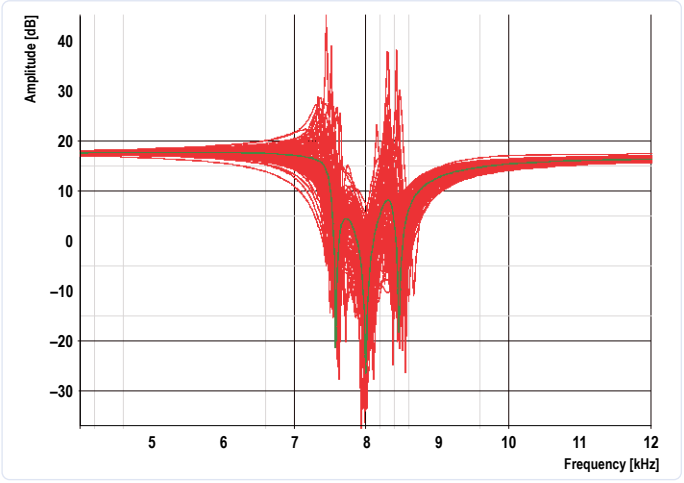


Figure 9: Simulated frequency response of a sixth-order SK band-stop filter. Legend: green = ideal response; red = Monte-Carlo simulation of the component tolerances.

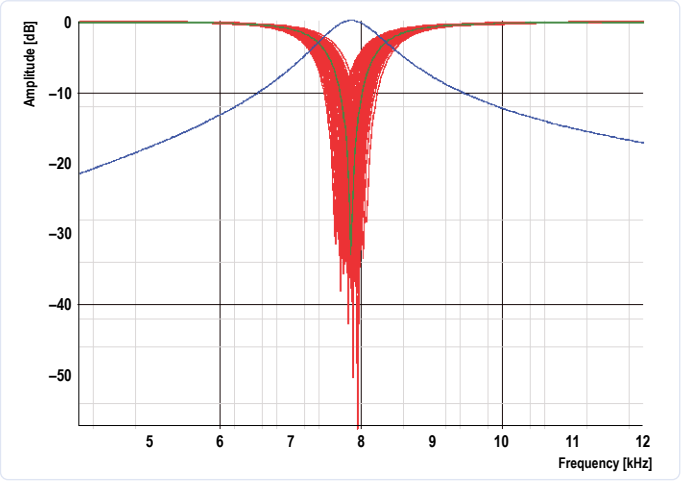


Figure 10: Simulated frequency response of a second-order MFB band-stop filter. Legend: green = ideal response; red = Monte-Carlo simulation of the component tolerances; blue = response of the band-pass (see text).

op-amp and used a simple double-T arrangement consisting of three resistors and three capacitors. Unfortunately, this concept is sensitive to component tolerances. The resistors, and especially the capacitors, should come from one batch. The frequency response can even change dramatically as a consequence of component ageing.

The frequency response of an 8 kHz SK band-stop filter can be seen in **Figure 8**. The ideal, hypothetical frequency response is shown in green. It is particularly interesting to examine the consequences of randomly varying the component values within their tolerance range (1% for the resistors and 5% for the capacitors). As the frequency response of **Figure 9** shows, SK band-pass filters of higher order are especially sensitive. In addition, the response in the stop-band has several valleys. The gain cannot be freely selected and is about a factor of 2 per stage. Overall, it can be said that it is better to select a different type of band-stop filter than use the Sallen-Key implementation.

The MFB band-stop filter is actually a band-pass filter, the output signal of which is subtracted from the input signal. Thus the filter produces, indirectly as it were, a band-stop filter. This variant is less sensitive to variations in component values, as the frequency response of the 2nd-order MFB band-stop filter in **Figure 10** shows. The ideal response of the band-pass section is shown here in blue.

A Bainter filter requires three op-amps. Further information on this variant can be found at [2] and [3]. In the circuit of **Figure 11**, the actual values for a band-stop filter with a frequency of just under 8 kHz are given. The corresponding frequency response can be admired in **Figure 12**. You can clearly see the 'nicer' frequency response, together with the reduced sensitivity to the component tolerances in its stop-band behaviour.

Interestingly, the older desktop version of my filter program only calculates the SK and MFB implementations. Texas Instruments now prefers you to use the newer web version [3], which only supports the Bainter variant.

All-pass filters

The filters discussed so far aim at a frequency-dependent change in amplitude. The frequency dependence of phase response or group

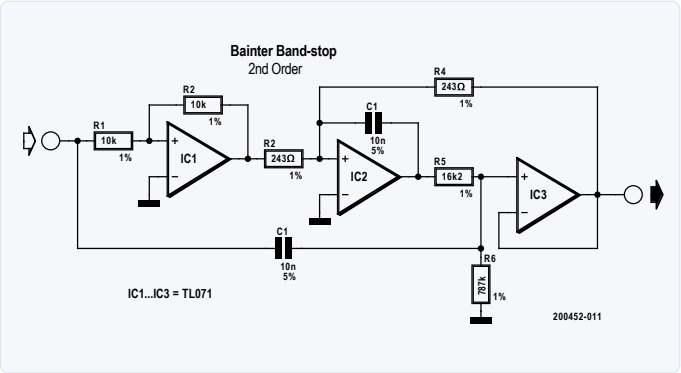


Figure 11: Basic schematic of a second-order Bainter band-stop filter. Three op-amps are required.

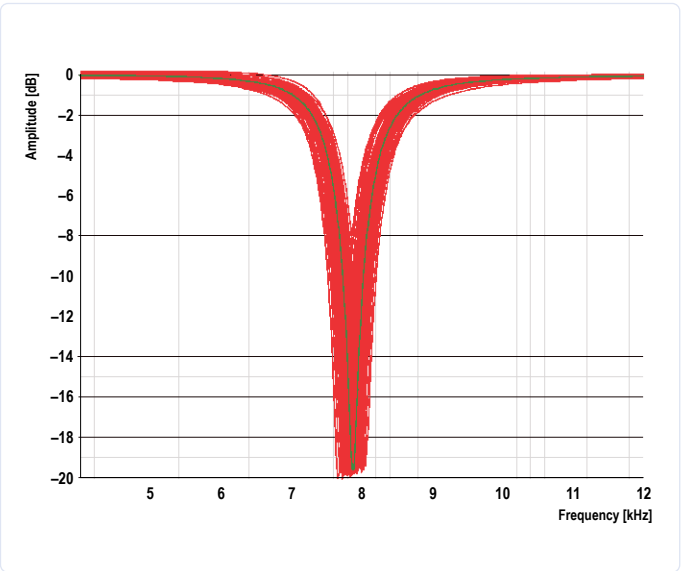


Figure 12: Simulated frequency response of a second-order Bainter band-stop filter. Legend: green = ideal response; red = Monte-Carlo simulation of the component tolerances.

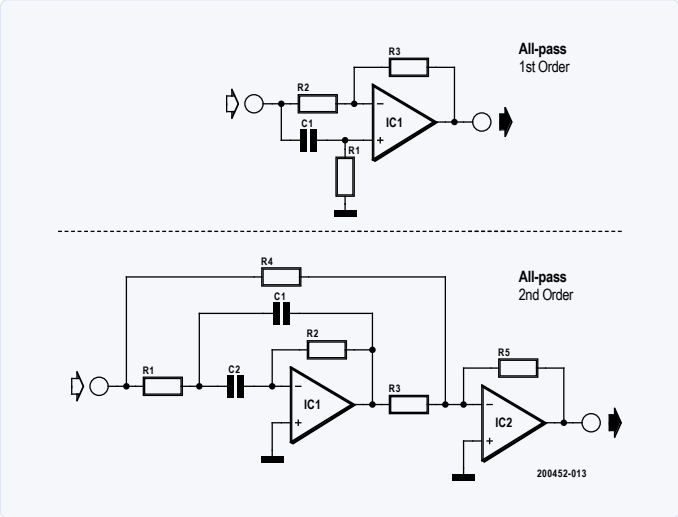


Figure 13: Basic schematics of first- and second-order all-pass filters.

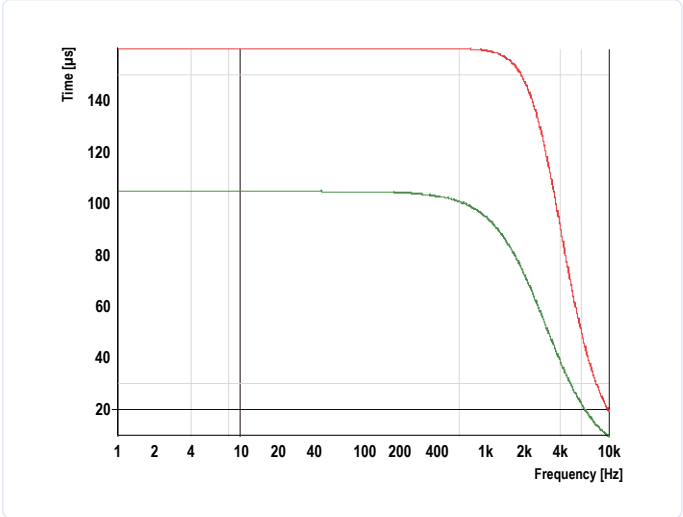


Figure 14: Frequency response of the group delay of the first-order all-pass (green) and second-order (red) of Figure 13.

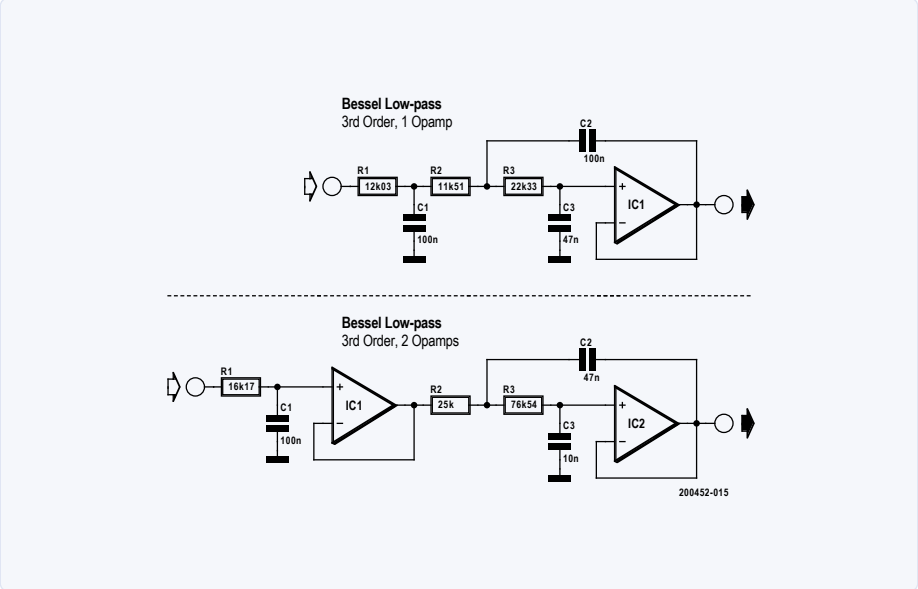


Figure 15: Schematics with component values for Bessel low-pass filters with one and two op-amps.

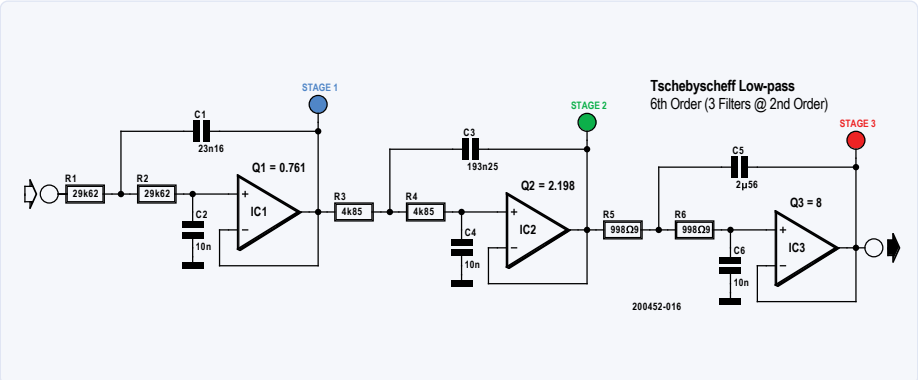


Figure 16: Schematic with component values of a sixth-order Chebyshev low-pass filter.

delay is rather secondary or even ignored (except for the Bessel filter).

With all-pass filters the amplitude does not depend on the frequency, instead remaining constant. The definition of a cut-off frequency related to amplitude makes no sense here. Instead the point at which the group delay drops by -3 dB (to 70.7 %) defines its characteristic.

Figure 13 shows the basic circuit of an all-pass filter of 1st and 2nd orders. **Figure 14** shows the corresponding group delays versus frequency.

Higher order filters

As a rule, higher order filters can also be implemented by connecting two or more first- and second-order filters in series. However, a 4th-order Chebyshev filter is not constructed from two identical Chebyshev filters of 2nd order. This is because the dimensioning of the components are different, as is the behaviour of the filter .

An op-amp is usually required for each first- or second-order filter stage — at least this is what the common tools for filter calculation suggest. Nevertheless, it is possible to build third- and even fourth-order filters using just one op-amp (with some restrictions). The latest web version of the TI filter tool does not offer this option any more, since op-amps are inexpensive and the increased effort is rewarded with a lower-performing filter. **Figure 15** shows a 3rd-order low-pass with one and two op-amps for comparison. The

values of the components were calculated by the filter tool for a cut-off frequency of 133 Hz.
There is nothing wrong with not designing a higher-order filter homogeneously and instead composing it from different filters, such as an SK low-pass and a second MFB low-pass. The individual filters can also be implemented as inverting or non-inverting as required, which is the case with SK and MFB filters anyway. The possibilities are only limited only by your imagination. In the case of the more exotic arrangements, there is a lot to be said for simulating the circuit [4] before building it. You will then know whether the desired filter properties have been achieved.
Another aspect that is relevant for higher-order filters is the filter quality factor Q. You can have the filter tool output this value separately for each stage. You can, in principle, freely choose the order of the filter stages since the outputs are all low impedance. However, if the stages with high Q come first, then it is more likely that clipping will occur than if the stages are reversed. The tool used here [5] conveniently sorts the individual stages by increasing Q.
Figure 16 shows an example of a multi-stage filter in the form of a 6th-order Chebyshev low-pass filter with a cut-off frequency of 1 kHz. **Figure 17** shows the frequency response after each of the three stages in different colours. If you simulate the three stages of the filter individually, you get the hypothetical individual frequency responses of **Figure 18**. If you swap the first and third stages in the circuit of **Figure 16**, you obtain the frequency responses of **Figure 19**. Note that the colours remain the same for each stage. The now-first stage remains red and the overall signal at the end is blue. The red curve in **Figure 17** and the blue curve in **Figure 19** are, therefore, almost identical. You can see in **Figure 19** that, at high signal levels around the cut-off frequency, clipping can easily occur in the first (red) and second (green) stages. Therefore, the arrangement of **Figure 16** is preferred over its reverse.

Gain

Active filters can have a gain > 1 in addition to their actual filter function. In some filters the gain has an affect on the values of the frequency-determining components. Occasionally a component not only determines the gain but it also affects the cut-off frequency. So, for example, you cannot simply change the value of R3 or R4 in the SK filter of **Figure 4** (with an order > 1) without also influencing the filter characteristics. With increasing gain, the 'spread' of the values (the difference between the smallest and largest value) of the resistors and capacitors generally increases too. This fact should be taken into account when designing the filter, as this spread in turn influences the required component tolerances. Fortunately, most filter calculation tools will split the desired total gain of a higher-order filter among the available op-amps. Running a simulation that randomly varies the component values within their tolerance range (keyword: Monte Carlo) you can investigate how sensitive a filter is to component variations.

Selecting component values

The same frequency behaviour can be obtained with different component values. When you increase the resistance values, the capacitor values used can be reduced, and vice versa. Larger resistance values increase noise, while values that are too small could cause distortion because the op-amps have to deliver higher currents. For audio filters, values in the range of 2 to 4 kΩ are a good compromise.

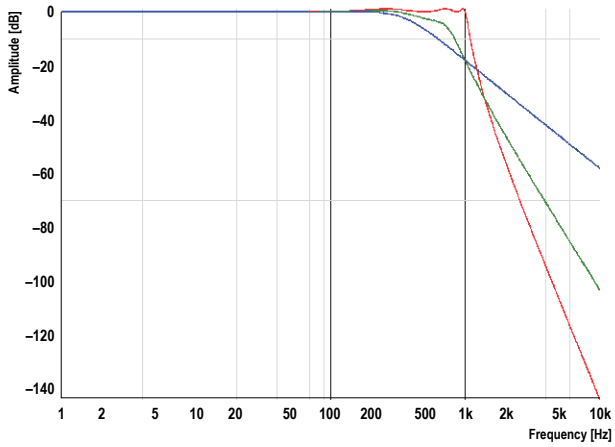


Figure 17: Frequency response of the three stages of the sixth-order Chebyshev low-pass filter in Figure 16. Legend: blue = stage 1; green = stage 2; red = stage 3 (output).

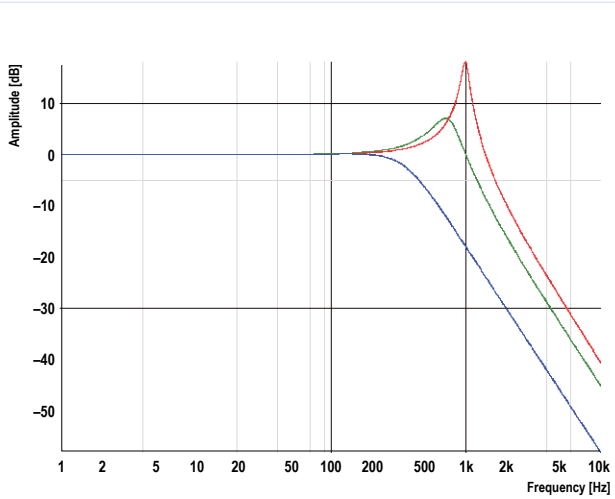


Figure 18: Simulated frequency response of the three individual filters in the sixth-order Chebyshev low-pass filter in Figure 16. Legend: blue = stage 1; green = stage 2; red = stage 3.

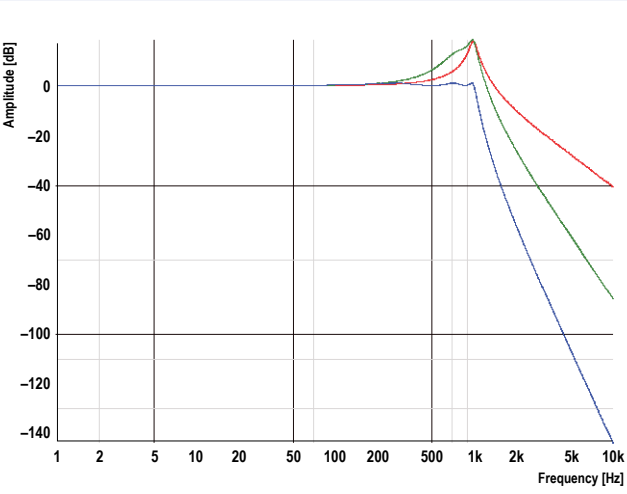


Figure 19: Frequency response of the three stages of the sixth-order Chebyshev low-pass filter with the first and third stages reversed compared to Figure 16. Legend: red = stage 1; green = stage 2; blue = stage 3 (output).

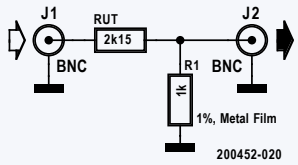


Figure 20: Test circuit for measuring the distortion of a resistor (RUT) using an audio spectrum analyser.

Questions or Comments?
You are welcome to contact the author of this article at the email address alfred_rosenkraenzer@gmx.de.

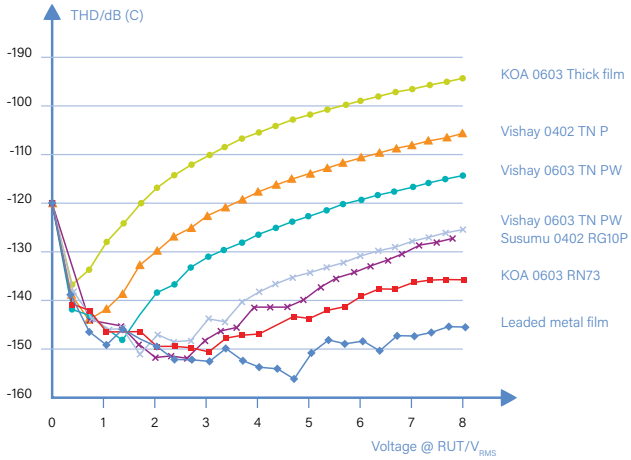


Figure 21: The distortion of different types of resistors at 1 kHz depends on the signal amplitude.

A good filter tool allows the component values to be changed. It is particularly important that you are not restricted to ideal component values (see **Figures 15** and **16**) that are rarely available, especially for capacitors. It is nice if you can specify the available E-series resistors and capacitors separately (as in **Figure 11**). However, sometimes this doesn't work and then you can try to get as close to the 'awkward' value as possible by connecting standard values in series or parallel. The TI filter tool also shows the deviation of the real filter curve from the ideal one. You can try to minimise the deviation by making changes.
A Monte-Carlo simulation takes the selected component tolerances into account and uses a set of curves to show vividly how the filter behaviour can change as a consequence of random variations. If there are large deviations in behaviour, it is recommended to design a less sensitive filter.

Component selection

Once you have calculated the values of the filter components, you still have to consider what type of components you should use and this has an impact on the filter quality and its cost. In the example of the bat detector, there is probably no need for a particularly low-distortion filter. But, on the other hand, if you would like to use a filter to generate a sine wave signal for audio measurement purposes, then you should probably invest a little more.

Resistors:
When selecting the components, the following is true: not all resistors are the same. A good metal film design produces measurably less distortion and noise than a 0805 surface-mount chip variant. **Figure 20** shows a test circuit that can be used to measure the characteristics of a certain resistor (RUT = Resistor Under Test) by connecting it to a high-quality audio spectrum analyser. Obviously, R1 should be a metal-film resistor with known, very good specifications.

Figure 21 shows the distortion of various types of resistors, depending on the signal amplitude. As you can see, SMD thick-film resistors are certainly a poor choice in terms of both distortion and noise. Larger variants are preferable for thin-film resistors. Leaded metal-film resistors show the best results. Guess what kind of resistors are used in the Audio Precision analyser I use.

Capacitors:
Similar considerations apply to capacitors. SMD capacitors smaller than 1 nF are mainly use NP0 dielectric and are therefore very good. With larger values, dielectric materials such as X7R and X5R are common. Unfortunately these are not suitable for low-distortion filters. Capacitors in the range from 10 nF to 1 µF are made from C0G dielectric but here the larger capacitance values become very

WEB LINKS

[1] **"Analogue Electronics Design (Part 1): Analogue Filter Theory", Elektor September/October 2020:** www.elektormagazine.com/200318-04
[2] **U. Tietze, et al, Electronic Circuits: Handbook for Design and Application:** www.springer.com/gp/book/9783540786559
[3] **B. Baker, "Bandstop Filters and the Bainter Topology," Analog Applications Journal, 2015:** www.ti.com/lit/an/slyt613/slyt613.pdf
[4] **"Versatile Circuit Simulation," SIMetrix Technologies:** http://www.simetrix.co.uk/
[5] **Filter Design Tool (web version), Texas Instruments:** https://webench.ti.com/filter-design-tool/

expensive. With capacitors too, the leaded types, such as MKT or Styroflex, are still the best choice. You should also keep an eye on the voltage rating, especially with the SMD versions and the larger value devices.

Op-amps:
Selecting the correct op-amp is not as easy as you might think either. First of all, you need to pay attention to the supply voltage range. In today's modern times there is a trend towards lower voltages. Instead of the long-established ±15 V, you will increasingly encounter ±5 V or even less. Lower voltages limit the available signal level and, not only that, they also make the signal-to-noise ratio worse. At higher signal levels the risk of distortion or even clipping increases. It is not for nothing that a power supply of ±15 V is still obligatory in professional audio equipment.

In general, you should obviously choose low noise and low distortion ICs. A further aspect is the gain-bandwidth product of the op-amp that is required by the filter design. If you use an op-amp that is too slow, you may not obtain the anticipated filter behaviour. You should therefore simulate the calculated filter design before you build a prototype, and be sure to measure the characteristics of the prototype hardware before you start series production.

200452-02

RELATED PRODUCTS

- **OWON SDS1102 2-Channel Digital Oscilloscope (100 MHz)**
  www.elektor.com/owon-sds1102-2-ch-digital-oscilloscope-100-mhz
- **Siglent SDG2042X Arbitrary Waveform Generator (40 MHz)**
  www.elektor.com/siglent-sdg2042x-arbitrary-waveform-generator-40-mhz
- **OWON XSA1015-TG Spectrum Analyser (9 kHz – 1.5 GHz)**
  www.elektor.com/owon-xsa1015-tg-spectrum-analyser-9-khz-1-5-ghz

Contributors
Idea, circuits and text: **Alfred Rosenkränzer (Germany)**
Schematics: **Patrick Wielders**
Translation: **Arthur de Beun**
Editing: **Stuart Cording**
Layout: **Giel Dols**

The Lap Counter

A Project for Mini Petrol-Heads

You may have reason to doubt it, but electronics engineers do have families, along with all the obligations that come with family life…

Contributors
Idea, schematic and photos: **Rainer Schuster**
Text: **Eric Bogers**
Translation: **Arthur de Beun**
Editing: **Stuart Cording**
Layout: **Giel Dols**

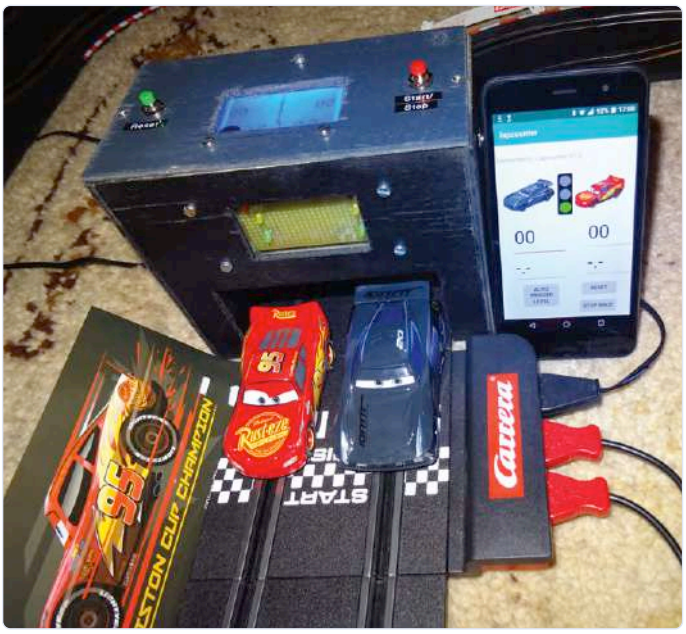


Figure 1: The lap counter fits exactly over a two-track race circuit.

And this is, therefore, also true for Rainer Schuster whose lab we have already visited in an earlier article in this series (in this context, also refer to the box **Labs wanted!**). For Christmas last year he gave his grandchildren a model race track but, as it is with children, they were soon complaining that it didn't have a lap counter.
Of course, no self-respecting electronics engineer will then immediately rush to the shops to buy such a counter ready-made – only DIY is worthy of consideration. And then, naturally, with all the requisite features:

- Display of lap count and lap time on a graphic display.
- Signalling (red/yellow/green) for starting and stopping a race.
- Audible signal (buzzer).
- Display of the progress of a race on a smartphone (Bluetooth).

The lap counter is designed for a Carrera Go model race track but is easily adapted for other brands. The number of laps (separately for two race cars) is measured using an infrared reflection light sensor; the actual lap time for each car is then calculated. The

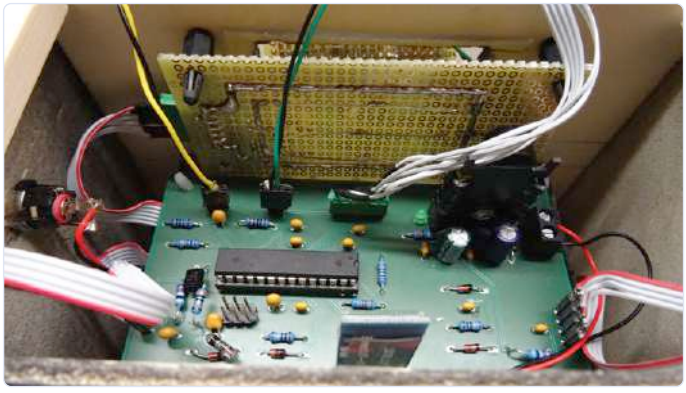


Figure 2: The electronics is divided across two 'real' circuit boards and a piece of prototyping board.

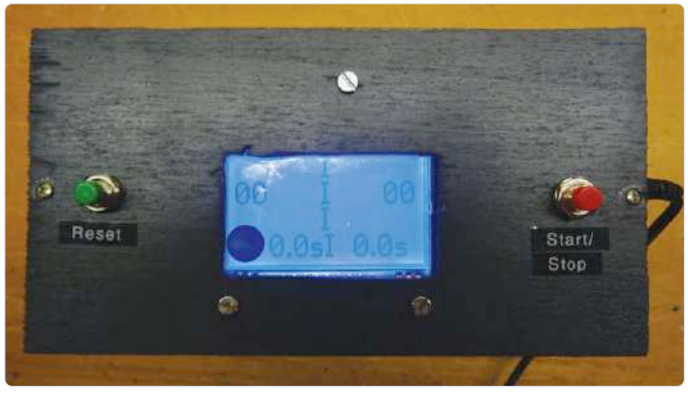


Figure 3: The display on the two push buttons are accommodated in the lid of the enclosure.

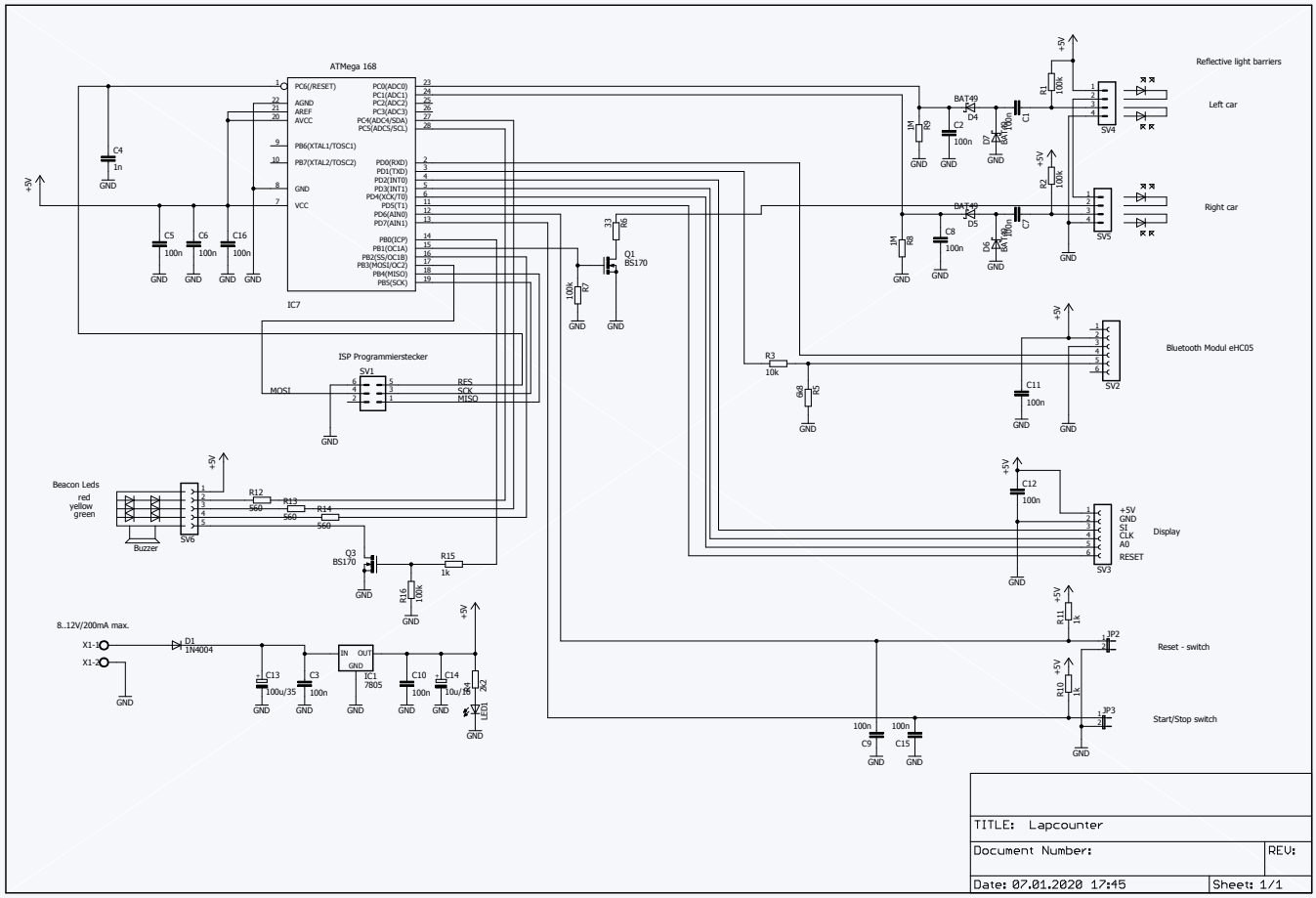


Figure 4: The electronics turn out to be remarkably simple.

LABS WANTED!

Are you the fortunate owner of a large/small/tidy/messy/interesting hobby lab? And do you develop large/small/remarkable/strange/world-shattering projects there? Email us a short description and a few photos (editor@elektor.com) and, with a bit of luck, your lab will become the subject of a future episode in this article series at www.elektormagazine.com/workspace-submission.

results are displayed on a graphic display with a resolution of 128 x 64 pixels. This same information can also be displayed on an (Android) smartphone. Everything is powered from a mains power adapter (8 - 12 VDC).

Figure 1 shows the lap counter in all its glory. The enclosure was designed more with a nod to durability rather than making a 'polished' impression. The entire assembly fits neatly over a two-track race circuit with the IR light sensors positioned on the inside of the 'legs'.

The hardware is accommodated on two printed circuit boards: one circuit board for the controller (in this case an ATmega168) and one circuit board for the display (the latter is mounted in the lid together with the reset and start/stop buttons). **Figure 2** shows this although the display board is not visible in the photo. The LEDs for the signalling are mounted on a piece of prototyping board. The display can be seen in **Figure 3**.

The IR light sensors that have been used here are of the type ITR9904. In order to minimise the effect of ambient light, the IR LEDs are driven with 2-ms-long pulses that are generated by the PWM output of the microcontroller.

The schematic of **Figure 4** should demonstrate that not a lot of 'tangible' electronics is required to built a well-functioning lap counter. Most of the 'intelligence' is hidden in the firmware that is loaded into the microcontroller. The author has prepared a video of the lap counter that is available at [1].

Are you interested in this project? Why not let us know (send an email to editor@elektor.com). If there is sufficient interest we will revisit this project in more detail next year. Until then, you can ask the author any questions using this same email address.

200045-04

WEB LINK

[1] **Video of the lap counter**: https://bit.ly/2ZxPwh0

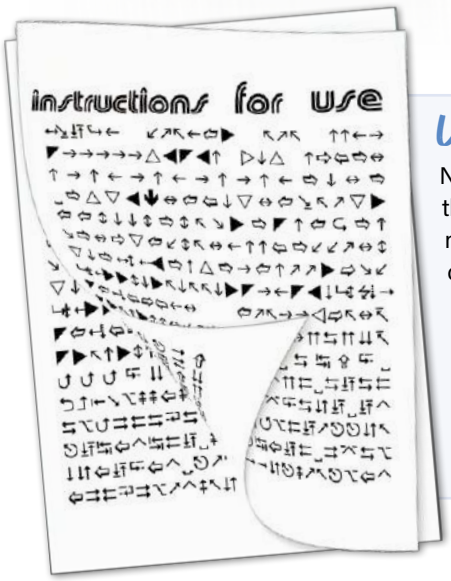
Developer's Zone

Tips & Tricks, Best Practices and Other Useful Information

By **Clemens Valens** (Elektor Labs)

FROM IDEA TO PRODUCT – PART 7 (END)

Previously in this series, we discussed the process of bringing a new product to market. It started by preparing a business plan and then designing the product and we went all the way to producing and testing it. So, now it is finally ready to be sold, right?



USER AND SERVICE MANUALS

No product is finished without a manual. Of course, to you everything is clear and logical, but the user may not think the same way as you do and so you have some explaning to do. A user manual must be written, and this is not a trivial task. A good manual is concise, clear, and complete. Are you up to the job or would it be a good idea to hire a technical writer for this?

Once you have a manual, make sure that people can find it easily. You may want to include it with the product, make it available for download or both.

Certain products also require a service manual. Such a manual is intended for people in the field who service your products. It explains for instance how to replace parts and how things should function or what kind of signal is supposed to be available on which test point. Again, producing such a manual is a big job that you may want to outsource.

Dilemma: put your product in a plain cardboard box or make an effort to please the user?

PACKAGING

As any marketing person can tell you, packaging can make or break a product. Packaging groups together the product, a manual, and often some accessories, and it helps to identify what is inside it. Good packaging allows for safe storage and transportation of the product. Great packaging makes me want to buy the product. Like a product, packaging needs a design effort. The dimensions, shape and material have to be defined and the graphics and text to be printed on it must be specified. Will stuffing material be needed? Custom packaging or a plain brown cardboard box? If custom, then who is going to make it? What will it cost? How will that affect the retail price of the product? And, BTW, who is going to stuff the products into those boxes?

USAGE EXAMPLES

When you produce audio amplifiers or cars, you don't have to tell your clients what to use them for. However, not all products are like this and some require usage examples. The fact that the user bought your product doesn't necessarily mean that he or she understands how to exploit all the neat features you built into it. A line in the user manual saying "*Thingy button: apply a Thingy*" isn't very helpful if you do not explain what a 'Thingy' is and why you would want to apply it (and to what). Provide not just one, but several unique examples that go beyond the most basic use cases.

SHIPPING, DISTRIBUTION AND MARKETING

Another thing to think about in an early stage is how you are going to distribute your product. Will you sell it yourself directly or do you prefer to leave that up to people specialized in sales like distributors and shops? If you take care of it yourself, you won't have to pay commissions, but you may have to hire someone to go to the post office every day. Or maybe make a deal with a courier service? Charge shipping per destination or one size fits all?

If you opt for a retailer, how do you know that he or she tries hard enough to sell it? Who is going to make the marketing effort? How much are you willing to increase the retail price by to pay for this? Or will you bite the bullet and lower your margin instead?

INVOICING & ACCOUNTING

It sounds surprising, but many would-be companies die an early death because they don't take the time to keep track of financial issues like following up on unpaid invoices or requests for quotes or even bookkeeping in general. Be it due to the desire to produce the best client experience possible or just because the demand is overwhelming, whatever the reason is, if the financial side of things is ignored for too long, the business will live a short life.

The unpaid invoices may be from your suppliers (who cares, they have money), but, like you who didn't pay his suppliers (yet), you too are now a supplier who must deal with slow- or non-paying clients. Chasing debtors is an annoying task, but if you don't send invoices, you won't have any debtors to chase and this is even worse.

Producing quotes costs time, but good quotes will bring in new customers.

Everybody knows how important it is to keep track of expenses and income, to keep things balanced, but it takes a bit of time to do so. Many people tend to procrastinate on this because there are so much other important things to do to get the business going. In a sense, they are right: if you procrastinate for too long, the problems disappear… together with your dreams.

SUPPORT AND AFTERSALES

The quality of your aftersales service directly affects the reputation of your company and reflects to sales (unless you're Apple). Aftersales can quickly become time-consuming, so who is going to communicate with your clients and how? These days many companies (try to) keep their phone number(s) and email addresses secret. The idea is to redirect all service requests to a web page with either a contact form or a frequently asked questions (FAQ) list (don't forget the answers) or a chat bot or a forum or what not. But even if you set up the best self-help service centre ever, there will always be a few clients who want to talk to a real person made of flesh and blood. And in the very special case that there aren't, a real person is still needed to read the emails that come in at info@acme-products.com, process the contact forms, handle the reports produced by the chat bot, and open the mail the postman delivered this morning.

RETURNS, REPAIRS AND WARRANTIES

Even though you have done your utmost best to ensure that all your products were built to the highest quality standards and function exactly as intended, eventually a percentage of them will fail. Hopefully, their number will be small, but a few will make it back to your offices with a – probably vague or unclear – complaint from its user. How do you handle product returns? Do you repair? Or is it cheaper to replace? And who will be doing this? If you replace, what do you do with the failing items? What is warranted and what isn't? How long is the warranty period anyway? Some countries specify minimum warranty periods for certain product types, so you may want to check that out.

WASTE DISPOSAL

In several countries manufacturers are responsible for the disposal of products that have been discarded by their users. This means that users can (and will) send products back to the manufacturer for destruction or recycling. The more products you sell, the more waste you will receive, and the pile of e-waste can grow quickly. Maybe some of these items can be refurbished? Maybe some sensitive parts have to be disposed of with special care? As with everything, no matter how you handle waste, there will be costs involved.

THINK ABOUT THE FUTURE

Of course, your product is great and it solves all the issues it was designed for, but somehow, after a varying amount of time, customers will start to ask for other features. It might even be that you come up yourself with a great idea for a product evolution or to expand the product line. Unfortunately, you are now all tied up in production, testing, shipping, packaging, marketing, accounting, and sales and after sales and so you haven't got any time left to spend on engineering and product design. Once again, procrastination will make this problem disappear, but your business will too.

OFFICE AND EMPLOYEES

When you work alone, you probably don't mind working from your garage or in the kitchen, but more space will be needed when you start hiring people. There is, of course, the home office, made popular by you know what, but it is not suitable when communication is intense or when soldering fumes block too much daylight. Hiring some office space can help here. For this you can choose something cheap at the far end of an industrial zone, but joining an incubator may also be an interesting option. They are usually better situated and can provide useful services. They can also help with things like financial matters, but make sure to carefully read the small print before you sign anything.

Besides having to be paid, employees must also be managed, which eats up time. It can be tempting to hire a friend, but a friend may be harder to manage. If you want to keep your friends, it is better to place an advertisement on a social media platform.

Are all your employees going to fit inside your kitchen?

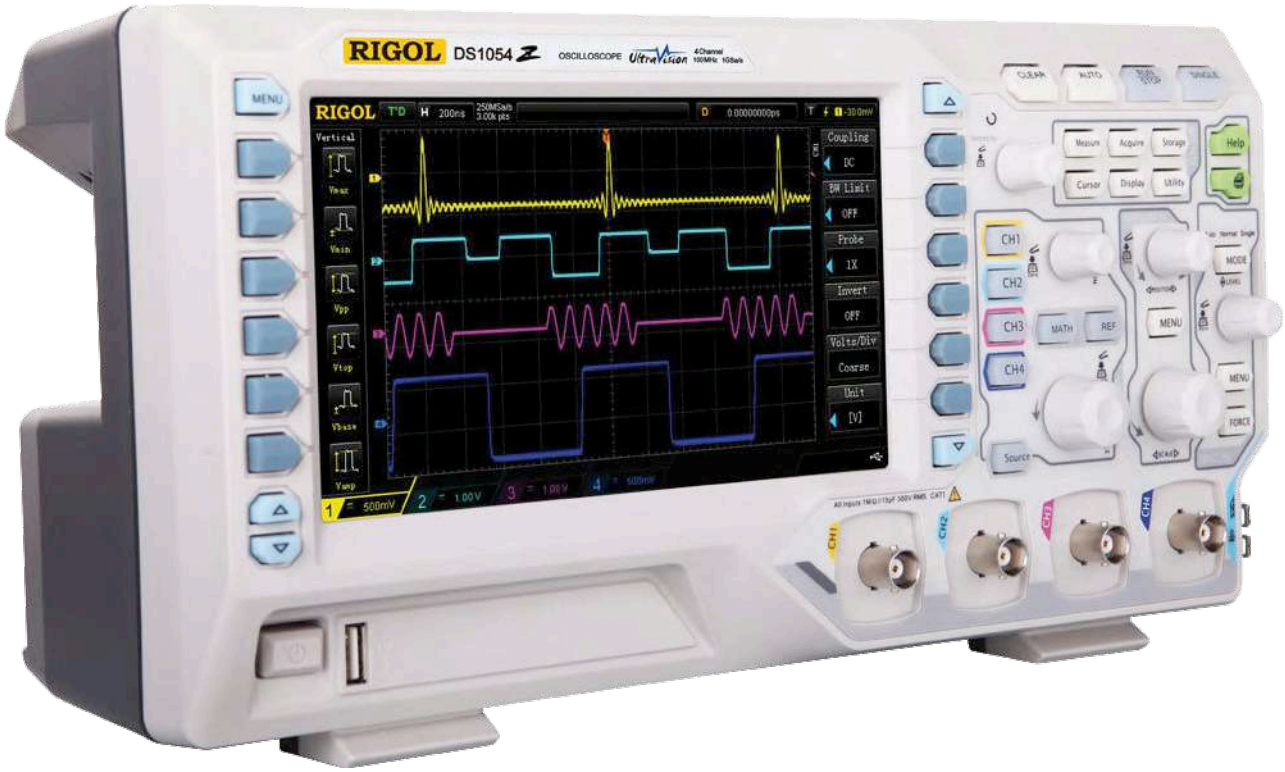
THE END

That's it for this series. There is much more to say, and I am sure I have forgotten one or two things. But you got the general idea, didn't you? If you have followed the guidelines from this article and the previous ones and added your own common sense to it, you may now be in business. Let's hope your product will be successful. Of course, you don't have to have everything in place on day one. Several things will evolve organically, but it is always useful to be prepared for what may follow. Good luck!

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

190369-E-01

Rigol DS1054Z
Four-Channel Oscilloscope



By **Harry Baggen** (Elektor)

If you are looking for a reliable, yet affordable, four-channel oscilloscope, consider the popular Rigol DS1054Z. For less than €400 this device offers a large display, several measurement functions, and many other capabilities. These alone are more than enough reasons to try out the DS1054Z in the lab.

The Rigol DS1054Z has been on the market for several years, but it remains one of the most popular four-channel oscilloscopes in the €400 price range. The model's most important specifications are, of course, the number of channels (4!), the sampling rate (1 Gsps for one channel), and the input bandwidth (50 MHz). Compared to competing measurement equipment, these are still very good specs for the price. As an extra you get four software options for free — namely a memory extension to 24 Mpts, an advanced trigger option, protocol decoders for RS-232, I²C and SPI, and a recording module. All in all, this makes the device very attractive.

Rigol DS1054Z Hardware

The DS1054Z sits in a modest housing of 31x16x12 cm and weighs just over 3 kg. Most of the front is occupied by a large 7" display with a resolution of 800x480 pixels, which is still pretty much the standard in this price range. On the right-hand side the controls are grouped together. Because of the limited front space, there is only one set of buttons for the vertical adjustment of the channels. With the help of push buttons you can select the channel that is controlled. On the right-hand side we find the field for the horizontal settings, the triggering, and a number of menu keys that determine which functions appear on the right-

Figure 1: The buttons for the measuring functions.

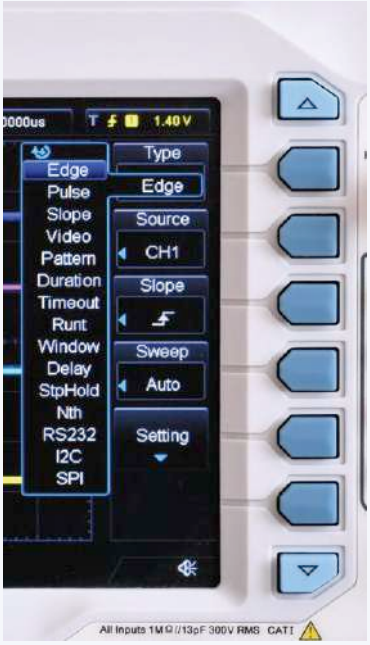


Figure 2: The channel configuration menu buttons.

hand side of the display: Measure, Acquire, Storage, Cursor, Display and Utility. There is also a multifunctional rotary knob for selection in various menus.

To the left and right of the display are a large number of function buttons. Those on the left (**Figure 1**) are for the measurement menu, where there is a set of measurement options for the horizontal range and a set for the vertical range. The selected measurement values appear at the bottom of the display. The set of buttons on the right (**Figure 2**) serve the function menus.

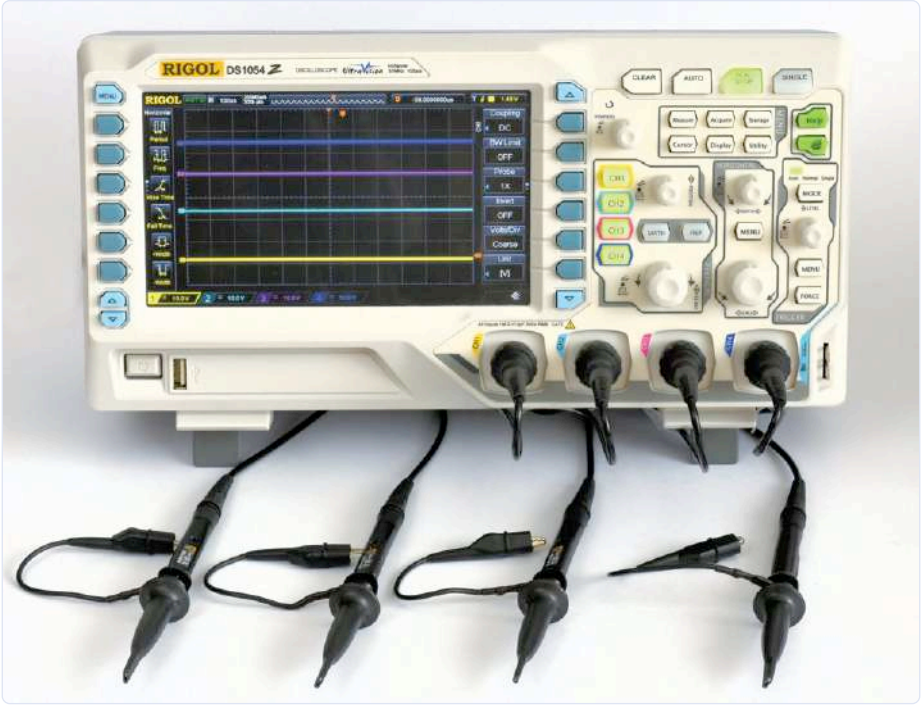


Figure 3: 4 Channels for less than €100 per channel.

The sturdy housing has two folding legs on the front side. At the rear, in addition to the mains connection, there are USB and LAN connectors together with a trigger output for serial measurements. A fan in the Rigol DS1054Z provides the necessary cooling inside the unit. This is audible, but I didn't find the sound disturbing.

Getting started

Operating the oscilloscope takes some getting used to because of the many buttons placed around the screen. But, otherwise, the design of the controls is similar to other devices in this price range (**Figure 3**). This also applies to the selection of channels using four push buttons. At the bottom of the display, you can see which channel is currently active. You should always take a look at this before turning the buttons for the vertical setting!

The combination of available bandwidth and maximum sample rate is well chosen for this oscilloscope. You might find the input bandwidth of 50 MHz a little low, but it is sufficient for most applications as long as you are not really working with HF circuits. Remember: the samplng rate drops to half or a quarter when using 2 or 3–4 channels simultaneously. At 50 MHz and 250 MS/s per channel you only have 5 samples/period, which is far too low for a good reconstruction of the signal. However, the combination of this bandwidth and sampling rate for a single channel does provide 20 samples/period.

The display of the DS1054Z is easy to read, even at an angle. I did have a tendency to set the brightness a little higher. A lot of information is visible on the screen, mainly due to the presence of the menus associated with the push buttons on both sides of the screen. All in all, that does result in a fairly busy display because a number of things are also shown at the top and bottom of the display. The available space for displaying the waveforms is, therefore, relatively small.

The trigger section of the DS1054Z offers no less than 15 different trigger functions including patterns, time-outs, windows, delays, as well as trigger capabilities for the supported serial protocols. The device has an internal sample memory of 24 Mpts. This is quite large but, just like with the A/D-converter, this memory also needs

to be divided over the active number of channels. Even then, the capacity is still more than sufficient.

Rigol has done its best to accommodate a whole host of measurement functions in this device (**Figure 4**). Using the buttons on the left you can quickly select from 32 parameters, allowing the measurement values (max. 5 at a time) to then appear at the bottom of the image. In addition, there is a 'normal' frequency counter. You can also choose to display 20 parameters at the same time. And there is also a static function that shows the deviations over a number of periods for up to 5 parameters. You can also take cursor measurements, both automatically and manually.

Under the Math button we find a large number of mathematical operations ranging from a simple summation to logarithm, exponent, integration, and differentiation. This list also includes the FFT function. This can be used to display the frequency spectrum of a signal. A nice extra, but I had some trouble to set all parameters in such a way that a usable picture appeared. Rigol can tinker with that in the software.

The DS1054Z also offers a lot of extras, particularly at this price. Here I've only described a few things that I've tried or noticed. It is, of course, also possible to connect the oscilloscope to a computer and control it from the PC. On the Rigol website the Windows programs UltraSigma and UltraScope are available for this purpose.

Conclusion

The Rigol DS1054Z is a versatile oscilloscope, especially now that it comes with so many software extras (**Figure 5**). Although it has been on the market for several years, it can still easily compete with newer devices in this price range. For €400 you get a reliable device with four channels and a lot of features that you will support you in the lab for years to come.

200463-03

Figure 4: Display of all measured values simultaneously.

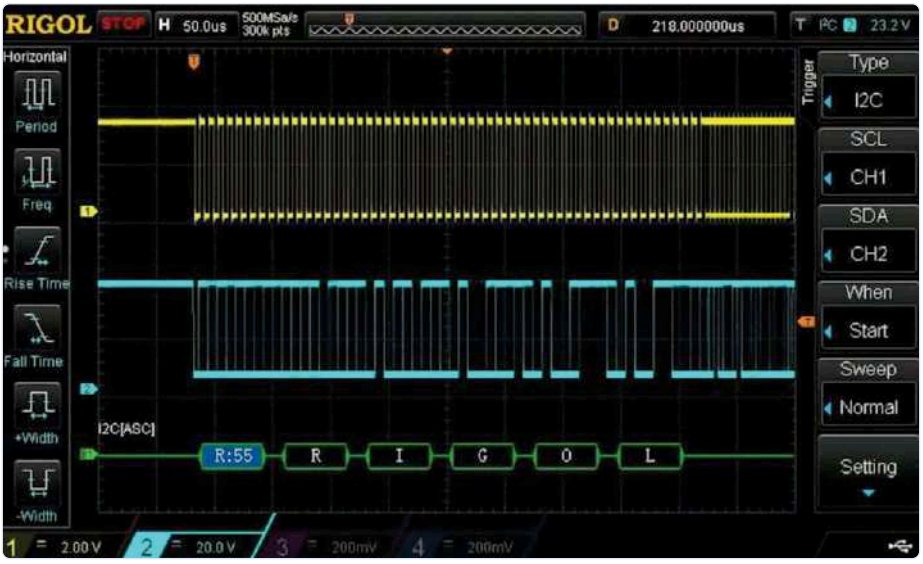
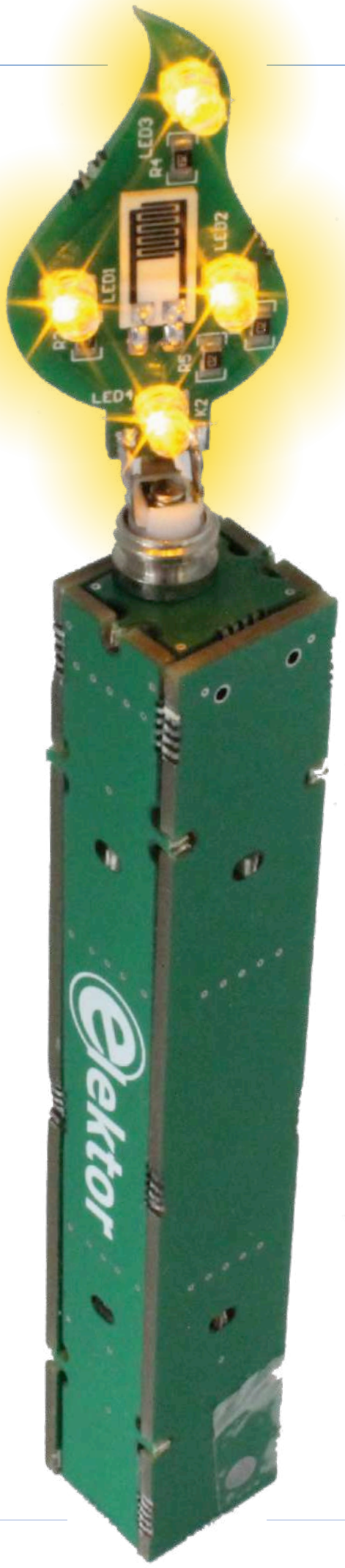


Figure 5: Decoding of I²C is also included.

Christmas Candle

Blow It Out!



Every year, we try our very best to present a new Christmas gadget in our magazine, which is certainly not an easy task. It must be something nice, funny, affordable, unique, relatively easy to build and preferably a perfect mix of all these aspects. And we are aware of the fact that the market is flooded with all kinds of Christmas gadgets for prices that are difficult (if not impossible) to compete with! Originality, or uniqueness, is perhaps the most difficult issue to tackle, and this time we decided to use two older Elektor projects and to combine these into something new!

Contributors
Idea: **Clemens Valens, Luc Lemmens**
Design and Text: **Luc Lemmens**
Illustrations: **Jan Visser, Luc Lemmens**
Schematic: **Patrick Wielders**
Editors: **Jens Nickel, C. J. Abate**
Layout: **Giel Dols**

PROJECT DECODER

Tags
Christmas, Holidays, Home & Garden
Level
entry level – intermediate level – expert level
Time
About 4 hours
Tools
Soldering tools (SMD and through-hole)
Price
€20 approx.

The first project we recycled for this year's Christmas gadget was published in the December issue of *Elektor* in 2011: the Electronic LED Candle [1]. From this article only the idea of 'blowing out an electronic LED candle' survived in this year's design; the electronics are changed completely. In 2011, a PIC16F1827 microcontroller controlled standard yellow SMD LEDs for the flickering light effect, whereas or new design uses 3mm LEDs with built-in electronics for this task.

Furthermore, the microcontroller's internal ADC measured the voltage across an NTC to detect if the candle was blown out.

Apparently, breath temperature is about the same as ambient temperature, so breathing on an NTC under normal conditions only has minimal effect on its resistance. The trick used in the older project was that the NTC was heated up by its own power dissipation and exhaling on the NTC cooled it down, resulting in a detectable voltage change on the microcontrollers analog input. However, in our new design, we wanted the candle to be powered by one single 1.5V battery and draining at least 15mA from the battery just for heating up the 'sensor' is rather a waste of energy, we want to use as much battery power as possible for the light effect. On top of that, discarding the PIC (plus software) using analog electronics made the design and building of the candle easier.

The Totally Archaic but Practical Interceptor of Radiation (AKA TAPIR) electro-smog sniffer [2] dates back to 2012 when it was published in *Elektor* magazine, and we used this project mostly for its 'mechanical' design: an enclosure made of smaller PCBs and containing, among others things, a battery holder for one AAA-cell (mini penlight) and stereo jack connector. In our new project the jack plug connects a small flame-shaped PCB which holds a sensor and four 3mm flickering LEDs, mimicking a candle. At least... if you think the rectangular casing looks like a candle well enough. But nothing prevents us from decorating it further on the outside afterwards, as long as the push button and the battery compartment remain accessible.

Special LEDs are used

The flickering candle LEDs used in this new Christmas project are cheap and readily available on the Internet, as always most affordable in web shops in the Far East. As the name suggests, they have built-in electronics to make the light flicker like a real candle flame, only an external current limiting resistor is needed. From the outside, they look just like ordinary 3mm or 5mm LEDs and are available in different colours. For our Christmas candle we need the 3mm version, preferably yellow or orange.

These components are used in (almost) all cheap battery-powered electronic candles you can buy, which contain only the LED, a resistor and a 3V coin cell or two AA or AAA batteries. If you can't find a supplier for the LEDs seperately, you can also buy some of these ready-made candles and scavenge the flickering LEDs to be used in this project.

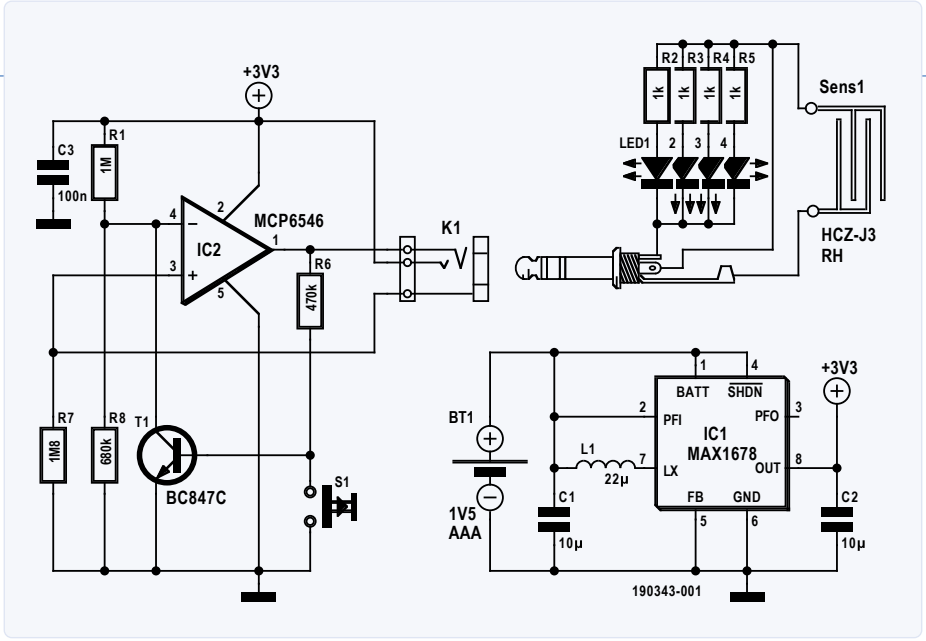


Figure 1: The schematics of the Christmas candle.

And some more hardware, of course

A 1.5V supply voltage from one battery is not enough to light the flickering candle LEDs properly, so we used a small step-up converter (IC1, a MAX1678) to boost the battery voltage up to 3.3V (see **Figure 1**). With a minimal input voltage of 0.7V, this IC is perfect to use up batteries that are discharged too far to be used in other appliances. In our prototype even an input voltage of 300mV was sufficient to light the candle! Perfect for squeezing out the last electrons from your old, 'empty' batteries.

As pointed out earlier, measuring temperature to detect if the candle is blown out is no option for a battery-powered device. To keep things simple, small and affordable, we chose an HCZ-J3 (Multicomp) humidity sensor to detect breath. Although these components

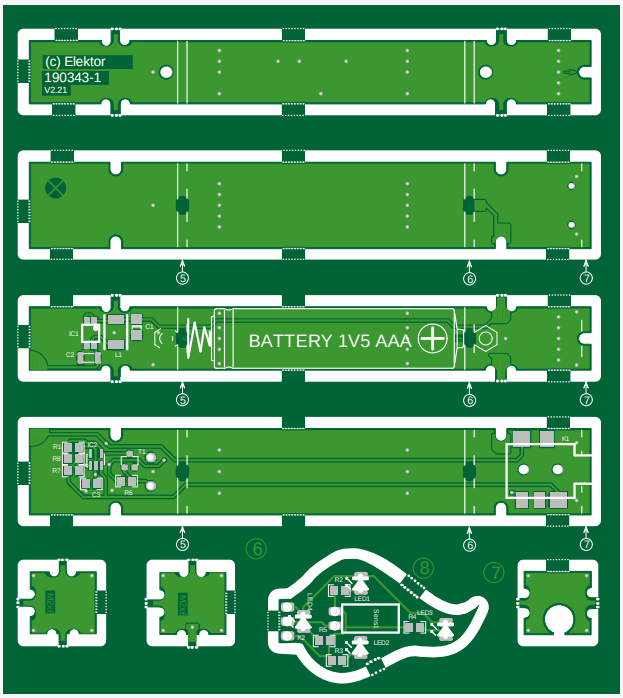


Figure 2: Eight boards are combined into one PCB panel (displayed here at 75% of their original size).

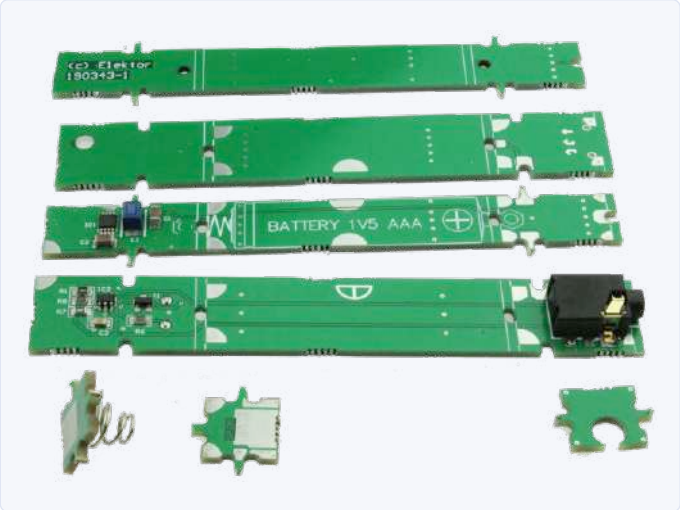


Figure 3: Solder most parts before taking the panel apart.

Figure 4a: The flame PCB, LEDs and sensor with the jack plug connected.

Figure 4b: Flame board, detail showing interconnection to the jack plug.

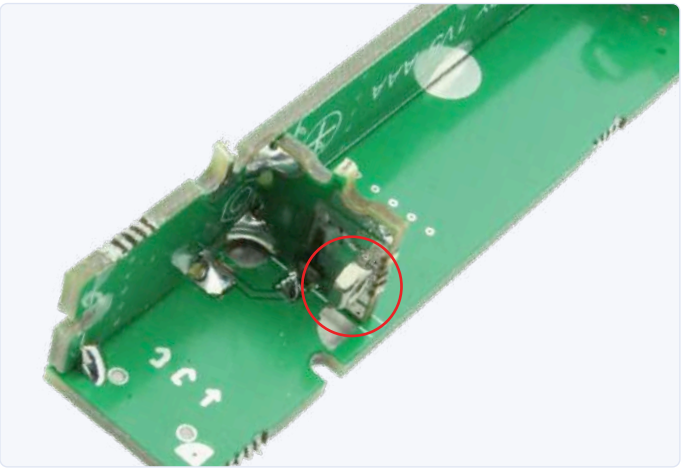


Figure 5: Battery contact boards mounted, see the nut for fixing the lid.

are designed to be used in AC circuits, they proved to work well in the circuit we used. The only downside with a humidity sensor is that it takes a few seconds before their resistance reverts to the ambient humidity level (i.e., it takes a while before the candle can be lit again).

The rest of the electronics are fairly straight forward: a low power comparator (IC2, MCP6546) triggers at the voltage on its non-inverting input. The voltage on this pin will rise when the humidity level at the sensor rises, i.e. when we exhale on the sensor. If it is higher than the voltage on the inverting input (± 1.35V from voltage divider R1 and R8), the output of the comparator goes high. Transistor T1 then pulls the inverting input to ground level, latching the LEDs in OFF state until pushbutton S1 is pressed, provided that the voltage level on the non-inverting input is (again) below the 1.35V threshold, of course.

Building the candle

First of all, a special note on the MCP6546 comparator IC: when you order this part, make sure that you get the MCP6546**R**. This suffix is very important as Microchip produces three SOT23-5 versions of this component that are NOT pin-compatible! The 'R'-version is the one you need for this candle.

The candle and its enclosure consist of eight smaller circuit boards, for production joined in one PCB- panel (**Figure 2**). You can download the Gerber files from the Elektor webpage and use a PCB service of your choice.

For your own convenience, leave this panel intact until (most) components are soldered, like in **Figure 3**. The individual PBCs are numbered (one through eight) and there is a kind of legend on the panel that shows which part of the PCB fits where.

There are silver-colored openings in the green solder mask at several edges and corners of the boards. These are areas that are used to join the boards by soldering. Some of the joints are only necessary to keep the boards together, others also important to interconnect the circuits on the PCBs.

Even though most of the components used in this project are SMD parts, soldering can be done with a normal iron with a small tip, using thin soldering wire. The job will be easier

if you have a solder paste dispenser, a hot air soldering tool or reflow oven of course, but it can be done with standard tools. In all cases, desoldering wick and solder flux may come in handy to remove excessive solder and (most of the time inevitable) short circuits, particularly between the pins of the two tiny integrated circuit packages. Assemble the latter components first and check the soldering before adding the other SMD parts. Finally, it's time to solder the through-hole parts.

Before soldering K2, the 3.5mm stereo jack plug that connects the flame PCB to the candle stick, this board must be cut from the frame. Remove the cap from the plug and use some short pieces of wire to solder the PCB to the plug. Take a close look at **Figure 4** to see how they must be interconnected.

To close the 'lid' of the battery compartment, you need to solder two hex-standoffs or nuts to the smaller (battery contact) PCBs, as shown in **Figure 5.** However, in our prototypes this proved to be not absolutely necessary, as the enclosure fits tight enough to keep the lid in place without using screws.

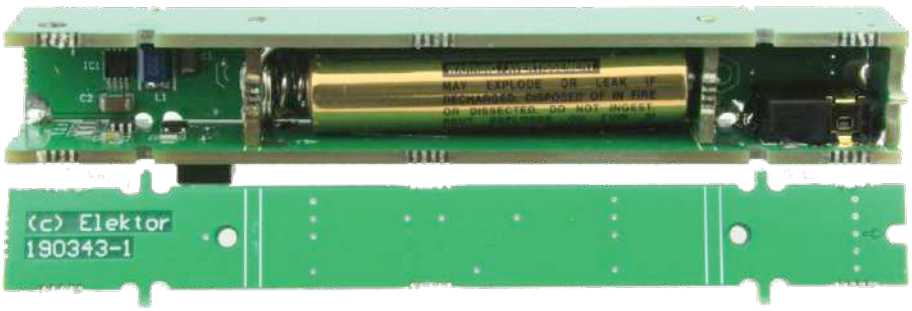
With all components on the boards, it's time to take all PCBs out of the panel, but do not immediately start soldering the enclosure! First, fit the individual boards together without connecting anything, making sure which PCBs fits how and where. You may need to remove (cut or file) small remainders of the panel to make it fit perfectly.

If you have the standoffs or nuts soldered for closing the candlestick, this is also the time to check if they are well aligned with the screw holes in the lid. Adjust if necessary. Then, solder some wires to interconnect the boards at their solder tabs and use a benchtop power supply for testing. This will give more space for measuring and repair, if needed. Of course, the wires will be removed later and the PCBs soldered together when the candle is working like it should.

And then finally, the candle is ready for use. Admittedly, there is nothing like real candlelight to create a Christmas atmosphere, but we hope that this electronic version will also be appreciated. We wish you a lot of fun building this Christmas gadget and — of course — a very merry Christmas!

190343-01

Questions or Comments?
Do you have questions or comments about his article? Email the author at luc.lemmens@elektor.com or contact Elektor at editor@elektor.com.



COMPONENT LIST

Resistors
(all SMD 0805)
R1 = 1 MΩ
R2,R3,R4,R5 = 1 kΩ
R6 = 470 kΩ
R7 = 1.8 MΩ
R8 = 680 kΩ

Inductor
L1 = 22 µH, size 1812,
TDK NLC453232T-220K-PF

Capacitors
C1,C2 = 10 µF, 10%, 25 V, X7R, size 1206
C3 = 100 nF, 10%, 50 V, X7R, size 0805

Semiconductors
LED1,LED2,LED3,LED4 = flickering candle LED yellow 3 mm (see text)
T1 = BC847C
IC1 = switching Boost regulator MAX1678EAU+
IC2 = Analogue comparator MCP6546RT-E/OT

Miscellaneous
K1 = 3.5 mm stereo jack connector SMD (CUI SJ1-3514-SMT-TR)
K2 = 3.5mm stereo jack plug
Battery contact spring (Keystone Electronic 211)
S1 = 6 mm pushbutton (Alcoswitch FSM2JRT)
Sens1 = Humidity sensor HCZ-J3 (Multicomp)
PCB 190343-1 V2.21

RELATED PRODUCTS

- **Gerber PCB files for free download at**
  www.elektormagazine.com/190343-01

WEB LINKS

[1] **Electronic LED Candle, Elektor 12/2011**: www.elektormagazine.com/magazine/elektor-201112/19783
[2] **TAPIR Sniffs it Out!. Elektor 7-8/2012**: www.elektormagazine.com/magazine/elektor-201207/19936
[3] **Download**: www.elektormagazine.com/190343-01

Tunable
Valve Sinewave Generator

Retro is back!

By **Prof. Dr Martin Ohsmann** (Germany)

In my younger days I built what was known as an '0-V-2' valve receiver. Fuelled by nostalgia for the pre-transistor era, I decided to build another circuit using valves. Designs for simple receivers and high-end amplifiers are a dime a dozen, which prompted me to go for something a little more unusual.

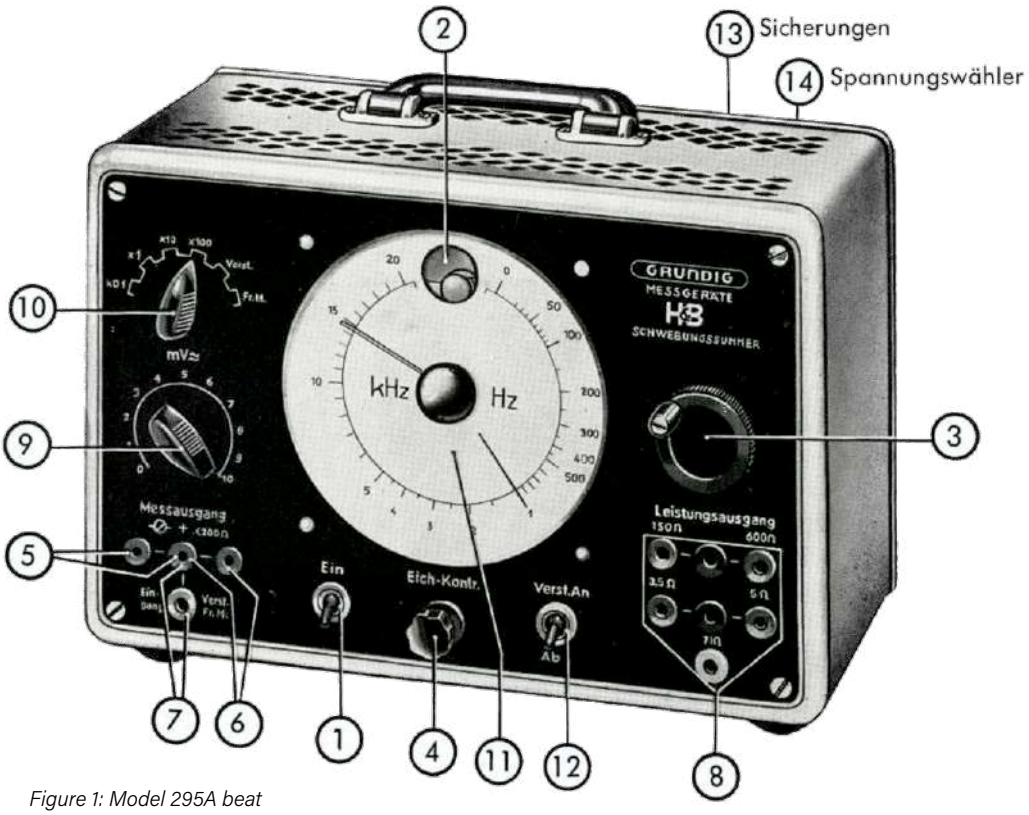


Figure 1: Model 295A beat frequency oscillator.

While preoccupied with these thoughts I happened upon the manual for the model 295A beat frequency oscillator made by Grundig, Hartmann and Braun (see **Figure 1**). This unit can be tuned in a single sweep over the entire audio frequency band from 20 Hz to 20 kHz without needing to change ranges. Such an enormous 1000:1 frequency ratio is not feasible with simple sinewave generators. This brings us, therefore, to the principle of heterodyning.

The heterodyne principle

Figure 2 shows the basic idea. Oscillator 1 creates a fixed frequency f_1, here approximately 235 kHz. Oscillator 2, in contrast, is adjustable and generates a frequency f_2 that can vary from 235 kHz to 270 kHz, which is less than a 20 % variation in frequency. It is easy to implement a narrow frequency range like this in a simple oscillator with the help of a variable capacitor.

The output signals from the two oscillators are fed to a mixer. The output of the mixer includes, among other products, the frequency difference $f_1 - f_2$. The other products (the sum of f_1 and f_2 as well as the sums and differences of integer multiples of them) can be removed using a suitable low-pass filter. Finally, we amplify the output signal to provide a low-impedance source. Using the heterodyne principle, it is also possible to build tunable RF oscillators that operating into the GHz range.

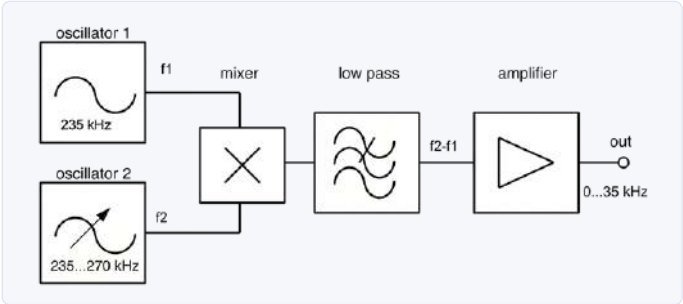


Figure 2: Principle of operation for the heterodyne signal generator.

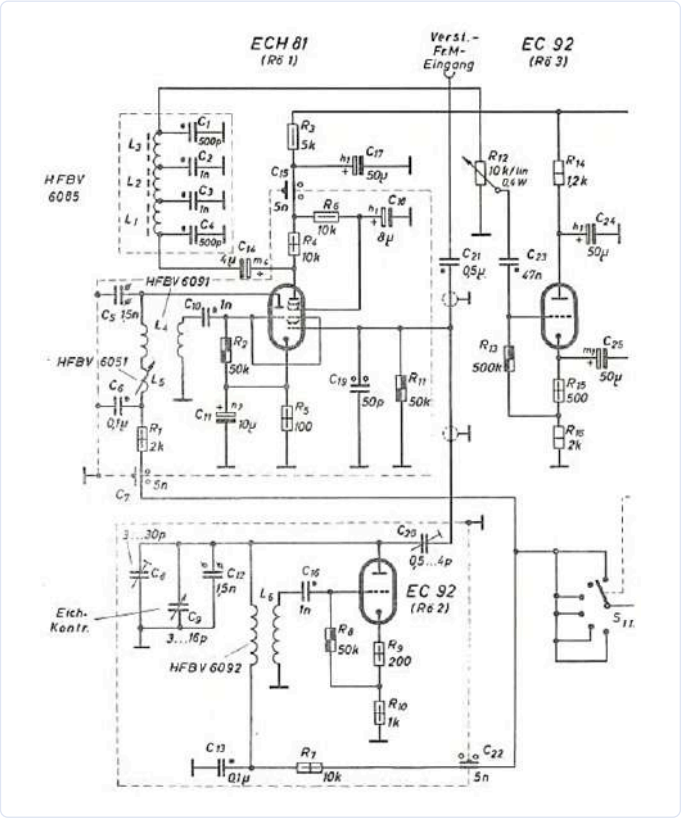


Figure 3: Extract from the circuit diagram of the model 295A.

The original

Figure 3 shows an extract from the circuit diagram including the two oscillators, the mixer, the low-pass filter and the final amplifier stage. The valve labelled 'Rö2' is responsible for generating the fixed frequency. Its anode is connected to a resonant circuit which determines the frequency of oscillation, and there is a transformer to provide feedback to the grid electrode.

The variable frequency is generated in the same way using the triode section of an ECH81 (left-hand part of the circuit next to the valve labelled 'Rö1'). The frequency is adjusted using variable inductor L5. The outputs from the two oscillators are then taken to grids G1 and G3 of the heptode section of the ECH81 and the mixer products appear at the heptode anode.

The low-pass filter is constructed from L1, L2 and L3 along with C1, C2, C3 and C4. The filtered output signal is then passed through a potentiometer to allow adjustment of the signal level to the valve labelled 'Rö3' that is configured as a cathode follower. The low-impedance sinewave output signal is available at the cathode of this valve.

My reconstruction

To make the construction of a version of this unit easier, I tried out a couple of modifications. Instead of a variable inductor to adjust the frequency I used a variable capacitor. The feedback circuit in the oscillators was also modified to use tapped inductors instead of transformers. My 'free form' design built on the bench for testing is shown in **Figure 4**. **Figure 5** shows the complete circuit diagram for your edification. Triode V1 implements the variable-frequency oscillator and its output signal passes via a potentiometer to grid G3 of the mixer valve V2.

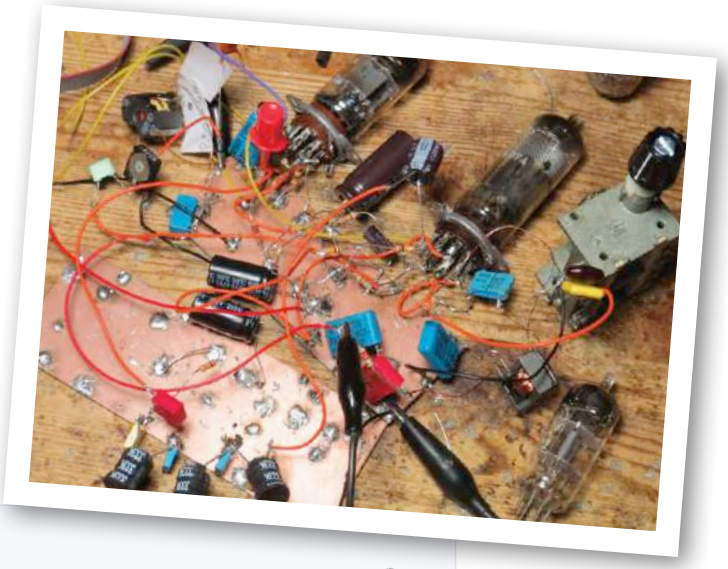


Figure 4: Experiments on the bench.

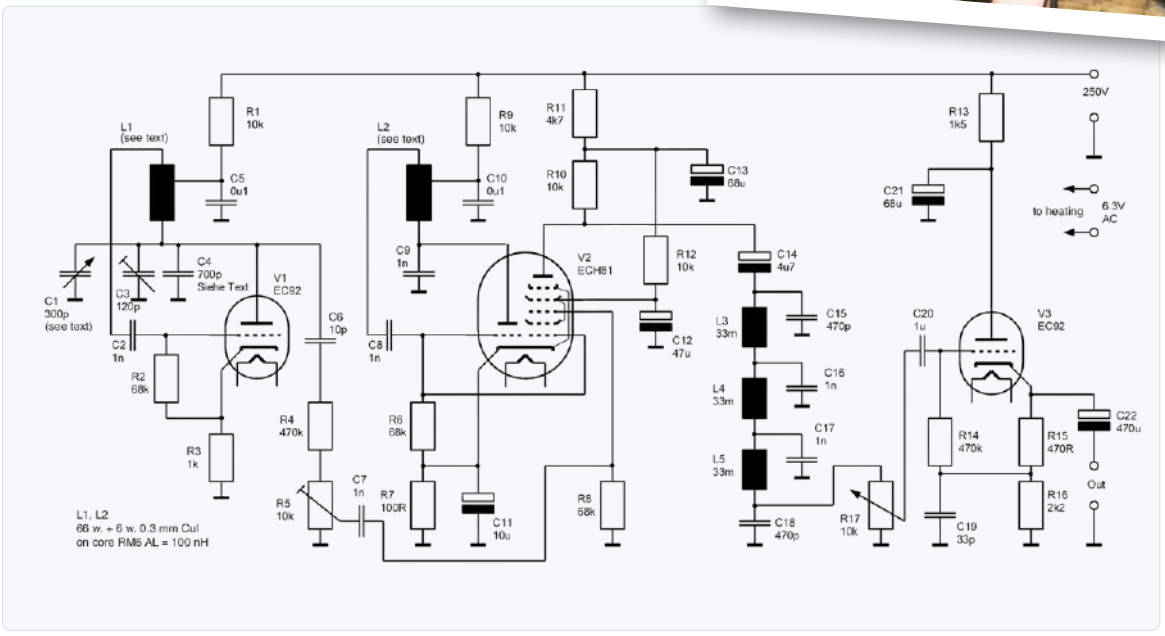


Figure 5: Circuit diagram of the optimized replica. The final prototype (apart from the power supply) can be seen in Figure 6.

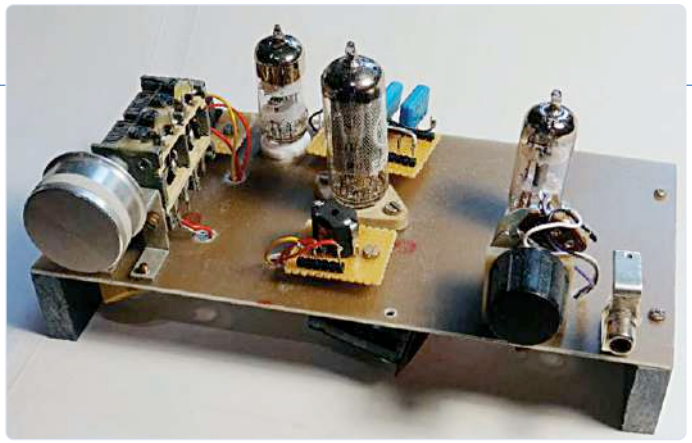


Figure 6: Final prototype of the sinewave generator.

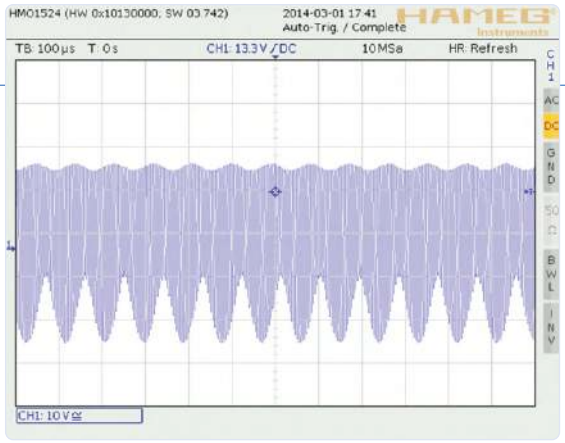


Figure 7: Signal at the anode of the mixer valve.

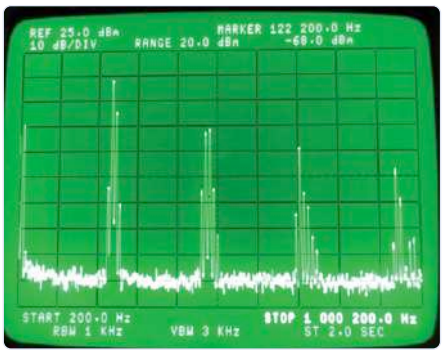


Figure 8: Spectrum of the mixer output.

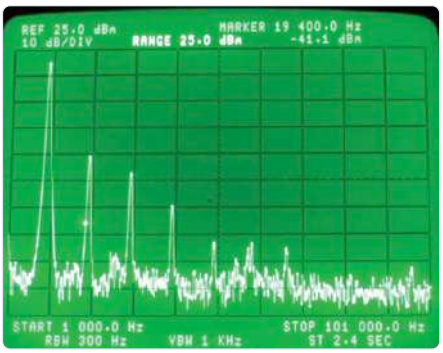


Figure 9: Output spectrum at high mixer drive level.

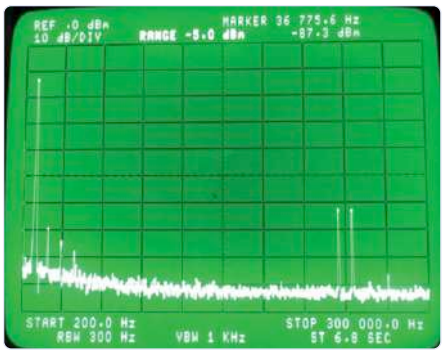


Figure 10: Output spectrum at lower mixer drive level.

The triode section of V2 is used to form the fixed-frequency oscillator and its output is connected to the grid G1 of the heptode. The mixer output appears on the heptode anode.

Measurements

In order to check the basic operation and signal quality of my final prototype (see **Figure 6**) I undertook a few measurements. I measured a signal amplitude of approximately 200 V_{pp} on the oscillator anodes. Of particular interest is, of course, the anode signal on the mixer valve, which is shown in **Figure 7**.

The rapid oscillations are the two high-frequency tones, while the lower envelope represents the generated sine wave signal. It is this that we subsequently extract with the low-pass filter.

Figure 8 shows the spectrum of the anode signal on the mixer valve. It is easy to see the many product tones generated from the two oscillators. Adjusting potentiometer R5 alters the amplitude of the output signal but, at the same time, it also alters the drive level of the mixer valve. If, for example, we set R5 so that a signal amplitude of 8 V_{pp} appears on grid G3, we obtain an output amplitude of around 7 V_{pp}. At such a high output level a considerable amount distortion in the output signal is generated. The spectrum is shown in **Figure 9**. The second harmonic is only around 35 dB below the fundamental and the distortion factor is not particularly impressive, lying at over 1%.

If, on the other hand, we adjust the amplitude of the signal on G3 to just 1 V_{pp}, we obtain an output signal amplitude of 2 V_{pp} whose spectrum, shown in **Figure 10**, looks a lot healthier. The second harmonic is now more than 50 dB below the fundamental and the distortion factor is down to around 0.3 %, which is much more satisfactory. There are, however, still residual tones at 235 kHz and 250 kHz being capacitively coupled through the low-pass filter to the output signal. These could be mitigated by improving the screening of the high-frequency parts of the circuit.

So, in conclusion, we can see that it is possible to build a useful, adjustable sinewave generator covering the whole of the audio frequency range using just three valves and a handful of other components.

Postscript: Perhaps not all hobbyists are familiar with the classic '0-V-2' RF terminology used by radio amateurs. The first number indicates the number of RF preamplifier stages before the detector, while the number after the 'V' gives the number of audio amplifier stages. Thus a type 0-V-2 receiver is an 'Audion' design with one valve, no RF preamplifier stages, and two audio amplifier stages.

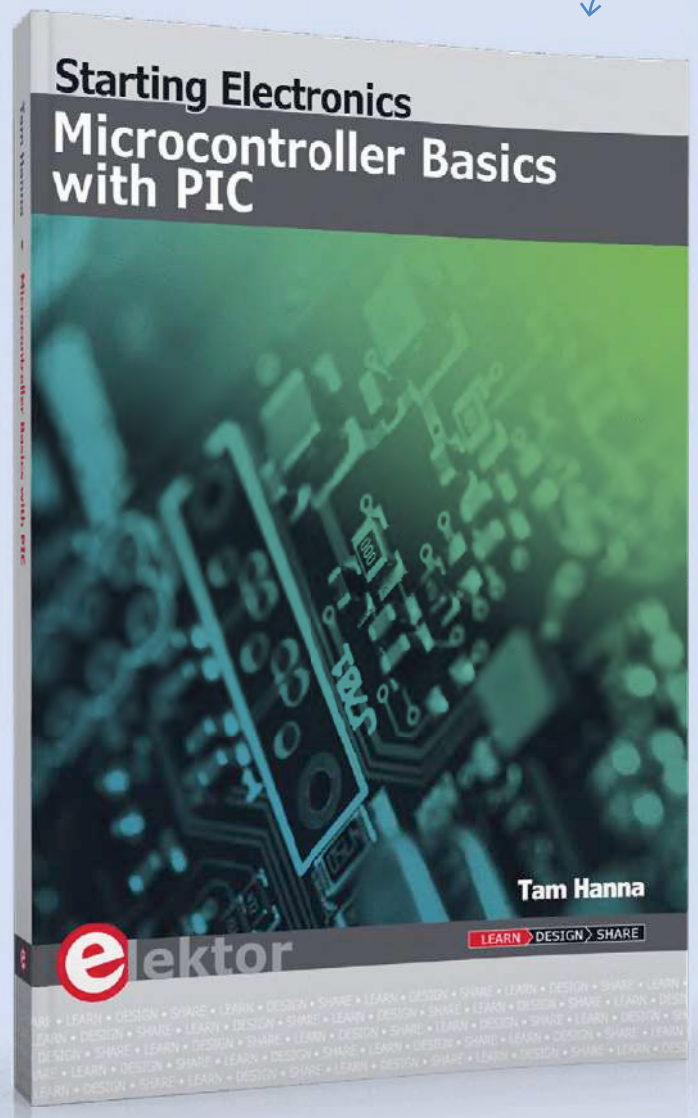
190312-02

SHOPPING LIST

- **'Valve Projects from 6 V to 60 V'**
  ('Röhren-Projekte von 6 bis 60 V', PDF in German only)
  www.elektor.de/roehren-projekte-von-6-bis-60-v-pdf
- **'Valve Amplifier Circuits'**
  ('Röhrenverstärker-Schaltungen', PDF in German only)
  www.elektor.de/roehrenverstaerker-schaltungen-pdf
- **JOY-iT JDS6600 Signal Generator (60 MHz) & Frequency Counter (100 MHz)**
  www.elektor.com/joy-it-jds6600-signal-generator-frequency-counter

NOW
€ 31.46
for members

Microcontroller Basics with PIC



Billions of low-power microcontrollers are deployed throughout the Internet of Things (IoT). Want to learn how these tiny devices work? Would you like to start building PIC microcontroller-based designs?

In Microcontroller Basics with PIC, author Tam Hanna presents all the essential aspects of microcontroller programming, without overloading you with unnecessary details. Topics covered:

- PIC microntrollers
- An intro to Assembly
- Program sequence control
- Getting started with C
- Hardware accelerated buses
- Storing data
- And much more

Get started with microcontrollers right away. The possibilities are endless!

NOW
€ 31.46
for members

Read more on:
www.elektor.com/19188

elektor
design > share > sell

The Rigol DG2072 Generator

By **Philippe Demerliac** | Cyrob (France)

Do you require a high signal fidelity in a reliable and capable waveform generator? Rigol's DG2000 Series might be a good solution. The hybrid touch interface and industrial design make it a smart option for your lab, while standard Ethernet and USB enable you to programmatically control your instrument. Let's take a closer look at the DG2072 generator.

Philippe Demerliac runs Cyrob.org and posts videos at https://youtu.be/tm_VqCWMY34.

Figure 1: The front of the DG2072.

Figure 2: The back of the DG2072.

My overall rating for the Rigol DG2072 Generator is 18/20. I hesitated to put it in my "pro" category, but due to its price and its performance, it will undoubtedly have its place in professional labs. The DG2072 is definitely a solid investment for hobbyists that should cover most of their needs for many years to come.

A few years ago, I bought a high-quality Rigol DG4162, but the DG2072 is even better! It has a tactile interface that is becoming the standard for measuring devices, and I am quickly becoming adept at using it. (I often get frustrated when I press the screens on my devices that don't have one.) All the measurements I took with the generator showed it was well within spec and well calibrated.

The DG2000 series is also quite innovative. The dual-tone mode, the generation of logic signals, and the sequence mode are very useful. So, for me, if you are looking for a quality DDS gen, it will be a great investment. The price is certainly higher than other competing models, but it is justified by the functional richness and the quality of the signals generated. Available at different prices depending on the maximum possible frequency, unless you really need it, the 50-MHz version is quite sufficient for amateur use. You might think it looks is a little "aggressive," but it just depends on your taste.

Pros and cons

Let's start with some of advantages:

- A pleasant touchscreen interface that responds very quickly.
- Modulation possibilities.
- Functional richness.
- The large number of waveforms are standard.
- Users can choose the output impedance.
- Users can define the amplitudes in V_{pk}, V_{rms}, dBm, etc.
- The "Marker" function in Wobulation mode.

Figure 3: Screen during startup.

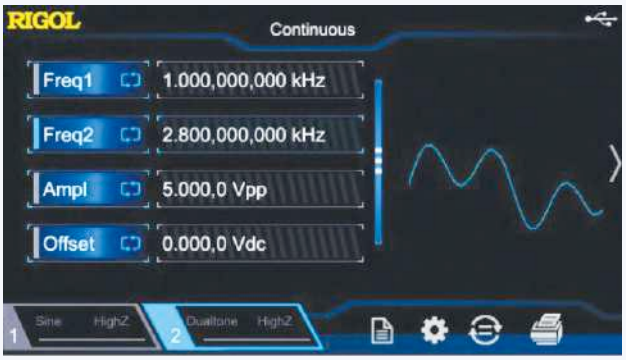


Figure 4: channel 2 in harmonic mode.

Figure 5: The Marker is available in Wobulation mode.

Figure 6: Sequence mode allows several waveforms to be chained together in "Direct" mode.

- Advanced modes (bit, RS232, sequence).
- The "Dual Tone" function.
- SiFi II techno for better reproduction.
- The "Vrai" inter on the front panel. (The *real* mains switch is on the front panel.)
- Silence (no fan noise).
- The quality of the touch screen and the hyper-readable display.
- The resolution of 16bit (65536 possibilities, or 152 µV per point under 50 Ω).
- 16 mega points of sampling length and speed of 250 MSa/s.
- Low phase noise (10 MHz: ≤105 dBc/Hz).
- Overall quality.

And now let's consider the disadvantages:

- Sockets on the rear panel.
- The rear sockets, which function as inputs or outputs depending on the modes.
- The switch on the front is "cheap." Why not a push button?
- The buttons on the front are a bit hard.
- The frequency meter mode: the frequency display is too small in frequency meter mode; the inability to define an offset in frequency meter mode; the inability to take screenshots in this mode; and limited entry dynamics.

First Contact

Figure 1 shows the front of the unit. Note that BNCs operate as inputs or outputs depending on the modes (**Figure 2**).

Figure 3 shows the screen during startup. Channel 1 is red. Channel 2 is blue. **Figure 4** shows channel 2 in harmonic mode. The Marker is available in Wobulation mode, which greatly facilitates its use (**Figure 5**).

Sequence mode allows several waveforms to be chained together in "Direct" mode, which eliminates phase accumulation faults (**Figure 6**). Simulating pseudo-random sequences of serial bits can really help in the development of digital transmission systems (**Figure 7**).

Figure 7: Simulating pseudo-random sequences of serial bits can help in the development of digital transmission systems.

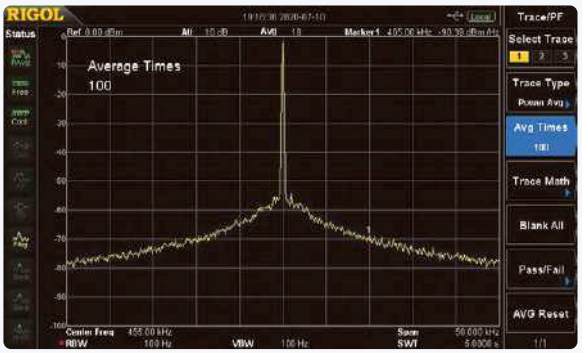


Figure 8: Spectral analysis.

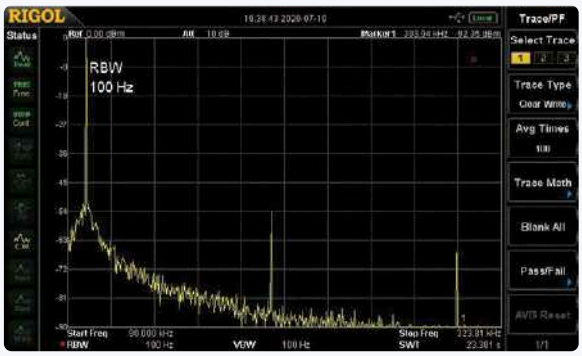


Figure 9: Output spectrum for a sine wave of 100 kHz to 0 dBm.

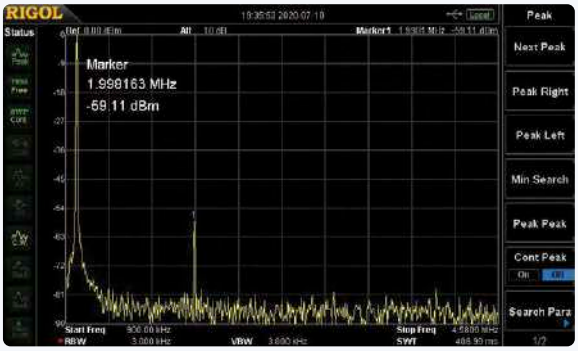


Figure 10: Spectrum for a sine wave from 1 MHz to 0 dBm.

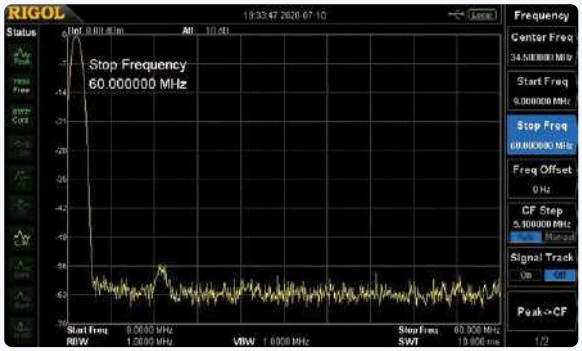


Figure 11: Spectrum for a sine wave from 10 MHz to 0 dBm.

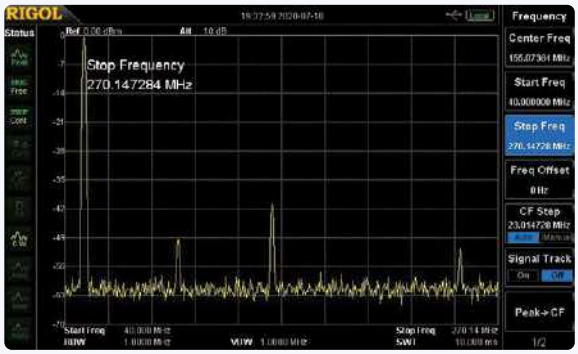


Figure 12: Spectrum for a 50-MHz sine wave at 0 dBm.

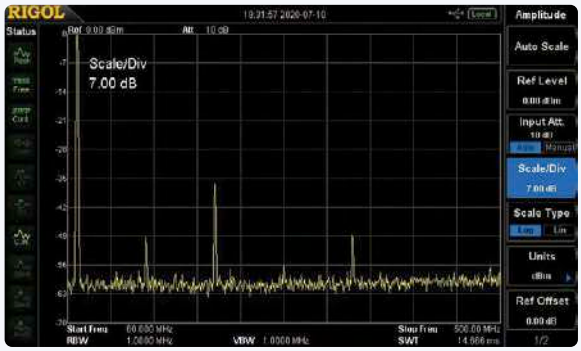


Figure 13: Spectrum for a 70-MHz sine wave at 0 dBm

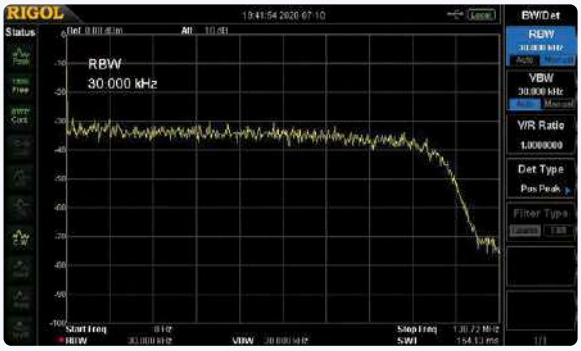


Figure 14: In white noise generator mode, the level is almost stable up to 100 MHz.

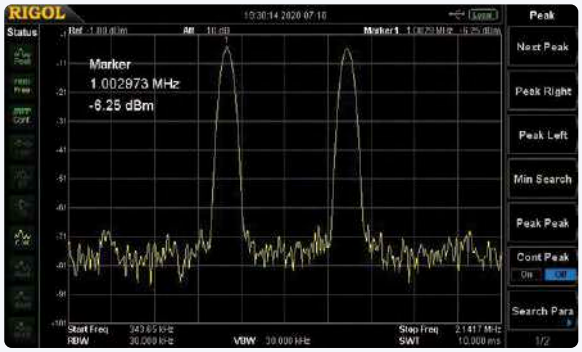


Figure 15: The Dual Tone mode and its camel curve.

Figure 16: I brought the connectors to the front.

Figure 17: The connectors are on the side.

Figure 18: I machined this part.

Spectral analysis

I made some measurements with the Rigol DG815TG spectrum analyzer. I attempted to measure the phase noise normalized to 10 kHz of the carrier, but my analyzer was not sensitive enough for that (**Figure 8**).

Figure 9 shows the output spectrum for a sine wave of 100 kHz to 0 dBm. The first harmonic is at -54 dB. **Figure 10** shows the spectrum for a sine wave from 1 MHz to 0 dBm. The first harmonic is at -59 dB.

Figure 11 shows the spectrum for a sine wave from 10 MHz to 0 dBm. The first harmonic is at -56 dB. **Figure 12** shows a spectrum for a 50-MHz sine wave at 0 dBm. The first harmonic is at -49 dB, the third highest at -40 dB.

Figure 13 shows the spectrum for a 70-MHz sine wave at 0 dBm. The first harmonic is at -49 dB, the third highest at -37 dB. In white noise generator mode, the level is almost stable up to 100 MHz, which is excellent (**Figure 14**)! **Figure 15** shows the "Dual Tone" mode and its "camel" curve.

Modifications

As usual, I was bothered by the connectors on the rear face, I brought them back to the front (**Figure 16**)! Of course, I did not modify the device. I just took advantage of the screws on the side (**Figure 17**). I had to machine the weirdly shaped piece you see in **Figure 18** and **Figure 19**. The thermo printable sleeves are easy to spot (**Figure 20**).

200453-01

Figure 19: How I connected the machined part.

Figure 20: Thermo printable sleeves.

RELATED PRODUCTS

- **Rigol DG2072 Function/Arbitrary Waveform Generator**
  www.elektor.com/dg2072

Questions or Comments?
Do you have questions or comments about this article? Email the author at philippe.demerliac@free.fr or contact Elektor at editor@elektor.com.

Editor's Note
Want to know more? Watch Cyrob's video about the Rigol DG2072 (in the French language) at https://youtu.be/tm_VqCWMY34.

WEB LINK

[1] **Cyrob, "Revue du DG2072 Rigol, merci Elektor!," July 11, 2020**: https://youtu.be/tm_VqCWMY34

Contributors
Author: **Philippe Demerliac** Editor, translation to English: **C. J. Abate** Layout: **Giel Dols**

KiCad Plugins and Add-ons

One of the cool features of the free and open-source schematic and PCB design tool KiCad is that it is extensible. Anyone can create plugins and add-ons for it in Python. Here we present a few of them.

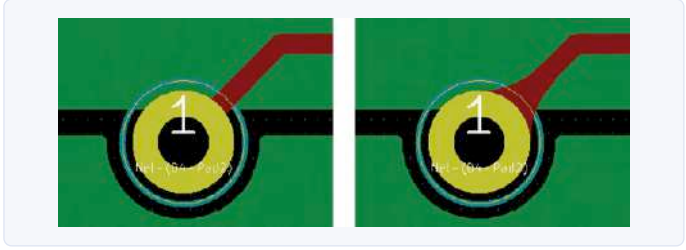
KiCad is a free and open-source tool for creating schematics and designing printed circuit boards (PCBs). Since its initial release way back in 1992, it has steadily been gaining popularity and at Elektor too we are using it more and more often.

One of the cool features of KiCad is that it is extensible, allowing anyone with an interest in (Python) programming (and electronic design) to develop plugins and add-ons for it. This has resulted in a multitude of tools and utilities being shared online. Some are more useful than others, and your mileage may vary, but checking some of them out may be worth it.

Our selection

Here are a few plugins and toolboxes that we hand-picked for you. A nice list of plugins and utilities for KiCad is maintained at https://github.com/xesscorp/kicad-3rd-party-tools.

Stretch – Action Plugin to make pretty PCBs. Paradoxically, even though KiCad is essentially a drawing program, it is not particularly good at it. Drawing straight lines works great, but curvy lines and complex shapes are not its strong points. Stretch remedies that by allowing PCB designs to be exchanged with Inkscape. Typical PCB design things are done in Pcbnew, artistic touches and other complex operations are added in Inkscape and the design is exported back to Pcbnew to e.g. generate production files.
https://github.com/JarrettR/Stretch



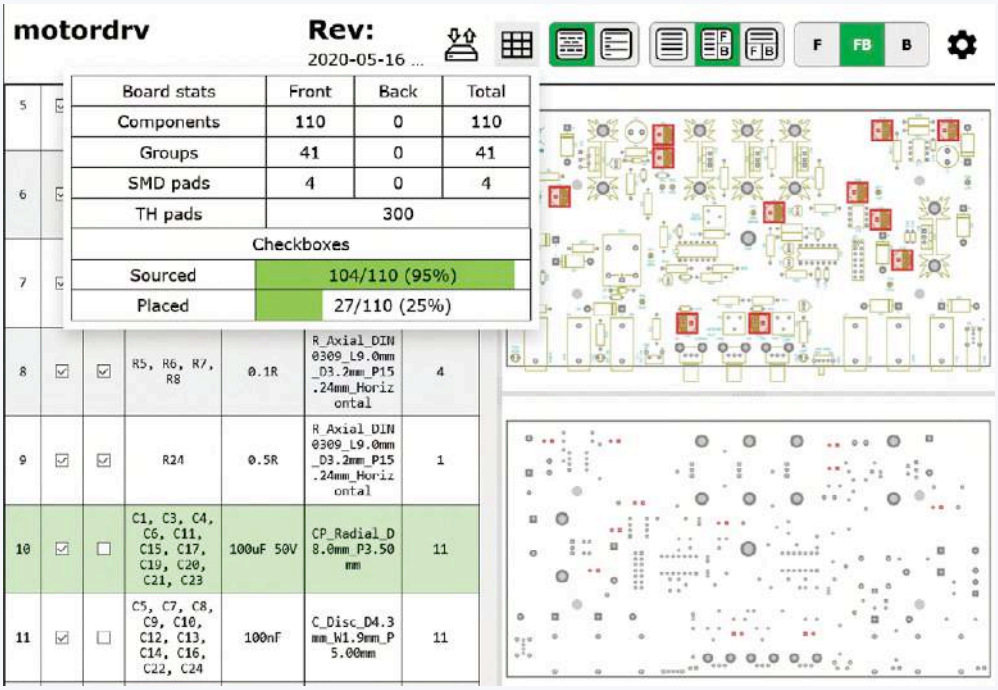
Action plugin 'Teardrops' before (left) and after (right).
Get it at https://github.com/NilujePerchut/kicad_scripts.

KiKit – Python toolbox for automatically producing panels, exporting production files, and creating board presentation pages. To make good use of this toolbox, some knowledge of Python programming may come in handy. Note that due to Python issues related to KiCad, this toolbox does not work on Windows (see inset).
https://github.com/yaqwsx/KiKit

Interactive HTML BOM – Action Plugin to create not only a bill of materials (BOM), but also an interactive component placement document that highlights graphically where a BOM part is located on the PCB. Parts can be searched by value or individually. Besides knowing where the parts go, this plugin also knows all the nets. This greatly speeds up locating a component or signal for board assembly, testing or repair.
https://github.com/openscopeproject/InteractiveHtmlBom

RF-Tools for KiCAD – This collection of Action Plugins facilitates the design of high-frequency and RF boards with features like track corner rounding, track length calculation, via fencing and solder mask expansion. It also has some utilities for RF footprint creation.
https://github.com/easyw/RF-tools-KiCAD

KiCad-Diff – Python3 program for keeping track of PCB revisions with Git, SVN or Fossil. It can generate visual as well as textual reports as a web page showing the differences between two revisions of a board design. Works on Linux, Mac-OS and Windows.
https://github.com/Gasman2014/KiCad-Diff



Example of an interactive report generated by the action plugin 'Interactive HTML BOM.'

KiField – A Python utility to extract all the component fields from a schematic or a library and place them into a spreadsheet for bulk editing. Field values can be modified, and new fields can even be added. When done the updated fields can be inserted back into the schematic or library.
https://xesscorp.github.io/KiField

200429-01

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

Contributors
Idea, Text and Illustrations: **Clemens Valens** Editor: **CJ Abate** Layout: **Giel Dols**

THINGS TO KNOW

- When installed properly so-called 'Action Plugins' are available from Pcbnew's menu 'Tools -> External Plugins...' and do something with the design currently loaded in Pcbnew. Stand-alone tools do not need Pcbnew to be running but can use its functions to work with data from files.
- On Windows KiCad installs its own Python interpreter, which means that the 'pcbnew' module is not installed in a system-wide Python installation you may have. Certain add-ons and toolboxes may therefore not work on Windows, so check this first before complaining about any problems you might encounter. This problem is hopefully resolved by a future version of KiCad.
- Details on plugin development for KiCad can be found at GitHub. Here you can also find the path to the folders where plugins must be installed.
- https://github.com/KiCad/kicad-source-mirror/blob/master/Documentation/development/pcbnew-plugins.md#typical-plugin-structure--ppi_pi_struct

Differential Oscilloscope Current Probe 2.0

Measure Currents Using Your Oscilloscope

By **Alfred Rosenkränzer** (Germany)

Measuring the dynamic behaviour of a circuit without a differential current probe is very challenging. To tackle such challenges we present an updated differential current probe design for oscilloscopes that has a single-ended output and wide bandwidth. Its standard gain of two can be increased by changing just a single resistor.

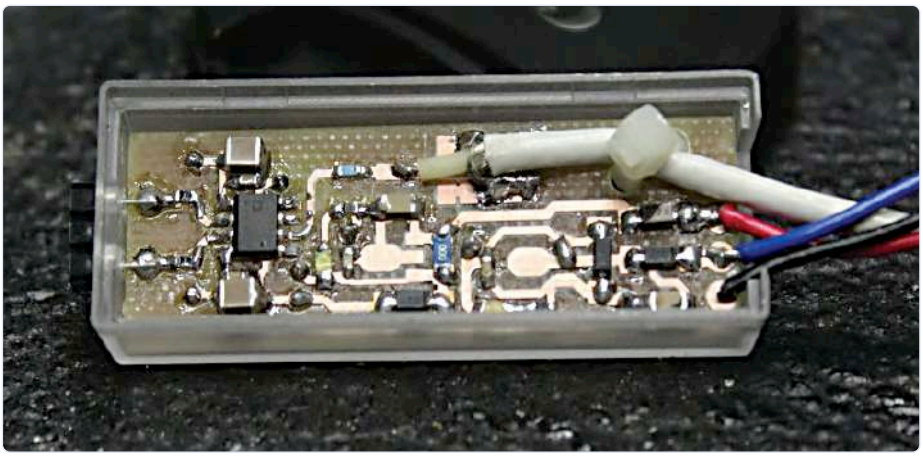


Figure 1: Prototype of the first version of the differential probe from [1].

PROJECT DECODER

Tags
Test and measurement, oscilloscope, current

Level
entry level – intermediate level – expert level

Time
Approx 2 hours

Tools
Soldering equipment (through-hole and SMD), mechanical tools

Cost
Approx £30 / $40 / €35

The '2.0' in the title indicates that this is not the first current probe the author has designed. **Figure 1** shows the prototype of Alfred Rosenkränzer's previous current probe design that was published in Elektor four years ago [1]. Compared to the old version, this new design offers considerably higher bandwidth and it was actually the limited 130 kHz bandwidth of version 1.0 that was the main motivation for developing this much improved model. However, this improvement is made at the expense of a reduced common-mode range on the inputs: read on for more details!

Current probes for oscilloscopes

The oscilloscope is indubitably the second most important piece of test equipment one can have. Compared to the multimeter, which occupies the number one spot, it has the advantage of being able to show the dynamic behaviour of a signal. However, it also has one decisive disadvantage: oscilloscopes are not normally capable of measuring currents directly. A current probe solves this problem. If you have ever wanted to see the dynamic behaviour of currents in a circuit, you will have come up against this difficulty. Of course you can insert a shunt resistor (or any other suitable resistor that comes to hand) and then measure the voltage drop across it using a normal (voltage) oscilloscope probe. The vertical amplifier, even on low-cost oscilloscopes, can display down to 5 mV per division, which is sensitive enough for this purpose. Unfortunately, this requires the ground of the oscilloscope to be connected to the other side of the resistor, which is not always desirable and which may affect the performance of the circuit. Furthermore, when using a two-channel oscilloscope, there is the possibility of introducing a short circuit if the second channel is used simultaneously to measure a voltage.

The solution to this problem is a compact differential amplifier that enables small voltage differences to be measured in a straightforward way without worrying about an overall offset voltage. A differential amplifier has what is called a 'common-mode voltage range' within which it only amplifies the difference between its two input voltages. Its output thus provides the difference as a voltage referenced to the ground potential of the oscilloscope, letting us conveniently measure small voltage drops across resistors anywhere in a circuit and (within limits) independent of any simultaneous voltage measurement.

The best part of this idea is that there are mass-produced ICs designed precisely for this task. Just add a couple of external components and the current probe design is practically complete.

Characteristics

The device chosen was the Analog Devices AD8421 instrumentation amplifier, mainly because of the wide bandwidth of 6 MHz claimed by the manufacturer. At under £10 the IC is also not too expensive and its low noise figures, low current consumption and high common-mode rejection seal the deal. More detailed characteristics can be found in the **AD8421 Technical Data** text box; for more information consult the manufacturer's data sheet [2].

Unfortunately, compared to the AD8479 used in the first version of the design, the AD8421 has a significant disadvantage. The earlier IC has a common-mode voltage range of around ±200 V to ±600 V, but the newer one must have its input levels maintained within its supply rails. With a ±15 V supply that means that the practical common-mode voltage range of the inputs is around ±12 V. Although that might sound rather low, ICs that combine all the features one would ideally want are rarer than hens' teeth. If you have a requirement to make measurements on a circuit with higher voltages, then version 1 from 2016 will be more suitable. If you need to measure higher-frequency signals on low-voltage circuits, this version will do the job better. It is simply a question of using the right tool for the job.

Current probe circuit

Alfred Rosenkränzer's project concept convinced the Elektor team and Ton Giesberts from the Elektor labs set to work on the idea and revise it slightly. The resulting circuit, shown in **Figure 2**, is configured for a gain factor of two. This is determined by the value of a single resistor, R5. The formula for the gain is:

$$A = 1 + (9.9\ k\Omega / R5)$$

If R5 is omitted the gain will be unity with a bandwidth of 10 MHz (according to the data sheet) since, in this case, the value of R5 is in effect infinite. With the suggested value of 10 kΩ for R5 the gain factor will be exactly 1.99, an error of less than 0.5 % relative to the desired value of 2, which is better than the tolerance of the resistor used. At this gain factor the predicted bandwidth of the circuit is at least 6 MHz and measurements indicate that around 8 MHz is achieved in practice. If you prefer a different gain, note that even with a gain of 100 the AD8421 still achieves a bandwidth of 2 MHz!

The input filters formed by R1 and C1, and by R2 and C2, provide simple protection against short-term overvoltage conditions and transients. They also compensate for the slight increase in gain of the AD8421 in the neighbourhood of its bandwidth limit, thus helping to flatten the response of the current probe.

Although the inputs to the AD8421 are very robust, it is nevertheless necessary to avoid applying long-term excessive voltages. As mentioned above, the ±15 V supply limits the practically-usable common-mode input voltage range to around ±12 V.

The current probe has its own voltage regulator circuit with an EMI filter formed from C13, C14 and common-mode choke L1. This attenuates possible interference from the power supply used for the probe, which is especially important if it is powered from a switch-mode supply. For occasional use the probe can be powered from a bench supply with a symmetrical output, although it is more convenient if it has its own mains supply. D3 and D4 protect against reverse polarity connection: they are not intended to rectify an AC input!

LED1 and LED2 indicate when the two supplies are present and the probe is active. Their series current-limiting resistors are chosen to have a high value to avoid dazzling the user. The two 10 MΩ resistors R3 and R4 ensure that unconnected inputs are pulled to

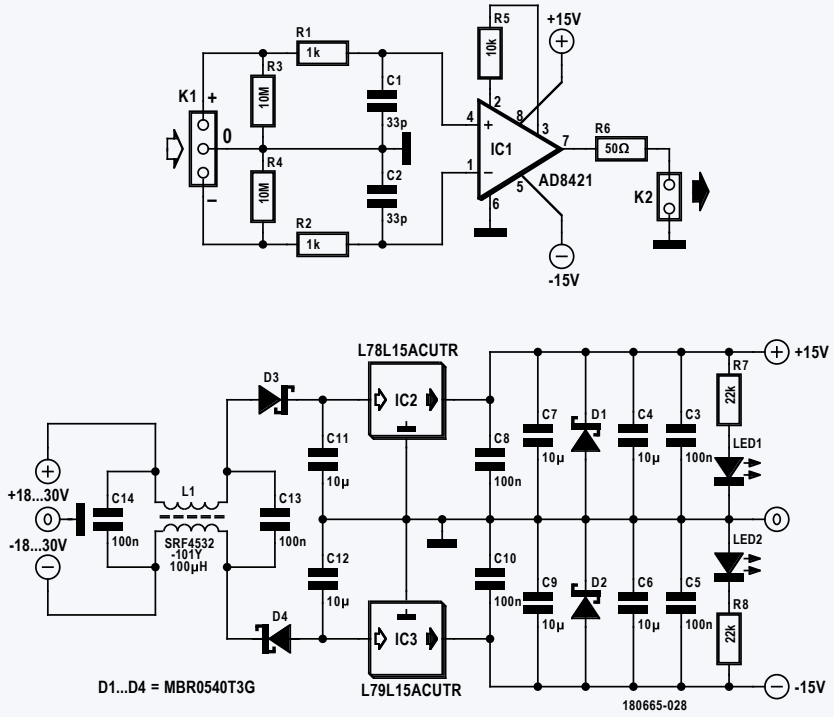


Figure 2: Complete circuit diagram of version 2.0 of the current probe. Note the close attention paid to the supply rails.

AD8421 TECHNICAL DATA

Supply voltage:
±2.5 to ±18 V

Current consumption:
2.3 mA max

Noise:
3.2 nV/√Hz at 1 kHz max;
200 fA/√Hz at 1 kHz max

Bandwidth:
10 MHz at gain = 1;
2 MHz at gain = 100

Common-mode rejection ratio (CMRR):
94 dB at DC, gain = 1;
80 dB at 20 kHz, gain = 1

Slew rate:
35 V/µs

Drift:
0.2 µV/°C

Input offset voltage:
1 ppm/°C in gain at nominal gain = 1

Input protection:
maximum 40 V
between '+IN' and '–VS' pins;
maximum 40 V
between '–IN' and '+VS' pins

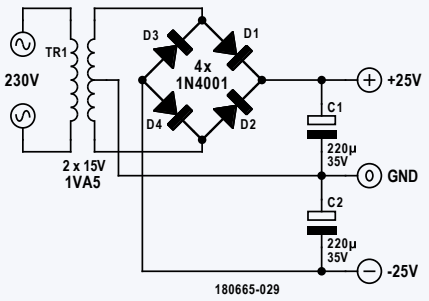


Figure 3: Circuit of a simple mains power supply using a transformer, full-wave rectifier and reservoir capacitors.

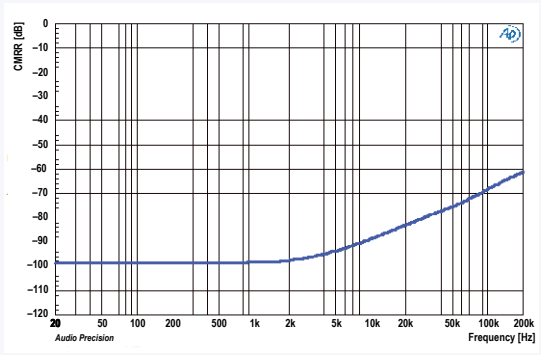


Figure 4: Measured common-mode rejection as a function of frequency.

Figure 5: The frequency response of the current probe extends to around 8 MHz.

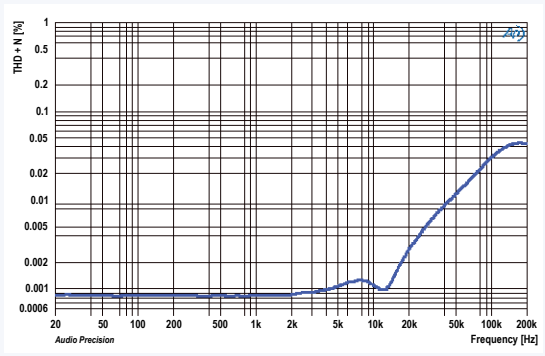


Figure 6: Distortion increases noticeably at around 10 kHz, but even at 200 kHz the level of distortion is very low.

ground potential so that the IC output delivers 0 V in this case. They also mean that the unit has a well-defined high input impedance. R6, a 50 Ω resistor, is required to match the output to a coaxial cable as the output of IC1 has a very low impedance.

Supply

The current draw of the circuit is low, so a 1.5 VA transformer with two 12 V or two 15 V secondary windings suffices, along with a full-wave rectifier and two 220 µF 35 V reservoir capacitors (**Figure 3**). Measurements indicate that the performance of the device was the same using this power supply as when using a bench supply. Because of the low load, the ripple on the power rails is also low, at only around 200 mV_{pp}.

The 15 V voltage regulators used in the design can accept input voltages from 17.5 V to 35 V. The use of Schottky diodes for D3 and D4 means that the current probe circuit needs an input voltage of at least ±18V. A small 12 V transformer is entirely adequate for TR1 as smaller transformers have a high impedance and, under low load, can output considerably more than their rated voltage. That means that, taking into account the 0.7 V forward voltage drop through D1 to D4, we can easily see up to ±20 V at the output of the power supply circuit. It is, however, necessary to measure this voltage to check. On the other hand, a small 18 V transformer can easily deliver up to ±35 V at the power supply output, which should be avoided if at all possible. A small 15 V transformer should deliver about ±26 V at the power supply output.

We carried out some tests to measure the behaviour of the circuit at reduced power supply voltages. We measured a 1 kHz signal with an amplitude of 3 V_{pp} that, because of the circuit's gain, produced 6 V_{pp} at the output of the current probe. The power supply was connected to a variable autotransformer and its input voltage gradually reduced. As long as the output of the power supply circuit remained above ±12 V no changes in the output were observed; below that, however, the signal at the output of the current probe started to clip. This is hardly surprising when, across D3 and D4 and the voltage regulators, there is a voltage drop of over 2 V and we need a little in reserve for the output circuit of IC1. The fact that no distortion was observed at just over ±12 V, even though at this point the two voltage regulators are no longer in regulation, is a tribute to the excellent power supply rejection ratio characteristic of the AD8421 (≥100 dB at 100 Hz).

Results

With a 1 kHz signal at 3 V_{pp} on each input the common mode rejection ratio is greater than 98 dB. From 2 kHz this steadily falls, reaching 61 dB at 200 kHz (**Figure 4**). This is better than expected. It should also be noted that the components of the input filters (R1 and C1, and R2 and C2), despite their tight tolerances, will not be exactly equal in value.

A single-ended measurement using a function generator on one input (with the other input connected to ground) revealed a real-life bandwidth of about 8 MHz (using the amplitude at 1 kHz as the reference). If R5 is used to set the gain to a low value there is a slight peak in the response in the region of the cutoff frequency, with a maximum reached at 7 MHz (**Figure 5**). The input filters have

Figure 7: A type RG316 coaxial cable with one end stripped and a plug crimped to its other end.

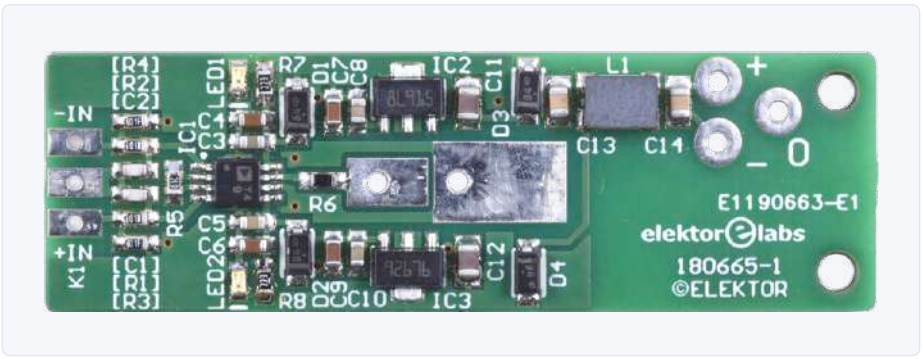


Figure 8: The fully-populated circuit board of the prototype. The use of 0603-size SMDs means that a steady soldering hand is needed.

a cutoff frequency of 4.8 MHz (ignoring the impedance of the source) which compensates for this peak. The peak is therefore responsible considering that the bandwidth of the current probe is higher than expected. At 8 MHz the phase of the output signal is already shifted by more than 180°. Sine waves in the megahertz range tend to appear more as triangle waves. Distortion steadily rises at frequencies over 10 kHz (**Figure 6**).

Cable

The output can be connected to an RG316 coaxial cable, which can be purchased by the metre. However, fitting a BNC plug to the other end of this cable requires the correct crimp tool. If this is not available, then an alternative is simply to buy a ready-made BNC coaxial cable, cut the plug from one end, strip back the insulation (**Figure 7**) and solder it directly to the circuit board. If you buy a longer BNC cable and cut it in the middle, you will have cables for two probe heads.

To make it easier to solder the RG316 cable directly, the circuit board has large pads in the middle (**Figure 8**) to which the central conductor and the screen can be soldered. The three pads at the top right can be used to solder the power supply connections. All five of these connections have holes into which 1.3 mm solder pins can be inserted as an alternative. If solder pins are used, they will probably need to be cut down to get the board to fit in the suggested enclosure. A cable tie through the 2.5 mm holes on the far right can be used to hold all the cables exiting the probe in place (**Figure 9**). Alternatively, hot-melt glue can be used for this purpose. It is a good idea to read the advice on construction below before making these connections. A 'socket' for the measurement connections is made by simply soldering a 3-pin section of SIL socket flat on the three pads on the left-hand side of the board (**Figure 8**). It is best to use the better-quality type with turned pins. **Figure 9** shows the fully populated board.

Construction advice

The components list suggests a suitable enclosure. First make an appropriately-sized rectangular slot on the left-hand side of the enclosure for the three-way SIL socket. Now place the populated and tested printed circuit board into the enclosure, feed the SIL socket into the slot from the left, hold it flat against the circuit board, and solder it down.

Now we come to the coaxial cable. We would recommend using RG316-type cable with a BNC plug appropriate for that type of cable. There are many different types of BNC plug for different types of cable (see the components list). The cable used in the prototype had an external diameter of 2.9 mm. If you are not familiar with assembling this type of plug, check on the manufacturer's website

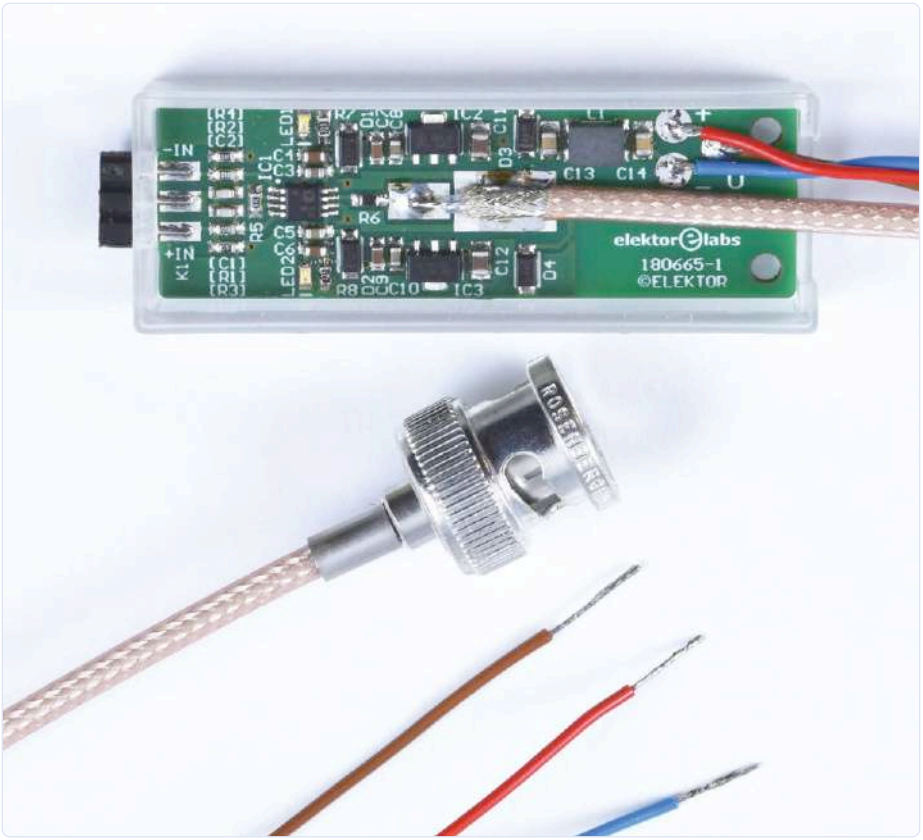


Figure 9: The prototype built into its enclosure with soldered cable and SIL socket.

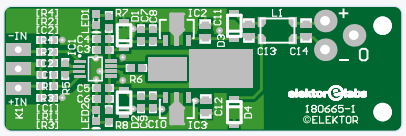
COMPONENT LIST

Resistors:
(all 1%, 100 mW, SMD 0603)
R1,R2 = 1 k
R3,R4 = 10 M
R5 = 10 k
R6 = 50 Ω
R7,R8 = 22 k

Capacitors:
C1,C2 = 33 p / 50 V, 1%, C0G/NP0, SMD 0603
C3,C5,C8,C10 = 100 n / 50 V, 10%, X7R, SMD 0603
C4,C6,C7,C9 = 10 µ / 25 V, 20%, X5R, SMD 0603
C11,C12 = 10 µ / 35 V, 10%, X5R, SMD 0805
C13,C14 = 100 n / 100 V, 10%, X7R, SMD 0805

Inductor:
L1 = 100 µH at 100 kHz, common-mode choke, 200 mA, SMD, e.g. SRF4532-101Y, Bourns

Semiconductors:
IC1 = AD8421ARMZ, MSOP8
IC2 = L78L15ACUTR, SOT89
IC3 = L79L15ACUTR, SOT89
LED1,LED2 = LED, red, SMD 0603
D1–D4 = MBR0540T3G, SOD123



Component side of the printed circuit board designed in the Elektor labs

Miscellaneous:
K1 = 3-way SIL socket, vertical mount, 0.1 inch pitch
3-way SIL socket as plug for test leads*
Type RG316 coaxial cable, 50 Ω, 2.9 mm diameter, 1 m, e.g. Conrad 606450*
BNC crimp plug for RG316, 50 Ω, e.g. Conrad 1490723*
USB1KL enclosure, 56x20x12 mm, e.g. Conrad 531276*
Alternative enclosure: TC-USB1 KL203, 56x20x12 mm
Printed circuit board 180665-1 v1.0*

* see text

MEASUREMENTS ON THE PROTOTYPE

Power supply voltage using the supply shown in Figure 3 with 12 V transformer: ±18.9 V. The current consumption on each rail was about 5 mA or 6 mA, mostly due to the voltage regulators and the LEDs.

Maximum values with a 1 kHz input signal at 4.8 V_{pp}:
U_{out}: 9.52 V with 100 kΩ load
THD+N: 0.1 %

Typical values with a 1 kHz input signal at 3 V_{pp}:
U_{out}: 5.97 V with oscilloscope input as load
THD+N: 0.00025% over 22 kHz bandwidth;
0.00083% bandwidth above 500 kHz

Common-mode rejection ratio with 3 V_{pp} on each input:
>98 dB at 20 Hz to 1 kHz;
>60 dB at 200 kHz

to see how to remove the correct amount of insulation before fitting it.
To solder the coaxial cable to the pads on the printed circuit board, first remove 12 mm of the jacket of the cable. Then cut the screening braid down to 6 mm and remove 4 mm of the insulation from the central conductor. There will now be a gap of 2 mm between this central conductor and the braid. Solder the central conductor first, and then the screen. For the latter, it is essential to work quickly. **Figure 9** shows how the soldered assembly should look.
Next, solder the three wires for the power supply. All that remains are the measurement input wires. Solder three thin wires to a three-pin SIL plug. For our prototype's 'test leads' we just soldered two lengths of enamelled copper wire to a SIL socket (**Figure 10**). Finally, the lid can be fitted to the enclosure and the unit is ready for use.

Making measurements

The probe should be powered separately from the circuit under test, ideally using the purpose-designed supply shown in **Figure 3**. When making measurements it is essential to connect the ground point of the input to the ground of the circuit under test. The two input measurement points can carry a common-mode voltage of up to ±12 V (and no more) relative to this ground. A note on the measurement cables: a turned-pin SIL socket has a typical mechanical life of a couple of hundred insertions, and so should last for a long time. If necessary the socket soldered to the circuit board can always be replaced easily, as its pins are soldered flat onto pads rather than using through-holes.
Instead of using the middle pin on this socket for the ground connection it is also possible simply to connect the pad marked '0' on the printed circuit board to the ground of the circuit under test. The high common-mode rejection ratio of the probe means that this

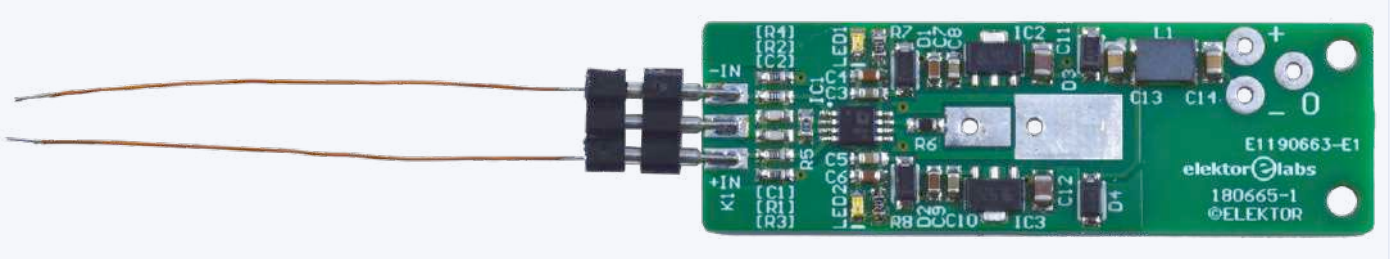


Figure 10: For our 'test leads' we used two lengths of thin enamelled copper wire soldered to a three-pin SIL socket. In this case the ground connection has to be made separately.

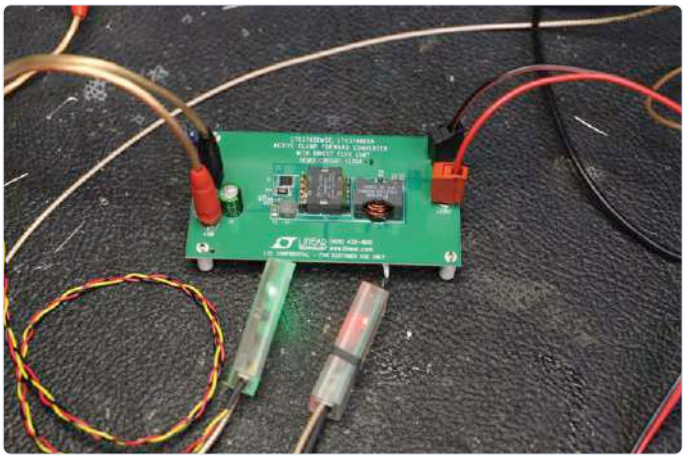


Figure 11: Here Alfred Rosenkränzer has set up two current probes to study a switching regulator circuit.

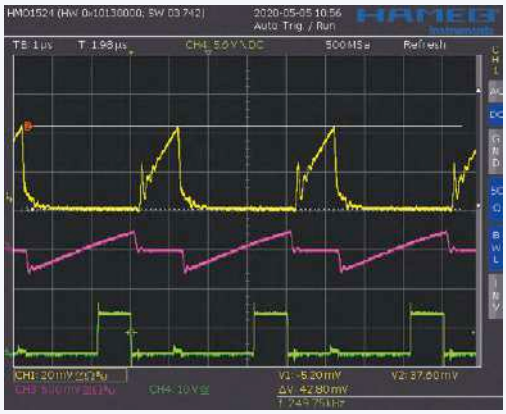


Figure 12: Version 2.0 of the current probe can also measure fast changes in current flow.

approach is unlikely to introduce interference or noise.

If multiple probes are powered from the same power supply then only one ground connection should be made to the circuit under test as, otherwise, an undesirable earth loop might be formed.

It is also important to note that, if the circuit under test is powered from a supply of more than 12 V, you should check that the voltages on either end of the shunt resistor are within the allowable ±12 V common-mode voltage range with a multimeter before attaching the probe.

Figure 11 shows the use of two probes to study a switching regulator circuit. **Figure 12** shows that, with this current probe, it is possible to capture very fast changes in current flow.

And, last but not least, the design files for the printed circuit board are available to download for free from the Elektor web page accompanying this article [3].

180665-02

Questions or Comments?
If you have any questions regarding this project, feel free to contact the author by e-mail at alfred_rosenkraenzer@gmx.de.

RELATED PRODUCTS

- **PeakTech 4350 current clamp meter**
  www.elektor.com/peaktech-4350-clamp-meter
- **Current transformer for oscilloscopes**
  www.elektor.com/potentialfreie-strommessung-fur-oszilloskope
- **OWON SDS1102 two-channel DSO (100 MHz)**
  www.elektor.com/owon-sds1102-2-ch-digital-oscilloscope-100-mhz

Contributors

Idea and circuit: **Alfred Rosenkränzer**
Revisions and text: **Ton Giesberts**
Circuit diagrams: **Patrick Wielders**
Translation: **Mark Owen**
Editing: **Stuart Cording**
Layout: **Giel Dols**

WEB LINKS

[1] **"A Simple Current Sensor Probe", A. Rosenkränzer,** Elektor September/October 2016: www.elektormagazine.com/magazine/elektor-201609/39807

[2] **AD8421 data sheet**: www.analog.com/media/en/technical-documentation/data-sheets/AD8421.pdf

[3] **Elektor web page for this article**: www.elektormagazine.com/180665-02

[4] **Labs project page**: www.elektormagazine.com/labs/differential-current-probe-180665

The Hewlett-Packard Interface Loop

Connect the world!

Blue Globe by Torty3, Wikimedia Commons under Creative Commons Attribution-Share Alike 3.0 Unported (https://commons.wikimedia.org/wiki/File:Blue_globe_icon.svg).

By **Dipl.-Inf. Karl-Ludwig Butte** (Germany)

In the present era of the Internet of Things, we expect electronic devices to be small, portable, battery powered, and – especially important – interconnected through a network, so they can automatically and independently monitor our environment with their sensors and take action with their actuators to make adjustments where necessary. But is this actually something that only arose after Mark Weiser proposed his vision of ubiquitous computing in a widely read article in the September 1991 issue of *Scientific American* [1]? The answer is no, and if you would like to know why, I invite you to take a trip in time with me back to the world of the Hewlett-Packard Interface Loop in 1981.

The most remarkable feature of the Hewlett-Packard HP-41C handheld computer, launched at the start of July 1979 [2], was the four ports for peripheral devices such as a barcode reader wand, a thermal printer, a magnetic card reader, or RAM and ROM extension modules. There was nothing else like it. Users quickly learned to appreciate this and started asking for many more options to connect the HP-41C to external electronic devices.

Hewlett-Packard had already accumulated relevant experience in interconnecting and controlling their test and measurement equipment over the Hewlett-Packard Interface Bus (HP-IB), which was formalised as the IEEE-488 industry standard. However, the HP-IB 8-bit parallel bus, with its thick cables containing at minimum 18 conductors and bulky 24-pin Amphenol plugs, was hardly compatible with the small battery-powered devices of the HP-41C family. To reconcile these two highly different worlds, the engineers at HP decided to serialise the data traffic. Instead of transmitting eight bits in parallel, the bits were sent sequentially, one at a time, and the hardware handshake was replaced by a software handshake. These measures allowed the cable to be reduced to a bare minimum of two conductors.

But there was still one detail needed to make the new interconnection system as reliable and robust as customers expected from legendary Hewlett-Packard quality. The HP engineers opted for a loop topology instead of a bus topology. This meant the cable ran from the loop controller (e.g. an HP-41C) to the first periph-

eral device, from there to the next peripheral device, and so on until the last peripheral device, after which it went back at the loop controller. This allowed the loop controller to directly check whether the number of received bits exactly matched the number of transmitted bits, so that any transmission errors could be corrected immediately.

On 14 December 1981 [3], Hewlett-Packard formally launched the Hewlett-Packard Interface Loop (HP-IL) for the HP-41C in the form of the 82160A HP-IL module (**Figure 1**). It simply plugged into one of the four previously mentioned ports of the HP-41C for peripheral devices. The HP-82161A HP-IL digital cassette drive and the HP-82162A HP-IL thermal printer (**Figure 2**) were launched at the same time as the HP-IL module, exceeding even the most optimistic expectations of HP customers. This clever system created shock and awe amongst HP's competitors, who had nothing comparable to offer. The number of peripheral devices grew in the course of time, as well as the number of computers for which an HP-IL module was available. For example, the HP-71B BASIC handheld computer [4] and the HP-75C/D also supported the HP Interface Loop, allowing them to use the same peripheral devices as the HP-41C. Suitable interface cards were available for the series 80 desktop devices (HP-85, 86 and 87) as well as the HP-110 and HP-150. In 1986 the HP-82973A HP-IL interface was launched as an ISA card for IBM PCs and compatible computers, and the HP-82166C HP-IL Development Kit enabled other companies to design their own devices compatible with HP-IL.

Figure 1: The HP-82160A HP Interface Loop module.

Figure 2: An HP-IL setup consisting of an HP-41C, a digital cassette drive and a thermal printer.

Peripheral devices from HP

The cassette drive was not just a standard audio cassette recorder as used by many manufacturers of home computers. Instead, HP designed a precision drive that perfectly matched the HP-41C system in terms of function, size and enclosure design (**Figure 3**). The cassettes were similar to those used in mini dictating devices. **Figure 4** shows a mini cassette in comparison to a standard compact audio cassette. The area to the left of the cassette compartment on the drive provided storage space for two cassettes (Figure 3). The cassette capacity was 131,072 bytes, roughly the same as a 5¼-inch diskette on a Commodore PET 2001 computer [5], which could store 176,640 bytes.

The HP-82162A thermal printer looked nearly the same as its older cousin HP-82143A, which could be connected directly to one of the four I/O ports of the HP-41C. The HP-IL thermal printer, however, came with a feature long desired by users: it could print HP-41C barcodes. The HP-82153A barcode reader, which also could be connected directly to one of the four I/O ports of the HP-41C, allowed programs and data to be read from books and magazines into the HP-41C much faster than tediously typing them in.

As previously mentioned, however, these two devices were only the start. **Table 1** shows an overview of the HP-IL compatible devices offered by Hewlett-Packard. Everything an HP enthusiast might want was available: data storage devices, printers, a wide variety of interfaces, and even acoustic couplers. However, the key was the test and measurement instruments. With a system consisting of an HP-3468 multimeter, an HP-82182A time module and an HP-41C, measurement sessions extending over a prolonged period could be fully automated. If a printer and/or a cassette drive was added to the interface loop, the measurement data could also be logged and stored. And with the HP-82168A acoustic coupler the data could be sent by phone to a central computer for further

Figure 3: The HP-82161A digital cassette drive.

Figure 4: A cassette for the digital cassette drive in comparison to a standard audio cassette.

Figure 5: Cover image of the Hewlett-Packard Journal, January 1983.

Table 1: Overview of HL-IL compatible peripheral devices available from Hewlett-Packard [6].

Device	Designation
HP 9114A/B	Diskette drive
HP 82161A	Digital cassette drive
HP 82162A	Thermal printer (HP-IL version of HP 82143A printer for HP-41C)
HP 82163A	Video interface
HP 92198A	80-column video interface (Mountain Computer)
HP 82168A, 92205M	Acoustic couplers (modems)
HP 82905A/B	Printers
HP 2225B	ThinkJet printer
HP 7470A	Graphics plotter
HP 2671A/G	Alphanumeric, graphic thermal printer
HP 1630 and HP1631	Logic analysers (can be loop controllers)
HP 3421A	Data acquisition & control unit
HP 3468A/B	Digital multimeter (HP-IL version of the HP 3478A with GPIB)
HP 5384A und HP 5385A	Frequency counters with option 003 (HP-IL)
HP 82164A	HP-IL/RS-232C interface (serial)
HP 82165A	HP-IL/GPIO interface (generic parallel output)
HP 82166A	HP-IL converter (smaller version of the GPIO interface for embedded applications)
HP 82166C	HP-IL Converter Prototyping Kit
HP 82169A	HP-IL/HPIB interface
HP 82985A	HP-IL/NTSC video

processing. From a purely functional perspective, many of today's IoT applications are entirely the same as this setup with the HP Interface Loop.

Mode of operation

The Hewlett-Packard Interface Loop joined the connected devices in a ring topology comparable to the token ring technology developed at Cambridge University (UK) in 1974 and then enhanced and marketed by IBM in the mid-1980s. The ring nature of these networks is illustrated very nicely on the cover of the January 1983 issue of the *Hewlett-Packard Journal* (**Figure 5**), in which the Interface Loop was described in detail.

Every HL-IL system was managed by a loop controller and could interconnect up to 30 devices. Each device was automatically assigned a device address by the loop controller. This worked the same way as automatic IP address assignment by DHCP in an Ethernet network. The loop controller ensured strict compliance with the loop protocol, facilitated by a large set of command groups. The loop controller could issue reset messages for the loop or individual devices to ensure clearly defined initial conditions. Another message group allowed the loop controller to determine the active talker or listener, enabling devices to communicate directly with each other. The command groups also included status queries, data exchange messages, and other messages for administration and synchronisation. The response of a particular device to a specific message could vary widely and was determined by the designer of the peripheral device. In response to a device clear (DCL) message, for example, the thermal printer would empty its internal print buffer, move the print head to the left side and advance the paper by one line. By contrast, the video interface would respond by clearing its internal video memory and positioning the cursor at the top left corner of the screen. To help developers integrate an HP-IL interface in their own device designs, Hewlett-Packard offered the HP-82166C HP-IL Interface Kit.

The HP-IL Interface Kit

HP strongly encouraged the hardware designer community to

develop devices with an HP-IL interface and published detailed technical descriptions for this purpose. They also brought out an interface kit (**Figure 6**). The special components included in the kit are shown in **Figure 7**. Along with three ASICs, the kit contained two connector modules for the special plugs of the loop connection cable, small isolation transformers in DIL packages for galvanic isolation of the devices, and a plug-in module for the HP-41C for software debugging.

HP-IL lives!

It is no longer possible to determine exactly when Hewlett-Packard announced the demise of the HP Interface Loop. In any case, the HP-42S, launched in 1988 as the successor to the HP-41C, did not have any I/O ports, let alone an HP-IL interface. However, the flexibility, versatility and technical superiority of the HP Interface Loop quickly won the hearts of designers worldwide, and this enthusiasm is still alive. There are a number of community projects that combine the HP-IL with today's technology. One of the best known projects is the PIL Box from Jean-Francois Garnier [7]. **Figure 8** shows the circuit board of the PIL Box, which implements an HL-IL to USB interface. Everyone in the Elektor community can easily imagine the new and unimagined possibilities this opens up.
As so often in those days, Hewlett-Packard was miles ahead of the state of the art with the HP Interface Loop. Thanks to the special attention given to development, production quality and detailed documentation, we can still have a lot of fun with this technology many years after the associated products were discontinued.

200329-02

Figure 6: The HP-82166C HP-IL Interface Kit.

Figure 7: The special components of the HP-IL Interface Kit.

Figure 8: The PIL Box interface (printed with the kind permission of the copyright holder J.-F. Garnier).

WEB LINKS

[1] **Weiser, Mark: "The Computer for the 21st Century". Scientific American, September 1991.**: https://www.lri.fr/~mbl/Stanford/CS477/papers/Weiser-SciAm.pdf

[2] **Hewlett-Packard HP-41C handheld computer**: https://www.finseth.com/hpdata/hp41c.php

[3] **Hewlett-Packard handheld computer with launch date**: http://www.vcalc.net/hp-date.htm

[4] **Butte, Karl-Ludwig: "Hewlett-Packard HP-71B Number Cruncher (1984)", Elektor 7/2014, p. 122**: www.elektormagazine.com/magazine/elektor-201407/26923

[5] **Butte, Karl-Ludwig: "Happy 40th Birthday, PET!", Elektor 5/2017, p. 90**: www.elektormagazine.com/magazine/elektor-201705/40337

[6] **Source Wikipedia**: https://en.wikipedia.org/wiki/HP-IL

[7] **HP-IL resource page**: http://www.jeffcalc.hp41.eu/hpil/

The Future of Machine Learning

An Interview with Daniel Situnayake

By **C. J. Abate** (United States)

Daniel Situnayake is a creative engineer who believes embedded machine learning (ML) is a "once-in-a-generation technology." Here he talks about ML's potential and introduces TinyML, as well as a few ideal applications. We also touch on his experience as a developer at Google.

Figure 1: Daniel Situnayake (Founding TinyML Engineer, Edge Impulse).

Transitioning from Academia

Abate: *Let's start with your background. When did you first become interested in embedded ML? Was it at Birmingham City University (BCU), where you earned a BSc in computer networks and security? Or was it after you left university?*

Situnayake: I didn't set out to work on embedded machine learning. In fact, I've taken a long and winding road on my way here, with many seemingly disparate experiences nudging me towards this field! After graduating from BCU, I was asked to stay on as a faculty member, teaching classes and consulting on behalf of the university. I focused on the topic of Automatic Identification and Data Capture technologies, which encompasses all the ways a computer can recognize and process physical objects — everything from biometrics, like facial recognition, through to RFID, barcoding, and smart cards. This was my first introduction to two things: computer vision algorithms, which at the time were not based on machine learning, and embedded systems.

A few jobs later in my career, after an acquisition, I found myself working on the nascent data science team at a consumer banking company. This was where I first encountered data science tools and workflows, and where I learned about working with large datasets. It wasn't a topic I'd ever been exposed to before, but I found it fascinating.

When we started Tiny Farms, I saw my first opportunity to join these fields together, using data science tools to interpret and react to embedded sensor data in an agricultural technology setting. But it was at Google, meeting Pete Warden (**Figure 2**) and Andy Selle from the TensorFlow Lite team, when I really grasped the potential of deep learning on tiny devices and it became my number one focus.

Abate: *When did you move to California? Was it for a specific work-related opportunity?*

Situnayake: I moved to California in 2009, the year after the global economy imploded. I'd spent the past few years taking a battery of mental and physical tests with the goal of being selected as a British Army officer cadet. I wanted to fly Apache helicopters! I was due to start officer training in May 2009, so I went on a farewell trip across the US with my best friend.

When we stopped in California, I met a girl. A couple of months later, and a few weeks into the Commissioning Course at Sandhurst, I decided that the life of a gentleman pilot was not for me — and that I'd be better suited to a laid-back life in California. I booked a flight, moved in with the girl I'd just met, and we got married six months later!

The marriage didn't work out, but I've been here ever since. Making my transition from academia, I joined an AI startup in Los Angeles

and then moved up to the Bay Area just in time for the 2010s tech boom. It's been a crazy ride.

Insect Farming Tech

Abate: *You co-founded Tiny Farms, which you describe on LinkedIn as "America's first insect farming technology company." How did that company come about? And what is the status of the company?*

Situnayake: Like all my best stories, this one also starts with a trip across the US. Two close friends of mine had been thinking about starting a project to address food security. During a cross-country trip, they stayed in a cabin by a lake that was surrounded by big, noisy grasshoppers. On a whim, they wondered if they might be edible. After a bit of research, they learned that edible insects were actually extremely important to global food security — but that techniques for raising them in captivity had barely been explored.

After some initial experimentation, they asked if I'd like to join them in starting a company. I saw a lot of potential for applying the techniques I'd learned around Automatic Identification and Data Capture to automating the rearing of insects, and in using data science to guide the iterative development of novel systems.

I jumped in with both feet! At first, it was a part-time project, but we eventually raised money and built an industrial scale insect farm in San Leandro, California. It was intense, exciting, and more than a little stressful trying to keep millions of crickets alive while dealing with customers, investors, and research. Around five years in, I decided it was time to throw in the towel. My co-founders carried on for a couple of years, but in the end, it was too difficult to reach product-market fit, and they called it a day. Tiny Farms lives on as an open-source repository of the designs and data we accumulated during seven years of operation [1].

Figure 2: Daniel Situnayke and Pete Warden sign their TinyML book.

Working at Google

Abate: *You spent almost three years with Google. Before we get into the details about your positions there, can you tell us a bit about what it was like joining such an impressive company?*

Situnayake: After five years building a farming company — which involved more time spent on construction work than it did on coding — I wanted to get back in at the deep end, and there was no company I admired technically more than Google.

After my first few days, I was blown away. I'd never worked at a large tech company before. With your Google badge, you can access facilities at any Google office, anywhere in the world, so it felt like I had become a citizen of a strange new type of country, with its own public services, infrastructure, and government.
Google has an amazing culture of freedom: you can pretty much work on whatever you want, as long as it makes an impact on your product. If you have a cool idea, you're encouraged to pursue it, and to bring other folks on board to help. This leads to a huge number of exciting projects, but it can also be a bit overwhelming: with all this choice, how do you decide where to apply yourself?

> Google has an amazing culture of freedom: you can pretty much work on whatever you want, as long as it makes an impact on your product.

This is where things get a bit tricky. Because there is always so much going on, there are potentially endless meetings, distractions, and things to think about. The first year or so of working at Google is about developing your inner filter so you don't feel overwhelmed by all the noise.

It's a bit of a myth that everyone working at Google is some kind of genius. There's definitely the same range of abilities inside of Google as in the rest of the tech industry. I've never been big on hero worship, and it's a nice feeling when you meet some of the top folks in your field and realize that they're normal people, just like you.

Abate: *You worked at Google as a Developer Advocate for TensorFlow Lite. What is TensorFlow Lite? And how did you educate developers? Through meetups? Online courses?*

Situnayake: TensorFlow is the name for Google's ecosystem of open-source tools for training, evaluating, and deploying deep learning models. It includes everything from the high-level Python code used to define model architectures all the way down to the low-level code used to execute those models on different types of processors.

TensorFlow Lite is the subset of TensorFlow tools that deal with deploying models to so-called "edge devices," which includes every-

thing smaller than a personal computer: phones, microcontrollers, and more. The TensorFlow Lite tools can take a deep learning model and optimize it so that it is better suited to run on these types of devices. It also includes the highly optimized code that runs on the devices themselves.

My favorite part of TensorFlow Lite is the stripped-down, super-fast variant that is designed to run deep learning models on tiny, cheap, low-power microcontrollers. Prior to the launch of TensorFlow Lite for Microcontrollers, developers had to write their own low-level code for running machine learning models on embedded devices. This made the barrier to entry for on-device machine learning incredibly high. TensorFlow's entry into this field knocked down a lot of walls, so suddenly anyone with some embedded experience could run models. It was very exciting to see the field of TinyML spring to life.

My role was to help the TensorFlow Lite team understand and connect with the developer ecosystem. All developers are different, so it's important to create multiple ways for them to engage. We did everything you can imagine, from in-person meetups and Google-funded conferences through to building easy to understand example code and writing the TinyML book!

Abate: *Why did you end up leaving Google? Was it a hard decision?*

Situnayake: Working with TensorFlow developers, I got to see that even the best machine learning tools and libraries require a huge amount of education to use. Despite some excellent courses and content, the required time investment puts machine learning out of reach for a great number of busy engineers. Training models is as much an art as a science, and the instincts and best practices take years to learn. Things are even harder with TinyML, because it's such a new field, and the constraints are not well known.

When I heard about Edge Impulse, the company I left Google to join, I was blown away. Even before the official launch, the founders had managed to build a product that allowed any developer to train deep learning models that would run on embedded hardware. I could tell from my work with the developer community that they had cracked some of the toughest problems in the field, abstracting away the pain points that make machine learning tools difficult to work with.

I believe that embedded machine learning is a once-in-a-generation technology. It's going to transform the world around us in ways that we have yet to realize. Edge Impulse was founded to get this technology into the hands of every engineer, so they can use their diverse perspectives and deep domain knowledge to solve complex problems in every part of our world.

As soon as I realized this potential, I decided to jump on board. It was sad leaving my friends at Google, but TinyML is a small world, and I still work closely with many of them!

Tiny Machine Learning (TinyML)

Abate: *What should pro engineers and serious makers know about TinyML? And can you share an example or two of an ultra-low-power machine learning application?*

Situnayake: Our world is crowded with billions of sensors, in all sorts of places, from home appliances to smart factories and the vehicles we drive. All of this sensor data represents a golden opportunity for developers to build devices that understand the world around them and make intelligent decisions that help people out.

Historically, the only way to make use of this data was to send it across the Internet to a powerful computer able to crunch the numbers, interpret the data, and trigger actions based on the results. This works just fine for some applications. For example, it's OK if data from a weather station takes a few seconds to get to a server. But there are many applications where this won't work.

One of my favorite examples of this is the camera trap. Scientists use these to monitor animals in the wild, often in remote places that have no Internet connection available. When an animal walks by, it triggers the camera trap's motion sensor, and a photograph is taken. Researchers will stop by periodically and pick up a memory card filled with animal photos.

The problem is, the motion sensor can be triggered by all sorts of things, like leaves moving in the wind, or species the researchers aren't interested in. It takes a long time to sift through all the photos, and the memory card gets filled up quickly, meaning people have to frequently go out and change it. Even if an Internet connection were available, the extra power drain would mean that the camera's batteries would run out and need to be changed.

By adding a TinyML model that is trained to recognize the correct type of animal, the trap can avoid taking photos of leaves moving in the wind. This means traps can be left out longer, which saves time and money for researchers.

There are four main constraints that make an application ideal for TinyML. If an application has limited connectivity, low tolerance for latency, a limited source of power, or a requirement for strong privacy, TinyML can potentially help, by avoiding the need to send data from the device.

It's worth noting that only certain problems can be solved by machine learning. They need to be problems where a bit of fuzziness is OK, since ML can rarely give absolute answers. It's perfect for taking animal photos, where the occasional mistake is acceptable, but it might not be reliable enough in some safety-critical applications.

Abate: *You co-authored the book,* TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers. *Since you cover Arduino, it leads me to believe that TinyML technology is accessible to non-experts and pro engineers alike. Besides reading your book, how can engineers and innovators who are unfamiliar with TinyML get started?*

Situnayake: As always, the best way to learn is to build some projects! A good place to start is by following some of the tutorials we provide in the Edge Impulse documentation: motion recognition with an accelerometer [2]; recognizing sounds from audio [3]; and adding sight to your sensors [4].

We've built Edge Impulse so you can use your mobile phone to collect data and test out models, so you can get started without needing any special hardware. Once you're familiar with the machine learning flow, you can grab your favorite dev board and start hacking!

I'd recommend the Arduino Nano 33 BLE Sense as a good starter board for TinyML. It has a speedy but low-power Cortex M-4F processor, and enough RAM for some interesting models. It's fully supported in Edge Impulse.

Edge Impluse

Abate: *You joined San Jose-based Edge Impulse in January 2020. How did that opportunity come about?*

Situnayake: I met the CEO, Zach Shelby, during my time at Google. He was previously a VP at ARM, who work very closely with the TensorFlow Lite team. When Zach announced he was leaving ARM to found a TinyML company, it caught my attention, and when they posted their first job, I had to apply!

Abate: *What is your workday like? Are you coding all day? Educating? Business development?*

Situnayake: One of the things I'm enjoying the most about life at Edge Impulse is I'm once again primarily an engineer. I spend most of my time hacking. I do still get to meet with customers, which is pretty rewarding, and crucial for knowing how to focus my engineering time. I'm also very active in the TinyML community, so I regularly give talks and help organize TinyML meetups. That said, I usually have a couple of meetings in the morning, then I can get my head down and write some code for the rest of the day. It's great!

Abate: *I see that the Edge Impulse team started COVID-19-related project in May — "Cough Detection with TinyML on Arduino"* (**Figure 3**)*. The project has been viewed more than 5,400 times. How has the response been from the engineering community? What is the status of the project?*

Situnayake: Even though Edge Impulse has only been around a few months, we already have a fantastic community from a diverse set of backgrounds. When COVID-19 hit, one of our community members — Kartik Thakore, a data scientist and biomedical engineer — was interested in building some projects to help. We thought this would be a great opportunity to showcase what can be done with TinyML, so we sent him some hardware and he put together a project to detect coughs using a TinyML model. The result was this project [5], which we hope inspires makers to tackle similar challenges!

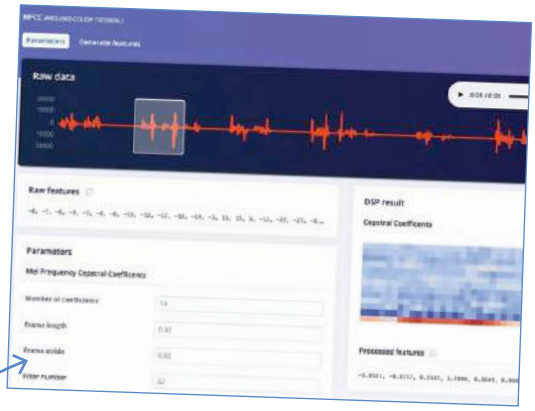


Figure 3: Cough detection with TinyML on Arduino (Source: Edge Impulse).

TinyML Projects

Abate: *Any other TinyML projects you want to highlight for the Elektor community?*

Situnayake: At Edge Impulse, we're strong believers in technology as a force for good, and some of my favorite projects are in that category. For example, Smart Parks are training TinyML models to detect wild elephants, so that villagers can be warned when they're headed towards town.

Another favorite category of mine is people enabling computers to do crazy things you never would have thought. For example, Benjamin Cabé created this amazing project that uses a gas sensor to recognize different brands of alcoholic spirits! If you'd like to reproduce it, there's an awesome tutorial from Seeed Studio.

200423-01

Questions or Comments?
Do you have questions or comments about his article?
Contact Elektor at editor@elektor.com.

Contributors
Interviewer: **C. J. Abate** Layout: **Giel Dols**

WEB LINKS

[1] **Open Tiny Farms**: www.opentinyfarms.com/
[2] **Edge Impulse, "Continuous Motion Recognition"**: https://docs.edgeimpulse.com/docs/continuous-motion-recognition
[3] **Edge Impulse, "Recognize Sounds from Audio"**: https://docs.edgeimpulse.com/docs/audio-classification
[4] **Edge Impulse, "Adding Sight to Your Sensors"**: https://docs.edgeimpulse.com/docs/image-classification
[5] **Edge Impulse, "Cough Detection with TinyML on Arduino"**: http://bit.ly/edge-cough-detection-project

Smartphone Apps for Android and iOS

Programming Apps within a Single Development Environment

By Veikko Krypczyk (Germany)

Increasingly these days, a smartphone app is required in order to interact with electronic products, such as Internet of Things (IoT) devices. Usually, both Android and iOS have to be supported. A multitude of different approaches are vying for the app developer's attention. Cross-platform approaches are required as well as an intuitive approach to creating the user interface. The development environment RAD Studio – better known as Delphi – can be a suitable approach.

More and more applications in the field of electronics rely on the sophisticated interaction of hardware and software. Apps for mobile devices, such as tablets and smartphones upon which the operating systems Android and iOS run, are increasingly being used. The challenge is to create an individual app for both systems that is exactly tailored to the requirements of your project. If you are not already involved in the development of mobile apps, getting started can be challenging. A variety of approaches compete with each other and the learning curves are often steep. Because of this, programming for even the simplest of devices can be a big deal, perhaps even the stumbling block for the entire project. This can be remedied by using a cross-platform approach that offers a largely intuitive approach to user interface design.

In this article we provide an overview of the available options when developing mobile apps before taking a closer look at a particularly interesting approach. With the help of RAD Studio - also known to many developers as Delphi, an efficient tool for the development of desktop applications - apps for Android and iOS can now be created from a single source code base with significantly reduced effort. Programming is undertaken in Delphi (object Pascal) and the comprehensive support of a graphic designer. An example project will show how it all works.

Apps for mobile devices

From a technical perspective, the following types of apps can be identified:

- Native apps
- Web apps
- Hybrid apps

Native apps are created exclusively for Android or iOS using the APIs of these

systems directly. Thus the resultant user interface fits optimally to the platform. A further advantage is that they do not suffer any restrictions when accessing the smartphone's or tablet's device-specific hardware. All the sensors of such devices can be addressed directly. The app itself is made available via the app stores and, if a native app is installed, it can also be run without Internet access (offline), if this makes sense. Any required data synchronisation can take place automatically when the device next goes online. Apps for iOS (Apple) are created using Xcode (development environment) and Swift (programming language). For Android, Android Studio and Kotlin or Java are used.

Web apps, on the other hand, are web applications geared to mobile devices. This primarily concerns the design of the user interface. Access to the system hardware is, as a result, restricted but some basic functions, such as positioning via GPS, are possible. These can be programmed using JavaScript APIs. However, it is not possible to distribute the apps via each platform's stores. They run on a server and therefore require a constantly-available Internet connection. An icon can be set up on the home screen for such apps so that users can quickly access them. The lack of offline functionality can be partially eliminated by using Progressive Web App (PWA) technology. A PWA is a symbiosis of a website and an app. By means of a ***service worker*** a caching function can be implemented. This service worker lies between the web server and the app on the mobile device. In order to create a web application for mobile devices, the entire range of web technologies are available such as common libraries and frameworks. Ultimately, such apps are based on HTML (structure), CSS (design) and JavaScript (interaction, logic), as a mobile device's browser can only interpret these languages.

Hybrid apps are based on web technologies. The web app is executed in an embedded web browser, i.e. it runs in a kind of 'sandbox,' which makes it platform independent. The system therefore believes it is executing a native app (the browser). There are several approaches for developing hybrid apps, such as Cordova from the Apache Software Foundation and the principle of operation is always very similar. The app opens a browser window in full screen mode when started. This ensures that the browser is not identifiable as such and the web address cannot be changed. The web app, created in HTML, CSS and JavaScript, is displayed in the browser. The framework also gives the app access to system functions such as the camera or address book. This is all done with the help of plug-ins.

Figure 1: The various approaches to app development.

Both web apps and hybrid apps are designed to operate on both Android and iOS. However, they have some disadvantages, such as poor performance and limited hardware access compared to native apps. Therefore, developers try to link both app approaches using cross-platform approaches. The goal is to come as close as possible to the native model (also in the appearance of the user interface) and to create the app for Android and iOS from a common source code base. There are a large number of cross-platform approaches that are very different in their approach, including Xamarin, RAD Studio, NativeScript and React Native. The above aspects are summarised in **Figure 1**.

In this article we will take a closer look at the development process using RAD Studio (Delphi). This approach seems to be especially well suited to electronic and IoT projects. The process will be familiar to many developers. With the help of a graphical designer, the user interface can be created in a manner similar to the development of applications for the Windows desktop. The logic is written in the language Delphi (Object Pascal), which is also intuitive and not particularly complex. Some electronics engineers will also be familiar with Pascal, perhaps from experiences programming embedded software for microcontrollers.

Installation and system configuration

We start the process by installing Delphi on Microsoft Windows and set it up for mobile development. After registration, the Community Edition can be downloaded [1]

RAD STUDIO, DELPHI AND C++Builder

Delphi and C++Builder are Embarcadero's integrated development environments for creating applications for different operating systems. Delphi uses the Delphi programming language of the same name (a further development of Object Pascal), while C++Builder uses C++. Both environments are combined in RAD Studio. There are several editions available. For professional use, there are the commercial products Professional, Enterprise and Architect. For limited commercial use, testing, learning, and for private and open source projects there is the free Community Edition, as was used here. The current version is 10.4 of RAD Studio.

Figure 2: Selection of the target platform(s).

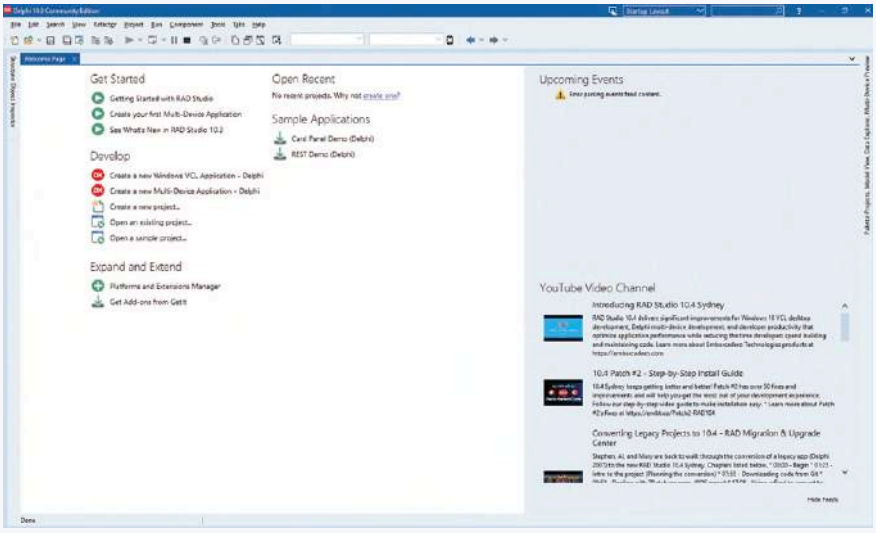


Figure 3: Start screen for the Delphi development environment.

and the installer started. During installation, a dialogue box appears with a platform selection option (**Figure 2**).

Select Android and iOS in addition to Windows. Next, you are prompted to select additional options. You should do this for the Android SDK/NDK. After installation, start the development environment (**Figure 3**).

For mobile development, some rework is necessary. If you want to test the app on iOS, you need access to an Apple PC with Xcode via a network. A simulator for iOS can only be run under macOS. Even a physical iPhone will need to be directly coupled with the Mac PC. Via the network the Mac PC is then accessed from Windows through RAD Studio (**Figure 4**).

To do this, the PAServer software must be installed on the Mac to enable remote access. If you are using a Mac PC, you can run Windows in a virtual machine (Parallels, Virtual Box, etc.) and install RAD Studio in that virtual machine. From this virtual machine, the Mac PC can then be accessed, i.e. all components for development are installed on one computer. The need for a Mac PC to build apps for iOS is not unique to RAD Studio and applies to all other cross-platform development tools.

You can create apps for Android directly from Windows. For testing you can connect a smartphone/tablet to your computer via USB or (alternatively) install an emulator. However, a 'real' device is much faster. Connect your smartphone to the computer with the USB cable and configure it appropriately, i.e. enable the developer permissions including USB debugging on the smartphone under the settings. Now check in the device manager whether your mobile device has been correctly recognised and configured. For the official documentation on setting up Delphi for mobile development, refer to [3] and [4]. This completes the configuration work. A first test ('*Hello World*') illustrates the procedure for developing a mobile app.

Hello Elektor

Now we can focus on creating a first project. In the Delphi development environment, select *File | New | Multi-Device Application - Delphi* from the main menu and click *Blank Application* (**Figure 5**) in the window, followed by OK.

Save the project. Using the tool bar we can create the user interface using visual controls. To do this, drag a *TLabel* type component onto the dialogue box for a test. You can also switch the preview between the differ-

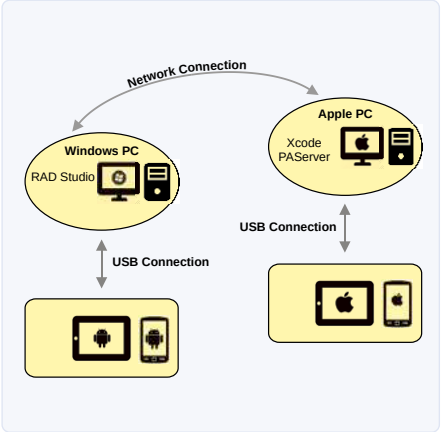


Figure 4: System configuration for developing iOS apps.

NO iOS WITHOUT macOS AND Xcode

iOS is a closed system, meaning Xcode (integrated development environment) and thus a Mac PC with macOS is required to create the app packages and distribute them to the mobile devices. The iOS simulator can also only be operated under macOS. Alternatively, you can use a cloud service by "renting" a hosted Mac PC with all the necessary development tools and control it via a remote connection. The service *MacInCloud* offers such a solution [2]. Its screen is then displayed on the developer's own computer. This solves the problem, allowing the iOS simulator to be used during programming and the app package to be created when complete. A real device can, of course, not be used via the cloud. The author has had a positive experience with the above-mentioned service. The PAServer software for remote control of the Mac is already pre-installed, so you can start right away after registration and login. To get started there is a flexible pay-as-you-go plan so you only need to pay for the time units spent using the Mac in the cloud.

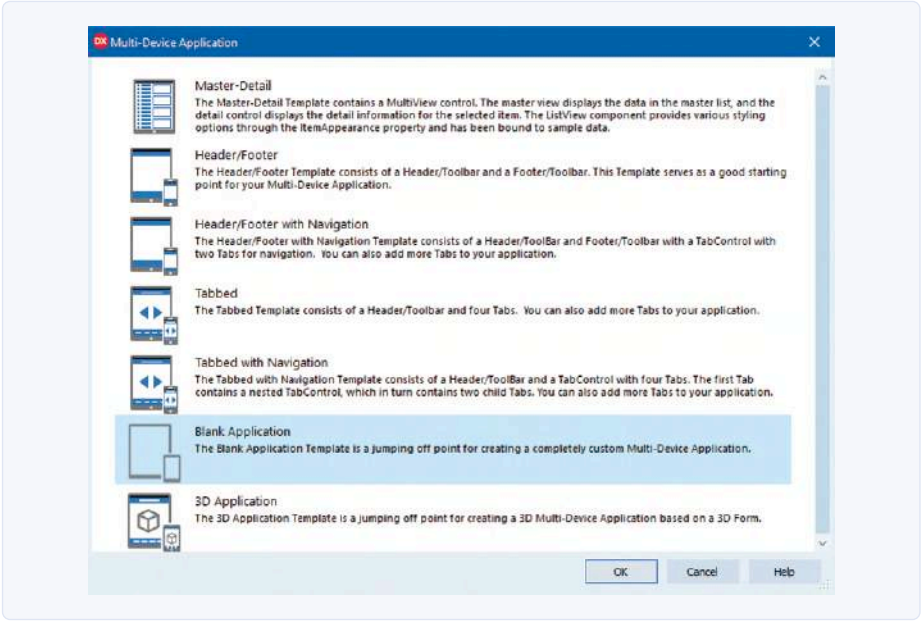


Figure 5: Creating a new multi-device, cross-platform app.

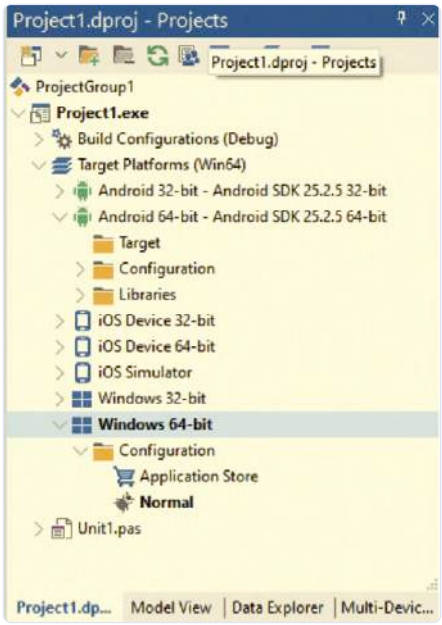


Figure 6: Target platform selected with Delphi.

ent target systems (Windows, Android, iOS). The properties of the components can be configured in the *Object Inspector*. Set the *Text* property to the value "*Hello Elektor*". Save the project and the associated files using the *File* menu. We now have a first test application that can run on the different platforms. Open the *Target Platforms* option in the *Project Manager* and select the 32-bit Windows platform by double-clicking it (**Figure 6**). Start the application (green arrow). This should now run as a 'normal' Windows application (**Figure 7**).

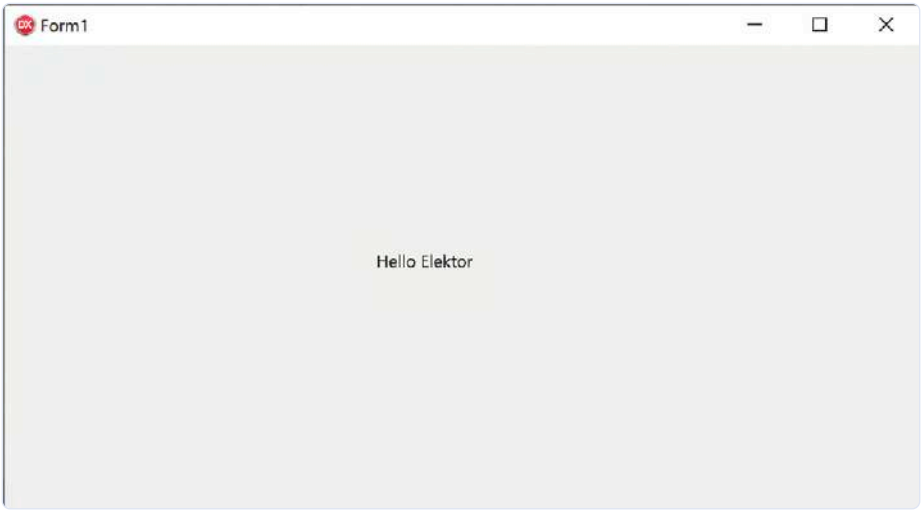


Figure 7: The test app as a Windows application.

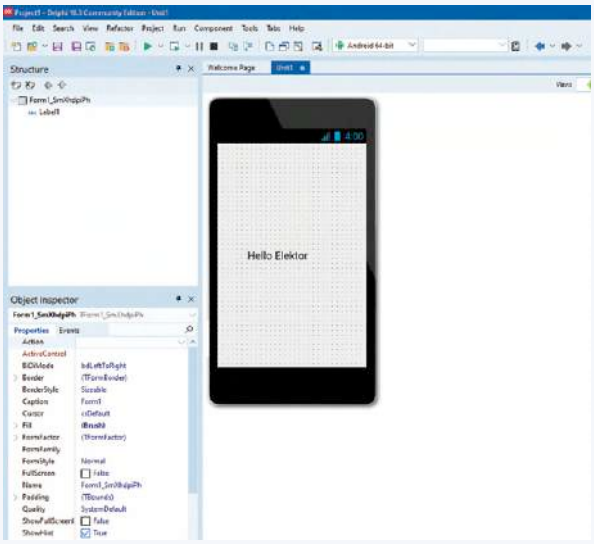


Figure 8: Preview of the app on Android in the Designer within RAD Studio.

Figure 9: "Hello Elektor" on the iPhone simulator.

Now close the application and go back to the development environment. Select Android 64-bit as the target platform. If the smartphone/tablet has been configured correctly as above, the connected device will now appear here. Activate it and start the app again. The app should be installed and executed on the mobile device. You can also preview the app for an Android device in the *Designer* of the development environment (**Figure 8**).

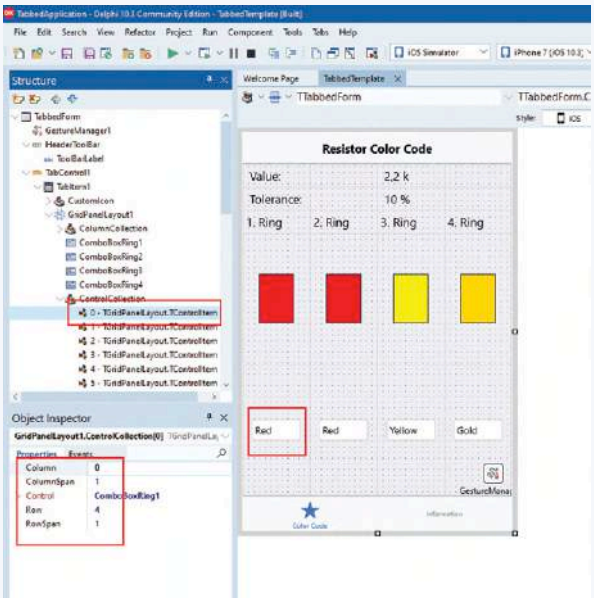


Figure 10: The positioning of the control element using a table.

If you have also configured the system to support iOS, you can connect to the Mac PC and, via the PAServer (see above), to Xcode, the simulator and an iPhone, running the app from there (**Figure 9**). After this minimal introduction, we now want to create a more comprehensive app and familiarise ourselves further with the development process.

An app for electronics engineers

In this section we approach the development of an app for determining a resistors value from its coloured rings. First of all we design the interface in the graphic designer. Again, choose *File | New | Multi-Device-Application - Delphi* as a template and then the *Tabbed* option. We are provided with tabs view (on top in Android, at the bottom in iOS). The contents of these tabs have to be filled in accordingly. The template contains four tabs (type *TTabItem*), which we reduce to two, i.e. we delete two tabs via *Structure / structure*. In tab one we place the elements for colour code selection and in tab two we place general information about the app. It has proved to be a good idea to make a sketch of the planned interface before implementing it, thus clarifying where the control elements should be placed (paper and pencil are sufficient). The arrangement of the control elements are usually not done with absolute position infor-

Colour	Ring 1 Position 1	Index in TComboBox	Ring 2 Position 2	Index in TComboBox	Ring 3 Multiplier	Index in TComboBox	Ring 4 Tolerance	Index in TComboBox
Black	0	0	0	0	1	0	-	-
Brown	1	1	1	1	10	1	1%	0
Red	2	2	2	2	100	2	2%	1
Orange	3	3	3	3	1,000	3	-	-
Yellow	4	4	4	4	10,000	4	-	-
Green	5	5	5	5	100,000	5	0.5%	2
Blue	6	6	6	6	1,000,000	6	0.25%	3
Violet	7	7	7	7	10,000,000	7	0.1%	4
Grey	8	8	8	8	-	-	0.05%	5
White	9	9	9	9	-	-	-	-
Gold					0.1	8	5%	6
Silver					0.01	9	10%	7

Table 1: Resistor colour ring values and position (index) in the respective selection boxes (TComboBox).

mation because you have to consider that apps on mobile devices are used with different screen sizes and resolutions. A table structure can be a good starting point for the rough allocation of space. The individual elements are then arranged in rows and columns. For each row and column we can again specify the size (absolute, but relative is better). For our app we choose a grid of five rows and four columns. We will us a control of the type *TGridPanelLayout*. Select it from the palette and drag it to the first tab. To ensure it takes up all the space, set the *Align* property to the *Client* value. Using the *Structure* view, add a total of five rows and four columns to this *TGridPanelLayout*. You can divide the size of the rows by percentages as follows: 7%, 7%, 7%, 40% and 39% (total = 100%). The four columns are divided equally, i.e. each with a size of 25%.

Now it is time to place the controls in this table layout. Select a desired element from the palette and drag it to the tab. For the colour selection we use controls of type *TComboBox*. In the *Structure* view we can then set the exact position (row and column) for each element under *TabItem1 | Control Collection*. In **Figure 10** you can see this for the colour selection of the first ring.

The positions *Column: 0* and *Row: 4* are selected because the selection box should also be placed at this position. For the display of text, such as labels or notes, we use controls of type *TLabel*. We also use *TLabel* for values that we want to display later, such as the calculated resistance value. The coloured rings are simulated by colour-filled rectangles, i.e. control elements of the type *TRectangle*. We want to adjust their colour from the source code later. In the selection fields for the colour rings (*TComboBox*) we add the available colours according to the well-known pattern in **Table 1**.

The layout of the application in the graphic designer can be seen in **Figure 11**. After starting the app (simulator or physical device) we

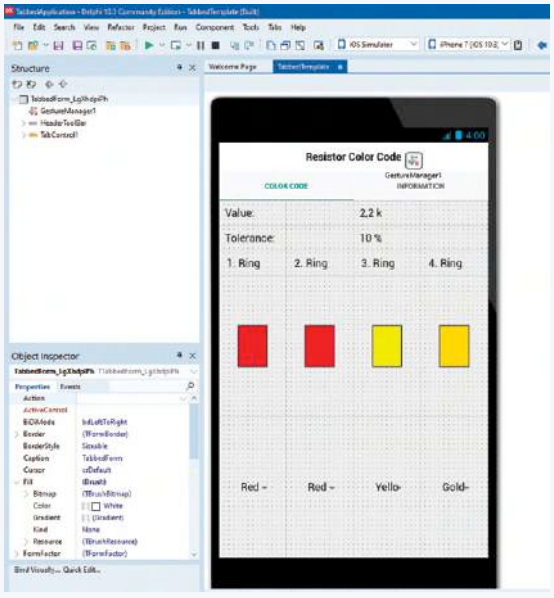


Figure 11: The layout of the app in Delphi — Android view.

Listing 1. Adjustment of the colour displayed dependent on the selection.

```
case digit1 of
    0:
    begin
        RectangleRing1.Fill.Color := TAlphaColors.Black;
    end;
    1:
    begin
        RectangleRing1.Fill.Color := TAlphaColors.Brown;
    end;
    2:
    begin
        RectangleRing1.Fill.Color := TAlphaColors.Red;
    end;
    …
    …
    9:
    begin
        RectangleRing1.Fill.Color := TAlphaColors.White;
    end;
end;
```

Listing 2. Calculation of the resistance value and formatting of the display.

```
if digit3 < 8 then
    value := Power10((digit1 * 10 + digit2), digit3)
  else if digit3 = 8 then
    value := (digit1 * 10 + digit2) * 0.1
  else if digit3 = 9 then
    value := (digit1 * 10 + digit2) * 0.01;

result := FloatToStr(value) + ' Ohm';

if value > 1000 then
    result := FloatToStr((value / 1000)) + ' k Ohm';

if value > 1000000 then
    result := FloatToStr((value / 1000000)) + ' M Ohm';

LabelValue.Text := result;
```

see the designed layout of the tab (**Figure 12**). Initially, we filled the *Information* tab only with a logo (an element of type *TImage*) and text (**Figure 13**).

Once you have got this far, the next step is to determine the logic for determining the resistance value. A recalculation and update of the display must be done every time we select a new colour value. To make it easier to identify the controls of the interface in the source code, we give the elements unique names. For this purpose, the property name of each element is given a unique name, for example *LabelValue*, *RectangleRing1*, *ComboBoxRing1,* etc.

The boxes for the colours of the rings are then all selected and we select the event *OnChange*. With a double-click in the object inspector we create a procedure for this event. As a result, this source code is called whenever we change the colour selection.

Next we adjust the colour of the rectangles according to the selection of the ring. This can be done via:

RectangleRing1.Fill.Color := TAlphaColors.Red
This will draw the rectangle in red. For the first selection box it will look like the one shown in **Listing 1**.

For the other coloured rings, the selection is adjusted according to the above-mentioned index (see **Table 1**). This is followed by the calculation of the resistance value, i.e:

(value of position 1 × 10 + value of position 2) × 10^multiplier.

The calculation is made in the unit Ohms. For values greater than 1,000 a conversion is made in kΩ, and for values > 1,000,000 in MΩ. For the ring colours gold and silver a multiplication with the values 0.1 and 0.01 is implemented. The value for the tolerance must only be selected from the above table (**Listing 2**). With that, our app is now functional! You can choose any colour combination of rings and the correct value for the resistance is displayed. This can now be used to create and deploy app packages for Android and/or iOS. It is also possible to share the app via the Google Play or Apple Store from Delphi. You can also download the source code for the sample application from the Elektor website for this article [5].

Further options

In this example we have already shown some of the possibilities of app development using Delphi. However, there is much more to it. Here are some examples:

- *Visual components*: In order to create appealing interfaces, a variety of components (buttons, text fields, check boxes, tabs, etc.) are available. You can place these controls using the graphical designer and configure them via the properties. The system automatically creates the appropriate elements for the target platform (Android, iOS, Windows) when compiling the application.
- *Non-visual components*: These provide important building blocks for programming that are needed over and over again. Examples are a timer component (timer) or components to facilitate access to databases.
- *Mobile app specifics*: Visual and non-visual components are offered which enable typical tasks in the programming of mobile apps to be solved more quickly, for example access to positioning functions, use of the camera of the smartphone, or the handling of data

exchange with servers (backends) in the cloud.

- These development options are needed in stages as the mobile app is developed if it is to reach the point of productive use. The development environment also provides support for generating the app packages needed for the distribution of the apps via the Google or Apple Store.

Conclusion and outlook

Although apps are only applications for the 'small' devices in your pocket, programming them is no less work. With suitable tools you can save yourself a lot of effort by developing simultaneously for both Android and iOS using a common base of source code. Delphi offers an intuitive approach that will probably seem familiar to many electronics engineers (Visual Basic, Windows Forms etc.). A user interface can be developed efficiently and quickly in the graphical editor. This approach is also useful if you have to write an application for Windows or Linux allowing the source code to be used several times. Only one click is require to switch between target platforms. The development environment generates the necessary app packages automatically and, in the ideal case, you will not need to make any further adjustments.

200265-04

Questions or Comments?
Do you have questions or comments regarding this article? Then get in touch with the author by email via v.krypczyk@larinet.com, or the Elektor editorial team at editor@elektor.com.

RELATED PRODUCTS

- **Book "Android App Development for Electronics Designers"**
  www.elektor.com/android-app-development-for-electronics-designers
- **Book "C# Programming for Windows and Android"**
  www.elektor.com/c-programming-for-windows-and-android

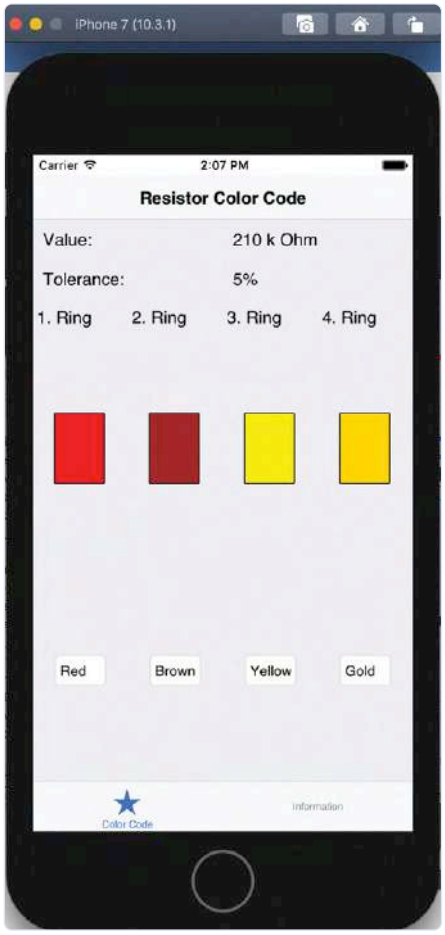


Figure 12: The app in action (not yet fully functional) in the iOS simulator.

Figure 13: The 'Information' tab.

Contributors
Author: **Dr. Veikko Krypczyk**
Translation and Editing: **Stuart Cording**
Layout: **Giel Dols**

WEB LINKS

[1] **RAD Studio Download**: www.embarcadero.com/products/delphi/starter/free-download
[2] **'MacInCloud' Service**: www.macincloud.com/
[3] **Android Mobile Application Development:** http://docwiki.embarcadero.com/RADStudio/Rio/en/Android_Mobile_Application_Development
[4] **iOS Mobile Application Development:** http://docwiki.embarcadero.com/RADStudio/Rio/en/IOS_Mobile_Application_Development
[5] Project page for this article: www.elektormagazine.com/200265-04

From Life's Experience

The Value of a Technical Education

By **Ilse Joostens** (Belgium)

A long time ago, the French teacher at the technical college, an enthusiastic museum visitor and opera lover, was in the classroom on a drizzly afternoon. He reported on an exhibition of the works of the surrealist Belgian painter René Magritte in Brussels. One of his most famous paintings, he said, showed a pipe with the caption "Ceci n'est pas une pipe". "But you won't understand that," he continued, dryly, with his report and then proceeded with the lesson. This made it clear (not very subtly) that he was actually looking down on us – after all, we were stupid science students and, above all, cultural barbarians with whom all this beauty had not been shared…

The waterfall system

Despite the numerous educational reforms and the recent concentration on STEM subjects, Belgian secondary education has traditionally been the scene of a race to the bottom. If you could count yourself among the better pupils in primary school at the time when I was about twelve years old, it was natural that you would move on to general secondary education. The majority of my classmates went in that direction and it was mainly the underachieving pupils, the fools, so to speak, who went into technical and vocational secondary education. Then race to the bottom really heated up. It was already painful when you didn't get along well at school and had to jump from Latin or the modern languages into economics, but it became completely disastrous when you were banished to the realms of a technical education. Fortunately, I was spared this cruel fate - veni, vidi, vici!

This 'waterfall system' results in a vicious circle as it acquires an influx of 'less able' students, lowers the level of education in technical subjects, thus lowering the general appreciation of technology, making parents more likely to send their 'gifted children' to general education even though they are interested in technology. In return, the drop-outs from technical education migrate to vocational education and training. It is then assumed that pupils in a technical subject have lower chances of success in higher education, which means that it is then the mainly generally-educated people, who are not handicapped by previous practical technical knowledge, who enter higher technical courses of study at colleges and universities. Once there, a mountain of theory is then poured into their heads. As a result of all the double, triple and circular integrals, differential equations, and other mathematics learnt we were able to calculate what was flowing through some switchgear, but we never saw any in their natural habitat. And no, Magnefix (switchgear solution from Eaton) is not a character from an Asterix book! Despite my high expectations, these were the most boring years of my life.

Techies: more than just problem solvers

It is a wonderful moment in life when you graduate cum laude or sine and show off your brand new diploma in a solemn proclamation. The very harsh reality, however, is that if you are unlucky, the day after the proclamation you are no longer a successful student but just an unemployed loser.

The business world today, which seems to revolve more and more around accountants, economists, opportunists, consultants and other white-collar criminals, is not especially friendly toward the technically-oriented amongst us either. We may concede to them using their woolly and meaningless management expressions, and favourite of the top management, but statements such as "they will solve the problem" or "let's just open a can of techies" do not command much respect [1]. This undervaluation of technicians is humorously addressed in the British sitcom "The IT Crowd" by Irish director Graham Linehan. As a former system administrator of a certain retail company, I have to admit that there is a lot of truth in the crazy experiences of Roy Trenneman and Maurice Moss — a highly recommended watch!

The myth of the STEM-deficit

For years we were deceived by the masses that there would be a lack of technical professionals and engineers [2]. But, in Belgium,

these positions have been on the list of so-called 'shortage professions' for years.

If there really was such a shortage one would expect that, within the framework of market forces, the wages and benefits for technicians would be correspondingly high. Nothing could be further from the truth: many technical jobs in Belgium are paid below average. With a standard nine-to-five office job, including making copies, making coffee and the watering of plants, you can easily earn much more than with a job as a service technician. And you don't have to crawl into dank-smelling rooms with no daylight and a fluorescent tube roaring over your head.

Even higher-end technical professionals are not particularly well taken care of. An acquaintance of mine, a Senior Electronics Hardware Engineer at an unspecified high-tech company, was merely presented with a framed certificate and a bunch of flowers as a thank you for his efforts after years of loyal service on the occasion of a staff party. Not very much to write home about, especially in comparison to the bonuses and salary increases that were and are regularly awarded to management and salespeople.

So, with that I will end here with the well-known saying: "Why be an engineer/scientist when you can be their boss?"

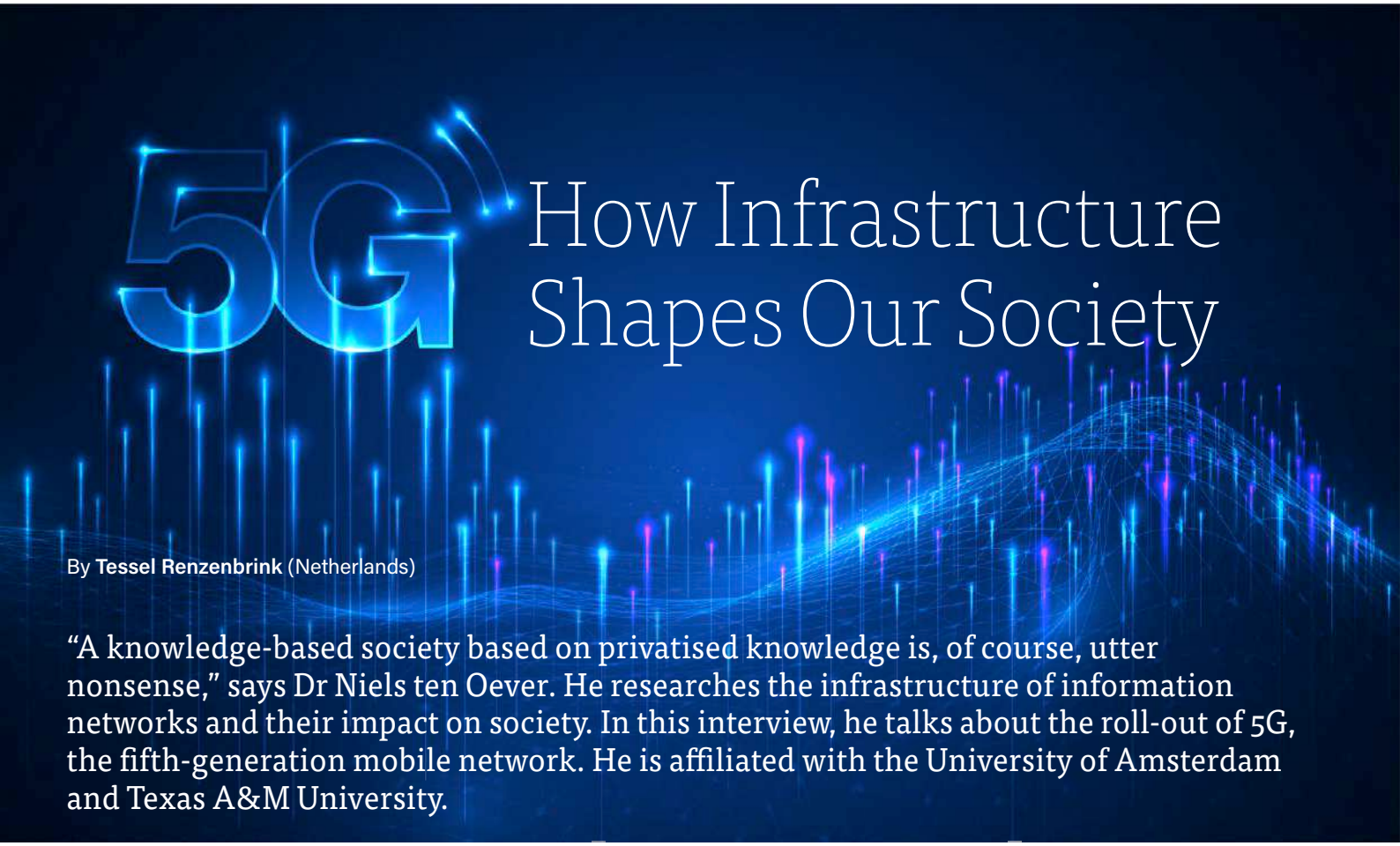
200455-04

Questions or Comments?
Do you have questions or comments regarding this article? Then get in touch with the Elektor editorial team at editor@elektor.com.

Contributors
Text: **Ilse Joostens (Belgium)** Editing and Translation: **Stuart Cording** Layout: **Giel Dols**

WEBLINKS

[1] **Bullshit Generator**: www.atrixnet.com/bs-generator.html
[2] **The fairy tale of 'beta deficiency'**: www.beteronderwijsnederland.nl/blogs/2009/02/het-sprookje-van-het-beta-tekort/



How Infrastructure Shapes Our Society

By **Tessel Renzenbrink** (Netherlands)

"A knowledge-based society based on privatised knowledge is, of course, utter nonsense," says Dr Niels ten Oever. He researches the infrastructure of information networks and their impact on society. In this interview, he talks about the roll-out of 5G, the fifth-generation mobile network. He is affiliated with the University of Amsterdam and Texas A&M University.

"It is typical of infrastructure that we often do not think about it", says Ten Oever. "Only when it breaks down - the electricity goes off or the roads are full of potholes - does it stand out. Nevertheless, we should not underestimate how decisive the design of infrastructure is. Take the road network, for example. How it is laid out has an impact on your daily life. A four-lane road flanked by narrow pavements encourages a different use of public space than promenades and wide cycle paths. The same applies to information infrastructure such as 5G. The architecture of the network delivers both possibilities and introduces limitations. The question is then: What do you optimise the network for?

Virtualisation of the network

"With the arrival of 5G, a new infrastructure will be rolled out," continues Ten Oever. "It operates on frequencies other than those used by 4G. So there will be new antennas but also new routers and switches. This also means that new computing paradigms can be rolled out. These routers and switches are much more like general purpose computers: devices on which we can simply run Linux. This is in contrast to the current generation of network equipment with its application-oriented hardware coupled with specific firmware. Because you can more easily fix problems in software [than in hardware], you can virtualise network functions."

Network virtualisation makes it possible to make the network increasingly intelligent. This means a farewell to one of the central design principles of the early Internet: the end-to-end principle. End-to-end means that any computer in the network can connect to any other connected computer. The idea was that the network itself does as little as possible. It consists of 'stupid pipelines' that only transport data. The intelligence - computing power and data storage - takes place at the edges of the network: the computers of the users. This trend of shifting intelligence from end devices to the network itself has been going on for years. Think of cloud computing where data storage and software applications are run on third party servers. The arrival of 5G makes it possible to continue the trend of more intelligence in the network with Information Centric Networking, where data is stored in the network itself. Ten Oever gives an example of what this can look like.

Game of Thrones in the bath

"You already have certain applications, such as Netflix, making recommendations based upon your viewing behaviour. But they could also predict where you will watch it: 'They always watch Game of Thrones in the bath', or 'They always listen to that podcast on their bike.' That podcast could then be cached in the smart lamppost that I always cycle past. Then, when you arrive at work, your computer could already be on and your favourite programmes are ready to use. Super helpful. But how do you keep control of it? Do you want to be in control? How do you know who is doing it and on what basis?

"We are not there yet. But the conditions for its existence, namely that general purpose infrastructure, is made possible with 5G", continues Ten Oever. "That will make it possible to optimise such a network more and more. That is every network operator's dream because you can then set up your network as efficiently as possible. Efficiency is a parameter that can be optimised. But this also means that a control option for the consumer will disappear. After all, it is becoming increasingly

Niels ten Oever researches the infrastructure of information networks. (Photo provided by Niels ten Oever).

unclear which device is connected to what services and what information it is sharing. You can foresee how this could lead to stronger filter bubbles. Or even lead to more privacy-sensitive data that could be extracted and even used to develop personal censorship profiles, so that a particular person is no longer allowed to view certain content, or allowed to come into contact with a another person."

The 5G standards and patents push

Ten Oever is investigating how decision-making on the standardisation of 5G works. 5G is a bundle of different technologies that will be standardised in three phases. Completion of the second phase is planned for 2020. The specifications for 5G are defined by the 3GPP consortium, an alliance of six standardisation organisations including the European ETSI. The final standards will eventually be adopted in the International Telecommunications Union (ITU) in which member countries have voting rights.

Ten Oever: "It is quite difficult to estimate why certain decisions are taken. But one of the reasons for developing certain technologies is patents. This is something that plays much less of a role in the Internet world. Companies there do have patents, but they use them defensively. That can be seen as a kind of truce: I have patents, you have patents, but we do not use them. But if you start, we'll get even.

"But in mobile telephony it works differently. In 3GPP much is based on proprietary hardware and software. That is also where most of the money is made. A company like Qualcomm earns more from patents than from the sale of hardware. Companies therefore try to push their patented technology in the development of the standards. We expect a huge explosion of new devices in the coming years. Imagine that, as a company, you get 10 cents per device in licensing. That's an enormous amount of money and you would never have to do anything for it again."

"We have no idea how these networks work"

These patents make the operation of 5G more opaque than the traditional internet. Ten Oever: "You and I can study how the Internet works because the knowledge about it is public. We can set up a mini Internet at home. But we cannot build a mini-5G network because much of the knowledge is behind patents and licences in the private sector". The lack of transparency also stands in the way of research and development, says Ten Oever. "At the moment we are still working with protocols. These are relatively simple rules, but they will all be optimised algorithmically in the near future. That raises a lot of complex questions. It would be good to think about this with a lot of people. You should involve universities and research institutes and open up the market to newcomers with new ideas. But we cannot do that now because we have no idea how those networks work. Patents are a kind of wall around the sharing of knowledge.

"Instead of protecting the interests of established companies, we should focus on open hardware and open software", says Ten Oever. "Because then we will create an open ecosystem in which innovation can take place. And that is where governments have a role to play. They could make demands of this kind. With infrastructures such as water and electricity supply, we all have very strict rules about how it should be done, how we control it and what it should comply with. That is much less the case with digital infrastructure. But when you become a vital infrastructure, you also have the responsibilities that come with it. There is still too little awareness of this in the Internet and telecoms world. And the government is insufficiently informed and does not sufficiently recognise the need to make demands of this kind.

"The human right to science should also cover our information infrastructure", concludes Ten Oever. "This has priority over protecting business interests. If you say that we are an information society, or a knowledge society, then that knowledge must be available. A knowledge-based society based on privatised knowledge is, of course, utter nonsense."

200468-03

Questions or Comments?
Do you have questions or comments regarding this article? Then get in touch with the Elektor editorial team at editor@elektor.com.

Contributors
Text: **Tessel Renzenbrink** (Netherlands) Editing and Translation: **Stuart Cording** Layout: **Giel Dols**

Practical ESP32 Multitasking (5)

Task event notification

By **Warren Gay** (Canada)

While FreeRTOS provides queue, semaphore and mutex functions for synchronisation, they can sometimes seem overcomplicated when needs are simple. FreeRTOS version V8.2.0 introduced the concept of *direct to task event notifications*, providing the programmer with a light-weight method to synchronise with tasks.

Each task includes a built-in 32-bit event notification value that is initialised to zero when the task is created. Because this is built into each task, no additional RAM is required. This demands significantly less memory when compared to, for example, the use of semaphore objects.

Limitations

There are, however, some limitations when using direct task notifications, such as:

- You cannot send a notification to an ISR, because an ISR is not a task. However, an ISR *can* notify a task.
- Only one task may be notified by a task notify call (event groups must be used to notify multiple tasks).
- Notification events cannot be buffered like queues.
- The *notifying* task (or ISR) does not block its execution to wait for the receiving task to receive the event. The event is merely posted and the call returns immediately, unimpeded.

If any of these restrictions are in conflict with your requirements, you will need to choose an alternative FreeRTOS mechanism instead.

Event Waiting

The first step in using task event notifications is to inform the receiving task to wait for a notification. In other words, the task must be set up to pause until an event is triggered. This is implemented by using of one of the following functions:

- `ulTaskNotifyTake()`
- `xTaskNotifyWait()`

The function `ulTaskNotifyTake()` is the simplest of the two and the function that is examined in this article. The `xTaskNotifyWait()` is more advanced and is explored in detail in the Elektor book *FreeRTOS for ESP32-Arduino*.

ulTaskNotifyTake()

The task receiving a notification blocks its execution by calling `ulTaskNotifyTake()`. The call requires two arguments:

```
uint32_t ulTaskNotifyTake(
  BaseType_t xClearCountOnExit, // pdFALSE or pdTRUE
  TickType_t xTicksToWait
);
```

The first argument operates on the 32-bit task notification word (also referred to as the task event word) and is defined in **Table 1**. The second parameter is the familiar *timeout* value in ticks (use the macro `pdMAX_DELAY` if you don't want to timeout). In all calls to `ulTaskNotifyTake()`, the execution of the calling task is blocked while the task event word remains at the value zero. When the value becomes non-zero, the argument `xClearCountOnExit` determines how the task event word is updated before returning.

Table 1. The meaning of the arguments xClearCountOnExit for ulTaskNotifyTake().

`xClearCountOnExit` Value	Effect on 32-bit task notification word
`pdFALSE`	Decrement value, blocking further execution if the value was zero prior to the decrement, before returning pre-decrement value.
`pdTRUE`	Clear value to 0, blocking further execution if value was already zero, before returning pre-clear value.

The value returned from the function call is the value of the event notification word *before* it was cleared or decremented. When a *timeout* occurs, the returned value will also be zero.

Binary Notification

When the argument `xClearCountOnExit` value is `pdTRUE`, the call to `ulTaskNotifyTake()` operates like the binary semaphore *take* operation. If the task notify event word is *already non-zero*, the call returns immediately and clears the event word to zero (its earlier, non-zero value is, however, returned). If the task notify word was zero at the time of the call, further task execution is *blocked* until the task is notified (subject to timeout). If a timeout occurs, the return value will always

be zero (reflecting the value of the event word at the time of return).
In this mode of operation, the call operates like a binary semaphore. If the task were to be notified twice before the receiving task calls `ulTaskNotifyTake()` then the returned value will be 2. However, the task event word is cleared to *zero* upon return. This effectively provides only one notify event in this particular case, but the actual number of notifications is easily determined by noting the return value.

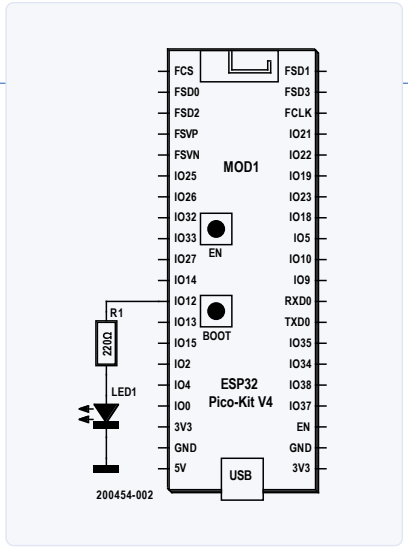


Figure 1: Schematic for the tasknfy.ino code.

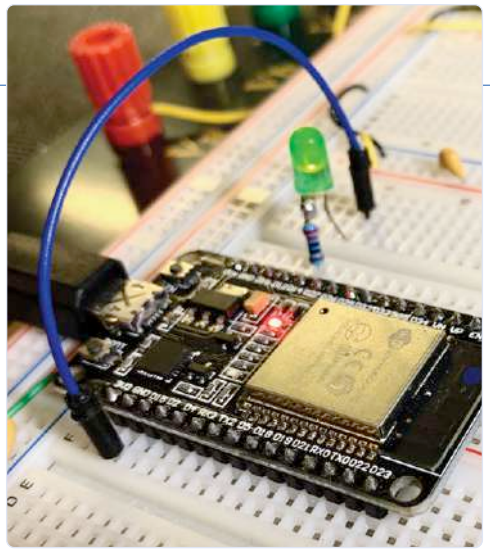


Figure 2: Setup of final demonstration circuit using GPIO 12.

Counting Notification

If the requirement is that multiple task notifies *must* cause multiple task wake-ups, then the `xClearCountOnExit` argument should be provided with the value `pdFALSE`. In such a use case, the receiving task blocks while the task event value is zero (as before). Each notification of the task increments the task event value, but it is only decremented by *one* for each call to `ulTaskNotifyTake()`. For example, if four notifications occur prior to the receiving task calling `ulTaskNotifyTake()`, then that call immediately returns without blocking until the task notify value has been decremented back to zero. In other words, the receiving task must make four calls to `ulTaskNotifyTake()` before the event word decrements all the way back to zero.

Give Notify

While the receiving task uses the function `ulTaskNotifyTake()`, the notification is provided by using `xTaskNotifyGive()`:

```
BaseType_t xTaskNotifyGive(TaskHandle_t xTaskToNotify);
```

This function only requires the handle of the task to be notified and always returns `pdTRUE` (the FreeRTOS manual indicates that `xTaskNotifyGive()` is defined as a macro).

Demonstration

A very simple demonstration is provided in **Listing 1** [1]. This program uses the Serial Monitor in order that you can see the reported event word value returned. **Figure 1** illustrates the schematic for wiring an optional LED, while **Figure 2** illustrates the breadboard setup. Just about any ESP32 dev board can be used where GPIO 12 is available.

The tasknfy1.ino program uses the `loopTask()` provided by Arduino to repeatedly notify `task1()` (line 42) at intervals of one second. The task1() function then blocks (line 11) until notified. Once notified, it then toggles the LED (line 12). The function `task1()` is notified by the use of the handle `htask1` (line 42). That handle was created in the `setup()` function at line 34.

Listing 1. The `xTaskNotifyGive()` and `ulTaskNotifyTake()` demonstration program tasknfy1.ino.

```
0001: // tasknfy1.ino
0002:
0003: #define GPIO_LED        12
0004:
0005: static TaskHandle_t htask1;
0006:
0007: static void task1(void *arg) {
0008:  uint32_t rv;
0009:
0010:  for (;;) {
0011:    rv = ulTaskNotifyTake(pdTRUE, portMAX_DELAY);
0012:    digitalWrite(GPIO_LED, digitalRead(GPIO_LED)^HIGH);
0013:    printf("Task notified: rv=%u\n", unsigned(rv));
0014:  }
0015: }
0016:
0017: void setup() {
0018:  int app_cpu = 0;
0019:  BaseType_t rc;
0020:
0021:  app_cpu = xPortGetCoreID();
0022:  pinMode(GPIO_LED, OUTPUT);
0023:  digitalWrite(GPIO_LED, LOW);
0024:
0025:  delay(2000);    // Allow USB to connect
0026:   printf("tasknfy1.ino:\n");
0027:
0028:  rc = xTaskCreatePinnedToCore(
0029:    task1,        // Task function
0030:    "task1",      // Name
0031:    3000,         // Stack size
0032:    nullptr,      // Parameters
0033:    1,            // Priority
0034:    &htask1,      // handle
0035:    app_cpu       // CPU
0036:  );
0037:  assert(rc == pdPASS);
0038: }
0039:
0040: void loop() {
0041:  delay(1000);
0042:  xTaskNotifyGive(htask1);
0043: }
```

```
0040: void loop() {
0041:   delay(1000);
0042:   xTaskNotifyGive(htask1);
0043: }
```

The `task1` code is almost as trivial, operating in an endless loop. It blocks its execution at the call to `ulTaskNotifyTake()` in line 11 to wait for a notify event:

```
0007: static void task1(void *arg) {
0008:  uint32_t rv;
0009:
0010:  for (;;) {
0011:   rv = ulTaskNotifyTake(pdTRUE, portMAX_DELAY);
0012:   digitalWrite(GPIO_LED, digitalRead(GPIO_LED)^HIGH);
0013:   printf("Task notified: rv=%u\n", unsigned(rv));
0014:   }
0015: }
```

The further execution of `task1` remains blocked until the task notify event arrives. The value assigned to `rv` (line 11) is the value of the event notification word *prior* to it being cleared (due to the `pdTRUE` argument). Notice how clean and simple this is — there are no other semaphore objects involved and, thus, no additional handles.
The resultant serial monitor session should look like this:

```
tasknfy1.ino:
Task notified: rv=1
Task notified: rv=1
...
```

If you want to notify a task from an ISR you need use the function named `xTaskNotifyGiveFromISR()` instead. The FreeRTOS "FromISR" suffix convention allows for safe operation from within an ISR, which often includes special handling.

```
void vTaskNotifyGiveFromISR(
  TaskHandle_t xTaskToNotify,
   BaseType_t *pxHigherPriorityTaskWoken
);
```

The second argument is the address of the ISR wakeup flag, indicating whether or not the scheduler should be invoked (supply `NULL`/`nullptr` when not required). This task notify function provides a very convenient way for an ISR routine to notify a task.

Advanced Notification

In this article we've only discussed the simplest pair of functions `xTaskNotifyGive()` (or `xTaskNotifyGiveFromISR()`) and `ulTaskNotifyTake()`. There are additional, advanced functions available, including the following:

- `xTaskNotify()` (or `xTaskNotifyFromISR()`)
- `xTaskNotifyAndQuery()` (or `xTaskNotifyAndQueryFromISR()`)
- `xTaskNotifyWait()`
- `xTaskNotifyStateClear()`
- `xTaskNotifyValueClear()`

These functions operate in a similar manner but support event processing at the bit-level. They also provide more selective forms of event testing and clearing. You can read about these in the FreeRTOS reference manual [2] or online [3].

Conclusion

The FreeRTOS task notification API provides the application developer with a lightweight event notification mechanism. This is particularly useful for implementing task notification from ISRs as it keeps the interrupt processing short, while allowing the hard work to be performed in the notified task. Even when not using interrupts it saves RAM, as well as simplifying the application, by not requiring additional synchronisation objects such as semaphores.

200454-01

Questions or Comments?
Do you have questions or comments about this article? Email the author at ve3wwg@gmail.com or contact Elektor at editor@elektor.com.

Contributors

Idea and Text: **Warren Gay** Schematic: **Patrick Wielders** Editor: **Stuart Cording** Layout: **Giel Dols**

WEB LINKS

[1] **Code for tasknfy.ino**: https://github.com/ve3wwg/FreeRTOS_for_ESP32/blob/master/tasknfy1/tasknfy1.ino
[2] **FreeRTOS Documentation**: www.freertos.org/Documentation/RTOS_book.html
[3] **FreeRTOS Documentation for Task Notification**: www.freertos.org/RTOS-task-notification-API.html

HP 10811 Oscillator

Peculiar Parts, the series

by **Neil Gruending** (Canada)

How do you measure time accurately? Just like voltage, a time measurement is only as good as the reference it's compared to. A crystal oscillator is a good starting point for many situations, but more work is required to make it a truly accurate reference. Let's take a look at one of Hewlett-Packard's solutions for this, the 10811 oscillator.

The core issue is that the output frequency of a crystal oscillator can fluctuate and change over time. Short-term fluctuations are usually due to frequency and phase modulation. Longer-term frequency drift can also happen with temperature changes and component ageing. If you are using the crystal for a CPU clock then these affects probably aren't something to worry about, but if you want to repeatedly make frequency measurements with an accuracy of 0.001 Hz, then all of those affects really matter.

A crystal's frequency error is lowest at its turnover temperature. For example, a conventional AT cut crystal will have a turnover temperature of about 25°C and the error will exponentially increase as the temperature increases or decreases. This isn't acceptable for a frequency reference so the HP 10811 uses a specially developed SC-cut crystal that dramatically reduces the second order frequency error over temperature in order that small temperature changes don't induce a large frequency change. Stabilising the crystal temperature near the turnover point also improves the accuracy, so the 10811 encloses the crystal in a temperature-controlled oven as shown in **Figure 1**. These features, combined with oscillator circuitry that is designed to minimise noise and phase errors, is what gives the 10811 its excellent accuracy.

The 10811 assembly is interesting because it's an early example of a flexible circuit board that's folded to fit within its enclosure, as shown in **Figure 2**. Referring to the top photo of the completely disassembled device, the clock and AGC circuitry are folded and attached to either side of the central metal structure. Then, the oven controller circuit is folded over a piece of thermal insulation and the TO220 heating elements are attached to the metal structure, as in the bottom photo. Once complete, everything is placed inside the thermally-insulated housing and sealed.

The HP 10811 oscillator was produced in several variations and can be found inside used test equipment that required a precision time reference, such as the frequency counter HP 5334A and GPS-disciplined frequency references like the HP Z3801A. The 10811 not only increases their accuracy, but it's stability means that longer-term measurements, such as Allan deviation [2], become possible, even in your lab!

190383-F-01

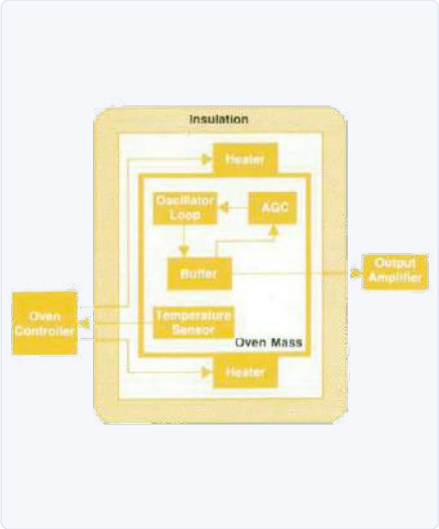


Figure 1: Block diagram of HP 10811 oscillator [1].

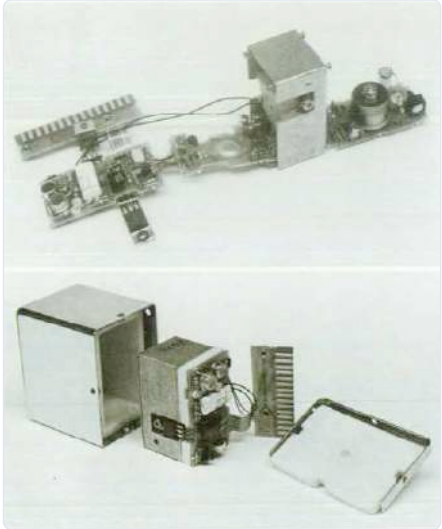


Figure 2: Photos of disassembled HP 10811 oscillator [1].

Contributors

Idea, Text and Images: **Neil Gruending**

Layout: **Giel Dols**

Editor: **Stuart Cording**

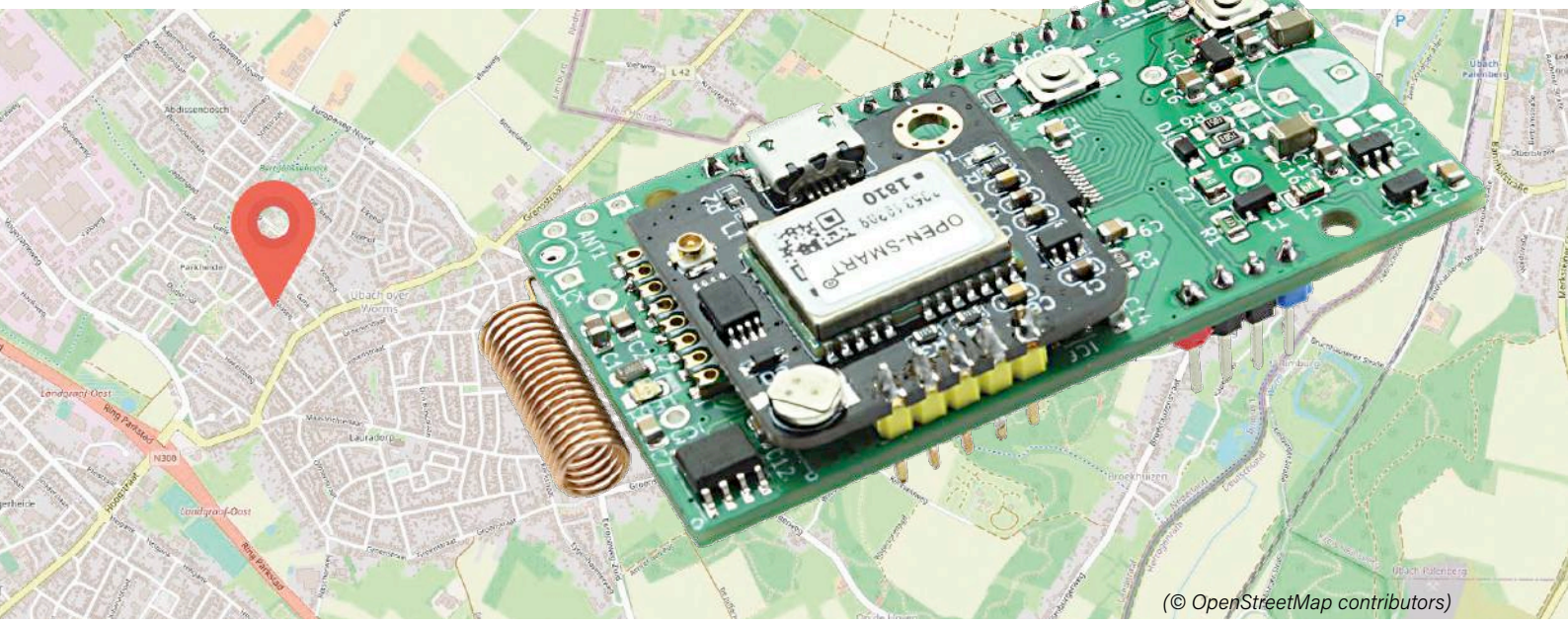
WEB LINK

[1] **Hewlett-Packard Journal - March 1981:** www.hpl.hp.com/hpjournal/pdfs/IssuePDFs/1981-03.pdf

[2] **Allan variance:** https://en.wikipedia.org/wiki/Allan_variance

LoRa GPS Tracker

With Open Hardware and Software



A GPS tracker that can send details of its geographic location at adjustable time intervals is a very useful device with many applications. This latest Elektor version is a neat solution. It doesn't use cellular networks like 3G or 4G but instead sends its positional information via the free LoRa network via Node-RED to a PC, Raspberry Pi, cell phone, tablet or other device where it can then be displayed on a map in a browser.

PROJECT DECODER

Tags
LoRa, RFM95, STM32, Raspberry Pi, Arduino, GPS, Education, OpenHardware, OpenStreetMap

Level
entry level – intermediate level – expert level

Time
Around 1 hour for complete construction incl. case and antenna

Tools
Soldering iron, screwdriver, drill

Cost
£80-£90/€90-€100/$105-$120

I imagine many of you have, at times, wanted to be reassured that some valuable belonging has not been moved from wherever you left it. Most people's first thought would be the family car, but items such as construction equipment or even farm animals can also end up in new and surprising locations. Such items were once equipped with GSM-GPS trackers like the ElekTrack [1] so that the rightful owner could determine the position of each individually-tracked item, or be alerted if it 'spontaneously' changed its location. However, a SIM card is always required here and, of course, network coverage, which can sometimes be patchy. Tracking also costs money, and roaming fees are sometimes not as cheap as you might think as some Russian researchers found out [2]. An alternative is a LoRaWAN–based tracker. LoRaWAN is a network operating in the license-free 868 MHz ISM band that offers long range while using low transmission power. 'The Things Network' is one such network using LoRaWAN. The gateways, i.e. the access points to the network, are operated by a community in which everyone can contribute with their own gateway, thus further expanding the network. The transport of data in this network is free of charge — ideal for communicating with remote networked sensors. If there is no gateway in your own environment you can set up your own for little outlay.

The LoRa node

The list of ingredients required to make a tracking device is quite basic but, just like the skill required to bake a good cake, the electronics developer needs to pay attention that they process all the ingredients correctly. In our case, this took a few weeks of Blood, Sweat & Tears (for younger readers: they were a

well-known jazz-rock band back in the 70s). Over the course of our preparations, our original concept for a LoRa GPS tracker became the universal Elektor LoRa node (**Figure 1**) that has many possible applications, of which the GPS tracker is only one. We intend that all the Elektor LoRa node projects and their offshoots in the past, present and future are, and will remain, 'open', both in terms of software and hardware. Since the Elektor LoRa node [3] was dealt with so extensively in the March/April issue of Elektor, along with the introduction to The Things Network [4], you should already be aware of their capabilities. When describing the hardware shown in **Figure 2**, we can concentrate on the essentials and use the space available here to get closer to the GPS module; the GPS software; the transport of the positional information; and the integration into

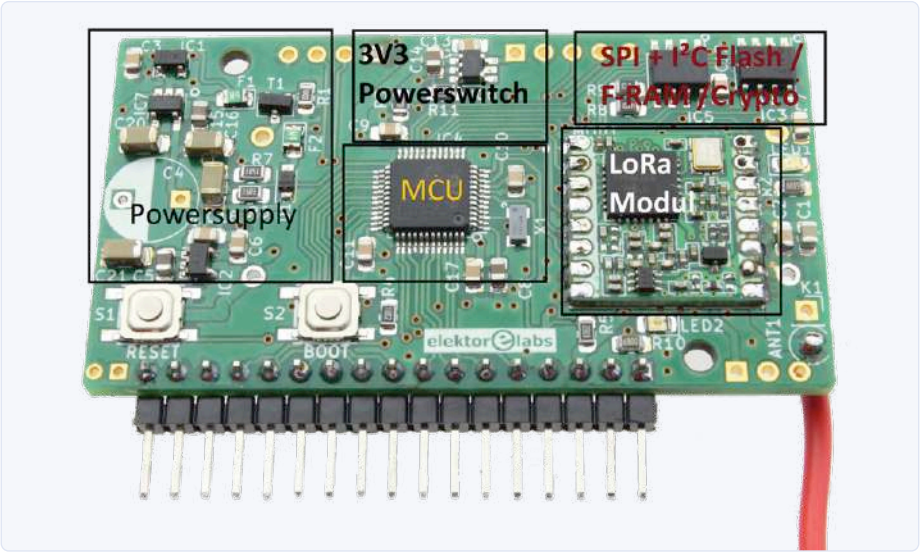


Figure 1: The Elektor LoRa node PCB fully populated.

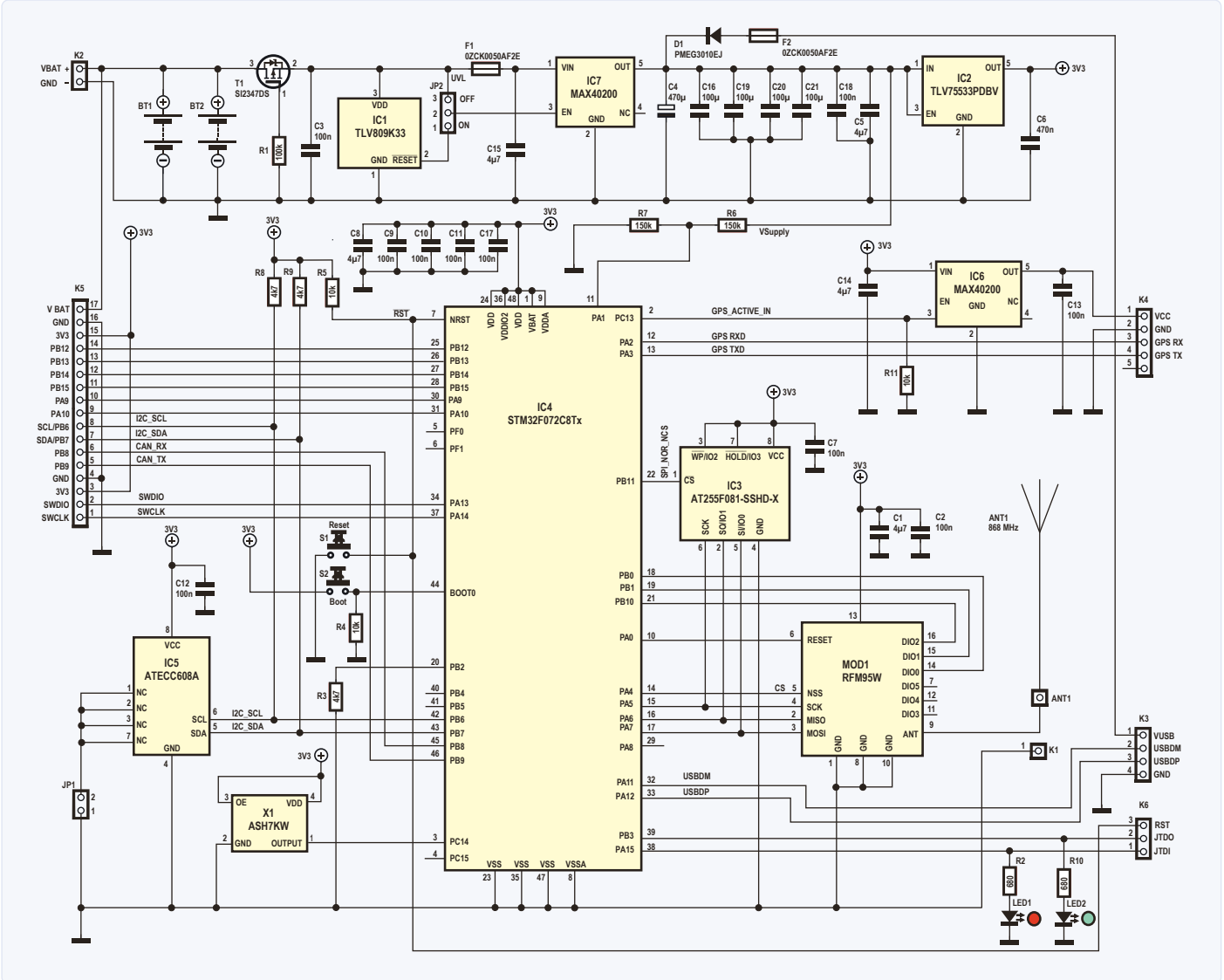


Figure 2: Schematic of the Lora node (from an earlier article).

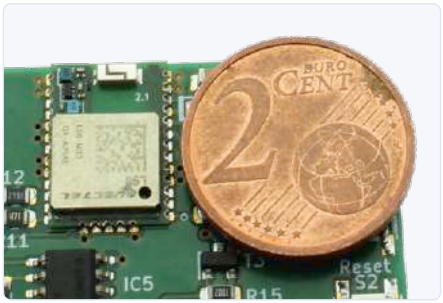


Figure 3: The tiny L96 GPS module.

The Things Network, MQTT and Node-RED, right through to its representation on a map.

The case

Here we chose an enclosure from the Hammond 1551 range that provide class IP54 protection (dust and splash-proof) and is suitable for outdoor use. This has enough room for the battery, baseboard and a GPS module.

The microcontroller

An STM32 microcontroller provides control. The STM32F72C6T has 64 kB of flash and 16 kB of RAM. Across the STM32 family of processors, many of those packaged in an LQFP48 outline are, in fact, pin-compatible, so a controller with more flash or even lower energy requirements can be fitted to the same board layout without too much effort. In addition to classic development using C/C++, the controller can also be programmed via the Arduino framework thanks to the STM32duino project. Using the (non-overwritable) bootloader in the MCU, a simple USB-to-serial converter is all you need to flash new firmware onto the chip.

The LoRa transmitter

The project uses an RFM95W LoRa transceiver, working in the 868 MHz ISM band.

Power to the circuit

The circuit is powered from an inexpensive, rechargeable Li-ion battery from the world of model-making and drones. In principle it would be possible to use AAA-format lithium cells, but you would then need a bigger case.

We have deliberately chosen not to fit a charging circuit, so the battery must be recharged externally by a suitable charger. Undervoltage is detected by the supply supervisor chip IC1 that safeguards the battery from deep discharge. The 'ideal-diode' chip IC7 protects against possible reverse powering of the circuit and thereby prevents a condition where the battery may become overcharged and dangerous. The protection circuit is itself protected by a 500 mA poly fuse directly on the battery. Thus a short circuit cannot lead to over-current, frazzled cables or PCB traces. These safety measures ensure that the circuit can also be powered directly from a 5 V source via another diode leaving the lithium battery to act as a backup energy source.

I²C/SPI flash/crypto module

A flash memory (IC3) and a crypto module (IC5) can be mounted on the board that communicate with the controller via the I²C/SPI interface. Both ICs are optional and not required in this application, so can be omitted

3.3 V regulator

The 3.3 V regulator IC2 can supply enough power to run not just the electronics of the LoRa node, but the GPS module as well.

The GPS module

The selection and use of a reliable GPS module took a little time. Originally, the tiny Quectel L96 module (**Figure 3**) was intended to be mounted directly to the baseboard. A design note for the L96 did mention the need for a suitable ground plane and the correct positioning of the module on it. In the first prototype version of the board we were a little lax and did not pay too much attention to this recommendation. In our second iteration we were more careful with the board layout, but GPS reception via the tiny antenna (the component above the label 2.1) was so poor that satellite acquisition took far too long. These long delays are, of course, quite wasteful in terms of the module's overall energy efficiency.

This resulted in the final version of the circuit in which the GPS receiver is not mounted directly on the PCB but fits 'on top' as an independent module. The GT-7U 1728 from Open-Smart (**Figure 4**) was our final choice. This remarkably cheap product from the Far East can be purchased in the Elektor store [3]. Mounting the GPS module on top of the board makes it easier to handle and provides the option of connecting an external antenna, which can improve acquisition time enormously. In addition, components on the baseboard can now be limited to the functional groups required for all the planned applications.

In the Elektor store you will find some basic data about this GPS module tucked away under 'For more information click here'. There is a link to an archive with drivers, the manual, Arduino and MCU code, test tools and a circuit diagram (**Figure 5**). It should be noted that when the GPS module is operated

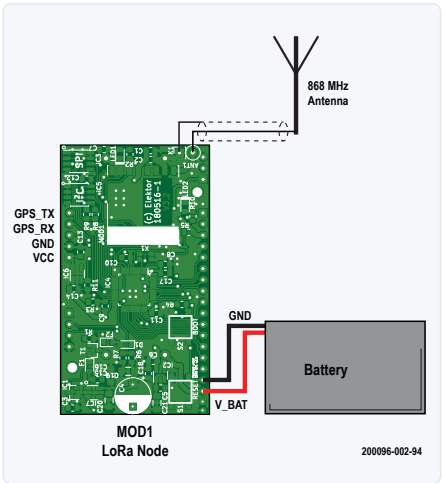


Figure 4: The LoRa Node GPS module.

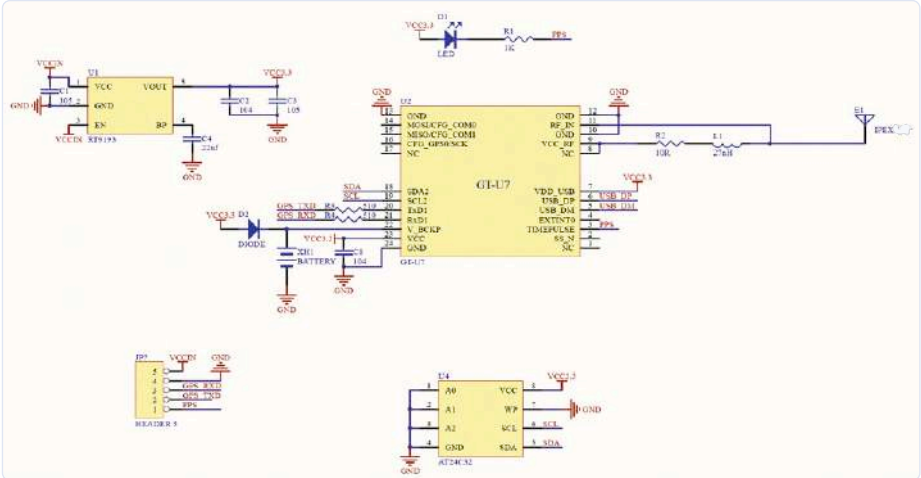


Figure 5: GPS module block diagram.

via the USB port, it is also powered via this port. The USB supply is connected directly to the VCC pin of the module (VCC for the module is specified at 5 V max.), so the tracker would be damaged if a USB connection was used directly. Without any protection, the 5 V of the USB would connect directly to the 3.3 V supply of the board.

Even if the DC voltage at the USB port were within the permitted voltage range (3.0 to 3.6 V), it may conflict with the voltage level supplied from the LoRa node. To prevent this, an 'ideal diode' (IC6) is used. If the voltage on the output side of the MAX40200 is higher than on the input side, the 'ideal diode' blocks teh flow of current. If the module is supplied from the 3.3 V of the baseboard, the 'ideal diode' allows current to flow and produces a voltage drop of only 85 mV so that the GPS module works within its operating parameters. In addition, the controller has the option of completely switching off the voltage to the module via the enable pin, but we do not use this feature.

The GPS module is piggybacked onto the base board (**Figure 6**) so that it is a few millimeters above the components on the LoRa node. In this way, the drone battery, PCB and GPS module make optimal use of the space available in the housing. In addition to the GPS antenna, which is located in the housing, the LoRa antenna must also be taken into account. A miniaturized helix antenna would be so small that it could fit into the housing, but the transmission path would be attenuated by 30 dBm so it would be almost impossible to send a signal to a LoRaWAN gateway. For this reason, an external antenna is fitted and can be just a simple wire or a fancy, waterproof construction as shown in [2].

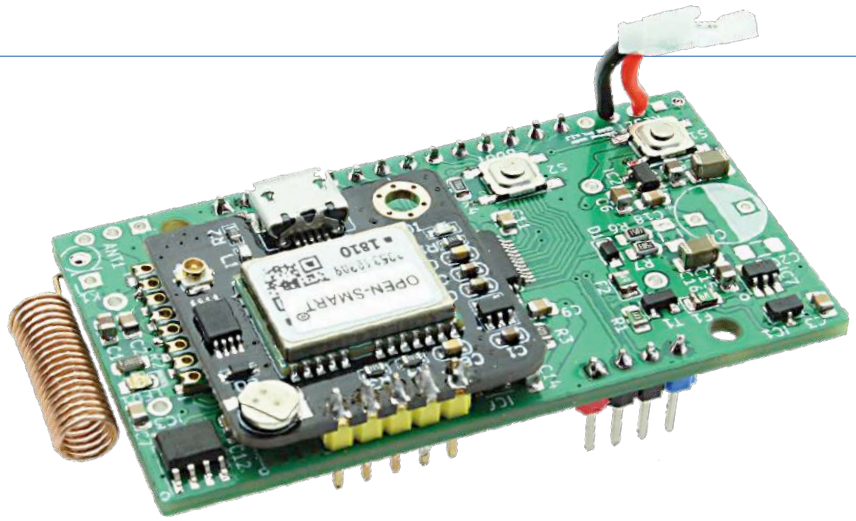


Figure 6: The GPS module sits proud of the PCB by a few millimeters.

The firmware

As can be seen in **Figure 7**, the firmware consists of four main parts: a GPS data decoder, the Arduino LMIC library as the LoRaWAN stack, a Low-Power library and a serial console for configuration. The TinyGPS++ library, which Elektor readers may already be familiar with from the "ESP32 as a Time Server" project [5], decodes the GPS messages. In order for the library to do its work, the incoming data must be read from the serial interface and transferred to the library for decoding. The second USART is used for this data to avoid any possible data collisions with other data streams, such as firmware uploads. This is initialized by calling `HardwareSerial SerialX(USART2);` in the sketch. The pins for controlling the GPS module are already defined in the *Board Description*. With the library used, the controller takes the incoming data from the UART's buffer and transfers it to the library. In order to determine whether the GPS module has acquired a new location, this library polls the current location and evaluates the response. The same LMIC library is used for LoRaWAN communication that already provides the LoRaWAN stack in the LoRa switch and a large number of other LoRaWAN Arduino projects. When using the library, it must be noted that the `os_runloop_once` function must be called as often as possible, and that there should be as few long interruptions as possible. Incidentally, the need to conform to these timing constraints is one reason why it is not so easy to get the LMIC library running on an ESP32.

For the data transport, a payload is transferred to the library and the response that the data has been sent is awaited. In most of the sketches, the configuration of the LoRaWAN library with all access data is defined when compiling, but here it can be changed at runtime. This configuration is performed via the serial command line and can be done, for example, with the serial monitor of the Arduino IDE (**Figure 10**). The basics of this command line come from the temperature-controlled DIY soldering station project [6], where it was used to set some parameters. A few functions have been added for the LoRa GPS Tracker, such as the parsing of HEX strings and a few minor modifications in order to be able to insert new commands more easily. The code itself can not only be used for the STM32; with a few modifications it also runs on the ESP32 and AVR microcontrollers.

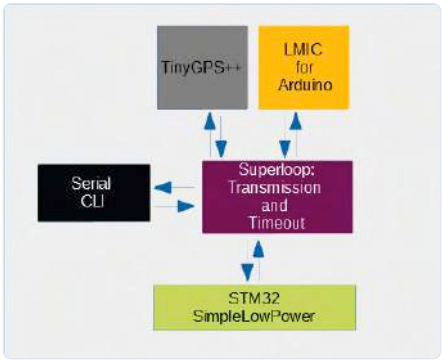


Figure 7: The firmware structure.

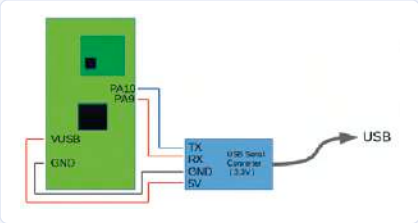
The serial interface works using the first UART and can be addressed via pins PA9 (RX MCU) and PA10 (TX MCU). The interface is configured to 115200 baud, 8 data bits, no parity and 1 stop bit (115200 baud 8N1). In the Arduino IDE the serial monitor communication speed only needs to be set to 115200.

The fourth part is a minimalist low-power library that ensures that the STM32 is put into sleep mode for a defined time and then wakes up again. The contents of the RAM are retained and the internal real-time clock continues to run. Waking up is initiated by an alarm set in the real-time clock. There is also a much more comprehensive low-power library, but it takes up more than 12 kB of flash memory.

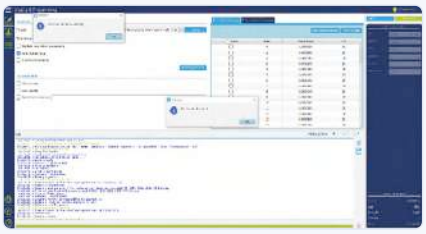
When describing the individual components, it should be remembered that the *LMIC* library limits the size of the send payload to 51 bytes. The shorter the message, the more runtime is available and less time is spent transmit-

FIRMWARE UPLOAD TO THE CONTROLLER

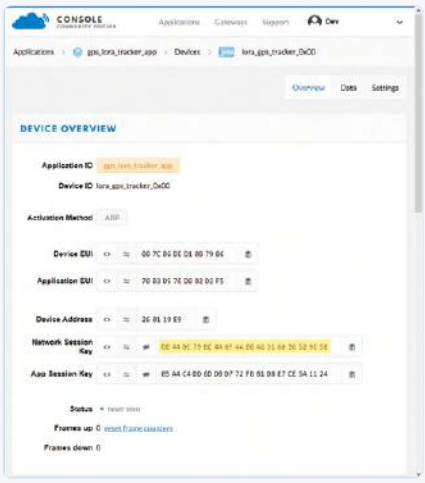
The hardware on its own will, of course, not be any use as a LoRa tracker without the firmware. This is available to download from the project page [9] as a HEX file. To get the firmware into the hardware, a simple USB-to-TTL converter (see Related Products) and four jumper leads (female to female) are all you need.



The HEX file can be programmed using the STM32Cube programmer. To do this, the chip must be switched to bootloader mode by (briefly) pressing the *Boot* and *Reset* buttons at the same. Now you can choose the appropriate serial interface on the right of the display and connect by pressing *Connect*. If no error messages appear, you can begin to program the firmware.



After successful programming, the chip can be brought out of bootloader mode by pressing the *Reset*. It then starts running the firmware. If you don't just want to program this firmware into the chip, you can adapt the code to your own needs in the Arduino IDE and then recompile (and upload) it. The Arduino IDE (from version 1.8.0) supports the Elektor LoRa board of the STM32duino project without the need to modify files. Be aware that if you install the current version 1.8.0 using the Arduino board manager, it contains a number of bugs that can cause the firmware to crash. These errors have however been fixed in the current version available from the Github repository of the STM32duino project.



IEEE 754 float (4Byte)	IEEE 754 float (4Byte)	4 Byte Status

Figure 8: GPS position data packet structure.

Reserved Bit 31 - 16	VBat/2 Bit 15 - 8	Time untill GPS-Fix Bit 7 - 1	GPS Invalid Bit 0

Figure 9: The status byte.

ting to the LoRaWAN, thereby conserving battery power. In the current firmware version, 12 bytes are sent for a location (**Figure 8**), but this is not optimal and could be reduced to 7 bytes.

The first IEEE754-Float value gives the longitude and the second the latitude. The last 4 bytes are used to provide status information (**Figure 9**) (whether the positional data is valid, acquisition time of GPS positional information, battery voltage level). This data can be used as additional information but, since the code is open source, you can easily add your own values.

Energy efficiency to the max

With any mobile device it's essential that great care is taken to keep energy wastage to a minimum and thereby extend the device's operational life between each battery charging cycle. Any part of the circuit that is not currently contributing to the operational status should be switched off or put into sleep mode. Switching off the GPS module is easy. A controller I/O pin switches the supply voltage to the GPS module on or off by using the enable pin of the ideal diode IC6. The LoRa module will automatically go into sleep mode after sending. The STM32 data sheet suggests that current to the controller should be in a range of 68 µA. When we measured the circuit's quiescent current without taking any further measures, we could see it was drawing 0.5 to 0.7 mA, of which the STM32 must be taking a big chunk. Where is all this excess current flowing? The answer was in the unused I/O pins. Any pin that has not been assigned in the code is by default configured as an input. This unused high impedance pin now picks up random signals causing their input stages to switch (see box 'Current consumption of unused I/O pins'). As a countermeasure, you can and should activate the internal pull-up resistors in order to pull the pins to a defined voltage level. In addition, some energy can be saved if pins for the UART to the GPS module are defined as inputs when the module is switched off, otherwise the TX pin from the STM32 could still connect 3.3 V to the I/O pin of the GPS module.

In addition to the STM32, other components also contribute to energy consumption, for example the LDO regulator, the ideal diodes, the LoRa module and the capacitors. We have made a good attempt at reducing energy wastage but we don't want to gild the lily. Realistically speaking, the LoRa GPS Tracker draws a quiescent current of 150 µA at an ambient temperature of 25°C.

When the LoRa GPS Tracker wakes up,

energy consumption increases significantly. The lion's share is taken by the GPS module, which requires around 60 mA, and is therefore largely responsible for the run time of the GPS tracker between charges. Now it's clear why the GPS module should acquire the location as quickly as possible! The LoRa module needs 100 mA peak during data transmission, but this interval is less than one second. The MCU needs about 2.5 mA with an 8 MHz CPU clock and thus provides sufficient computing time for processing the positional data and sending it via the LoRaWAN. If maximum computing power is not required, an attempt could be made to reduce the CPU clock rate as much as possible. The maximum operating time from one charge of the battery is essentially a factor of the repetition rate at which location data is sent, and the GPS acquisition timeout period that prevents the excess power consumption caused by continuous signal acquisition attempts in areas of poor GPS signal.

Parameter settings

A few of the basic settings can be changed using the serial monitor available in the Arduino IDE. This means that it's not necessary to download the firmware again each time one of these settings is changed. Using this feature you can change:

- Time interval for sending of GPS location
- Timeout for GPS position acquisition
- LoRa data rate

These three parameters play key roles in the energy requirements of the LoRa GPS Tracker. The interval for sending the position information is set with `set INTERVAL` and this time can be requested using `get INTERVAL`. The same applies to the timeout for GPS positional data acquisition, which can be set or queried using `set GPSTIMEOUT` and `get GPSTIMEOUT` respectively. The default settings are 15 minutes for the positional data transmission interval, and 10 minutes for acquisition timeout in the event of poor/no GPS reception.

The LoRa data rate can be set in the range SF7 to SF10, where SF7 is the highest data rate and thus the shortest transmission time, and SF10 is the slowest, so that the modulation uses more energy per unit of time. SF7 is the default and ensures quick data transmission, but also results in a shorter range. When the value of SF7 is increased, the range is increased but, if the tracker is in motion, it also means that there may be disruptions in data transmission: a double-edged sword! It also influences the number of times positional data can be sent per day; the *The Things Network* allows 30 seconds of broadcast time per day; with a setting of SF7 the transmission of a location message takes 62 ms; with SF10 it is 412 ms. That means with SF7 we can send about 450 messages per day, but with SF10 only around 70 messages.

In order to be able to send data via LoRaWAN, the end-device key must be entered. The LoRa GPS Tracker uses the ABP mode. This brings us to the configuration requirements for The Things Network.

Configuration in The Things Network

An introduction to The Things Network was already provided in detail in the March/April [4], edition of Elektor, so we will not go over the same ground again. After creating a user account and logging in, an application must first be created by clicking on *Applications.* Then, on the following page, click on *add application* to create an application to which the LoRa GPS Tracker can be assigned. In the following dialog a unique name (*gps_lora_tracker_app*) and a description (*Test Sensor Nodes for Developement*) of the application must be given.

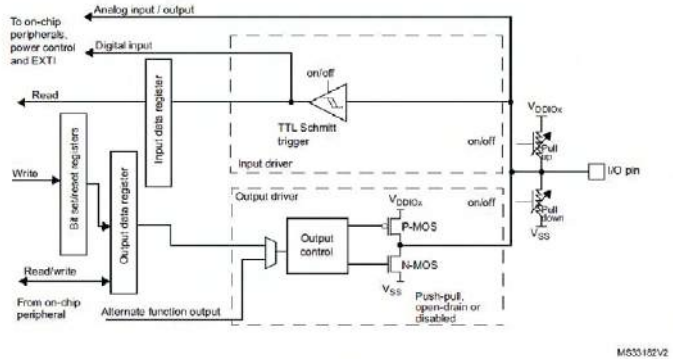
LoRa devices are always assigned to an application so that the data arriving in *The Things Network* can be decoded and further processed appropriately. For the LoRa GPS Tracker, this does not play a role at first, as the data is processed by a server (a Raspberry Pi) running Node-RED.

Now the application has now been created, it's necessary to register our LoRa GPS Tracker device. Selecting *Register device* will open a dialogue where you can register the LoRa GPS Tracker parameters. A unique name for each LoRa GPS Tracker is assigned before the window is closed by clicking on *Register*. Now, under *Settings* for the registered device, it is necessary to change the activation method from "Over the Air Activation" (OTAA) to "Activation by Personalization" (ABP), which allows code running in the LoRa GPS Tracker to talk to The Things Network.

After *Save*, the required information (*Network Session Key*, *Application Session Key* und *Device Address*) can be set up in the LoRa GPS Tracker by using the serial console.

You don't need to laboriously type in the three keys; simply select them using the clipboard symbol in the The Things Network (*Device Overview*) and copy to the serial monitor (**Figure 10**). During transfer, the keys are converted internally and saved in EEPROM. Once all parameters have been entered, the LoRa GPS Tracker is ready for use.

CURRENT CONSUMPTION OF UNUSED I/O PINS



The structure of a typical I/O pin of an STM processor is described in the Reference Manual RM0091. It features a Schmitt trigger stage to ensure noisy input signals will be converted to a clean, digital 1 or 0 by using voltage thresholds and a hysteresis band. Its input impedance is very high; if the pin is not connected to anything it acts as an antenna and will pick up signals which cause the input to switch between 1 and 0. This results in a small amount of current will being drawn. With the unused pins of this board this equated to a measured current consumption of 0.5 to 0.7 mA.

ALTERNATE PIN FUNCTIONS: FUN FOR THE ENTIRE DEVELOPMENT TEAM!

Alternate Pin Functions have only fairly recently found their way into the world of AVR microcontrollers, while they have long been a standard feature in Cortex-M-based controllers such as the STM32. For all pins with alternative functions, you should decide early on in the board layout phase whether they will be routed to a connector, even if you are not planning to use them immediately. It would be annoying if, after completing the project, you notice that a pin that could be used as the serial clock for a second SPI interface but it has not been allocated a connector pin. This is where coordination between software and hardware developers is essential; it is prudent to ensure as many pins with *Alternate Pin Functions* as possible are made accessible to give some level of future-proofing to the design.

From LoRa GPS Tracker to the map

After the LoRa GPS Tracker has been set up it sends data to the LoRaWAN that will need to be processed further before the location can be displayed on a map. This can be done with the help of Node-RED; here a Raspberry Pi can be used as the computer (although not necessarily — you could also make use of a virtual machine or a computer on which Node-RED is already installed). There is no Node-RED in the Raspberry Pi OS (previously known as Raspbian). The installation (with Raspbian, you must have Raspbian Jessie as a minimum version) and the setup of Node-RED is described in detail on the website [7]. An introduction to Node-RED has been already covered in a previous Elektor article [8].

At the moment, data from LoRa GPS Tracker is being sent to The Things Network (but not saved there) and then forwarded to an available system. The MQTT protocol, which has already been used in other Elektor projects, is used as the interface for collecting the data. The advantage here is that there are ready-made components for The Things Network in Node-RED, for which only a few parameters have to be specified. This saves a lot of work! For the *UplinkMessages* node, a connection with the application must first be established. The *Application ID* and the *Access Key* are required from the *Console* of The Things Network. This data is then used to identify you to the server as a legitimate user

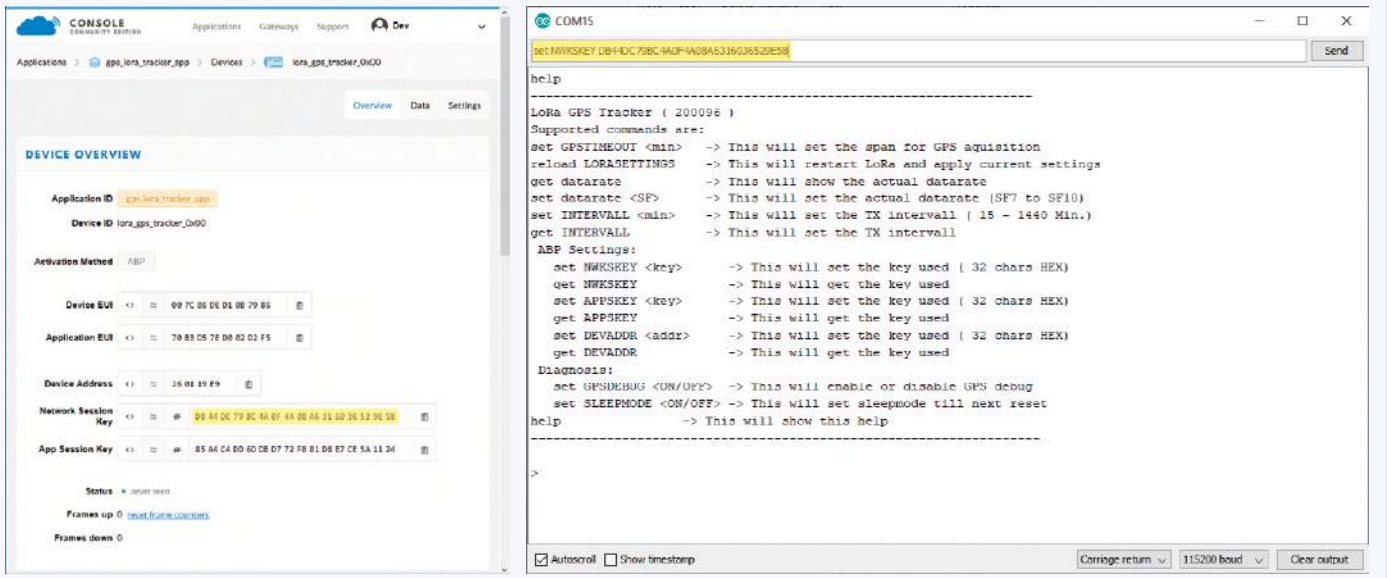


Figure 10: Copy the keys from the **Device Overview** *in the serial monitor.*

Figure 11: Transmission of TTN data in Node-RED (Node_Red_TTN_node_Info and TTN_Access).

Questions or Comments?
You are welcome to contact the author at mathias.claussen@elektor.com.

in order to receive the data. This data is now copied from the *TTN Console* and entered in Node-RED at the appropriate places at the *UplinkMessages* node (**Figure 11**).

Once the node is selected, a new dialog is opened with the pencil symbol next to *App*. Data from the *TTN Console* can now be entered here using Copy and Paste, transferred by clicking *Update,* and then closing the settings with *Done*. As with any other change, a *Deploy* is now required. The node should then show *Connected* in the overview (**Figure 12**).

Now new messages can be received from the node for further processing. These messages not only contain the user data sent by the LoRa GPS Tracker, but also other information about the gateway which received the message. With this information the user can determine the location of the gateway which will prove to be very useful in a moment.

If the LoRa GPS Tracker was not able to determine an exact position, it still sends data in which it is also noted that no valid GPS location information is available. An attempt can then be made to at least determine the position of the gateway that received the data. This means that the exact location of the tracker cannot be determined, but it will be somewhere within a 2 km (approx) radius of the gateway location (**Figure 13**).

The display itself is a World-Map extension, an interactive map that can be opened in a browser. In the enclosed Node-RED example, this can be acessed via: `:1883/worldmap` (**Figure 14**).

The source of the map data is the OpenStreetMap project. Locations can be set for objects displayed on this map, for which an icon, a radius and a few other parameters of its location are indicated.

One disadvantage is that the locations are not retained after the browser is closed. Therefore, the last 2048 data points that were entered on the map are stored in the RAM of the Node-RED server. When the map is called up again, these are also displayed on the map. This enables routes and changes in position to be traced over a longer period of time, and they are also stored (for each individual tracker) in an Excel-compatible .CSV file. The display on the world map is, therefore, not a completely finished solution for the presentation of the position data, but it should serve to show the flow of data from the LoRa GPS Tracker to the user and also function as a template for your own ideas.

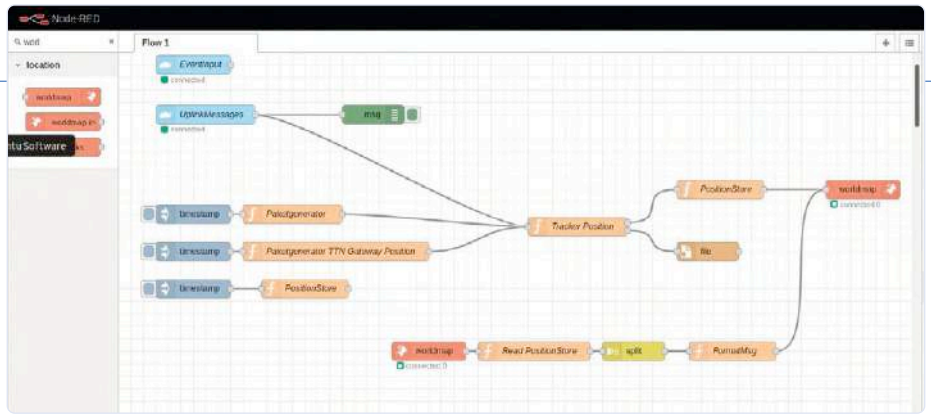


Figure 12: Flow chart in Node-RED.

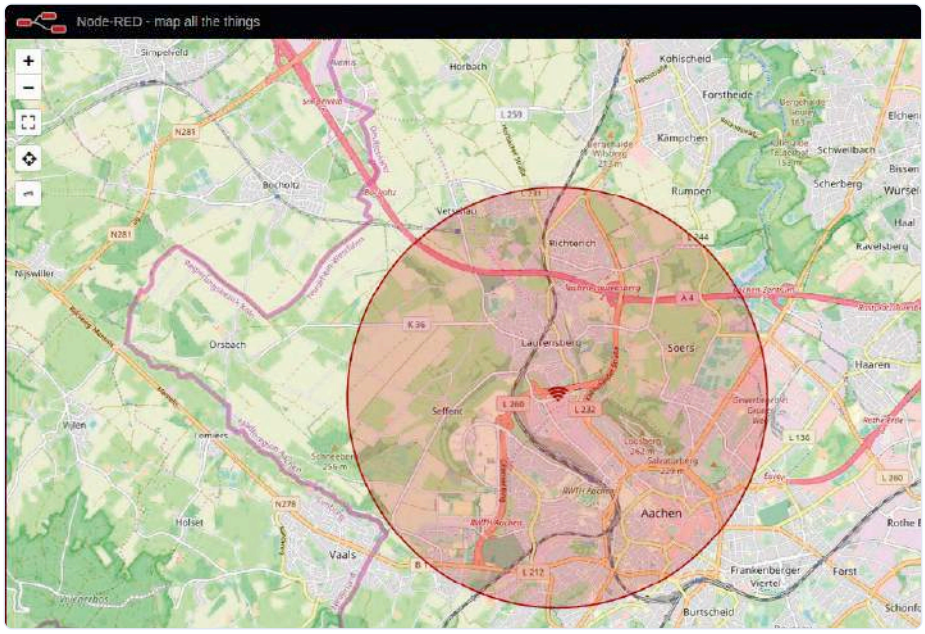


Figure 13: Even without GPS reception we can see the node position on the map using the gateway's location (© OpenStreetMap contributors).

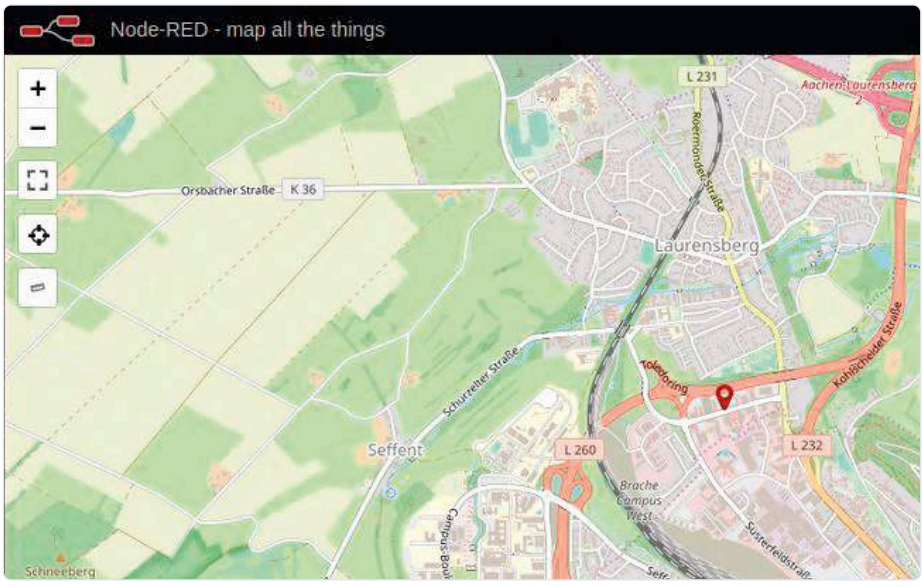


Figure 14: On the map you can see the exact location of the GPS Tracker in the Elektor Lab — now you know where we live! (© OSM contributors).

COMPONENT LIST

Required Modules:

Elektor LoRa Node (PCB available from Elektor*, parts list and KiCAD-files from [10], firmware downloadable from [10])
RFM95 Ultra LoRa Transceiver Module (868/915 MHz) *
OPEN-SMART GPS – Serial GPS Module *
SMA cable with pigtail and solder tail
868 MHz antenna SMA
Hammond 1551K enclosure
Battery cable and plug (2 mm pitch) (Molex 51005)
Rechargeable battery (see text)

* *See 'Related Products' box*

RELATED PRODUCTS

- **Elektor LoRa Node bare PCB 180516-1**
  www.elektor.com/elektor-lora-node-bare-pcb-180516-1
- **RFM95 Ultra LoRa Transceiver Module (868/915 MHz)**
  www.elektor.com/rfm95-ultra-lora-transceiver-module-868-915-mhz
- **OPEN-SMART GPS – Serial GPS Module**
  www.elektor.com/open-smart-gps-serial-gps-module-for-arduino-apm2-5-flight-control
- **USB to TTL Converter UART Module CH340G (3.3 V/5.5 V)**
  www.elektor.com/ch340-usb-to-ttl-converter-uart-module-ch340g-3-3-v-5-5-v
- **Book: *Programming with Node-RED***
  www.elektor.com/programming-with-node-red
  www.elektor.com/programming-with-node-red-e-book

Summary

LoRaWAN offers an interesting platform to track the location of objects. With the solution presented here, the geographical location of an item can be determined, transmitted via LoRaWAN and displayed on a separate computer. Thanks to the open source nature of the hardware and software, anyone who wants to can take a look behind the scenes and adapt parts of the project to meet their own needs.

The complete data for the LoRa GPS Tracker project is available as a KiCad project on GitHub [10] and can be used under the OSHL V1.2 license. The software code and the Node-Red project to display the location can also be found on GitHub. In addition to its use as a GPS tracker, there are other applications for the Elektor LoRa node which we are already working on, but progress on this front has inevitably been slowed by the measures introduced to contain the spread of COVID19.

200096-03

Contributors

Idea, development, text and images: **Mathias Claußen**
Text: **Rolf Gerstendorf**
Translation: **Martin Cooke**
Editing: **Stuart Cording**
Layout: **Giel Dols**

WEB LINKS

[1] **C. Vossen, "ElekTrack," Elektor 10/2007:** www.elektormagazine.com/magazine/elektor-200710/18645
[2] **Expensive roaming charges**: www.bbc.com/news/world-europe-50180781
[3] **M. Claußen and L. Lemmens, "The Elektor LoRa Node," Elektor March/April 2020:** www.elektormagazine.com/magazine/elektor-141/57130
[4] **M. Claußen, "My First LoRaWAN" Elektor March/April 2020:** www.elektormagazine.com/magazine/elektor-141/57159
[5] **ESP32 as a Time Server**: www.elektormagazine.com/magazine/elektor-100/50916
[6] **DIY Temperature Controlled Solder Station**: www.elektormagazine.com/magazine/elektor-70/42342
[7] **Node-RED Getting Started**: https://nodered.org/docs/getting-started/raspberrypi
[8] **Starting with Node-RED**: www.elektormagazine.com/magazine/elektor-151/58744
[9] **Project page for this article**: www.elektormagazine.com/200096-03
[10] **KiCad files on GitHub**: https://github.com/ElektorLabs/180516-Elektor_LoRa_Node

Programming **the Finite State Machine**

with 8-bit PICs in Assembly and C

By **Andrew Pratt** (United Kingdom)

This installment of *Elektor Books* presents three 'learning-by-doing' sections and two 'nice to know' paragraphs excerpted from the book *Programming the Finite State Machine with 8-Bit PICs in Assembly and C*. Aimed at Microsoft Windows and Linux users, the book is based on a simple toolchain including an FTDI serial lead as the programmer, and open-source 'gpasm' as the assembler. Here we show the PIC assembly-language approach to an LED Flasher, Multiple FSMs in one program, and a 7-segment LED driver.

A more complicated LED flasher

One of the criticisms of assembly language programs is that they can be difficult to read. This can be true even if you have only just created them! Having a standard layout for your program makes this is a lot easier. Also, the finite state diagram gives an overview of the whole operation. The assembly code is in separate blocks that are the *states*. You will find that by referring to the state diagram, following the assembly code is straightforward.

As an example of a more complicated program, **Figure 2-5** is the state diagram for the same LED but now it flashes on and off with the same intervals as before but only for 3 flash cycles (reference is to a program described in a previous section of the book — *Ed.*). It then flashes 10 times faster for 10 cycles before going back to the slower speed. Note there are four states with six transitions. There are off and on states for fast flashing and off and on states for slow flashing. The whole diagram can be divided into four quadrants with two dividing lines: one between fast and slow and one between on and off.

Figure 2-6 shows the oscilloscope trace for the voltage across the LED.

The program is shown in **Listing 1**. By referring to the state diagram, you should be able to understand what is happening. There is one new thing to explain: the use of the $ sign. This means the address of the current line, so instead of jumping to a label, we can jump so many lines. Writing *GOTO $+3* will jump forwards 3 lines, for example.

Running more than one 'machine' in a program

The two FSM (finite state machine) programs have so far had a small block of code for each state and have run that code over and over until some input event causes a jump to another state. I will now suggest a way of running more than one machine at once in a program: This is like doing two things at once by performing part of one task then part of another task then back to the first and so on. This is time multiplexing, the CPU in the microcontroller can only carry out one instruction at a time so if tasks are to be carried out together they must share the processor's time. Unlike timers and other peripheral parts of the PIC that are separate hardware entities that can truly run concurrently, if there is more than one task in the program, they only appear to run concurrently. They are

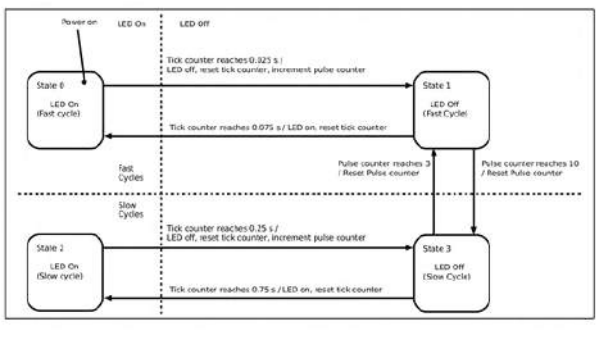


Figure 2-5: State diagram for the LED-flashing 'Cycles of Cycles' program.

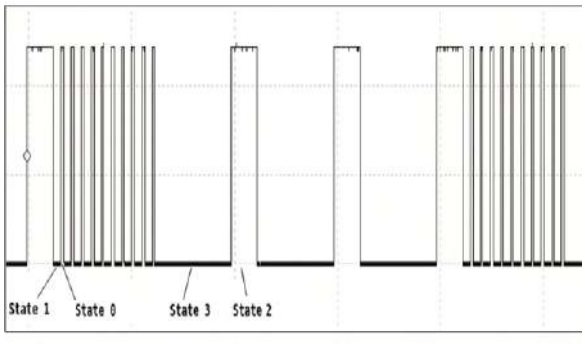


Figure 2-6: Oscilloscope trace for the 'Cycle of Cycles' program.

Listing 1: 'Cycle of Cycle' program (prog_02_02.asm).

```
; prog_02_02.asm
; Page numbers refer to the data sheet DS40001413E.
; Tables refer to the book.

LIST     P=12f1822
#INCLUDE <p12f1822.inc>

    RADIX DEC                    ; Default numbers are to base 10.

    __CONFIG 0X8007, (_FOSC_INTOSC & _WDTE_OFF & _PWRTE_OFF & _CP_OFF & _BOREN_OFF & _IESO_OFF & _FCMEN_
   OFF )
    __CONFIG 0X8008, (_LVP_ON)


    CBLOCK 0x70                  ; In commom RAM, accessible in all banks.
    TICK_COUNTER
    PULSE_COUNTER
    ENDC

    ORG 0X00
    GOTO START

    ORG 0X04                     ; Interrupt vector.
    BCF INTCON, TMR0IF           ; Clear the tmr0 overflow interrupt flag. INTCON is in all banks.
    INCF TICK_COUNTER, F         ; Increment by one counter_off.
    RETFIE                       ; Return from interrupt
START
    MOVLB 1                      ; To access TRISA, OSCCON, OPTION_REG.
    BCF  TRISA, 2                ; Configure PORTA bit 2 (chip pin 5) as an output
    MOVLW 0x70
    MOVWF OSCCON                 ; Page 65 32MHz. See Table 2-1 in the book.
    MOVLW 0xD7
    MOVWF OPTION_REG             ; Page 164 in the data sheet and Table 2-2 in the book.
    MOVLW 0xE0
    MOVWF INTCON                 ; Page 86 in the data sheet and Table 2-3 in the book.
    MOVLB 0                      ; To access PORTA for the rest of the program.
    BSF PORTA, 2                 ; Inialise LED on;
;-----------Machine States.
S0                               ; LED on. Fast cycle.
    MOVLW 3
    SUBWF TICK_COUNTER, W        ; C in STATUS is set when TICK_COUNTER >= to 3.
    BTFSS STATUS, C              ; Skip the next instruction if C in STATUS is found set.
    GOTO S0                      ; C in STATUS found not set. Stay in this state.
    BCF PORTA, 2                 ; Turn LED off
    CLRF TICK_COUNTER            ; Reset the tick counter.
    INCF PULSE_COUNTER, F        ; Add 1 to PULSE_COUNTER and put the result in PULSE_COUNTER.
                                 ; Continue to S1

S1                               ; LED off. Fast cycle.
    MOVLW 10
    SUBWF PULSE_COUNTER, W       ; C in STATUS is set when PULSE_COUNTER >= to 10.
    BTFSS STATUS, C              ; Skip the next instruction if C in STATUS is found set.
    GOTO $+3                     ; Have not reached 10 pulses jump to ticks check.
    CLRF PULSE_COUNTER           ; 10 Pulses have been counted, reset PULSE_COUNTER.
    GOTO S3                      ; Transition to S3.
    MOVLW 9                      ; Load W with 9 for ticks check.
    SUBWF TICK_COUNTER, W        ; C in STATUS is set when TICK_COUNTER >= to 9.
    BTFSS STATUS, C              ; Skip the next instruction if C in STATUS is found set.
    GOTO S1                      ; C in STATUS found set, 9 ticks counted.
    CLRF TICK_COUNTER
    BSF PORTA, 2                 ; Turn LED on.
    GOTO S0

S2
    MOVLW 31
    SUBWF TICK_COUNTER, W        ; C in STATUS is set when TICK_COUNTER >= to 3.
    BTFSS STATUS, C              ; Skip the next instruction if C in STATUS is found set.
    GOTO S2                      ; C in STATUS found not set.
    BCF PORTA, 2                 ; Turn LED off
```

```
    CLRF TICK_COUNTER           ; Reset the tick counter.
    INCF PULSE_COUNTER, F       ; Add 1 to PULSE_COUNTER and put the result in PULSE_COUNTER.
                                ; Continue to S3

S3                              ; LED on. Slow cycle.
    MOVLW 3
    SUBWF PULSE_COUNTER, W      ; C in STATUS is set when PULSE_COUNTER >= to 3.
    BTFSS STATUS, C             ; Skip the next instruction if C in STATUS is found set.
    GOTO $+3                    ; ave not reached 3 pulses jump to ticks check.
    CLRF PULSE_COUNTER          ; Reset PULSE_COUNTER.
    GOTO S1                     ; Transition to S1.
    MOVLW 91
    SUBWF TICK_COUNTER, W       ; C in STATUS is set when TICK_COUNTER >= to 9.
    BTFSS STATUS, C             ; Skip the next instruction if C in STATUS is found set.
    GOTO S3                     ; C in STATUS found set.
    CLRF TICK_COUNTER
    BSF PORTA, 2                ; Turn LED on.
    GOTO S2

    END
```

running as threads, with each thread sharing the processor's time. To achieve this multi-threading, the first machine will execute the block of code for its current state and then the block of code for the other machine's current state and then back to the first machine. These different machines are independent. They just happen to use the same CPU by time-sharing. They can also share memory to access each other's data.

In the examples so far, the state was defined by being in a certain block of code now the state is going to be defined by the value stored in a register. Let the register that holds the state of machine 0 be *STATE_M0* and for machine 1 be *STATE_M1*. **Figure 2-7** shows how this program flow will be.

To control this switching between states, we can use the instruction *BRW*. The *BRW* instruction is a *branch-with-W* meaning that the program counter will increase by the value held in W. In other words, you can do a calculated jump.

After a block of state code is executed, the program will jump back to the switch for the next machine where the *BRW* instruction will

Listing 2: Fictitious program with 2-state machine 0, and 3-state machine 1.

```
SWITCH_M0
     MOVFW STATE_M0       ; Moves the value of STATE_M0 to W.
     BRW                  ; The program will jump from here depending on W.
    GOTO M0_S0            ; The jump will be to this line if W = 0.
     GOTO M0_S1           ; The jump will be to this line if W = 1.

SWITCH_M1
     MOVFW STATE_M0       ; Moves the value of STATE_M0 to W.
     BRW                  ; The program will jump from here depending on W.
    GOTO M1_S0            ; The jump will be to this line if W = 0.
     GOTO M1_S1           ; The jump will be to this line if W = 1.
    GOTO M1_S2            ; The jump will be to this line if W = 1.
M0_S0
     -------------        ; code to decide if a change of state is needed if so change STATE_M0. GOTO
     SWITCH_M1            ; Go to the other machine's switch.

 M0_S1
     -------------        ; code to decide if a change of state is needed  if so change STATE_M0.
    GOTO SWITCH_M1        ; Go to the other machine's switch.

 M1_S0
     -------------        ; code to decide if a change of state is needed  if so change STATE_M1.
    GOTO SWITCH_M0        ; Go to the other machine's switch.

 M1_S1
     -------------        ; code to decide if a change of state is needed  if so change STATE_M1.
    GOTO SWITCH_M0        ; Go to the other machine's switch.

M1_S2
     -------------        ; code to decide if a change of state is needed  if so change STATE_M1.
    GOTO SWITCH_M0        ; Go to the other machine's switch.
```

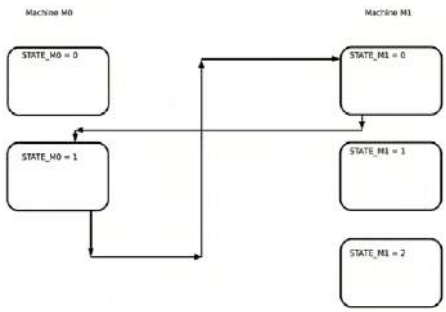


Figure 2-7: Example of program flow between two machines in the same program.

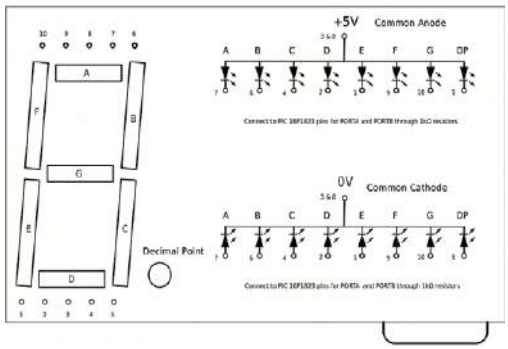


Figure 2-8: Connections for typical 7-segment LED displays.

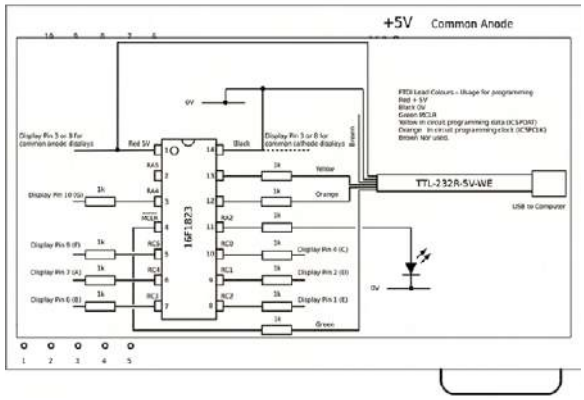


Figure 2-9: Connections for the PIC 16F1823 and 7-segment LED display.

go to the next line if the state value is 0 or skip a line if it is 1 and skip two lines if the value is 2. The program flow is not trapped in the state blocks but routes round via the switch code. To run two machines as threads, there will be two state variables: *STATE_M0*, *STATE_M1*, and two switches. **Listing 2** shows a simplified code snippet of this fictitious program, with machine 0 having two states and machine 1 having 3 states.

Notice how the program flow alternates between machines when leaving a block of state code.

Driving a 7-segment LED display

For a practical demonstration of the use of the switches to handle two machines in the same program, this example will have a flashing LED and a 7-segment display that counts the number of flashes. There is a choice of a liquid crystal or an LED display. LED displays are easier as they can be driven with direct current, while liquid crystal displays require alternating voltage. LEDs have the disadvantage that they require several milliamps per segment. There are lots of 7-segment LED displays available. I am using a single digit display. **Figure 2-8** shows the physical connections for one typical pattern. The displays come in two distinct arrangements, common cathode, and common anode. With common cathodes, the cathodes must be connected to 0 V. With common anode, anodes must be connected to +5 V. The segment connections are then connected to the output pins of the PIC. The logic of the output code has to be designed to suit the polarity of the display. It would be easier if you use a display with this pinout arrangement, but if not you will have to work out your own wiring. The LA-601VB common anode display that can be supplied by RS Components is similar to the one I used. Farnell has a similar one (part number SA52-11SRWA). As there are a total of 8 LEDs to drive, the 16F1823 PIC is used. This has an extra IO port, PORTC. **Figure 2-9** shows the 'circuit diagram' for connections to the PIC.

The next part of the design process is to draw the finite state machine diagram (see **Figure 2-10**). Machine M1 is the display driver. On power-on, the LED is initialised to on, and then machine M0 starts in state 0. After 0.25 s, the timer expires and machine 0 transitions to state 1. The timer is reset, the LED turned off, the pulse counter is incremented by 1 and a signal called UPDATE is set to high. This UPDATE signal is used by machine 1. While in state 1, when the timer expires at 0.75 s, the machine transitions to state 0 and the timer resets and its LED turned on. This continues indefinitely. When in state 1, if the pulse counter has reached 10, there is a transition to the same state. This is to reset the pulse counter to zero. Meanwhile, machine M1 is running independently but time sharing the CPU. Machine M1 starts in state 0 and the display will be blank. When the UPDATE signal goes high, the machine transitions to state 2 and resets the UPDATE signal to low. Depending on the value of the variable N, the machine transitions back to state 1, driving on and off the necessary segments of the display. The machine will remain in state 0 waiting for another update signal. The output details shown for machine M1 are for a common-anode display. The comments in the code show the alternatives for a common-cathode display.

Due to space contraints, program 'prog_02_03.asm' can't be printed here but you can get it quasi instantly from the book's web page [1]. Most of its operation has been already explained, except for one thing you have to watch out for: the READ MODIFY WRITE problem that can happen when using the *BSF* and *BCF* instructions. Up to now, we have only been setting one output pin by setting the bit value on *PORTA*. This works if you are only working with one output pin on a port but something strange happens when trying to alter an output shortly after altering a different pin. To understand the problem, we have to look at basic electrical theory regarding charging and discharging capacitors, along with how the PIC handles changing the value of one pin at a time. When the PIC writes to outputs it can only write to a complete byte or eight bits, so to change one bit it has to read the state of the pins, modify the one it wants to alter, and then write back to the outputs. The

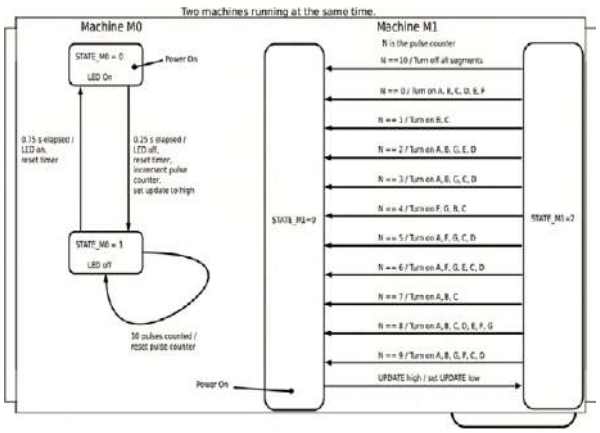


Figure 2-10: State diagram for program 02-03 (not printed here but available at [1]).

problem is that say *PORTA* bit 5 (physical pin 2) has been changed from 0 to 1, and then in the next instruction, you want to change the value on *PORTA* bit 4 (physical pin 3). The PIC reads the values on the PORTA pins — you might expect the value on bit 5 to be high as you have just set it, but NO because in the real world the electrical circuit could be capacitive and it will take time for the voltage to rise, resulting in the value of bit 5 reading as low. When the outputs are written back to port, bit 5 (pin 2) gets set back to low. This is an example of where testing in a simulator can let you down. Generally, when using *BCF* and *BSF* you should write to the latch registers for the ports (*LATA* and *LATC*). If however, you are writing to the whole port as a register, for example, using *MOVWF PORTA*, then this is safe.

The differences between the PIC 12F1822 and the 16F1823

The obvious difference is that the 12F1822 has 8 pins, while the 16F1823 has 14. This is because the 16F1823 has two I/O ports. Apart from this difference, programming is almost the same. You must quote the correct include file at the top of your source code and state the correct controller for your chip. Below are the relevant lines:

```
LIST P=12F1822
#INCLUDE <p12f1822.inc>

LIST P=16F1823
#INCLUDE <p16f1823.inc>
```

The advantage of a smaller chip is that it takes up less space on a circuit board and there are fewer pins to solder if you don't need the extra port. You must consult the datasheet as there are always traps to fall into. There will be occasions when you will stare at your code trying to understand what is wrong, but the answer will be in the datasheet.

Interrupts and state diagrams

I have not tried to show interrupts in any way on the state diagrams as they do not change the state of the machine. When an interrupt occurs, the main program is suspended: see a previous section in the book. Therefore no change in state occurs until after the return from interrupt and then only if the interrupt has changed the input to the state machine (such as an increase in a counter changing the result of comparison) .

200447-01

Contributors

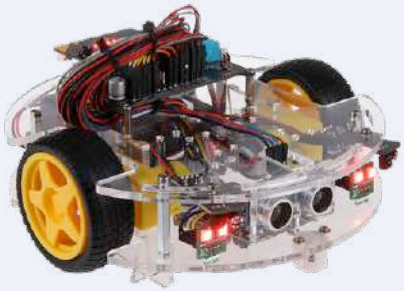
Author: **Andrew Pratt**
Layout: **Giel Dols**
Editor: **Jan Buiting**

Questions or Comments?

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

WEB LINK

[1] **Three .asm program files**
(on web page, scroll down to: Downloads):
www.elektor.com/programming-the-finite-state-machine



Joy-Car Robot (incl. BBC micro:bit)

Price: €84.95

Member Price: €76.46

www.elektor.com/19408

Get Started with the SensorTile.box Bundle

Price: €69.95

Member Price: €62.96

www.elektor.com/19404

Peak UTP05E Network Cable Analyser (Ultimate Atlas IT Kit)

Price: €259.00

Member Price: €233.10

www.elektor.com/19373

Dragino PG1301 LoRaWAN GPS Concentrator for Raspberry Pi (868 MHz)

Price: €124.95

Member Price: €112.46

www.elektor.com/19367

Hexadoku The Original Elektorized Sudoku

Traditionally, the last page of *Elektor magazine* is reserved for our puzzle with an electronics slant: welcome to Hexadoku! Find the solution in the gray boxes, submit it to us by email, and you automatically enter the prize draw for one of five Elektor store vouchers.

The Hexadoku puzzle employs numbers in the hexadecimal range 0 through F. In the diagram composed of 16 × 16 boxes, enter numbers such that **all** hexadecimal numbers 0 through F (that's 0-9 and A-F) occur once only in each row, once in each column and in each of the 4×4 boxes (marked by the thicker black lines).

A number of clues are given in the puzzle and these determine the start situation.

Correct entries received enter a prize draw. All you need to do is send us **the numbers in the gray boxes**.

SOLVE HEXADOKU AND WIN!

Correct solutions received from the entire Elektor readership automatically enter a prize draw for five Elektor store vouchers worth **€50.00 each**, which should encourage all Elektor readers to participate.

PARTICIPATE!

Ultimately December 8, 2020, supply your name, street address and the solution (the numbers in the gray boxes) by email to: **hexadoku@elektor.com**

PRIZE WINNERS

The solution of Hexadoku in edition 5/2020 (September & October) is: **35904**.

The store vouchers have been awarded to:

Charlotte Mies (The Netherlands), Jean-Marie Mahieu (Belgium), Roberto Visentin (Italy), Herman Pusch (Germany) and Neil Wood (United Kingdom).

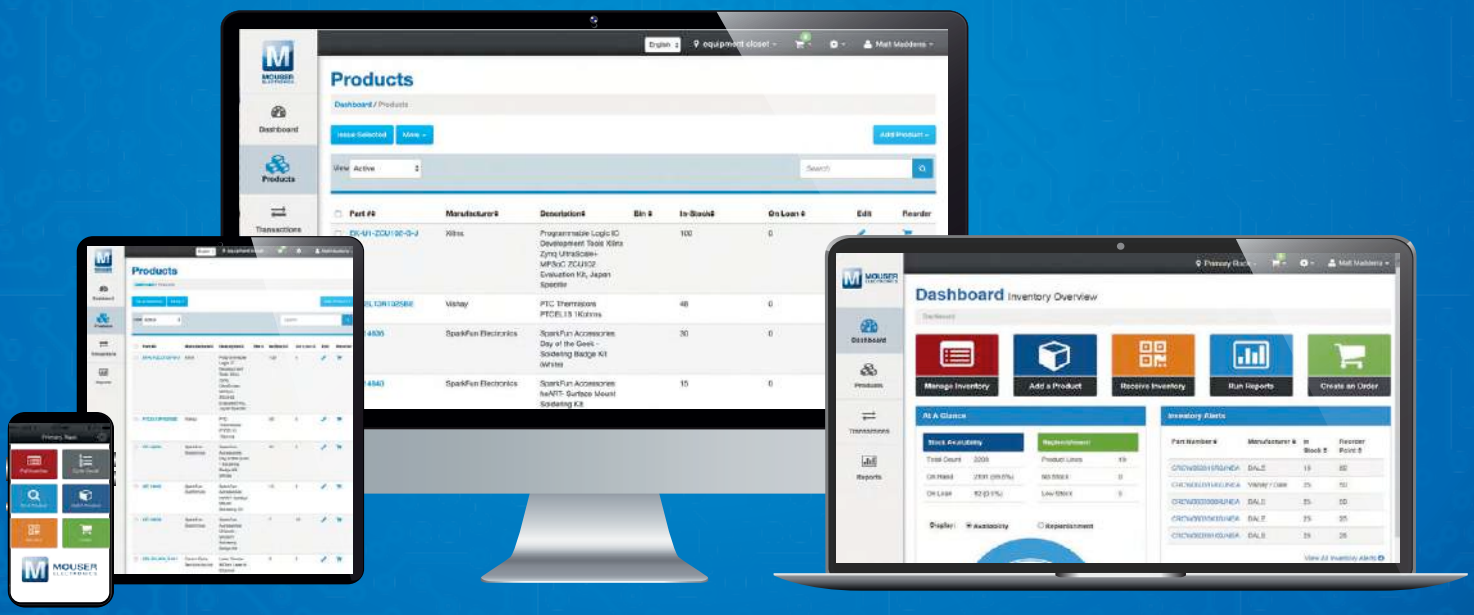
Congratulations everyone!

	B	3			E	4	8		2		7	1	C		
		6			B		2		3			F			
C						6	A	B		F					2
9								C	8	0			5		
	5	B	7			D	0			2		4	E	C	
		8				1				6	B			7	
E	0						B				8		6		A
	3			A			C		D	1				F	0
1	C				4	B		3			A			2	
6		A		C				7						D	3
	8			6	9				1				A		
	2	5	B		1			F	4			9	7	0	
		2			A	3	5								C
B					2		6	1	5						D
			3			E		2		A			0		
		0	D	B		7		8	6	9			2	3	

0	F	7	6	8	4	D	B	1	E	A	5	9	C	2	3
E	9	1	A	3	6	F	7	8	B	C	2	5	0	4	D
8	2	C	3	9	A	0	5	F	7	D	4	1	B	E	6
B	D	4	5	1	2	C	E	3	6	9	0	F	7	A	8
9	3	6	8	B	5	2	A	E	1	0	C	D	F	7	4
A	4	5	D	0	7	1	3	9	F	B	8	6	2	C	E
1	0	B	E	C	8	6	F	4	D	2	7	3	5	9	A
7	C	F	2	D	9	E	4	A	3	5	6	B	8	0	1
C	1	2	7	E	D	4	8	B	0	6	9	A	3	5	F
D	A	8	9	F	B	5	0	C	2	4	3	E	1	6	7
F	E	0	B	6	3	7	9	D	5	1	A	C	4	8	2
5	6	3	4	2	1	A	C	7	8	E	F	0	9	D	B
2	8	D	C	7	F	B	6	0	9	3	E	4	A	1	5
3	5	9	0	4	E	8	1	6	A	7	B	2	D	F	C
4	7	E	1	A	0	3	2	5	C	F	D	8	6	B	9
6	B	A	F	5	C	9	D	2	4	8	1	7	E	3	0

The competition is not open to employees of Elektor International Media, its subsidiaries, licensees and/or associated publishing houses.

Free inventory management tool



Pull Inventory

Cycle Count

Find A Product

Add a Product

Receive

Order

mouser.co.uk/inventory-management

MOUSER ELECTRONICS

PROTEUS DESIGN SUITE

High Speed Design Features

- Differential Pair Routing
- Automatic Phase Matching
- Length Matching / Net Tuning
- Use for USB, Ethernet, DDR3 etc.

Performance without the price premium.

Find out more and
configure your package on our
website or call the team on (+44)1756 753440

Labcenter Electronics

www.labcenter.com