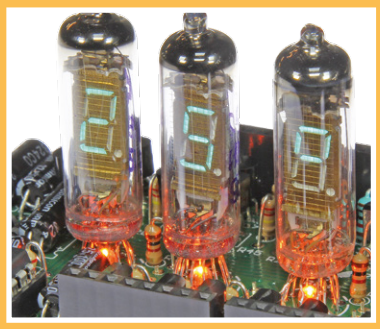
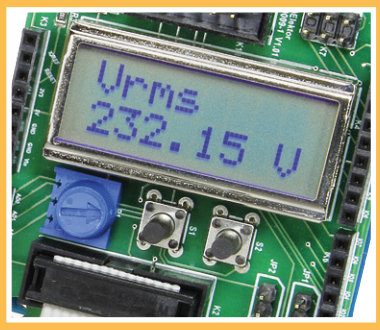
Edition 5/2015 | September & October 2015
www.elektormagazine.com

elektor

LEARN • DESIGN • SHARE



VFD Shield
for Arduino



AC/DC
Power Meter

AAA Cell Replacement
on USB

Android I/O Board

Control embedded electronics from your Android phone or tablet

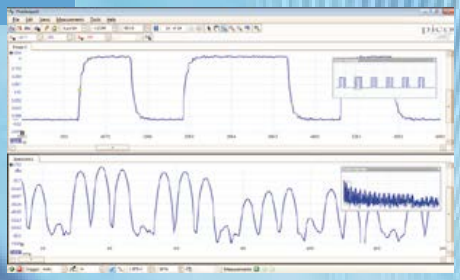
In This Edition:
11 Labs Projects
2 Reader's Projects
3 Programming Courses
7 New Modules & PCBs
1 Review
and lots more

AC/DC Power Meter • **Android I/O Board** • Arduino Sound-Level Protector • **ARM Micros and the Analog World** • Bike Inclinometer • **BL600 e-BoB** • **I²C and temperature sensor** • **Camelback Water Level Indicator** • Dimensioning a 3D Model • **Err-lectronics** • FFT Analysis in VB • **Hexadoku** • **I/O App (2)** • **Learn-'n-Go Infrared Remote Controlled Dimmer** • Over-Air LiPo Battery State Monitor • **PIC Assembler Crash Course** • **Retronics: Preco 39 Years On** • RPI Measures Electricity Consumption • **Selected Gerber Files** • **Selenium Rectifiers** • **Single Wire LCD Interface** • SMD Codes Revealed • TinkerForge Bricks & Bricklets • **USB Pseudo Battery** • VFD Shield for Arduino • **Wireless I²C Sensor … and more**

PC OSCILLOSCOPES

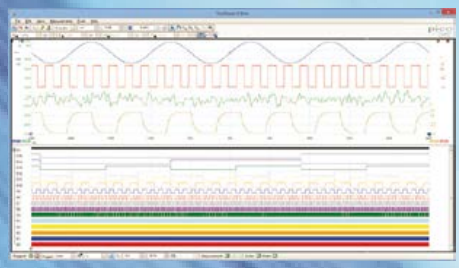
pico Technology

Low cost



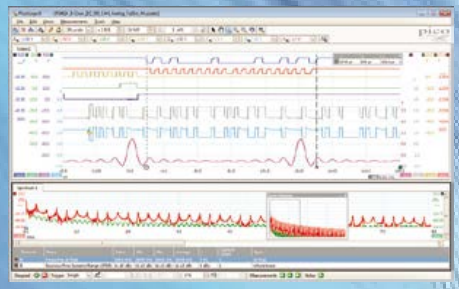
- 10 MHz to 200 MHz bandwidth
- 100 MS to 1 GS/s sampling
- 8 bit resolution (12 bit enhanced)
- 8 to 48 kS buffer memory
- USB powered
- Prices from $129 €99 £79

MSO



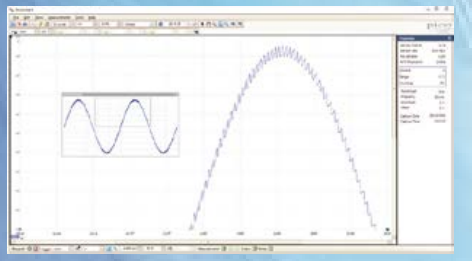
- 2 or 4 analog channels + 16 digital
- 50 to 200 MHz bandwidth
- 8 bit resolution (12 bit enhanced)
- 64 to 512 MS buffer memory
- USB or AC adaptor powered
- Prices from $819 €659 £499

Eight channels



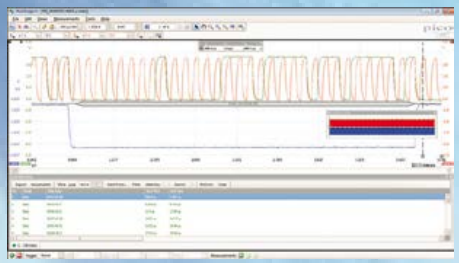
- 20 MHz bandwidth
- 80 MS/s sampling
- 12 bit resolution (16 bit enhanced)
- 256 MS buffer memory
- USB powered
- Just $2305 €1845 £1395

Flexible resolution



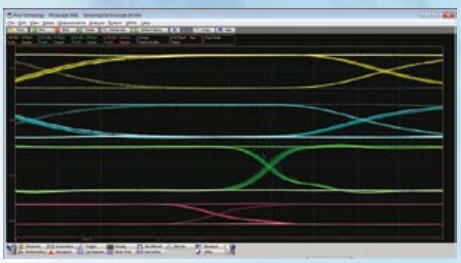
- 8, 12, 14, 15 & 16 bits all in one device
- 60 to 200 MHz bandwidth
- 250 MS/s to 1 GS/s sampling
- 8 to 512 MS buffer memory
- USB or AC adaptor powered
- Prices from $1155 €929 £699

2 GS memory



- 250 MHz to 1 GHz bandwidth
- 5 GS/s sampling
- 8 bit resolution (12 bit enhanced)
- 256 MS to 2 GS buffer memory
- AC adaptor powered
- Prices from $3295 €2645 £1995

20 GHz sampling



- DC to 20 GHz bandwidth
- 17.5 ps rise time
- 16 bit, 60 dB dynamic range
- AC adaptor powered
- Sig. gen, CDR, diff. TDR/TDT
- Prices from $14,995 €12,035 £9,085

Full software included as standard with serial bus decoding and analysis (CAN, LIN, RS232, I2C, I2S, SPI, FlexRay), segmented memory, mask testing, spectrum analysis, and software development kit (SDK) all as standard, with free software updates. Five years warranty real time oscilloscopes, 2 years warranty sampling oscilloscopes.

www.picotech.com/PS411

Edition 5/2015
Volume 41, No. 464 & 465
September & October 2015

ISSN 1947-3753 (USA / Canada distribution)
ISSN 1757-0875 (UK / 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, UK
Phone: (+44) (0)20 7692 8344

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

USA / Canada Memberships:
Elektor USA
P.O. Box 462228
Escondido, CA 92046
Phone: 800-269-6301
E-mail: elektor@pcspublink.com
Internet: www.elektor.com/member

UK / ROW Memberships:
Please use London address
E-mail: service@elektor.com
Internet: www.elektor.com/member

Advertising & Sponsoring:
Johan Dijk
Phone: +31 6 15894245
E-mail: johan.dijk@eimworld.com

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

Printed in the USA Printed in the Netherlands

Beyond Repair, never Beyond Measurement

If we have to accept that the electronic engineer is no longer the highly respected tech wizard on the block or street, always ahead of mass produced consumer stuff on the shelves, and forever pushing the performance of 'his' components, let's content ourselves with doing measurements and spreading the word on perfecting the art of measuring in the world of electronics. Even if your daughter's 14-inch super VGA LCD monitor is not worth repairing as it would have been 20 years ago, there is value and satisfaction in hunting down the one electrolytic capacitor with the bad ESR, replacing it, and finally moving that monitor to your workspace where an old PC is running programs Windows 8 does not like. As a spinoff there should be enough impressive numbers and Dilbert-ish commbuzzwords that can be linked to your repair job, for your daughter to copy-paste a report with for her economics classes, like "bathtub curve", "costing estimate" and "parts resources", as well as a ton of acronyms like DOA, FOA, BR, DIY. And she can have a new HDMI 23-inch LED screen.
Your part of the job requires knowledge, basic test equipment and some confidence. If designing a circuit is fun and creative, so is repairing and restoring one labeled BR (beyond repair). In both cases, expert test gear is required and a deep knowledge of what you are measuring. Browsing this edition of Elektor I hope you have as little difficulty finding measurement-related articles as I had while compiling the lot and going through the pre-press phases: the *PIC Assembler Course* this month touches on analog circuitry, this intrinsically involves ADC'ing real-world quantities; *FFT Analysis* on page 30 should be obvious; just like *AC/DC Power Meter* on page 46; and of course the *RPi Electricity Consumption Meter* (page 59). Challenging articles, all three. At a less ambitious level at least electronically there's the *Camelback Water Level Indicator* (page 76) and the *Bike Inclinometer* (page 80).
Although all projects mentioned here appear designed for a purpose, the articles hopefully contain elements that widen, trigger or channel your own creativity. Our humble contributions this month include how-to's of power metering the proper way, water level metering the cheap way (with two 4060's), and LiPo voltage metering, hey, over-air! Measure it — understand it — stay in the lead!

Enjoy reading this edition
Jan Buiting, Editor-in-Chief

THIS EDITION

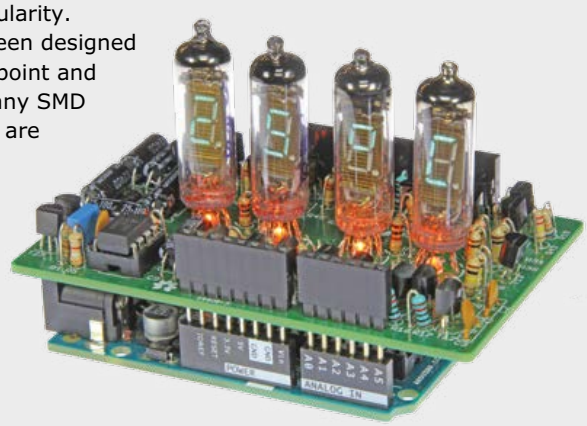
Volume 41 – Edition 5/2015
No. 464 & 465 September & October 2015

LEARN DESIGN SHARE

LEARN DESIGN SHARE

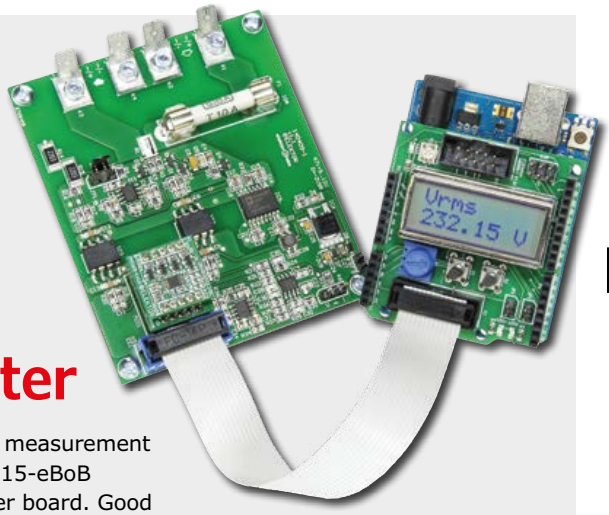
VFD Shield for Arduino

Nixie tubes have enjoyed renewed popularity for some time with retro & vintage enthusiasts and on the Internet there are countless designs for Nixie clocks to be found. In Elektor too, we have published several projects with Nixie tubes in recent years. However, in this project we do not use Nixies, but instead make use of VFD (Vacuum Fluorescent Display) tubes. These are somewhat different from the well-known Nixie tubes and are currently gaining popularity.
This Arduino shield has been designed from an educational viewpoint and therefore does not have any SMD components or parts that are difficult to obtain.



46

AC/DC Power Meter



This project consists of a measurement board holding the ADS1115-eBoB together with a small filter board. Good isolation is guaranteed between the power connections (AC or DC) on the primary side and the measurement signal outputs on the secondary side. The signals on the secondary side are digitized by the ADS1115, and the digital data is fed to an Arduino Uno with an Elektor Prototyping Shield. The displayed quantities for AC are effective power, voltage and current, as well reactive power and power factor.

Android I/O Board

Part 1

Control embedded electronics from your Android phone or tablet

64

124

elektor magazine

LEARN | DESIGN | SHARE

NEXT EDITION

MEASSY Audio Measurement System

This compact device combines a high-quality microphone preamplifier with phantom power supply with a small audio output stage in a single enclosure. You guessed it: MEASSY is geared to loudspeaker system measurements.

Compact 60-watt Audio Amp

This audio output stage is an all-discrete design. The circuit contains only commonly available components, is easy to build, takes up little board space, yet delivers fine sound quality.

Red Pitaya Does FM Stereo Radio

This article underscores the sheer versatility of the Red Pitaya development system. By adding a small prestage, and doing efficient digital editing, FM broadcasts (stereo and RDS) can be decoded.

Edition 6 / 2015 covering November & December is published on October 15, 2015.

Delivery of printed copies to Gold members subject to transport.

Contents and article titles subject to change.

Elektor Circuits &

Elektor breaks the constraints of a magazine. It's a community of active e-engineers — from novices to professionals — eager to learn, make, design, and share surprising electronics.

57
Countries

246817
Enthusiastic Members

10
Experts &

Elektor Post

The e-inspiration weekly

Never monostable and with trigger signals all over the contents, Elektor's dot-Post weekly newsletter has the ability to ring in the weekend with gossip, techtalk, stray bits, previews and news flashes. And a project every other week.

www.elektor.com/newsletter

Elektor Community

Become a member, Green or Gold

Membership of the Elektor Community is the surest way to enjoy classic electronics and embedded technologies side by side, ranging from beginner to pro. With direct access to Elektor Labs, forums, discounts, weekly newsletters, biweekly online projects, article archives, search engines, and back articles Green and Gold Members have permanent priority seating. Go GREEN if you want the magazine front to back delivered online only, or GOLD for the sumptuous package including printed copies.

www.elektor.com/memberships

Elektor TV

We're tubed too

No film set, suits, or Action! but you can rely on a camera rolling whenever things start humming, booting, displaying or smoking at Labs, or indeed any place or event our presenters find video compatible. Check out elektor.tv.

www.youtube.com/user/ElektorIM

Elektor PCB Service

Boards at your service

Forget the chemicals, get your electronic project to work as expected by ordering a ready-manufactured circuit board. Fast turnaround, pure quality, worldwide shipping.

www.elektorpcbservice.com

Elektor Labs

Learn, Design & Share

The techno creative center of Elektor that's steeped in hard core electronics all the way from scribble to PCB, component and kit. Wide open and accessible through its own website, Labs is where projects large, small, analog, digital, new and old skool are sketched, built, discussed, debugged and fine-tuned for replication and use by you.

www.elektor-labs.com

Elektor Academy

Ride the learning curve

Webinars, seminars, courses, presentations, workshops, lectures, in-company trainings, DVDs, and demos are just a few of the methods Elektor is using to spread the word about electronics both at hobby and professional levels.

www.elektor-academy.com

Connections Guide

Elektor Magazine

Close to 1024 pages of surprising electronics a year

If you prefer to absorb electronics over being absorbed, stick to reading Elektor's flagship product DMA'ed to you by their international editorial team. Whether arriving online or on paper every magazine edition is packed with electronics all-sorts for you to enjoy and explore in your own time. Free! Sign up!

www.elektormagazine.com

Elektor Web Store

Fill your shopping cart

Elektor has confidence in the products and services generated by Labs, Magazine, and selected business partners. That's why a brightly illuminated online retail store is open 24/7/365, with ordering and payment facilities for clients all over the world. An Aladdin's Cave of electronic parts and gizmos.

www.elektor.com

Elektor Books & DVDs

E-Information Powerpacks

It's hard to find a field of electronics not covered in depth and with authority by the products in our book and DVD portfolio. From reference work to programming course, 8-bit to ARM, Antenna to Zener diode; it's all there.

www.elektor.com

Become a member today!

GOLD	GREEN	FREE
€1.75 per week £1.27 / US $1.97	€1.31 per week £0.97 / US $1.49	
✔ Elektor Annual DVD	✖ Elektor Annual DVD	✖ Elektor Annual DVD
✔ 6x Elektor Magazine (Print)	✖ 6x Elektor Magazine (Print)	✖ 6x Elektor Magazine (Print)
✔ 6x Elektor Magazine (Digital)	✔ 6x Elektor Magazine (Digital)	✖ 6x Elektor Magazine (Digital)
✔ Access to Elektor Archive	✔ Access to Elektor Archive	✖ Access to Elektor Archive
✔ Access to Elektor.LABS	✔ Access to Elektor.LABS	✖ Access to Elektor.LABS
✔ 10% Discount in Elektor Store	✔ 10% Discount in Elektor Store	✖ 10% Discount in Elektor Store
✔ Elektor.POST Newsletter	✔ Elektor.POST Newsletter	✔ Elektor.POST Newsletter
✔ 25 Extra Elektor Projects	✔ 25 Extra Elektor Projects	✔ 25 Extra Elektor Projects
✔ Exclusive Offers	✔ Exclusive Offers	✔ Exclusive Offers
www.elektor.com/gold	www.elektor.com/green	www.elektor.com/newsletter

KCS TraceME

LoRa ™ Internet of Things

KCS has extended their successful TraceME product line with an advanced module, targeted for worldwide mobility in the Internet of Things era. The latest development of the TraceME GPS/GPRS Track and Trace module will combine the RF location based positioning solution with the LoRa™ technology. This combination offers 'smart objects' being even smarter, since LoRa™ enables long range, battery friendly communication in a wide variety of (M2M) applications. Supporting GPRS/SMS and optional 3G, Wi-Fi, Bluetooth LE, ANT/ANT+ and iBeacon™ provides easy integration with existing wireless networks and mobile apps. Other variants in the high/mid-range and budget-line will follow soon.

ANTI-THEFT module based on RF

KCS TraceME product line offers an intelligent location based positioning solution for indoor and anti-theft applications.The solution is based on RF with an intelligent algorithm of measuring the propagation time of transmitted (proprietary protocol) signals. Unique features are: minimum size (46x21x6.5mm), weight (7 grams for fully equipped PCB) and a standby battery lifespan of more than 10 years.'Listen before talk' algorithm makes it practically impossible to locate the module, which secures the valuable vehicle or asset. Supporting GPRS/SMS and optional 3G, Wi-Fi, Bluetooth LE, ANT/ANT+ and iBeacon provide easy integration with existing wireless networks and mobile apps.

www.Trace.ME

Welcome to the LEARN section

By **Jens Nickel**

@ the Maker Faire

By the time you read this the last discarded coffee cup, wire clipping and solidified solder blob will already be swept up and consigned to the trash after this year's Maker Faire held in Hannover, Germany. I would like to thank all our colleagues at *Make:* for ensuring this event was such a success. I don't know how, but they even managed to arrange good weather! The hands-on activities were particularly inspiring for all the (really) young engineers in attendance. The eight soldering stations at our workshops (photo) were only allowed to cool down once the last 'Electronic Dice' kit had been fully assembled.

It was also a good opportunity to check out many of the latest developments, boards and projects that were on show. One product that caught my eye was the system of hardware modules from Brick ‚R' Knowledge (www.brickrknowledge.de); these well designed modules look particularly useful for (but not just for) the teaching of electronics. Another interesting project is the 'SenseBox'. This IoT sensor kit is also targeted at education and includes an Internet platform to collect data from numerous environmental sensors measuring air pressure, temperature, humidity and incident light levels. Two versions of the system are available; one for education and another for DIY use.

Learning to Program (2)

I have already received a number of emails in connection with the '(High)School Board' suggestion and the 'Learning to program' editorial, many thanks for those, your feedback is always welcomed. Amongst other things a reader pointed me toward 'Project Oberon' by Niklaus Wirth. I must admit I had not been aware of this FPGA-based processor hardware (see for example www.xilinx.com/publications/archives/xcell/Xcell91.pdf, from page 30). Another reader suggested the low-cost Launchpad range of development kits from TI. Once again I was grateful for the comments from another reader who, like me, wanted a board programmable using C# supporting both PC and embedded environments that could be programmed by using just one programming language. Perhaps we should begin with a Raspberry Pi 2 running Windows — quite intriguing.

And then there was...

Another episode from the ongoing 'stupid-mistakes-made-while-designing-and-developing' saga. I would of course prefer to keep these to myself but that way nobody would profit from my errors. Finally I got to porting the MIDI Analyzer (featured in the last edition) firmware from the Arduino Uno to the Elektor Xmega Webserver board. My EFL configurator had been set up so that with just a single key press an Xmega project is generated with all the necessary files and the hardware-independent source code for the MIDI analyzer. Now all I need to do is change the number of the UART interface in the code which talks to the MIDI module. The compiler worked its magic without any errors but something here didn't look right; the power LED on the MIDI module did not light up when it is plugged into the Xmega board. I chose a second board, this time it worked. A quick look at the schematic revealed the reason why: The first board was missing a jumper! Sometime in the past I had, for some reason, removed it. The lesson here is that it always pays to double check the documentation before you bring a board back into use again. This is especially true when you are totally confident you already know the board inside out! ⏮

(150319)

The analog world

From 8 to 32 bits: ARM Microcontrollers for Beginners (5)

By **Viacheslav Gromov** (Germany)

So far in this course we have looked at the SAM D20's wide range of digital peripheral blocks; now it is the turn of the analog circuitry. It will come as no surprise to readers with experience from the world of 8-bit microcontrollers that we will be looking in detail at both the analog-to-digital converter (ADC) and analog comparator (AC); and to those we also add the digital-to-analog converter (DAC).

Our SAM D20 includes a powerful ADC with a resolution selectable between eight, ten and twelve bits. Its structure is illustrated in **Figure 1**. The same rule of thumb applies as in the case of eight-bit microcontrollers: the lower the resolution, the shorter the conversion time. The SAM D20's ADC can manage up to 350,000 conversions per second, while in oversampling mode it can achieve a resolution of 16 bits. As many as 32 inputs to the converter are available, of which up to ten can be used as inverting inputs and 25 as non-inverting inputs (some being able to function as either). Some of the inputs are designed to be used with internal sources such as the DAC or temperature sensor. As can be seen from the block diagram, the ADC always requires a non-inverting and an inverting input and will convert the voltage difference between these two inputs into a digital value. The inverting input can of course simply be connected to ground. The ADC also requires a reference voltage, of which several are available. If, for example, the voltage reference is 1 V, the maximum voltage the ADC can measure will also be 1 V; if the resolution is set to 12 bits, then the output will be expressed in steps of approximately 0.0002 V. Other functions available include amplifying the input signal with a gain of between 0.5 and 16, and a window comparison mode to monitor whether the measured values fall within a specified range. An interrupt can be triggered if the limits are exceeded. The ADC can be configured to carry out a single conversion or to carry out continuous conversions when started. Further information on the ADC can be found in the datasheet starting on page 481.

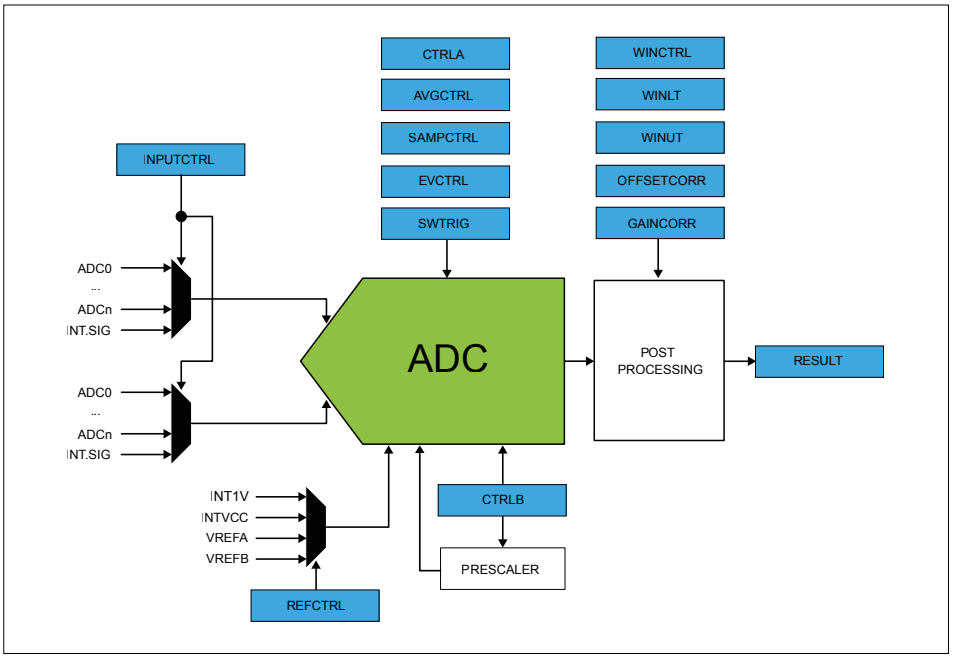


Figure 1. Outline structure of the analog-to-digital converter (ADC).

First experiments with the ADC

The ADC is an important peripheral block, and so we of course want to see what it can do in practice. **Figure 2** shows a simple circuit that can be built using an LM335 analogue temperature sensor. The two components can easily be soldered together in mid-air (see **Figure 3**). To obtain accurate readings the cable should be kept reasonably short.

The temperature sensor is easy to use and exhibits good linearity [1]. We would like to measure the temperature using the ADC and output the readings once per second to the PC over the virtual serial port (UART). The project 'ADC Test1' shows how this is accomplished: like all the code accompanying this article it is available for download at [2]. Supplementary documents including code listings are also available from the same location.

At the beginning of the main file are the commands to create the instance structures for the U(S)ART and for the ADC, the function prototypes, the declaration

of various variables and the by-now-familiar configuration function to set up the U(S)ART for EDBG communication at 9600 baud. Next comes the configuration function for the ADC, as shown in **Listing 1** [2]. As is usual when using the ASF, we begin by declaring a configuration structure called `config_adc` whose members will be filled with the required settings using symbolic constants. The variables `positive_input` and `negative_input` select the source for the non-inverting input (AIN0 on PB00) and for the inverting input (GND). The structure member `reference` is used to set the reference source to AREFB on pin PA04. We have connected AREFB to VCC, which means that the reference voltage will be 3.3 V. The overall 3.3 V range is converted at a resolution of 12 bits, as specified by the structure member `resolution`. The input gain is set to unity using the member `gain_factor`, which means that the input voltage will be converted at 1:1 without amplification. We disable division of the ADC result and disable window mode using the members `divide_result` and `window_mode` respectively. At the end of the function the settings we have written to the configuration structure are passed to the ADC and it is enabled. From this point on the ADC can be referred to via its instance structure `adc_instance`.

In the main function we first initialize the microcontroller and then call the two configuration functions described above. Then there is an infinite loop containing the core of the program (see **Listing 2** [2]). It begins with a call to carry out a single conversion, passing the instance structure as the argument: `adc_start_conversion(&adc_instance);`. The we wait in a while-loop repeatedly calling the function `adc_read(&adc_instance, &data);` which attempts to read the ADC conversion result. The loop terminates when the conversion is complete and the function no longer returns STATUS_BUSY; the result will be stored in the variable 'data', a pointer to which was passed to the read function. The calculation `volt = data * 0.000805;` converts the reading into a value in volts, the quantity 0.000805 being equal to 3.3 V/4096, the smallest step the ADC can resolve in 12-bit mode. The command `temperature = 25 + (volt - 2.945) / 0.01;` converts the voltage measured across the LM335 into a temperature value. The first step is to subtract 2.495 V, which is the voltage we expect to see at our reference temperature of 25 °C. The result of this is divided by 0.01, which corresponds to the 10 mV/°C slope characteristic of the device. Finally we center the result about the 25 °C reference. It is possible to adjust this calculation to calibrate the sensor.

The next two commands convert the floating-point temperature value into ASCII format in an array ready for sending to the UART. First the value is multiplied by 100 so that the first two digits after the decimal point are brought before it. The `sprintf()` function (similar to the print command in other systems) formats this value as a string in `temperature_buffer`. The following for-loop, in which the variable i is incremented by 1 on each iteration, sends the four characters to the UART using the command `usart_write_wait(&usart_instance, temperature_buffer[i]);`.The if-statements before that call determine where the decimal point should be inserted. If the value is greater than or equal to 1000, the decimal point (ASCII 46) is output after the second digit; if the value is less than 1000, in which case there is only one

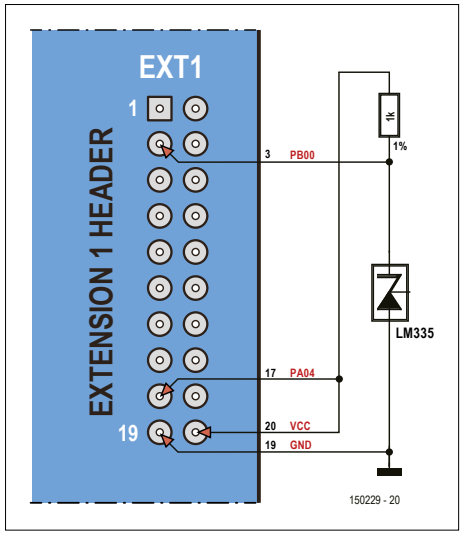


Figure 2. The analog temperature sensor with a resistor.

Figure 3. The two components can be hooked up in mid-air.

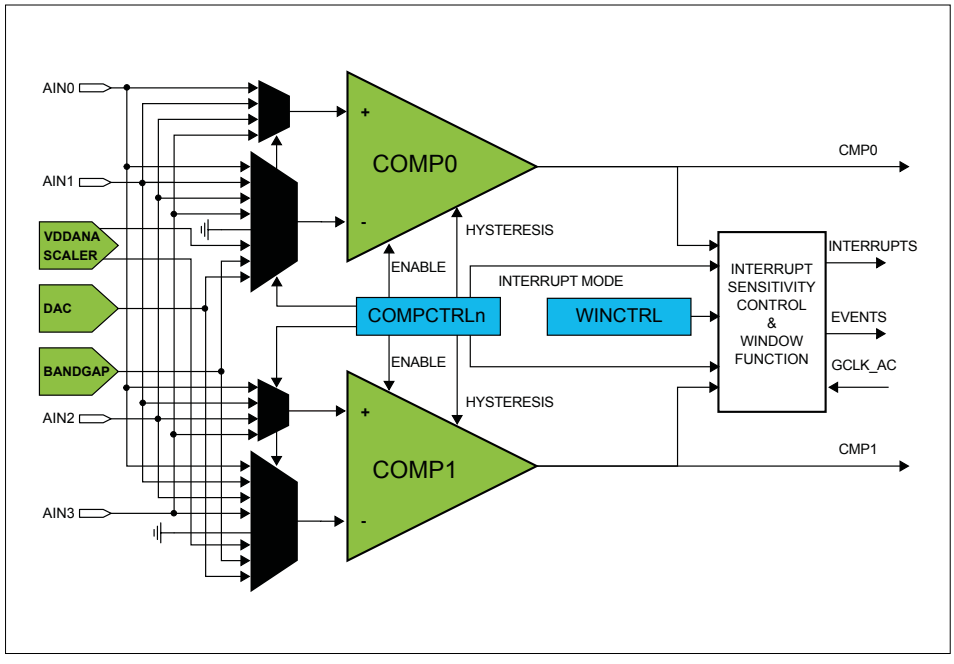


Figure 4. Structure of the analog comparators.

digit before the decimal point, it is output after the first digit. The final if-statement inside the for-loop sends a line feed (ASCII 10) character after the final digit, so that the next reading will be output on a new line. The infinite loop finishes with a one second delay, implemented by a call to the function `delay_s(1);`. Readings are thus output at approximately one per second.

To keep things simple and to make the demonstration clearer, for this project we use the ASF Wizard to include the polled variants of both the U(S)ART and ADC libraries. In the polled variants it is necessary to wait until the command to send a character to the U(S)ART returns the value STATUS_OK to indicate the successful transmission of the character before proceeding, and this is why the while-loops are needed in the code.

If you transfer the program to the board and establish a connection with it using the terminal emulator program (not forgetting to activate the DTR signal!) you should see the readings on the screen. For various reasons the readings may deviate by a couple of degrees from the true temperature value and may fluctuate a little [3].

The analog comparator

The SAM D20 includes two analog comparators (ACs) which can compare two voltages and indicate whether the difference is positive (the first input is higher than the second) or negative (the second input is higher than the first). The structure of the ACs is shown in **Figure 4**: it is clear that there are many possible sources for the inputs to the comparators and that, like the ADC, it is possible for them to operate together in window mode. The analog comparators can trigger interrupts or events, and a hysteresis can be set to reduce sensitivity to noise. A digital filter is also available, again to control noise sensitivity. Like the ADC, the AC can be configured to operate continuously or, to conserve power, to carry out a single comparison. More on the AC can be found starting on page 521 of the datasheet.

The AC in practice

We will now look at a quick project to demonstrate the analog comparator. We will arrange for LED0 on the SAM D20 Xplained board to light when the voltage on potentiometer P1 is higher than that on potentiometer P2, and to go out otherwise. The two potentiometers are connected as shown in **Figure 5**, most conveniently, as shown in **Figure 6**, using a breadboard. The potentiometers are wired as voltage dividers so that a variable voltage from 0 V to 3.3 V is presented to each of the two comparator inputs.

The code for the project 'First step with AC' is relatively easy to understand. Note that the callback variant of the AC ASF library is selected in the ASF Wizard.

At the beginning of the main program file we include the header file 'asf.h' as usual and then give the function prototypes. We then create an instance structure `ac_instance` of type `ac_module`. Next we come to a small function with the three lines:

```
struct ac_config config_ac;
ac_get_config_defaults(&config_ac);
ac_init(&ac_instance, AC, &config_ac);
```

The seasoned eye will recognize this as nothing more than the declaration of a configuration structure, its population with default values and then its transfer to the AC peripheral block. We will subsequently be able to refer to the comparator using the instance structure `ac_instance`. The default settings basically cover the selection of the clock source: in this case the source is GCLK generator 0, which also drives the CPU and hence does not need to be enabled separately.

The longest and most important configuration function is the one that sets up channel 0 of the comparator and the required input pins (see **Listing 3** [2]) and the code is rather more complex. We do not need the second channel for our project, as we only want to compare two voltages. As can be seen from the listing, we start by declaring the configuration structure `config_ac_chan` and initialize it with its default values using `ac_chan_get_config_defaults(&config_ac_chan);`. Using the dot operator we then populate several of the members of the structure with the values we require. First we set the member `sample_mode` to `single shot`, so that the comparator only performs a single comparison when started. We then make internal connections between the non-inverting input of the comparator and AIN0 (PA04), and between the inverting input and AIN1 (PA05). These are different from the AIN0 and AIN1 pins used by the ADC: be careful not to mix them up!

Next the so-called 'majority of five' digital filter is enabled to increase immunity to noise on the input signals. The member `interrupt_selection` is set to specify when an interrupt is to be triggered:. In this case we set it to the symbolic constant `AC_CHAN_INTERRUPT_SELECTION_END_OF_COMPARE` which causes the callback function to be called after each AC comparison operation.

The function then proceeds to specify the configuration of the input pins PA04 and PA05 more precisely. In each case we first declare the configuration structure `acx_pin_conf` (where `x` is 0 or 1) for the pin and then load the values `SYSTEM_PINMUX_PIN_DIR_INPUT` and `MUX_PA0xB_AC_AINx` into the structure members `direction` and `mux_position` respectively. As its name indicates, the first variable determines the direction of the pin (whether it is an input or an output): in this case we use the symbolic constant shown to configure it as an input. The second variable determines the function of the pin. The first section of code refers to the required symbolic constants and structure members for pin PA04 (AIN0) and the second section to those for pin PA05 (AIN1). At the end of each section the settings stored in the configuration structure are transferred to the corresponding pin, using the command `system_pinmux_pin_set_config(PIN_PA0xB_AC_AINx, &acx_pin_conf);` which simply requires the appropriate symbolic constant and a pointer to the configuration structure. Following these two sections we have a call to the function to transfer the settings in the configuration structure `config_ac_chan` that we had set up at the beginning of the function to channel 0, followed by a call to enable the channel. In both calls we use the instance structure `ac_instance` as a parameter.

Next in the file is the callback function, called `callback_function_ac()`. This interrupt service routine (ISR) is called every time an analog comparison completes and triggers its interrupt. The ISR contains the most important part of the program, taking the form of a switch-case statement, which, depending on the result of the comparison operation, turns LED0 on the Xplained Pro board on or off. The relevant part of the code is:

```
case 10: port_pin_set_output_
level(LED_0_PIN, 1);
    break;
case 12: port_pin_set_output_
level(LED_0_PIN, 0);
    break;
```

This code uses the output of the command `ac_chan_get_status(&ac_instance, AC_CHAN_CHANNEL_0);` to determine the result of the last analog comparison. The command returns a bit mask of the channel flags: 10, if the voltage on P1 (AIN0) is greater than that on P2, and 12 if it is lower. In the second case LED0 is turned on: note that the output is active low. The advantage of using the on-board LED is that we do not have to configure its pin manually, as it is automatically configured for us.

Following the switch-case block the ISR proceeds to trigger the next comparison operation with the command `ac_chan_trigger_single_shot(&ac_instance, AC_CHAN_CHANNEL_0);`. Again, we simply require a pointer to the instance structure for the AC and the symbolic constant to specify the channel to be used.

The last function declared before we reach the main loop is `configure_ac_callback()`, which registers the interrupt and enables it. The two commands to do this follow the conventional ASF pattern: the first, to register the callback, requires a pointer to the instance structure for the AC module, the name of the callback ISR and the type of interrupt (in this case `AC_CALLBACK_COMPARATOR_0`); the second is similar, but it does not require the name of the ISR.

Now we come at last to the main function, which begins by initializing the microcontroller and then calling the configuration functions we have described above. There then follows the command `ac_enable(&ac_instance);`, which enables the whole AC peripheral block, and then the command to initiate the first comparison operation. The infinite loop is empty, as from this point on all the work is done in the calls to the ISR described above: after each call to the ISR, it starts the next comparison which will in turn trigger the ISR again. Thus, after the interrupt is triggered for the first time, the code effectively runs in an infinite loop of interrupts.

The 'Start without Debugging' command of the development environment can be used to load the project onto a board to which the potentiometers have been connected [4].

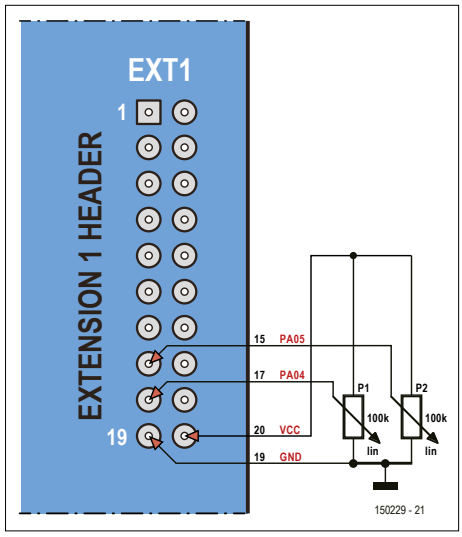


Figure 5. This circuit with two potentiometers allows the analog comparators to be tested.

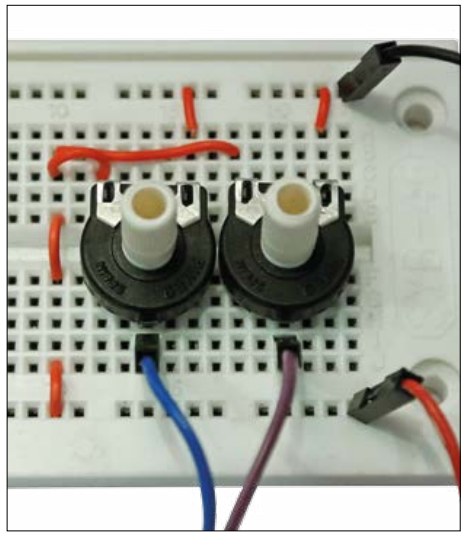


Figure 6. The two potentiometers on a breadboard.

The DAC

A digital-to-analog converter (DAC) is a comparatively rare feature of eight-bit microcontrollers. A DAC converts a digital value into an analog voltage, the opposite function to the ADC, and is often used in audio applications. Our SAM D20 includes one DAC, whose block diagram is shown in **Figure 7**. The SAM D20's DAC only offers one channel, but has a reasonably high resolution of 10 bits and a maximum sample rate of 350 kHz.

On the left of the block diagram you can see a number of registers, mostly to do with the configuration of the DAC, which we will not examine here in detail. We will however mention that the data register that holds the ten-bit value currently being converted is accompanied by a buffer register of the same size. If required, the two registers work on a first-in-first-out (FIFO) basis, so the most recently loaded data is not always converted immediately. If the DAC receives a command to convert a digital value to analog when the DATA register is empty it can be made to trigger an EMPTY event. Conversely, another peripheral within the microcontroller, such as a timer, can be used to trigger a START event that causes the DAC to initiate a conversion.

The block diagram shows the wide range of reference sources and output options. As in the case of the ADC, the DAC can be configured in software for operation with an external voltage reference, with the power supply (AVCC) as the reference, or with the internal accurate 1 V reference

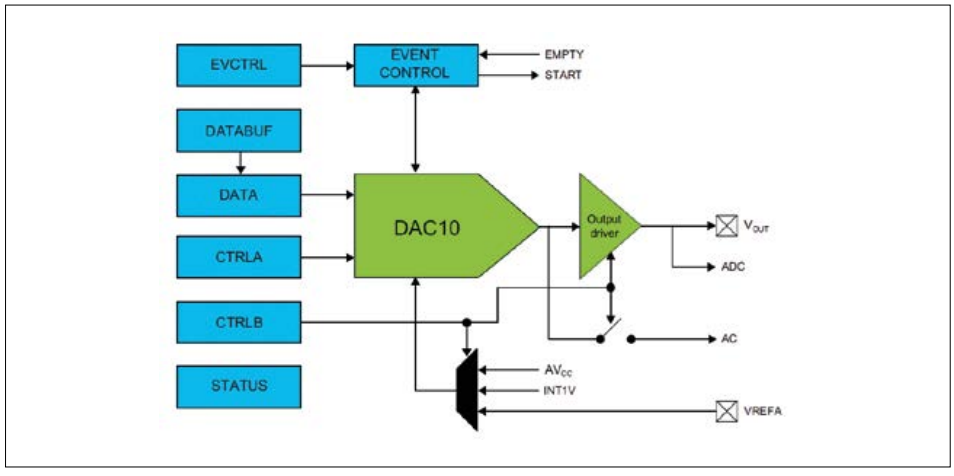


Figure 7. The arrangement of the DAC block.

voltage. The converted voltage can be output on the external VOUT pin (PA02), routed to an ADC input (with an optional internal amplifier in between), or routed to an input of an analog comparator. As with most of the peripheral blocks on the SAM D20 the DAC can receive its clock from a range of sources, and its clock must be synchronized with that of the internal data bus when a read or write operation occurs. More on this can be found in the datasheet starting on page 550.

To get to know this peripheral a little better, we have developed a small example program using the polled variant of the ASF DAC library that generates a sinewave signal with an amplitude of 3.3 V and a frequency of approximately 500 Hz. A piezo sounder can be connected as shown in **Figure 8**, or the waveform can be inspected using an oscilloscope. Because of the low amplitude the sound from the piezo crystal is rather quiet: to improve matters it can be attached to a sound box.

Now to the code in the main file. After including the header file 'asf.h' and prototyping the functions we declare a variable 'i', which will subsequently be used in a for-loop, and we declare and initialize an array 'sinus' (English: sine) that contains 360 sine values as provided by the handy online sinewave generator at [5]. The lowest value is zero and the highest is 1023, and so the full 10-bit resolution of the DAC will be used.

The two configuration functions for the DAC (with instance structure `dac_instance` of type `dac_module`) in **Listing 4** [2] set up the DAC block according to the usual pattern. First a configuration structure called `config_dac` is declared. Then it is populated with default values, and then the dot operator is used to set the members of the structure to reflect the required settings. In this case the DAC receives its clock from GCLK generator 0, the output of the DAC is taken to output pin PA02, and the 3.3 V supply voltage is used as the reference using the command `config_dac.reference = DAC_REFERENCE_AVCC;`. Finally the settings stored in the configuration structure are transferred to the DAC with the command `dac_init(&dac_instance, DAC, &config_dac);`.

The second configuration function is even easier to understand. Here DAC channel 0 is configured separately using the declared configuration structure `config_dac_chan`, which is populated with the default settings. The command

```
dac_chan_set_config(&dac_instance,
DAC_CHANNEL_0, &config_dac_chan);
```

is then used to transfer the settings to the channel.

The main function initializes the microcontroller and the SysTick timer (needed for the delay function). It then calls the two configuration functions and finally enables the DAC module with the command `dac_enable(&dac_instance);`.

The infinite loop is the core of the whole program. It consists of a single for-loop in which the function `dac_chan_write(&dac_instance, DAC_CHANNEL_0, sinus[i]);` is called to transfer the samples stored in the 'sinus' array without delays to the DAC for conversion to a voltage. After compiling and transferring the program '500Hz with DAC' a clean sinewave signal at 500 Hz appears at the output. But, given that there are no delays in the code, why does it not run faster? It is simply the case that the DAC requires time to execute the write command. If the clock frequency is increased, or the sinewave table made shorter, a much higher output frequency can be achieved. Conversely, if a delay is introduced into the for-loop, the output frequency will be reduced. With a little more work it would be possible to play music using the DAC, or it could be used, for example, as part of an analog circuit tester [6].

This installment of the course has looked at the important analog interfaces of the SAM D20. With the foundations now firmly in place, we will look in the next installment at an ambitious and exciting project using the SPI bus. ⏮

(150229)

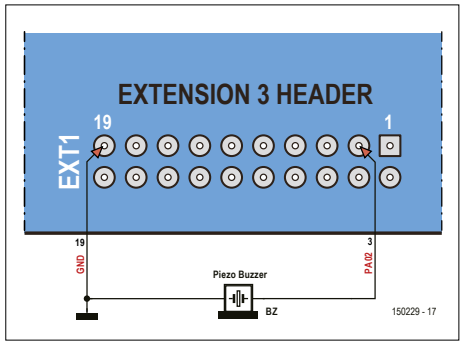


Figure 8. The circuit consists of just the board and a piezo sounder.

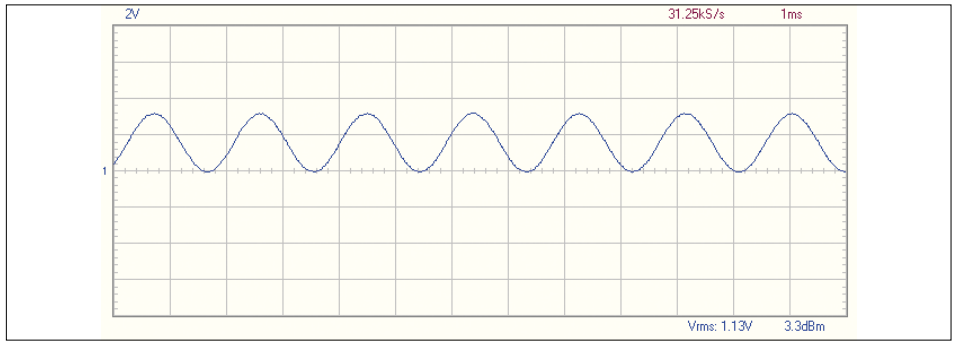


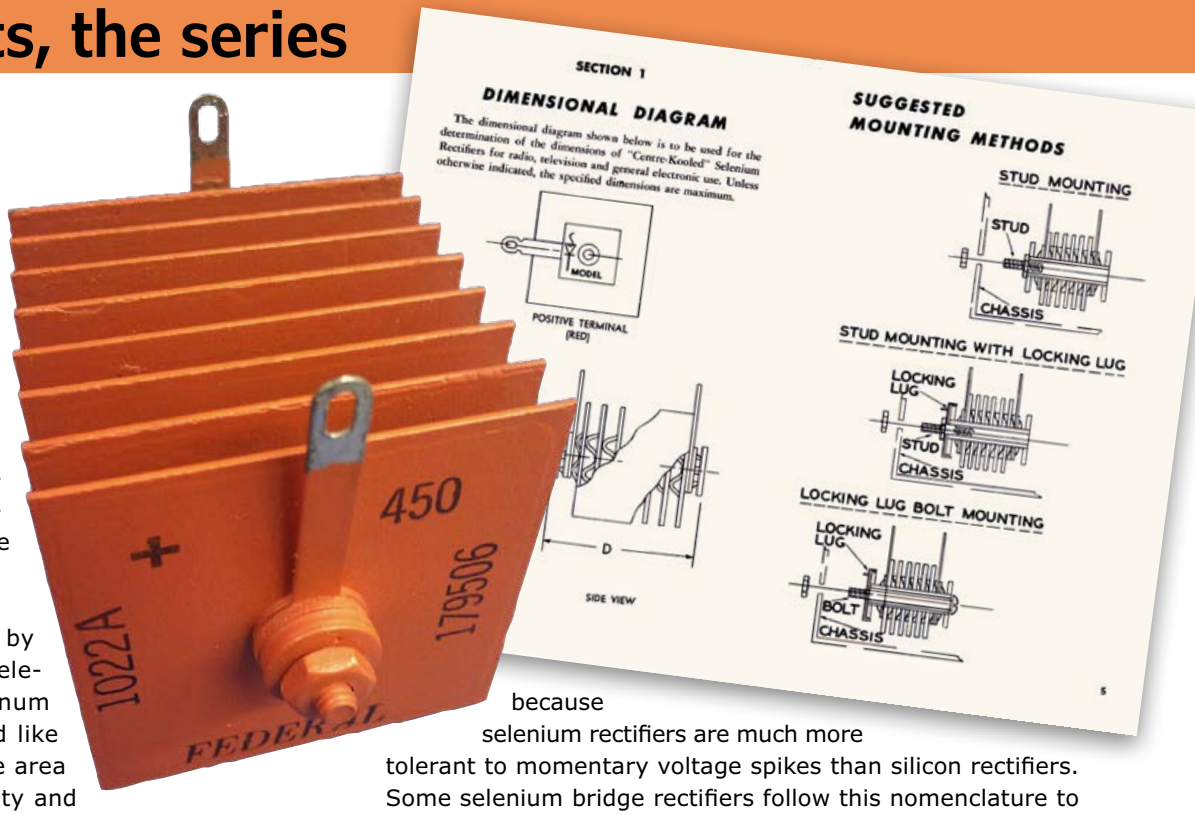
Figure 9. The clean output signal of the DAC shown on an oscilloscope.

Web Links

[1] www.ti.com/lit/ds/symlink/lm235.pdf
[2] www.elektormagazine.com/150229
[3] www.atmel.com/images/atmel-42109-sam-d20-analog-to-digital-converter-driver-adc_application-note_at03243.pdf
[4] www.atmel.com/Images/Atmel-42106-SAM-D20-Analog-Comparator-Driver_Application-Note_AT03242.pdf
[5] www.daycounter.com/Calculators/Sine-Generator-Calculator.phtml
[6] www.atmel.com/Images/Atmel-42110-SAM-D20-Digital-to-Analog-Driver-DAC_Application-Note_AT03244.pdf

Selenium Rectifiers

Peculiar Parts, the series



By **Neil Gruending** (Canada)

Everyone knows that rectifying an AC voltage will generate a DC voltage. Silicon based rectifiers do this very well but in they weren't available for substantial currents in the 50's and 60's so selenium rectifiers were used instead. They are quite a bit bigger than their silicon counterparts but still relatively efficient for a metal based device — at that time.

A selenium rectifier is made by sandwiching a thin layer of selenium between steel or aluminum plates which are then stacked like in **Figure 1**. The plate surface area determines the current capacity and the number of stacked plates determines the rectifier voltage rating. Each plate adds about 20 V to the reverse voltage so for example the rectifier in Figure 1 has a PIV (peak inverse voltage) rating of 160 V because it has eight plates.

Selenium rectifiers were used in power supplies quite a bit because they have a significant efficiency advantage over vacuum tubes from that era. For example, a 120-V selenium rectifier will have a typical forward voltage drop of 5 V whereas a vacuum tube rectifier could have a drop of 10 to 25 V. The tube would also need some heating current which also reduces its efficiency.

If you have any 1950's and 1960's era radios, televisions, test equipment or boatanchors then they will probably contain selenium rectifiers in their power supplies. It's always a good idea to replace them with silicon diodes even if everything is working properly. That's because as selenium rectifiers age their forward voltage and leakage currents will increase which can affect a circuit's performance. In a worst case scenario this can cause them to overheat and catch fire, releasing a toxic foul smelling smoke. Many people say that the smoke has a garlic or onion smell to it.

Fortunately selenium rectifiers whether single or bridge can't handle a lot of forward current and are typically rated for a forward current of about 100-250 mA which makes them easy to replace with a silicon diode. The 1N400x, 1N540x, and BYxxx series diodes are commonly used since they come in a wide range of voltages and a 1-amp rating is usually more than enough. Just make sure that you allow for a pretty wide safety margin because selenium rectifiers are much more tolerant to momentary voltage spikes than silicon rectifiers. Some selenium bridge rectifiers follow this nomenclature to identify the main electrical ratings: B*xxx*C*yyy*, where *xxx* = PIV and *yyy* = milli-amps. Easy. The numbering system was continued with their silicon counterparts.

Once you've chosen a Si diode the next step is to add a series resistor to emulate a selenium rectifier's natural current limiting ability and larger voltage drop. The easiest way to determine the resistance value is to divide your desired voltage drop (5 to 10 volts) by the expected forward current. Also make sure that you use a resistor with a high enough wattage rating; 5-watt, 100-ohm to 200-ohm resistors are typically used.

I normally don't like to replace older parts like that but unfortunately it's necessary for selenium rectifiers because it's not a matter of *if* they will fail but *when*. And when they do, it can take days for the smoke and foul smell to clear and the missus will be unhappy. However, if you want to stick to your guns there is a Selenium Rectifier Handbook [2] available if you want to learn more about them.

(150283)

[1] http://en.wikipedia.org/wiki/Selenium_rectifier#/media/File:Selenium_Rectifier.jpg

[2] www.tubebooks.org/Books/srhbst.pdf

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

DesignSpark Mechanical
CAD Tips & Tricks (2)
Dimensioning a 3D Model

By **Neil Gruending** (Canada)

In the previous installment I showed you how to add a connector to a PCB model and then modify the enclosure for it. Now let's add some dimensions to the enclosure for the connector cutout.

Creating Annotation Planes

Before we can dimension our model we need somewhere to place them. DesignSpark Mechanical does this with annotation planes which are a flat surface in a design to store documentation information like dimensions. Annotation planes are just like regular planes because they can be aligned to an edge, line, axis or a combination of all three. The easy way to set up an annotation plane is to click on the Dimension tool in the Investigate menu tab and select the enclosure face with the connector cutout like in **Figure 1**.

The Dimension tool will highlight the line or face under the cursor and the associated plane as a wireframe rectangle to indicate where the plane would go. In this example you can see where the plane would go with the selected face but since the enclosure has tapered sides the annotation plane is slightly angled relative to the Z axis. Sometimes this is ok but if you wanted an annotation plane without the angle then we have to select an axis first.

You use the Axis tool is in the Insert menu tab to create an axis for the annotation plane. For this example you would then click on the upper inside edge of the connector cutout face to add a dashed and dotted line that we can then use for our annotation plane like in **Figure 2**. Now we have an annotation plane that's parallel with the Z axis that we can use to add dimensions to model.

Adding dimensions

Now let's add some dimensions to the annotation plane we just created and finish up with the Dimension tool. I find that it's easiest to view the annotation plane head on, so I rotated the model by clicking on the Plan View button in the Orient menu to get the view in **Figure 3**. Then I used the Dimension tool to add the linear dimensions shown.

Adding dimensions is really easy with the Dimension tool because most of the time you just have to click on the two elements that you want to measure. The elements can be any point, line or face. If the elements are parallel with each other the Dimension tool will create a linear dimension that measures the distance between them. Otherwise the Dimension tool will measure the angle between them. Once the dimension is created then you can then place it with your mouse.

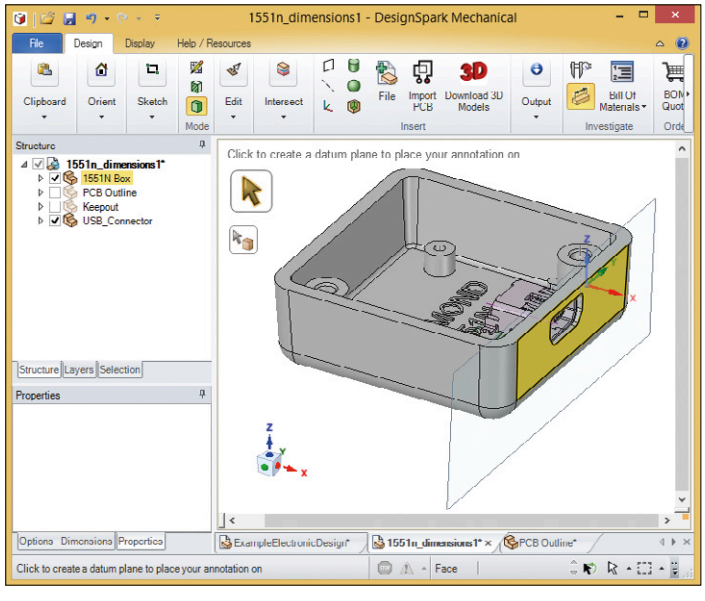


Figure 1. Connector cutout face.

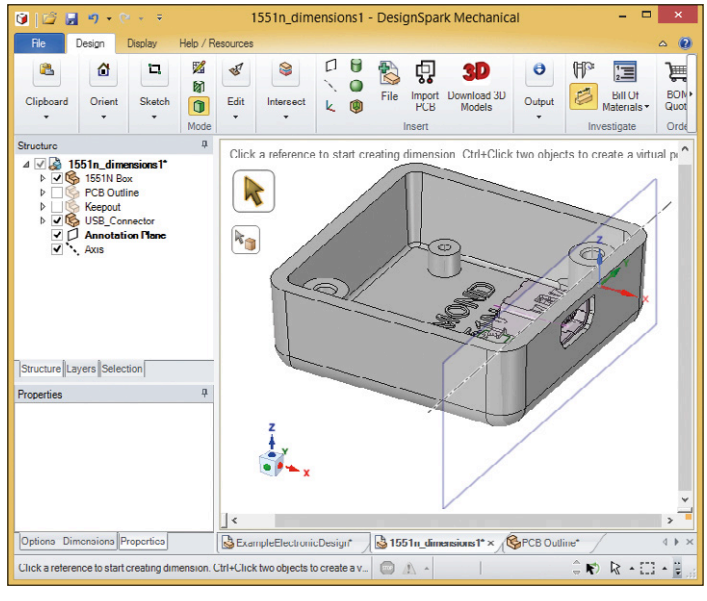


Figure 2. Connector face annotation plane.

The Dimension tool is still active at this point so it will modify the measurement lines and arrows as necessary while you move the dimension around. Another nice feature of the tool is that when only one element is selected it will display what the measurement would be as you mouse over other elements in the design.

You can also easily add radius dimensions which only require one measurement point. The trick is that you have to click on the top, bottom, left or right portions of the arc. Otherwise the Dimension tool will assume that you are trying to measure between two points. It's also possible to measure an arc by using the CTRL key while clicking on it. If you hold down CTRL while clicking on an arc it will measure the length of the entire arc or you can hold the CTRL key while dragging the dimension. The lower left corner of Figure 3 also shows the option panel when the Dimension tool is active. Here is where you can set the dimension and reference orientations used by the Dimension tool when adding new dimensions. The default settings are all automatic but it makes sense to set the orientations you want if you were adding a large number of dimensions.

Editing dimensions

Dimensions in the model are also intelligent objects. This means that you can edit them at any time by clicking on them. For example, if you click on the dimension text it will be highlighted like in **Figure 4**. You adjust the size of the text by clicking on the small circles and moving them in the appropriate direction. If you hover the mouse pointer over the dotted box then you can move the text.

Right clicking on a dimension will open the floating toolbar shown in **Figure 5**. Here is where you can add a note to the dimension, change the tolerance type, add a data field or insert symbols into the dimension text respectively. Other things like the arrow type can be set properties window.

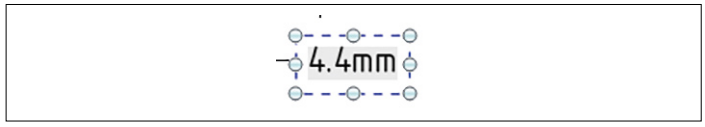


Figure 4. Modifying dimension text.

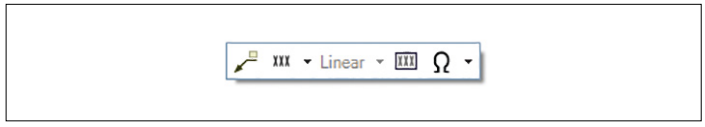


Figure 5. Dimension toolbar.

So far all of the dimensions have been in millimeters but if you wanted to change the measurement units you can go into the File menu, select DesignSpark Options and then click on Units. Here you can change the units used for length, angles, mass, etc. You can also specify measurement precision.

Using Annotation Planes

Once you're done adding dimensions to the annotation plane you will get something like **Figure 6**. As you can see, all of the dimensions are contained to the one plane which makes them very easy to manipulate as a group. For example, to disable displaying the dimensions you can clear the Annotation Plane checkbox in the model structure window.

Conclusion

The DesignSpark Mechanical Dimension tool is a really powerful way to add dimensions to a model. I've only touched on its flexibility here — I encourage you to try it out to see what it can do! ⏮

(150218)

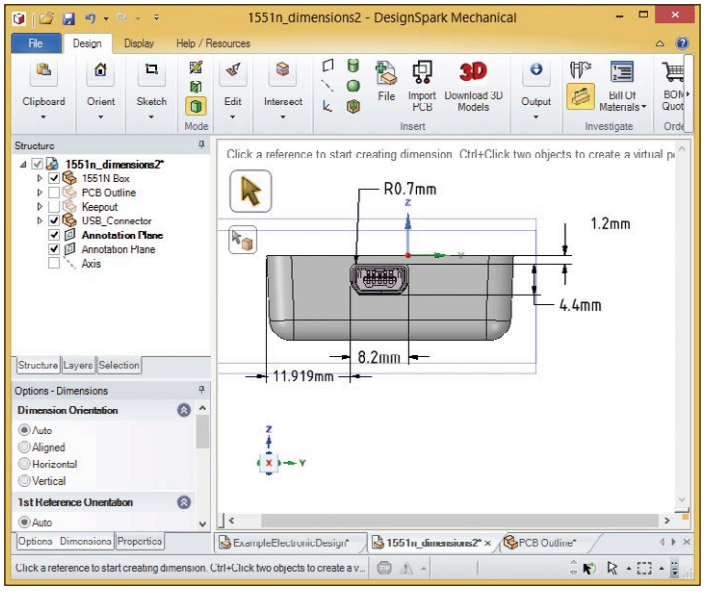


Figure 3. Dimensioned cutout.

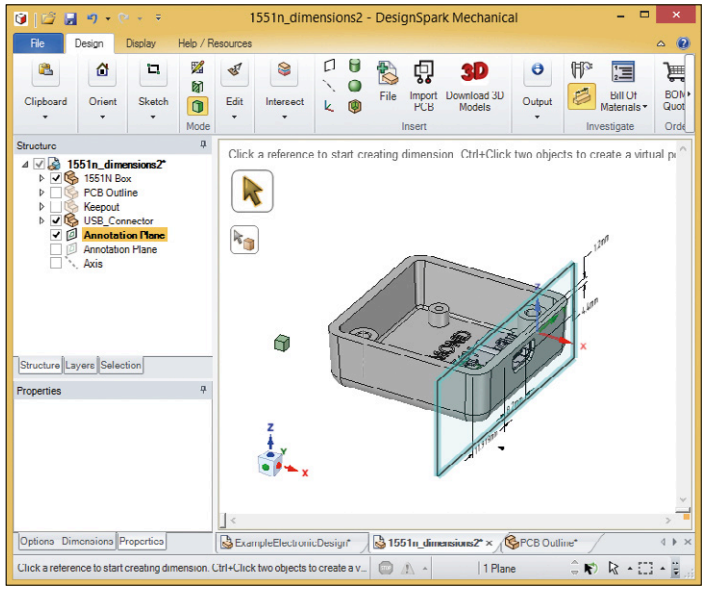
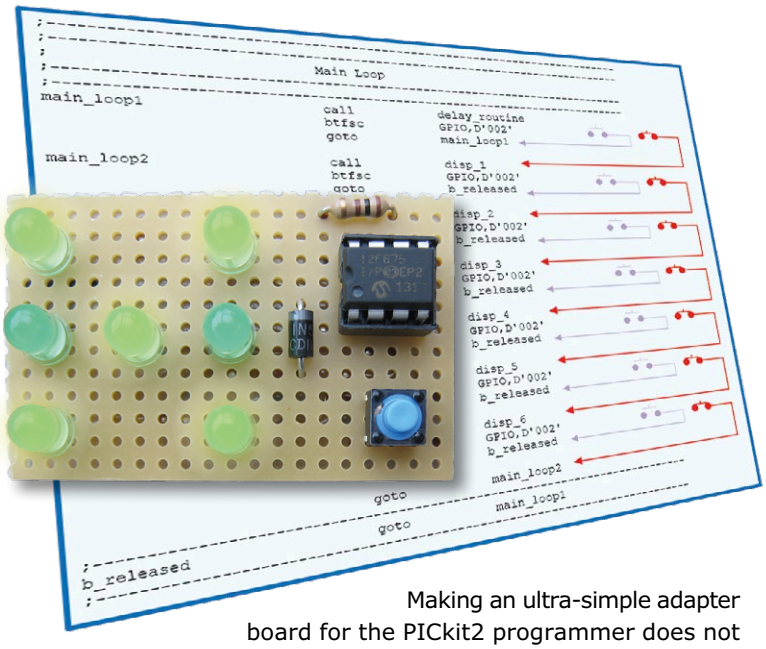


Figure 6. Dimensioned annotation plane.

PIC® Assembler Crash Course

(2) Mini dev board and electronic dice



By **Miroslav Cina** (Germany) miroslav.cina@t-online.de

In the first part of this course we learnt the basics of Assembler programming and the fundamental hardware characteristics of the PIC controller used. Now we move on, making a homebrew mini development board and learning some additional Assembler commands. For our demo application we shall program an electronic die.

Making an ultra-simple adapter board for the PICkit2 programmer does not take long. A board of this kind makes programming of compatible microcontrollers extremely easy: just plug in the chip, connect the board to the programmer, start the programming software, transfer the ready-assembled firmware and it's done. You can make adapter boards for controllers with varying pin-counts with very little effort, as every microcontroller from Microchip has these necessary programming pins: V_{pp}, ICSPDAT and ICSPCLK. With adapter boards all you need do then is to connect these pins (plus V_{cc} and GND of course) with the pins of the same name on the PICkit programmer.
For the PIC12F675 8-pin controller used here we make an 8-pin variant shown in **Figure 1**. As you can see, things don't come much simpler than this. All you need is a scrap of perfboard, an 8-pin IC socket and a 6-way header. You can even do without the LED and its series resistor, although having a visible check that the controller is powered up is a handy feature. **Figure 2** shows how the finished adapter board plugs into the programmer.

Register report

Before we discuss the additional Assembler commands needed for our demo application, let's just take a quick glance at the most important Registers.

Config Register

Almost all Registers in a microcontroller are stored in SRAM but the Configuration Register is an exception. The Register contains a total of 14 bits, of which at this stage only four bits are relevant initially.
The three LSBs (**L**east **S**ignificant **B**its = lowest value Bits) 'FOSC2:FOSC0' are provided for configuring the oscillator. In this training exercise the following two settings are used for these three bits.

010: Use crystal-controlled RF oscillator. In this mode an externally controlled oscillator is used. The frequency is determined with an external crystal connected between pins 2 (GP5) and 3 (GP4). This method enables you to have a clock rate of 20 MHz maximum. The crystal uses up two pins, so only four out of the total of six remain usable for I/O purposes.
100: Use internal oscillator. In this mode the internal oscillator is activated, with a clock rate of 4 MHz. The clock speed is

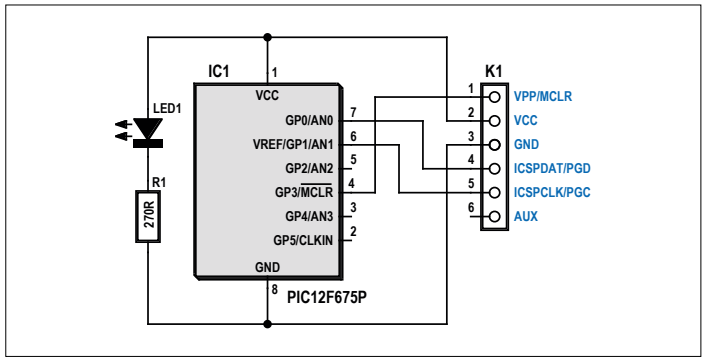


Figure 1. Schematic of the adapter board (which is really a minimalist version of a development board).

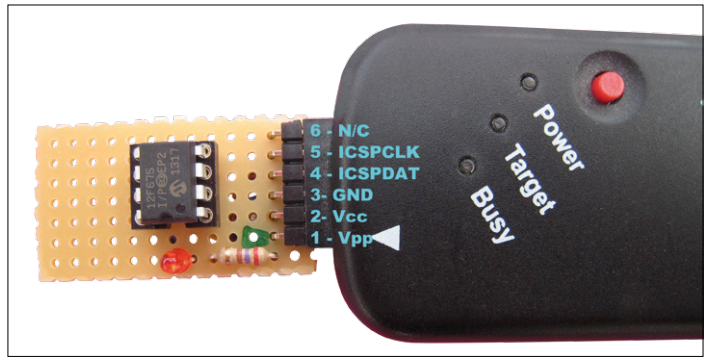


Figure 2. Finished adapter board, hooked up to the PICkit2 programmer.

IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

Figure 3. Assignment of the bits in the Status Register.

not as accurate as when using crystal control but its tolerance of ±1 % is still perfectly adequate for many applications. The great advantage is that no additional components are required, also GP4 and GP5 are available for general I/O use.

Bit 5 (MCLRE) of the Configuration Register controls the function of pin 4. If MCLRE = 1, pin 4 functions as 'Master Clear' — in other words as the reset pin. When MCLRE = 0 pin 4 can be used as an input, at the cost of losing the reset function.

Status Register

The Status Register is where we find information on arithmetic operations already carried out. **Figure 3** shows the allocation of the individual Bits of the Register.
The two MSBs (**M**ost **S**ignificant **B**its = highest value Bits) IRP and RP1 are reserved and should always be 0 when writing. They are not used at all with the PIC12F675.
Bit 5 is RP0, the Register Bank Select Bit. Its function (as already described in Part 1) is for toggling (switching within) the memory bank. When its value = 0 all subsequent commands will access the memory locations 00h to 7Fh. Alternatively when its value = 1, locations 80h to FFh are accessed.
The two bits TO and PD reveal how the last reset came about (for further details see the data sheet [1]).
Bit 2 Z (= Zero) is activated the result of the last operation is zero — this need not necessarily have been an arithmetic operation. In addition and subtraction the two LSBs indicate overflows that have arisen (carry and borrow situations).

Option Register

The Option Register enables other settings to be activated. An example is given in our demo application (see below), where clearing Bit 7 activates the pull-up function for the inputs.

Memory organization

For programming in Assembler you need to be well-versed in the memory organization of the microcontroller being used. Memory is typically (and in this case, definitely) classified into three types:

- Program memory (Flash)
- Data memory (SRAM)
- EEPROM

Program memory is used in the main for installing the software — and exclusively for this purpose in the case of the PIC12F675 we are using. A 'word' in the program memory amounts to 14 bits here. For programming this information is not particularly significant but it's definitely worth making clear that the program memory of a PIC12F1675 contains not merely 1024 **Bytes** (each of 8 bits) but 1024 **Words** (each of 14 bits).
The data memory is where all the specific processor and application Registers are located. With the exception of the predefined Processor Register, the content of the data memory is undefined at switch-on. Its content is lost when power is disconnected — we're dealing with volatile SRAM here. Unlike program memory, the data memory is organized in the classic manner using Bytes each 8 bits long. A detailed representation of the memory map can be found starting on page 9 in the data sheet [1]. It's worth noting here that the 64 Bytes of application memory lie within the address range from 20h to 5Fh. In the similarly Byte-organized EEPROM you can store software values that will safely survive any reset or reboot (we are dealing with non-volatile memory here). All the same, writing data too frequently should be avoided.

Other commands

Next follows a description of the commands required for our demo project that are in addition to those already covered in the first part of this crash course. Incidentally you will find a handy list of all commands in the datasheets for every microcontroller from Microchip [1].

Our description begins by rounding off a command that we already used in Part 1.

DECFSZ

In Part 1 we described the function of this command (= **DEC**rement **F** and **S**kip if **Z**ero) for Looping. As well as Loops we can also use the DECFSZ command for creating Branches in the program flow (a Loop is really only a special case of Branching). **Figure 4** illustrates the flow diagram of the command.
It's assumed that a Variable 'X' is located at memory location 20h. A complete Branching operation requires three commands (see **Figure 5**). By employing the Argument '0' in DECFSZ the result of the subtraction (value in 'X' minus 1) is written to the W Register. If the result is 00h, the next command is skipped and the third command 'goto Brnch_B' is carried out. This takes place only when the value of 'X' is 01h prior to decrementing 01h. In all other situations we end up at 'Brnch_A'.

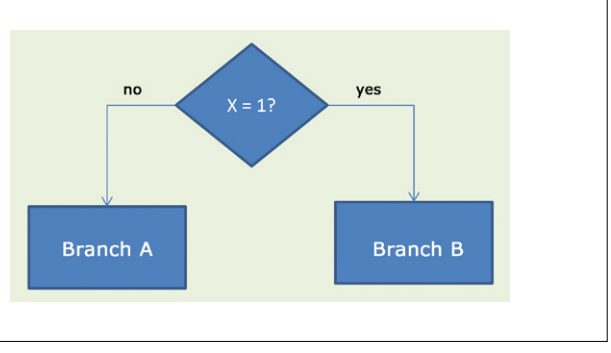


Figure 4. Flow diagram for conditional branching.

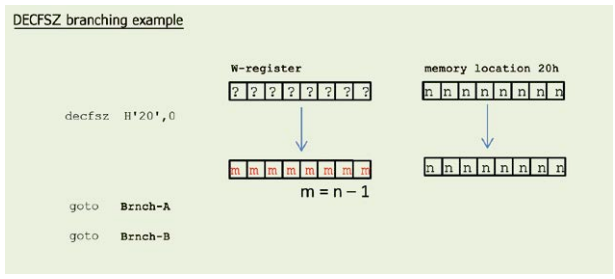


Figure 5. Sample code for conditional branching.

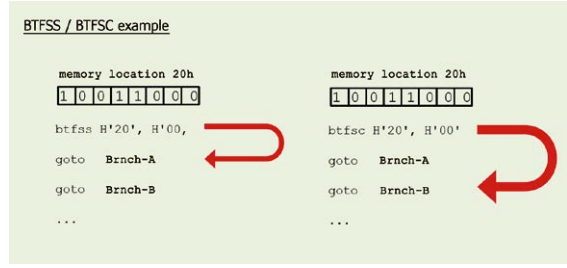


Figure 6. Sample loop using the commands BTFSS / BTFSC.

INCFSZ

The DECFSZ command has now been described in detail. INCFSZ is identical except that it handles incrementing instead of decrementing.

BTFSS and BTFSC

Branching operations also use the two commands BTFSS (= **B**it **T**est **F** and **S**kip if **S**et) and BTFSC (= **B**it **T**est **F** and **S**kip if **C**lear). Using these we can poll the status of individual Bits of a memory location and Branch off accordingly.
The syntax of the command is:

```
btfss f,b
```

and correspondingly

```
btfsc f,b
```

Argument 'f' has a value ranging from 00h to 7Fh and indicates the address of the memory location. Argument 'b' is a numerical value from 00h to 07h and defines the Bit to be checked. These commands do not alter the content of the W Register nor of the memory location.

In the example shown in **Figure 6** a value of 98h (= 10011000b) is stored in memory location 20h. On the left-hand side the command BTFSS checks whether the value of Bit 00h is '1'. If yes, we skip the next command 'goto Brnch_A'. When the value of Bit 0 is '0', however, it is not skipped and instead the command 'goto Brnch_A' is carried out.
On the right-hand side we see the operation of the command BTFSC. Here we skip when Bit 0 has the value '0' — which applies to the value at memory location 20h. The command 'goto Brnch_A' is therefore skipped and the program resumes with the command 'goto Brnch_B'.

ADDWF

This command adds the content of the W Register to the content of a memory location.
The syntax is:

```
addwf   f,d
```

Argument 'f' stands for the target address and has a value of from 00h to 7Fh. Argument 'd' can be either '0' or '1' and defines where the result is to be stored. If d = 0, the result is saved in the W Register and the memory location remains

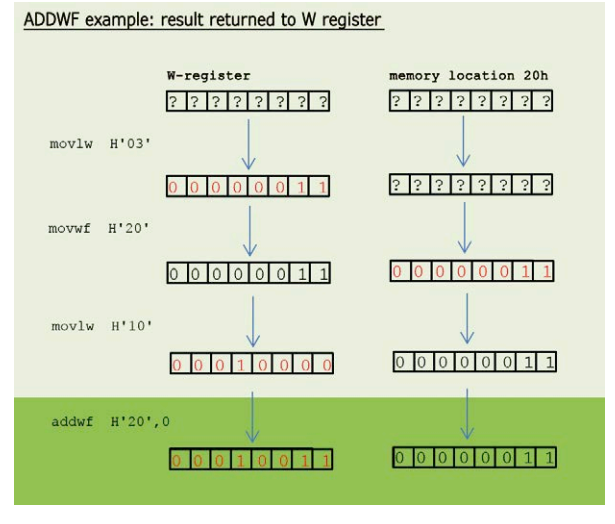


Figure 7. Sample code for ADDWF: result written to W Register.

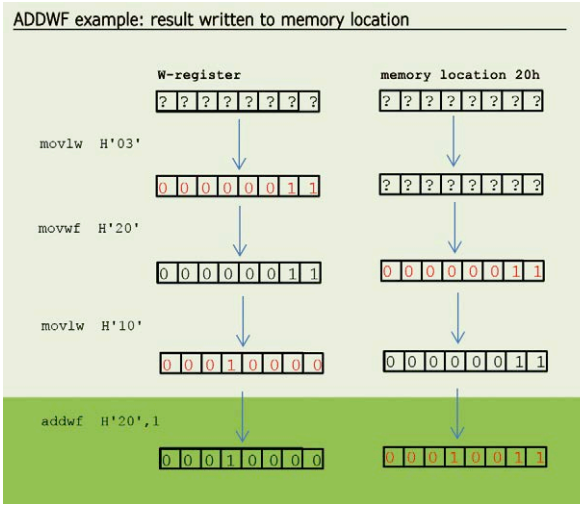


Figure 8. Sample code for ADDWF: result written to SRAM memory location.

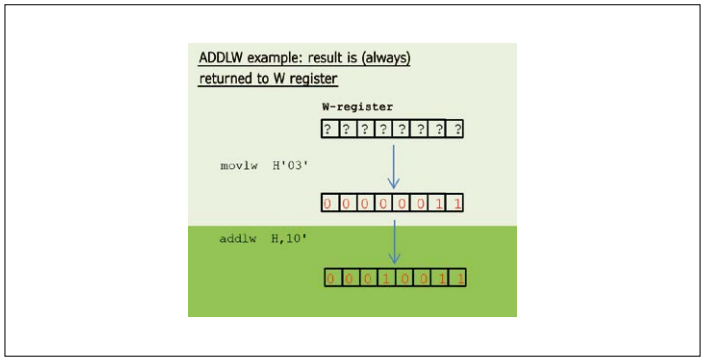


Figure 9. Addition of a Constant to the W Register.

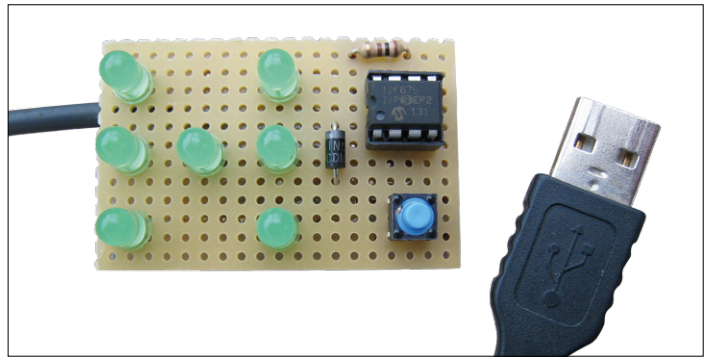


Figure 10. Prototype of our electronic die.

untouched. When d = 1 the store position is overwritten with the result.

In the following example the sum of 3 + 16 (decimal) is calculated. First we deposit the '3' into memory location 20h, after which '16' is written to the W Register and then the addition is carried out. The result is written either into the W Register (**Figure 7**) or in memory position 20h (**Figure 8**).

ANDWF
As regards syntax and handling this command works identically to ADDWF. The difference lies in the function: ANDWF executes the logical operation 'AND' bitwise (bit-by-bit).

ADDLW
ADDLW also executes an addition in a similar way to the command ADDWF. However, ADDWF adds a value in memory to the W Register, whereas ADDLW adds a Constant to the W Register.
The syntax is:

```
addlw k
```

Argument 'k' has a value of from 00h to FFh and is the Constant that is added to the contents of W Register. The result is saved to the W Register.

In the example shown in **Figure 9** we first load the value 03h into the W Register, after which the Constant 10h is added.

ANDLW
Apart from an AND operation instead of an addition, this command is identical to ADDLW.

CLRF and CLRW
The CLRF command appeared already in Part 1 (a specified memory position is set to 00h). Analogously to this, the CLRW command sets the W Register to 00h.
The syntax is:

```
clrf f
```

and correspondingly for

```
clrw
```

INCF and DECF
Using these command we can increment (INCF; +1) or decrement (DECF; -1).
The syntax is:

```
incf f,d
```

and correspondingly

```
decf f,d
```

Arguments 'f' and 'd' have the exact same meaning as in the ADDWF command. The result is written either to the W Register or at the original memory location, according to 'd' (either '0' or '1').

Electronic dice
Creating an electronic die (or several dice, which as you know is technically the plural of the word die!) is no more complicated than making an LED flash. In **Figure 10** you will see a USB connection, as (for simplicity's sake) the circuit is powered from a USB port on a PC. Beyond this the die has nothing else to do with the USB interface. You could power it just as easily using a wall-wart power supply or with three standard or rechargeable batteries connected in series.

If you look closely at some real dice you will note that the spots on them are arranged in seven different positions. The appearance of the individual numbers follows the scheme shown in **Figure 11**. If you use LEDS to represent the spots, this will use up exactly four outputs of a microcontroller. If that surprises you, here is how we do it: LEDs 6 and 7, 2 and 3 together with 4 and 5 are always lit or unlit together. These pairs are therefore always controlled together by a single GPIO port pin. LED1 needs one control wire only, with R1 wired in series so that it does not light up brighter than the others.

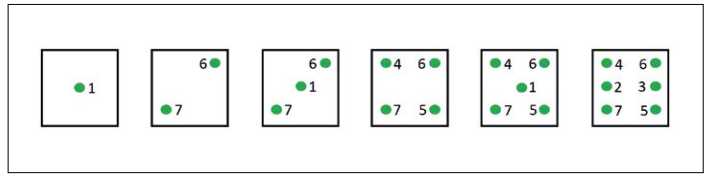


Figure 11. The numeric values of a die depicted.

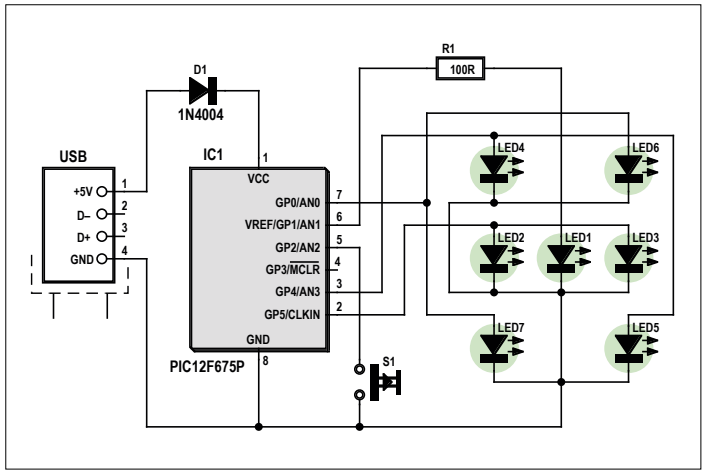


Figure 12. Schematic of an electronic die.

Connecting LEDs direct to GPIO-Pins (see schematic in **Figure 12**) is definitely not rule-conformant but it's not really a problem, considering the low currents involved. To do our bit for energy-saving the 5 V from the USB port is dropped by around 0.7 V to 4.3 V using D1. If you are using three NiMH rechargeables (3 x 1.2 V) as your power source you can omit the diode (and fit a wire link instead). You can 'throw a die' by pressing push button S1 briefly.
One more little tip: because of their higher forward voltage, blue and white LEDs will not work in this circuit.

Dice firmware

Before we go poking buttons, it's worth examining precisely what the software needs to do with them. The specification for these dice looks like this:
At start-up (reset) the die should display a demonstration. This demo is executed, displaying each of the counts (1 to 6) sequentially, until press button S1 is pressed.

```
;*****************************************************************
;*                          Dice                                 *
;*****************************************************************
;*                    v 1.05 - 14.03.2015                        *
;*****************************************************************
;*       Hardw.: PIC12F675 used                                  *
;*       OSC.....: Internal osc. used    POWER.....:  5V USB     *
;*****************************************************************
;-----------------------------------------------------
;Pins connection:
;-----------------
;GP2 -- button
;GP3 -- N/C
;-----------------------------------------------------
;             GP4                    GP0
;
;
;
;             GP5          GP1       GP5
;
;
;
;             GP0                    GP4
;-----------------------------------------------------

 INCLUDE "P12F675.INC"

_U002                                EQU     B'00000110000100'          ;MCLR = input
            __CONFIG     _U002

;*****************************************************************
;       Variables                       s
;*****************************************************************
TIMER1                       EQU     H'20'                  ;Used in delay routine
TIMER2                       EQU     H'21'                  ; "          "
;-----------------------------------------------------------------      Dice

                             bsf     STATUS,RP0             ;Switch to register bank 1
                             movlw   B'11001100'            ;GP0, 1, 4, 5 = out; GP2, 3 = in
                             movwf   TRISIO
;
                             clrf    ANSEL                  ;GPIO are digital I/O's

                             bcf     OPTION_REG,D'007'      ;enable pull-ups
                             movlw   B'00000100'
                             movwf   WPU                    ;enable pull-up for GP2

                             bcf     STATUS,RP0             ;Switch Back to reg. Bank 0
```

Figure 13. Dice code: Initialization.

The dice logic is implemented as an endless loop:
If S1 is pressed it counts up through the values 1 to 6 continually and displays these. The persistence of vision makes it looks as if value 7 is being displayed (all LEDs illuminated). The rapid counting ceases when S1 is released. The current count status is then displayed permanently.
You're now ready to start.

The firmware for the die is divided into two parts: the die and the display. It goes without saying that a fair amount needs to be initialized following a reset or starting from scratch.
In the code shown in **Figure 13** you can see this configuration where we put marker 1. The internal oscillator and GP3 are defined as the input — consequently no MCLR.
At marker 2 Variables TIMER1 and TIMER2 are defined at memory locations 20h and 21h.
At markers 3 and 4 we are operating in memory bank 1, which is why Bit RP0 of the Status Register is set to '1' initially.
At marker 3 the first two commands (movlw B'11001100' and movwf TRSIO) arrange for GP0, GP1, GP4 and GP5 to be configured as outputs, at the same time as the relevant Bits of the TRISIO Register are set to '0'. The four LEDS are connected to these four pins. GP2 is configures as an input that is used to poll the press button status. To achieve this Bit 2 of the TRISIO Registers is set to '1'. The command 'clrf ANSEL' disables the analog functionality.
At marker 4 an external pull-up resistor for the press button is made redundant. The PIC12F675 microcontroller makes so-called 'weak pull-ups' available for all GPIO pins (apart from GP3) configured as inputs, you see. For this purpose Bit 7 of the Option Register must be set to '0'. The command 'bcf OPTION_REG,D'007'' handles this in the program. Next we use the Bits of the WPU Register to determine for which Pins the internal pull-ups should in fact be active. If Bit x of the WPU Register is set to '1', the pull-up resistor of the relevant GPx pin is active. The internal pull-ups are not active by default, as they can consume up to 400 µA additional current. Where it matters, you can then install a high-resistance external pull-up or if necessary, even an external pull-down.

Following initialization the program jumps into the demo loop (**Figure 14**). Implementation is very simple at this stage — everything takes place in the subroutine 'start_effect', which is invoked here simply. The demo section is cancelled by pressing S1.

Demo routine

The demonstration routine in **Figure 15** is very straightforward. You will recognize six near-identical program segments for each numeric result. A subroutine is employed for representing the numbers: sub-program 'disp_1' looks after lighting the correct LEDs for '1'. Sub-program 'disp_2' handles 2' and so on. After a count is displayed, the software waits briefly for the next one. The 'delay_routine' command — already used in the first part of this course — handles this. An interesting point at this stage is the command: btfss GPIO,D'002'. This ensures that if the push button is not pressed, the 'Return' command is skipped and the next count is displayed. When the button is pressed the skipping stops, as the input is now presenting a '0'. The program is resumed by hitting 'Return'. This again

makes sure that the subroutine 'start_effect' and hence the demo are closed. The main program now restarts.
Now for the display routines. The code in this section (**Figure 16**) is particularly straightforward. There are six routines in total. In each of these the first command 'movlw' uses a Constant to determine which LEDs should be on and off. In the binary Constants a '0' stands for LEDs that stay dark and a '1' those that are illuminated. The 'movwf GPIO' command passes this information to the Port. A brief pause (sub-program 'dr2') follows this and then the display routine ends. In the demo section this short delay would not be required but it does not cause any further problem either.

Main program

This section is constructed on the same lines as the demo routine. In **Figure 17** the section with the label 'main_loop1' ensures that when the button is not depressed, the current count is displayed constantly. In the main program we also poll the current value of input GP2 and, once the button is pressed, continue counting and displaying the current figure, just as we did in the demo routine. The whole thing happens so rapidly that to all appearances the figure '7' is being displayed. After releasing the button the 'goto b_released' command is performed and the counting stops. The current status is displayed permanently. This numeral is the result of throwing the die. On the line labeled 'b_release' we jump straight back to 'main_loop1' and consequently the program flow is stopped. We could have also jumped straight back to 'main_loop1' but if you had the intention to add extra features to the program (to give it more life), you could implement extra effects at the label 'b_release'. You could for instance fade out the count slowly or flash the result, and only after this revert to the static display. The sub-program 'dr2' establishes how rapidly the figures are changing. Actually you could omit this altogether. For test purposes you could slow down the running of the program enough to make visible how the figures are altered.
As always, the software can be downloaded from the Elektor website [4].

Outlook

With this we have reached the end of the second part of this Assembler course. A couple of new commands have been described and we have shown how you can create working electronic dice with little effort. Quite complex applications can be achieved along the same lines. Hopefully it is now clear to you that Assembler programming really is simpler than you imagined.

I hope that you enjoyed this too! The next installment will demonstrate how you can use a simple microcontroller like the PIC12F675 as a substitute for the NE555 timer. ⏮

(150274)

```
;-------------------------------------------------------------------
;-------------------------------------------------------------------
;                       Effect
;-------------------------------------------------------------------
;-------------------------------------------------------------------
                        clrf        GPIO
                        call        start_effect
```

Figure 14. Dice code: Jumping straight to the demo routine.

```
;-------------------------------------------------------------------
start_effect            call        disp_1
                        call        delay_routine
                        btfss       GPIO,D'002'
                        return

                        call        disp_2
                        call        delay_routine
                        btfss       GPIO,D'002'
                        return

                        call        disp_3
                        call        delay_routine
                        btfss       GPIO,D'002'
                        return

                        call        disp_4
                        call        delay_routine
                        btfss       GPIO,D'002'
                        return

                        call        disp_5
                        call        delay_routine
                        btfss       GPIO,D'002'
                        return

                        call        disp_6
                        call        delay_routine
                        btfss       GPIO,D'002'
                        return

                        goto        start_effect
;-------------------------------------------------------------------
```

Figure 15. Dice code: Outputting the figures in the demo routine.

```
;-------------------------------------------------------------------
disp_1                  movlw       B'00000010'
                        movwf       GPIO
                        call        dr2
                        return
;-------------------------------------------------------------------
disp_2                  movlw       B'00000001'
                        movwf       GPIO
                        call        dr2
                        return
;-------------------------------------------------------------------
;...
```

Figure 16. Dice code: Display routines.

```
;-------------------------------------------------------------------
;-------------------------------------------------------------------
;                       Main Loop
;-------------------------------------------------------------------
;-------------------------------------------------------------------
main_loop1              call        delay_routine
                        btfsc       GPIO,D'002'
                        goto        main_loop1

main_loop2              call        disp_1
                        btfsc       GPIO,D'002'
                        goto        b_released

                        call        disp_2
                        btfsc       GPIO,D'002'
                        goto        b_released

                        call        disp_3
                        btfsc       GPIO,D'002'
                        goto        b_released

                        call        disp_4
                        btfsc       GPIO,D'002'
                        goto        b_released

                        call        disp_5
                        btfsc       GPIO,D'002'
                        goto        b_released

                        call        disp_6
                        btfsc       GPIO,D'002'
                        goto        b_released

                        goto        main_loop2
;-------------------------------------------------------------------
b_released              goto        main_loop1
;-------------------------------------------------------------------
```

Figure 17. Dice code: Main program.

Web Links

[1] PIC12F675: www.microchip.com/wwwproducts/Devices.aspx?product=PIC12F675

[2] MPLAB IDE: www.microchip.com/pagehandler/en-us/family/mplabx

[3] First part of this course: www.elektormagazine.com/130483

[4] Software: www.elektormagazine.com/150274

I/O App

Control external hardware with a Windows app

By Dr Veikko Krypczyk (Germany)

The basics of how to program apps for current versions of Microsoft Windows are fairly simple to master. Here we use one such app to control LEDs and relays and to display the status of inputs on a PC or tablet.

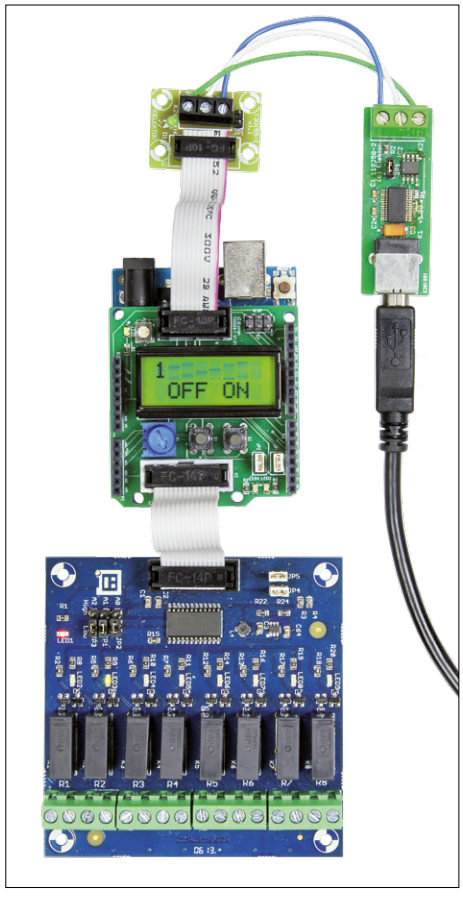


Figure 1. Modular hardware brings greater flexibility.

Windows Store 'apps' are modern in design, cut down to the essentials in functionality, and fun to use. In the 'Learn' section of the previous edition we gave a handy introduction to programming Windows Store apps for the Windows 8.1 operating system.

For electronics hobbyists communicating with external hardware is of particular interest, with a wide range of possible uses. Touch-based operation and a modern, minimalistic user interface are ideal for many control applications. The apps can be run on a conventional desktop PC, on a notebook or on a tablet. Home-made microcontroller-based hardware can be connected to the USB port, allowing communication to occur in both directions.

Hardware

For our simple project we will make use of an Arduino Uno single-board computer equipped with some expansion hardware that has appeared several times in previous Elektor magazine projects. At Elektor Labs we have designed our own shield [2a] that allows external hardware such as relays to be connected to the Arduino. The board includes two LEDs that can be used for testing, an LCD panel, two buttons and a potentiometer. On the input side the shield is connected to the computer over USB via an RS-485 module and an USB-to-RS-485 converter. RS-485 signaling is relatively immune to interference and so connections can be made over long distances. But the most important aspect is the possibility of connecting additional hardware, and this is readily done using the 14-pin Embedded Extension Connector provided. The connector is also known as a Gnublin connector, as the Gnublin module designed by Benedikt Sauter and his team can also be attached here. Another module that can be connected is an expansion board carrying eight relays. The hardware modules and their connections are shown in **Figure 1**, and the overall set-up is described at [2b].

Simple control commands

The highlight of this project is the firmware, which is described at [2b] and which can be downloaded via the web page accompanying this article [1]. Using a simple protocol (see [2c]) ASCII commands are sent to the Arduino Uno. For example, the command 'L 1 0 + <CR>' turns on the LED on the shield. If the relay board is connected, one of the eight relays can be turned on using the com-

mand 'R 0 X + <CR>', where X is a digit from 0 to 7 that determines which relay is affected. If the '+' is replaced by a '-' in these commands, the LED or relay will be turned off.

The state of the buttons on the shield can be interrogated using the commands 'B 0 0 ? <CR>' and 'B 0 1 ? <CR>'. The command 'A 0 0 # <CR>' returns the position of the potentiometer as a value from 0 to 1023 represented in hexadecimal.

Virtual COM port

In order to have our app interact with the hardware we simply need to be able to send and receive data over a virtual COM port, or VCP. The COM port is 'virtual' because in fact the characters are sent back and forth over USB, with the Windows VCP driver emulating a standard serial interface as far as the higher-level software, such as a terminal emulator or our own app, is concerned.
Real COM port serial interfaces are increasingly rarely found on modern PCs, but the advantage is that the COM port API provided by the operating system makes setting up communications easy. To control our hardware we install a virtual COM port driver supplied by device manufacturer FTDI. Our program then sees a serial COM connection and can use it to transmit and receive data.

A terminal emulator program can be used to check that everything is working as it should. It is necessary to tell the terminal emulator which COM port is allocated to the FTDI chip on the USB-to-RS-485 converter board, and to set its communication speed to 9600 baud.

Tricky situation

It is easy to use the .NET framework to send and receive characters over a (virtual) serial interface using the dedicated 'SerialPort' class. The class encapsulates the Windows API calls that have to do with controlling a serial port. The properties of the class can be managed to set the communication speed, the name of the port and other parameters. Simple methods ('Write', 'Read') allow data to be written and read. This approach works without problems in a conventional Windows desktop application. However, Store apps are unfortunately not allowed to have direct access to the class: the apps run isolated from the system environment and from other applications in a 'sandbox' provided by the operating system. The sandbox approach improves security, but comes with disadvantages. Direct access to many system functions, to sensors and in particular to external hardware is restricted or impossible. Also, even in the case where the required access is possible, it must be requested by declaring it in the application manifest when the program is developed. Then, when the app is installed, the user must agree to the requested access permissions. In general it is not possible to use system functions for direct access from an app, and in particular the Win32 APIs are not available for use by Store apps. In some cases alternatives are available (see **Table 1**), but unfortunately not in the case of the serial interface.
However, there are some potential solutions. With the release of Windows 8.1 Microsoft has made improvements that allow (generic) access to the USB port. This is done using the 'Winusb.sys' system library, which must be available on the target computer. From the app side we use the classes in the namespace Windows.Devices.Usb [5]. There is a description at [6] of how to talk to the FTDI device using this method, which it would be feasible to use in our case. However, most readers will be more familiar with working with virtual COM ports, and so we have tried to find an alternative approach which allows us to get around the limitations of the sandbox.

Kommunikation meistern

Die Lösung besteht darin, in die Projektmappe eine *Brokered Windows Runtime Component* einzubinden [7]. Nachfolgend werden die einzelnen Schritte beschrieben, um den Zugriff auf die Hardware über einen COM-Port (via USB) zu ermöglichen [8]:

1. First create an app as explained in the article in the 'Learn' section of this issue.
2. Using the Solution Explorer add two further projects based on the 'Brokered WinRT Component Project Templates'. If these templates are not available on the development machine they must be located on-line (see **Figure 2**). The first of the two projects that are added is a C# project, which we call 'SerialPortSampleAppBroked': it is this project which will contain the code that carries out the communication using the SerialPort class, implementing the interfaces that will be needed by the app. However, the project cannot be included directly as a reference in the app, as this will be rejected by the compiler; to obtain access to the interface we must do so indirectly, via a proxy. The proxy is a separate project (writ-

Table 1. Alternative libraries for low-level access from Store apps [4].

System function under Win32 (for desktop applications)	Alternative API for Windows Store apps
Bluetooth	Windows.Networking.Proximity
Device enumeration (Function Discovery, PnP-X, WSD)	Windows.Devices.Enumeration
FAX	access not possible
Location API	Windows.Devices.Geolocation
Print	Windows.Graphics.Printing
Sensors	Windows.Devices.Sensors
Serial and parallel ports	access not possible
SMS	Windows.Devices.Sms
UPnP	Windows.Devices.Enumeration.Pnp
Windows Portable Devices	Windows.Devices.Portable
WSD	Windows.Devices.Enumeration

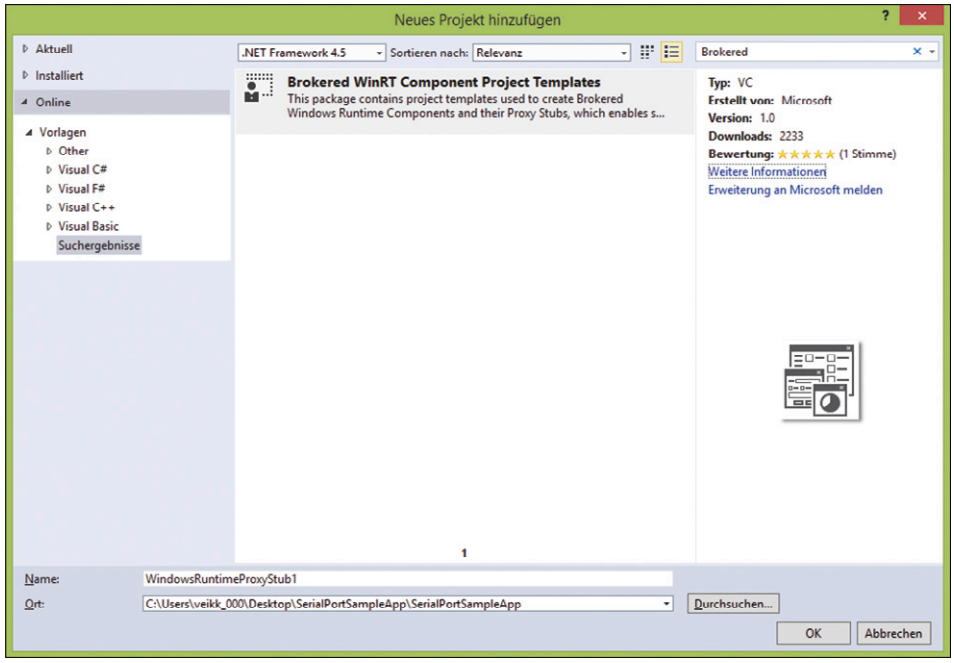


Figure 2. Adding a project (using the Brokered Windows Runtime Component template).

ten in C++) within the solution, and we have decided to call it 'SerialPortSampleProxy'. No code is implemented within the proxy project: it acts only as a bridge between the Store app and the Windows components.

3. With all the projects in place we can now set up the necessary references. In the Solution Explorer under 'References' for the proxy project 'SerialPortSampleProxy' add a reference to the 'SerialPortSampleAppBroked' project. Then, likewise, add a reference from the 'SerialPortSampleApp' project to the proxy project: see **Figure 3**.

The overall architecture is shown in **Figure 4**. Although the procedures above seem complicated they do not really present an obstacle to the developer. Once the references between the projects have been set up, we do not need to be concerned with them further as we develop our app, and we can simply use the SerialPort class to provide communication. However, there is a disadvantage that should not be overlooked: the use of 'Brokered Windows Runtime Components' is regarded by Microsoft as a critical security weakness, and so it is not possible to distribute the app via the Store. The only way to install the app is via 'side loading', which requires a developer account on the target device. Side loading is therefore only appropriate for apps that are not intended for widespread public distribution [9]; for homebrew projects this is not too severe a limitation.

Example app

We implemented the ideas above in a small app which allows a couple of outputs to be activated and deactivated using buttons, and which also displays the status of the hardware inputs (see **Figure 5**). The user interface can easily be adapted to suit any particular application, adding simple icons to support intuitive touch-based operation. The XAML code that defines the user interface is shown in **Listing 1**.

Listing 2 shows the SP class that we have written ourselves, which must be included in the 'SerialPortSampleAppBroked' project. Output of data on the serial interface is done using the 'Write' command, and likewise input is done using the 'Read' command.

The listing shows only the constructor for the SP class. It is inside the constructor that the relevant port is selected and the various parameters, including the baud rate, are initialized. Furthermore the constructor can also accept a parameter in the form of a command string as described above, which will be output to the port. This makes accessing the port from the app

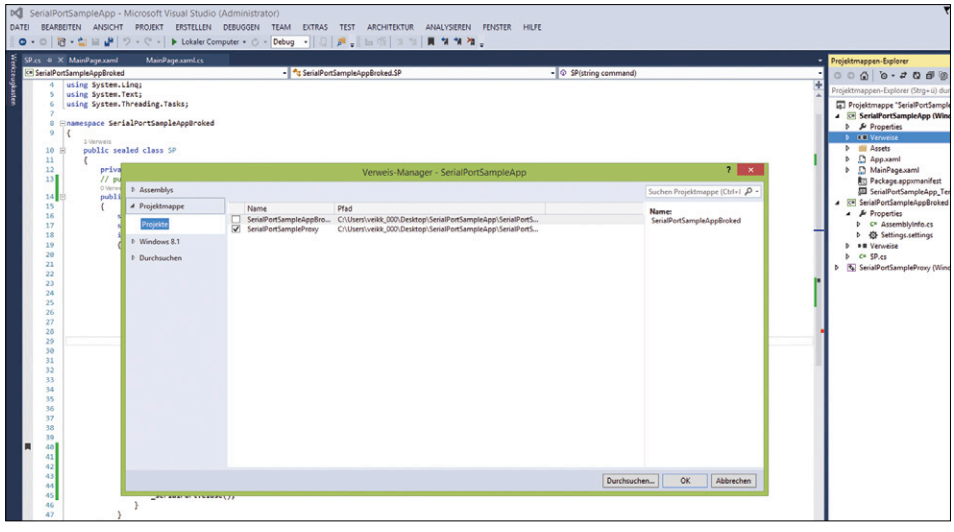


Figure 3. Reference from the app to the proxy project.

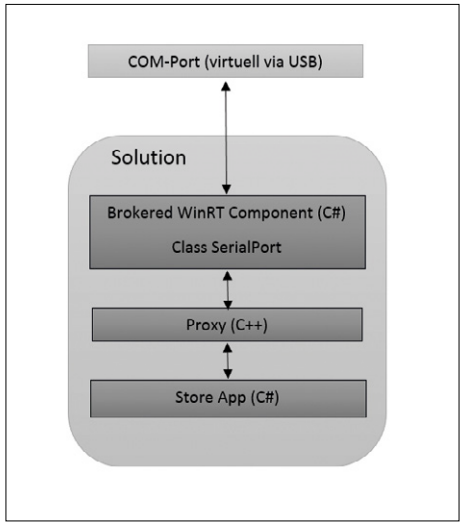


Figure 4. Overall software architecture to allow access to the COM port from an app.

very straightforward: for example, sending the command 'L 1 1 + <CR>' can be done in a single line as follows.

```
SerialPortSampleAppBroked.SP client
= new SerialPortSampleAppBroked.SP("L
1 1 +\r\n");
```

Other methods in the SP class allow the inputs to be interrogated by reading from the serial port. If it is desired to react to a change in levels on the inputs they must be polled in sequence, for example under control of a timer (see the DispatcherTimer class [10]).

Figure 5. The example app can control LEDs and relays.

Listing 1. Excerpt from the XAML code that describes the user interface for the example app.

```
<Page
    x:Class="SerialPortSampleApp.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:SerialPortSampleApp"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">
    <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
        <Grid.RowDefinitions>
            <RowDefinition Height="150"/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
        </Grid.RowDefinitions>
        <Image Source="Assets/SplashScreen.png" HorizontalAlignment="Left" Margin="50,20"/>
        <StackPanel Orientation="Horizontal"  Grid.Row="1" Margin="20">
            <ToggleButton Margin="10"  Width="200" Height="100"  Content="LED 1" FontSize="24"/>
            <ToggleButton Margin="10"  Width="200"  Height="100" Content="LED 2" FontSize="24"/>
        </StackPanel>
        <StackPanel Orientation="Horizontal"  Grid.Row="2" Margin="20">
            <ToggleButton Margin="10"  Width="200" Height="100" Content="Relay 1" FontSize="24"/>
            <ToggleButton Margin="10"  Width="200" Height="100" Content="Relay 2" FontSize="24"/>
            <ToggleButton Margin="10"  Width="200" Height="100" Content="Relay 3" FontSize="24"/>
            <ToggleButton Margin="10"  Width="200" Height="100" Content="Relay 4" FontSize="24"/>
            <ToggleButton Margin="10"  Width="200" Height="100" Content="Relay 5" FontSize="24"/>
            <ToggleButton Margin="10"  Width="200" Height="100" Content="Relay 6" FontSize="24"/>
            <ToggleButton Margin="10"  Width="200" Height="100" Content="Relay 7" FontSize="24"/>
            <ToggleButton Margin="10"  Width="200" Height="100" Content="Relay 8" FontSize="24"/>
        </StackPanel>
        <StackPanel Orientation="Horizontal"  Grid.Row="3" Margin="20">
            <ToggleButton Margin="10"  Width="200" Height="100" Content="Button 1" FontSize="24"
IsHitTestVisible="False"/>
            <ToggleButton Margin="10"  Width="200" Height="100" Content="Button 2" FontSize="24"
IsHitTestVisible="False" IsChecked="True"/>
        </StackPanel>
    </Grid>
</Page>
```

Conclusion

Apps are attractive because of their ease of use and focus on a single function. Windows Store apps can run on tablets and on notebooks; in most cases these have at least one USB port and so an app can offer an alternative to a conventional desktop program for controlling external hardware. The required programming, with the logic expressed in C# and the user interface in XML, is by no means a black art; and it is possible to make the user interface simple and attractive, following the maxim 'less is more'. There are indeed technical obstacles in interfacing to the hardware, but these can be overcome. ⏮

(150276)

Listing 2. C# code for communicating over the COM port.

```
public sealed class SP
    {
        private SerialPort _serialPort;
        public SP(string command)
        {
            string portName = "";
            string[] ports = SerialPort.GetPortNames();
            if (ports.Count() > 0)
            {
                int portInt = 0;
                foreach (var port in ports)
                {
                    var portNumber = port.Substring(port.Length - 1);
                    if (portInt < Int32.Parse(portNumber))
                    {
                        portInt = Int32.Parse(portNumber);
                    }
                }
                if (portInt != 0)
                {
                    portName = "COM" + portInt;
                    _serialPort = new SerialPort(
                        portName,
                        9600,
                        Parity.None,
                        8,
                        StopBits.One);
                    _serialPort.ReadTimeout = 200;
                    if (_serialPort != null && _serialPort.IsOpen)
                        _serialPort.Close();
                    _serialPort.Open();
                    _serialPort.Write(command);
                    _serialPort.Close();
                }
            }
        }
    }
```

Web Links

[1] www.elektor-magazine.com/150276

[2a] Elektor July & August 2014: www.elektormagazine.com/140009

[2b] Elektor November 2014: www.elektormagazine.com/140328

[2c] Elektor June 2013: www.elektormagazine.com/130154

[3] www.ftdichip.com/Drivers/VCP.htm

[4] https://msdn.microsoft.com/en-us/library/windows/apps/hh464945.aspx

[5] https://msdn.microsoft.com/en-us/library/windows/hardware/dn303342%28v=vs.85%29.aspx

[6] www.ftdichip.com/Support/Documents/InstallGuides/AN_271%20D2xx%20WinRT%20Guide.pdf

[7] https://msdn.microsoft.com/en-us/library/windows/apps/dn630195.aspx

[8] http://ioi.solutions/serial-port-access-metro-style-app/

[9] https://technet.microsoft.com/en-gb/windows/jj874388.aspx

[10] www.wpf-tutorial.com/misc/dispatchertimer

Tips and Tricks

From readers for readers

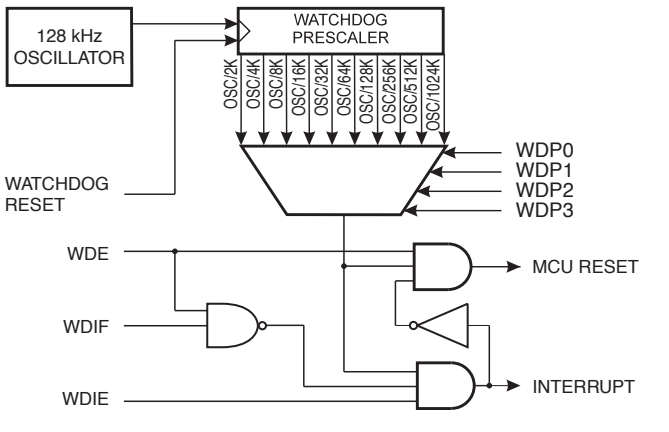
Here are some more neat solutions from our readers, sure to make life a little easier for engineers and electronics tinkerers.

Safety Tip for AVR Controllers

From Andreas Riedenauer (FAE), Ineltek Mitte GmbH

The Watchdog Timer (WDT) issues a reset to the controller if it stops functioning correctly or 'hangs'. This situation can occur not only as a result of a software failure but also by a powerful electromagnetic event or exposure of the controller to x-rays from sources such as cosmic radiation.

In normal operation the WDT will continually be prevented from generating a reset by the regular execution of the WDR (Watch Dog Reset) instruction performed in a loop somewhere in the program. If the instruction is not issued in sufficient time the timer runs out and issues a system reset. When this occurs, the Special Function Register (SFR, also known as the IO register) and the ports are returned to their default state. They will be configured as inputs with inactive pullups. It is important to ensure that this state does not cause a problem for the circuitry connected to the I/Os, in particular if they are used to control the current to inductive loads where a loss of signal could allow unacceptable levels of current (saturation!).



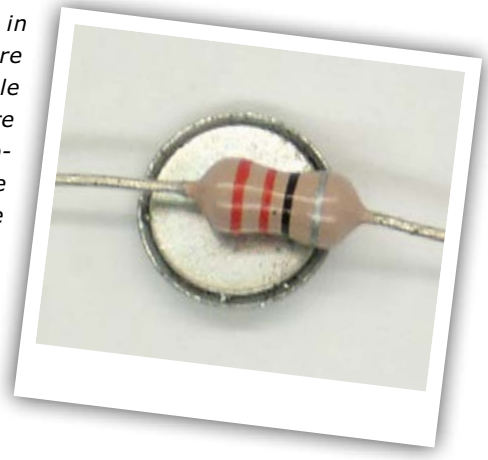
Source: Atmel ATmega644 data sheet

Sometimes operation of the controller's clock crystal can be influenced and even stopped by severe mechanical vibration; the watchdog mechanism will come to the rescue here and protect the power driver stage, but beware; should the system clock stop just as the WDR command is executed (according to Murphy's Law the probability of this unlikely event occurring is equal to 1) the outputs will remain frozen in their last state! The advice here is to make sure the WDR command is issued when all other outputs are in a benign state i.e. if they were to stay in that condition for any length of time, no harm would result.

Variometer Tuning

From Burkhard Kainka

The fixed value inductor shown has a nominal value of 22 µH. When a magnet is brought close to the coil it has the effect of reducing its inductance. In general if a ferrite core is exposed to a strong magnetic field it can become partially or fully saturated, decreasing its permeability. Using this technique the value of inductance can be adjusted in a range of 1:10 and can be used to alter the resonant frequency of an LC filter.



This type of variometer or 'permeability tuning' is in fact used in some radio receiver designs (particularly in older car radios) where it replaces the variable tuning capacitor. There can however be a problem with damping. While the inductive resistance at higher frequencies falls, the series damping resistance of the coil remains constant, reducing the filter's Q factor. Using variable capacitor tuning, the bandwidth at the upper end of the frequency band is about the same but with variometer tuning the bandwidth becomes wider. Partially magnetizing the ferrite rod of a medium-wave direct-conversion receiver demonstrated this to good effect. Tuning at the lower frequency end of the spectrum is easy but the upper frequency range has poorer sensitivity. ⏮

(150227)

Got a neat solution for a tricky problem? Using components or tools in ways they were never intended to be used? Think your idea to solve a problem is better than the usual method? Have you discovered a work-around that you want to share with us and fellow makers? Don't hang around; write to us now, for every tip we publish you'll earn 40 pounds!

FFT Analysis in VB

Display spectral curves over time too

By **Kurt Diedrich** (Germany)

Versatile as it is, the oscilloscope is not normally up to the task of examining the frequency components contained in a signal. But the combination of a PC — the other universal tool — and suitable software renders them superbly. This article explains how you can achieve this using Visual Basic.

Fourier analysis, named in honor of the French mathematician who developed it in 1822, identifies in purely arithmetical terms the amplitude of the frequencies contained in a signal. On this basis the FFT (**F**ast **F**ourier **T**ransformation) commonly used today was developed in 1965 by James Cooley and John W. Tukey. With its help we can represent a signal as a two-dimensional diagram, in which its amplitude is plotted over frequency (**Figure 1**).

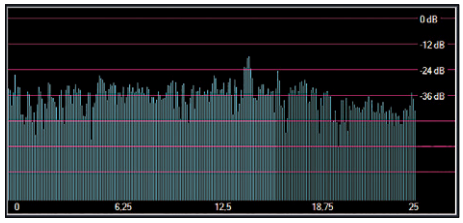


Figure 1. Result of a classic FFT: intensity display of the frequencies (up to 25 Hz here) contained in a signal extract as lines of varying height.

This has been exploited for decades in physical formats; besides simple implementations such as LED bar-graph displays based on fixed bandpasses and used largely as eye candy, for laboratory applications there are also special very accurate and correspondingly expensive spectrum analyzers, with which we can display the frequencies present in a signal as a spectrum. Today, however, the processing power of normal PCs extends easily to handling digitized (audio) signals not only in file form but also to calculating informative spectral displays from this data at lightning speed, without any need for specialized hardware.

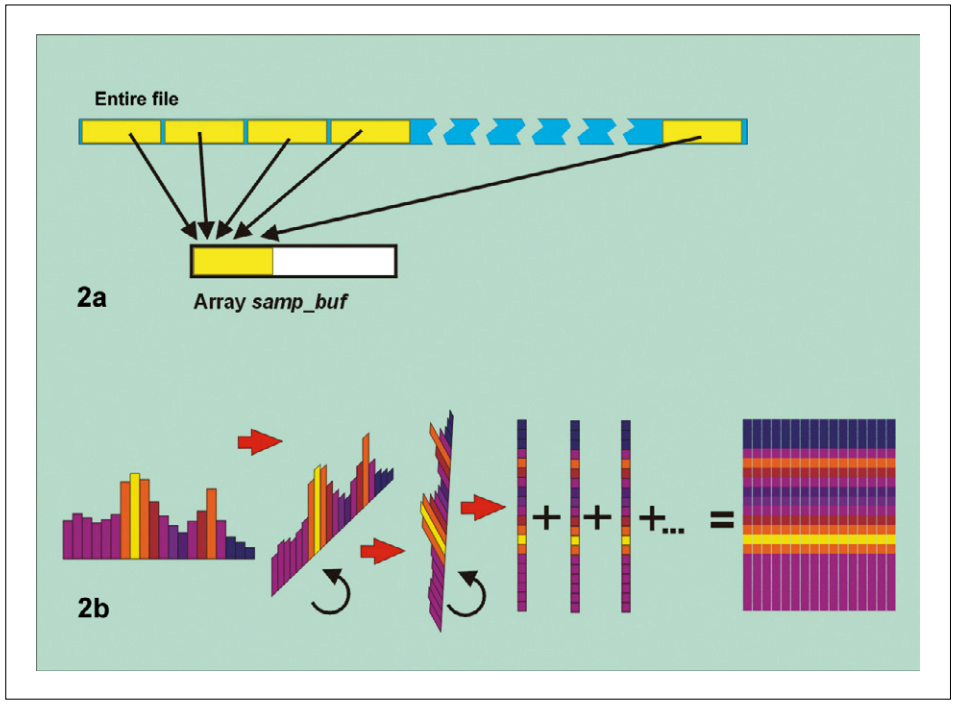


Figure 2. This 'FFT versus Time' graphic arises from calculating consecutive time intervals (Figure 2a). Representing the spectrum over time is produced by tilting and rotating (by 90 degrees), also aligning the display shown in Figure 1. We replace the height of the bars by color values that can be interpreted intuitively (Figure 2b).

Time vs. frequency

A problem arises with signals that vary over time: the more accurate the analysis required, the longer the time period we must investigate. Since we are indentifying the frequency amplitudes contained over an interval of time, the principles involved mean we are battling with a kind of 'Fourier uncertainty principle' in which the higher the temporal resolution of the analysis, the briefer the interval for investigation, which leads us back to coarser frequency resolution—and vice versa.

The FFT versus Time analysis adopted in the program shown here goes one step further. It generates not just the static spectrum of a defined signal segment but also captures consecutive time intervals (**Figure 2a**). In this way we can display the variation of the frequencies contained in a signal as a spectral progression even over lengthy periods of time. To display the results we require three dimensions: time, frequency and intensity. The X-axis corresponds generally to time and the Y-axis to frequency. The amplitude is represented by the color or brightness of a point in the time-frequency area (**Figure 2b**). This produces a series of color-coded spectra. A frequency-modulated sinewave tone (such as a police siren) in a display of this kind appears as bright, horizontal wavy line on a dark background.

Formulae and code

The Internet offers a vast amount of information on the subject of FFT, requiring unfortunately a solid mathematical background on account of the complex formulae involved. It is far simpler to go straight to the code for calculating an FFT. First of all then, here is the VB code (**Listing 1**) for a frequency analysis at a specific moment (time interval).A (fairly long) .wav file of a piece of music will serve as an example. We are looking for the amplitudes of the frequencies for a specific time segment Δt, beginning at the moment *t*. For this purpose we shall copy, let's say, 1,024 samples from the moment *t* into the Array *samp_buf* (see **Figure 3**). The code required for this is not included in the sample script above.

Listing 1.

```
Private Sub FFT()
  For n = 0 To 512
    a(n) = 0
    b(n) = 0
  Next n
  For n = 0 To 512
    For i = 1 To 1024
      a(n) = a(n) + samp_buf(i) * Math.Cos(2 * pi * n * i / 1024)
      b(n) = b(n) + samp_buf(i) * Math.Sin(2 * pi * n * i / 1024)
    Next i
  Next n
  For cnt = 0 To 512
    grafik(cnt) = Math.Sqrt((a(cnt) * a(cnt)) + (b(cnt) * b(cnt)))
  Next cnt
End Sub
```

To calculate the FFT we require both the buffer Arrays *a* and *b* (reset to zero before each analysis) and the Array *grafik* for the results, the content of which will be displayed later.

The two nested loops (with *n* and *i*) are responsible for calculating the FFT analysis. The outer loop subdivides the frequency range (sampling rate / 2) present in the file into 512 parts. Each value of *n* therefore indicates the intensity of the frequency in the time domain under examination on a scale from 1 to 512.

The Variable *i* in the inner loop determines the length of the time interval to be examined. An optimal result arises when $i = 2n$. The sample values in the inner loop are multiplied with Sine *and/or Cosine 2 Pi* and the loop Variables and also summed in the Arrays *a* and *b*. Only after a complete iteration of the inner loop is a single intensity value calculated in the frequency scale from 0 to 512.

In the final loop *cnt* the content of the Arrays *a* and *b* are squared, added and the root thereof is taken. This root is the final result and is then stored in the Array *graphic*. It contains the averaged amplitudes of the frequencies on a scale from 1 to 512 and represents the spectrum of the time interval from *t* to *t + 1,024* samples.

All Arrays are declared as 'single'. For graphical representation of *FFT vs. Time* the results contained in the Array *graphic* need to be converted into integer color values. You can find more about this in the commentary to the source code.

Saving calculation time

For the spectrum of a specified moment in time within a file such as shown in Figure 1 the code listed is completely adequate. Nevertheless several hundred (or even thousand) analyses of this kind need to be performed sequentially during an FFT versus Time analysis. To avoid the need to wait several minutes, the execution of the code shown above should ideally last no longer than a couple of milliseconds.

We can achieve this with one of many tricks. The expression included in the inner loop:

```
Math.Cos(2 * pi * n * i / 1024)
```

contains no values from the file under examination, only constants that had already been calculated previously and could then be recycled. Since every multiplication already inside a nested loop extends the calculation time drastically, these values are calculated in a Function of their own called *pre_FFT* before the actual FFT activity. The results of *pre_FFT* are then available in the Arrays *memory1* and *memory 2* (German '*speicher*' originally used for memory). We then need merely to read out their content and transfer this into the actual FFT loop. Calculating time is reduced significantly in this manner. Listing 2 shows the code for the Function *pre_FFT*.

The relevant lines in the Function *FFT* are altered as follows:

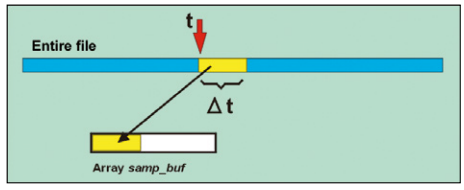


Figure 3. The graphic seen in Figure 1 emerges when an FFT analysis of a time interval Δt with a fixed number of samples is undertaken at a defined moment in time *t*.

```
a(n) = a(n) + samp_buf(i) *
   memory1(n, i)
b(n) = b(n) + samp_buf(i) *
   memory2(n, i)
```

There are two other means of accelerating the process, although we won't examine them in depth here. Anyone interested can find further references both in the program source code [1| and in the online help for the program.

Resampling

Let's assume that the sampling rate of a file amounts to 400 Hz. An FFT made with the code shown above produces the amplitudes of the frequencies from 1 to 200 Hz (on the basis of the sampling theorem) with a resolution of 1/512 of the entire range. We will also assume that the low frequencies between 1 and 10 Hz are of particular interest. Here, unfortunately, we cannot identify much about them, as the scale is squeezed together at the bottom edge. A Y-zoom feature would be helpful here — and nothing could be simpler than this. Within the inner loop

Listing 2. The code for the function `pre_FFT`

```
private sub pre_FFT
   For n = 0 To 512
      For i = 1 To 1024
         speicher1(n, i) = Math.Cos(2 * pi * n * i / 1024)
         speicher2(n, i) = Math.Sin(2 * pi * n * i / 1024)
      Next i
   Next n
End Sub
```

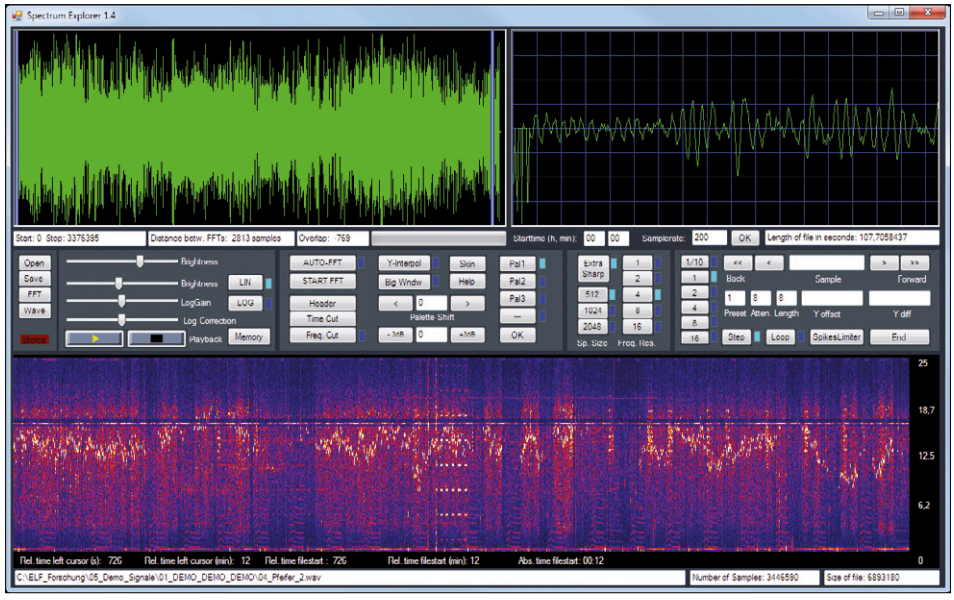


Figure 4. Programming interface showing the sample file *04_Pfeifer_2.wav*. A sinewave signal varying in frequency can be recognized immediately.

we still calculate 1,024 values but in the Array *samp_buf* the program does not advance by one sampling per step of calculation but by 2, 4, 8 or even 16 samplings, which leads to a 2, 4, 8 or even 16-fold elongation of the frequency range

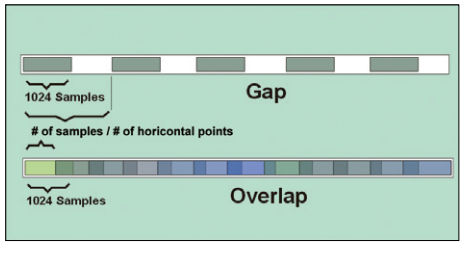


Figure 5. In order to assign a vertical colored line to each horizontal pixel of a display, we perform an analysis of an always fixed number of samplings at the calculated temporal positions. Gaps or overlaps will occur (dark patches in the lower columns), according to the file lengths involved.

in the Y direction. For this the Variable *i* in the line

```
memory1(n, i) = Math.Cos(2 * pi * n
   * i / 1024)
```

needs to be multiplied simply by 2, 4, 8 or 16, or else by a Variable (such as downsampling) that can be adjusted by means of the user interface. This is precisely what takes place in the program shown here:

```
memory1(n, i* downsampling) = Math.
   Cos(2 * pi * n * i / 1024)
```

Another loop

So far, from a file of the length *l*, only a brief segment of time *t* has been extracted, analyzed and processed for the result to be displayed in the form of amplitude over frequency.
The analysis described now involves the entire signal. For this reason we need to subdivide the total signal into *n* segments and carry out a separate analysis for each of these segments. The graphical result consists of vertical lines aligned with one another, with each line representing one of the calculated time intervals (Figures 2a and b).

Let's assume that the FFT graphic seen on the screen in the lower half of Figure **4** is exactly 1,000 pixels wide. To avoid aliasing effects, we therefore need to subdivide the file into 1,000 segments before analysis takes place. The FFT analysis described above has to be carried out for each of the segments. As we only ever calculate 1,024 samples, either gaps or overlaps will arise, according to the length of the files, as can be seen in **Figure 5**. As the number of segments in the current program is always equal, because of the constant width of the image window, the absolute file size will determine whether overlaps or gaps occur. As far as the display quality is concerned, it makes no real difference in the current program.

In the code this segmentation takes place using an additional loop; the code shown above is embedded in the loop, the Counter Variable of which proceeds through the values from 0 to the image width (in pixels). In the current program the value of this is 1,185 (maximum utilization of the standard monitor screen width). Care must also be taken that in the inner loop following each complete iteration while addressing *samp-buf* a leap forward occurs by the factor of *File length in samples / Width of window*. For example, a file with 6 million samples produces the value *Int (6,000,000 / 1,185) = 5,063*.

Windowing

If we extract a segment of specified width (FFT window) from a signal, the sine waves contained in the signal will also be cut out or cut off in haphazard fashion. These steeply sloped intersection edges correspond to powerful harmonics that would falsify the result and produce very noisy FFT versus Time graphics. To avoid such artifacts, the values to be analyzed need to be overlaid with an envelope curve for the time window. Numerous reliable varieties of these exist, such as those derived from sine and/or cosine functions. In this way the values

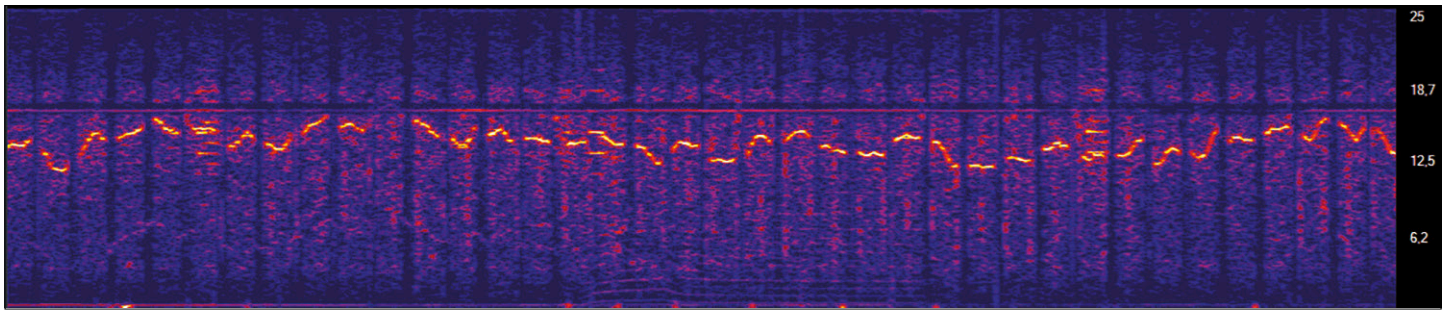


Figure 6. Zoom action achieved by operating the slider control in the upper left-hand overview window.

located close to the edges of the window are weighted less strongly and the artifacts mentioned reduced in the same process. In the program described here we make use of the so-called *Hanning Window* [3], which can be traced very easily in the source code.

Installation and operation

If you simply wish to use the program, without adding lines of your own to the source code, you can confine your download to the following three files: *Spectrum_Explorer.exe, AxInterop.WMPLib.dll* and *Interop.WMPLib.dll*. You will find these in the package at [1] (under *Bin ® Release*). The .exe file can be run with just a double click on any computer on which *Net-Framework 4* has been installed (tested on Windows 7 with 32 and 64 bits). As well as the .exe file you will require the two DLLs (in the same folder), which take care of the acoustic playback of files. VB programmers know how to deal with the downloaded code.

The program was written originally for analyzing the files of the ELF receiver showcased in the September 2014 issue of Elektor [2]. It is therefore intended exclusively for mono .wav files with 16-bit resolution.

A description of the use and function of this code in minute detail is beyond the scope of this article. For this reason the download package for this article also includes a PDF file that will acquaint you with the main features of the program. In addition the source code has extensive comments. Pressing the Help button on the user interface additionally calls up a replica of the user interface that explains each function by clicking on the relevant key. The **Quick Start Guide** in the text box should be fully adequate for your first explorations, however. To drill down to fine detail you will find the PDF instructions better. Make time to try out the other files too by pressing all the buttons…

(140246)

Web links

[1] www.elektor-magazine.com/140246
[2] www.elektor-magazine.com/140035
[3] http://en.wikipedia.org/wiki/Window_function#Hann_.28Hanning.29_window

Quick Start Guide

- Click on *Open*.
- Load the sample file *04_pfeifer_2.wav* (16 Bit, Mono, 200 Hz sampling rate).
- The lower part of the screen displays the resulting analysis, corresponding to that seen in Figure 5.
- By default the analysis is performed ‘rapidly but in low resolution’.
- Deactivate *Y-Interpol* [ation] and activate *Extra Sharp*.
- Enter the value *200* in the *Samplerate* input field and click on *OK* (otherwise the frequency scaling will be wrong).
- Clicking on *OK* and/or on *START FFT* initiates a new analysis.

The graphics now appear sharper than previously. By operating the slider control at upper left in the window you can make out that well-nigh the entire range can be displayed of a recording that can be up to nearly three hours long. The maximum frequency displayed amounts to 25 Hz (see right-hand scale). The very wavy line in the mid-frequency range shows that sinewave signals must be present here; these vary constantly and irregularly in frequency. This can be investigated further as follows:

- Use the mouse to move the right-hand slider in the upper-left window as far to the left as a finger’s width from the left-hand slider.
- Click once more on *Start FFT*.

After a small reduction in brightness moving the second *Brightness* slide control downwards you will see the new result: equally long sinewave tones of short duration (**Figure 6**), which are permanently changing frequency.

4Discovery: Resistive-touch Intelligent Display with Diablo16 Processor

The 4Discovery is an Intelligent Display Module with a 3.5-inch, 480x320 pixel TFT display with resistive touch and features the powerful DIABLO16 processor, which enables standalone functionality and is easily programmed using the 4D Workshop4 IDE. The 4Discovery is designed to be mounted to a standard light switch flush/mounting box, which allows quick and easy installation into a wall and makes it possible to create a multitude of home automation and control panel application straight out of the box.

The 4Discovery also features micro-SD memory storage, 16 MB of Flash Memory storage, Real Time Clock, 2-wire RS485 Interface which can act as Master or Slave with additional changeover wire, Optional WiFi, Optional Crypto Authentication security chip for secure transmissions, and a switch-mode power supply enabling a wide input voltage range.

Driving the display and peripherals is the DIABLO16 processor, which enables stand-alone functionality and is programmed using the 4D Systems Workshop4 IDE Software. The Workshop4 IDE enables graphic solutions to be constructed rapidly and with ease due to its design being solely for 4D Systems' graphics processors.The 4Discovery is available with or without WiFi as well as in various starter kit configurations. It is available for purchase directly from 4D Systems or at one of your local distributors. The software tools are available as a free download.

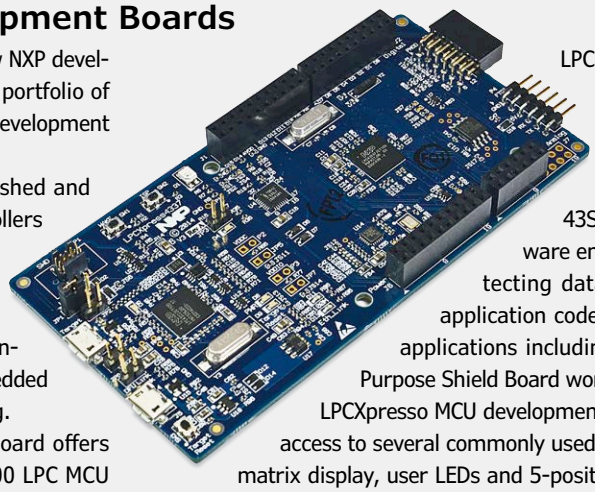
www.4dsystems.com.au (150273-3)

New NXP V3 LPC Xpresso Development Boards

Farnell element14 has announced the addition of three new NXP development boards in the LPC Xpresso family to its extensive portfolio of technology products, which includes the biggest range of development kits available globally from one single source.

These new LPC Xpresso V3 boards are part of an established and popular toolchain featuring the power of LPC microcontrollers based on ARMs Cortex-M cores. The boards are able to work with most existing LPC Xpresso accessories and Arduino compatible shields to enhance functional capabilities, offering more versatility for applications in motor control, power management, white goods, RFID readers, embedded audio applications, industrial automation, and e-metering.

LPC Xpresso 18S37, the ARM Cortex MCU development board offers evaluation and prototyping opportunities for the LPC1800 LPC MCU family; it features an ARM Cortex-M3 core running up to 180 MHz. Both LPC Xpresso 4337 and 43S37 ARM Cortex MCU development boards offer evaluation and prototyping opportunities for the LPC4300 LPC MCU family; it features dual ARM cores (M4F and M0+) running up to 208 MHz.



Both the 18S37 and 43S37 include an AES hardware encryption engine for protecting data communications and application code in required in multiple applications including IoT. The LPC General Purpose Shield Board works with V2, V3, and MAX LPCXpresso MCU development boards to provide easy access to several commonly used peripherals like LCD dot matrix display, user LEDs and 5-position joystick, inertial and temperature sensors and more.

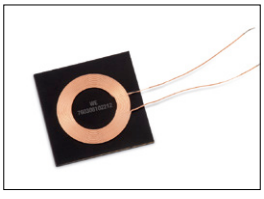
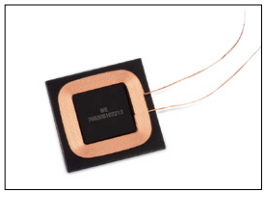
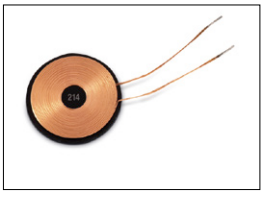
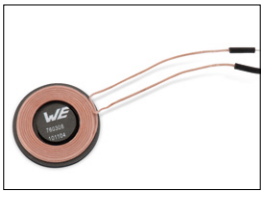
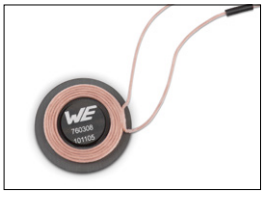
www.element14.com/community/docs/DOC-74693 (150273-4)

Wireless Power Supply for Wearables

Würth Elektronik eiSos GmbH & Co. KG, now offers wireless-power coils for wearables. Since devices such as fitness sensors and smart watches are generally encapsulated to protect them from the effects of the environment, but due to the variety of integrated sensory components (GPS, radio) have to be recharged frequently, the contactless transfer of energy without a charging plug is the call of the day.

By expanding its portfolio of proven wireless power-transfer products, Würth Elektronik eiSos is taking a leading position in the trend, and now offers two new transmitter and three receiver coils that are optimally suited for solutions for charging wearables and mobile devices. The coils are adapted to the already existing standard chip sets for the WPC Qi standard, making it easy to use already available designs simply by exchanging the coils for wearables.

The striking feature of the WE-WPCC 760308102212 and WE-WPCC 760308102213 receiver coils is their extreme thinness of just 0.64 mm, making them the component of choice for use in wearables with space for relatively large coil areas (29 x 29 mm), but with strong limitations regarding their height. In addition, the broad ferrite projection of the WE-WPCC 760308102212 unit offers excellent shielding properties – an important

aspect in applications that are sensitive to electromagnetic interference. The selection of the inductance value used in the WE-WPCC 760308102213 product makes it a dual-standard receiver. Fitted with the corresponding receiver chip, the coil is capable of responding to transmitters that are compatible with either the WPC Qi standard or the PMA standard.

To make charging stations that are ideally compatible with these receiver coils, Würth Elektronik eiSos also offers two new wireless power-charging transmitter coils, the WE-WPCC 760308101104 and WE-WPCC 760308101105 products. Both coils have an external diameter of just 21 mm, and use high-quality ferrite for shielding and stranded wire to keep the RDC to a minimum. All wireless power coils are directly available from Würth Elektronik with immediate effect; free-of-charge samples can be sent out on request.

www.we-online.de (150335-6)

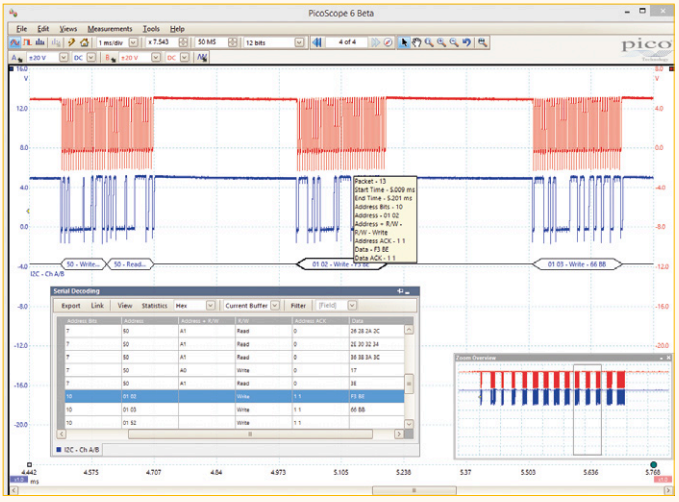
Free Picoscope Upgrade Supports 16 Serial Buses

In response to the proliferation of serial bus communications in electronic designs, the latest version of PicoScope now has decoding and analysis of 16 protocols included as standard. When used with the deep memory included in most PicoScope oscilloscopes, this new software release creates an extremely powerful debugging and troubleshooting tool at no extra cost to PicoScope owners.

PicoScope R6.11.4 beta can be downloaded free of charge and runs with any PicoScope.

Supported protocols, grouped by application, are as follows:

- Automotive: CAN, LIN, FlexRay, and SENT (Fast and Slow);
- Avionics: ARINC 429;
- Computer & networking: RS-232/UART, USB (LS, FS, HS), PS/2, Ethernet 100BASE-TX, Ethernet 10BASE-T;
- Embedded systems: I²C, I²S, SPI, 1-Wire;
- Lighting: DMX 512;
- Model railways: DCC.



The PicoScope serial decoding display can be viewed as a basic bus waveform correlated with the source analog or digital channels. This layout helps to troubleshoot data errors caused by noise spikes, faulty line drivers and mismatched impedances. Alternatively, the table view lists every packet or frame with timings, decoded data, status flags and optional voltage measurements. You can filter and search the table by any field, export data to a spreadsheet, and even define human-readable names for specified address data values by creating a 'link file'. PicoScope can decode multiple serial buses at once, with any mixture of protocols, limited only by the number of channels (maximum of 20 for MSO models). PicoScope R6.11.4 beta can be downloaded, free of charge, from www.picotech.com/downloads.

To read more details on Pico Technology's serial decoding protocols please visit: www.picotech.com/library/oscilloscopes/serial-bus-decoding-protocol-analysis

www.picotech.com (150335-1)

Microchip: CAN Flexible Data-Rate Transceiver

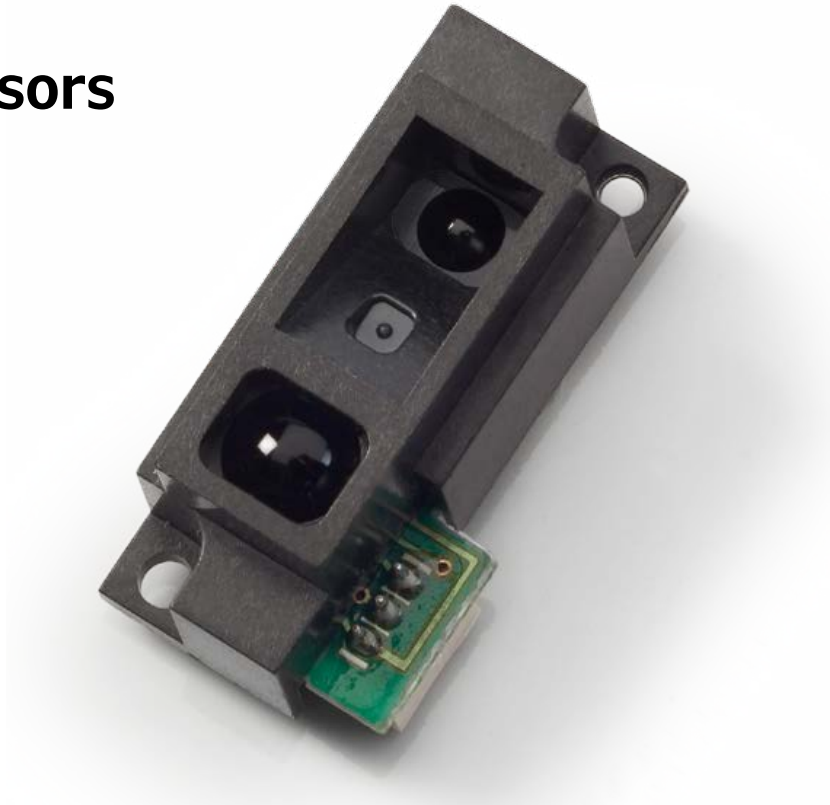
Microchip Technology Inc., announced a new family of CAN FD (Flexible Data-Rate) transceivers, the MCP2561/2FD. As an interface between a CAN (Controller Area Network) protocol controller and the physical two-wire CAN bus, these transceivers can serve both the CAN and CAN FD protocols. This product family not only helps automotive and industrial manufacturers with today's CAN communication needs, but also provides a path for the newer CAN FD networks that are increasingly in demand. With their robustness and industry-leading features, including data rates of up to 8 Mb/s, the MCP2561/2FD transceivers enable customers to implement and realize the benefits of CAN FD. These new transceivers have one of the industry's lowest standby current consumption (<5 µA typical), helping meet ECU low-power budget requirements. Additionally,

these devices support operation in the –40C to 150C temperature range, enabling usage in harsh environments. The new family of MCP2561/2FD CAN FD transceivers is available in 8-pin PDIP, SOIC and 3x3 mm DFN (leadless) packages, providing additional design flexibility for space-limited applications. The family also provides two options. The MCP2561FD comes in an 8-pin package and features a SPLIT pin. This SPLIT pin helps to stabilize the common mode in biased split-termination schemes. The MCP2562FD is available in an 8-pin package and features a Vio pin. This Vio pin can be tied to a secondary supply in order to internally level shift the digital I/Os for easy microcontroller interfacing. This is beneficial when a system is using a microcontroller at a Vdd less than 5V (for example, 1.8V or 3.3V), and eliminates the need for an external level translator, decreasing system cost and complexity. The MCP2561FD and MCP2562FD transceivers are both available now for sampling and volume production in 8-pin PDIP, SOIC and 3x3 mm DFN packages.

www.microchip.com/CAN-Transceivers-032315a (150335-3)

Man and Machine:
the Distance that Brings us Closer

Distance measuring sensors mirror human biology to enhance accuracy



By **Yuji Hamatake** (Manager, Sharp Devices Europe)

The devices that surround us are rapidly becoming more automated and intelligent. Enhanced functionality and sophistication requires these machines to perceive their surroundings with greater accuracy and reliability.

Distance measuring sensors are just one technology that is making a wide range of devices including everything from highly autonomous robots to paper towel dispensers smarter and more responsive to the needs of their human creators. Predictably, the technical means of establishing more natural man-machine interactions frequently reveals interesting parallels to human biology.

Stereopsis (also known as binocular parallax) is an important part of the human body's ability to perceive depth using both eyes. And it just so happens that stereopsis functions according to the very same principle that more sophisticated distance measuring sensors employ to yield precise results. It's known as triangulation, which lets us calculate the location of an unknown point in a triangle based on the distance between the two known points and knowledge of the respective angles.

Is there anybody out there?

Many of the photoelectric sensors in use today rely on a much simpler design that does not involve triangulation. These devices instead use an emitter of light (most often infrared) coupled with a photoelectric receiver capable of sensing the emitter's light. The emitter and light sensor can either be mounted together within a single component or be installed as two discrete units to form what is known as a through-beam arrangement. Retroreflective sensors send out a beam of light towards a reflector which returns the light back to the unit. In both cases, the receiver detects when the beam is interrupted by an object.

Diffuse reflectance proximity sensing devices rely on the reflectance (*syn*. reflectivity) of objects in their vicinity. When no object is present, the light emitted by the sensor is not returned to it. When an object enters the sensor's range, it reflects a portion of the emitted light back to the integrated photodetector, signaling the presence of an object.

Of the three types of photoelectric sensing described here, only the last (diffuse reflection) is suitable for small or portable devices which can't accommodate a two-piece retroreflective assembly. Devices such as touchless water taps in restrooms, for instance, are unable to accommodate separate emitter and receiver components.

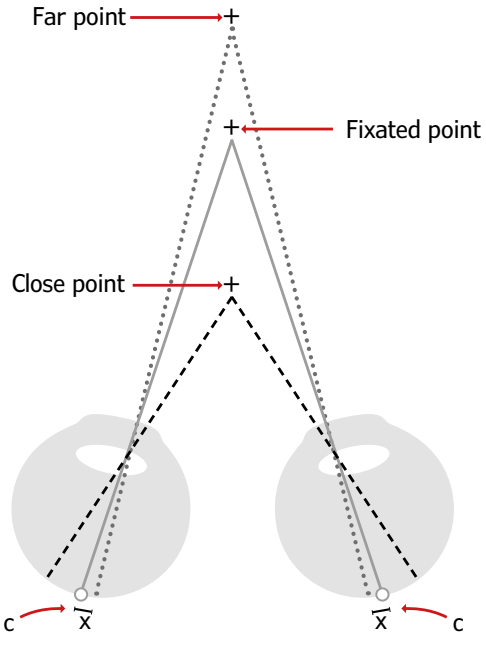
But estimating distance based on the amount of reflected light can be problematic. Diffuse reflective sensors happen to have an Achilles' heel: color. The color and texture of an object's surface can have a great impact on how much of the emitter's light is reflected back to it. And since simple sensors such as these are not able to detect and adjust for the color of objects, the accuracy of diffuse reflection sensors is limited.

An interesting angle

As an illuminated object approaches you, it's not only the intensity of the light reflected that changes. The angle does, too. Provided, of course, one measures that angle from a slightly different spot than where the emitter is located. Calculating the angle of incidence of the light hitting the offset receiver would yield the distance between the emitter and the

Triangulation: a natural approach to distance measurement

Stereopsis aids human depth perception by detecting distances of objects projected onto the retina.



In Sharp distance measuring sensors, a position sensitive detector (PSD) functions as the light receiving surface.

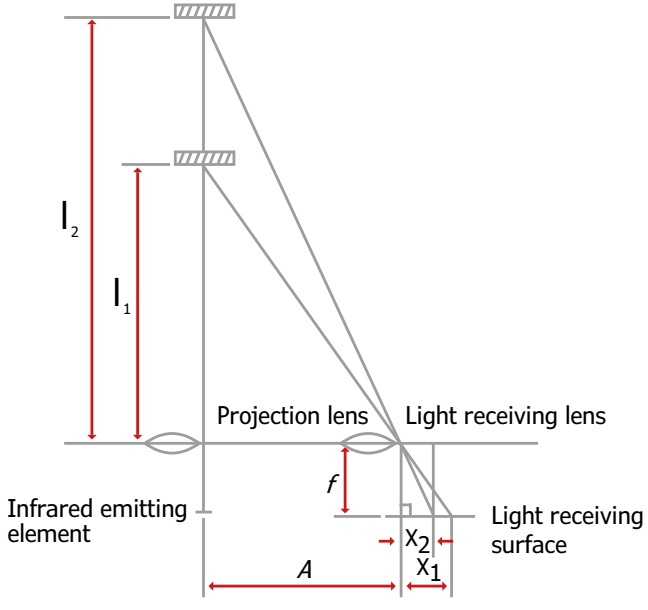


image: Sharp Devices Europe

receiver if we apply a little trigonometry, which is of course based on the theory of triangulation. But trigonometry is not exactly how sophisticated infrared distance sensors, such as those manufactured by Sharp, determine distance, although the underlying principles are the same.

Instead, two optical lenses are utilized, one covering the emitter and one covering the receiver. Behind the receiving lens, there is a position sensitive detector (PSD) that measures the intensity of the incoming IR radiation using a linear array of photodiodes. For the analog distance sensors from Sharp, the PSD's output varies in relation to the angle of incidence of the incoming light and generates the output voltage, which can then be used to calculate the distance of the object using the following formula:

$$\Delta x = x1 - x2 = \left(\frac{1}{\ell_1} - \frac{2}{\ell_2}\right) \cdot A \cdot f$$

Precisely the right sensor

The demands of individual applications often vary greatly. It's important to choose the right sensor according to a number of parameters including digital vs. analog output, minimum detection distance, maximum range, and not least of all cost and size. That's why so many engineers and product developers rely on Sharp distance measuring sensors — it's difficult to find a better selection of accurate components than Sharp's comprehensive lineup. With over a dozen different models covering ranges from as close as 1.5 cm and extending all the way out to 550 cm, Sharp's infrared distance measuring sensors are well-known across a number of different industries for their accuracy. Potential applications cover everything from document processing, sanitary equipment, and robotics to consumer electronics, gaming consoles, and more. With affordable and reliable choices like these available, it's clear that the hard-working and ubiquitous IR distance sensor is destined to bring a more human touch to an ever-increasing number of man-machine interactions.

(150238)

Welcome to Elektor Labs

Elektor Labs is the place where projects large, small, analog, digital, new and old skool are sketched, built, discussed, debugged and fine-tuned for replication by you.

Our offer: Become Famous

Most engineers and budding authors we come across are just too modest. If they do not see the attraction and beauty of a design idea scribbled on a coaster and worked out later at home, others may, and should. Let Elektor Labs help you hone your design to perfection, let the editors assist with text & graphics, and reap the rewards by seeing your name in print in Elektor magazine's celebrated LABS section. Sure, we are happy to negotiate payment, but the actual remuneration will be fame & glory in electronics land, and your name added to the long list of extremely successful e-authors. Our get-u-famous formula also applies to book authors, bloggers and video makers. Students and youngsters: being in publication is a current boost like no other in getting a job!

Our Experts and Designers

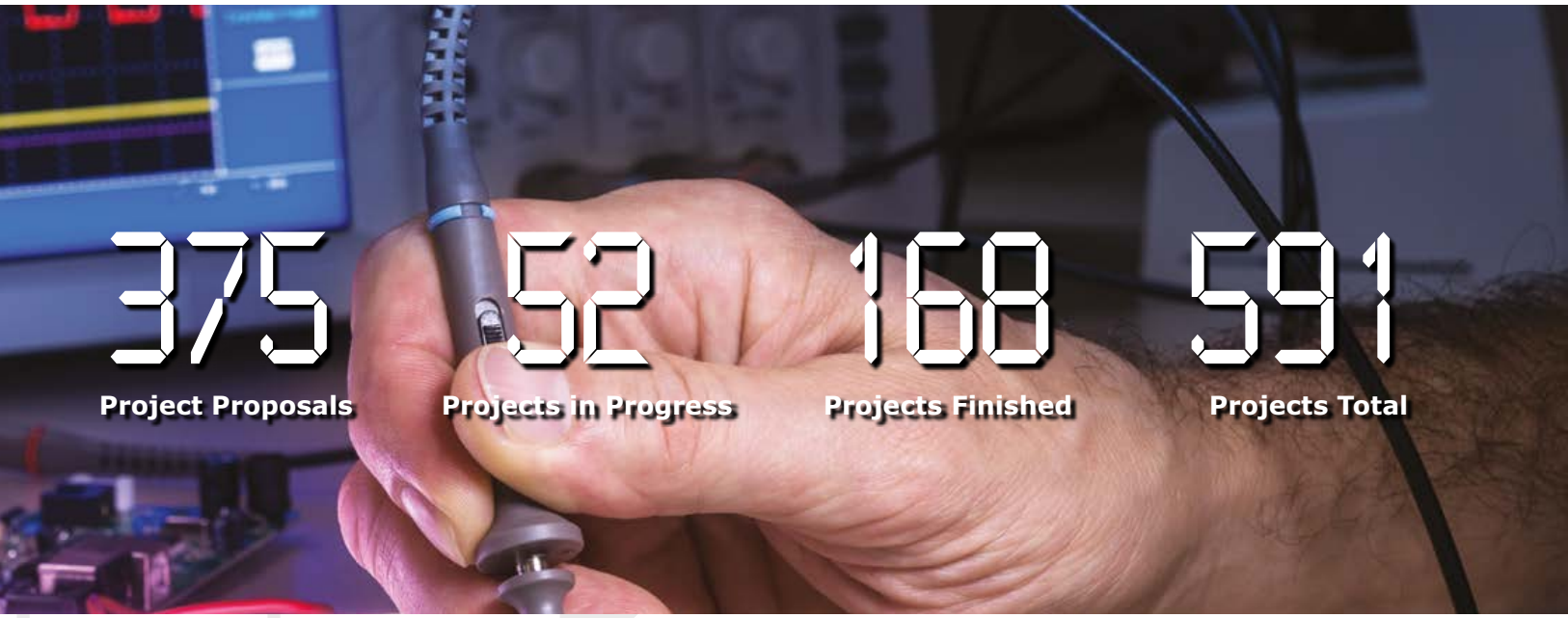
Besides experienced support staff and BSc, MSc qualified engineers with a total working life in electronics of about 200 years, the Lab has access to Elektor's vast network of experts for consultation, critical advice, and assistance with specialized assignments.

Our Facilities

We are sumptuously housed in three spacious rooms at Elektor House where we try unsuccessfully to keep our solder jobs and prototypes off our computer desks. We have water, 218 volts electricity and coffee nearby. PCB milling, prototype assembly, SMD reworking, audio testing, pizza cooking, and mechanical work are deferred to converted cellars in the basement.

Our Standards

All projects and products going through the Labs pipeline are produced to high engineering standards. In practice, prototypes of projects labeled LABS in the magazine must be demonstrated to work to specification on certified, calibrated test equipment available locally. BOMs and schematics must match perfectly. Kits are sampled for completeness. We are ROHS compatible, lead free and comply with electrical safety standards applicable in our location. If engineering errors are found these will be put up for public notification.



Our Products

Our products are visible in Elektor magazine, as well as on the Elektor.Labs and Elektor Store websites. The range includes text write-ups for editors, prototype photography, PCBs including SMD-prestuffed, PCB files, project software, programmed devices, semi-kits, tools, modules, videos, and service desk information.

Our Webinars

The more talkative of our Lab engineers do not stop at testing prototypes, they happily share technology related problems, insights, get-u-going information, and design skills on live camera at Elektot.tv. Labs' webinars are free to attend and extremely interactive. They are announced in Elektor.POST, and webcast live from Elektor House in Holland. Do plug in!

Our History

The origins of Elektor Labs go back to the early 1970s when soldering and writing was a one-man, single-desk job. Ove the years Labs staff have not only witnessed the arrival of the transistor, the IC, the microcontroller, and the SMD, but actually jumped on these parts as soon as they were out of the professional-only woods. Once special branches, Audio Labs, Micro Labs, PCB Labs, and Mechanical have converged back again into a single activity.

elektor labs

Sharing Electronics Projects

Search

Home Proposals ➡ In Progress ➡ Finished

Log in

Upload your own projects!

On our very own Elektor-Labs website, you can share and showcase your ideas and project proposals to thousands of other electronic engineers. The site also allows you to follow other specialists' activities, supply comment, and so push the projects forward. Here's the best part, the top projects are eligible for processing in our test lab, in preparation for publication in Elektor magazine.

Who's this for?

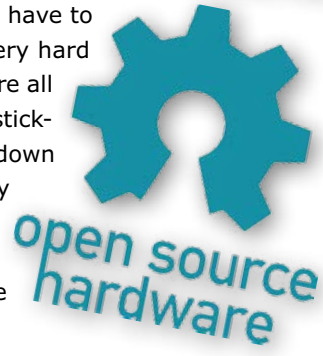
Although only members can log on to our Elektor.Labs website to publish projects and contributions, anyone can view along with the other projects. If you'd like to see your project published in Elektor magazine, which is published in four languages and read by tens of thousands of electronics specialists all over the world, then become a Green of Gold member (**www.elektor.com/member**) or log in as an existing member of our Elektor community!

Welcome to the DESIGN section

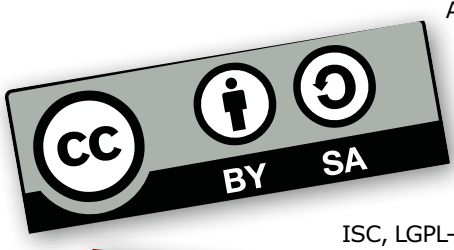
By **Clemens Valens,** Elektor Labs

Licence to… gosh, whom/what exactly?

One of the most convoluted subjects today in electronic design is licensing. If your design is closed as most commercial products are, it is assumed to be protected by international trade and copyright laws. These appear pretty complicated to understand and apply by a layman, but there is lots of jurisprudence detailing what can and what can't be done. However, when you want to release a design as open source, be it hardware, software or both, things get really complicated. The problems emanate from very nature of the open source movement itself. A typical open source license allows the licensee to adapt the licensed work, which often results in numerous 'derived' designs, all slightly different. The same effect is seen to encroach upon open source licenses. There are now so many different open source licenses out on there on the web that it is hard if not impossible to find the one that's just right for you. The licences being legal documents that have to stand up in court — globally if possible — they are very hard to understand. What's more, they keep evolving. Where all you wanted was donate your design to the world by sticking some open source license on it, you got bogged down in a swamp of legal gibberish. And you can't get away with something like "this is free, do whatever you want with it, no responsibility taken" because that will be considered too light hearted in a lawsuit after someone did something terrible with your work.



A Useless & Incomplete List of Licences (in alphabetical order)



AAL, AFL-3.0, AGPL-3.0, Apache-2.0, APL-1.0, APSL-2.0, Artistic-2.0, BSD-2-Clause, BSD-3-Clause, BSL-1.0, CATOSL-1.1, CC BY, CC BY-NC, CC BY-NC-ND, CC BY-ND, CC BY-SA, CC-BY-NC-SA, CDDL-1.0, CECILL-2.1, CERN OHL, CNRI-Python, CPAL-1.0, CUA-OPL-1.0, ECL-2.0, EFL-2.0, Entessa, EPL-1.0, EUDatagrid, EUPL-1.1, Fair, Frameworx-1.0, GPL-2.0, GPL-3.0, HPND, IPA, IPL-1.0, ISC, LGPL-2.1, LGPL-3.0, LPL-1.02, LPPL-1.3c, MirOS, MIT, Motosoto, MPL-2.0, MS-PL, MS-RL, Multics, NASA-1.3, Naumen, NCSA, NGPL, Nokia, POSL-3.0, NTP, OCLC-2.0, OFL-1.1, OGTSL, OSL-3.0, PHP-3.0, PostgreSQL, Python-2.0, QPL-1.0, RPL-1.5, RPSL-1.0, RSCPL, SimPL-2.0, Sleepycat, SPL-1.0, TAPR, UPL, VSL-1.0, W3C, Watcom-1.0, WXwindows, Xnet, Zlib, ZPL-2.0. ⏮

(150332-I)

Learn-'n-Go Infrared Remote Controlled Dimmer

for lamps and heaters

Original design: **Juan Canton** (México)
Post Engineering: **Ton Giesberts** (Elektor Labs) Editing: **Jan Buiting**

This IR remote controlled dimmer works with a triac and in its standard configuration is suitable for light electric heaters and incandescent lamps up to about 1000 watts. The PIC microcontroller used in the project can learn control codes from almost any remote control. Here's power dimming from the comfy chair.

The initial requirements for the project, a "multipurpose remote control receiver for AC line powered lamps" were formulated by the author as follows:

1. Operation off 50 Hz or 60 Hz power grids;
2. Operation off 117 or 230 VAC power grids;
3. Operation with only one button;
4. Suitable for controlling the brightness level of any incandescent or halogen lamp, or the heat produced by a small electric furnace or other heater;
5. An ability to learn up to eight control codes of a common infrared remote control.

With the above in mind a circuit was designed by the author, posted on elektor-labs.com and subsequently perfected by Elektor Labs for the purpose of this publication. The transition from Juan's original circuit proposition to the final project published here is well documented by Ton and may be read at the Elektor Labs website [1]. Basically the circuit was changed from an in-line power dimmer to one with 115 or 230 V AC-IN, and dimmed AC-OUT. The best thing about the circuit, its learning ability, was fully retained though.

Towards a circuit

The result is pictured in the schematic, **Figure 1**, which is typical of today's black box-ish microcontroller-based circuitry: little to see in the way of external components, and *mucho PIC* handling many things in software. To the in-crowd, the triac and the IR decoder are telltale of "something to do with AC-line, ampères and remote control". To better understand the operation of the circuit, it can be thought of as divided in six sections, each of which is discussed below.

AC-line power supply

What's needed? In a nutshell: V_{cc} = 5 volts; I_{load} = 4 mA or so net, "stolen" from the domestic 115/230 VAC line with acceptably low dissipation. Cost and space considerations were found to rule out a dedicated power transformer with its usual complement of rectifier and reservoir capacitor(s).
The next option traditionally is using a dropper resistance to lower the voltage from 230 (or 115) volts AC down to 5 volts DC, in this case with a minimum load current you'd *think* is 4 mA. Alas this has to be doubled at least, due to half-wave rectification only. As an example, the following calculation was done for a dropper R (based on 230 VAC line);

$R = (V_{LINE} - V_{cc}) / 2\ I_{load}$
$R = 225 / 0.008$
$R = 28{,}125\ \Omega$

which sadly dissipates:
$P = I^2 R$
$P = 1.8$ W

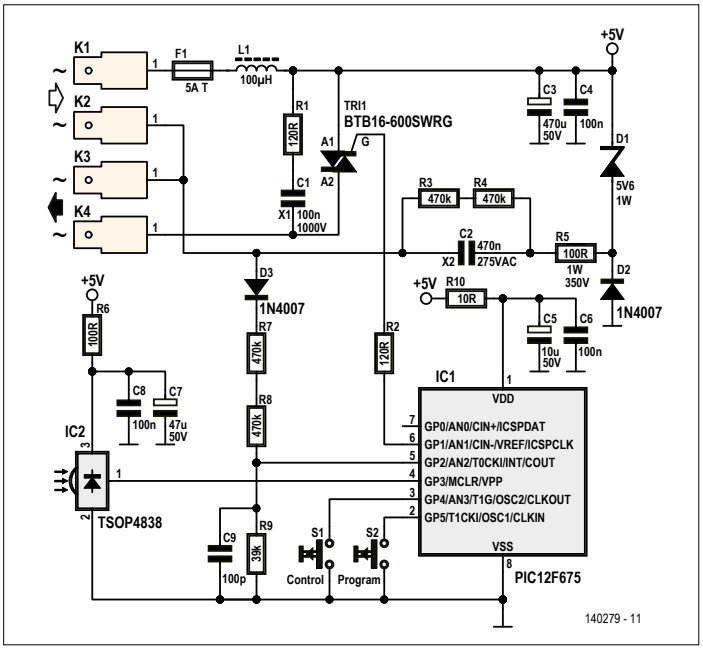


Figure 1. Schematic of the learning dimmer for AC line powered incandescent lamps and (light) electric heaters.

or about 0.9 watts at 115 VAC using a 14 kΩ resistance. This result seems just about tolerable, but in case the circuit is connected in series with the load, when the lamp lights at it brightest, not enough voltage may be left to power the circuit reliably, particularly with 115 VAC grids.

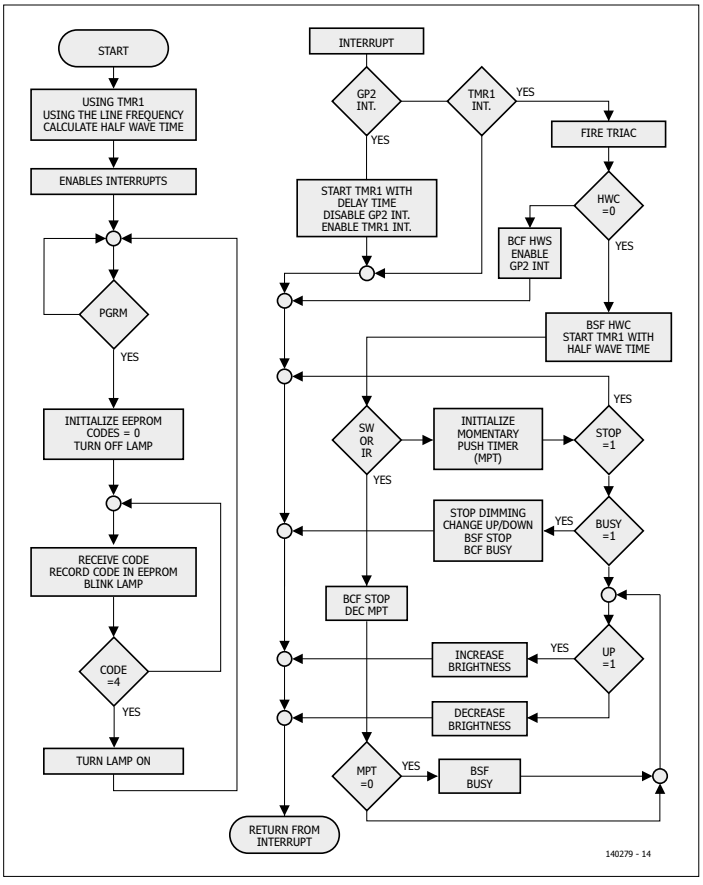


Figure 2. Software flow diagram. The actual program was written in PIC assembly code.

The third option is a combination of a resistance (*R*), a capacitance (*C*), and a clamping device (i.e. rectifier). In the R5-C2-D2 constellation we have here, at 50 Hz and 8 mA current draw, the 0.47 µF capacitor used (C2) is adequate for reliable powering of the triac control circuitry, also considering that the trigger action lasts only microseconds.

Besides the all-important dropper capacitor C2, the power supply circuit comprises R3+R4, D2, D1, C4 and C3, and its operation is truly simple. The reactance X_c of dropper C2 together with the resistance of R5 and the clamping/rectifying action of D2 effectively limits the current applied to 5.6-V zener diode D1. C3 removes most of the mains-borne voltage variations, and finally C4 suppresses any high frequencies and noise.

The AC line voltages and the appliance are connected to connector pairs K1-K2 and K3-K4 respectively. The incoming line voltage is protected with 5-amp (115 V; 10-amp) fuse F1 and any switching noise due to the triacs gets suppressed by choke L1.

Zero-crossing detector

Looking at D3, R7+R8, and R9-C9, we have a half-wave rectifier rather than a zero-crossing detector. Consequently only one crossing is detected and the microcontroller software calculates the trigger instance to go with the second half cycle.

Trigger circuit

Both the A1 pin of the triac and the microcontroller supply rail (+5 V) are tied to the AC line voltage. To trigger the triac, its gate G is driven to ground via current limiting resistor R2. This combination takes advantage of the fact that the microcontroller has higher current sink capacity than source capacity, and triggers the triac with “negative” current into its gate, which is where most of these elements need least current to operate.

Snubber network

This is composed of C1 (note: X1-class cap here) and R1, and its function is to avoid false turn-on of the triac. Triac type BTB16-600SWRG is a ‘digital’ one (suffix SWRG) with a relatively low trigger current of 10 mA (min.). On the down side it’s not snubberless, hence the need for RC network R1/C1.

Infrared (IR) receiver module

Receiver module IC2 decodes the signal of any common infrared remote control transmitter. We tested a number of RC5 remotes. When the microcontroller is in programming routine (discussed further on) it receives the code and stores it in its on-chip EEPROM. When in normal use the same code is recognized, the micro responds as if the local Control switch (S1) is pressed. This allows you to operate the dimmer from a distance.

Filters

The circuit has a filter to avoid radiation and disturbances on the AC grid owing to the triac operation. This filter inductor L1 is either an off the shelf Murata type, or can be home made from 10 turns of 22 AWG (0.6 mm diam.) enameled copper wire on a 25-mm toroid core. Without experience in winging a suppressor choke for use at AC line, we strongly suggest going for the Murata device.

Considering that the circuit is powered with a voltage that may have severe variations, the supply voltages to the microcon-

troller and the infrared receiver need to be filtered. For the microcontroller this is done by R10, C5 and C6; for the IR module, by R6, C7, and C8.
That concludes the description of the design as far as the hardware goes.

Software

One way of getting to understand the software executed *magically* & *invisibly* by the PIC micro is to look at the flow diagram in **Figure 2**. When the supply is applied, the following happens. Note that for all instances of "brightness" you can also read "heater power" if an electric furnace is connected as a load.

1. The microcontroller calculates the AC line frequency (50 Hz or 60 Hz — Elektor is read all over the world) and completely turns off the lamp. It waits for Control switch S1 to be pressed.
2. If Control is pressed and released immediately, the brightness will increase slowly until it reaches the maximum level.
3. If Control is pressed and remains pressed, brightness will increase until the switch is released.
4. If Control is pressed and released, and the lamp lights fully or partially, the dimmer will decrease brightness to zero.
5. If Control is pressed and remains pressed, and the lights fully or partially, the dimmer will decrease brightness until switch released.

Fitting with an Elektor publication aimed at enthusiasts like you, the source code for the microcontroller is available for free downloading from the web page created for the project [2], for studying and of course home burning of a controller (**Figure 3**). The software was written in **assembly language** — a piece of the program is shown in **Listing 1**. Archive file 140279-11.zip is for 4 codes, while no. 140279-12.zip is for 8 codes (see below).
Further advice on electrical safety aspects and component choice may be viewed at the Elektor Labs page [1].

IR response code programming

It is very simple to program the dimmer. To start programming it is advisable but not necessary that the lamp lights at max. brightness. To program, follow the next steps:

1. Hold the Program switch pressed for 3 seconds. This will turn the lamp (heater) on and off twice, showing that the

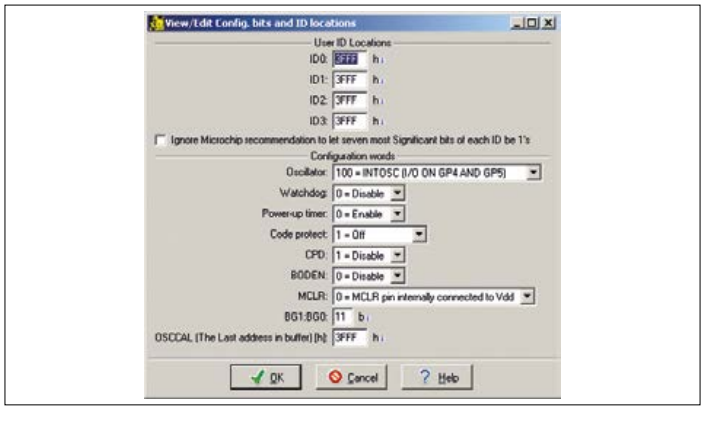


Figure 3. PIC fuse settings for all you embedded moonshiners out there.

Listing 1. Program extract (PIC assembly language)

```
;################################################
GET_PULSE  BTFSS REC_SW
      GOTO  RUN
      CLRF  TMR0
GET_P0   BTFSC TMR0,5
      GOTO  GET_P1
      BTFSS IR_SIGN  ;WAIT FOR HIGH ON "IR_SIGN" PIN
      GOTO  GET_P0
GET_P1   CLRF  TMR0
GET_P2   BTFSC TMR0,5
      GOTO  GET_P3
      BTFSC IR_SIGN  ;WAIT FOR LOW ON "IR_SIGN" PIN
      GOTO  GET_P2
GET_P3   MOVFW TMR0   ;GET PULSE TIME
      MOVWF INDF
      MOVLW B'00001111'
      ANDWF INDF
      CLRF  TMR0
GET_P4   BTFSC TMR0,5
      GOTO  MS3A
      BTFSS IR_SIGN  ;WAIT FOR HIGH ON "IR_SIGN" PIN
      GOTO  GET_P4
MS3A   CLRF  TMR0
MS4   BTFSC TMR0,5
      GOTO  MS4A
      BTFSC IR_SIGN  ;WAIT FOR LOW ON "IR_SIGN" PIN
      GOTO  MS4
MS4A   MOVLW B'00001111'  ;GET PULSE TIME
      ANDWF TMR0
      SWAPF TMR0,W
      ADDWF INDF
      RETURN
;################################################
END_IR   MOVLW IR_TIMEOUT
      MOVWF WAIT_IR
      MOVLW .158
      MOVWF TMR0
      BCF INTCON,T0IF
      BTFSS IR_SIGN
      GOTO  END_IR
      BTFSS INTCON,T0IF
      GOTO  $ -3
      DECFSZ  WAIT_IR
      GOTO  $ -8
      RETURN
;################################################
```

Web Links

[1] Project history: www.elektor-labs.com/project/ir-remote-control-learning-dimmer-or-heat-control-140279-i.14109.html

[2] Project software: www.elektor-magazine.com/140279

circuit has entered the programming routine.
2. Using your remote control transmitter at a distance of 3 feet, press the button you want to store. When the dimmer receives and stores the code, it will turn the lamp on and off as a confirmation that the code has been programmed.
3. Repeat step 2 seven more times. Once the eight codes have been stored the lamp will turn on and stay on to show the end of the programming.

Software version **140279-12** supports 8 codes, which may be preferable over 4 in case of RC5 remote controls, which do not necessary send the same code twice on pressing the same button. Ready-programmed PIC no. **140279-41** from the Elektor Store supports 8 codes.

The IR code can be the same for the four (eight) available positions. Also, different remote control transmitters can be used (of the same standard though). Once the dimmer is pro-

Component List

Resistors
R1 = 120Ω 5%, 5W
R2 = 120Ω 5%, 0.25W, 250V
R3,R4,R7,R8 = 470kΩ 5%, 0.25W, 250V
R5 = 100Ω 5%, 1W, 350V
R6 = 100Ω 5%, 0.25W, 250V
R9 = 39kΩ 5%, 0.25W, 250V
R10 = 10Ω 5%, 0.25W, 250V

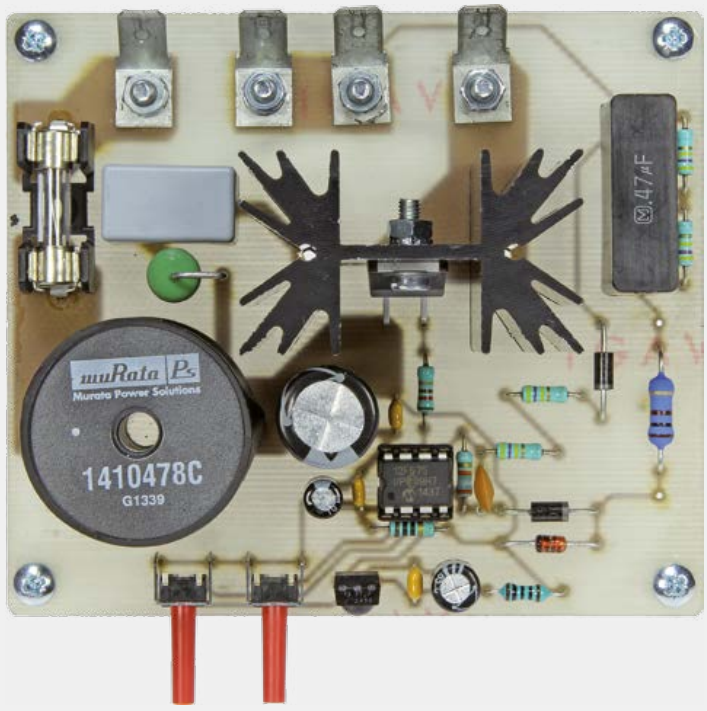
Capacitors
C1 = 100nF 20%, 1000V, X1 class, 15mm lead spacing
C2 = 470nF 20%, 275 VAC, X2 class, 22.5mm lead spacing
C3 = 470µF 20%, 50V, diameter 13mm, 5mm lead spacing
C4,C6,C8 = 100nF 10%, 50V, ceramic X7R, 0.2" (5.08mm) mm lead spacing
C5 = 10µF 20%, 50V, diameter 5mm, 2 mm lead spacing
C7 = 47µF 20%, 50V, diameter 6.3mm, 0.1" (2.54 mm) lead spacing
C9 = 100pF 5%, 100V, ceramic C0G/NP0, 0.2" (5.08mm) lead spacing

Inductors
L1 = 100µH 10%, 7.8A, R(DC) = 0.04Ω, Murata Power Solutions type 1410478C

Semiconductors
D1 = 1N4734A, 5V6 zener diode, 1W
D2,D3 = 1N4007, 1000V, 1A
TRI1 = BTB16-600SWRG
IC1 = PIC12F675, programmed, Elektor Store # 140279-41
IC2 = TSOP4838

Miscellaneous
K1-K4 = 0.25" (6.35mm) terminal, Faston, screw type, 3.3mm hole
S1,S2 = switch, tactile, 6x6 mm, PCB edge, Alps SKHHNKA010 (Mouser # 688-SKHHNK)
HS1 = heatsink, PCB mount, Fischer type SK 129 38,1 STS, 6.5 °C/W. For 1250 watts max.
F1 = fuse holder, 20x5 mm, 500V, 10A
F1 = 5A fuse, slow-blow, (T), 5x20mm
PCB 140279-1 v2.0
Case: Hammond type 1591ETBU

Not on PCB: electrical outlet socket, DIN 49440, 16A, 250V, IP20, panel mount (Schuko; German style); RS Components part no. 824-5627. UK and US socket types: see text.



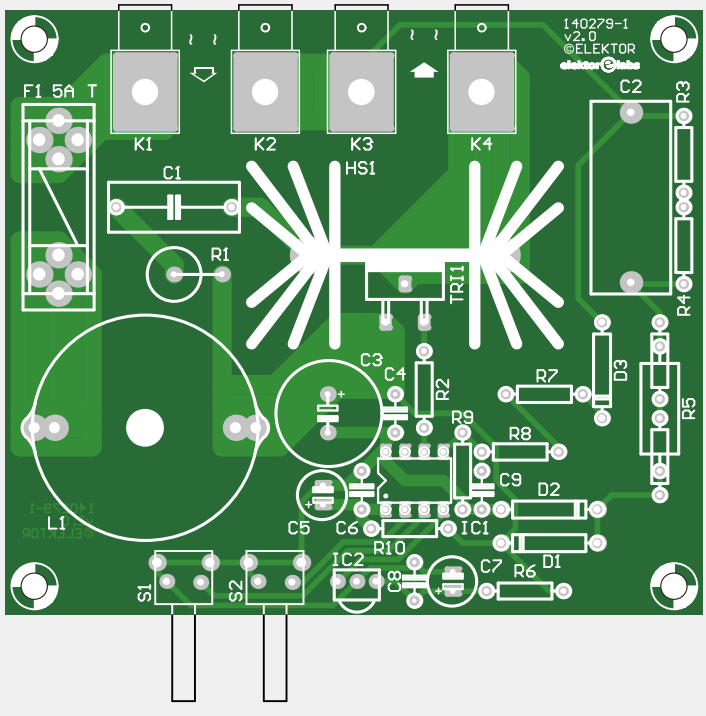


Figure 4. The single-sided printed circuit board for the dimmer was designed with all due attention given to electrical safety.

grammed, the programmed buttons to control will have the same function as the control switch.

Construction

The circuit is connected to, and operates off, the AC power line which requires great attention to be paid to electrical safety. This was implemented by Elektor Labs doing the PCB design (**Figure 4**) and building the prototype in an isolated (but transparent!) case, drilled to accept a certified AC power inlet and outlet. The short pieces of AC wiring used are 2.5 mm² (13 AWG approx.) in view of the maximum load power of about 1000 watts (note 5 amp fuse @ 230 V). The triac is specified 16 amps max. so output powers up to 2000 watts are within reach provided you use a larger heatsink than the one pictured here and indicated in the parts list (still observing electrical safety).

The little Control and Program pushbuttons are actuated with plastic pins protruding from holes in the short side panel, again respecting minimum distance requirements for 230 VAC.

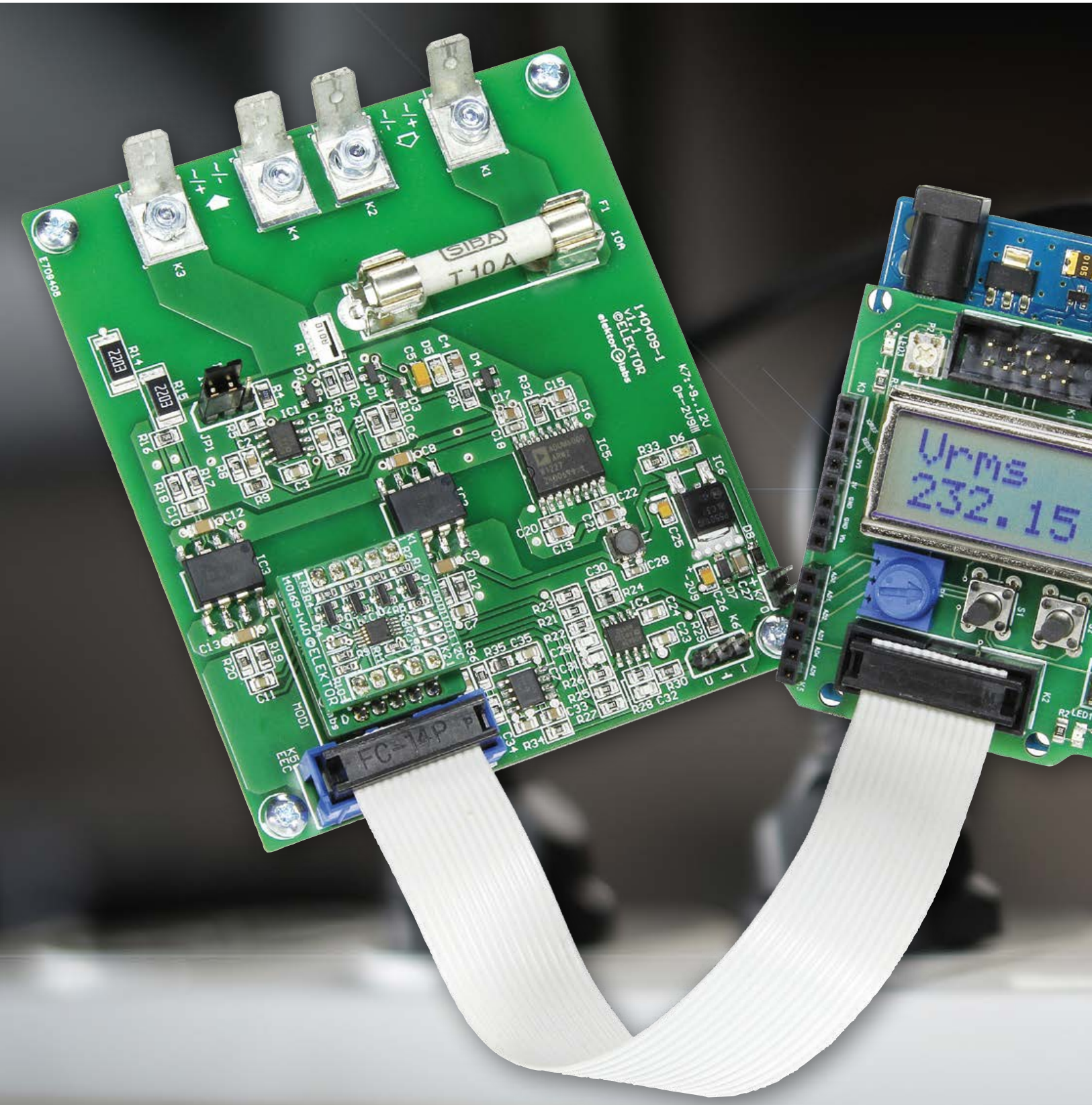
Protective earth (PE) is not indicated in the schematic, meaning it is not present on the circuit board. It **must** be wired though straight from the IEC appliance inlet to the outlet socket.

To avoid having to drill a hole for the optical receiver in the enclosure and so create a potential safety issue, we used a type 1591ETBU blue transparent polycarbonate housing from Hammond Manufacturing which is not too expensive (less than €14 at Newark/Farnell). The IR signal is weakened somewhat by the case but we still achieved a distance of 5 m (15 ft) before the circuit quit responding. Nylon M3 screws are used to secure the board to the case bottom at a height of 4 mm minimum. This distance may be secured with the help of two M3 nuts on each screw to create a standoff. The AC inlet is an IEC appliance socket, the AC outlet used will depend on the country you live in. At Elektor Labs, a light blue, Schuko-type earthed AC power socket was used that fits the size and matches the color of the case beautifully. Readers in the UK should go for RS Components part # 824-5646, those in the US, for # 824-5655 — both come with a cover lid. The total cost of the Hammond box and the AC power socket is not high given their essential role in electrical safety of the project, not forgetting an attractive yet professional look. Sticklers for splash proofing: the Schuko outlet is also available with a lid, it's RS part # 824-5630.

In case a light flickering is visible when fully dimmed, try lowering R7 and R8 to 220 kΩ. Also upping the value of C9 can help (<10 nF). A larger value will cause a larger shift in the triggering point of the triac. ⏮

(140279)

Finally, serious words of caution: Do not operate or work on the finished circuit at any time when it is not fully enclosed and secured in its isolating case. All requirements, recommendations and legislation relevant to electrical safety in your country, state or area apply.



In the December 2014 issue of Elektor we described a breakout board with a 4-channel, 16-bit A/D converter, dubbed the ADS1115-eBoB [1]. This IC was previously used in the ELF receiver [2] and the 16-bit ADC module [3]. With the advent of a convenient breakout board holding this tiny IC, we thought it would be fun to come up with some other applications. The IC has a fairly low sample rate of just 860 Hz, so it is only suitable for use in very low-frequency measurement circuits. With that thought in mind, an AC power meter is an obvious choice. Although you can buy ready-made power meters these days at fairly low cost, they are closed systems — and that goes against the grain with true electronics enthusiasts. In addition, they do not have an output for connection to an oscilloscope or multimeter.

The project presented here consists of a measurement board holding the ADS1115-eBoB together with a small filter board.

AC/DC Power Meter

With high accuracy and a wide measuring range

Hardware: **Ton Giesberts** (Elektor Labs) Software: **Luc Lemmens** (Elektor Labs)

When making voltage, current and power measurements on AC line-powered devices, you need an instrument with good galvanic isolation between the primary and secondary sides. The instrument described here, based on an Arduino Uno with the Elektor Prototyping Shield, provides Class II insulation and displays several quantities directly on an LCD. It also has separate measurement signal outputs for connection to a multimeter or an oscilloscope.

The circuitry on the measurement board provides good isolation between the power connections (AC or DC) on the primary side and the measurement signal outputs on the secondary side. The signals on the secondary side are digitized by the ADS1115, and the digital data is fed to an Arduino Uno with an Elektor Prototyping Shield (no. 140009-91), which processes the data and shows various readings on the display. The displayed quantities for AC are effective power, voltage and current, as well reactive power and power factor. If you connect a DC circuit to the input, the meter shows the power, voltage and current.

Galvanic isolation

Power meters are mainly used to make measurements on line-powered equipment, and it is very important to have good galvanic isolation between the measurement circuitry on the one side and the data and measurement connector outputs on the other side. This is often imple-

Features

- Three measuring ranges: 10 A, 1.77 A and 0.177 A
- Maximum input voltage 500 V_{AC}/300 V_{DC}
- Minimum measurable power 0.1 W
- Separate isolated measurement outputs for voltage and current
- Class II insulation between primary and secondary sides
- Measurement data processing by Arduino
- Measured values shown on a two-line LCD
- Displays voltage, current, effective power, reactive power and power factor for AC
- Displays power, voltage and current for DC

mented with optocouplers and opamps, but that approach results in a fairly high component count. Another option is to use a special isolation amplifier. Our search for a suitable type led us to the AMC1100 from Texas Instruments. This fully differential isolation amplifier has a minimum bandwidth of 60 kHz and a rated galvanic isolation of 4250 V_{peak}, with a rated working voltage of 1200 V_{peak}. With this isolation amplifier and a good PCB layout, you can obtain Class II insulation, which makes the circuit very safe. The IC is housed in a gullwing-8 SMD package, which looks like an 8-pin DIP package with the pins bent out to the side. The circuit also needs two completely separate power supplies — one for the input side and the other for the output side. In theory you could use standard power modules with unregulated output voltage for this purpose, but most of them are not specified for Class II and you would need extra components, including separate voltage regulators. A better solution is to us an IC specifically designed for this purpose. Here we chose the ADuM6000 from Analog Devices, which is an isolated DC/DC converter with a regulated

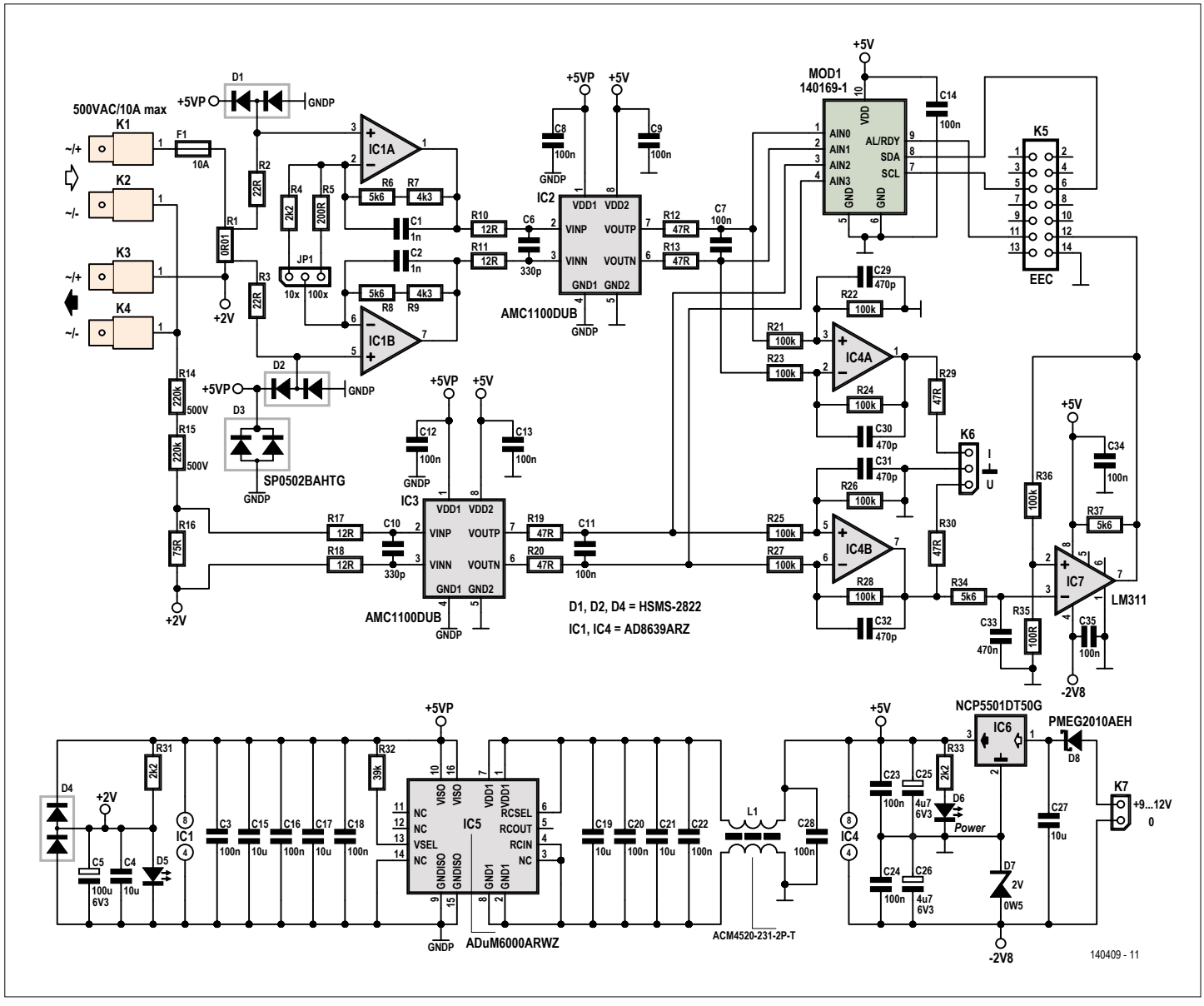


Figure 1. The power meter uses a pair of AMC1100 ICs to measure voltage and current.

output voltage of 5 V or 3.3 V. The maximum working insulation voltage V_{IORM} is 846 V_{peak}.

The circuit

The easiest way to measure current is to connect a resistor in series with the load and measure the voltage over the resistor. That's how virtually every multimeter works. **Figure 1** shows the complete circuit diagram, with the measurement terminals on the left (two inputs and two outputs; K1 to K4). The sense resistor R1 (0.01 Ω) is connected between K1 and K3. There is also a 10-amp fuse in series with the resistor.

Faston blade ('spade') terminals mounted on the PCB with screws are used for the measurement inputs and outputs. They provide good mechanical and electrical connections and can easily accommodate fairly thick wires.

To minimize the effect on the load, the resistance of the sense resistor should be kept as low as possible, but the voltage drop over the resistor also needs to be high enough for accurate measurement at the lowest current level you want to measure. If the sense voltage is very low, you can always boost it with an opamp. Here this is handled by an adjustable gain stage built around an additional opamp (IC1) ahead of the isolation amplifier. For this we choose the AD8639, a dual opamp with auto-zero inputs and rail-to-rail outputs. The typical offset voltage of this IC is 3 µV (9 µV max.), and the maximum offset drift is only 0.06 µV. It is therefore an outstanding choice for this application, for both AC and DC measurements. The gain bandwidth product is 1.35 MHz. This means that the bandwidth at the maximum desired gain is still greater than 10 kHz, which is more than enough for our purpose.

IC1 is configured as an instrumentation amplifier with a differential gain of $1 + 2 \times (R6 + R7)/R_G$, where R4 or R5 can be selected for R_G. The gain (the ratio of the voltage between the outputs of IC1A and IC1B to the voltage over R1) is 1 when no jumper is fitted on JP1. You can fit a jumper on JP1 to obtain a gain of 10 or a gain of 100. In combination with the gain of IC2, these gain settings provide three current ranges with sinusoidal currents: 17.7 A (gain = 1; limited to 10 A by fuse F1), 1.77 A (gain = 10), and 0.177 A (gain = 100). The maximum RMS values will be lower for signals with high crest factors. Depending on the noise level, the minimum power that can be measured with the circuit is less than 0.1 W (ADC output at −50 dB full scale).

To keep the signal on the sense resistor within the common-mode range of IC1, one end of R1 is connected to a +2 V bias voltage provided by an LED (D5) on the isolated side of IC5, which acts as a reference diode. D5 also serves as a power indicator for the isolated side.

The maximum allowable differential voltage of the input signal to the AMC1100 isolation amplifier in the next stage (IC2) is 250 mV. The IC has a gain of 8, resulting in a maximum output voltage of 2 V. IC2 has a differential input with a switched-capacitor circuit. The output noise level is specified as 3.1 mV_{rms} without further details. If this specification applies to the full bandwidth, the actual noise level will be lower if the bandwidth is limited ahead of the ADC. For this reason, we developed a filter board with four sections, which is inserted ahead of the ADC module (see the Filter Module inset). If the ADC were connected directly to the output of IC2, the signal to noise ratio would be 645. This can be increased by a factor of √60 by reducing the input bandwidth of the ADC, yielding an S/N ratio of 5,000. This makes it possible to measure smaller signals over the sense resistor.

The voltage over terminals K1 and K2 is measured by a second AMC1100 (IC3). The voltage divider R14-R15-R16 attenuates the voltage over these terminals by a factor of 5,900, resulting in a voltage of 39 mV at the input of IC3 at 230 VAC line voltage. Resistor R16 of the voltage divider is connected to the +2 V reference voltage. The outputs of the two AMC1100s are connected to the four inputs of the ADS1115 BoB.

As previously mentioned, the circuit can also be used to measure high voltages safely with an oscilloscope or multimeter. The differential outputs of the AMC1100s are not particularly suitable for this; a single-ended output is easier to use. An additional AC8639 dual opamp (IC4) is included for this purpose. It converts the differential signals from the two isolation amplifiers into single-ended signals with the same amplitude. The common-mode offset at the outputs of the AMC1100 (2.55 V typical at V_{DD} = 5 V) is reduced to nearly zero but not entirely eliminated. An offset of several millivolts (less than 12 mV) is possible at the individual outputs of IC4, depending on the difference between the DC voltages at the individual outputs of the AMC1100.

The common-mode input range of the AD8639 extends from −0.1 V to 3 V with a single 5 V supply voltage. To bring the output voltages of the AMC1100s within that range, voltage dividers R21/R22 and R25/R26 reduce the DC voltage on the outputs of each AMC1100 from 2.55 V to 1.275 V. According to our measurements, the maximum differential output signal that the AMC1100 can deliver is approximately 2.5 V (1.25 V_{peak}) This means that the maximum voltage on the inputs of IC4 will not exceed 1.9 V (1.275 V_{DC} plus 0.625 V, which is half the peak AC voltage). That is well within the common-mode range of the AD8639.

An LM311 comparator (IC7) is included to detect zero crossings. It has an open-collector output. This IC operates with +5 V and −2.8 V supply voltages instead of +5 V and ground to allow zero crossings to be detected fairly easily without a lot of extra components. Resistor R37 is a pull-up for the output, and R35/R36 provide strong positive feedback to ensure reliable zero crossing detection. The comparator output is connected to pin 12 of K5. The best zero crossing detection accuracy is obtained with negative pulses. In that situation the output of IC7 is still negative and there is no hysteresis, and the non-inverting input of the comparator is at ground level.

An EEC connector (K5) is provided for transferring the data to the Arduino. It allows the I^2C signals from the A/D converter and several other signals to be sent to connector K2 on an Arduino prototyping shield mounted on the Arduino Uno board.

The 5 V supply voltage for the secondary side is provided by a low-drop voltage regulator (IC6). That is sufficient for most of the ICs in the circuit. However, IC4 also needs a negative supply voltage in order to deliver an AC signal without any DC offset, and we additionally want to be able to measure DC voltages. This means that the outputs of K6 also need to be able to deliver negative voltages. For this purpose, a virtual ground is cre-

ated with the aid of a 2 V Zener diode (D7) in series with the ground pin of the low-drop voltage regulator (IC6).

The isolated supply voltage for the input sections of the AMC1100s and the input amplifier IC1 is provided by the ADuM6000 (IC5). The integrated voltage regulator is set to 5 V by connecting VSEL to Viso. A common-mode choke (L1) is used with this IC to suppress any noise that may be present. It is somewhat overdimensioned with a current rating of 3 A, but we chose it for its low series resistance (50 mΩ per winding).

Measuring with Arduino

The maximum sample rate of the A/D converter in free-running mode is only 860 Hz and the IC has just one physical ADC to serve the four inputs, so simultaneous measurement of voltage and current at the maximum sample rate is not possible. In order to make voltage and current measurements at nearly the same time, the ADC would have to be switched back and forth by I²C bus commands, and that would cause too much delay between the two measurements. (For a more detailed explanation, see the Escaped from the Labs article "And Then Just a Bit of Software" in the July & August 2015 edition).

We therefore chose a measuring method in which a series of voltage measure-

Component List Measurement Board

Resistors

R1 = 0.01Ω 2W, 1% (Ohmite FC4L64R010FER)
R2,R3 = 22Ω, 0.25W, 5%, SMD 0805
R4,R31,R33 = 2.20kΩ, 0.125W, 1%, SMD 0805
R5 = 200Ω, 0.125W, 1%, SMD 0805
R6,R8,R34,R37 = 5.6kΩ, 0.125W, 1%, SMD 0805
R7,R9 = 4.3kΩ, 0.125W, 1%, SMD 0805
R10,R11,R17,R18 = 12Ω, 0.125W, 1%, SMD 0805
R12,R13,R19,R20,R29,R30 = 47Ω, 0.125W, 1%, SMD 0805
R14,R15 = 220kΩ, 1.5W, 1%, 500 V, SMD 2512
R16 = 75Ω, 0.125W, 1%, SMD 0805
R21–R28,R36 = 100kΩ, 0.125W, 1%, SMD 0805
R32 = 39kΩ, 0.125W, 5%, SMD 0805
R35 = 100Ω, 0.125W, 1%, SMD 0805

Capacitors

C1,C2 = 1nF 50V, 5%, SMD 0805, C0G/NP0
C3,C7,C11,C14,C23,C24,C34,C35 = 100nF 50 V, 10%, SMD 0805, X7R
C4,C15,C17,C19,C21 = 10µF 10V, 10%, SMD 0805, X7R
C5 = 100µF 6.3V, 20%, SMD Case A (1206), tantalum
C6,C10 = 330pF 50V, 5%, SMD 0805, C0G/NP0
C8,C9,C12,C13 = 100nF 50V, 10%, SMD 1206, X7R
C16,C18,C20,C22,C28 = 100nF 25V, 10%, SMD 0603, X7R
C25,C26 = 4.7µF 6.3V, 10%, SMD Case R (0805), tantalum
C27 = 10µF 25V, +80/–20%, SMD 1206, Y5V
C29-C32 = 470pF 50V, 5%, SMD 0805, C0G/NP0
C33 = 470nF 16V, 10%, SMD 0805, X7R

Inductor

L1 = ACM4520-231-2P-T, SMD, common mode choke 3A, 2 x 50mΩ, 230Ω @100MHz

Semiconductors

D1,D2,D4 = HSMS-2822-TR1G, SMD SOT-23
D3 = SP0502BAHTG, SMD SOT-23
D5,D6 = LED, green, SMD 0805
D7 = BZT52C2V0-7-F, SMD SOD-123 (zener diode 2 V 0.5W)
D8 = PMEG2010AEH, 20V 1A, SMD SOD-123F
IC1,IC4 = AD8639ARZ, SMD SOIC-8
IC2,IC3 = AMC1100DUB, SMD Gullwing-8 (SOP-8)
IC5 = ADuM6000ARWZ, SMD RW-16 (SOIC_W-16)
IC6 = NCP5501DT50G, SMD DPAK3
IC7 = LM311D, SMD SOIC-8

Miscellaneous

K1–K4 = 0.25'' Faston plug with 3.3mmm screw terminal
K5 = 14-pin (2x7) pinheader, 0.1'' pitch
K6,JP1 = 3-pin pinheader, 0.1'' pitch
K7 = 2-pin pinheader, 0.1'' pitch
MOD1 = 10-pin (2x5) socket, 0.1'' pitch
F1 = fuseholder, PCB mount, 32A 600V, 6.3 x 32mm
F1 = 10A, 500V_{AC}/300V_{DC}, antisurge, 6.3 x 32mm
Module no. 140169-91 (ADS1115-eBOB)
Filter PCB no. 140169-2
PCB no. 140409-1

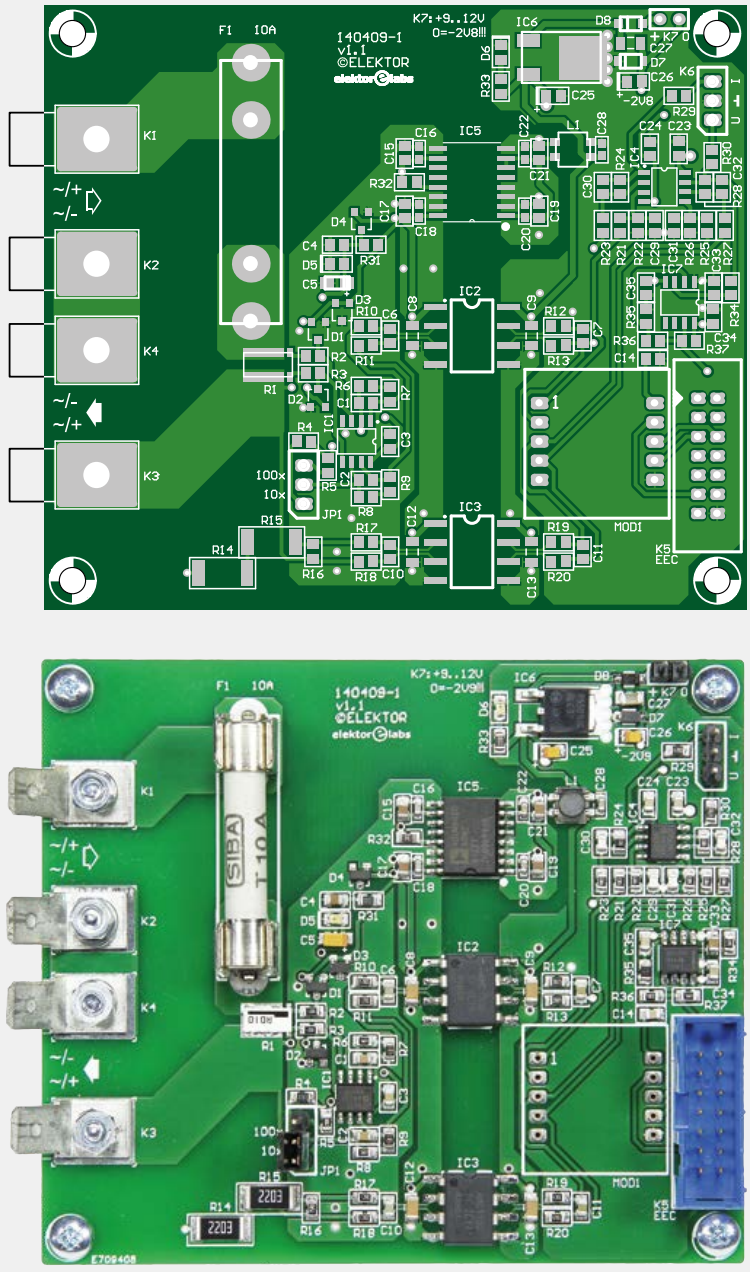


Figure 2. The PCB layout of the measurement board. The wide copper-free strip visible in the middle under the three large ICs provides galvanic isolation between the primary and secondary sides.

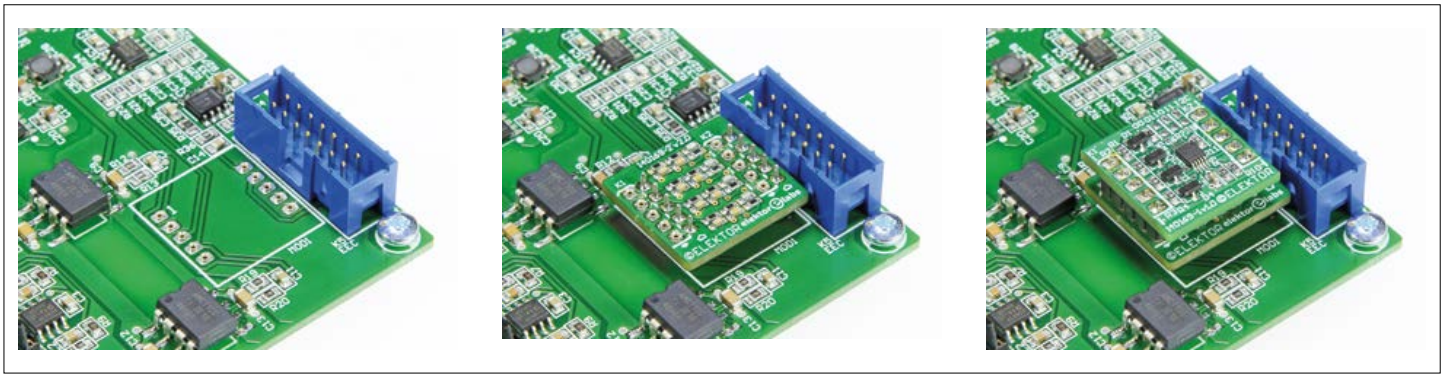


Figure 3. The A/D converter BoB and the filter board are stacked on top of each other and inserted in connector MOD1 on the measurement board.

ments (172, to be precise) are first made, followed by a series of the same number of current measurements. Both series are triggered by zero crossings of the voltage in order to define the phase relationship between the voltage and current measurements. For this purpose, the hardware of the power meter includes a comparator (IC7) connected to pin 9 of the Arduino board (I/O pin B1), which detects the zero crossings. When this synchronization signal is missing, the software automatically switches to DC measurement mode, and it switches back to AC mode when the signal is again present.

The software for the power meter (a sketch for the Arduino Uno, which can be downloaded for free from [4]), measures and/or calculates the following quantities (see also [5]):

For AC:

$$P_{VA} = \frac{\sum_{i=1}^{N} V[i] \times I[i]}{N}$$

$$V_{RMS} = \sqrt{\left(\sum_{i=1}^{N} V^2[i]\right) / N}$$

$$I_{RMS} = \sqrt{\left(\sum_{i=1}^{N} I^2[i]\right) / N}$$

$$P_{RMS} = V_{RMS} \times I_{RMS}$$

$$P\text{-factor} = \frac{P_{RMS}}{P_{VA}}$$

For DC:

$$V = \left(\sum_{i=1}^{N} V[i]\right) / N$$

$$I = \left(\sum_{i=1}^{N} I[i]\right) / N$$

$$P = V \times I$$

These quantities are displayed on the LCD in a cyclic loop. The screens for the various readings are shown in **Figure 4**.

Construction

The measurement board PCB (140409-1) has a fairly spacious layout (**Figure 2**). Nearly all the components are SMDs, which of course makes assembly more difficult for relatively inexperienced builders, but none of the components are extremely small. Mount all of the SMDs first, followed by the headers, connectors and fuse holder. Then attach the blade connectors firmly to the board with small screws, nuts and split washers. The ADS1115 BoB (140169-1) is available fully assembled from the Elektor Store.

You will have to assemble the associated filter board (PCB 140169-2) yourself. It has SMDs on both sides. The filter board has two 5-position connectors on the top side for connection to the ADC BoB and two 5-position pin headers on the bottom side for attachment to the measurement PCB. Plug the ADC BoB on top of the filter board, and then plug the filter board into the MOD1 connector on the measurement PCB. The Arduino Uno with the prototyping shield is connected to the measurement board by a piece of 14-lead flat cable with matching IDC headers, as shown in the photos.

It is essential to mount the measurement board safely in a properly insulated enclosure in order to prevent accidental contact with high voltage (AC line voltage) while making measurements.

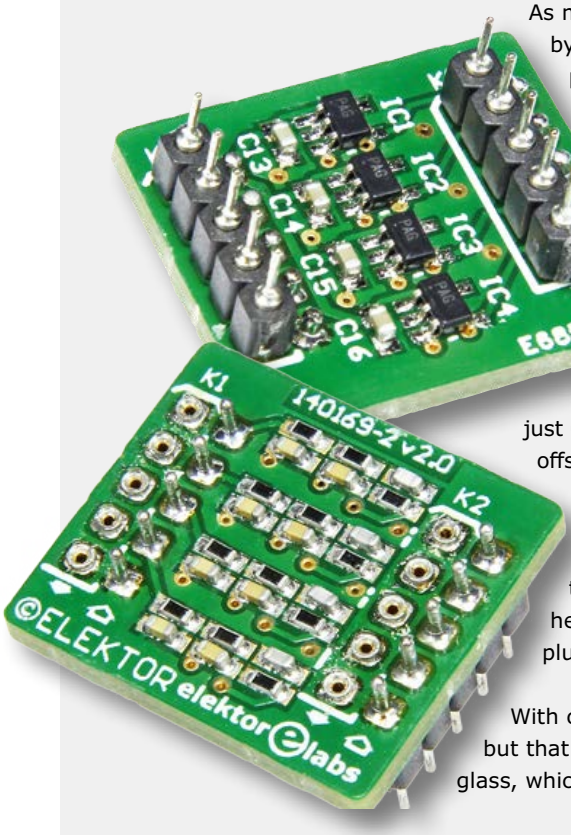
You can let the blade connectors protrude from the enclosure at one end, but a safer option is to connect them to banana jacks or suitable AC power connectors. At the other end you need an opening for the flat cable, and you can fit some jacks or sockets for the measurement signals from K6 and a power connector wired to K7. On the Arduino prototyping shield you have to fit two extra connectors to transfer the signals to the EEC connector already present on the shield. For this purpose, connect K2 pin 11 to K4 pin 1 and K2 pin 12 to K4 pin 2, using thin insulated wire or enameled copper wire. The measurement board can be powered from an AC adapter with an output voltage of 9–12 V DC and a current rating of 200 mA or more. The Arduino board must be powered from a separate AC adapter.

Calibration

The software allows the offset and gain (or attenuation) of the analog front end on the measurement board to be adjusted to calibrate the voltage and current measurement. The adjustment factors are saved in the internal EEPROM of the Arduino at the end of the calibration procedure, so you should only have to do the calibration once.

After a reset the software checks whether the relevant memory locations have been

Filter Module

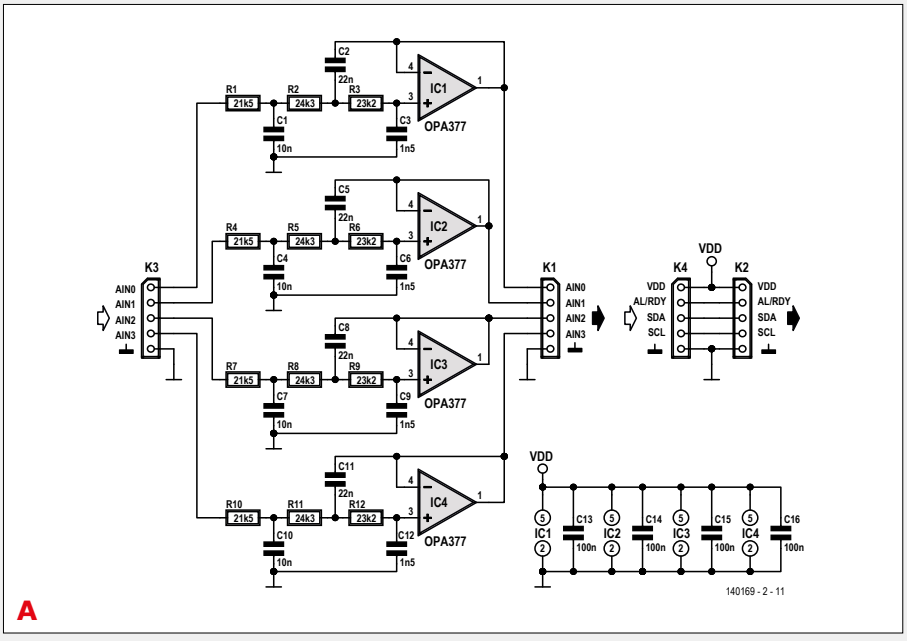


As mentioned in this article, the sensitivity of the meter can be increased considerably by using filters to drastically reduce the bandwidth ahead of the ADC module. For this purpose we developed a filter module containing four third-order active filters (one for each input of the module). They are dimensioned for a cutoff frequency of 1 kHz. That value may appear fairly high, but it was selected to keep the effect on the phase relationship of 50 Hz or 60 Hz measurement signals and their first harmonics as small as possible.

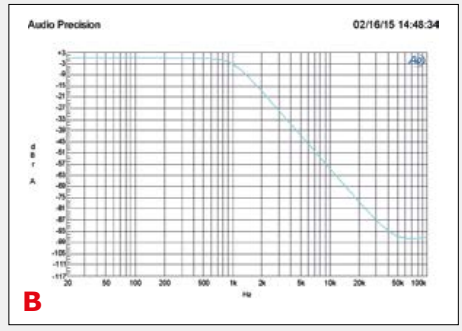
We chose a Sallen & Key configuration for the filters (**Figure A**) with a third-order Butterworth characteristic, as shown by the frequency response curve in **Figure B**. We opted for OPA377 opamps. These opamps feature rail-to-rail operation and have very good noise characteristics for types with CMOS inputs. The input bias current is typically just 0.2 pA, so the filter resistors do not cause any additional offset. The typical input offset and offset drift versus temperature are just 0.25 µV and 0.32 µV/°C, respectively.

The compact PCB fits precisely beneath the ADS e-BoB. To make this possible, the SMDs are placed on both sides of the board (**Figure C**). The board has two connectors on the top side (socket headers or SIL IC sockets) and two pin headers on the bottom with thin round pins, so that everything can simply be plugged together.

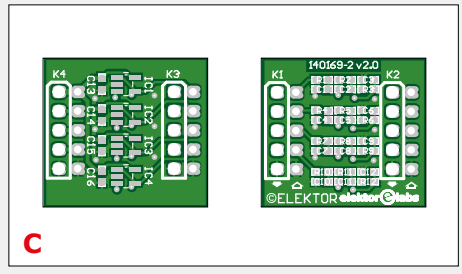
With components mounted on both sides of the board, assembly is certainly not easy, but that is the price for keeping the board so small. Work carefully and use a magnifying glass, which comes in very handy here.



A



B



C

Component List Filter Module

Resistors

R1,R4,R7,R10 = 21.5kΩ 1%, 0.1 W, SMD 0603
R2,R5,R8,R11 = 24.3kΩ, 1%, 0.1 W, SMD 0603
R3,R6,R9,R12 = 23.2kΩ, 1%, 0.1 W, SMD 0603

Capacitors

C1,C4,C7,C10 = 10nF 50V, 5%, C0G/NP0 or X7R, SMD 0603
C2,C5,C8,C11 = 22nF 50V, 5%, C0G/NP0 or X7R, SMD 0603
C3,C6,C9,C12 = 1.5nF 50V, 5%, C0G/NP0 or X7R, SMD 0603
C13,C14,C15,C16 = 100nF 50V, 10%, X7R, SMD 0603

Semiconductors

IC1,IC2,IC3,IC4 = OPA377AIDBVT

Miscellaneous

K1,K2 = 5-pin connector, 0.1'' pitch
K3,K4 = 5-pin pinheader, 0.1'' pitch
PCB # 140169-2

written, which means they do not contain the value '255'. If they do, the program automatically jumps to the calibration routine; otherwise it goes directly to measurement mode. You can redo the calibration at any time by pressing S1 right after the supply voltage is switched on. At each step of the calibration procedure, you first see a screen that tells you what you are about to do, and after you press S1 the screen for the actual calibration adjustment appears.

The first step is to zero the offset of the voltage measurement (*0V input*). Leave terminals K1 and K2 unconnected, and press buttons S1 and/or S2 on the Arduino prototyping shield to adjust the reading on the LCD to 0 V. After a 5-second interval with no button presses, the software goes to the next step: setting the gain factor for voltage measurement (*input V*). For this adjustment, connect a known high voltage (50 V or more) to the K1/K2 terminals together with a reliable multimeter, and again use S1 and S2 to adjust the reading on the LCD to match the reading on the multimeter.

Here again, a 5-second interval with no button presses takes you to the next step: zeroing the current offset (*0A input*). Start with K1/K2 again disconnected and adjust the offset with the two buttons to obtain a zero reading, the same as for the voltage offset. The final step is setting the gain factor for current measurement (*input I*). For this adjustment, connect a DC voltage source to K1/K2 and connect a load and an ammeter to K3/K4. Then you can adjust the gain factor for current measurement in the same way as previously described for the voltage. ⏮

(140409-I)

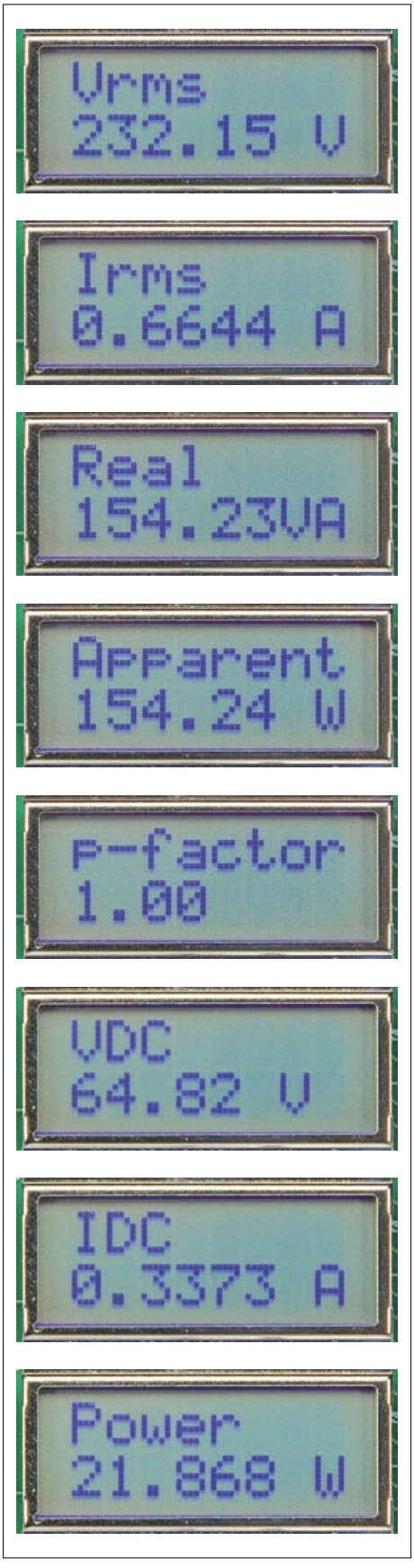


Figure 4. The readings appear sequentially on the display. For AC they are voltage, current, effective power, reactive power and power factor. For DC they are voltage, current and power.

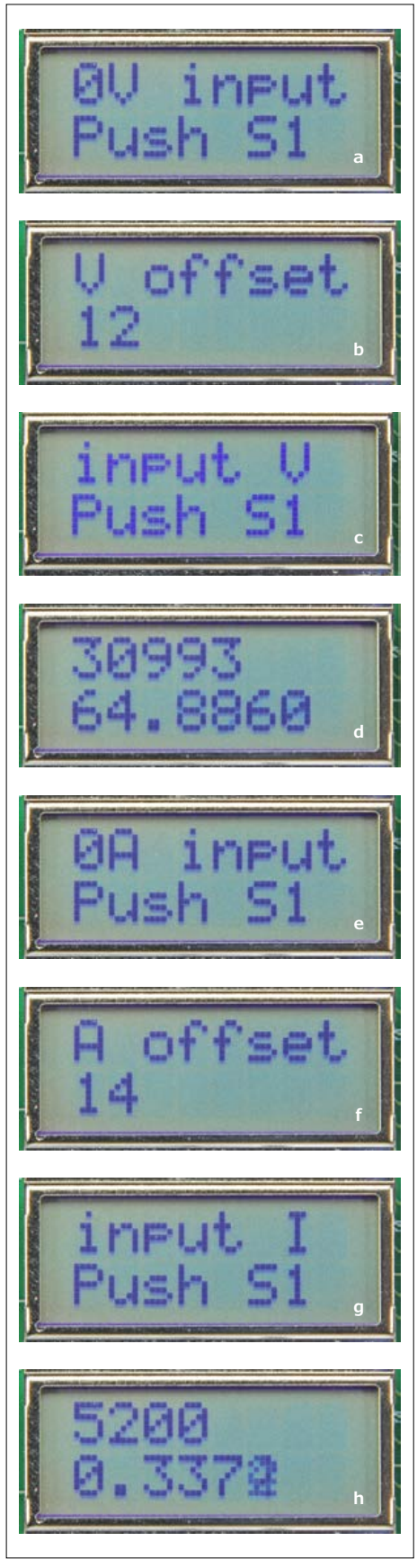


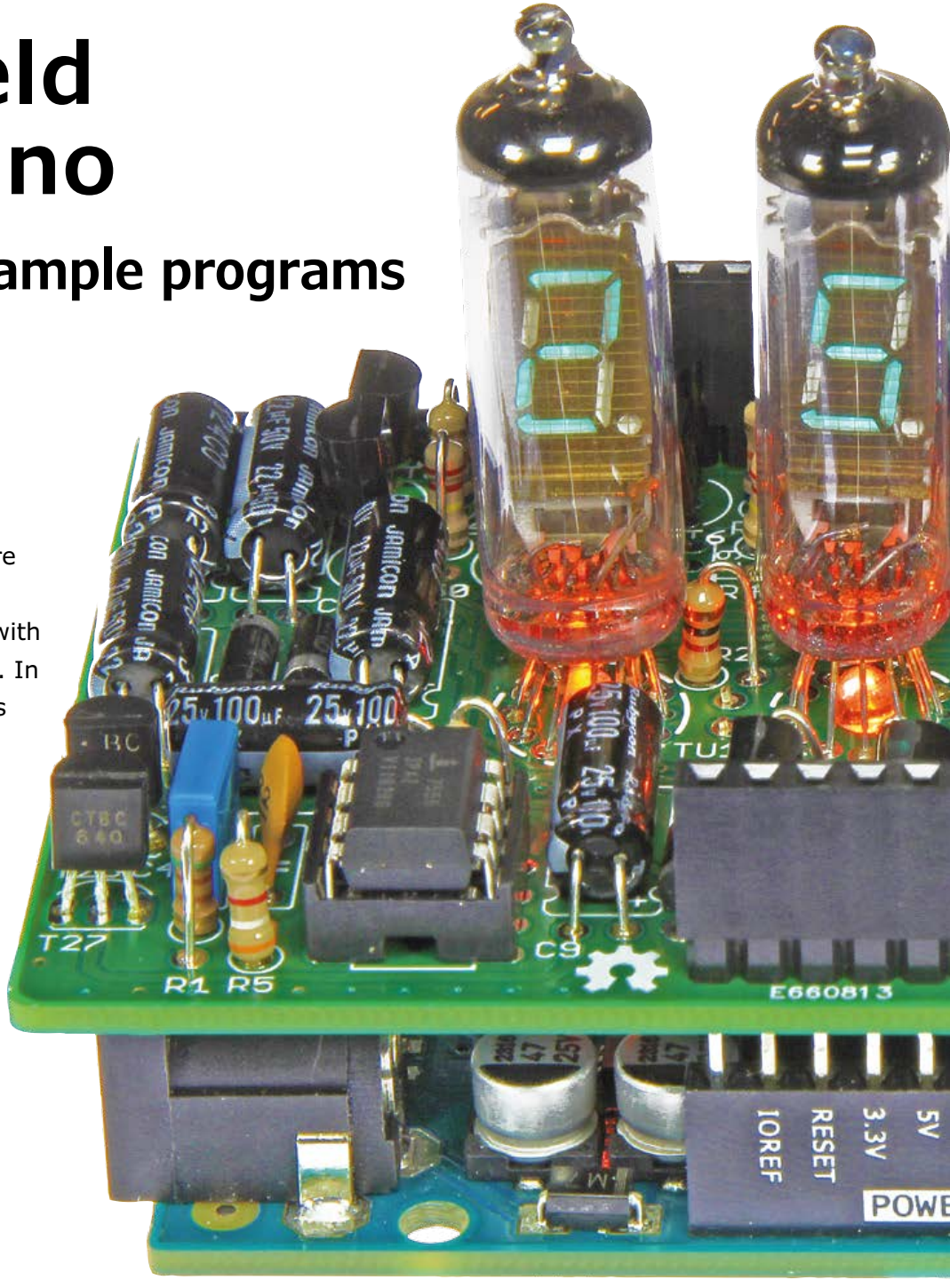
Figure 5. The calibration screens. These are displayed in the following sequence: offset for voltage measurement (a & b), gain for voltage measurement (c & d), offset for current measurement (e & f), and gain for current measurement (g & h).

Web Links

[1] www.elektormagazine.com/140169
[2] www.elektormagazine.com/140035
[3] www.elektormagazine.com/130485
[4] www.elektormagazine.com/140409
[5] http://openenergymonitor.org/emon/buildingblocks/ac-power-arduino-maths

VFD Shield for Arduino

With several example programs



By **Ilse Joostens** (Belgium)

The Arduino shield presented here contains four Russian IV3(a) 7-segment VFD tubes, together with the necessary control electronics. In addition, a quartet of 3-mm LEDs provide an attractive background illumination. The shield can be combined with an Arduino Uno and there is software available to implement a clock, a voltmeter, a thermometer and a demonstration program.

Nixie tubes have enjoyed renewed popularity for some time with retro & vintage enthusiasts and on the Internet there are countless designs for Nixie clocks to be found. In Elektor too, we have published several projects with Nixie tubes in recent years. However, in this project we do not use Nixies, but instead make use of VFD (Vacuum Fluorescent Display) tubes. These are somewhat different from the well-known Nixie tubes and are currently gaining popularity.

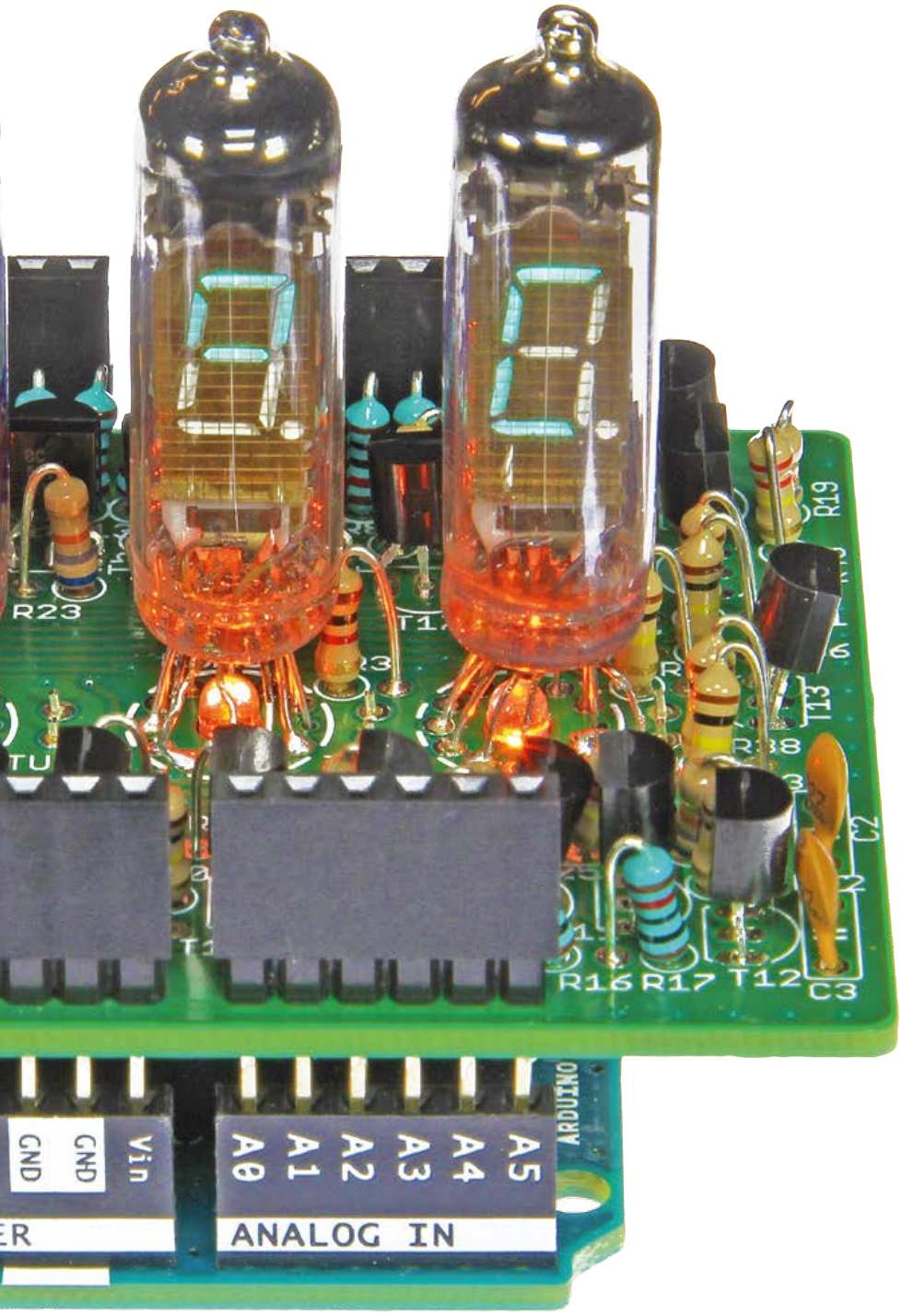
This Arduino shield has been designed from an educational viewpoint and therefore does not have any SMD components or parts that are difficult to obtain. There is a good chance that the typical electronics hobbyist will have the majority of the necessary parts already. In addition, this shield does not make use of dangerous voltages (maximum of 35 V).

Operating principle of the IV-3 VFD tube

The IV-3 VFD tube (**Figure 1**) is a miniature VFD tube with a digit height of 8 mm. It comprises a glass tube with a vacuum, a grid and seven phosphor coated segments plus a decimal point on a ceramic substrate.

An electric current flowing through the filament it will heat it up and cause electrons to be released. Because this filament is covered in a thin layer of alkaline metal oxides it already starts to release electrons efficiently at relatively low temperatures. As a result of the low temperature, the glow of the filament is barely perceptible and therefore does not interfere with the display.

When a positive voltage is applied to one of the segments, the electrons that



Characteristics

- compatible with Arduino development boards
- regulated 12-V power supply voltage, via Arduino board
- possible applications: clock, thermometer, voltmeter, counter, scoreboard, ...
- multiple Arduino example ketches available

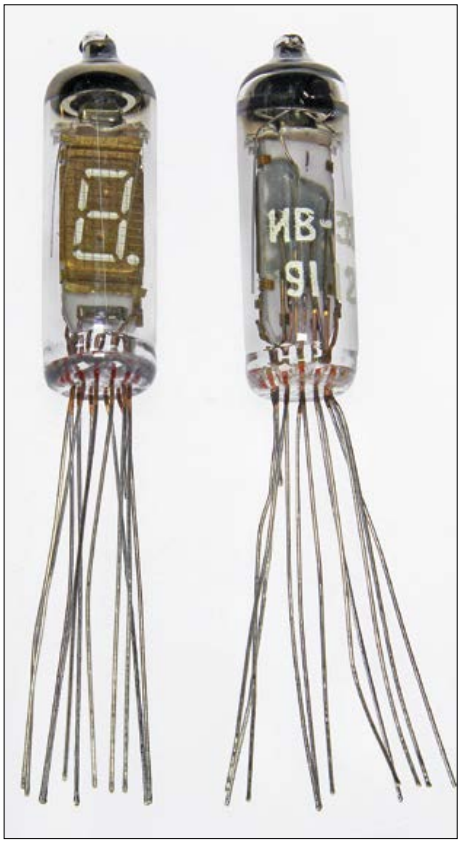


Figure 1. Front and back of an IV-3 VFD tube.

are in the vicinity of the filament will be attracted and collide with the phosphor layer. The energy liberated through this impact causes the phosphor to light up in the familiar green-blue color. This operating principle is comparable to that of a cathode ray picture tube (CRT), but here the phosphor was chosen so that it will light up clearly at the low energy level of these electrons. As a consequence, about 30 V is sufficient for the segments to glow brightly, this is in contrast with the many kilo-volts required in a standard cathode ray picture tube.

The tube also contains a grid, which is placed between the filament and the segments. By applying a small negative voltage to this grid the electron stream towards to segments is blocked, irrespective of the voltage on the segments. In this way you can turn the VFD tube 'on' or 'off' in its entirety. In this aspect the VFD tube actually functions like a triode and in theory you could even use it as an amplifier, see [1].

Now it is — obviously — not the intention to build an audio amplifier, but the grid permits the control of multiple VFD tubes (just as with 7-segment LED displays) to be multiplexed, which we have effectively implemented with this shield.

Finally, the dark-gray/silver colored layer at the top the VFD tube is the 'getter'. This is a chemically reactive layer which helps maintain the vacuum by absorbing any remnant gas molecules.

Design and operation of the circuit

The circuit of **Figure 2** consists mainly of four parts: a voltage converter which increases the power supply voltage of

12 V to about 32 V, a power supply for the filaments, a driver circuit (buffers) for the segments and grids, and finally a switching transistor and four LEDs for the background lighting.

The voltage converter is essentially a voltage tripler built around a 7555 IC, a buffer stage and a few capacitors and Schottky diodes. In theory the output voltage should amount to 36 V when the supply voltage is 12 V. But in practice it is a little less and also because of the presence of D1 in the ground line a voltage of about 32 V is reached, which is sufficient for the IV-3 tubes that we use here. The 7555 (IC1) is configured as an oscillator, where C1 and R5 determine the output frequency. Since the output of the 7555 can only supply a limited amount of current, a push-pull driver stage around T26 and T27 has been added. This does, however, create a problem in that the output voltage of the 7555 in the High state is not high enough to drive T26 fully into saturation. For this reason the 7555 is powered via R4 from the doubled power supply voltage at the node of D3/D4/C6.

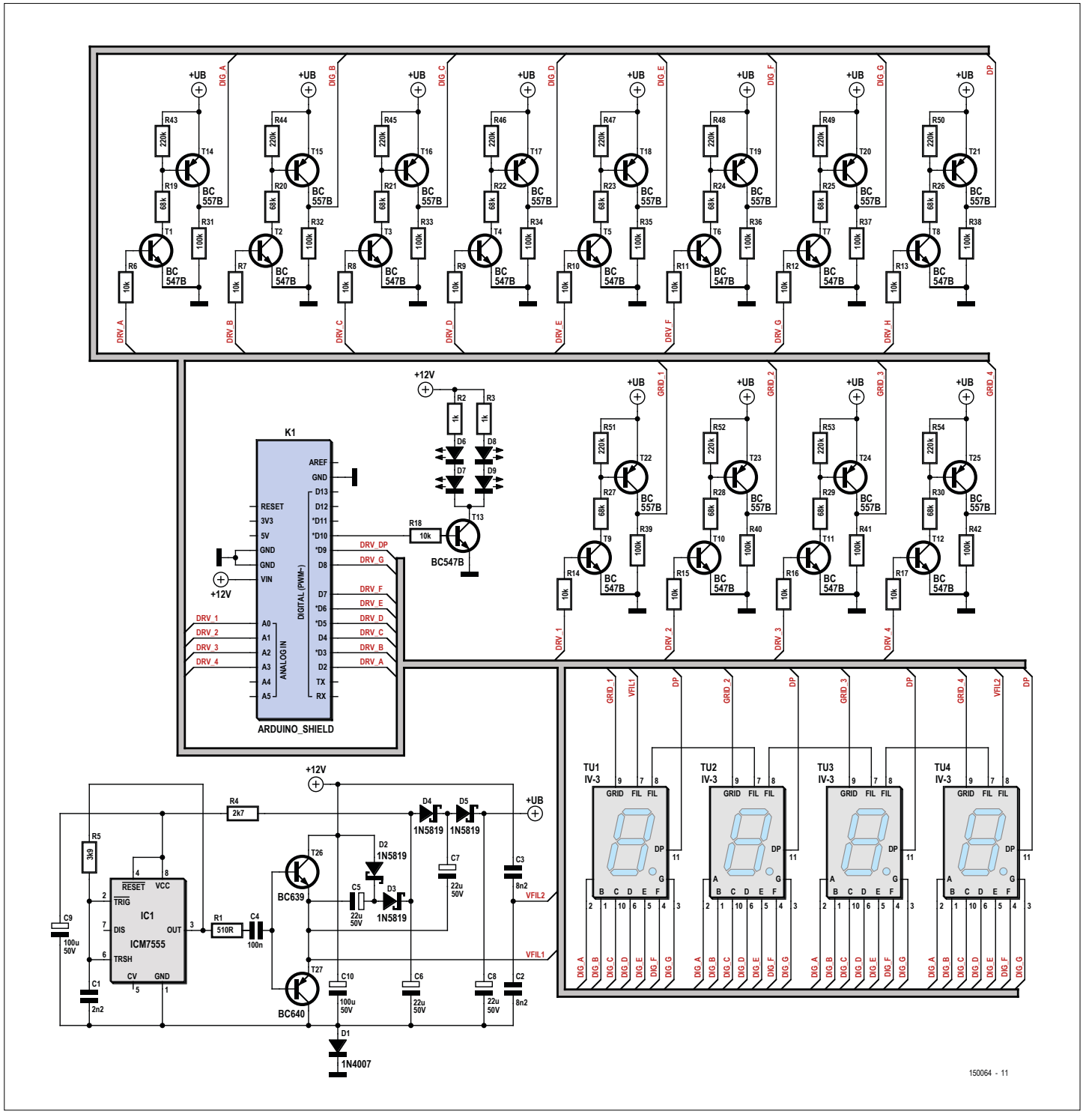


Figure 2. The circuit for the control of the VF tubes comprises a voltage tripler around IC1 and a number of driver stages for the segments and grids (T1 through T12 and T14 through T25).

No Nixies for a change, but VFD tubes

In practice, the power supply voltage of the 7555, because of the voltage drop across R4, amounts to about 14 V. However, the base voltage of T26 is now much higher than its collector voltage, which causes its base-collector diode junction to go into conduction and which would limit the output voltage from the 7555. R1 and particularly C4 have been added to prevent this from happening, where C4 also provides a sufficiently high base-current impulse towards T26 whenever the output of the 7555 switches from low to high. This allows capacitors C5 through C8 of the voltage-tripler to be charged quickly. This increases the efficiency of the circuit considerably, which also has the advantage that there is no need to heatsink T26 and T27, even when there is a sizable load connected to the output voltage.

Finally, the voltage drop across D1 ensures that the entire power supply is raised about 0.7 V above the ground level. When the voltage on a segment or grid of the VFD tube is pulled to ground then this voltage is actually slightly negative with respect to the filament so that the segments are guaranteed to stay dark. The filaments of the VFD tubes in this design are connected in series for simplicity and to limit the number of components that are required. Additionally, the filaments are not powered with DC but with AC, so that the voltage gradient across the filaments does not result in a visible difference in brightness between segments (and VFD tubes). For this purpose, one end of the filaments is connected to the common emitters of T26 and T27 and the other end with the junction of C2 and C3. Because C2 and C3 block any DC current the result is not just an AC voltage, but their impedance also limits the current into the filaments. For this reason the values of C2 and C3 (in combination with R5/C1, which determine the frequency) are quite critical.
Finally, the control of the segments and grids is obtained with the aid of NPN/PNP transistor pairs. At a logic High level from the Arduino (5 V), the PNP transistor switches the 32-V power supply voltage to the corresponding segment or grid. At a logic Low level the segment or grid is connected to ground via a 100-kΩ resistor. The segments and decimal points of the four tubes are connected in parallel so that a total of 12 data lines are required for the multiplexed control. Consequently the SDA and SCL signals from the Arduino remain available and therefore an I^2C RTC, for example, or another I^2C IC can be used in combination with this shield.

Software

Several different example programs are available for this shield: clock, voltmeter, thermometer and a demonstration program. All programs are Arduino sketches and the complete set can be downloaded as a zip file from [2]. The source code for each program contains a detailed explanation about how to use it, how it works and any external libraries that may be required.
The clock indicates the date and time. The Arduino does not have a real-time-clock, so the user has to set the date and time via the serial connection. This is easily done with the 'Serial Monitor' in the Arduino IDE.
The voltmeter measures the voltage at pin A5 of the Uno and shows the measured value on the VFD tubes. The input range goes from 0 to 5 V (display: 0.00 to 5.00 V).
The thermometer reads the temperature from a 1-Wire sensor and shows the value obtained in degrees Celsius. This program supports the DS18B20, DS18S20 and DS1820 devices. The program enumerates the 1-Wire bus periodically. When a supported chip is found the program reads the temperature from it and then shows the value on the tubes. Because of this method of operation, it is possible for the user to change the 1-Wire device while to program is active.
The demonstration program shows an animation on the tubes and a PWM animation on the LEDs. This program can also be used to verify that the shield has been built correctly.

Each example program contains a driver section for the tubes. This driver is interrupt based with the aid of an ISR (interrupt service routine). The tubes are activated one at a time in a continuous cycle. There is therefore only one tube illuminated at any one time. The driver contains the following steps:

- Switch the grid and segments of the active tube off as quickly as possible.
- Wait 40 µs to prevent 'ghosting'.
- Select the next tube according to the cyclic pattern.
- Write to pins 2 through 9 to update the segments.
- Switch the grid of the tube on.

When the time between switching off the previous grid and switching on the next grid is too short, then segments of the previous tube can inadvertently have a faint glow. This phenomenon is called 'ghosting'. The waiting time of 40 µs prevents ghosting. This waiting time was determined experimentally. Note that the actual time of switching from one grid to the next is a little longer, because other code is executed as well.
The multiplexing of the tubes is done at a rate of 250 Hz. Each tube is visible 62.5 times per second for a period of nearly 4 ms. This frequency is high enough so that he human eye perceives a flicker-free image.
The information to be displayed is stored in a data structure (`tube_list`), an array of four elements, one for each tube. Each element contains an on/off-flag for the decimal point and an index into a table with the combinations for the segments.
The demonstration program also contains the PWM drive for the LEDs. One PWM cycle is divided into eight steps. The step frequency is 500 Hz, the PWM frequency is 62.5 Hz. At this frequency the LEDs do not appear to flicker to the eye, even at the minimum duty cycle of 1/8.
The ISR is associated with Timer 1. The Arduino 'programming language' contains no elements to work with timers. The source code for the programs uses AVR Libc to implement Timer 1 and the ISR in C code. Note that he Arduino 'programming language' itself also makes use of AVR Libc.

Construction

The printed circuit board designed for this circuit can be seen in **Figure 3**. Only through-hole components have been used, so that construction should not be a problem, even for those with modest soldering skills.

Elektor offers a complete kit with four tubes for this project (available from the Elektor Store). There is an extensive description of the assembly procedure of the shield in a separate (English-language) assembly manual, which is also available from [2], but here we will briefly touch on the salient points.

All parts are mounted on the side with the screen print. Fit the resistors, diodes, capacitors and the IC socket first. The resistors are all placed upright, so one wire is bent back along the resistor body. The electrolytic capacitors are mounted horizontally on the board, so the connecting wires have to be bent into a right-angle first.

After this the headers are fitted. It is a good idea to insert these into the circuit board and then temporarily plug it all into an Uno. Turn everything over and solder a few pins on each connector. Then you know for sure that everything fits. Remove the Uno and solder the remaining pins.

Now it is the turn of the transistors, followed by the four LEDs for the background illumination. The LEDs are mounted in the holes underneath the tubes. The connections for each LED are a little further away. Bend the wires into the correct shape and check a few times until the LED and the wires fit neatly in the appropriate holes.

The VFD tubes are mounted last. These or provided with relatively long connecting wires which are threaded through the circuit board. If this turns out to be difficult, then you can cut a short length from each of the wires in a spiral shape, so that each wire has a slightly different length. Place each tube above the circuit board at such a height that the bottom is just below the top of the headers. With four tubes inserted into the board, solder one wire per tube. Check again that they are all in a neat line and upright and then solder the remaining wires — it is not possible to correct the position of the tubes afterwards! Finally the 7555 IC is plugged into its socket.

Now download the software from [2] and load the demonstration sketch via the USB connection on the Arduino. Disconnect the Arduino from the PC and plug the shield on top of the Arduino, after the USB connector on the Arduino is covered with a small piece of insulation tape (to prevent it coming into contact with the soldered connections on the shield). Connect a AC power adapter, which can supply a regulated output voltage of 12 V (300 mA min), to the Arduino.

When you plug the adapter in, and everything is fitted correctly and soldered, the tubes will count in an endless loop and the brightness of the LEDs will vary slowly. Congratulations, your VFD shield works as it should. Now you can try one of the other software examples or perhaps start to program for the application that you have in mind for this shield. ⏮

(150064)

Web Links

[1] https://www.youtube.com/watch?v=aiszksJs9C8

[2] www.elektormagazine.com/150064

Component List

Resistors

R1 = 510Ω
R2,R3 = 1kΩ
R4 = 2.7kΩ
R5 = 3.9kΩ
R6–R18 = 10kΩ
R19–R30 = 68kΩ
R31–R42 = 100kΩ
R43–R54 = 220kΩ

Capacitors

C1 = 2.2nF
C2,C3 = 8.2nF
C4 = 100nF
C5–C8 = 22µF 50V, radial
C9,C10 = 100µF 50V, radial

Semiconductors

D1 = 1N4007
D2–D5 = 1N5819 Schottky diode
D6–D9 = LED, 3mm, select color
T1–T13 = BC547B
T14–T25 = BC557B
T26 = BC639
T27 = BC640

Miscellaneous

TU1–TY4 = IV3(a) 7-segment VFD tube
8-pin IC socket for IC1
Set of Arduino compatible pinheaders (1 pc 6-pin, 2 pcs 8-pin, 1 pc 10-pin)
Construction kit including 4 VFD tubes: # 150064-71 from www.elektor.com

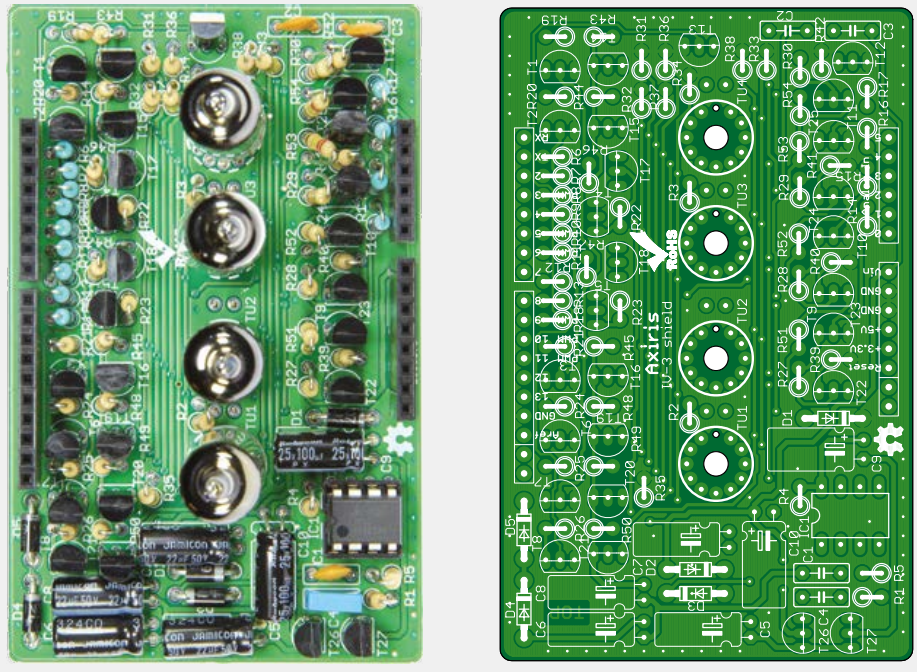


Figure 3. The circuit board contains only through-hole components so construction is straightforward.

RPi Measures Electricity Consumption

Using Python, PHO and MySQL

By **Zeno Otten** (The Netherlands) (www.zenot.nl)

This article describes how to build a smart meter remote readout using a Raspberry Pi. With just a few components and a bit of programming, you can collect and process data from a smart electricity meter and display it on a web page in various forms.

In the author's country household electricity bills have risen by more than 5 percent per year over the last 17 years. Two factors are responsible for this: rising fuel prices on the international market and a constantly increasing number of household appliances and devices that consume electricity. As a result, electricity bills have become a significant factor in household budgets.

If you want to reduce your electricity consumption, you first have to be aware of it and you need some way to measure it effectively. Inline measurement and recording of your electricity consumption can help you reduce your consumption and determine how much power individual devices use. Armed with this understanding and information, you can make decisions that cut your electricity bill.

Measuring your electricity consumption

Every electricity customer has an electricity meter somewhere about the premises. Most older-model meters have a rotating disk. The faster this disk turns, the more electricity you are using. The meter is typically read once a month, and you receive a bill corresponding to your consumption. Clearly, if you want to see how you can reduce your electricity consumption you need real data more often than just once a month. Inline measurement, which means continuous measurement, gives you a lot more data. You can determine the power consumption of individual appliances by measuring how fast the meter disk turns [2, 3].

As result of technological progress and computerization, utility companies are increasingly installing smart meters. These meters can be read remotely, and the meter readings are automatically sent to the utility company. Smart meters can also measure the power fed in from a solar power system, so you can get a deduction for feeding electricity into the grid.

This type of meter has a serial data port for inline measurement of electricity consumption. It's easy to read the data from the meter in digital form through this port. The data can be read out every 10 seconds, which is ideal for recording all sorts of variations in your electricity consumption. If you store the data and process it for displaying on a web page, you can view your electricity consumption any time you want on a PC, smartphone or tablet.

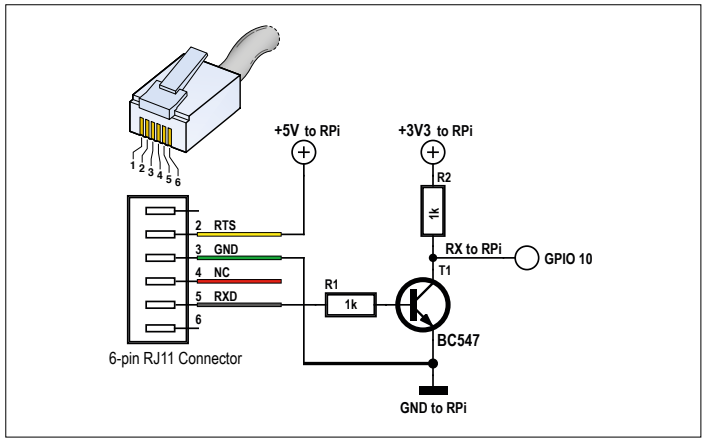


Figure 1. Input circuit for connecting the electricity meter to the RPi.

Many companies are already taking advantage of this ability to monitor and manage energy consumption. They offer various types of "energy management" tools.
However, it's more fun to build one yourself. If you already have a smart meter at home, you can get started with this project. Otherwise you can ask your utility company to install a smart meter.
In this article we show you how to use a Raspberry Pi computer (type 1) with a smart meter to measure, record and analyze your electricity consumption and present the results on a web page. It can also record and display infeed energy from a solar power system.

Hardware

The Raspberry Pi, nicknamed RPi, is presently one of the most popular single-board computers. It gives you a complete Linux-based computer in credit-card format. The RPi has enough I/O and ports to read and process the data from smart electricity meters. In addition, you can store the data on the RPi and even run a MySQL database directly on the RPi. You can also view the results from the RPi with an Apache web server and a PHP web page installed on the board.
The author's electricity meter has a serial output port with an inverted data signal. **Figure 1** shows the circuitry necessary to input this signal to the RPi. Pin 2 is the request input (RTS), pen 5 is the data output (RxD) and pin 3 is the ground connection. The signal levels are 5-V compatible. The data signal is inverted by the BC547 transistor and then applied to GPIO pin 10 of the RPi with a high signal level of 3.3 V, which is the maximum allowable input voltage on the GPIO pins of the RPi. As you can see from the schematic, the signal input circuitry is very simple.
The meter sends a data package every 10 seconds over the RxD line, formatted as a set of lines containing the consumer data (see below). The meter keeps on sending data as long as the RTS line is held at a High level.

Software

Although the hardware for this project does not require much effort, the software will take a bit of your time. The following description of the necessary steps, interleaved with extracts from the program code, clearly shows how a Raspberry Pi can be used as the basis for a remote smart meter readout.

The Raspbian Linux distribution is installed on the SD card of the RPi.
The commands that must be sent to the RPi in a command prompt window are shown in **green** in this article. Code lines in Python are shown in **red**, MySQL instructions are shown in **blue**, and PHP code lines are shown in **brown**.

In order to use the serial port as shown in Figure 1, the serial port login must be disabled (see also [1]). Open the file `/etc/inittab` in a word processor and place a # character before the line `T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100 in`. The RPi normally uses the serial port to send data during the startup process. You have to disable this if you want to program the serial port yourself and use it for your own purposes. To do this, edit the file `/boot/cmdline.txt` and delete all references to `ttyAMA0`, which is the name of the serial port. After editing, the file will look like this:

```
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2
rootfstype=ext4 elevator=deadline rootwait
```

Then reboot the RPi.

Python

The Python programming environment is installed on the RPi as standard. All of the data capture software is written in Python version 2.6 and is available in the download for this project [2]. The following instruction installs a special library with functions that support programming of the serial port:

```
sudo apt-get install python-serial
```

(see [3] for more information)

The smart meter uses the RS232 protocol with the attributes defined as follows in Python:

```
ser = serial.Serial()
ser.baudrate = 9600
ser.bytesize=serial.SEVENBITS
ser.parity=serial.PARITY_EVEN
ser.stopbits=serial.STOPBITS_ONE
ser.xonxoff=0
ser.rtscts=0
ser.timeout=20
ser.port="/dev/ttyAMA0"
```

The data is read in on the RxD line (GPIO 10) [here Dutch "regel" means "line"]:

```
regel = ser.readline()
```

The lines in the data set from the electricity meter look rather cryptic at first:

```
0-0:96.1.1(8549D31373643876483764837648364834)
1-0:1.8.1(01238.000*kWh)
1-0:1.8.2(02224.000*kWh)
1-0:2.8.1(02327.000*kWh)
1-0:2.8.2(04528.000*kWh)
0-0:96.14.0(0002)
```

```
1-0:1.7.0(0000.00*kW)
1-0:2.7.0(0000.12*kW)
0-0:17.0.0(999*A)
0-0:96.3.10(1)
0-0:96.13.1()
0-0:96.13.0()
0-1:24.1.0(3)
0-1:96.1.0(8549D313736438764837648364834)
0-1:24.3.0(121106150000)(00)(60)(1)(0-1:24.2.1)(m3)
(00054.386)
0-1:24.4.0(1)
!
```

The output ends with an exclamation mark (!). You can discover the meanings of all of these lines from the manual for the meter, but the numerical values that we are mainly interested in are easy to see. We store the values in lines 1-0:1.8.1, 1-0:1.8.2, 1-0:2.8.1, 1-0:2.8.2, 1-0:1.7.0 and 1-0:2.7.0 in the variables var1 to var6 of the Python program.
The numerical values are located in positions 10 to 15 of each line:

```
if regel[0:9] == "1-0:1.8.1":
            var1 = regel[10:15]
            print "Consumed (L)     ", var1
```

var1 = Consumption (L) in kWh
var2 = Consumption (H) in kWh
var3 = Infeed (L) in kWh
var4 = Infeed (H) in kWh
var5 = Current power consumption in W
var6 = Current power infeed in W

The values of variables var1 to var6 are stored in a MySQL database every 10 minutes. For this purpose we have to import the MySQLdb library into Python:

```
import MySQLdb as mdb
```

The database is assigned the name "databasenaam" here and the table where the measurements are stored is named "metingen1". The Dutch terms "inlognaam" and "wachtwoord" mean "login name" and "password":

```
con = mdb.connect('localhost', 'inlognaam', 'wachtwoord',
'databasenaam')
        with con:
             cur = con.cursor()
            cur.execute("INSERT INTO metingen1(tijd,-
Data1) VALUES (%s)" , ( now,var1))
             print  now, "Data written in database "
             cur.close()
```

In the actual program P1.py in the download [2], the six variables are stored in the database together with the variable "tijd" [time].

MySQL
The MySQL database can be stored on a computer connected to the same network as the RPi, but it can also be installed directly on the RPi. The author opted to store the database in the microSD card on the RPi board because there was plenty of room for it (the author has an 8-GB SD card).
In order to use a MySQL database on the RPi, you first have to install several items listed below. To do this, enter the following command in the command prompt window:

```
sudo apt-get install mysql-server python-mysqldb
```

During the installation you will be asked for a root password for the database. After this you can create a new database. In this example, we create a database to hold measurement data for the user "user_name".

Start the MySQL browser by entering the following command in the command prompt window:

```
$ mysql -u root -p
```

Then type the following MySQL commands (replace "user_name" by the actual user name and "databasename" and "databaseName" by the actual database name):

```
mysql> CREATE DATABASE databasename
mysql> USE databasename
mysql> CREATE USER 'user_name'@'localhost' IDENTIFIED
BY 'password';
mysql> GRANT ALL PRIVILEGES ON  databaseName.* TO
'user_name'@'localhost'
mysql> GRANT ALL PRIVILEGES ON  databaseName.* TO
'user_name'@'localhost'
mysql> FLUSH PRIVILEGES;
mysql> quit
```

Now you can create a table to hold the measurement data, such as "metingen1" in this example. In the following example the table is created with the two variables "tijd" [time] and "Data1", with the first variable in time-date format (day-month-year-hh:mm:ss) and the second variable defined as a string with a maximum length of 11 characters.

```
mysql > CREATE TABLE metingen1 (tijd timedate,Data1
varchar(11));
```

A total of six data fields are defined in the actual P1.py program.

MySQL is a complete and very extensive programming language. However, in this project we only use the commands (queries) necessary to store the data. Information about other commands and extensions is readily available on the Internet. We also need queries to retrieve the stored data from the database and present it somewhere, for example on a web page. These queries are described further on.

Now you can start filling the database with measurement data captured from the meter by the P1.py program in the download [2].

Web server
The data needs to be accessible and suitable for presentation on a tablet, laptop or other smart device. To do this with the

Figure 2. Overview with current meter readings.

RPi as well, you first have to turn it into a web server. For this purpose we installed the Apache web server software (see [4] for information and the manual). Install the following programs on the RPi by typing the commands listed below in a command prompt window:

```
sudo apt-get update
sudo apt-get install apache2
sudo apt-get install mysql-server
```

Next you have to install the following software in order to work with PHP:

```
sudo apt-get install php5
sudo apt-get install php5-mysql
sudo apt-get install php5-gd
```

To test the web server and PHP, enter the address `http://localhost/index.html` in a browser window to navigate to the web page on the RPi. The test page should appear if everything is working properly.

The web pages are stored at `/var/www/index.html` by default. To implement the web server, replace `index.html` by `/var/www/index.php` with your own content.

Now the web server is running on the RPi. It will most likely also be available on your home network. Computers on your home network can access the RPi at `http://RPi_IP_address/index.php`.
If you want to be able to access the RPi from the Internet as well, you will have to adjust some settings on your router. However, that is highly dependent on your home network and the specific router model.

PHP web page

The web page you access from your web browser needs to link up with the database where the meter data is stored. You also need to be able to choose which data you want to see and how much data should be shown.

This project displays the current meter readings on the web page along with three charts. The charts show the trends in electricity consumption over the last eight hours and the last 24 hours. The average consumption is also determined and shown on the charts.
The third chart shows when appliances are switched on and off and how much power they consume.

Linking to the database

Creating a link to a MySQL database in PHP is easy. The following code lines provide access to the database with the name "database" (replace the user name, password and database name with suitable values).

```
$host = "localhost";
$user = "user_name";
$pass = "password";
$dbase = "database";

$link = mysql_connect($host, $user, $pass);
if (!$link) {
    die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully<br>';

mysql_select_db($dbase, $link) or die('Could not select database.');
```

Now data can be read from the database. For this you use a query which selects the desired data. The following is an example of a query formulated in PHP:

```
$query  =  'SELECT tijd, Data1,Data2,Data3,Data4,Data5,-Data6 FROM metingen1 ORDER BY
                    tijd ASC LIMIT 144'
```

This query selects the readings for the last 144 time intervals with the corresponding Data1 to Data6 values from the "metingen1" table, arranged in increasing time order. It effectively forms a new table containing the requested data, which can be used to generate the charts.

Results

When a browser visits the website at `http://77.162.137.240/p4.php`, the RPi generates a web page with the current meter readings at the top (**Figure 2**).

The "Consumption (L)" figure in kilowatt-hours is the consumption during the low rate period, which is typically between 11pm and 6am and on weekends. The high rate (H) applies the rest of the time. The amount you have to pay per kilowatt-hour depends on your electricity supplier. If you know the rates, you could modify the code to show actual costs here.

If you generate your own electricity, for example with solar panels, the amount of energy fed into the grid is shown by the Production (H) and Production (L) figures.

The current power consumption in watts is also shown.

Trends

We use the tool phpgraph.php [5] to produce the charts in PHP that show the curves of electricity consumption versus time. This library makes it easy to draw charts from a set of data, which in this case consists of energy consumption figures over a specific time period — for example, the last eight hours in **Figure 3**.

In this example time is plotted on the horizontal axis and electricity consumption in watts is plotted on the vertical axis. The zero point is set to the current time and the time axis is negative, so the readings go back in time as you move to the right along the time axis.
The actual times are not shown on the X axis because they would take up a lot of space on the chart. The number of readings per hour is six. This figure was chosen to keep the size of the database within reasonable bounds, but in principle it is also possible to store readings every ten seconds.
From the chart you can see that the consumption during the night (approximately the last five hours on the chart) was approximately 150 to 200 watts. That was mainly due to the refrigerator/freezer and several other small loads.
The peaks in the chart were caused by a ceramic cooktop that we are presently using.
When the solar panels generate more power than your current consumption level, the extra power is fed into the grid. This is also measured by the meter, and it can be seen in the chart shown in **Figure 4**. In that case the consumption is negative.
With the data stored in the database, you can also use another query to display a curve of consumption over the past 24 hours with six readings per hour (**Figure 5**). The query necessary for this takes the following form in PHP:

```
$query2 = 'SELECT tijd, Data1,Data2,Data3,Data4,Data5,-
Data6 FROM metingen1 ORDER BY tijd DESC LIMIT 144';  //
laatste 24 uur, 6 metingen per uur
```

The high peak at roughly 2.4 kW was caused by the cooktop. From this you can also see that the refrigerator/freezer consumes about 160 watts.
The final trend chart (**Figure 6**) shows when loads are switched on or off and how much power they consume (in watts). This is determined by calculating how much the consumption increases or decreases in the previous chart.
The resolution for detecting load switching is approximately 10 watts, which is determined by the output from the electricity meter.

Conclusion

The remote smart meter readout is built around a Raspberry Pi. With some simple circuitry and relatively little effort, you can collect and process data from a smart electricity meter.
The software necessary for this takes more effort, but it allows all of the functions to be implemented on the Raspberry Pi. This makes the remote readout inexpensive and easy to build yourself.

Thanks to the use of standard programming environments such as Python, PHP and MySQL, doing this project yourself should not create any significant problems.
Storing the data in a local MySQL database allows the current and historical data (trend charts) to be presented on a website. This lets you see how much the various appliances in your household impact your monthly electricity bill.
With this knowledge, you can reduce your consumption and see the results directly. You can also see how much power a particular appliance uses by briefly switching it on and off.

(150313-I)

Web Links

[1] www.hobbytronics.co.uk/Raspberry-pi-serial-port
[2] www.elektormagazine.com/150313
[3] http://elinux.org/Serial_port_programming
[4] www.penguintutor.com/linux/RaspberryPi-webserver
[5] www.ebrueggeman.com

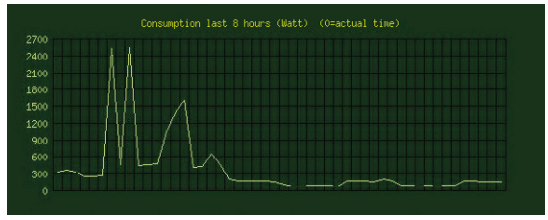


Figure 3. Electricity consumption during the last eight hours.

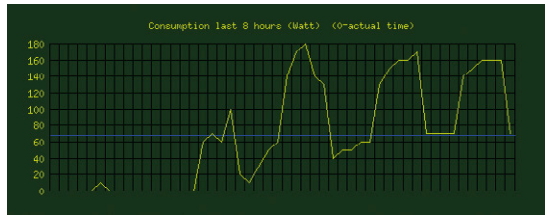


Figure 4. The net consumption can even be negative when the solar panels deliver a lot of power.

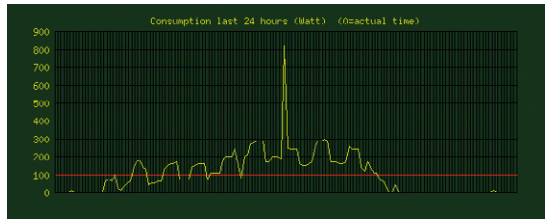


Figure 5. Electricity consumption during the last 24 hours.

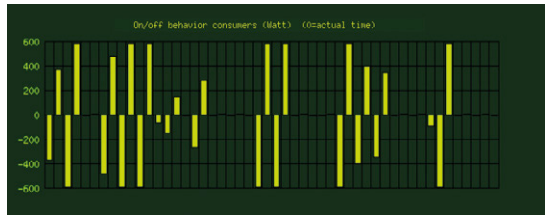
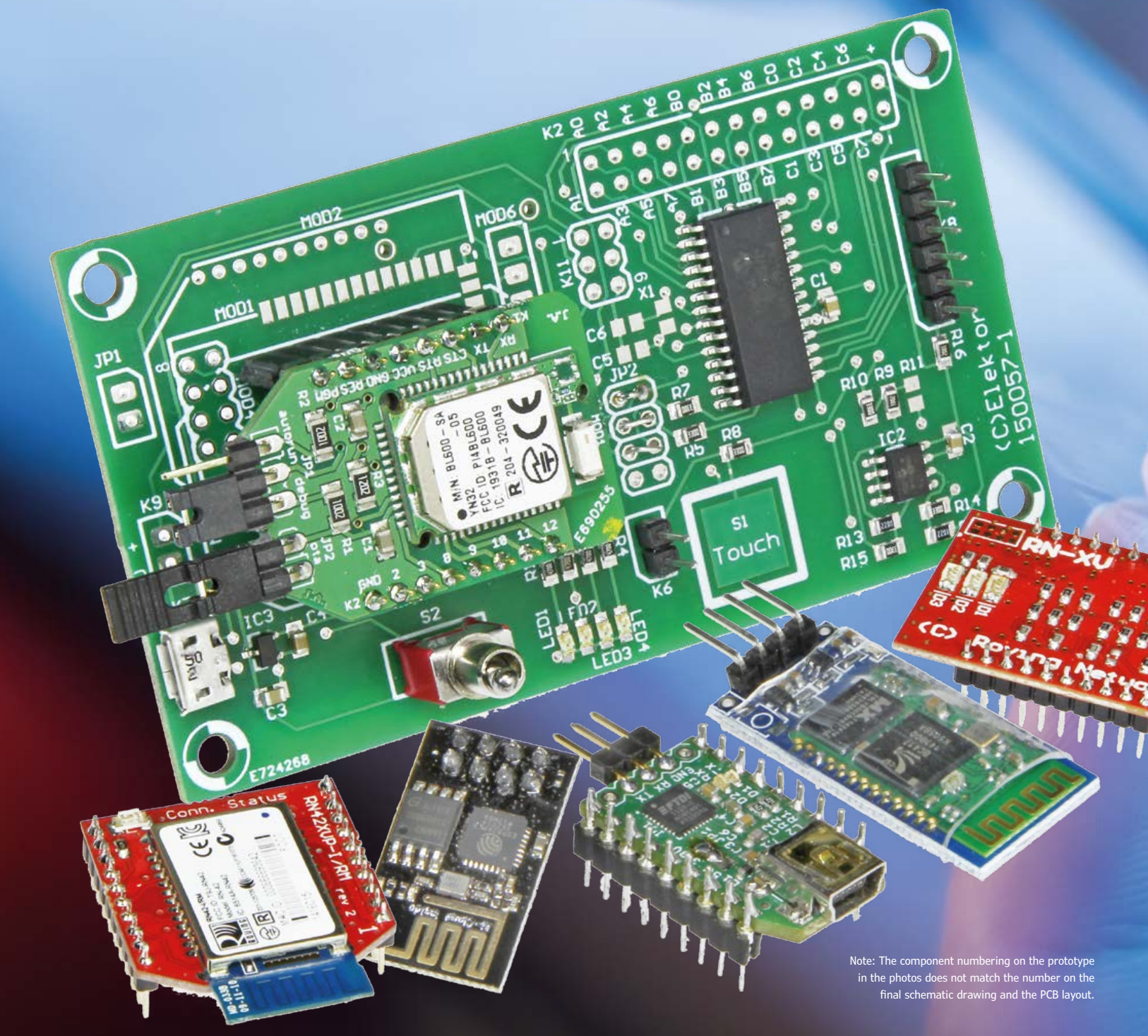


Figure 6. Here you can see when an appliance is switched on or off.

Android I/O Board (1)

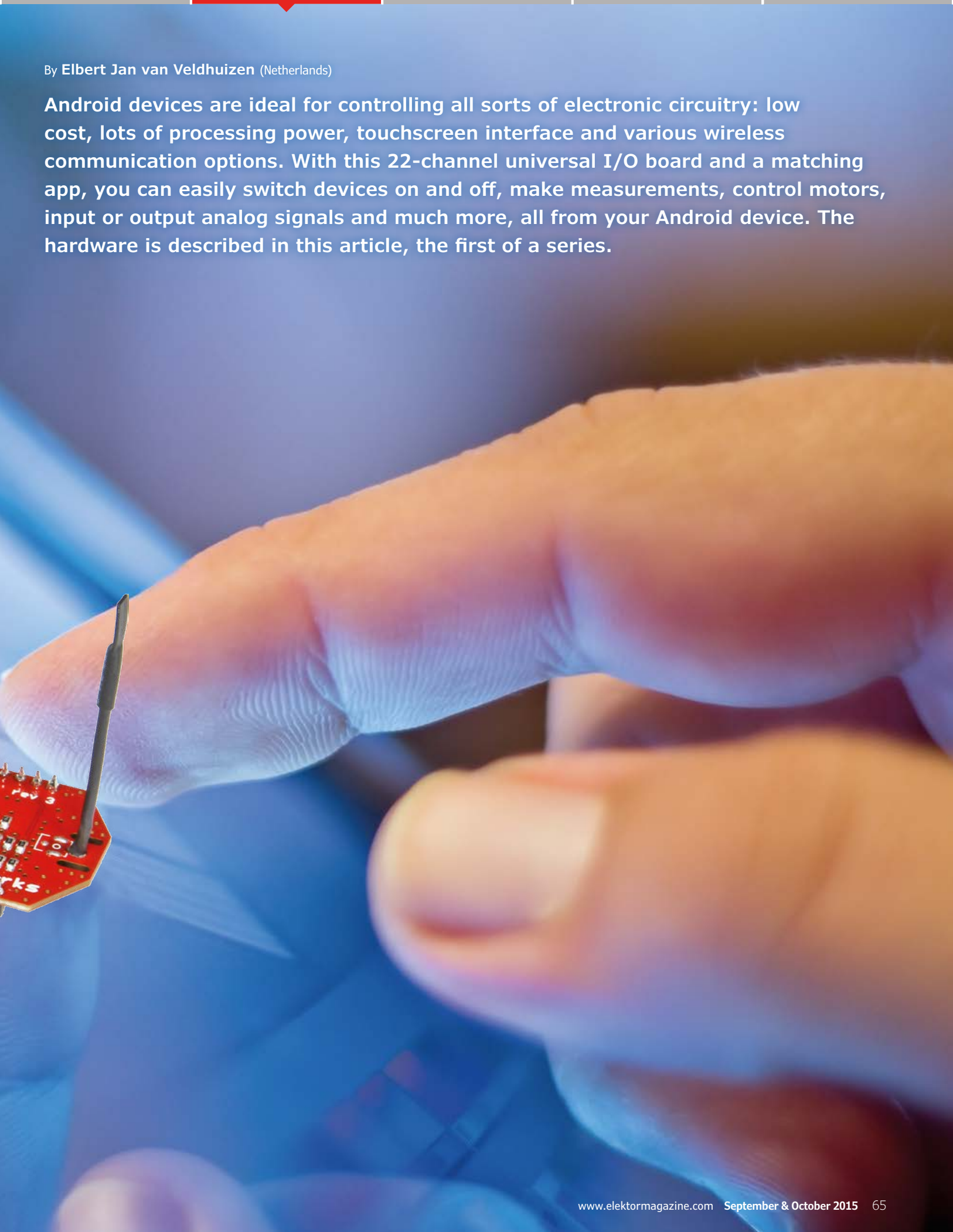
Control embedded electronics from your Android phone or tablet



Note: The component numbering on the prototype in the photos does not match the number on the final schematic drawing and the PCB layout.

By **Elbert Jan van Veldhuizen** (Netherlands)

Android devices are ideal for controlling all sorts of electronic circuitry: low cost, lots of processing power, touchscreen interface and various wireless communication options. With this 22-channel universal I/O board and a matching app, you can easily switch devices on and off, make measurements, control motors, input or output analog signals and much more, all from your Android device. The hardware is described in this article, the first of a series.



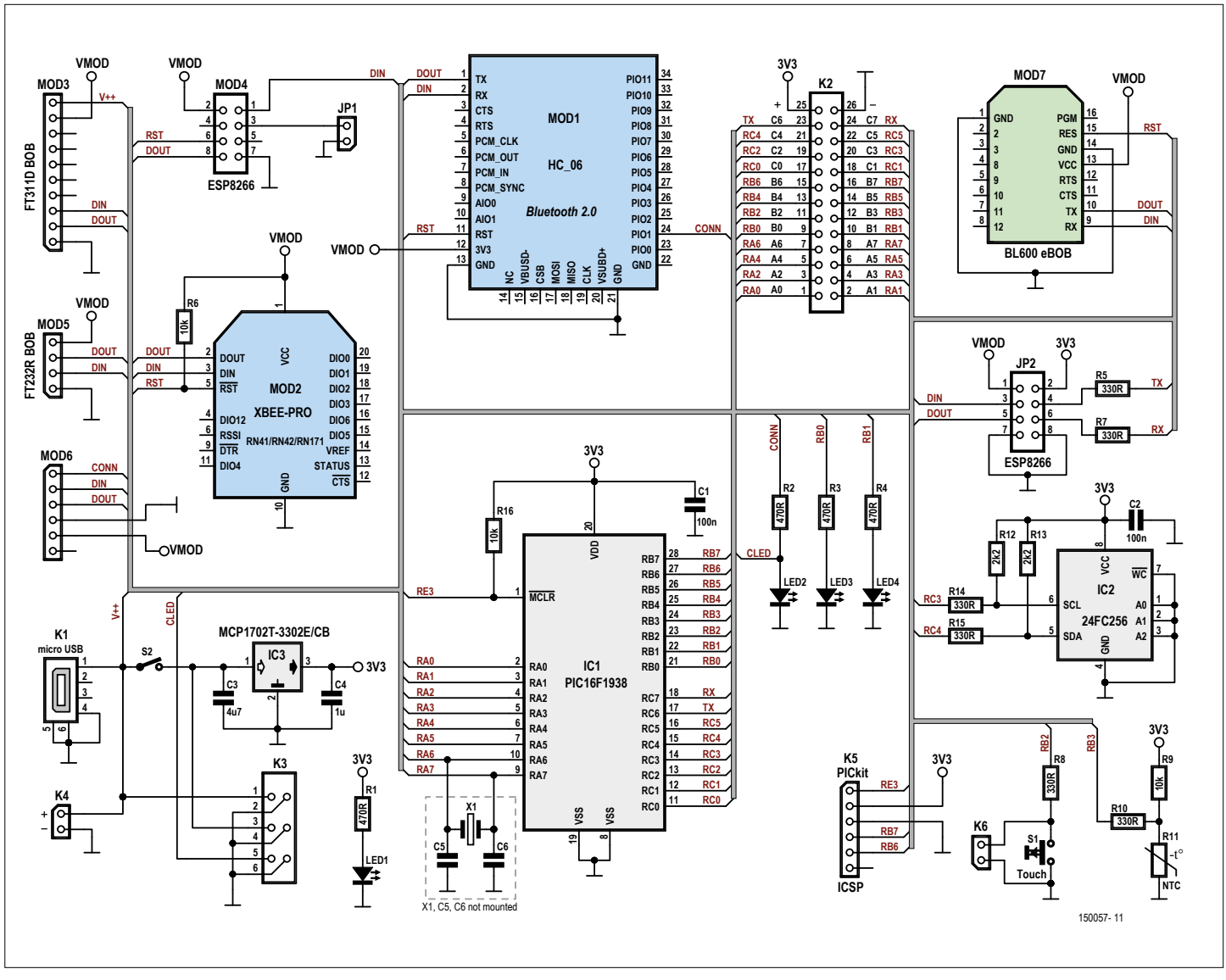


Figure 1. Schematic diagram of the I/O board. The actual number of components on the board is a lot less because only one communication module is mounted.

Table 1. Overview of the Android I/O board features.

Function	Pin count (max. 22) or value	Remark
Digital inputs	22	8 pins with optional weak pullup
Digital outputs	22	Maximum Iout 25 mA
ADC	11	Resolution 10 bits; approximately 20 kS/s
Cap-sense switch	8	7 pins, overlapping with the ADC pins
PWM output	8	Software generated; 4 x 2 in bridge mode
Counter	16-bit counter	8 MHz max.
Data rate	Max. 100 transactions per second	Connection 9,600 baud
Data flash memory (I2C)	32 Kbytes	Populated with 25FC256
On-board extras	4 LEDs, 1 NTC, 1 touch pad	
Supply voltage	4–13 V DC, unregulated 3.3 V DC, regulated	
Current consumption (without load)	30 mA max. (Bluetooth) 250 mA max. (Wi-Fi) 5 mA max. (USB), plus USB module current	
Supply voltage connection	Micro USB, PCB connector	

In this series of articles we describe a circuit board and a demo Android app that let you read and control 22 I/O pins using very simple commands. The board can communicate with an Android smartphone or tablet over Bluetooth, Wi-Fi or USB. The I/O pins on the board can be used not only for digital inputs and outputs, but also for analog to digital conversion, capacitive-sense buttons, analog PWM output and pulse counting.

The firmware of the on-board microcontroller can easily be modified to add functions for real-time applications. This is supported by a boot loader in the microcontroller. The author has developed Java classes (libraries) with consistent method naming for driving the Android I/O board. This enables users to start programming app functions right away without having to first delve into the details of the software.
The circuit board and the microcontroller functions are described in this first article. In the subsequent articles we will describe the Bluetooth, Wi-Fi and USB options and an app for controlling the Android I/O board. In addition, we will look at some data logger functions and show you how to modify the firmware and download it with the boot loader.

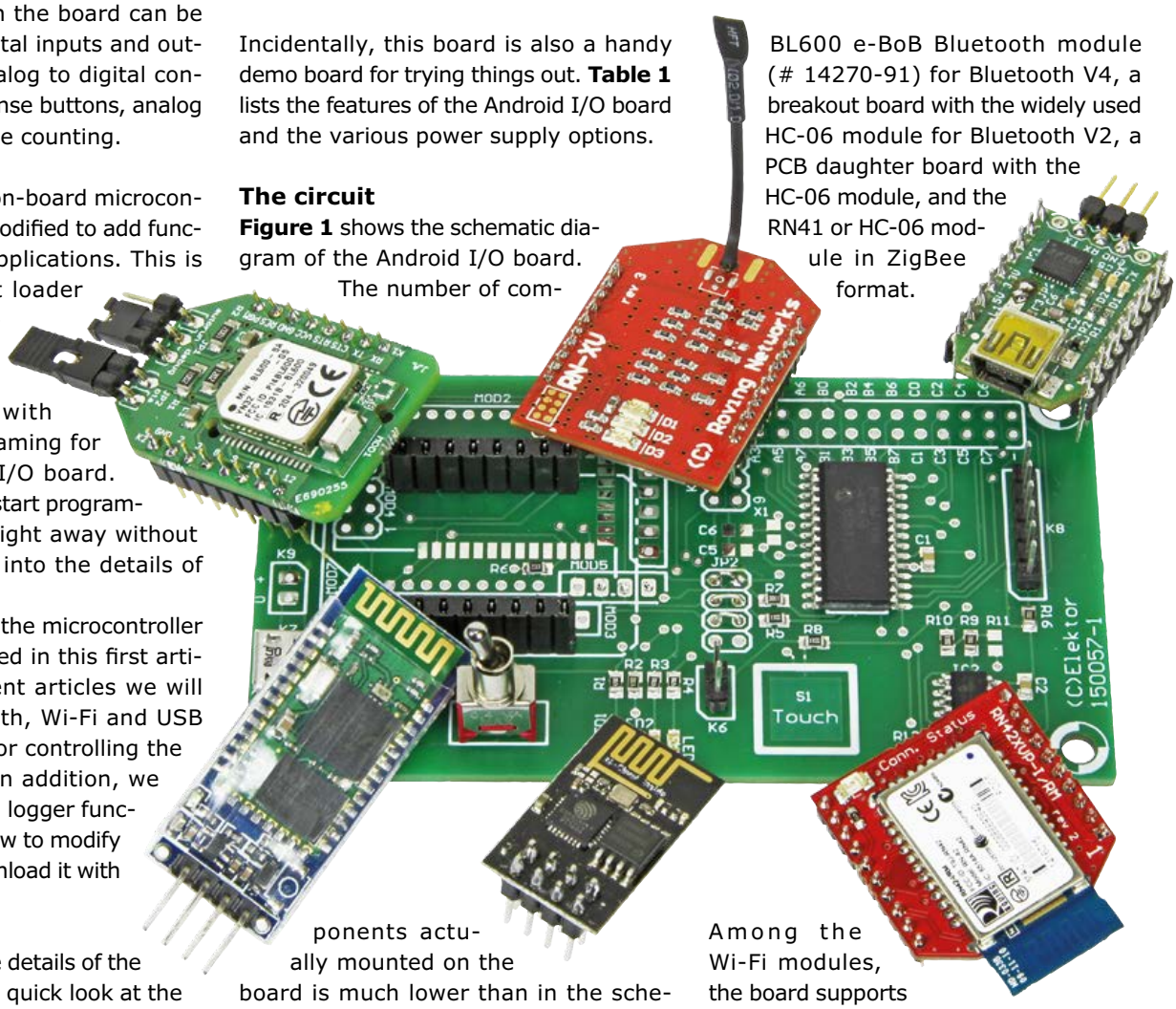
Before getting into the details of the hardware, let's take a quick look at the features. The Android I/O board has 22 I/O pins available as standard. It also has a capacitive sensing pad (etched on the board), four LEDs and an NTC thermistor.

> The board can contact an Android smartphone or tablet using Bluetooth, WiFi or USB.

Incidentally, this board is also a handy demo board for trying things out. **Table 1** lists the features of the Android I/O board and the various power supply options.

The circuit

Figure 1 shows the schematic diagram of the Android I/O board.
The number of components actually mounted on the board is much lower than in the schematic because the PCB is designed to hold various types of communication module. There is room for the most popular Bluetooth modules, in particular the Elektor BL600 e-BoB Bluetooth module (# 14270-91) for Bluetooth V4, a breakout board with the widely used HC-06 module for Bluetooth V2, a PCB daughter board with the HC-06 module, and the RN41 or HC-06 module in ZigBee format.



Among the Wi-Fi modules, the board supports the RN171-VX (ZigBee format) and the ESP8266 (PCB daughter board). The board also has connectors for the Elektor Android breakout board

Component List

Resistors

R1,R2,R3,R4 = 470Ω 1%, SMD 0805
R5,R7,R8,R10,R14,R15 = 330Ω 1%, SMD 0805
R6,R9,R16 = 10kΩ 1%, SMD 0805
R11 = 10kΩ NTC, SMD 0805 (e.g. Vishay NTCS0805E3103JHT)
R12,R13 = 2.2kΩ 1%, SMD 0805

Capacitors

C1,C2 = 100nF 25 V, 10%, SMD 0805
C3 = 4.7µF 25V, 10%, SMD 0805
C4 = 1µF 25V, 10%, SMD 0805

Semiconductors

LED1,LED2,LED3,LED4 = low-current LED, green, SMD 0805
IC1 = PIC16F1938-I/SO, SOIC28, programmed, Elektor Store # 150057-41
IC2 = 24FC256-I/SN, SOIC8
IC3 = MCP1702T-3302E/CB, SOT23A

Miscellaneous

K1 = micro-USB 2.0 connector type B, SMD
K2 = 26-pin (2x13) pinheader, 0.1'' pitch
K3 = 6-pin (2x3) pinheader, 0.1'' pitch
K4 = 2-pin pinheader, 0.1'' pitch
K5 = 6-pin pinheader, 0.1'' pitch
S2 = toggle switch, 0.1'' pitch
PCB # 150057-1
or
PCB with SMD parts preassembled: Elektor Store # 150057-91

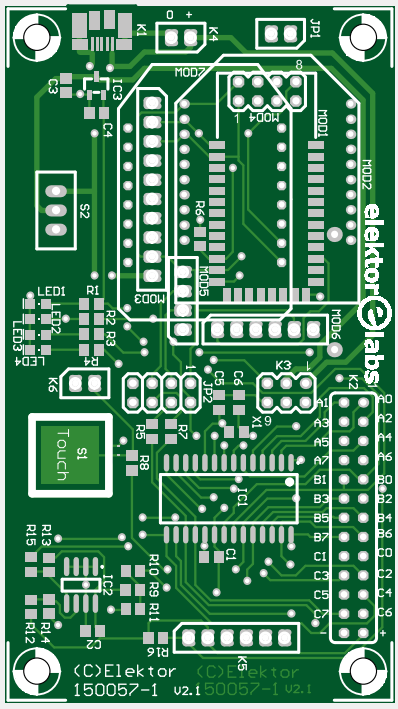


Figure 2. The PCB for the I/O board. The board has room for mounting seven different module types.

Connection Module (fit one only):

Mod1 = HC-06/-05 Bluetooth breakout module
- soldered straight onto board

Mod2 = HC-06/-05 Bluetooth ZigBee-shaped module
Mod2 = RN41XVC Bluetooth module
Mod2 = RN171 WLAN module (with wire antenna)
- mount onto two 10-way pinheader receptacles, 2mm pitch

Mod3 = Android Breakout-Board (Elektor Store # 130516-91)
- mount onto two 10-way pinheader receptacles, 0.1'' pitch

Mod4 = ESP8266 WiFi module (2x4-pin connection)
- mount onto 2x3-way pinheader receptacles, 0.1'' pitch

Mod5 = FT232R USB/Serial Bridge BoB (Elektor Store # 110553-91)
- mount onto 3-way pinheader receptacle, 2.5mm pitch (3-pin pinheader on BoB)

Mod6 = HC-06 module (6-pin version)
- mount onto 6-pin pinheader receptacle, 0.1'' pitch

Mod7 = BL600 E-BoB (Elektor Store # 140270-91)
- mount onto two 8-way pinheader receptacles, 0.1'' pitch

Table 2. Functions of all headers and connectors.

K1	Micro USB connector for power to board and module
K2	2x13-pin header for all 24 I/O lines of the PIC microcontroller, 3.3 V and ground
K3	2x3-pin connector for daisy-chaining several I/O boards
K4	2-pin supply voltage connector for board (4–13 V)
K5	6-pin programming connector for PICkit 2 or PICkit 3
K6	2-pin header for launching the boot loader by touch switch or jumper
JP1	2-pin header for modifying the firmware of the EPS8266 module
JP2	2x4-pin header for configuring the signal and supply lines to the other modules
MOD1 to MOD7	Connectors for mounting various Bluetooth, Wi-Fi and USB modules (see components list)

(# 130516-91) and the Elektor FT232R USB/serial Bridge BoB (# 110553-91), both of which provide a USB link to a PC, tablet or smartphone through a UST to OTC adapter cable.

All of these modules output the data received from the Android device or PC as serial data. This serial data goes to a PIC microcontroller, namely a PIC16F1938 with 24 I/O pins. As will be described later, the firmware in the microcontroller interprets the commands and configures the I/O pins as inputs or outputs for digital or analog signals.

All I/O pins of the microcontroller are accessible on a two-row pin header. Two of these pins are used for serial data transfer to the communication module.

That completes the general description; now let's look at the components in the schematic diagram. To start with, the board has a 3.3-V voltage regulator (IC3). There are two options on the board for connecting a power source to the voltage regulator: a micro USB connector (K1) and a pair of solder pads, which can optionally be fitted with a pin header (K4). IC3 is an MCP1702-33 low-drop voltage regulator. It provides the 3.3 V supply voltage for the Bluetooth or Wi-Fi module and the rest of the circuit. Power can be switched with switch S2. If you do not need a power switch, you can bridge the two mounting holes on the board with a piece of wire.

The microcontroller is a 28-pin PIC16F1938 from Microchip (IC1). The link between the PIC16F1938 and the Bluetooth, Wi-Fi or USB module runs through jumper station JP2 (for debugging) and the protection resistors R5 and R7. All I/O lines are available on connector K2.

For extra storage capacity (for use as a data logger, for example), the board has a 32 Kbyte serial I^2C flash memory (IC2). The I^2C pins of this IC have protection resistors (R14 and R15) and pullup resistors (R12 and R13). There is also an NTC thermistor (R11) on board for temperature measurement. It forms a voltage divider in combination with R9. The LEDs LED1 to LED4 are provided as indicators for various signals. LED1 is lit when the 3.3 V supply voltage is present, and LED2 is only active when the HC-06 breakout module is mounted. LED3 and LED4 can be used for various purposes; LED3 also indicates that the boot loader is active. S1 is the capacitive sense switch,

which consists of a pad etched on the PCB. Header K6, which is necessary for activating the boot loader in the PIC, is wired in parallel with S1.

The board also allows a clock crystal to be connected to the microcontroller. The boot loader software described in this series of articles does not support this option, which requires a different boot loader and the appropriate configuration word settings. In that case pins A6 and A7 on K2 can no longer be used by the firmware.

Universal PCB

At first glance the component overlay for this circuit board (**Figure 2**) appears rather cluttered due to all the module mounting positions marked on the board, but on closer examination you will quickly see exactly where the connections for modules MOD1 to MOD6 are located. After you decide which module you want to use, you only have to mount that module and the other positions remain unused. Except for the connectors and the switch, all of the components are SMDs, which means that manual assembly is not especially easy. That is why we offer a pre-assembled board (# 150075-91 [1]) with all of the SMDs and some of the connectors already mounted. Of course, you can also order a bare PCB and solder all the components on it yourself. However, we recommend that you wait with this until next month, when we will describe the various.

There are several pins and headers on the board for configuring all sorts of settings. For the sake of clarity, **Table 2** provides an overview of all the connectors and headers along with brief descriptions.

In normal operation, JP2 passes the Tx and Rx signals between the installed module and the PIC microcontroller. Usually pin 3 is connected to pin 4 and pin 5 is connected to pin 6. Pins 7 and 8 are joined together by a PCB track, which is handy when an external module is connected to JP2. The 3.3 V supply voltage from the voltage regulator is available on pin 2, and pin 1 is connected to the supply voltage line for the module. There are two options for supplying power to a module:

The module has to be supplied with power from the 3.3 V voltage regulator. This applies to the Bluetooth and Wi-Fi modules. For this purpose, pins 1 and 2 must be connected together.

If an FT232R USB/serial bridge BoB

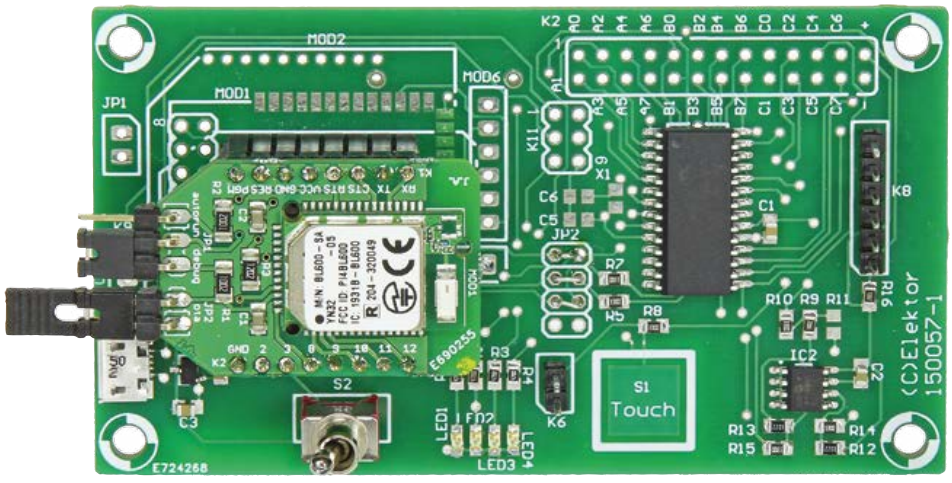


Figure 3. The assembled board fitted with the Elektor BL600 e-BoB.

Table 3. I/O pin functions and function numbers.

Pin	High-impedance input (0)	Weak pullup input (1)	Output (2)	ADC (3)	Cap sense (4)	PWM (5)	Speciaal (6)
A0	X		X	X			
A1	X		X	X			
A2	X		X	X			
A3	X		X	X			
A4	X		X		X		
A5	X		X	X	X		
A6	X1		X			X	
A7	X1		X			X	PWM bridge with A6
B0	X	X	X	X	X	X	
B1	X	X	X	X	X	X	PWM bridge with B0
B2	X	X	X	X	X		
B3	X	X	X	X	X		
B4	X1	X1	X	X	X		
B5	X1	X1	X	X	X		
B6	X	X	X			X	
B7	X	X	X			X	PWM bridge with B6
C0	X1		X				Counter
C1	X1		X			X	
C2	X1		X			X	PWM bridge with C1
C3	X1						I2C: SCL
C4	X						I2C: SDA
C5	X		X				
C6	TX (to Bluetooth, WiFi or USB module)						
C7	RX (from Bluetooth, WiFi or USB module)						
S							Data logger
X							Real Time Clock
Y							Version / Serial Number
Z							Global settings

[1] A new value can be sent

(110553-91) or an Android breakout board (# 130516-91) is mounted, the BoB is powered by 5 V from its USB connection. In that case pins 1 and 2 must never be connected together.

Commands

Even though you can't work hands-on with the I/O board just yet, let's look at the commands you can use to control the board.

Here we assume that a link to the board has already been established over Bluetooth, Wi-Fi or USB. That means there is a transparent serial connection between the Android device and the microcontroller on the Android I/O board. Commands can therefore be sent to the microcontroller from an app. The firmware interprets these commands and controls the I/O pins accordingly. The names of these pins match the microcontroller designations: A0 to A7, B0 to B7 and C0 to C7 (see also **Table 3**).

There are four types of command:

- **W** for "write", which writes a specific value to a specific pin. The format is W <pin> <value>. For example, "w a3 1" sets I/O pin A3 high.
- **R** for "read", which reads the value of a specific pin. The format is R <pin>. For example, "r b4" reads the value of pin B4. The format of the microcontroller response is R <pin> <value> (in this example "R B4 1").
- **S** for "settings". This defines the function of a pin. The format is S <pin> <function>, where <function> is a number. Here "0" makes the pin a high-impedance input, "1" makes it a weak pullup input, "2" makes it an output, "3" makes it an ADC input, "4" makes it a capacitive sensing input, "5" makes it a PWM output, and "6" is reserved for a special functions. For example, "s b3 3" makes pin B3 an ADC input.
  Several other parameters can also be set. Each pin has three additional 8-bit status registers that can be accessed by putting "1", "2" or "3" after the pin name. For example, the command "s b32 8" sets the status 2 register of pin B3 to a value of 8. The previously described pin functions are defined by the status 0 register, so "s b3 3" is the same as "s b30 3".
- **G** for "get". This can be used to read a setting or value. The format is G <pin>. The microcontroller responds with G <pin> <value>. For example, "g b3" yields "G B30 3".

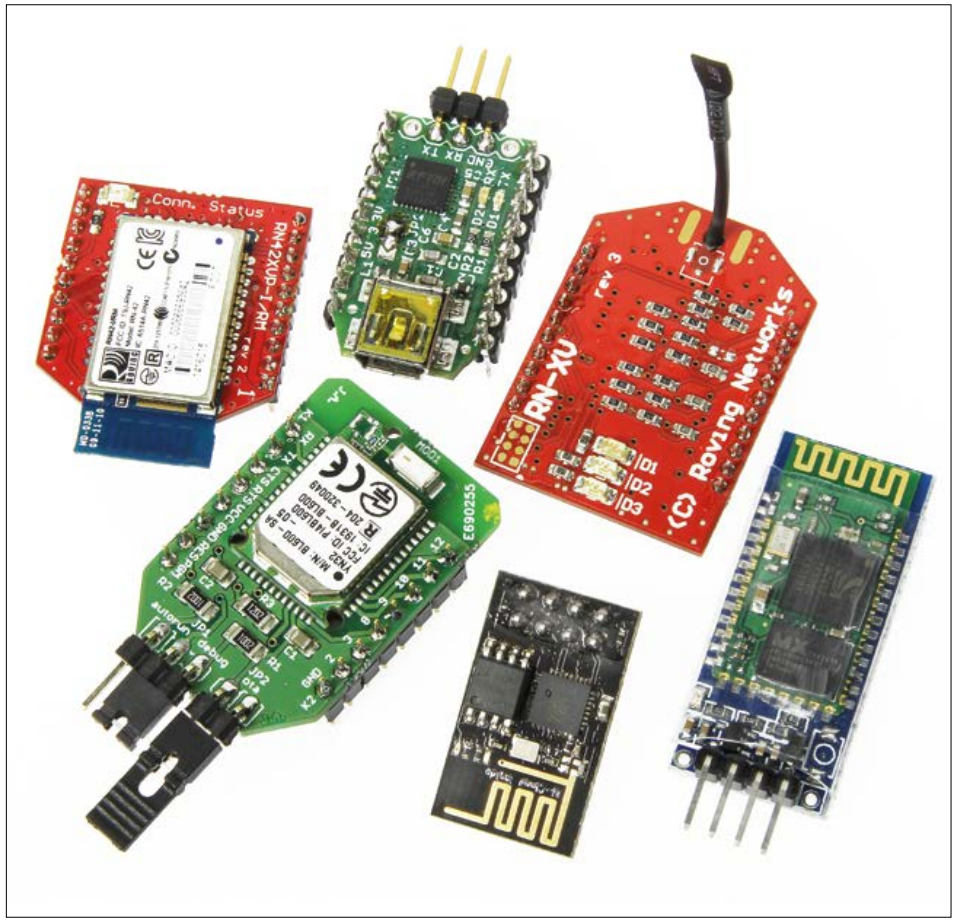


Figure 4. Some of the modules that can be mounted on the board.

The commands and pin names are not case sensitive, but the response from the microcontroller is always upper case. The microcontroller echoes the input in debug mode, so it is handy to use lower case for the commands. That way it is clear whether the text comes from the Android device (lower case) or the Android I/O board (upper case).

All of the functions are described in detail in a separate document. This document (150057-W) can be downloaded for free from [1].

In part 2 we will talk about the Bluetooth, Wi-Fi and USB modules and look at an app that uses the Android I/O board to expose and etch PCBs. We will also describe the standard classes that make it easy for you to control the Android I/O board from your own app. ⏮

(150057-I)

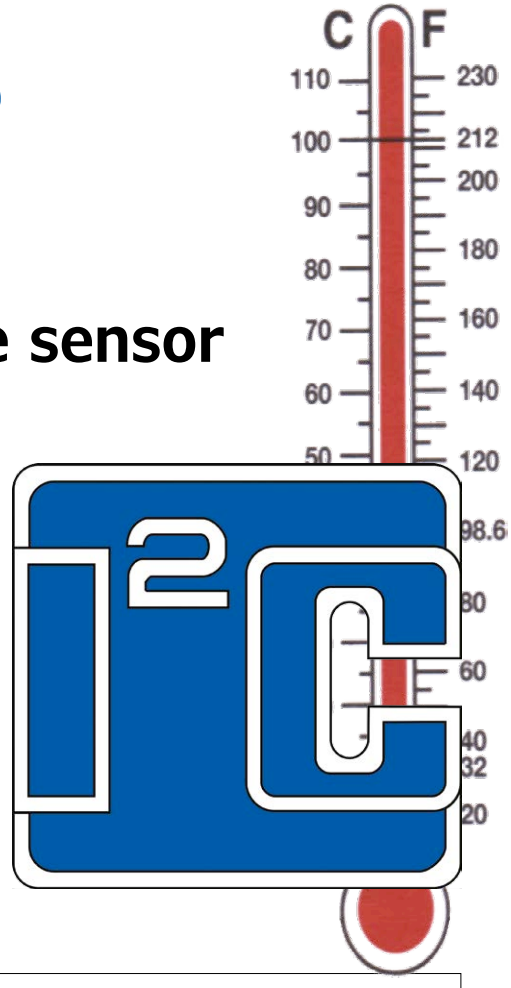
Web Links

[1] www.elektormagazine.com/150057

BL600 e-BoB

Part 4

The I²C port and the temperature sensor



By **Jennifer Aubinais** (France), elektor@aubinais.net

Our little wireless communications module has only seven input/output ports, but its I²C port gives it considerable possibilities for extension. We can use it for communication with various components such as temperature sensors, D-A and A-D converters, etc. Here we will use it for exchanging data, the new service Health Thermometer via Bluetooth Low Energy instead of the UART service used in previous installments.

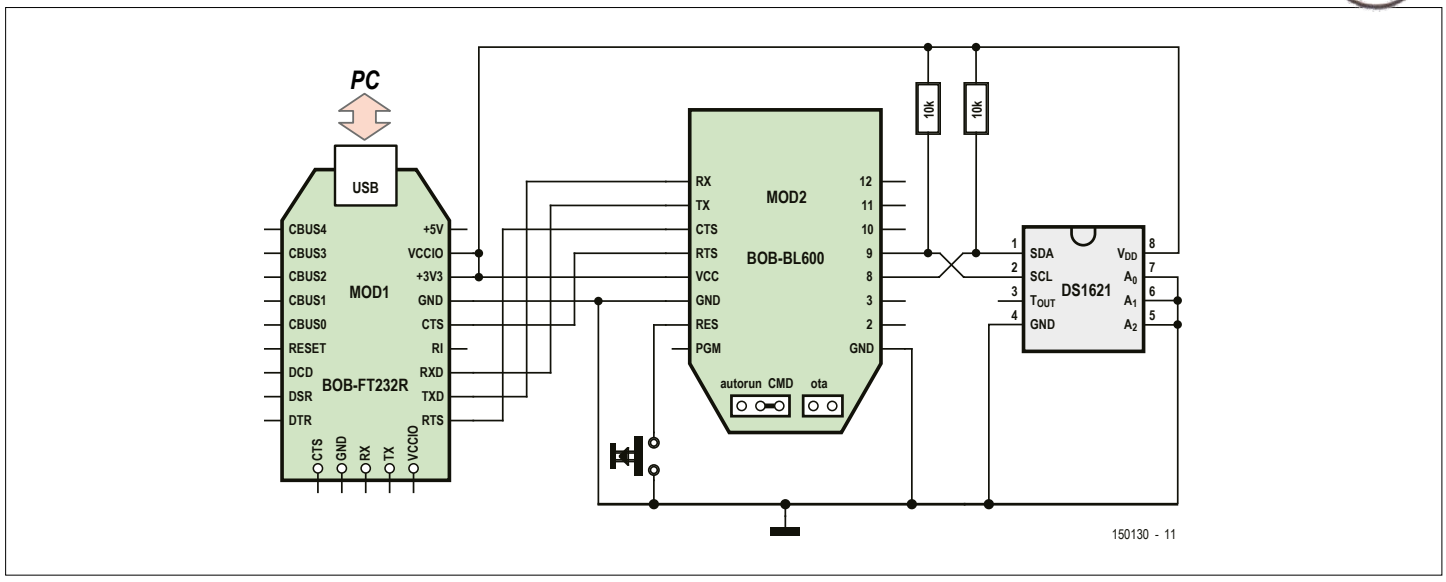


Figure 1. Schematic of a wireless thermometer with a DS1621 temperature sensor which communicates with the BL600 by I²C. The implementation of the e-BoB FT232 has been described in previous installments.

We have learned how to use events in smartBASIC in the previous issue. Now we can start with the I²C port of the BL600 by connecting it with the DS1621 temperature sensor [1] from Maxim Integrated.

It may seem surprising to communicate via I²C with a Bluetooth module… well, "you ain't seen nothin' yet", because this module has several more amazing resources, which will be the subject of future articles. For the moment, we will measure the temperature and display it on a smartphone, either under Android in an Android application which may be downloaded from Google Play [2], or under iOS with an application which may be downloaded from the Apple Store [7] or a small program which can be downloaded from the Elektor site [6].

Listing 1

```
DIM rc, handle, txt$
rc=I2cOpen(100000,0,handle)
IF rc!= 0 THEN
  SPRINT #txt$,INTEGER.H'rc
  DbgMsg("Failed to open I2C interface with error
code 0x" + Right$(txt$,4))
  PRINT "\nDO RESET"
  STOP
ELSE
  DbgMsgVal("\nI2C open success \nHandle is
",handle)
ENDIF
```

Listing 2

```
DIM x, nSlaveAddr, nRegAddr, nRegVal, txt2$
nSlaveAddr = 0x48 :  nRegAddr = 0xAC : nRegVal =
  0xAA
rc = I2cWriteReg8(nSlaveAddr, nRegAddr, nRegVal)
IF rc!= 0 THEN
  SPRINT #txt$,INTEGER.H'rc
  DbgMsg("Failed to Write to slave/register " +
  Right$(txt$,4))
ELSE
  SPRINT #txt$,INTEGER.H'nRegVal
  SPRINT #txt2$,INTEGER.H'nRegAddr
  DbgMsg("0x" + Right$(txt$,2) + " written
  successfully to register 0x" + Right$(txt2$,2))
ENDIF
```

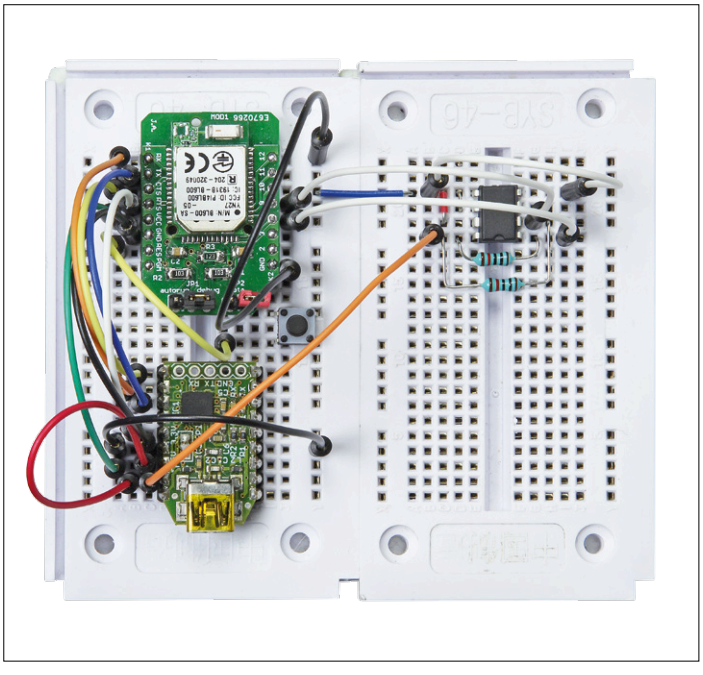


Figure 2. Photo of the whole assembly on a breadboard.

Note: the use of the tools and commands to which I refer here is covered in previous instalments of this series [3] to which the reader may refer.

I²C port and temperature sensor

The DS1621 temperature sensor is an ideal illustration of the read and write functions of the I²C port: the port SDA (data) on pin 8 of the e-BoB BL600 is connected to pin 1 of the DS1621; the port SCK (clock) on pin 9 of the e-BoB is connected to pin 2 of the DS1621 (**Figure 1**). Laird Technologies confirm that pullup resistors (between 4.7 and 10 kΩ) must be used on the I²C bus lines. Following the documentation of the DS1621, we will send it some commands and then read its different registers in order to calculate the temperature. The application Temperature for Android issued by Laird Technologies allows us to trace a graph of the temperature measured by the sensor. The application which we give you for use with iOS, which uses the service Health Thermometer, allows us to do the same thing on an Apple product.

a. Open I²C

The first step is the opening of the I²C port, followed by the processing of the returned code by a simple *if* statement (**Listing 1**).

```
Rc = I2COpen(100000,0,handle)
```

- nClockHz = 100000; clock frequency
- nCfgFlags = 0; must be set to 0
- nHandle = if the returned code is 0, this variable is further used to read, write and close the I²C interface

rc = the returned code must be 0 to continue. If not, we convert its value to a chain of characters using the SPRINT function with the INTEGER.H option to convert to hexadecimal, then display the error code (see doc. Laird Technologies), before stopping the program using STOP.

b. Write a byte to I²C

The next step is to write a byte to the I²C port, to be sent to the DS1621. We have chosen the Access Config register (0xAC), which is read/write, and as an example we will write to it the value 0xAA (**Listing 2**).

```
Rc = I2CWriteReg8(nSlaveAddr,nRedAddr,nRegVal)
```

- nSlaveAddr = slave address of the component, between 0 and 127
- nRedAddr = register address of the component
- nRedVal = value (8 bits) to write to the address of the desired register

Component List

Resistors
R1,R2 = 10kΩ

Semiconductors
IC1 = DS1621

Miscellaneous
K1 = pushbutton connection
MOD1 = FT232e-BoB, assembled, # 110553-91 (www.elektor.com)
MOD2 = BL600e-BoB, assembled, # 140270-91 (www.elektor.com)

Listing 3

```
nSlaveAddr = 0x48 :  nRegAddr = 0xAA
rc = I2cReadReg8(nSlaveAddr, nRegAddr, nRegVal)
IF rc!= 0 THEN
  SPRINT #txt$,INTEGER.H'rc
  DbgMsg("Failed to Read from slave/register " +
  Right$(txt$,4))
ELSE
  SPRINT #txt$,INTEGER.H'nRegVal
  SPRINT #txt2$,INTEGER.H'nRegAddr
  DbgMsg("Value read from register 0x" +
  Right$(txt2$,2) + " is 0x" + Right$(txt$,2))
ENDIF
```

rc = the returned code must be 0 to continue. If not, we convert its value to a chain of characters, then display the error code (see Laird Technologies' documentation.)

c. Read a byte from I²C

We read the register 0xAC which is the register of the high byte of the temperature of the DS1621. As we have not done the initialization procedure of the DS1621, it will not contain a usable value (**Listing 3**).

```
Rc = I2CReadReg8(nSlaveAddr,nRedAddr,nRegVal)
```

- `nSlaveAddr` = slave address of the component, between 0 and 127
- `nRedAddr` = register address of the component
- `nRedVal` = value (8 bits) read from the address of the desired register

rc = the returned code must be 0 to continue. If not, we convert its value to a chain of characters, then display the error code (see doc. Laird Technologies)

d. Close I²C

The last step is to close the I²C port. Laird Technologies recommend doing it twice (**Listing 4**).

```
I2Close(handle)
```

- **nHandle** = value created by `I2cOpen`

The file `demoI2C.sb` may be downloaded from the Elektor site [6].

e. Read the DS1621 using the serial port of the e-BoB FT232

We will not discuss the DS1621 [1] in detail, but we have made available for you a complete program for using it, notably using the functions `I2COpen`, `I2CWriteReg8`, `I2CReadReg8` and `I2CClose` described above. The datasheet of the temperature sensor suggests the following equation to transform the measured data of the DS1621 to a temperature value:

$$TEMPERATURE = TEMP\_READ - 0.25 + \frac{(COUNT\_PER\_C - COUNT\_REMAIN)}{COUNT\_PER\_C}$$

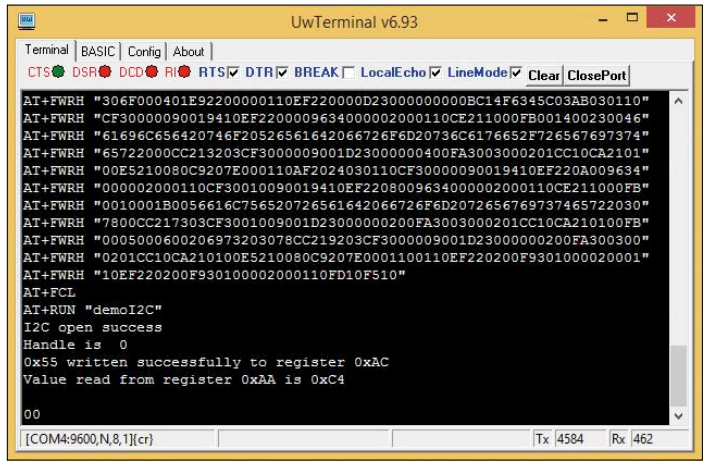


Figure 3. Writing to, and then reading from, the I²C port passing correctly.

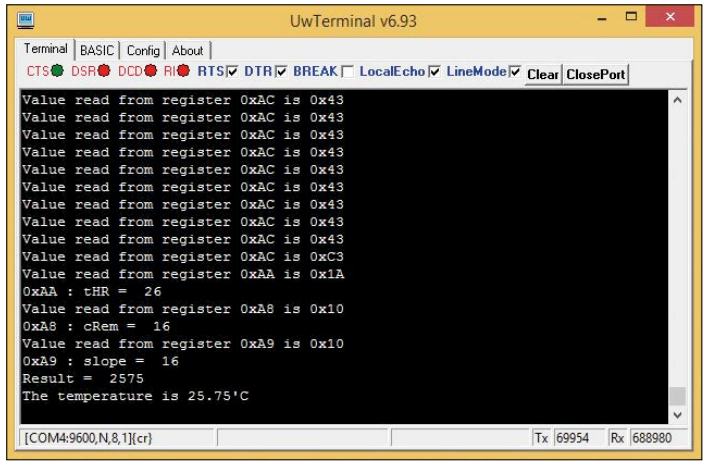


Figure 4. Display of the temperature in degrees: after bit 7 of the configuration register 0xAC goes to 1 (0x43 -> 0xC3), we can read the temperature register and do the calculations.

Listing 4

```
I2cClose(handle)   //close the port
I2cClose(handle)   //no harm done doing it again
```

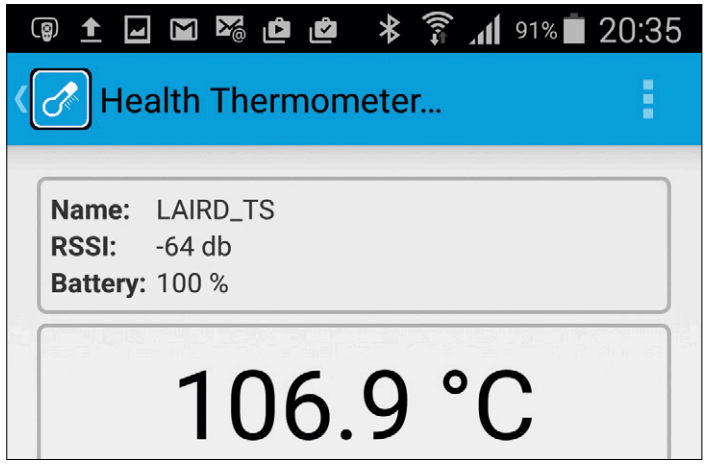


Figure 5. Don't panic, it wasn't 106.9 °C at the author's house or at Elektor Labs!

Listing 5

```
FUNCTION InitI2C()
DIM rc, txt$ // DIM Handle is defined at top of library
DIM rc, x, a, tC, flag, handle, txt$
  rc=I2cOpen(100000,0,handle)
  IF rc!= 0 THEN
    SPRINT #txt$,INTEGER.H'rc
    DbgMsg("Failed to open I2C interface with error
  code 0x" + Right$(txt$,4))
    PRINT "\nDO RESET"
    STOP
  ELSE
    DbgMsgVal("\nI2C open success \nHandle is
  ",handle)
  ENDIF
ENDFUNC rc
'-----------------------------------------------------
FUNCTION InitDS1621()
  DIM rc
  // Tout = active high; 1-shot mode
  DbgMsg("SetConfig")
  rc = SetConfig(POL | ONE_SHOT)
ENDFUNC rc
'-----------------------------------------------------
FUNCTION ConvertDS1621()
  DIM rc, tC
  DbgMsg("DS1621")
  DO
  DbgMsg("\nStartConversion")
  rc = StartConversion() // initiate conversion
  tC = GetHrTemp() // read high-resolution temperature
ENDFUNC tC
'-----------------------------------------------------
FUNCTION ConvertTempTxt$(tC)
  DIM rc, x, a, flag, txt$
  flag = 0
  IF (tC < 0) THEN
    tC = -tC   // fix for integer division
    flag = 1   // indicate negative
  ENDIF
  SPRINT #txt$,tC
  SELECT  StrLen(txt$)
    CASE 1
      txt$ = "0.0" + txt$
    CASE 2
      txt$ = "0." + txt$
    CASE 3
      txt$ = Left$(txt$,1) + "." + Right$(txt$,2)
    CASE 4
      txt$ = Left$(txt$,2) + "." + Right$(txt$,2)
    CASE ELSE
  ENDSELECT
  IF (flag == 1) THEN
    txt$ = "-" + txt$
  ENDIF
  PRINT "The temperature is "+ txt$ + "'C\n"
  for x = 0 to 40000
  next
  DOWHILE (1)
ENDFUNC txt$
'-----------------------------------------------------
FUNCTION CloseI2C()
  I2cClose(handle)  //close the port
  I2cClose(handle)  //no harm done doing it again
ENDFUNC 1
```

As the BL600 can only use integers (from 0 to 65535), all the values are multiplied by 100, after which the temperature will be in 100ths of a degree Celsius. This suits us well for displaying text in the form of a temperature in degrees: it is only necessary to correctly place the decimal point (**Figure 4**).

tHR = (tHR * 100 – 25) + ((slope – cRem) * 100 / slope)

The file DS1621.sb may be downloaded from the Elektor magazine website [6].

f. The DS1621 in HTS (Health Thermometer Service) mode

In my article *Bluetooth Low Energy Wireless Thermometer* published in the January & February 2015 edition of Elektor [4], it was the smartphone application which did the calculations. Here, it's the BL600 that does them. It's easy, thanks to Laird Technologies who offer software for our e-BoB BL600 and an application Temperature for Android. We thus have a wireless thermometer with the BL600 and the DS1621, which displays the temperature on your Android smartphone.

Use of the file from Laird Technologies

We simply compile the file htss.health.thermometer.sensor.sb. To verify that it works, we launch the application Temperature. You will notice that the temperature displayed is over 100 °C ! This is normal, as we have not connected an LM20 sensor on pin 4 of the BL600 as is expected with the development kit DVK-BL600-SA.

The file htss.health.thermometer.sensor.sb is in the software package you can download for the BL600 [5] (**Figure 5**)

Transformation of the program DS1621.sb to a library

First step: we copy our program DS1621.sb to DS1621.sblib. We are separating the main program (main) into several functions (**Listing 5**) which we will call in our new program.

Second step: we modify the program htss.health.thermometer.sensor.sb and rename it $autorun$.htss.ds1621.sb. Here are the modifications:

- Modify DEVICENAME: JA_HTS
- Add the library that we have just created: ds1621.sblib
- Delete the following functions: Adc2mv, Mv2Temperature
- In the function HandlerTimer0, replace the code in red by the code in green in **Listing 6**

Communicate via I²C with a Bluetooth module? You ain't seen nothin' yet!

Listing 6

```
mv = Adc2Mv(GpioRead(4))
DbgMsg("\nAdc mV=")
PRINT mv
tmp = Mv2Temperature(mv)
tmp = ConvertDS1621()
tmp = tmp / 10
```

Listing 7

```
InitTempSensor()
rc = InitI2C()
rc = ConvertDS1621()
TimerStart(0,TEMPERATURE_POLL_MS,1)
```

- In the main program (`main`) replace the code in red by the code in green in **Listing 7**

Nothing complicated; in place of the original analog LM20 sensor we have simply substituted our DS1621. For starters, it's I²C and further, it has a higher resolution.
The file `$autorun$.htss.ds1621.sb` may be downloaded from the Elektor website [6].

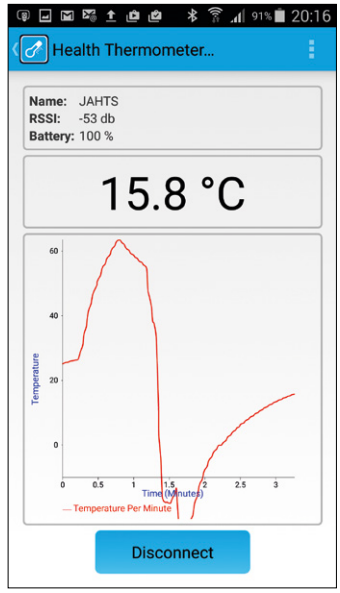


Figure 6. Graph obtained with the DS1621 connected via I²C to our e-BoB BL600. You will notice the peak of over 60 °C (using a hair-dryer) and a dip to –20 °C (obtained with a Freezit aerosol). The DS1621 offers a measurement range of –55 °C to +125 °C.

Figure 7. Screen capture of the BLE HTS application on iPhone.

The program under Android

We now use the application HTM (Health Thermometer Per Minute), from the Laird Toolkit that we have already downloaded. After the scan, we choose the DEVICENAME JA_HTS given to our e-BoB. A red graph line appears; the author amused herself well here — guess how? (answers in the caption of **Figure 6.**) Amazing, huh?

The iOS program

After this example of the use of the HTS service with Android, instead of the Bluetooth Low Energy module's UART service, used since the beginning of this series, we also offer licensed iOS developers the source code of an equivalent small program for Apple (**Figure 7**). This file `BLE HTS.zip` can be downloaded from the Elektor website [6]. On the website of Laird Technologies [5], you can find the complete source code for their application `Toolkit` for iOS comprising the services UART, HTS….

Call for contributions

In this series of articles, we have called upon the program Serial from the Toolkit from Laird Technologies. This has made our lives much easier, but isn't it now time to make your own application for your Android phone? From the next article we will offer you a source code, the simplest possible, which will allow you to increase your knowledge of the e-BoB BL600. Thanks to this module, available ready-to-use from the Elektor Store [8], implementation of Bluetooth communication is easy. Elektor is ready to publish other applications. If you have any ideas for using this module, or designs or projects in development, we invite you to share them with us, the author, and the Elektor community on the website www.elektor-labs.com. Your project might be chosen to appear in a forthcoming edition of Elektor magazine or the Elektor.POST newsletter.

(150130)

Web Links

[1] www.maximintegrated.com/en/products/analog/sensors-and-sensor-interface/DS1621.html
[2] https://play.google.com
[3] Elektor edition 2/2015 (March & April)
[4] Elektor edition 1/2015 (January & February)
[5] https://laird-ews-support.desk.com/?b_id=1945
[6] www.elektormagazine.com/150130
[7] https://itunes.apple.com/en/app/bl600/id594855763?mt=8
[8] www.elektor.com/bl600-e-bob-140270-91

Camelback Water Level Indicator

Lonely Planet approved 4 sure

By **Laurent Labbe** (France, China)

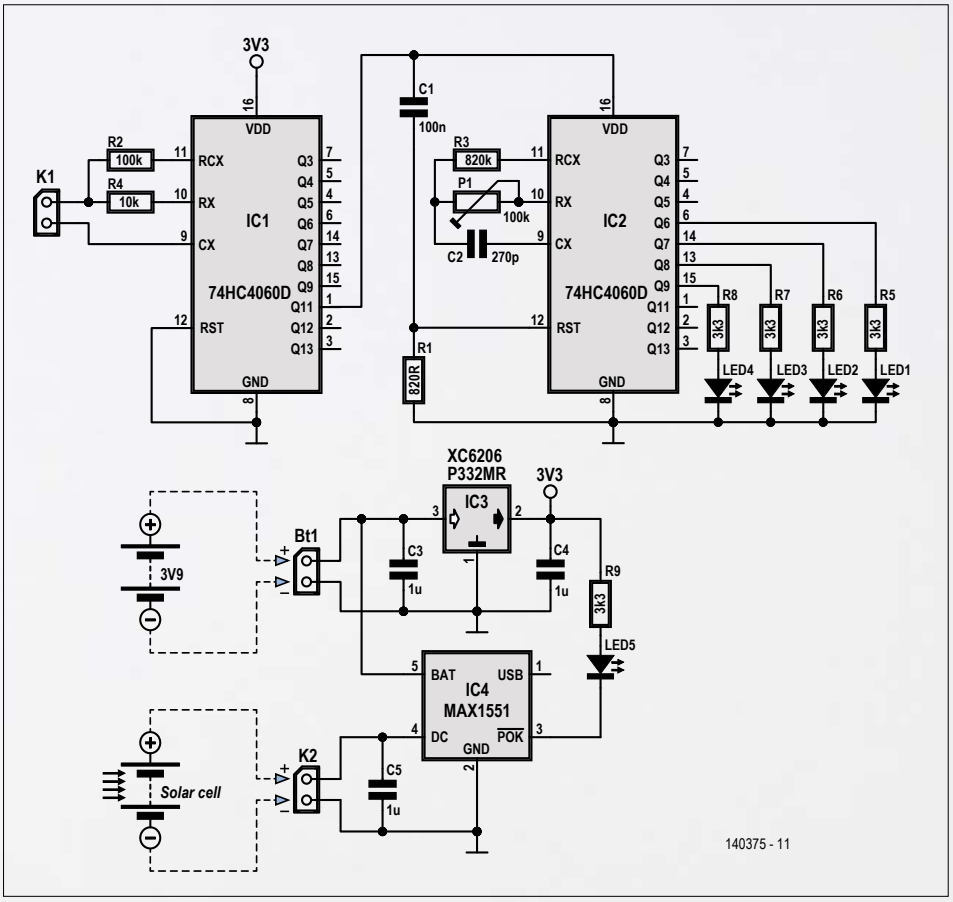
Easily outranking GPS, a credit card and fair weather, a supply of drinking water (H_2O) is the first, foremost and quintessential requirement of any backpacker or hiker keen on surviving in the jungle, desert, New York, or the outback. Hence the availability of 'camelback' rucksacks specially designed to contain a collapsible water bag with a tube and mouthpiece. Sadly with the water tank on your back it's not easy to tell how much there's left to drink. With our solar-charged level indicator though you're good to go *Into the Wild*, come rain or snow.

This project — originally called Camel Bag Water Indicator — went through the Full Monty of Elektor Labs, meaning it was posted there in the *Proposals* class where it was seen, discussed and rated by members. Next it got moved to *In Progress*, and finally to *Finished* meaning you can now read about it in Elektor magazine. During the *In Progress* stage the project got adopted by lab worker Luc Lemmens who produced an Elektor-style PCB for it and went through the rigors of making the project reproducible by you, readers of Elektor magazine. Luc worked closely together with project author Laurent Labbe, whose story on developing the project is rendered below.

Triggers and considerations

During many mountain bike tours in hot weather conditions in and around my domicile, I am used to carry a 'camelback' for my drinking water supply whilst on the move (the commercial product name is Camelbak, *Ed.*).

Figure 1. Schematic of the camelback (Camelbak) water level indicator. Note how, as a power-saving kluge, the first 4060 powers the second.



During these trips I was always frustrated to be guesstimating the quantity of water left in the bag. I spent a long time thinking about this issue during these trips. I imagined I could put a sensor inside the water bag, or some water debit sensor, but nothing satisfied me. Until I saw Paul Cordonnier's capacitive liquid detector in his 'Siphonic Rain Gauge' article in Elektor magazine, edition March & April 2015 [1]. Paul's idea to place the sensor outside the water vessel rather than inside it, really appealed to me. Great idea, Paul, thanks a lot, *merci bien*.

Not a fata morgana

I began by doing a few tests with a 74HC4060 oscillator/divider IC supplied at

Component List

Resistors
Default: 5%, 0.125W, 0805
R1 = 820Ω
R2 = 100kΩ
R3 = 820kΩ
R4 = 10kΩ
R5,R6,R7,R8,R9 = 3.3kΩ
P1 = 100kΩ trimpot, SMD

Capacitors
C1 = 100nF 5%, 50V, 0805
C2 = 270pF 5%, 50V, 0805
C3,C4,C5 = 1µF 10%, 16V, 0805

Semiconductors
LED1,LED2,LED3,LED4,LED5 = LED, green, low current, 0805
IC1,IC2 = 74HC4060 (SOIC-16)
IC3 = XC6206P332MR (SOT23-3)
IC4 = MAX1551

Miscellaneous
Solar panel 5V, 150mW
4.2V cellphone Li+ battery
PCB 140375-1 V1.0
Two adhesive copper strips, length 15-20cm

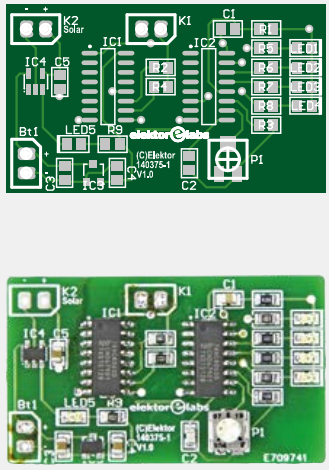


Figure 2. As we're talking portability here, every ounce and millimeter counts. Hence the PCB got designed to take SMD parts.

3.3 V (to allow it running off a lithium cell) and with a plastic bag, first filled with water and then emptied out. It worked: a variation of frequency could be observed.

To make it simpler, I used the 4060's Q11 output on pin 1, which gave a frequency of 150 Hz when empty, and 60 Hz when full.

At this point we can proceed to discuss the schematic in **Figure 1**. How to display the 'data' from the DIY capacitive sensor connected to K1? As the Q11 frequency is lower when the bag is full, the logic 1's are effectively longer. The idea arose to use this signal to launch another 4060 to count during this time and using its 128/256/512/1024 divider outputs to create a water level readout using LEDs. The RST (CLR) input on the 4060 is active-High, and to save precious current the 4060 reading the sensor (IC1) powers the 4060 doing the readout (IC2). With a pulse at power-on to clear the counters, this works if we choose the clock frequency just right at around 30 kHz.

Here, high-efficiency LEDs are used with 3.3-kΩ resistors (R5–R8) ensuring the readout is visible even in bright sunshine. The sensor consists of two parallel running adhesive strips of copper foil or copper laminate, secured vertically ☺ on the outside ☺ of the water container.

For calibration it is useful that the clock of the second 4060 be adjustable. Admittedly it is a bit tricky to find the right

Figure 3. The author's prototype of the camelback water supply, hinged to the front of his rucksack.

Figure 4. The water reservoir with the capacitive sensor secured to it. Oh yes, these reservoirs can be bought at hiking and outdoor outlets.

Figure 5. The DIY sensor works equally well when mounted at the other side of the reservoir.

adjustment on P1 between 'empty' detection and 'full'. The prototype worked though, and was accurate enough for my needs. I put the lot into a transparent box on my camelbak — very light and useful. At a later stage while contributing on the project at www.elektor-labs.com I thought of adding a solar power panel to recharge the lithium battery, and even later, to use a microcontroller to implement automatic calibration for 'empty' and 'full'. Of these ideas, the first was worked out, the second is 'under active consideration' at the time of writing this article.

The circuit is powered by a small (150-mAh) LiPo battery, which is being charged by a 5-V, 50-mA solar cell via a dedicated charger IC, the MAX1551.

Again using my prototype I found that the battery was charged from 3.3 V to 3.7 V after a few hours under sunshine. I'm thinking of adding another cell to increase the current. After a small trip (two hours biking) enjoying some sunshine, by the end of the afternoon I measured 3.72 V, meaning the battery was at same level as on departure. So, although conditions were not perfect, the battery did not discharge.

The overall consumption of the circuit is roughly 3 mA, hence long-term autonomy of operation should not be a problem. An external plug for a USB charger may be considered as a cool extension.

Construction

Small as the circuit board and the parts may be, it is a challenge, not a *fata morgana*, to assemble at home using manual soldering on the PCB design for the project (**Figure 2**). Luc Lemmens at Elektor Labs proved it using patience, pure skill, and the now well-published bag of tricks to get these tiny parts properly secured first and then soldered.

Good attention has to be paid to boxing up the camelback water level indicator, as well as the sensor construction, and for this it is useful to view the photos of the author's prototype in **Figures 3, 4 and 5**. Make sure everything is secure in and on the box and you're good to go. First hiker or pedaler to arrive at Elektor House [2] with a working CWLI and at an outside temperature exceeding 30 °C (86 °F) gets a free tour of the Elektor Labs, an Elektor Labs ruler, and a visit to the water well in the castle basement. Competition closes October 1, 2015. Competition not open to employees of Elektor International Media, it subsidiaries, licensees, and/or associated publishing houses.

(140375)

Web Link

[1] Siphonic Rain Gauge, www.elektormagazine.com/120554

[2] @ 51.016224, 5.838898; Mon-Thu 8 am – 5 pm, Fri 8 am – 1 pm. Ring doorbell.

Bike Inclinometer

Compact solution using a PIC

By Jelle Aarnoudse (NL)
BikeInclinoMeter@Ziggo.nl

Cycling enthusiasts who enjoy riding through hilly terrain (and this doesn't necessarily have to be a col of the first category) will often want to know how steep that slope is that they are attempting to climb. This is not a function that is normally found on a cycle computer, but with only a few components you can build such an inclinometer yourself.

Hello World… a small part of the Netherlands is neither flat nor below sea level. Cycling though the hills of South Limburg, I was often surprised how one incline would cost me much more energy (and sweat) than another. While my little cycle computer keeps track of all kinds of things, gradient is not one of them. I couldn't find anything appropriate in the bike shop and if you enjoy playing with

PIC microcontrollers and the like, then, well, you devise something yourself. This became the inclinometer, which for this occasion I named the 'bike inclinometer'. It is a simple thing, with a few clever features, that is housed in a very small plastic enclosure which is attached to the top tube of my bicycle using a Terry clip.

After some searching for a suitable sensor, I found a chip made by VTI Technologies (these days part of Murata), which generates an analog signal that is proportional to the angle of inclination (see [1] and the connection details in **Figure 1**). At the time, my PIC supplier also sold nice, small LCDs for €2, which were probably intended for mobile phones and could be controlled with I²C. These were ideal for my application. To complete the circuit a few other components were necessary to provide the PIC with a positive and the LCD with a negative power supply voltage. This finally resulted in the schematic of **Figure 2**.

The microcontroller

For those projects that do not require exact time measurements, the PIC16F88 controller is a fantastic, versatile workhorse: an internal oscillator (which avoids the need of a crystal), flash program memory for 4096 instructions, 386

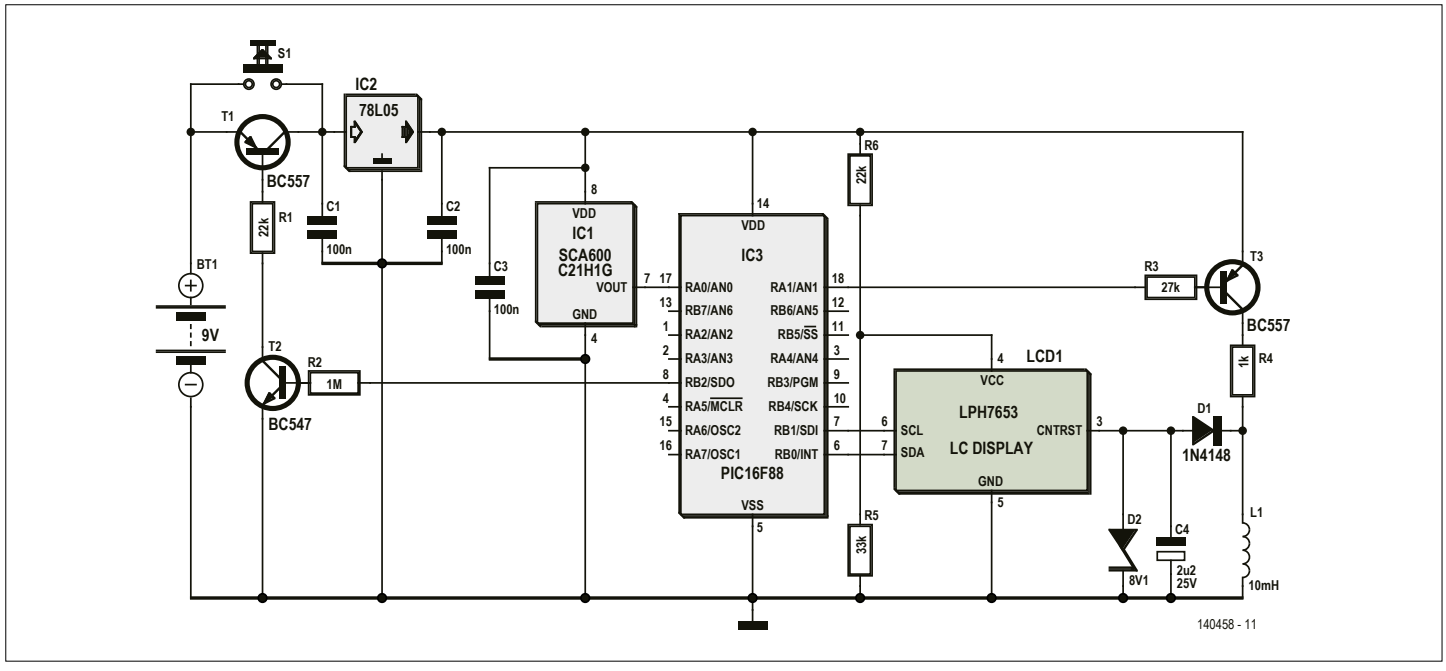


Figure 1. The schematic for the bike inclinometer.

The program for the inclinometer was written in assembly language

bytes of RAM, 256 bytes of data EEPROM, 10-bit ADC, PWM output, USART, two comparators, two 8-bit timers and one 16-bit timer.

The power supply

The PIC operates from a 5 V supply, while the LCD requires 3 V and –8 V. The first two are provided by a 9-V battery, a 78L05 and a voltage divider.

How to generate the –8 V supply for the LCD required a little bit more thought. Until I realized that the PIC, after running the inclination program, had plenty of time to spare to generate an additional clock signal, which could serve as the control signal for a simple switching power supply. This consists of T3, R3, R4, D1, L1, C4 and D2. When RA1 (pin 18) is low, T3 conducts and stores energy in L1. When RA1 is high again, T3 blocks and L1 reacts with a back-EMF. As a consequence, D1 starts to conduct and this charges C4, which results in a negative voltage which is nicely limited to –8 V by zener diode D2.

I always forget to turn off a gadget like this, so an auto-switch-off was definitely an essential feature. I solved this with T1 and T2. When S1 is pressed, the PIC starts. One of the first actions it carries out, is setting RB4 high, which turns on both T1 and T2 and so continues to provide power to the unit. The PIC program is configured in such a way that RB4 remains high, provided the output voltage from IC1 changes regularly. If this hasn't changed for more than 5% for about 20 minutes then the bike is apparently stationary. RB4 is then made low and the unit switches itself off.

The SCA600 C21H1G, an angle of inclination meter???

Initially I didn't question how such a chip manages to measure the angle of inclination. I only did that after I was surprised to find that the thing worked perfectly well when stationary, but the value on the display changed constantly while cycling, although the gradient didn't change. I noticed that the value was higher on the down-stroke of the pedal compared to when the pedals where about vertical and therefore little pedaling force was applied. This thing therefore does not measure inclination at all, but instead measures acceleration! I should have know this, of course, because at the top of the datasheet it states in big letters: accelerometer. The solution to this problem was simple: just average over a longer period of time. How this is done will be dealt with in the description of the program.

The program

The program for the inclinometer has been written in assembly language using MPLAB, since this allows the code to be kept as small and efficient as possible. The entire hex-code fits neatly in the 4 KB of available program memory. The assembly language and hex code files can be freely downloaded from [2].

The program has to carry out the following tasks: provide the negative power supply voltage for the LCD; read the

Web Links

[1] www.datasheetarchive.com/SCA600-C21H1G-datasheet.html

[2] www.elektor-magazine.com/140458

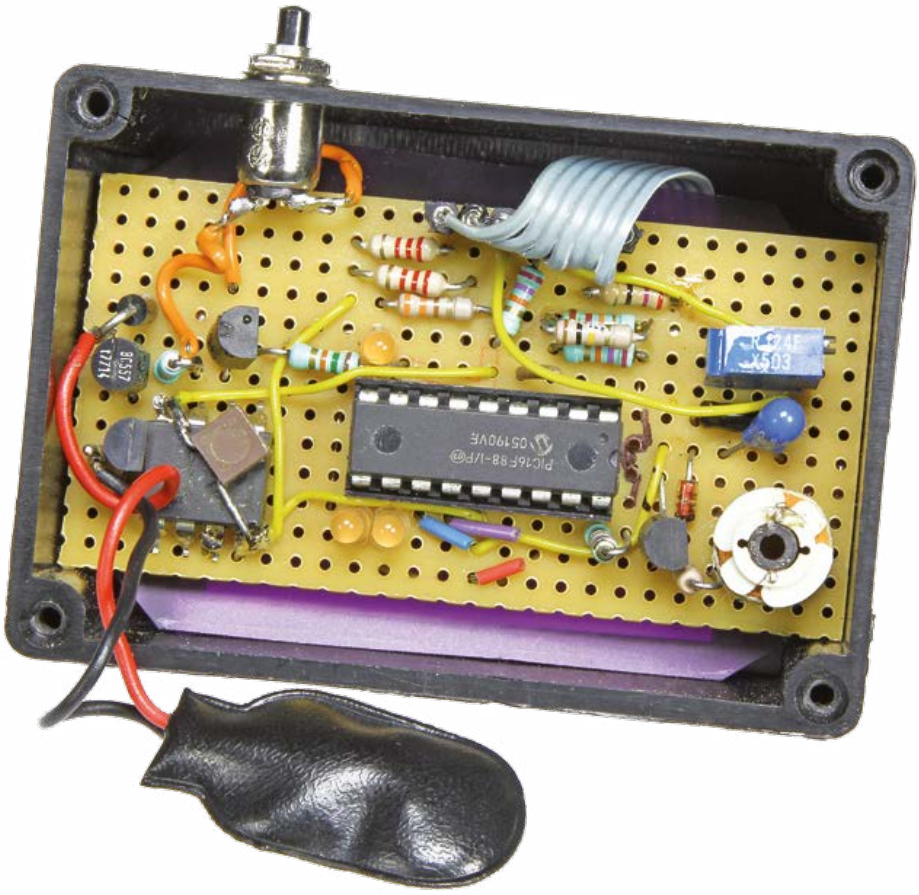


Figure 2. This is how the prototype fits together.

chips, which you can use to generate just about any desired voltage from two AA batteries. This is much more efficient than a linear voltage regulator. But, as a reassurance: In this circuit the 9-V battery easily lasts 2 years, because the circuit draws very little current.

The top tube of my bicycle, where I mounted the meter, is exactly horizontal. This is not true for every bicycle. For a next iteration of the circuit I would add another button, which would allow a zero-calibration to be carried out.

This is not a ready-to-build project, but more of a description of how I arrived here, the challenges I encountered and how I solved them. The display described here is no longer available and there are now better accelerometers. But this is a good starting point for similar projects and, if necessary, I'm certainly prepared to help readers with their implementations.

(140458)

accelerometer chip; calculate the moving average and display the result via the I²C protocol on the LCD.

To generate the negative LCD voltage requires a square wave. This is produced by Timer0, which generates in interrupt every 25 microseconds and which in turn toggles RA1.

We read the output from the accelerometer using the ADC in the PIC. The result of which is placed in a circular array of 64 bytes. After each update the average is determined and this is then sent to the display.

The characters that are to be shown on the display are stored as bitmaps in the program memory and are retrieved from there as required and sent to the display. Immediately after switching the unit on, an image of a bicycle is shown as a welcome message, which is also stored as a bitmap in the program.

In a few places in the program, the unused outputs of the PIC are made high or low. I had LEDs connected to these outputs during the testing phase of the software, so that I could check whether all the individual program parts operated correctly.

The construction

Because this is a simple, one-off circuit, I built everything on a small piece of prototyping board. This sits as a sandwich between the display and the 9-V battery. This all fits snugly in a small enclosure and is therefore held tightly in place.

Finally

If I was to build this thing again, then for the power supply I would use one of those modern switching power supply

Wireless I²C Sensor

Enhances Elektor's nixie tube thermometer/hygrometer

By Hans Oppermann (Germany)

The nixie tube thermometer/hygrometer featured in Elektor June 2012 combined components from the good old days of electronics with a modern microcontroller. To make the device even easier to use, this upgrade uses two wireless modules to connect the sensor and temperature display.

Time-honored nixie tubes and ultra-modern microcontroller technology were first brought together by Dieter Lues in Elektor for January 2011 in a thermometer. Hans Oppermann (that's me!) then refined this design to create a thermometer/hygrometer with improved characteristics. Both versions fascinated many Elektor readers and despite the unconventional mixture of components, people have reproduced them over and over.

All the same, this nixie thermometer/hygrometer device (from now on we'll call it NTH for short) suffered from one major shortcoming: where you position the circuit is not necessarily the place where you want to measure the temperature. If you put the sensor on the PCB directly next to the electronics, heat from the nixies and the decoder IC will falsify the result measured. Having forked out $20/£15/€17 already for the SHT21 sensor [1] alone, then you might be more interested in accurate measurement results than fabulous aesthetics! It's true that you could separate the sensor, with its I²C interface, from the rest of the circuit by several meters of cable (that's the most that an I²C bus can manage), but then the cable would degrade the visual impression again. No, it's better to stick with the wireless solution, using a radio connection alone.

Redesign for wireless

The most important prerequisite for this redesign was the freedom to re-use the sensor and existing assembled PCB without needing any modification to the software in the Microchip PIC16F876 controller. Sensor and controller alike should be unaware of (and unaffected by) the modification. As far as the controller on the nixie board is concerned, the inbound radio connection towards SHT21 (I²C slave) should act as a (virtual) sensor and in the outbound direction, towards the sensor, as a (virtual) controller (I2C master). The transmitter can then read data from the SHT21 sensor (transmitted by radio) to the receiver, which then decrypts the data and converts it back into I²C signals.

Given these requirements, would anybody consider any device other than the RFM12B module [2] from HopeRF? The module (**Figure 1**) is small, affordable, easy to handle and well known to most Elektor readers from other projects (and consequently available also from the Elektor Store [3]). The wireless module is a transceiver, which means that it works both as transmitter and as receiver operating in the 433 MHz ISM frequency band. The RFM12B has another feature that is very advantageous here, namely extremely low current consumption. The last thing to mention is that the transmitter and sensor combination is not a fixed device and therefore needs to manage a very long time using the energy from its CR2032 battery. Therefore at each end (transmitter and receiver) we employ extremely low-power PIC LF controllers. I was able to stretch the battery life to about eight months in practical operation by devising special settings for the status of the ports and programming the lon-

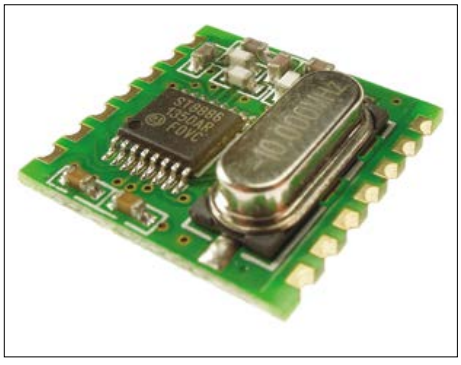


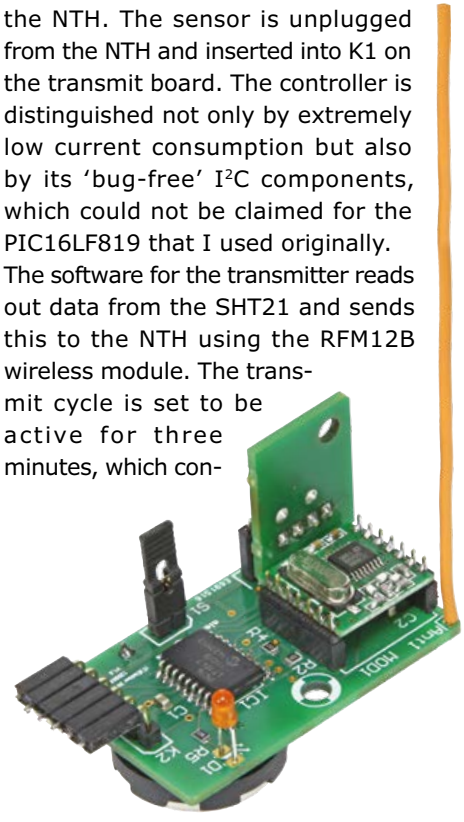
Figure 1. Small and keenly priced with high-performance: the RMF12 transceiver.

gest possible sleep times! Naturally the quality of the CR2032 batteries used also has some bearing on this.

Transmitter

As the schematic in **Figure 2** shows, the transmitter consists of the processor PIC16LF1847, the wireless module RFM12B and the sensor SHT21 in the NTH. The sensor is unplugged from the NTH and inserted into K1 on the transmit board. The controller is distinguished not only by extremely low current consumption but also by its 'bug-free' I²C components, which could not be claimed for the PIC16LF819 that I used originally.

The software for the transmitter reads out data from the SHT21 and sends this to the NTH using the RFM12B wireless module. The transmit cycle is set to be active for three minutes, which considering the lifespan of the batteries is very useful, as during the periods of 'radio silence' the current consumption of the complete transmitter is a negligible 3 µA. R1 and R3 form a voltage divider for checking the battery voltage when the transmitter is powered up. Battery status is indicated by the number of times that the LED flashes (1x: >2.5 V; 2x: >2.7 V and 3x: >2.9 V). At other times the LED flashes only briefly (50 ms), whenever data is being transferred. So no worries about squandering valuable button cell energy. If the battery voltage drops below 2.5 V during active operation, a temperature value of 77.7 is transmitted (so long as the feeble transmitter is still in a position to do this).



Receiver

When you look for a schematic of the receiver you won't find one! Fear not, we didn't forget to include it, because actually it's concealed inside Figure 2. The circuits are so similar that we made do without a second illustration. The components marked in gray are not included in the receiver circuitry.

In fact the receiver consists almost only of the PICLF1847 with an LED and the wireless module RFM12B plus antenna. The circuit is plugged into K2 on the nixie board close to the sensor.

The LED here functions as an indicator for the watchdog timer, which is initi-

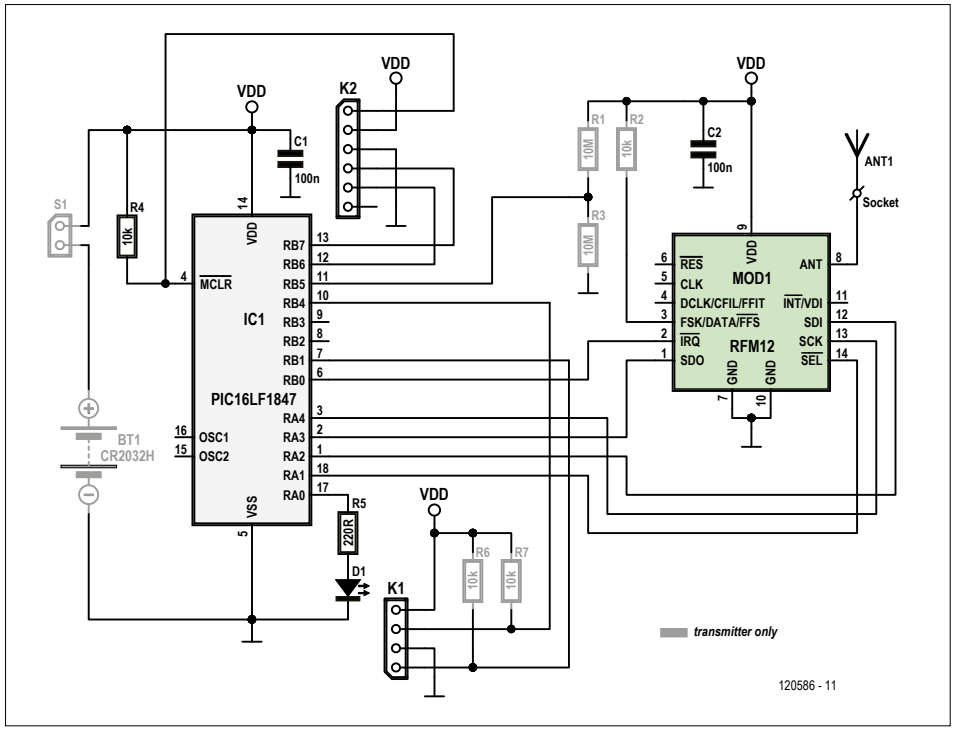


Figure 2. The transmitter and receiver share almost identical circuitry.

Overview of the most important software functions of the transmitter:

- Initialize the PIC, the RFM12B and the SHT21
- While loop
- SHT21_measure:
- --- measure temperature with "Hold Master Mode (HM)"
- --- measure humidity with "Hold Master Mode (HM)"
- Arrange the bytes into the same format as the SHT21 would deliver them if it was connected direct and not over a wireless link.
- 6 values are stored in the Array **mres** for transfer (2 bytes temperature, CRC, 2 bytes humidity, CRC)
- rfm12_snd_msg:
- send 6 values from **mres** using RFM12B
- Check_voltage:
- test battery voltage. If below 2.4V, send 77.7 (as far as this is still possible).
- Sleep state **TX_CYCLE** minutes
- End of While loop

ated every time that a data packet is received. If no data packet is received within around 3 to 5 minutes, then the LED is illuminated. Absence of data packets can occur temporarily, or else permanently when the battery in the transmitter is exhausted. You can check the latter case easily by turning the transmitter off and back on again, counting how many times the transmitter LED flashes (everyone can count up to three!).

The receiver software essentially emulates the SHT21 sensor and replicates an I2C slave. The way readings are processed is of course fine-tuned for the needs of the SHT21 but it should nevertheless be simple to adapt the software to other sensors as well.
Altering the cycle time for the transmitter is achieved using the constant TX_CYCLE. The same-named constant must of course be changed in the receiver, in order to match the watchdog timer.

People who wish to program the two PIC16LF1847 chips themselves should perform the fuse settings as shown in **Table 1**.

Table 1. Fuse settings for programming the transmitter and receiver controllers.

Fuse settings	120586-41 Receiver	120586-42 Transmitter
Oscillator	internal RC, I/O on oscillator pins	
Watchdog timer	ON	
Protect	OFF	
LVP	OFF	
PUT	ON	OFF
Brownout detection	OFF	
Config word 1	39DC	39FC
Config word 2	1EFF	

Construction

As with the schematics, so also for the printed circuit boards. Thanks to the almost identical circuitry for the PICs (albeit with differing firmware) and wireless modules, we have only one PCB layout for transmitter and receiver. A double set of unpopulated boards is available in the Elektor Shop [4] and the same applies for the PIC controllers already provided with the transmitter or receiver software. Fitting components on the PCBs differs only in the couple of passives that need to be installed for the transmitter and are absent on the receiver. Not mentioned at all yet is the six-pin connector K2. This provides an in-circuit programming interface for the PIC16LF1847, which is pin-compatible with the connector of PICkit2/3.

Populating the two PCBs, whose layout can be seen in **Figure 3**, should not be problematic. All components (SMDs in the main) are soldered on the side of the PCB marked with the component designations; only the battery holder for the transmitter belongs on the rear side, as shown clearly in **Figure 4**.

In order to be able to plug the receiver board into the header strip on the nixie board, we solder a straight pin header at K1, but fixed the 'wrong way round' (i.e. on the back of the board). On the transmitter board we mount a straight female connector strip at K1, the right way round this time, so that the sensor board with its angled pin header can be plugged in, as shown in the photo.

The wireless section on each PCB uses socket headers having a 2-mm pitch; these match the pinheader strips provided on the modules themselves. There are two reasons for this: firstly, there are components located directly under the modules, and secondly, programming of the microcontroller on the board should take place only when the radio module and the sensor are not plugged together. The supply voltage for these components is in fact 3.3 V, whereas the voltage on the programming interface is higher (5 V). Not a big difference but sufficient to damage or destroy the components! Therefore we need to plug the wireless module into a socket and remove the module and the sensor (and of course the battery) before carrying out any programming. PICkit2/3 can certainly be set to use 3.3 V (and adjusts itself automatically to 3.3 V when PIC16LF1847 is selected), but this is an important warning in case you use a different programmer or you accidentally choose the wrong controller setting.

The RFM12 wireless modules are supplied not only for 433 MHz but also for two other ISM/LPR frequency bands, 868 MHz and 915 MHz used in North and South America. Each version demands its own initializa-

The receiver firmware turns out rather simpler than that of the transmitter:

- Receive the 6 **mres** bytes from the transmitter in the Function **isr0** and file them in the Array **str** or **str1** as appropriate
- Pass the 6 bytes from **str1** on command from the Master into the Function **sspinterupt**. This represents the core function of the Slave software.

```
sspiinterrupt:
state = i2c_isr_state();   // Indicates the status of the data
transfer (read/write and which data is to be transferred. For details
see CCS C-Compiler documentation.

According to function code 0xE5 or 0xE3 data is stored in the Array
str.
```

tion, since at switch-on the relevant frequency needs to be written to one of the internal registers of the RFM12 module.

The software covers all of the frequencies and to determine which frequency you wish to use the following pins need to be set either High (1) or Low (0):

RB6 (K2-5)	RB7 (K2-4)	Frequency
0	0	433 MHz
1	0	868 MHz
0	1	915 MHz

At switch-on the software reads these settings and sets up the corresponding frequency.

VDD (K2-1) and GND (K2-3) are also present at connector K2. The best thing to do is take a small female header strip (partly visible in **Figure 5** behind the far right-most tube), solder together some scraps of wire (to set RB6 and RB7 to the correct levels) and plug this 'assembly' into the header. Incidentally, K2 on the receiver board is a straight header (intentionally so) but a right-angled on the transmitter board. The latter is not necessary and you can choose to have whichever you wish.

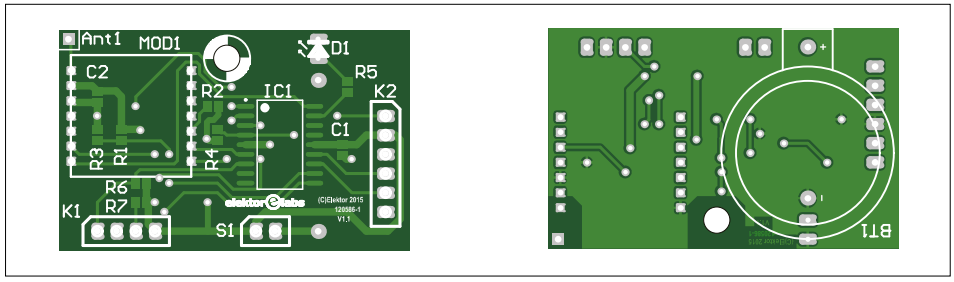


Figure 3. A small printed circuit board for receiver and transmitter, with the battery conveniently fitted on the other side.

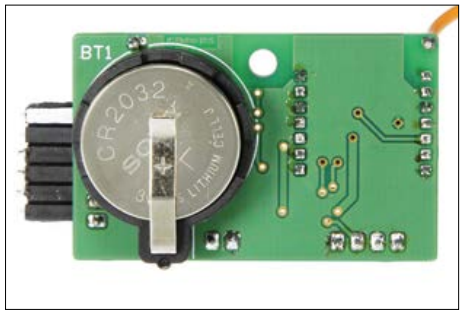


Figure 4. The button cell fits here on the rear side of the PCB.

Figure 5. Nixie board with receiver installed in action.

One final modest but important detail: the quarter-wave antennas. These can be made very simply from stiff wire about 17 cm long (8.5 cm for 868 MHz and 7.8 cm for 915 MHz), corresponding to a quarter wavelength in these LPR bands. A solder pad is provided in one corner of the board for connecting the antenna. ⏮

(120586)

Component List, Transmitter

Resistors
All 5%, 0.125W, SMD 0805
R1,R3 = 10MΩ
R2,R4,R6,R7 = 10kΩ
R5 = 220Ω

Capacitors
C1,C2 = 100nF, ceramic, SMD 0805

Semiconductors
D1 = LED, red, 3mm
IC1 = PIC16LF1847-I/SO, programmed, Elektor Store # 120586-42

Miscellaneous
Mod1 = RFM12B-433-S, 433-MHz transceiver, Hope RF Electronic, Elektor Store # 130559-91 with 17-cm wire antenna (see text for ISM/LPR band alternatives)
S1 = switch (1x on) or jumper
K1 = 4-way pinheader receptacle strip, vertical
2 pcs 7-way 2-mm pitch socket strips and pinheaders, (Farnell # 547-3239, 605-8796)
BT1 = CR2032 Lithium button cell with PCB mount holder (Multicomp CH25-2032LF)

Component List, Receiver

Resistors
All 5%, 0.125W, SMD 0805
R4 = 10kΩ
R5 = 220Ω

Capacitors
C1,C2 = 100nF, ceramic, SMD 0805

Semiconductors
D1 = LED, red, 3mm
IC1 = PIC16LF1847-I/SO, programmed, Elektor Store # 120586-41

Miscellaneous
Mod1 = RFM12B-433-S, 433-MHz transceiver, Hope RF Electronic, Elektor Store # 130559-91
with 17-cm wire antenna (see text for ISM/LPR band alternatives)
PCB # 120568, set of 2, from Elektor Store
2 pcs 7-way 2-mm pitch pinheaders and receptacle strips (Farnell # 547-3239, 605-8796)

Web Links

[1] Sensor SHT-2x: www.sensirion.com/en/produkte/feuchte-und-temperatur/feuchte-temperatursensor-sht2x/
[2] Wireless module: www.hoperf.com/upload/rf/RFM12B.pdf
[3] www.elektor.com/130559-91
[4] www.elektormagazine.com/120586

Arduino Sound-Level Protector

With three-color traffic light

By Clemens Valens (Elektor Labs)

The Arduino is a real jack-of-all-trades. You can use this inexpensive microcontroller board for all kinds of things thanks to the expansion connectors for which countless ‘shields’ are available. In this article we show you how you can realize a ‘sound-level traffic light’ using an Uno, together with an Elektor-developed multipurpose shield and a handful of common parts.

The book “Mastering Microcontrollers — helped By Arduino” [1] is an excellent starting point for anyone who would like to get started using microcontrollers and their programming, without the requirement for a lot of prior knowledge. The book also describes several different hardware examples, which you can usually build on a small piece of experimenting board. In the second edition of the book an extra chapter was added, which contains even more examples. A universal shield was developed for these examples (no. 129009-1), which is available from the Elektor Store, either by itself or as part of a kit. This can be used to quickly build the various circuits.

This circuit board offers space for an LCD, two pushbuttons, two power transistors, temperature, air pressure and humidity sensors, a microphone preamplifier, an LDR, an IR sensor and a remote control receiver. In addition there are two LED outputs and a buzzer, and there is a protected and filtered analog input. Furthermore, there are also two protected digital inputs and outputs, which can also function as a serial port.

Arduino is a veritable multitalent that fits a thousand and one applications complex and simple!

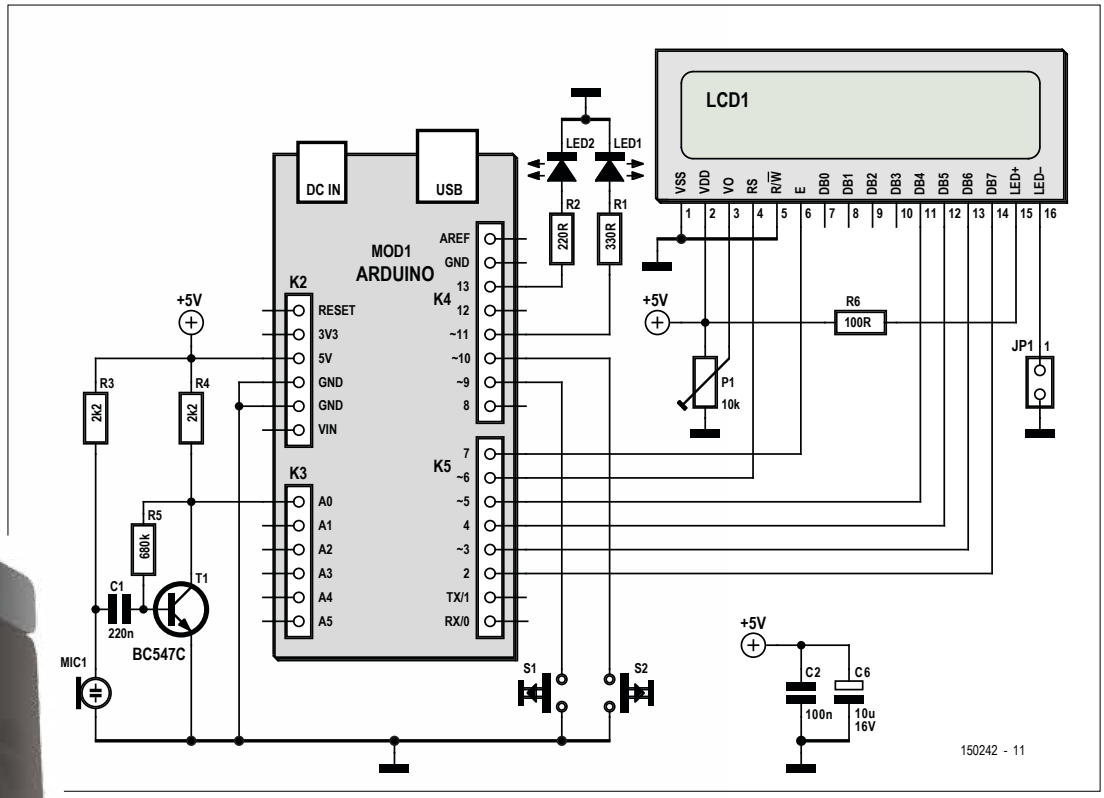


Figure 1. The entire circuit comprises only a few components which can all be fitted on the multipurpose shield.

Not everything can be fitted on the circuit board at the same time, but it still accommodates many of them simultaneously.

The idea behind this circuit was to use a few LEDs to make a kind of sound-level traffic light, where the colors of the traffic light indicate different sound levels. In this case red means 'too loud'. Handy for a concert or when the neighbors are having a loud party again.

It is, of course, possible to realize such a circuit entirely using only discrete components, but here we will show you how you could build one using an Arduino and a multipurpose shield. Besides the LEDs there is also an LCD that operates as a VU meter and with a few push-buttons it is easy to set the different thresholds. This is an excellent project to gain some programming experience with an Arduino!

The circuit

On the multipurpose shield 129009-1 there is space available for a microphone preamplifier. This can be used by the Arduino to react to sound. The microphone is an electret type and the amplifier is a classic single-transistor stage with a gain of about 1000 (see **Figure 1**). The output of the amplifier is connected to the Arduino's analog input A0.

Originally it was intended to use two LEDs for the indication, a green one and a red one. The shield also has space for these. However, as I was building this circuit I realized that I did not have a usable, 5-mm, green LED on hand. But I did happen to have a bi-color red/green LED with a common cathode, so I used that instead. Fitting it in the position of LED2, the third connection can then easily be connected to LED1, because the buzzer, which shares an output with LED1, is positioned close to LED2 on the circuit board. Thanks to the bi-color LED I now have a third color available, namely orange. So this ultimately became a 'real' traffic light.

Component List

Resistors
Default: 5%, 0.25W
R1 = 330Ω
R2 = 150Ω
R3,R4 = 2.2kΩ
R5 = 680kΩ
R6 = 100Ω
P1 = 10kΩ preset, horizontal

Capacitors
C1 = 220nF
C2 = 100nF
C6 = 10µF 50V

Semiconductors
LED1 = green, 5 mm
LED2 = red, 5 mm
Alternatively, LED2 = bicolor, common cathode
T1 = BC547C

Miscellaneous
S1,S2 = pushbutton, make contact, 6x6mm
MIC1 = electret microphone, 6mm
JP1 = 2-pin pinheader, 0.1" pitch
K2,K3 = 6-pin pinheader, 0.1" pitch
K4,K5 = 8-pin pinheader, 0.1" pitch mm
LCD1 = 2 x 16 characters, with backlight
K6 = 16-pin pinheader, 0.1" pitch
For LCD: 16-pin connector, 0.1" pitch
Jumper
PCB # 129009-1

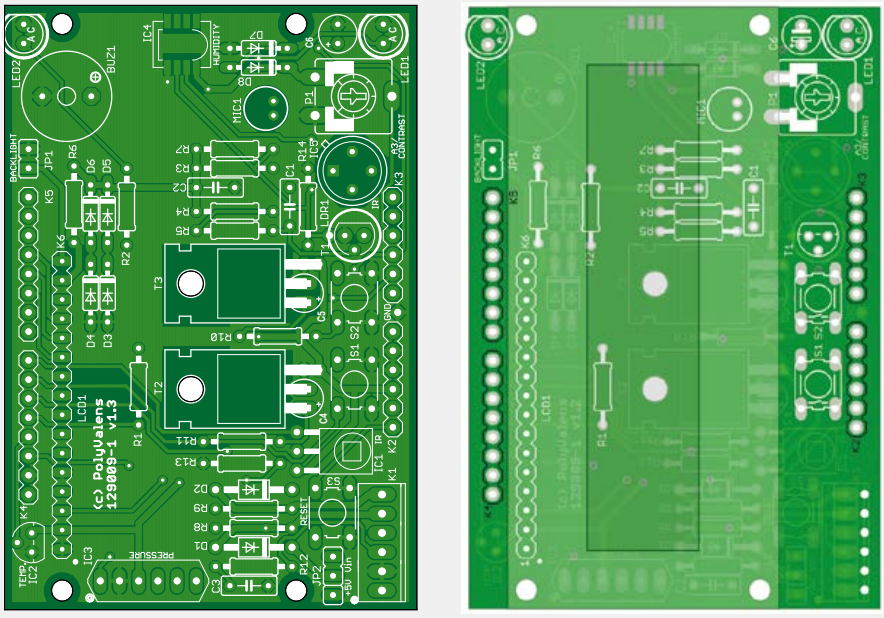
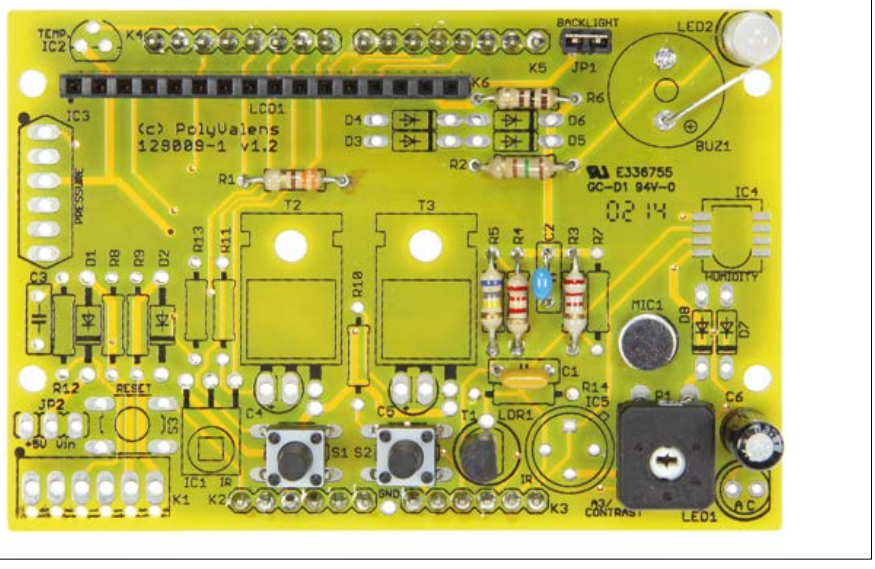


Figure 1. The entire circuit comprises only a few components which can all be fitted on the multipurpose shield.



Figuur 3. De opgebouwde print, het LCD is voor de duidelijkheid nog niet op de connector gestoken.

Because we don't have a sound-level meter at the Elektor lab any more, I was not able to calibrate the circuit in dB. To make the circuit as general-purpose as possible, it would be very convenient if the user could adjust the thresholds themselves. For this reason I have added an LCD in 4-bit mode to the circuit, as well as two pushbuttons. The display can show the raw values as well as the processed ones, and using the buttons a menu can be navigated. P1 is used for the contrast setting of the LCD and a jumper on JP1 can be used to turn on the backlight. All these components will also fit on the multipurpose shield without any problems.

The software

The program, an Arduino sketch, of course, ultimately became quite extensive in order to demonstrate a number of different tricks and techniques.
The sound level is measured with *analogRead*. This happens four times in a row, after which the values are averaged. The reason for this is not so much to filter the noise, but to slow down the program. As it happens, the function *analogRead* is very slow and four of these, one after the other, reduce the loop time of the main loop down to about 2 kHz. This is then also the sample rate of the sound signal.
The peak amplitude of the measured signal is determined with the aid of a leaky integrator, so that the peak value follows the sound level, but reduces to zero when the sound is gone.
The peak value is filtered by a first-order low-pass filter with a cut-off frequency of about 0.1 Hz, to build in some delay. For this a genuine floating-point IIR filter-algorithm has been used (with explanation), so that you can see how this is actually implemented in practice.
Once per second the actual filtered peak value is compared with the configured thresholds and the result is shown on the display and via the LEDs. During this update the sound level is not sampled, because it appeared that the switching of the LEDs had an impact on the analog input of the Arduino.

There are three thresholds for the sound level: green, orange and red. 'Off' is no or very quiet noise (background noise), green is a little bit louder (a quiet bit of background music, default value = 100), orange is a little louder still (singing along,

default value = 200) and red is very loud (yelling along with the music, deafeningly loud music, default value = 300).
The display shows the sound level as a numeric percentage of the highest (red) threshold. This value is also displayed as a horizontal bar on the second line. This makes it possible to read the display from a distance, but is also used as a demonstration of how you can program such a thing. The display also indicates the peak value as a number from 0 to 1023. This is very useful when selecting the thresholds.
Setting the thresholds is done by simultaneously pressing the two pushbuttons. There now appears the option of setting the green threshold (0 – 1023). Pressing both buttons again brings us to the orange threshold and once more gives the red threshold and finally the setup mode is exited by pressing both buttons for the last time. The selected threshold levels are stored in the EEPROM of the microcontroller.

Construction

Figure 2 shows the layout of the multipurpose shield. For a schematic of all the individual circuits that can be built on this board we refer to the book [1]. Only those components that have to be fitted for this particular application are shown in **Figure 2**. Note that connectors K2 through K5 have to be mounted on the solder side. The LCD is connected to the shield using a 16-way pin header and a corresponding connector. The manner in which the bi-color LED is connected can be clearly seen in **Figure 4**.
The built-up shield is plugged on top of an Arduino Uno, after this it can be loaded with the ino-file that is available as a free download from [2].

Conclusion

As you can see, this circuit demonstrates all kinds of things:

- Sound-level measurements with an Arduino
- Generating an alarm
- Digital filtering
- Driving an LCD
- Driving LEDs
- Reading switches
- Implementing a Setup menu
- Displaying a bar graph
- Using LCD custom characters
- Storing and then retrieving values from the EEPROM

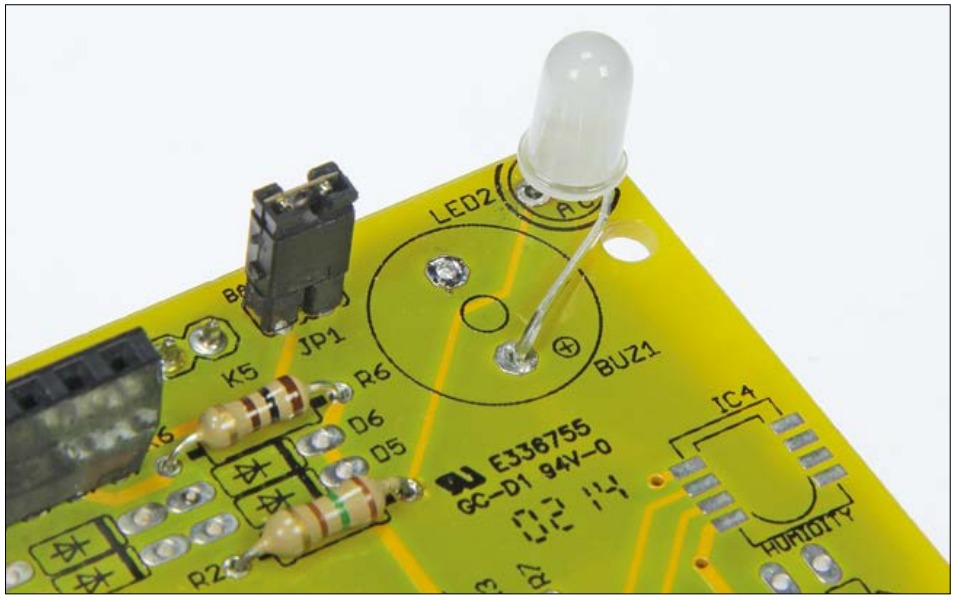


Figure 4. This is how the bicolor LED is mounted.
The lead for the green LED goes to the + connection for BUZ1.

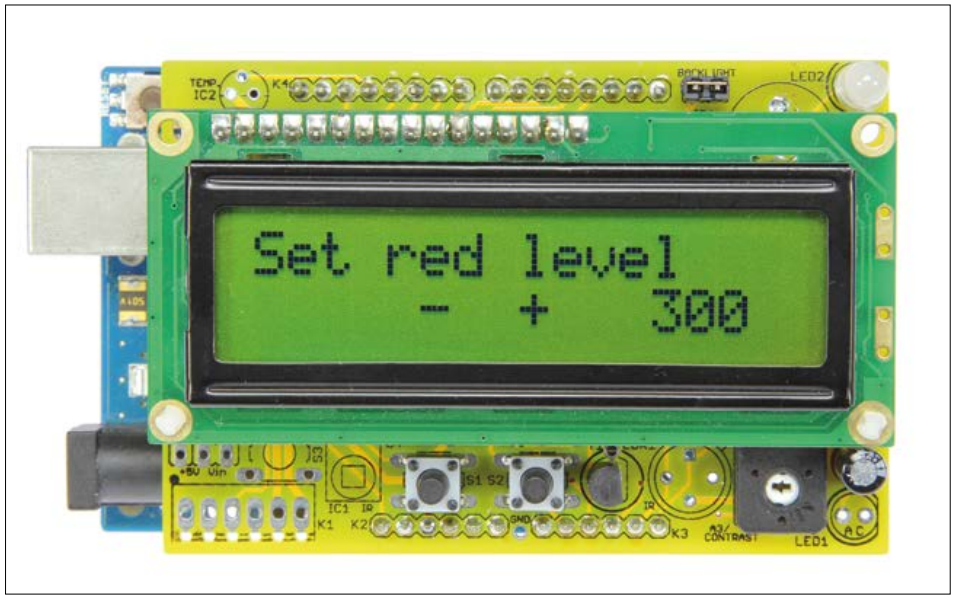


Figure 5. Setting the threshold level at which the red LED turns on.

As an exercise for the reader there is still the serial port. This can be used to send the measured values to a PC or so, where they can be logged. Handy as evidence in the event of a protracted disturbance. ⏮

(150242)

Web Links

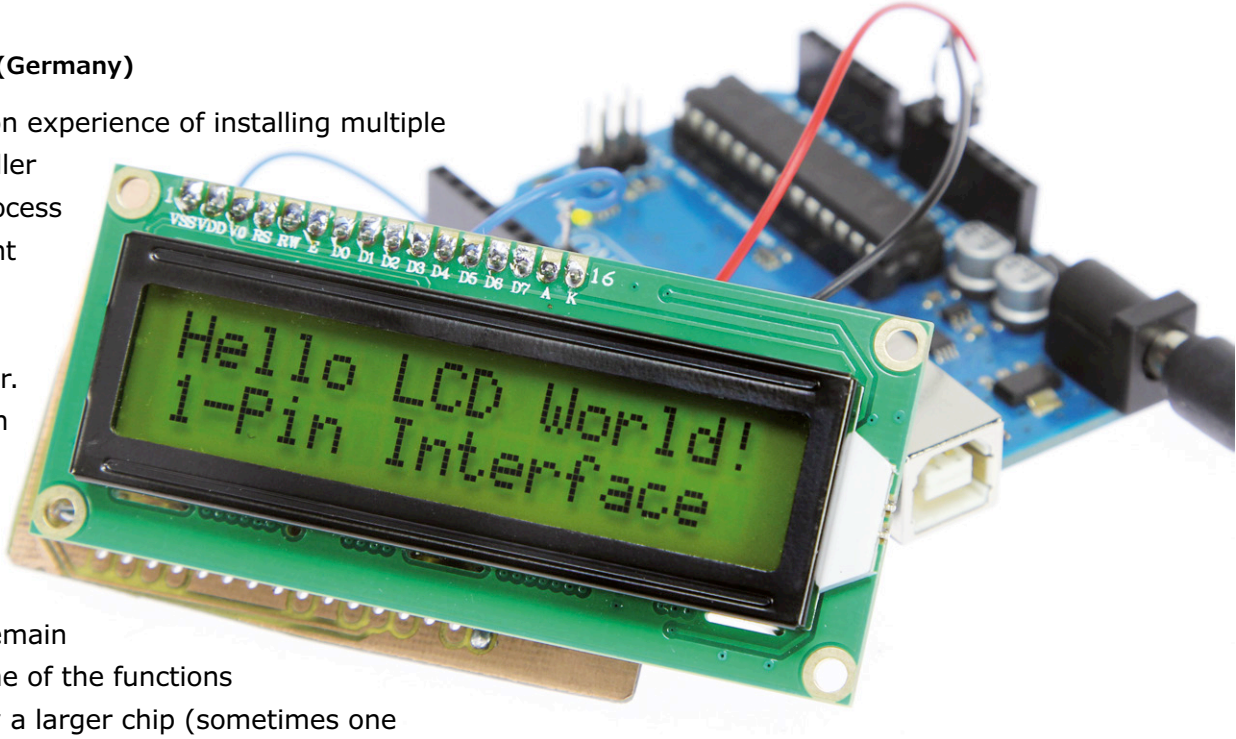
[1] www.elektor.com/mastering-microcontrollers-helped-by-arduino

[2] www.elektormagazine.com/150242

Single Wire LCD Interface

Control your LCD display using just one data wire

By Detlef Hanemann (Germany)



Anyone with hands-on experience of installing multiple types of microcontroller knows that in the process of circuit development they are often forced to add one extra function after another. Eventually they reach the stage where the number of I/O pins available is no longer sufficient. Only three options remain now: do without some of the functions (reluctantly), employ a larger chip (sometimes one doesn't exist) or use a so-called port expander (at extra cost). All three workarounds have their pros and cons. A developer's life would be made considerably easier using peripherals that took up only one port pin of the microcontroller. And that's precisely what the interface for commonly used LCDs described here does.

Using small microcontroller like, say, an 8-pin ATtiny from Atmel, we're left with just five GPIOs remaining after discounting the two supply pins, assuming we wish to retain the reset pin. That's not many. A larger chip with more pins not only takes up more surface area of the PCB but we still run out of pins rapidly after connecting all the press buttons, LEDs and other peripherals, especially if they demand that data is transmitted in parallel, as many LCD modules do. In addition, the increased pin-count usually complicates the soldering, because the inter-pin distances are reduced.

If you neither wish to use larger controllers nor forego functions, then you are forced to use a port expander, consisting usually of shift registers connected either by I²C (slow) or by SPI (requiring an extra pin). Even then, the shift registers use up at least two or even three pins of the microcontroller for converting the data into parallel format and transferring this to suit the particular interface of the LCD module. This kind of kludge certainly leaves potential for optimization.

Technical specification

- Universal interface for parallel LCDs
- Serial addressing using only one GPIO line
- Plugs into LCD modules with differing pin configurations
- Compatible with low-cost LCD modules
- Minimum transfer time for a character = 0.8 ms
- Compatible with all microcontrollers

Optimal solution

The attainable optimum just has to be the use of a single data line. Anything fewer would be impossible and any more than one not really necessary, as an LCD only receives data; it doesn't send any. An optimum solution should not require reinventing the wheel each time either. Therefore we have relocated the necessary conversion from serial to parallel away from the actual microcontroller circuit into a module that contains the necessary conversion electronics, so that inexpensive, widely available LCDs can be used. A display upgraded in this way can then be connected to any microcontroller with minimal effort.

Single-wire solution

Long before the Internet age people were already aware of methods of tricking a shift register into accepting serial data using fewer inputs than were actually necessary. The simplest possible shift register has at least two inputs, one for data and one for the clock. The clock is necessary to determine the point in time at which the level on the data input is valid. However, if you generate a special combined data + clock format, you can then separate the data from the clock using a simple resistor and capacitor (RC) element. **Figure 1** shows how this process works

in principle. There is one prerequisite, that the shift register is edge-triggered and reacts to a rising edge — which is the case with many of these chips. When there is no signal transfer happening, the quiescent level is High. In this state both inputs are at High level and there is no activity. If a short pulse (measured by the time constant of the RC low-pass) occurs, the data input remains High and as a result the rising edge of the (negative-going) pulse clocks a High into the internal register. If we now wish to push a Low into the register, then we need only make the negative input pulse much longer, so that the voltage at the data input falls to Low. The input signal now consists of a sequence of short and long negative pulses — depending on the bit pattern being transmitted. If you select the timings appropriately, the solution functions very reliably.

To avoid the data piling up sequentially bit by bit on the parallel outputs and causing indeterminate states there during the clocking-in period, shift register ICs are generally provided with an output register that takes in data from the actual shift register only 'on demand' and then defines and organizes the data at the outputs in one fell swoop. To this end, therefore, a third input is necessary. As would seem natural, a solution was found from authors on the Internet such as [1] by simply using a second RC element with greater time constants. Here, however, short and long pulses would not suffice for bringing the input for data transfer down to Low. This pulse must then be very long. Altogether we now require a control signal containing the short (High), long (Low) and very long (acceptance) pulse. This solution also works well, albeit with a couple of restrictions: the final bit must always be Low and can therefore not be used for serial-parallel conversion.

For connecting an LCD module we're left with one further problem, however. Although data is transferred to commonplace two-line LCDs using the Hitachi HD44780 controller (or compatible types) mainly in the '4-bit parallel' mode, which handles seven data bits easily, it expects to see a positive pulse in order for the LCD to accept the data delivered by the shift register. To generate a pulse of this kind we need to transfer each data word twice over, which is inconvenient and wastes time [3]. Therefore the author came up with a circuit using a simple alternative control signal to achieve a different method of data acquisition into the output register and delivery to the LCD module.

Single-wire circuitry

As seen in **Figure 2**, the control signal passes from Pin 3 of K1 to the two different low-pass filters. The fast filter (R1 and C1) has a time constant of approximately 10 µs and drives the data input Pin 14 of IC1. The slow one (R2 and C2) has a time constant of around 200 µs, however, and does not control the internal data acquisition for IC1 (as with the solution in [1]). Instead it enables a reset of its Pin 10. Data acquisition is therefore solved in a different way.

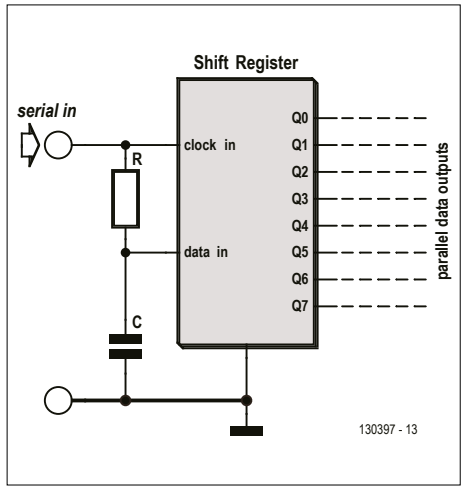


Figure 1. Block diagram for controlling a shift register via an RC low-pass filter using only one data line.

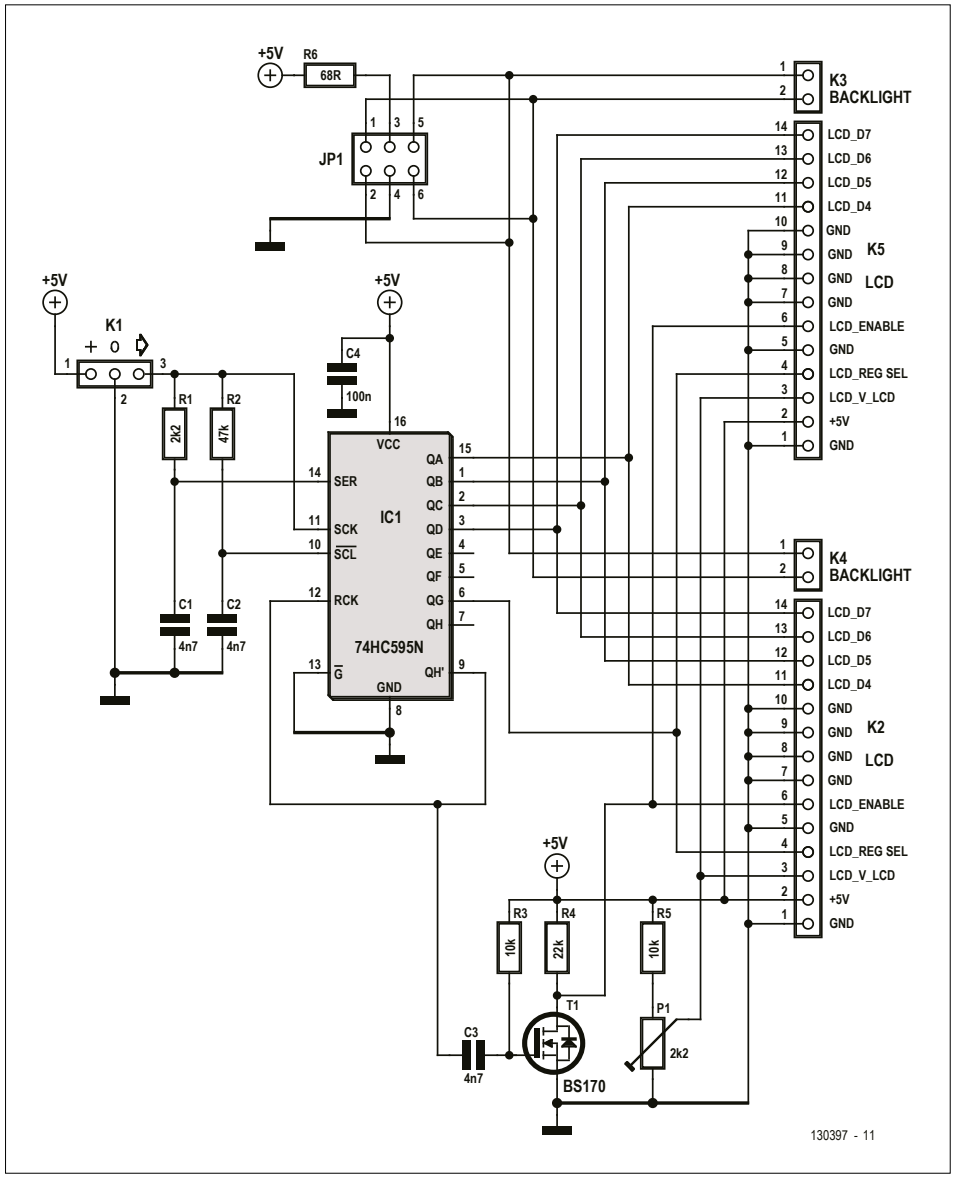


Figure 2. The author's solution with two RC low-pass filters uses one element of the 74HC595 shift register.

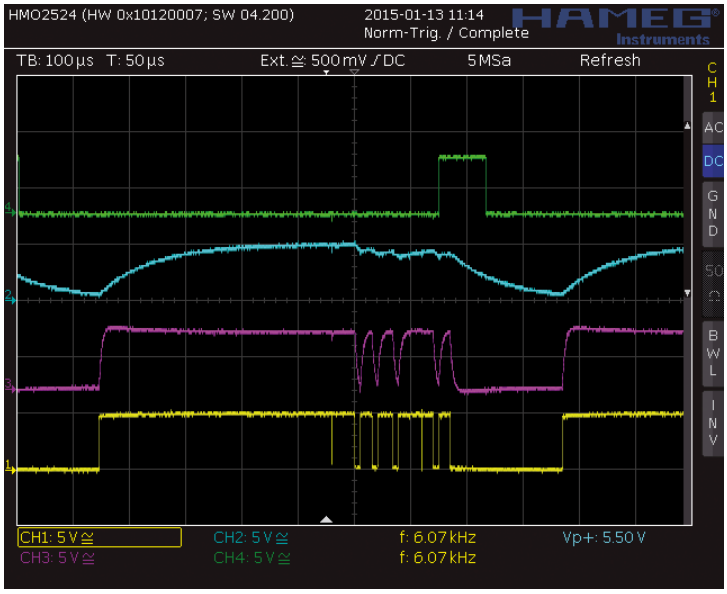


Figure 3. Oscillogram with the following signals: yellow = control signal, violet = data input Pin 14, blue = reset input (Pin 10) and green = signal at QH' (Pin 9 of IC1).

Responsibility for data acquisition into the output register of IC1 is handled by Pin 12, which also looks after the LCD_ Enable input of the LCD module at K3 or K4 respectively. The positive pulse for the LCD is produced at the end of each data word transferred using the monostable constructed around T1. This is feasible thanks to a special feature of the 74HC595 shift register employed, in which the very first High applied for bit 7 arranges on the last clock timing for the acquisition of data into the output register of IC1. Pin 12 of IC1 is linked to its QH' output, which in fact is responsible for cascading ICs of this kin d and is consequently linked directly to the shift register. The reset input of IC1 deletes — in this IC only — the status of the shift register but not the output register. With all the bits clocked in and a reset triggered by a fairly long negative control pulse on Pin 10, QH' is now set to Low, with QH and the other outputs remaining as they were. C3 and R3 together activate the monostable of T1 to enable acquisition of data for the LCD. After some recovery time for R2/C2 the next data word can be transferred. **Figure 3** shows the progress over time of the various signal levels at the relevant stages of the circuit.

The timings for the Low, High and Reset pulses selected by the author mean that the transfer of an 8-bit data word to the LCD takes around 1.3 ms. As you can see, other than the four data bits QA to QD (bits 0 to 3), the only other information needed is QG for controlling the register selection of the LCD. QE and QF offer two additional pins you can define for your own purposes. While Luc Lemmens in Elektor Labs was doing some fine tuning and developing a PCB layout, he provided K2/K4 along with K3/K4 as pinheader strips for connecting the LCD display. Regular two-line LCD modules vary mainly in whether physical connectors are provided on the upper or lower surface of the device (and in which order) or not at all. Other variations concern the polarity of the backlighting (and frequently even the pin assignment). This makes it impossible to say whether our interface truly works with all modules. Luc is certain, however, that by providing K2 plus K4 together with K3 and K5 (and selectable polarity for the backlight using JP1), he has achieved the best possible flexibility and compatibility. In addition, Luc has reduced the time constants of the RC elements against the author's version for more stable data transfer. Incidentally, in your own programs you should not make the duration of the Low pulses any shorter (to avoid jeopardizing stability). On the other hand you are free to lengthen the High pauses. In the demo software Luc has also made provision for a trigger signal to Arduino Digital pin 9 (see prototype photo). This is for test purposes and you can of course omit this in your own implementation to save pins.

Component List

Resistors
All 5%, 0.25W.
R1 = 2.2kΩ
R2 = 47kΩ
R3,R5 = 10kΩ
R4 = 22kΩ
R6 = 68Ω
P1 = 2.2kΩ trimpot, large, horizontal

Capacitors
C1,C3 = 4.7nF, 0.1" pitch
C4 = 100nF, 5mm pitch

Semiconductors
T1 = BS170
IC1 = 74HC595, DIL

Miscellaneous
K1 = 3-pin pinheader, 0.1" pitch
K2+K4,K3+K5 = 16-pin pinheader or 16-way pinheader receptacle*, 0.1" pitch
JP1 = 6-pin (2x3) pinheader, 0.1" pitch
2 pcs jumper, 0.1"pitch
LCD module, 2x16 characters*
PCB # 130397-1*

* see text

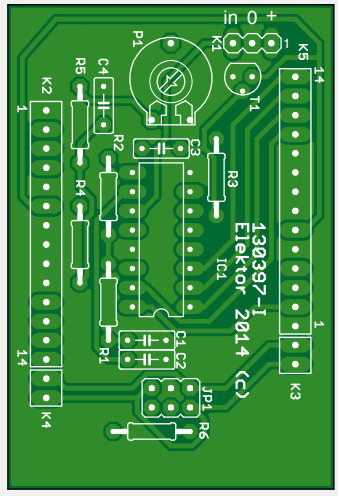


Figure 4. Component side of the PCB for single-wire LCD interface.

Construction and operation

There is really not very much to say about populating the board seen in **Figure 4**. With only one wired IC and ten passive non-SMD components, there is nothing insurmountable even for raw recruits to

soldering. The only thing to watch out for is the functional order of the connection pins of the LCD and whether it is equipped with male or female connectors. The pin assignment is usually printed on the LCD board — if not, the data sheet will assist. **Figures 5** and **6** show the prototype plus LCD, together with an Arduino.

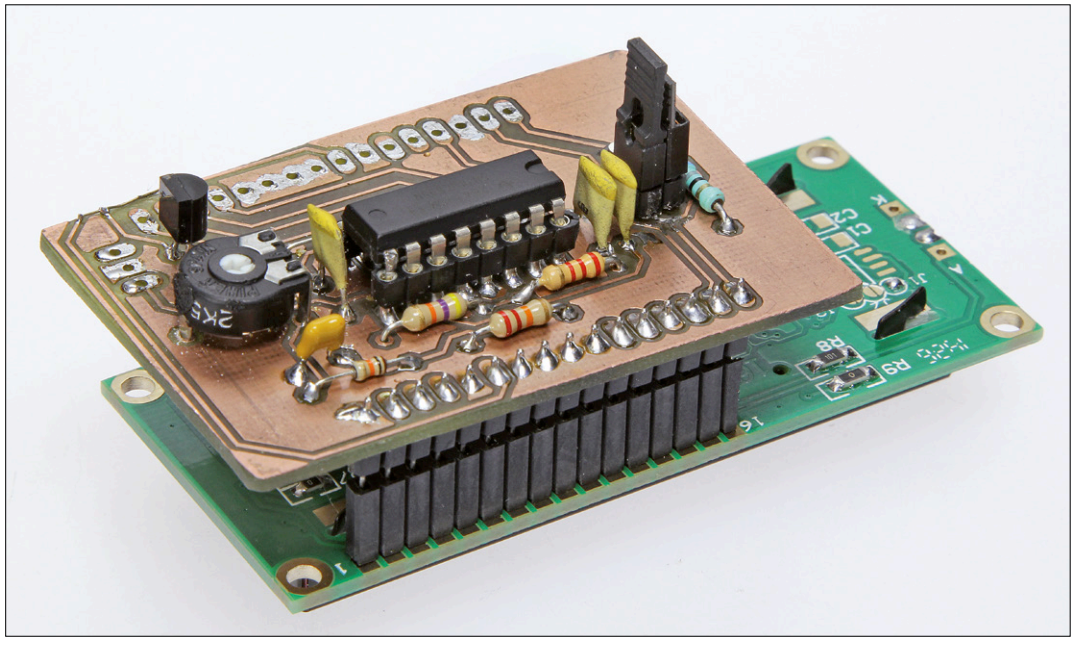


Figure 5. Prototype of the single-wire LCD interface, mounted on the rear of an LCD module.

Take care with the connections and polarity of the LCD backlight. Two jumpers look after the latter. When jumpers JP1 are closer to K3 (linking 1 with 3 and 2 with 4), they arrange for the (positively charged) anode to be on the outermost terminal of K3 and on the innermost terminal at K4. However, if the two jumpers of JP1 are closer to K4, the situation is reversed. The combination of interface and LCD is linked direct to the microcontroller extension board via K1. It's important here that the supply voltage of the microcontroller being used is the same as that of the LCD module. If the character stream output by the (correctly) programmed microcontroller cannot be read on the display, it's worth first checking whether the contrast is set correctly using P1.

As usual the layout files for the PCB as well as the sample code (for the Arduino) are available free at the web page for this article. Configuring this for the particular microcontroller used should not pose any problems thanks to the ample commentary provided. You can of course also obtain the small double-sided PCB ready etched, drilled and overprinted in the Elektor Store [1]. ⏮

(130397)

Web Links

[1] Roman Black's solution: www.romanblack.com/shift1.htm (and do check out the rest of his 'minimalist design' website)

[2] www.elektormagazine.com/130397

[3] www.elektor-labs.com/node/3598

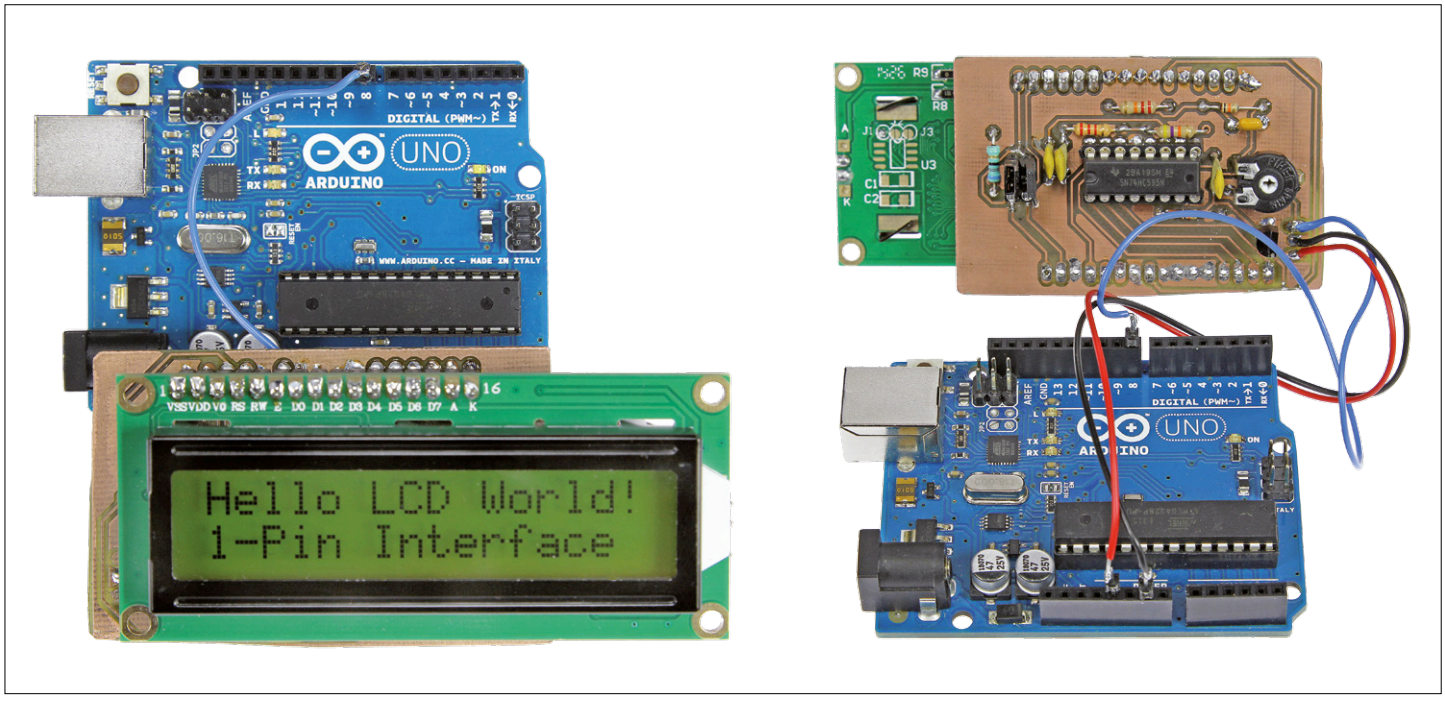


Figure 6. Single-wire LCD interface in action, connecting an Arduino board to a display.

Over-Air LiPo Battery State Monitor

Zigbee'd measurement of voltage, temperature, and current up to 170 amps

By **Michel Kuenemann** & **Gilles Krebs** (France)

The board described here measures the voltage and the current up to 170 A supplied by the battery in your scale model, then transmits these data via radio to your remote-control transmitter, or via USB to a Qt (say: *cutie*) application on a PC. And even if you're not into modelling, you can adapt this project for your home automation or robotics applications, based on ZigBee and 1-Wire or I²C buses.

Most electrically-propelled scale models are currently fitted with Lithium polymer batteries. Thanks to constant improvements in manufacturing processes, these accumulators, at least the most powerful of them, are now able to comfortably supply currents of several hundred amps. So with currents of this order, it's a good idea to monitor them, as unless appropriate precautions are taken, they can cause major damage: burn up a board, the model, or yourself — or worse.

When we need to measure high currents, the standard technique of using a shunt is still a good solution, as its low resistance allows us to limit heating and the inevitable losses it introduces. The 500 µΩ / 8 W shunt in an SMD package selected for this project dissipates only 1.25 W when carrying a current of 50 A. As you know, when a current encounters a resistance, even as low as the resistance of a copper track, a potential difference appears across this résistance. Thus if we insert our shunt into the battery 'negative' rail, a high current can cause a potential difference on the 0 V track that will result in an offset with respect to the 'real' 0 V. To avoid this sort of problem, we have adopted a different solution.

If we place the shunt in the 'positive' lead, the battery voltage will be present on one of the measuring amplifier inputs, while the other input sees this same voltage, less the small volts drop across the shunt. Hence both inputs are seeing a voltage that's almost identical (common mode) and which is also very high. In order to amplify this rather special signal, we have used an amplifier capable of handling a common-mode input right up to its supply voltage. The gain of this amplifier has been set for a full-scale reading of 170 A.
The battery voltage measurement is achieved via a conventional resistive divider, set for a full-scale of over 30 V.
The temperature measurement is achieved using a small analogue sensor in a SOT23 package, which will be positioned under the PCB in contact with the battery.
The radio transmission is by way of a Microchip MRF24J40 2.4 GHz RF module [1]. As this module comes in two versions, one providing 1 mW and the other 100 mW, we have designed the PCB with a dual layout allowing either version to be fitted. The 1 mW version offers a range of around 100 m (300 ft.), while the 100 mW module claims a range of over 1 km (3,000 ft.). This module has already been described in the '2.4 GHz Transmitter and Receiver for Model Aeroplanes' [2].
The PIC18F26K20 8-bit microcontroller, running on its internal clock, is tasked with digitizing the signals and feeding them to the RF module that is going to transmit them.
USB connectivity is provided by the very standard FT232RL.

Two expansion connectors give access to an analogue input port, a 1-Wire bus, and an I²C bus operating in 'bit bang' mode, i.e. implemented in software.
The board offers several powering options. It can derive its power from the following four sources:

- the USB socket;
- the serial connection (UART) connector.
- the positive pole of the battery;
- the battery balancing tap, at the point where the second element is connected (7.4 V);

Since one picture is worth a thousand words, **Figure 1** shows how to connect up the board in each powering configuration. Diodes D1, D3, D4, and D7 prevent the various sources from being connected in parallel in the event of more than one being connected at the same time.
Note that the regulator's maximum input voltage is 16 V, limited to 15 V by the protective zener diode. So when using LiPo batteries with more than three elements, it will be necessary to power the board from the battery balancing tap (J11 is provided for easy connection), or choose another source (USB or UART socket).

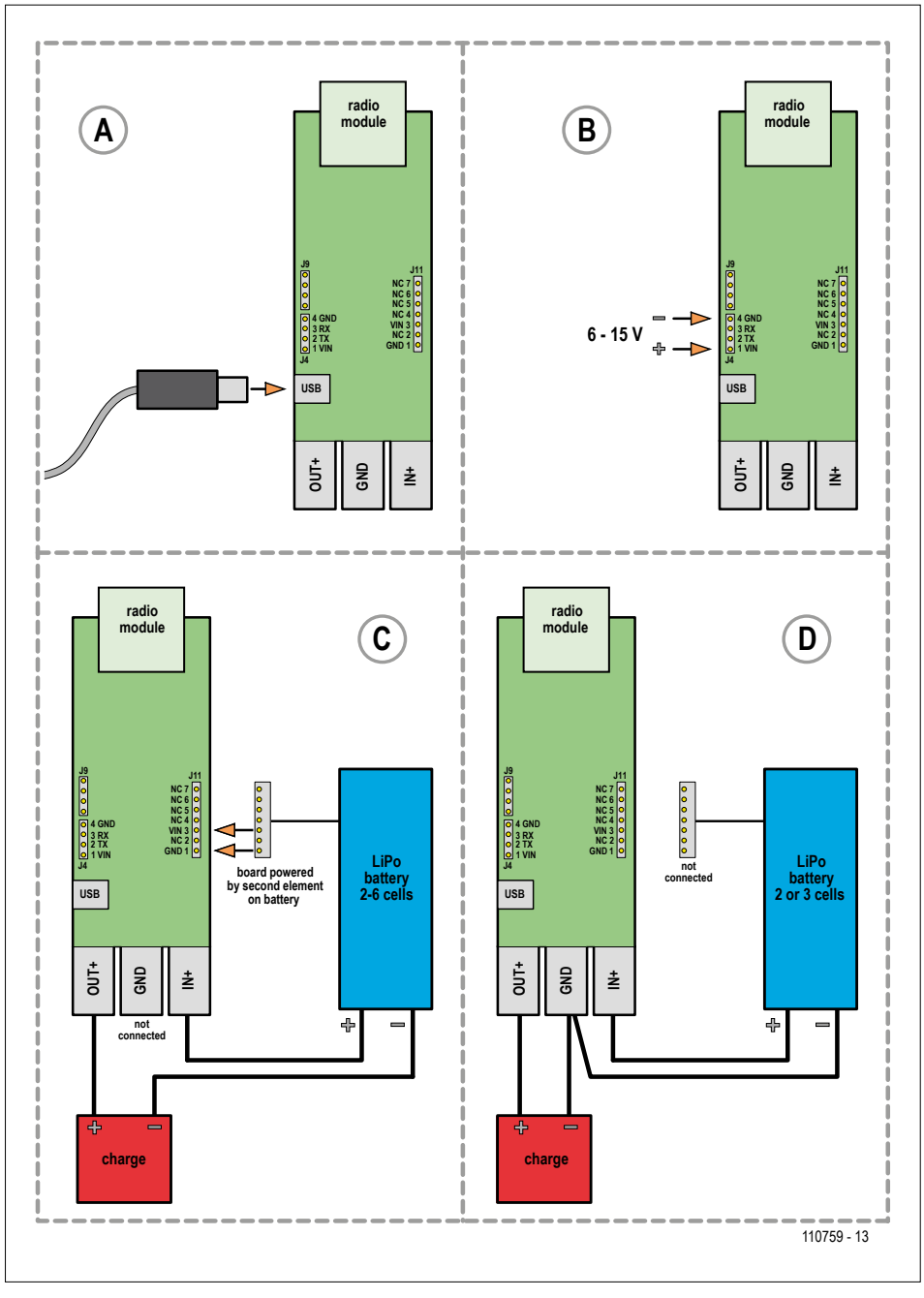


Figure 1. Four ways of powering the board. A: via the USB socket; B: via the UART connector; C: via the balancing tap; and D: via the power feed (for 2S or 3S LiPo batteries only).

Technical specifications

- Compatible with LiPo batteries from 2S (7.4 V) up to 6S (22.1 V)
- Continuous current 80 A, 170 A for a few seconds
- Current measuring range: 0 – 170 A
- Insertion resistance 1 mΩ in the battery 'positive' line
- Temperature measurement built in to the board
- Connections: direct soldered, PK or DEAN sockets
- Provides measurements of voltage and current with calculation of instantaneous power drawn
- Summing of current supplied
- Compatible with previously published 2.4. GHz Transceiver [2]
- Open communication interfaces: USB, UART, I²C, 1-Wire
- ZigBee compatible 2.4 GHz technology
- Line-of-sight range of over 100 m (300 ft) or over 1 km (3,000 ft), depending on type of radio module.

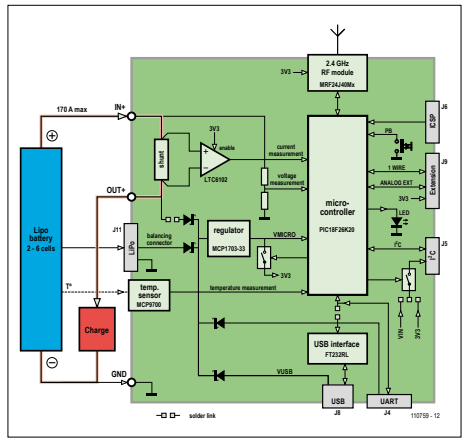


Figure 2. This block diagram makes it easier to follow the circuit diagram given in Figure 3.

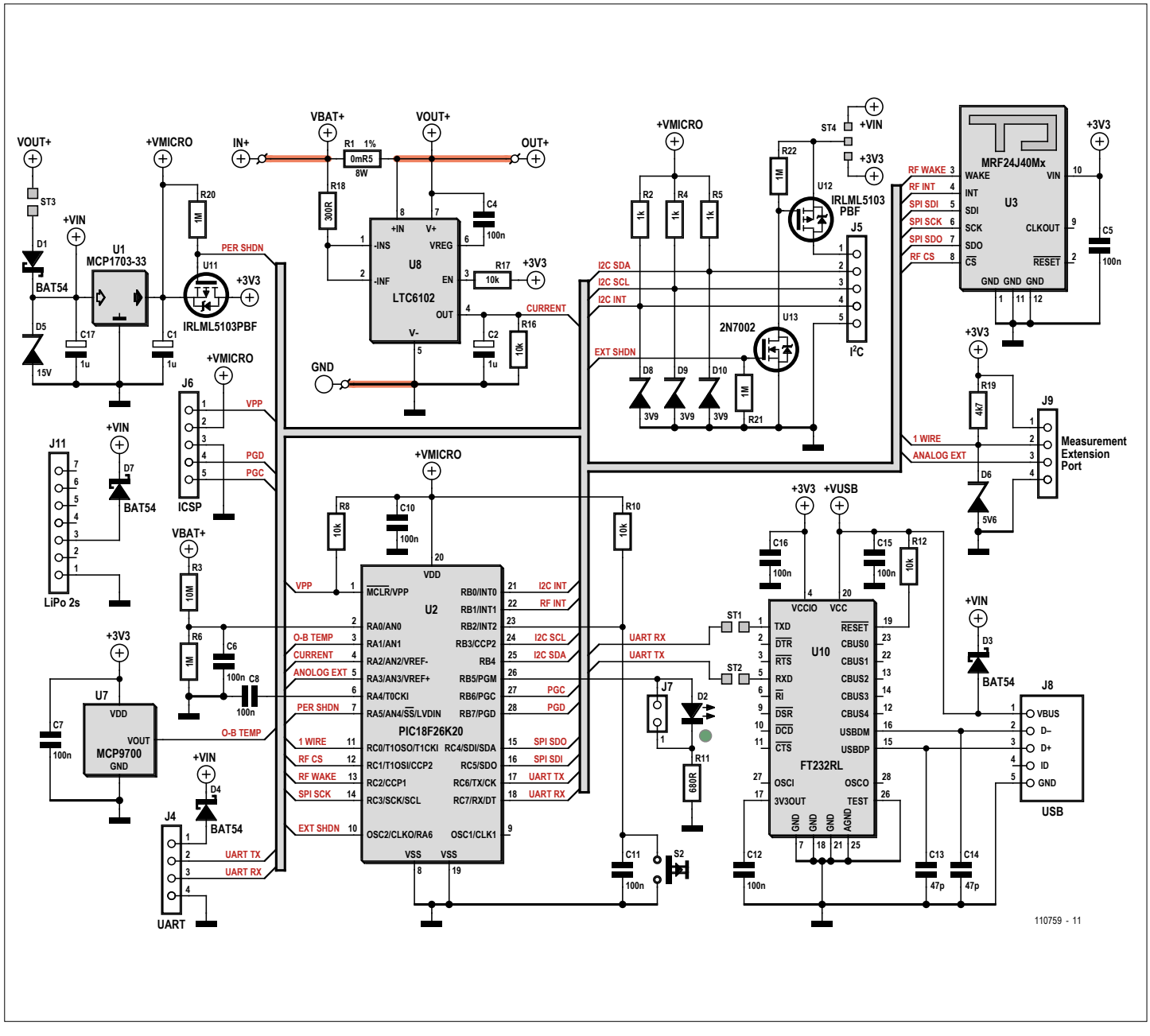


Figure 3. Circuit diagram for the board. The LTC6102 IC (U8) makes it possible to measure currents up to 170 A using a small microcontroller.

An open project – If you're not into modelling, don't be afraid to modify this project for your home automation or robotics applications, based on ZigBee and 1-Wire or I²C busses. And do tell us all about your projects.

Two switches that can be driven by the microcontroller allow accurate control over the power consumption, by managing the power to the peripherals.

And to finish, an LED and a push-button provide the 'Man/Machine Interface' — minimalist, but quite sufficient for this application. **Figure 2** summarizes the architecture of the board.

Circuit diagram

A glance at the circuit of the board (**Figure 3**) immediately shows the main elements described above. We can also see the detail of the decoupling and supply protection. The supplies to the peripherals are switched via P-channel MOSFETs. Apart from the connectors, the components are all in SMD packages. This technique allowed us to reduce the board to dimensions compatible with a current model of LiPo battery with a capacity of 2,200 mAh (**Figure 4**).
The expansion connectors are on a 2.54 mm (0.1") pitch and thus easy to extend to an breadboard or 'Labdec' experimental system.

Software

The software package provided with this project, available from [3], comprises three parts:

- the board firmware (EP24.hex);
- the software for PC under Windows (EP24control.exe);
- an update for the 2.4 GHz Transceiver [2] software (RC-Transceiver-SW-Package.zip).

The microcontroller software (EP24.hex) measures the voltage, current, and temperature values once per second, and handles the voltage and temperature alarms. It then transmits these data over the USB connection and over the 2.4 GHz link if the radio module is present on the board. If you want to adapt the transmission interval or gain parameters to suit your needs, you'll need to modify and recompile the source code before flashing your board.

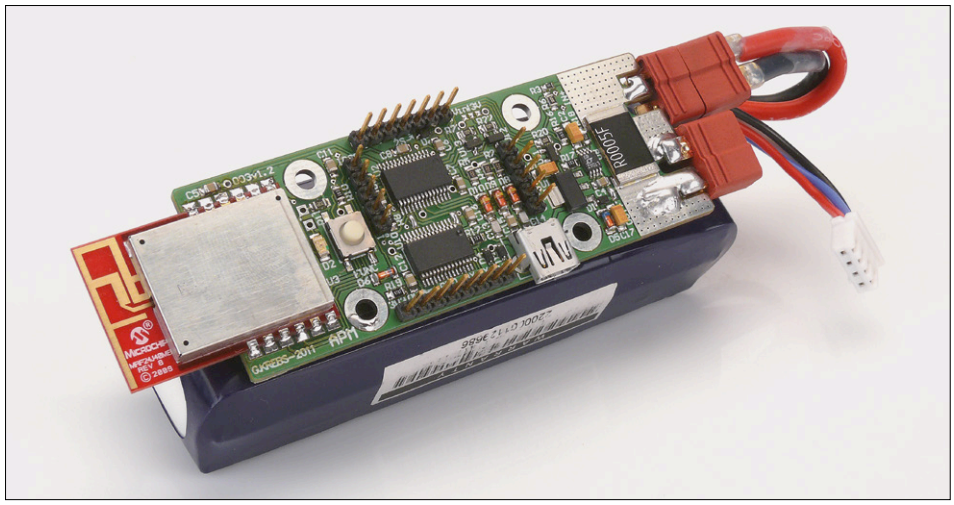


Figure 4. The finished board fitted to a 3S 2,200 mAh LiPo battery.

This software has three operating modes.

Mode 1
If the board includes a radio module, the software will automatically send the voltage, current, and temperature measurements by radio and over the USB link as soon as it is powered up.

Mode 2
If the board includes a radio module and the push-button is pressed while the connection is made to the USB socket, the software will go into receive and will automatically send the voltage, current, and temperature measurements received by radio to the USB link for display in the EP24control window.

Mode 3
If the board does not include a radio module, the software will automatically send the voltage, current, and temperature measurements over the USB link. It will be possible to display these data on a PC using the EP24control software. This application for a computer under Windows is used to present the data received over the USB link in the form of a graph. A number of indicators show the status of the alarms transmitted by the on-board application. It is possible to save these data to the hard drive in a .CSV format file by clicking the save button. EP24control has the special feature of having been developed under Qt, a free environment that makes it possible to efficiently develop communicating applications (see *About Qt* box).

Construction and testing

It's perfectly possible to solder the SMD components manually, as long as you have patience and take the greatest care. Fitting the shunt is a little tricky, as it can only be done using a spot heater, unless you use reflow soldering. For the moment, don't fit the MRF24J40M module.
After checking the components are the right way round and your soldering, make the solder bridges ST1 and ST2. These make it possible to connect the microcontroller to the USB interface.
Connect the USB socket to your PC and to the board. Under Windows, with any luck the FT232 device should be recognized without needing anything to be installed first; otherwise download the driver from the www.ftdichip.com website and install it. If your system does have the pilot, but has failed to recognize the device, check the components and soldering again.

The next step will be to flash the EP24.hex application using your preferred programmer, connected to the ICSP connec-

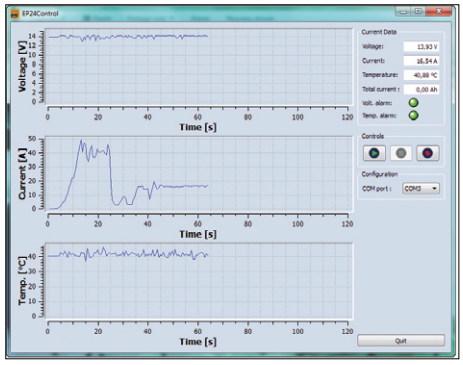


Figure 5. The EP24Control tool developed using Qt from Nokia lets you display the battery voltage, the current it is supplying, and its temperature.

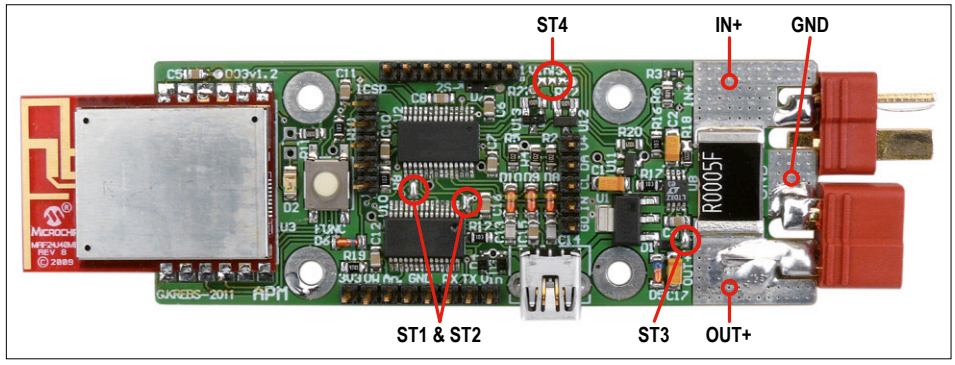


Figure 6. Here's what your board ought to look like once built and tested. The solder bridges to be made are ST1 & ST2 for the USB serial connection, ST3 to power the board from the positive pole of the battery, and ST4 to select the voltage of the I²C extension (3.3 V or the board input voltage).

tor. When the board is booted up, the LED should start flashing not unlike like a heartbeat at a frequency of 1 Hz, indicating normal microcontroller activity.

Now install the EP24control application on your PC (available for download, see links at the end of this article). Run it and select the communication port to which your board is connected. Then click on the button to launch data acquisition (**Figure 5**). The temperature ought to show a logical value, while the voltage and current ought to be virtually zero. If this is the case, it's time to move on to the serious stuff; if not, check the quality of your work again carefully.

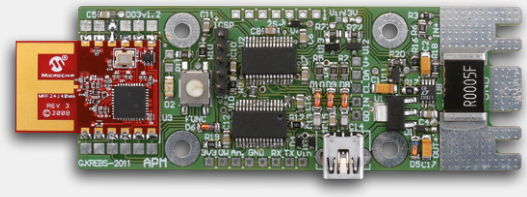
Fit your board with the power connectors you've chosen (soldered wires, PK or DEAN sockets), and then connect a battery and a variable load to the board. Don't forget to connect up the balancing tap too. The voltage value displayed in the EP24control window ought now to correspond to the battery voltage.
Increase the load current gradually up to at least 50 A, if possible, and check that the value displayed reflects the actual situation. If there's a discrepancy of more than 5 %, check carefully the quality of the solder connections to the shunt.

Now fit the MRF24J40 module and reboot the board without connecting up the USB socket. The board is now self-contained, powered by the battery (**Figure 6**).

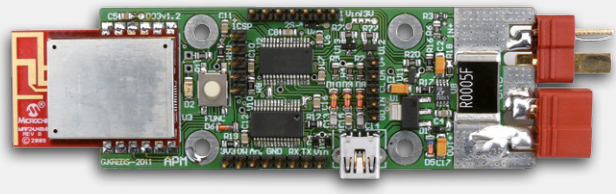
As far as the radio module and the power connections are concerned, there are several options open to you, please refer to the *Options for equipping the board* box about this.

Equipping the board

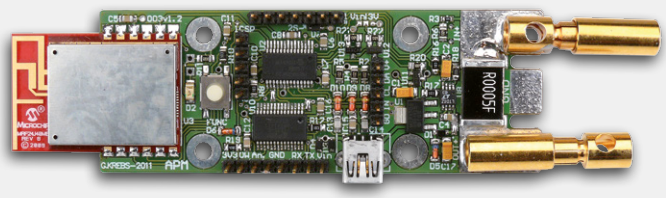
Depending on your own preferences and needs, you can equip your board in various ways. Here are a few examples.



Board fitted with an MRF24J4MA module, offering a range of 100 m (300 ft.). No power connectors are fitted.



Board fitted with an MRF24J4MB module, offering a range of 1,000 m (3,000 ft.) and 'DEAN'-type battery connectors.



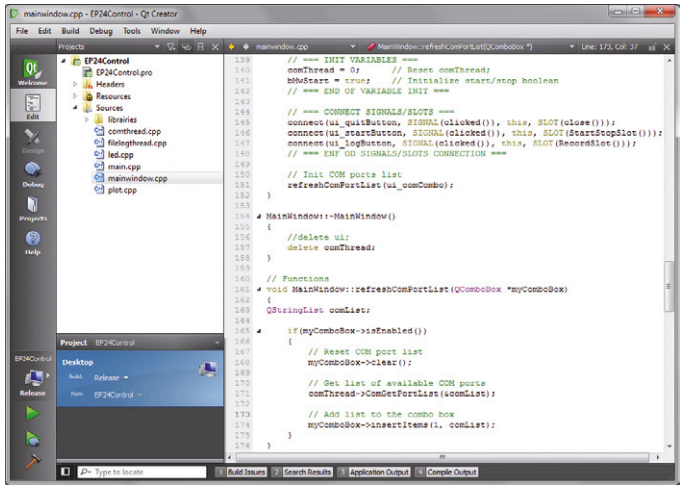
Board fitted with an MRF24J4MB module and magnificent gold-plated 5.5 mm PK sockets, capable of handling 170 A peak.

About the authors

Michel Kuenemann has published several articles in Elektor, including the 2.4 GHz Transmitter & Receiver for Model Aeroplanes [2].
Gilles Krebs is an electronics engineer currently involved in designing aeronautical test beds. An astronomy and photography enthusiast, in his spare time he constructs numerous electronics projects connected with these other interests.

About Qt

EP24control was developed under Qt (pronounced 'cutie'), a C++ class library distributed under a GNU licence. The special feature of Qt is that it is multi-platform (Windows, Mac OS X, and Unix environments under X Window System), just by recompiling the source code. The Qt SDK (Software Development Kit) includes the libraries, the Qt compiler, and the Qt Creator development environment, which sets out to be very user friendly, in particular with a direct link between the functions of the Qt library and the online documentation, very clear and accompanied with numerous examples. Moreover, the Qt Designer tool also included makes it very easy to design its graphics interfaces by easily making the link between the elements of the interface (buttons, indicators, and so on) and the code.
Electronics engineers wishing to make their Qt application communicate with their electronics boards fitted with an RS-232 serial connection will have to use an additional library like QextSerialPort to add the necessary functions to Qt. For technical applications, the Qwt library offers classes that let you display various indicators, controls, and graphs in your application.



Internet Links

Qt: http://qt.nokia.com/
QextSerialPort: http://code.google.com/p/qextserialport/
Qwt: http://qwt.sourceforge.net/

If you have built the 2.4 GHz Transmitter & Receiver [2], you'll be able to see the battery data appear on the fourth line of the display (**Figure 7**). To take advantage of the latest improvements, you should update the software for the transmitter and your receivers. The software package for the 2.4 GHz transceiver (RC-Transceiver-SW-Package.zip file) will enable you to do this.

Notes and warnings

The MRF24J40MB radio module supplied by Microchip has the approvals needed for Europe (ETSI), the USA (FCC) and Canada (IC). Its output power is 100 mW. Channels 20–26, limited to 10 mW in the author's home country, are not used in our system.

This project involves the use of batteries employing lithium polymer technology that can explode and cause fires if they are subjected to excessive voltages or currents. Take all necessary precautions to avoid short-circuits and overloads during testing and while using them in the model.

(110759)

Internet Links

[1] Microchip 2.4 GHz modules: www.microchip.com/wwwproducts/Devices.aspx?dDocName=en027752
[2] 2.4 GHz Transceiver: www.elektormagazine.com/110109
[3] Software and PCB design: www.elektormagazine.com/110759
[4] ZigBee network analysis tool – ZENA: www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en520682
[5] Gilles Krebs's website: www.p67world.com

USB Pseudo Battery

Replaces 1.5-V AAA cells

By **Danny Winkler** (Germany)

In one respect all batteries are identical: sooner or later they die on you! Even rechargeable types are not immune from this problem, as they always need charging again. This makes indoor use inconvenient of devices designed only for portable use, with no provision for connecting an AC adapter. All the same, wherever you find electronic gadgets a solution is not far away. A pseudo-battery with a USB connection provides permanent power.

Many's the time that Danny Winkler got annoyed with his little MP3 player/recorder: whenever he just felt like recording a song he liked … the battery was flat. And the MP3 device had no provision for connecting an AC supply. Such rotten luck. But one fine day he felt inspired to turn his back on wired components and finally start grappling with those cursed tiny SMDs for once. Simultaneously that irritating problem with the portable MP3 player occurred again. "Fine!" he thought to himself; "I can now kill two birds with one soldering iron." He set to work and developed a small circuit that would replace an AAA battery, taking its power from an AC adapter. It was obvious to him that by using SMDs, it had to be possible to create a board so small that it could be tucked inside a battery compartment and replace the actual cell. And success proved him right.

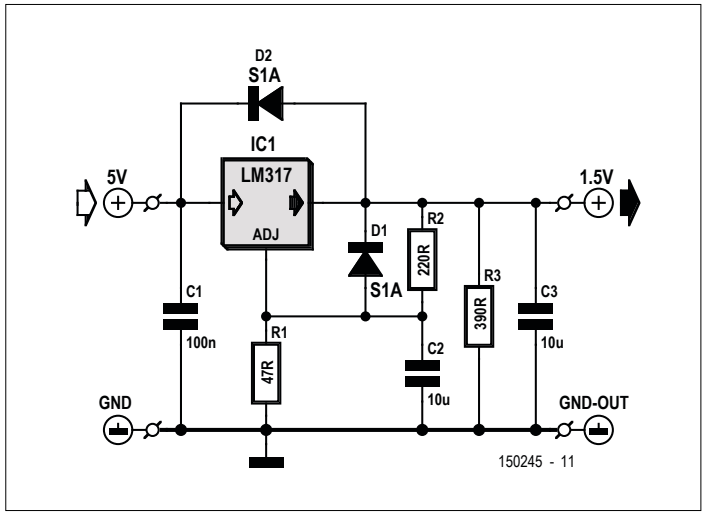


Figure 1. Schematic for the electronic battery replacement.

Component List

Resistors
Default: 5%, 100mW, SMD 0805
R1 = 47Ω
R2 = 220Ω
R3 = 390Ω

Capacitors
Default: 10%, multilayer, SMD 0805)
C1 = 100nF
C2,C3 = 10µF

Semiconductors
D1,D2 = S1A (DO-214AC or SMB)
IC1 = LM317 (SOT-223)

Miscellaneous
PCB # 150245-1 v1.0

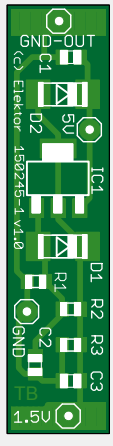


Figure 2. Component placement legend for the AAA format PCB.

AAA substitution circuit

His first notion, of making the PCB tiny enough to have exactly the same space footprint as an AAA battery, was rapidly followed by another. A low-cost USB charger intended for smartphones and suchlike ought to provide a good circuit for the power supply. Small PSUs of this kind don't cost much and deliver 5.2 V with a power rating of at least 1 A. All he needed now as a voltage regulator that would drop the 5.2 V down to the 1.5 V typical of primary cells. And *voila*! The schematic in **Figure 1** had more or less drawn itself!

As Danny had previously achieved good results with the 'full size' version of the LM317 adjustable voltage regulator, he wanted this time to test out the miniature version in the SOT223 package. Using R1 and R2 the output voltage was adjusted to 1.5 V, employing the formula

$$U_{out} = 1.25\ V \times (1 + R1/R2)$$

A couple of capacitors ensure stable operation and the two diodes protect against disproportionate voltages on the pins of the voltage regulator. And that's it.

Ersatz battery

All that remained now was designing a PCB that would enable replacing an AAA battery. **Figure 2** shows the component layout legend. The PCB layout as well as the CAD data can both be downloaded free from the Elektor Magazine website [2]. You can see what the completed prototype looks like in **Figure 3** and **Figure 4**.

There is plenty of breathing space on the PCB and the SMD components are in the reasonably manageable 0805 format, meaning that you're far removed from the grain of dust 'state of aggregation' in which you dare not even breathe until the parts are secured in place with solder. Danny really wants SMD deniers and hardcore fans of wired components to have the courage to try this project, adding that you're not absolutely compelled to use SMD solder paste and similar contrivances. The board can also be assembled with normal solder, although he does just recommend using tweezers. At each end of the PCB you need to solder small scraps of thin copper sheet (phosphor bronze would be even better), to simulate the contact surfaces of the battery (**Figure 5**).

After installing the components and checking the soldered joints as well as seeing that the diodes are the right way round, you should measure the output voltage before risking an active test on an (as yet) living portable device. If you are using one of those low-cost USB chargers as power supply, then you can simply chop off the unused connector at the appropriate end of the USB cable and solder the red and black wires to the matching 5 V and ground pins on the PCB. ⏮

(150245)

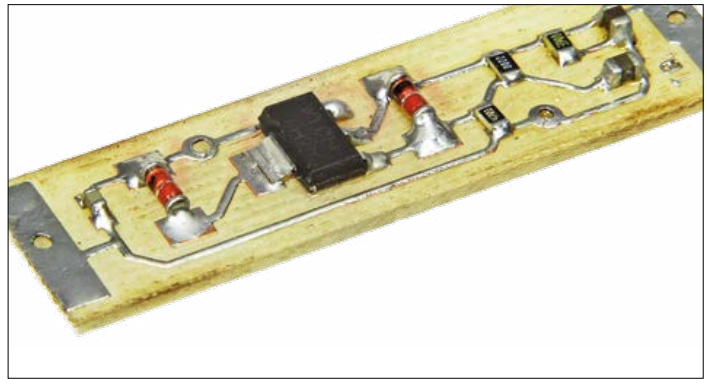


Figure 3. Finished prototype.

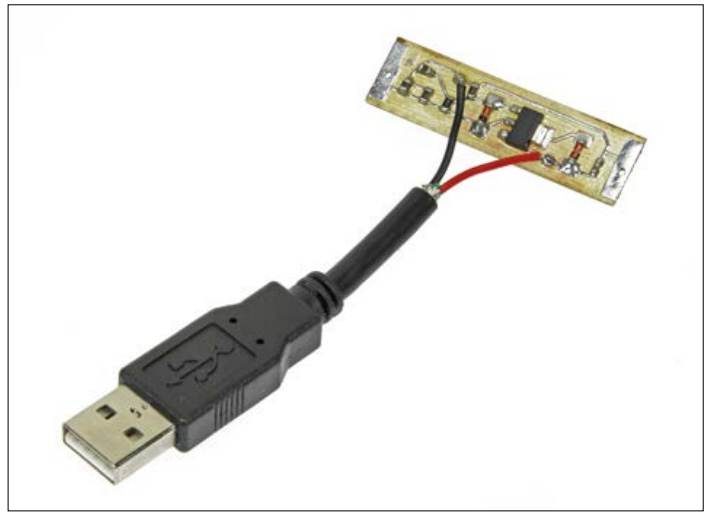


Figure 4. Prototype with USB connector attached.

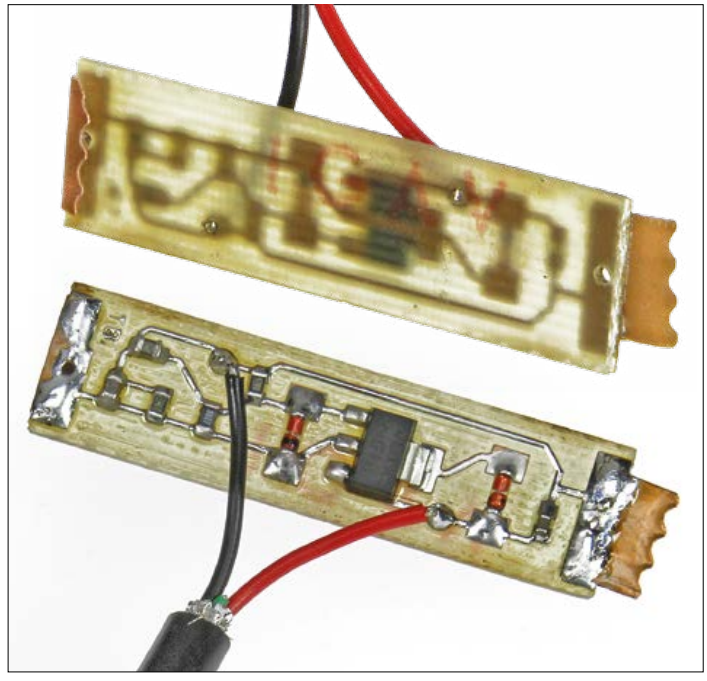


Figure 5. Two small scraps of thin sheet copper (or better, phosphor bronze) soldered to each end of the PCB will ensure good contact in the battery compartment.

Web Links

[1] LM317 datasheet: www.ti.com/lit/ds/symlink/lm117.pdf

[2] www.elektormagazine.com/150245

welcome in your ONLINE STORE

EDITOR'S CHOICE

The CrazyFlie 2.0 is a small 'open source' quadcopter with a steep learning curve — at least, to learn to fly as a noob. CrazyFlie's dimensions (140 mm across the rotor blade tips) do make it a challenge to fly, but once you've mastered the flight control via tablet or smartphone, it's pretty addictive. With a top speed of 30 km/h you should be mindful of the controls, because that's fast! The CrazyFlie was developed by three Swedish designers.

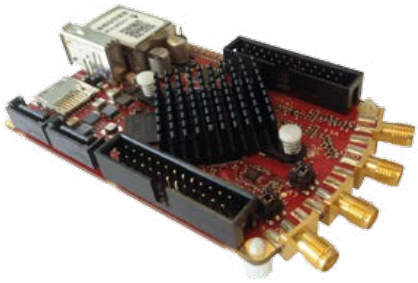
They have set up a dedicated website for support: www.bitcraze.io, where all software right up to a full image of a virtual machine can be downloaded. All parameter and log settings are user adjustable and if you really want to jump in at the deep end, you can play around with the firmware.

Thijs Beckers, Elektor Labs

www.elektor.com/crazyflie

Elektor Bestsellers

1. Red Pitaya
   www.elektor.com/red-pitaya



2. GREEN Membership
   www.elektor.com/green-membership
3. Microcontroller Based Radio Telemetry Projects
   www.elektor.com/radio-telemetry
4. SmartScope
   www.elektor.com/smartscope
5. Elektor Labs Proto Board
   www.elektor.com/labs-protoboard
6. The Bottle Builder
   www.elektor.com/bottle-builder
7. Intel Edison Book
   www.elektor.com/edison-book

Microcontroller Based Radio Telemetry Projects

This book is written for people who want to learn more about radio telemetry applications and microcontroller programming using the PIC18F series of microcontrollers. The design of a radio telemetry based mini weather station is considered as an example system in the book where the developed system can measure the temperature, humidity, atmospheric pressure, carbon monoxide level, nitrogen dioxide concentration, air quality level, wind direction wind speed and more.

member price: £26.95 • €35.96 • US $41.00

www.elektor.com/radio-telemetry

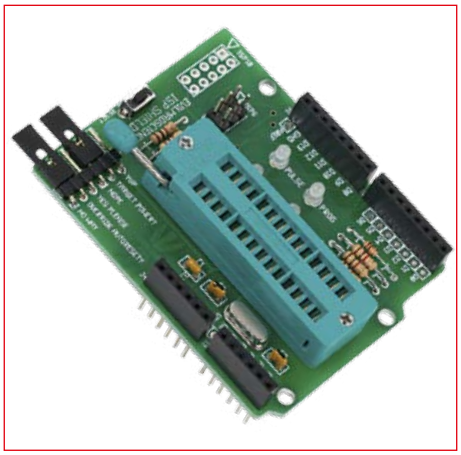
Vanderveen Trans Tube Amplifiers

In this book Menno van der Veen describes one of his research projects focusing on the question of whether full compensation for distortion in tubes and output transformers is possible. A variety of methods have been developed with the aim of attaining this goal. One of them has largely been forgotten: transconductance, which means converting current into voltage or voltage into current. Menno van der Veen has breathed new life into this method.

member price: £21.95 • €26.96 • US $31.00

www.elektor.com/transie

ISP Shield for Arduino



With this simple solder-it-yourself kit you build your own AVR-flashing bootloader-burning butt-kicking ISP Shield that enables your Arduino to be used as an AVR in-system programmer. Use it for burning bootloaders onto blank AVR chips, directly within the Arduino programming environment, either in the provided ZIF socket or on an external target board. Of course, it isn't just limited to bootloaders. There's so much more...

member price: £12.95 • €17.96 • US $21.00

www.elektor.com/arduino-isp-shield

The EAGLE Companion

This book provides the experienced EAGLE user with insight into using some of the more advanced features of EAGLE software. This book is intended as an enduring document covering the more advanced modules, commands, and functions which make up EAGLE. It is hoped that this book will sit on the desk or the bookshelf of the EAGLE user, and provide a quick, succinct reference to assist with more complex applications and uses of EAGLE. Complementing the EAGLE Advanced User's Guide, the EAGLE User Language manual is included in this book in unabridged form, reproduced with permission of CadSoft GmbH.

MEMBER PRICE: £26.95 • €35.95 • US $41.00

www.elektor.com/the-eagle-companion

The EAGLE Companion

Limited time offer for GREEN and GOLD members:
20% Discount plus free shipping!

Elektor Labs Solder Starter Kit

Complete DIY kit with all necessary tools!

Analog Circuit Design

Thump! 3k286 Book Pages on Analog Design!

Solar Panel Voltage Converter for IoT Devices

The IoT Micro Power Supply is a small module intended for powering IoT sensors and other low-power devices. The IoT Micro PSU harvests energy from a small solar cell in a very efficient manner and can even work indoors. A ready-built module of this Solar Panel Voltage Converter for IoT Devices is now available from Elektor at a great price!

member price: £19.95 • €26.96 • US $31.00

www.elektor.com/iot-micro-psu

Piccolino

The Piccolino rapid development board can be used to quickly design microcontroller circuits. The Piccolino has a fast 16f887 PIC microcontroller, voltage regulator, and communications module, and can be easily extended using its four headers. This book contains 30 projects based on the Piccolino. The clear descriptions along with circuit diagrams and photos will make the building of all the projects an enjoyable experience!

member price: £26.95 • €35.96 • US $41.00

www.elektor.com/piccolino-book

RFID Starter Kit for Arduino

This kit provides a comprehensive extension for all your Arduino projects. Not only does it give hands-on access to RFID, the other devices in the kit open up a wide range of possibilities for measurement, readout and control. Now available from Elektor!

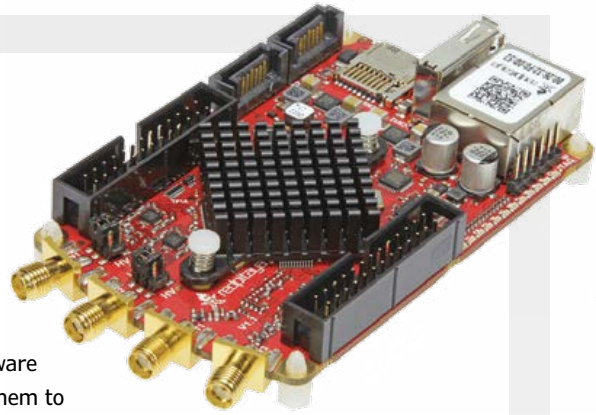
member price: £32.95 • €44.96 • US $51.00

www.elektor.com/rfid-arduino-kit

REVIEW OF THE MONTH

By Luc Henderieckx

Open-source community wakeup call



The hardware and its specifications are impressive, what's lacking however is the software needed to use the Red Pitaya in real-world situations. The few examples of available software work well, but lack almost all settings required to perform useful measurements. I fail to see to why Elektor does not start to develop some applications as this hardware is exactly what the magazine is all about, and use them to kick off a course on programming... There are lots of microcontroller articles in Elektor, but this kind of IT-hardware gets as close as possible to the world of electronics DIY. When will this start or is it beyond the capabilities of Elektor?

Elektor responds:

Brilliant idea! We too are impressed by Red Pitaya and will look at the possibilities of launching some activity in the Elektor.LABS community.

Many thanks, Thijs Beckers | Elektor.LABS

Submit a review of your favorite Elektor product and qualify to win a €100 voucher for redeeming in the Elektor Store.

For further information, please visit www.elektor.com/rotm

Elektor Labs Solder Starter Kit

This solder kit offers the basic equipment that anyone who enjoys hands-on electronics should have in his/her 'shack'. The digital solder station is equipped with electronics which allows for setting the soldering tip temperature between 160 and 480°C. The tip is isolated from the AC line and operates on 24 VDC. The station accurately displays the current and set temperature of the tip.

The DT-830B digital voltmeter (DVM) measures voltage (AC and DC), current and resistance and has settings for diode check and transistor hfe measurement.

Several other tools are included in this kit, like a cable stripper, desoldering tool, a small screwdriver annex voltage tester, and much more.

MEMBER PRICE: £97.95 • €134.96 • US $151.00

www.elektor.com/elss

The EAGLE Companion

Limited time offer for GREEN and GOLD members:
20% Discount plus free shipping!

Elektor Labs Solder Starter Kit

Complete DIY kit with all necessary tools!

Analog Circuit Design

Thump! 3k286 Book Pages on Analog Design!

Raspberry Pi Advanced Programming

This book is about advanced programming of the Raspberry Pi computer using the Python programming language. The book explains in simple terms and with examples: how to configure the Raspberry Pi computer; how to install and use the Linux operating system and the desktop; how to write advanced programs using the Python programming language; how to use graphics in our programs and how to develop hardware based projects using the Raspberry Pi.

member price: £29.95 • €40.46 • US $46.00

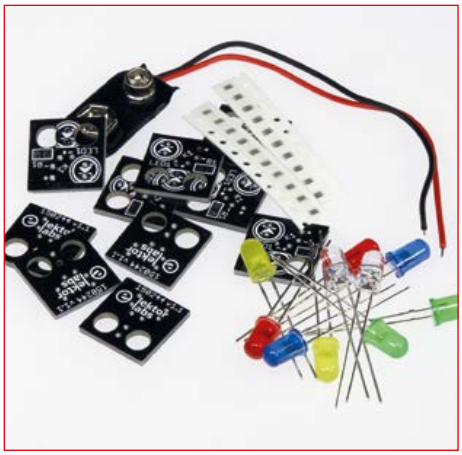
www.elektor.com/rpi-book

Mystery Product

We dare you ...

www.elektor.com/mystery

Lego LED Kit



The LEGO™ LED circuit board is a small PCB compatible with the famous bricks from Denmark that can hold an LED, a current limiting resistor and an optional connector. It is great for adding lighting effects to your constructions and it is 100% customizable. This kit contains 10 circuit boards, 10 LEDs of different colors, 10 current limiting resistors of 390 ohms and 10 current limiting resistors of 820 ohms. A 9 V battery clip is in the kit too.

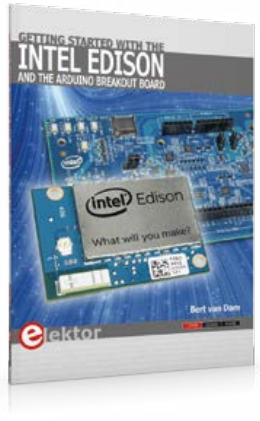
member price: £6.95 • €8.96 • US $11.00

www.elektor.com/legoled

RECOMMENDED 4 SHOPPING

- GOLD Membership
  www.elektor.com/gold
- BL600 e-BoB
  www.elektor.com/bl600
- DVD Elektor 1980 through 1989
  www.elektor.com/eighties
- Raspberry Pi Maker Kit
  www.elektor.com/rpikit
- Python Programming and GUIs
  www.elektor.com/python
- T-Board Audio Amplifier
  www.elektor.com/t-b-audio
- Retronics
  www.elektor.com/retro
- Java Cookbook
  www.elektor.com/java
- Elektor Labs Ruler
  www.elektor.com/ruler
- Electronic Dice
  www.elektor.com/dice

Getting Started with the Intel Edison

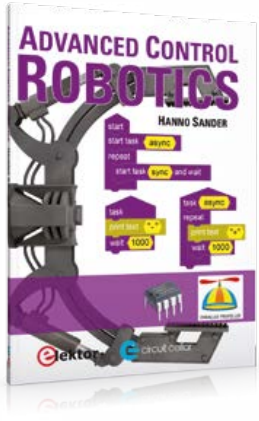


This book is a must have for all those with an active interest in the Internet of Things. 'Getting Started with the Intel Edison' focuses its attention on the Edison, a tiny computer, the size of a postage stamp, with a lot of power and built-in wireless communication capabilities. In 128 pages, renowned author Bert van Dam helps readers get up to speed with the Edison by making it accessible and easy to use. Explore the wonderful world of the Intel Edison now!

member price: £21.95 • €26.96 • US $31.00

www.elektor.com/edison-book

Advanced Control Robotics

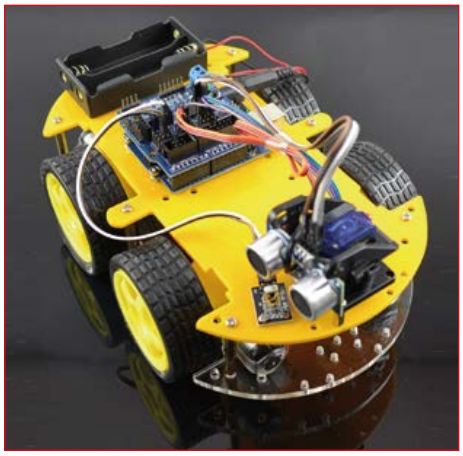


Hanno Sander's Advanced Control Robotics simplifies the theory and best practices of advanced robot technologies. You're taught basic embedded design theory and presented handy code samples, essential schematics, and valuable design tips (from construction to debugging). The book is intended to help roboticists of various skill levels take their designs to the next level with microcontrollers and the knowhow to implement them effectively.

member price: £26.95 • €35.96 • US $41.00

www.elektor.com/robotics-book

Robot Car Kit for Arduino



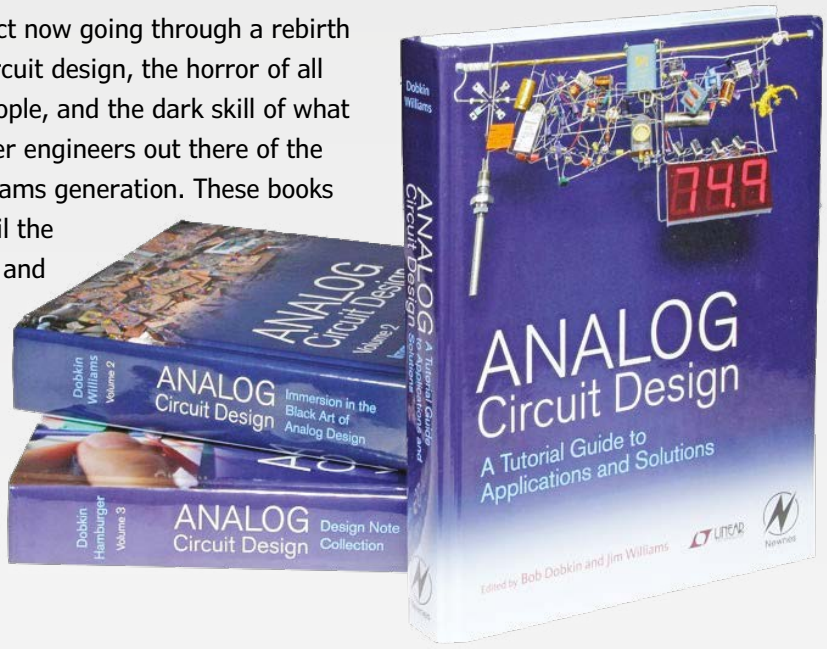
This four-wheel-Drive Smart Car can be controlled with an Android Bluetooth device or by infrared (included). Driven by an Arduino UNO the robot can avoid obstacles or follow a track. A mobile development platform, the 4-Wheel Drive Smart Car is great for study purposes and of course for the fun of it! The kit is loaded with expansion possibilities that provide sheer endless fun and creativity.

member price: £65.95 • €89.96 • US $101.00

www.elektor.com/arduino-car-kit

Analog Circuit Design Bundle

Three world renowned and class leading books on the art of Analog Design using the tutorial approach are now available from Elektor. Never before have we seen such thick and heavy books on a subject now going through a rebirth like no other: analog circuit design, the horror of all microcontroller-only people, and the dark skill of what some say are a few older engineers out there of the Bob Pease and Jim Williams generation. These books are guaranteed to unveil the darkest sides of opamp and discrete part based circuit design, as well as designing for low noise and total analog response, something no microcontroller can ever hope to achieve.



Be smart. Buy all three volumes as a set and take advantage of a 10% bundle discount!

MEMBER PRICE: £129.95 • €179.00 • US $201.00

www.elektor.com/acd-bundle

The EAGLE Companion

Limited time offer for GREEN and GOLD members:
20% Discount plus free shipping!

Elektor Labs Solder Starter Kit

Complete DIY kit with all necessary tools!

Analog Circuit Design

Thump! 3k286 Book Pages on Analog Design!

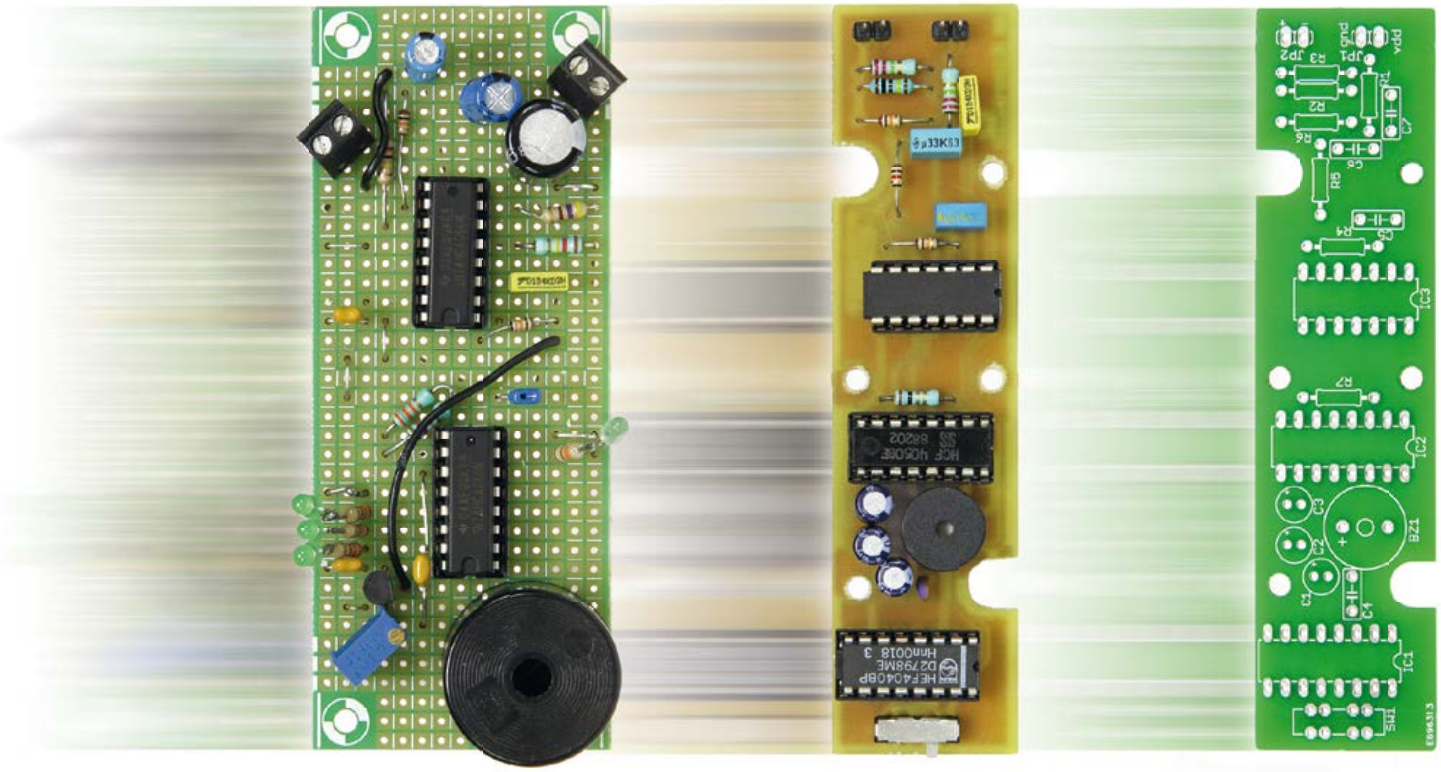
LEARN DESIGN SHARE

elektor PCB Service

if offered in collaboration with EURO CIRCUITS

Generate your own PCB
using the Elektor PCB Service

Affordable **High Quality** **Reliable**



The Elektor PCB Service is the most extensive fully customized service for printed circuit board production in Europe. With convenient online tools allowing you to visualize and analyze your design before you order and pay .

- For beginners, there is the **NAKED-Prototype Service**:
  This produces single and double-sided PCBs without solder masks.

- For a more advanced service, there is the **PCB Visualizer** that shows you how your PCB will look after production, with a **PCB Checker** performing a DRC for you and the **PCB Configurator** that lets you customize your order details.

Smart menus and select options guide you through the ordering process. You can see in advance exactly what our machines can produce so there won't be any surprises!

So start your next project here:

www.elektorPCBservice.com

Welcome to the **SHARE** section

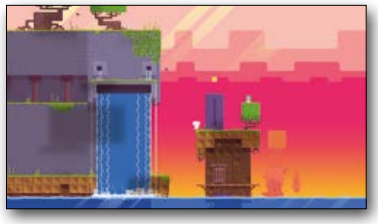
By **Jaime González-Arintero** jaime.glez.arintero@eimworld.com

Play Hard

Beware, *Cheap Philosophy Ahead*! Please let me explain. Last week I was watching a short documentary exploring the reasons why humans enjoy playing games. The conclusion was, more or less, that life is a game on its own, but since the goal is not really clear to everyone, we seem to need milestones to reach. In the absence of milestones, individuals will set them… right, individually. Maybe.
As most guys of my generation, I used to play videogames when I was a kid. Since I didn't find high school very amusing, the games were definitely more rewarding. Back then, videogames were acutely expensive, so it was considered normal to share them among friends. Computer games were hacked and shared very often and no wonder they came with more sophisticated copy protection techniques all the time. Internet access was for the happy few, but you could always visit the library, find guidelines at forums, ask someone via IRC, and once back home, try different things offline. Then it was only "You vs. the Game." I'm not especially proud of that, but admit to having done it too. Nowadays I gladly pay for everything, hard copy or digital. Back then, we were kids. And it was challenging and tricky. It was big fun.
Most games (and in general commercial software too) could be hacked by means of a cloned CD, or a *mounted* drive (by means of a virtual drive software, e.g. "DAEMON Tools"). Other games required more steps like overwriting registers, files, DLLs, using key-gens, and so on. These were widely known, and you could easily find step-by-step guides in a fishy .zip file downloaded somewhere. You just had to follow the instructions, and *voilà*: game up and running. Sometimes things were messier, and (for example) a key was internally generated by the installer based on your computer's ID or any other serial number locally available, and "embedded" directly in the executable. If you were lucky and the application itself was not encrypted, such keys could be retrieved using a hex editor. Even though you were just following steps detailed by someone else, it was impossible to suppress the feeling of being the ultimate hacker! (or cracker, I should say)
The funny part of the story is that once games were successfully cracked, and properly working in my (offline) machine, I rarely played them. Sometimes I didn't even adjust the settings, nor finish the tutorials. Compared to doing the crack playing there was zilch excitement in playing. It turned out that "making it work" was the actual game.
After hitting the university, things started to make more sense to me and I could see "goals" everywhere, so videogames dropped in importance, as did the act of cracking. At some point, I felt that designing my own (crappy and simple) parking detector, or a brightness controller circuit, was indeed way more amusing than shooting random enemies.

Braid is a masterpiece that opened up parts of my brain hitherto unused. Concepts like propagation delay were never depicted so beautifully.

If you do not know **Portal**, you should. It's a wonderful, mind-bending title that will change the way you look at the mirror every morning.



Fez makes you solve puzzles and find your way alternating between 2-D and 3-D perspectives. You did that before, right? Well, this way, you didn't.

Play Harder

But hey, what I really wanted to say is: it's not a complete loss. My disappointment with videogames didn't last forever, and I've always wanted to recommend a few titles to my fellow geeks. If you still feel like playing videogames and you're up for some brain exercise, I compiled some mind-expanding games I love in a brief list. And no, I'm not talking about "The Incredible Machine," or mere puzzle games that obviously resemble circuit design. You probably had a tough day routing that PCB, but anyway: there's always time left for more challenges!
Wanna see more screenshots, larger images, active links... Or you just feel like commenting and sharing this article with friends? Easy. Navigate to [http://po.st/playharder].

German Brickwork

Rapid e-prototyping with TinkerForge's Bricks and Bricklets

By **Clemens Valens** (Elektor.Labs)

Prototyping a complex electronics control system in a short time is possible by using ready-made modules specifically designed for this purpose. Today so many of these modular systems exist that it is difficult to spot the one that's right for you. Maybe the Brick & Bricklet approach from TinkerForge is what you are looking for? We tried it for you.

The test system used in this article consisted of a DC Brick motor driver module and two Bricklets: Pushbutton and 'Rotary Poti' (potentiometer).

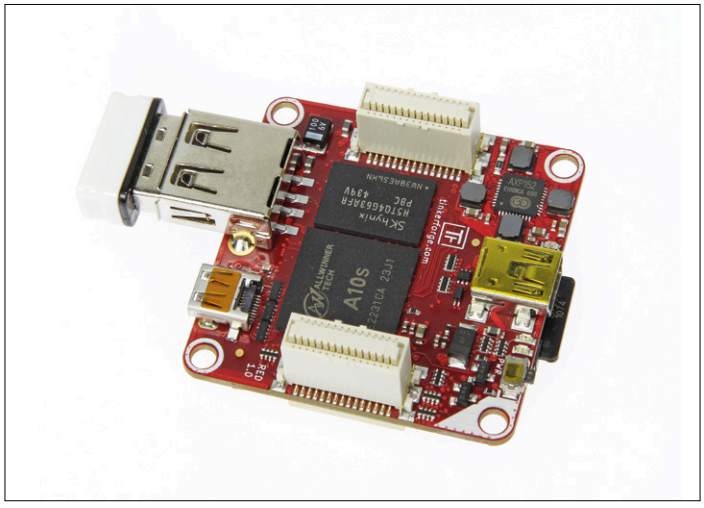
Rapid Prototyping seems to be a booming market, looking at the sheer number of tools that have been and are being developed. Ranging from low-cost hobbyist approaches like Arduino to the highly professional tools from big-gun multinationals, a system tailored to your application and budget very likely exist. All these tools have the same objective: to make prototyping electronics systems as easy as possible.

One of the rapid prototyping systems that landed on my desk came from TinkerForge [1], a fledgling German company that's trying hard to go all the way. Although I am hard to impress, I must admit to liking the build quality of the modules that came out of the box.

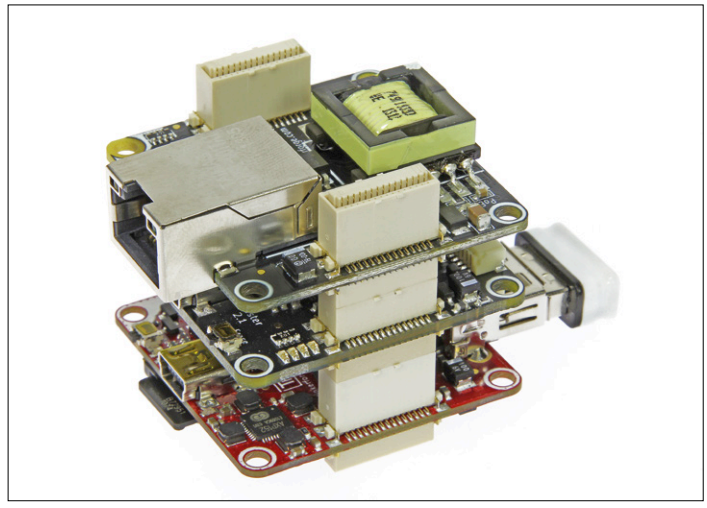
What is it?

TinkerForge modules are electronic building blocks that can be stacked and wired up to create a complex system. The modules by themselves do nothing, it is up to the user to write software to make the system work. This software is a PC application that can be written in a number of languages, from C and C++ right up to LabVIEW and Mathematica. Modules come as Bricks, Bricklets, extensions and accessories. A Brick is a microcontroller module with a specialization (e.g. a motor driver); a Bricklet contains a sensor (e.g. a potentiometer) or an actuator (e.g. a LED). A typical system consists of at least one Brick to which Bricklets may be attached. The PC

application communicates with the Brick, it sends commands to it and requests data from it; the Brick communicates with its Bricklets, if any.
It is important to understand that a PC is invariably involved, and it's the PC running the application, not the Brick. Unless you invest in a RED Brick — a Brick capable of replacing the PC, but more on that later…
A Brick can have a limited number of Bricklets connected to it, the exact number depends on the Brick, but Bricks may be stacked to add Bricklet connections. Bricklets on the other hand cannot be stacked and are connected with cables. That's logical, because they usually have to be mounted somewhere in the system, close to a motor, on a control panel, or similar. Stacking Bricks is also useful for adding functionality to a system.
Extension modules allow the interconnection of Brick stacks as well as the control of a Brick or a stack over a network — great when the system is installed in a remote or inaccessible place but must be controlled from a control room.
Finally, accessories are items like connectors, nuts & bolts, displays, etc. that are needed to complete a system and that may be obtained from other sources.



The RED Brick with Wi-Fi dongle and SD card.



Stack 'em! RED Brick (bottom), Master Brick (middle) and an Ethernet extension module (top).

Who is it for?

TinkerForge modules are aimed at people who want to build something that needs electronics to operate, but who lack time, skills or a desire (all combinations apply) to design their own electronic hardware. The sub-species is called Makers these days, or tinkerers, which explains the name TinkerForge.
Although the user is not supposed to have deep knowledge of electronics, some programming skills are required — actually, the more the better. Detailed tutorials are available to help you getting started, seasoned programmer or not.
Applications can be written in C/C++, C#, Delphi/Lazarus, Java, JavaScript, LabVIEW, Mathematica, MATLAB/Octave, Perl, PHP, Python, Ruby, Shell and Visual Basic .NET. Mobile platforms are included too so you can also use C/C++ for IOS, C# for Windows Phone and Java for Android. I think there are only few products in the world that support more programming languages.

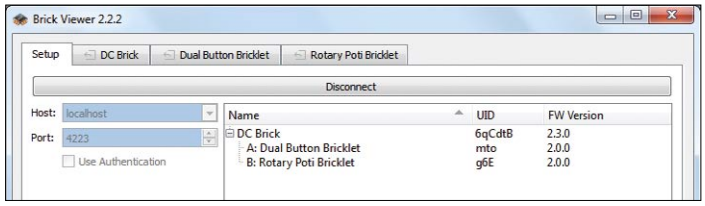
What can you do with it? Is it better?

To answer these two questions it best to play with the system. Here's what I did.

I started with the *First Steps* tutorial from the website. It employs the DC Brick and so did I. This Brick is not a power supply for a stack as I had assumed, but a Brick with an on-board DC motor driver. Connect the Brick to the PC using its mini USB port; connect a DC motor to the spring connectors (beware, they are rigid). Now some software must be installed:

- **Brick Daemon** (brickd) — a little program running in the background that handles the communication between the Brick and the user application.
- **Brick Viewer** (brickv) — a nifty tool showing the configuration of the system, which Bricklet is connected to which Brick, etc. It also provides interfaces to control the Bricks and Bricklets.

Brick Daemon starts automatically when you connect a Brick, while Brick Viewer must be launched manually. Click its Connect button to make contact with your Brick system. Now the Brick should show up in the tree and a user interface tab for it should be available too. The viewer also shows the Brick's UID which you will need when you write your own applications. It is the handle by which you address an object in your system. By activating the user interface tab of the DC Brick in the Viewer I was able to control the speed and direction of my little motor.
Next step is to check if you can achieve the same thing by writing your own application. In what follows I assume that you know the basics of the programming language you picked, and that the necessary compilers/interpreters are ready installed on your computer.
Kick-off is installing the API bindings for your programming language. Detailed instructions on how to do this for your programming language are available on the TF website. I chose to use Python 2.7, Python 3 is supported too.
To test if everything works, download the examples for the pro-



The Brick Viewer connected to a DC Brick that has two Bricklets connected to it. Note the UIDs you will need for programming.

gramming language of your choice. Since I was playing with the DC Brick and Python, I tried the script `example_configuration.py`. It must be edited to work with your setup and you need Brick Viewer for that. Open the script in an editor and edit the parameters HOST, PORT and UID to match the information found in the Viewer. HOST and PORT are probably correct, but not the UID which is unique for every module, so change it and save the file. If you run the script, the motor will start spinning. When you press <Enter> the script is halted, but the motor keeps running. You can stop it manually by entering the following commands on the Python command line:

```
>>> ipcon = IPConnection()
>>> dc = DC("6qCdtB",ipcon)
>>> ipcon.connect("localhost",4223)
>>> dc.set_velocity(0)
```

The first three lines mark the steps that have to be taken to establish a connection with a Brick (note the method of using the HOST, PORT and UID parameters); the last line stops the motor. To clean up after yourself, issue the command

```
>>> ipcon.disconnect()
```

When experimenting with the DC Brick do not forget that you must enable the motor before anything happens (why?). Use the command `dc.enable()` to do so.
If you have gotten this far, then you are ready to explore the rest of the TinkerForge products.

Bricklets

There exists a substantial number of Bricklets, and there will be more in the future. Examples are the Analog In and Analog Out Bricklets, the Barometer, Distance, GPS, IO16, Joystick Bricklets, and many more. As already said, Bricklets connect to Bricks and only to Bricks, excepting the RED Brick: a special Brick that does not accept Bricklets. The Master Brick can have up to four Bricklets attached to it, the DC Brick only two. The connection between the Brick and the Bricklet is through a 10-way cable that is hard to manufacture yourself so it is easier to buy; it comes in several lengths. Disconnect the power to your Brick before connecting one or more Bricklets.
I connected the potentiometer Bricklet "Rotary Poti" (German origins, remember?) and the Dual Button Bricklet to my DC Brick. After powering the Brick and reconnecting to it in the Viewer, the Bricklets showed up too, hanging under the DC Brick. Also a tab for each Bricklet was added. The potentiometer tab shows a graph of the potentiometer's position (in degrees) and the dual button tab allows you to activate the LEDs on the Bricklet and control the way they work — all pretty neat. With the help of the examples provided it is easy to create a simple application that controls the motor speed with the pot, and the pushbuttons to enable and disable the motor. The LEDs in the pushbuttons can provide visual feedback. Strangely though, the pushbuttons seemed to work the wrong way around or maybe I got something wrong. My example script can be downloaded from the article's webpage [2].

Extensions

With Bricks and Bricklets you build a stack that can be used to do something, like controlling a motor as I just did. The stack is connected to a PC that runs the Brick Daemon and the application program. But how do you control a stack that is not sitting next to your computer? This is where Tinkerforge Extensions come in. These modules allow two stacks to communicate and make the whole system look as if everything is right there on your desk. Currently there are four types: Ethernet, Wi-Fi, RS-485 and Chibi (an open-source wireless 802.15.4 communication stack that failed to achieve the popularity the developers had hoped for).
Extensions require the Master Brick — the DC Brick won't do. This is only a financial problem because you can stick a Master Brick on a DC Brick. The Ethernet Extension also works with the RED Brick. I tried the Ethernet Extension with the Master Brick and although everything seemed to be fine & dandy I could not figure out how to use it. The TF website says you can use your phone or some other mobile device to control the stack without the need for the Daemon, but there is no clue on how to accomplish this. Documentation is lacking here and so are the examples.
I did discover a little trick to figure out the IP address of the Extension when it is in DHCP mode, i.e. when it gets an IP address assigned 'by the network'. Just change the Address box temporarily to Static IP and it will indicate its current configuration. Write it down and change the setting back to DHCP. Do not save the configuration.
Since I didn't have any other Extensions to try I left the issue for what it is.

RED Brick

So far every Brick stack required a PC to run the application, which is not always desirable. Hence TinkerForge's *Rapid Embedded Development* (RED) Brick, a special Brick that can run the PC application. Actually, it is a small Linux (Debian-based) computer that executes applications written in most of the supported programming languages (not the proprietary languages LabVIEW, Mathematica and MATLAB). The program can be written, debugged and tested on the PC. Once ready it can be transferred to the RED Brick where it will be executed without any changes. Programs can be scheduled (execution on boot-up, every hour, etc.) and monitored. It is even possible to execute multiple programs simultaneously.
If you have ever played around with Raspberry Pi or other embedded Linux systems you may the complexity and frustrations of their configuration. TinkerForge solved this for their RED Brick by integrating an extensive configuration panel in the Brick Viewer. Thanks to this GUI it's a breeze to set up a Wi-Fi connection (just add a USB Wi-Fi dongle), and to see what's happening on the board. There is also a console where

command-line warriors can enter Linux commands. Shutting down or rebooting the board is also handled here, but in a less obvious place: the System button in the right upper corner of the RED Brick tab. I say every embedded Linux board should have a configuration tool like this.

Once you have the RED Brick connected to a network, either with a Wi-Fi dongle or by using the Ethernet Extension, and you have figured out its IP address, you can connect to the RED Brick web interface that shows information about the installed programs. You can replace this interface with your own; web applications written in HTML/JavaScript, Python and PHP are supported.

The RED Brick also has an HDMI interface and if you connect a suitable display, the Brick will boot into a GUI. Our test kit came with a touch screen, but the touch part did not work even though the touch LED on the display said that everything was all right. Maybe a configuration to do somewhere?

Stay in touch

When you start playing with Bricks and Bricklets it is highly recommended to visit the General Discussion forum [3] on a regular basis (also available in German) because this is the place where software and hardware updates are announced. There is quite a lot of activity going on. When I embarked on this article I downloaded Brick Viewer 2.2.2, when I looked again two months later the version was up at 2.2.5 with many Brick firmware updates in between.

Open source

Everything related to Bricks, Bricklets and Extensions is open source. Consequently you have access to the source code of everything, and you can consult the schematics of every module.

Conclusion

Now that we know what we can do with the TinkerForge modules we can try to answer the one remaining question: Is It Better? It depends on your application. Personally I was impressed by this rapid prototyping system. It supports so many programming languages that anyone who has ever accomplished some programming in his/her life should be able to get cracking. There is plenty of well-written documentation, and examples galore. Fair enough, there are a few imperfections in the Viewer and yes, it is true that when it comes to the more advanced features you may feel a bit left out, but there is always TinkerUnity.org where you can interact with other users and the TinkerForge team itself. And did I say that the quality of the components is second to none? Hats off to TinkerForge!

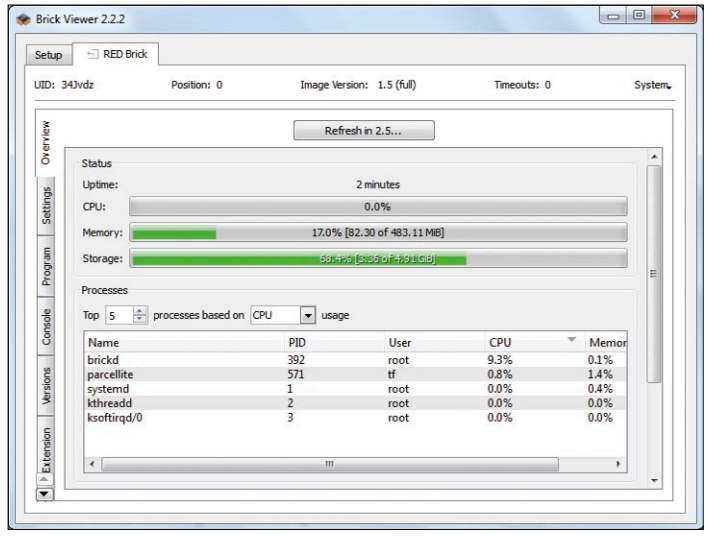
(140543)

Web Links

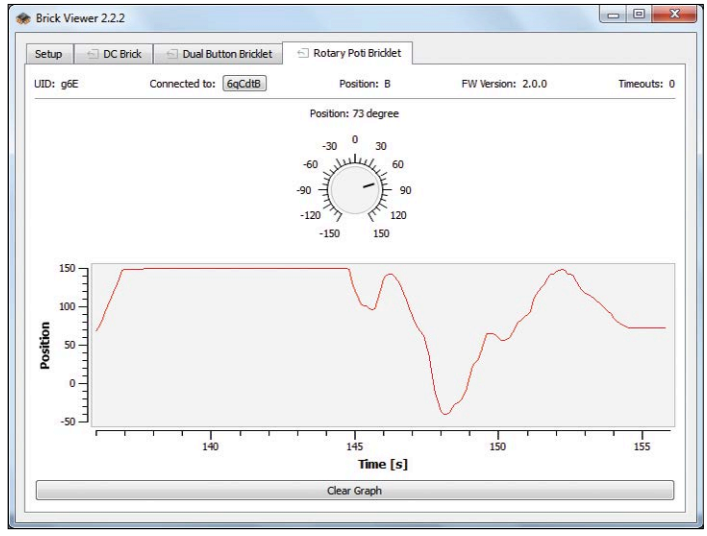
[1] www.tinkerforge.com

[2] www.elektormagazine.com/140543

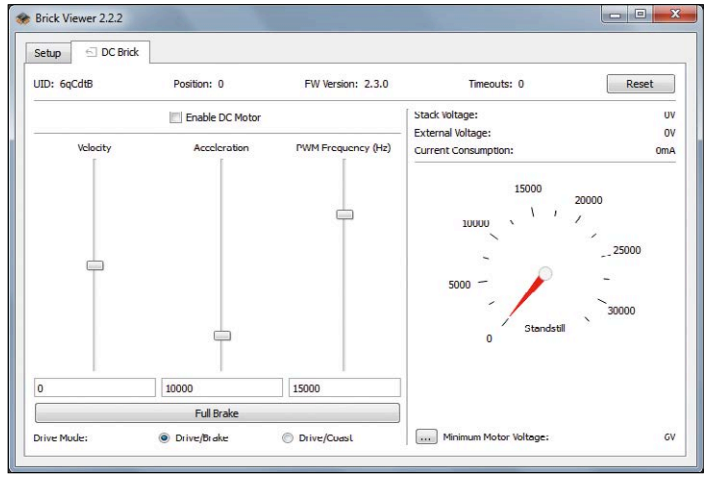
[3] www.tinkerunity.org/forum/index.php



The Linux configuration interface of the RED Brick in the Viewer. The list of active programs includes the 'brickd' Brick Daemon.



Here you can view what the potentiometer Bricklet is doing.



The user interface for the DC Brick in the Viewer allows you to control the speed and direction of the motor.

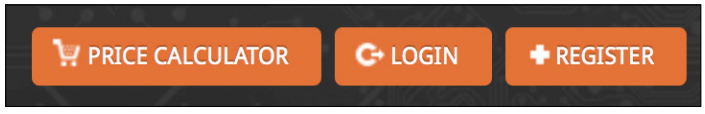
Selected Gerber Files from the Labs

By **Thijs Beckers** (Elektor Labs)

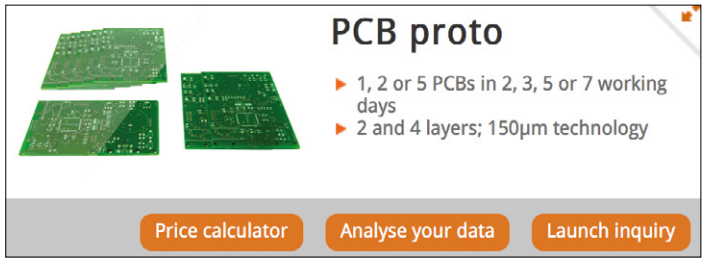
Recently as a trial a few 'Gerber' PCB manufacturing files became available through the Elektor online Store. What can you do with them and how does it work?

Gerber files are a set of documents that describe how a PCB should be manufactured. They contain descriptions of the various layers of the PCB, like bottom and top copper layer, silk screen layer, solder mask layer, drill information and descriptions of any extra layer you can think of that is needed for the production of a PCB. The data is saved in a 2D binary vector image open file format. Currently the "Extended Gerber or RS-274X" is the standard industry format. Although a deprecated "Standard Gerber" format exists, it should be avoided. The Gerber file format was originally developed by the Gerber Systems Corp., a division of Gerber Scientific, founded by Joseph Gerber — hence the name. It was designed to drive their vector photo plotters for the PCB industry in the 1960s and 1970s. As Gerber was the market leader at the time, its format became a *de facto* standard for the industry. And it still is.
With the decision to provide access to Elektor Labs' design data files and make a portion of our highly appreciated PCB manufacturing files available through our online Store (www.elektor.com), we hope to stimulate the number of Elektor projects being replicated and so support the booming maker community. With the Gerber files available at cost through our Store you are now free to have a company of your choice manufacture Elektor PCBs exactly as Elektor Labs intended them to be. After all, we use the same set of files for our own prototypes and production. And because of this, you should not have any troubles having 'our' PCBs manufactured professionally by any PCB pooling service.
Importantly, the new Gerber file service does not and will not cover all projects published in Elektor. The selection of PCBs supported is governed by Labs. At the time of writing the service is being rolled out and it is best to search for "Gerber" on the Store website; the items returned are what's available at the time of viewing.
Due to the vast number of manufacturers out there we simply cannot show how the ordering works with each and every one of them. As a quick start though we'll guide you through the ordering process with our 'Trusted & Approved' supplier Eurocircuits, which should be similar with your preferred manufacturer (although the services of Eurocircuits are quite extensive and options may not be available at other PCB pooling manufacturers). ⏮

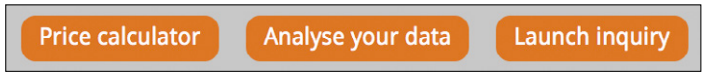
(150342)



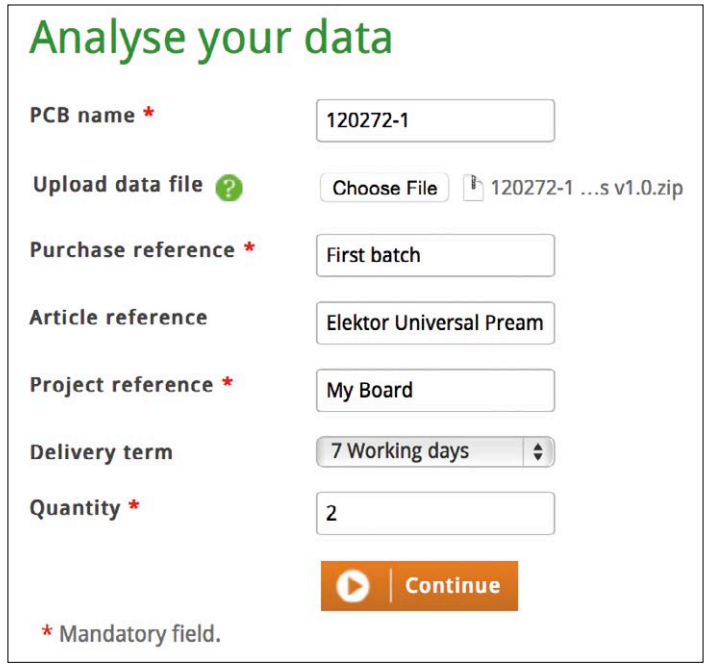
Step 1. Go to eurocircuits.com and select "Price Calculator"



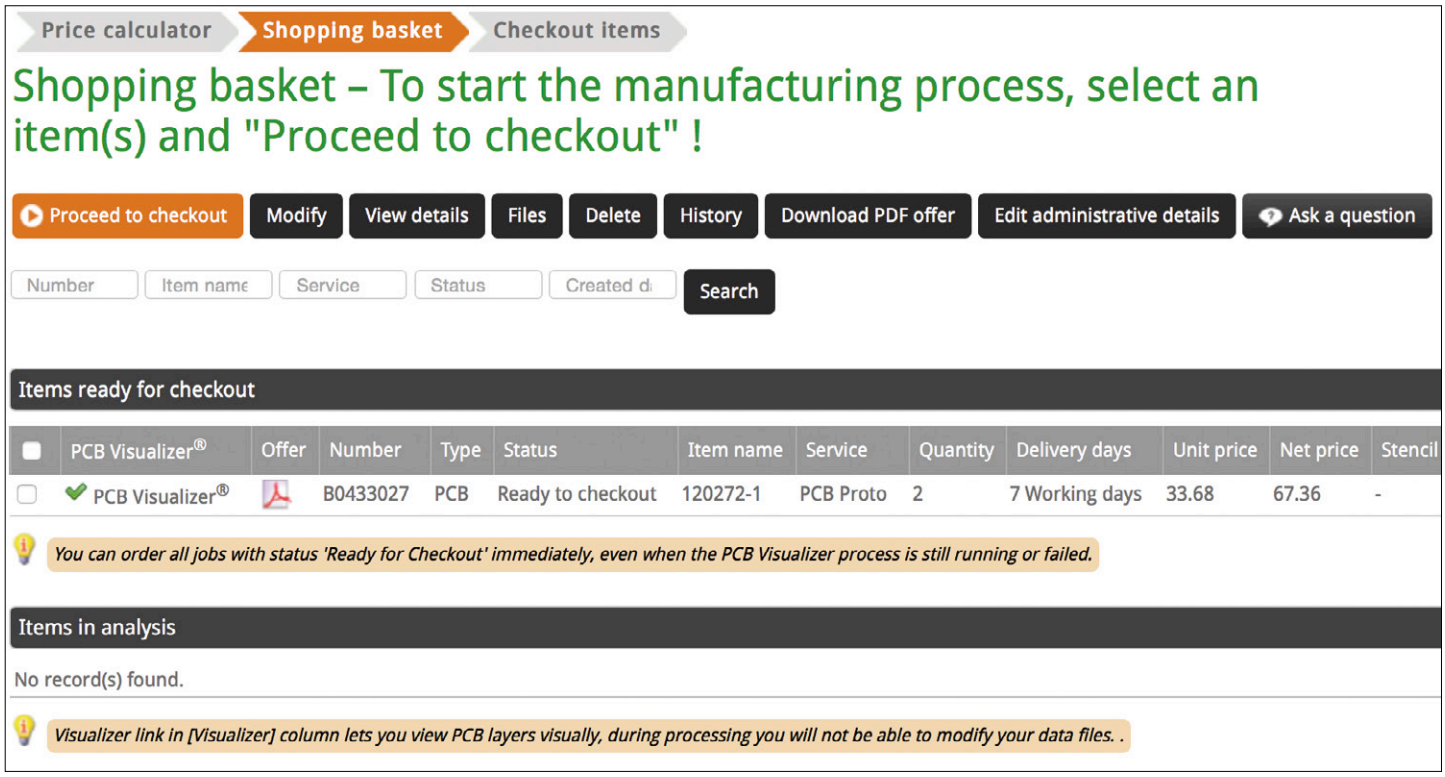
Step 2. On the next page, select "Analyze your data". For single boards you can use the "PCB proto" or "NAKED proto" service.



Step 3. For Eurocircuits to be able to combine the Gerber files you are about to upload with your personal data (like shipping address and billing), you need to register (or if you're already registered, log in).



Step 4. After logging in, the website takes you to a form where you submit your project and PCB data and upload your Gerber files. Before uploading, compress them into one zip file.



Step 5. After submitting your data, clicking 'Continue' takes you to an overview of your submitted data. Eurocircuits offer you extensive preview options for your PCB using their 'PCB Visualizer'. This tool lets you check beforehand for any possible hiccups and difficulties in the production process.
Now tick the designs you wish to proceed with (you can add multiple boards to your order and review them before checking out) and click 'Proceed to checkout'.

Step 6. The final things for you to complete the order are to check your shipping and invoice addresses, and click 'submit'. Then your professionally produced PCB(s) should be on their way to you soon!

Eurocircuits' PCB Visualizer shows your PCB's production details and lets you spot and correct issues beforehand

It's been a hot, hot summer

at dot-Labs, at least…

While summer is very much a local experience, electronics is global. So although we cannot claim that summer's been warm all over the globe this year, as far as soldering irons and Elektor Labs members are concerned, it's been a scorcher. To prove it, here's another compilation of projects that have been evolving at .Labs and already look amazing. And this raises the question: will Autumn be even hotter? Who knows!

http://po.st/astroclock

It's Time to Know a Little More

This astronomical clock provides a ton of information based on the coordinates of your current position, and it shows everything using no fewer than four displays! Greenwich time, zodiac sign, time, day, city, season, humidity, temperature, atmospheric pressure… it even changes its appearance depending on the day cycles. Control freaks — this clock is for you!

Introducing… Our New Drummer. He's Analog.

Sure, music production software is convenient, but moving sliders on a screen will never match the feeling of adjusting your own beats and twiddling with real knobs. This beatbox integrates 16-32 step sequencers with an additional MIDI control. We all know that analog is where it's at, so the 12 different sounds are generated by an analog source. Feeling nostalgic for the '80s?

http://po.st/beatbox

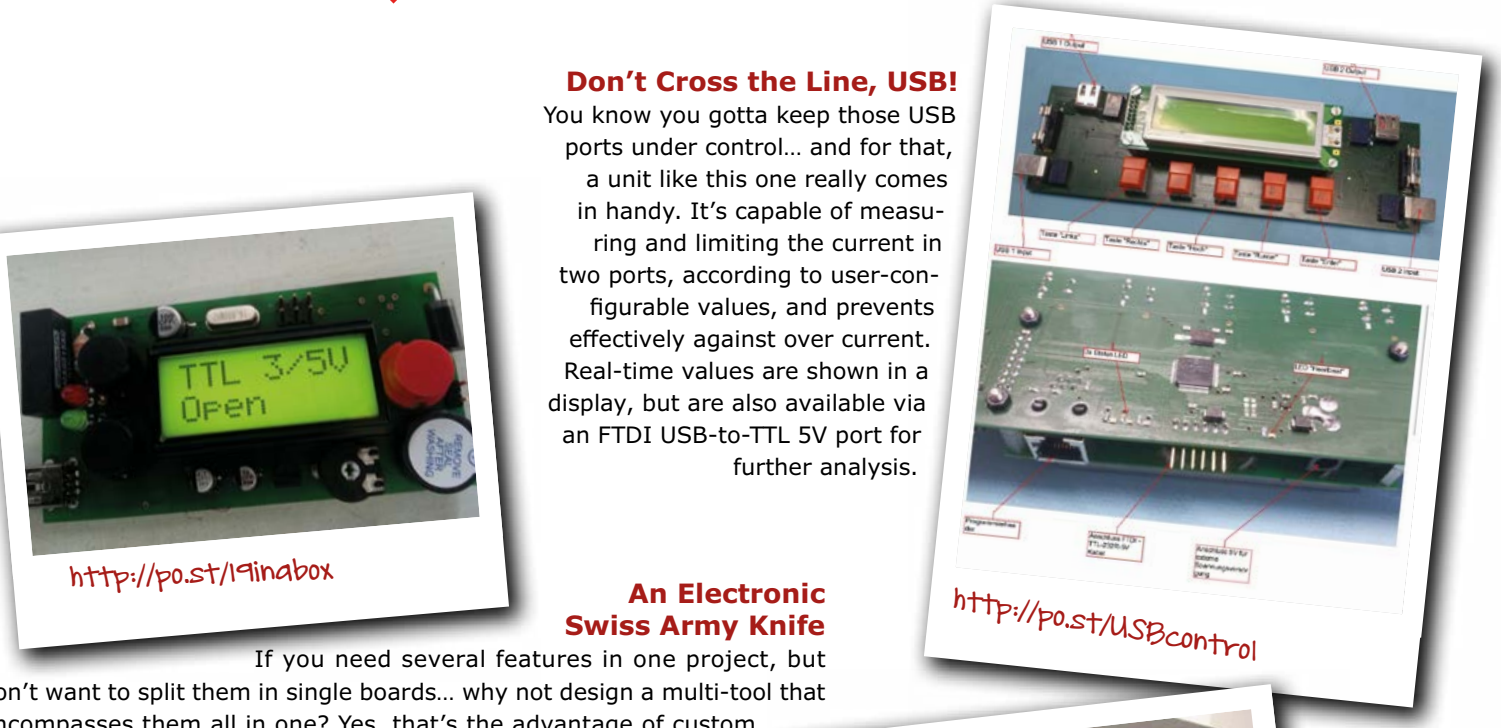
http://po.st/levelsensor

In My Level Sensor I Trust

Looking for a trustworthy level sensor to monitor liquid in a well or tank? This design consists on an emitter module and three receivers which get alerts generated by three probes (high, low, and critical level/alarm). Apart from this, it can also activate a pump to avoid a flooded basement or automatically control the level in a tank, for example. If you just need a multilevel alarm, the circuit is scalable, so the first part of the circuit can work alone.

Don't Cross the Line, USB!

You know you gotta keep those USB ports under control… and for that, a unit like this one really comes in handy. It's capable of measuring and limiting the current in two ports, according to user-configurable values, and prevents effectively against over current. Real-time values are shown in a display, but are also available via an FTDI USB-to-TTL 5V port for further analysis.



http://po.st/19inabox

http://po.st/USBcontrol

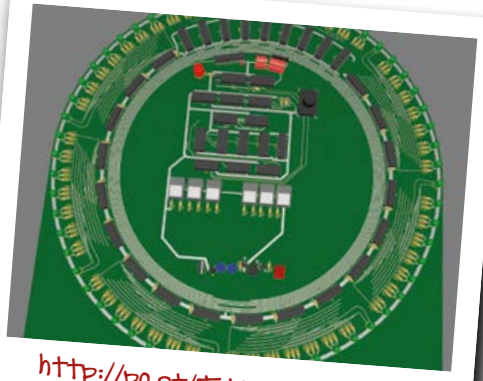
An Electronic Swiss Army Knife

If you need several features in one project, but don't want to split them in single boards… why not design a multi-tool that encompasses them all in one? Yes, that's the advantage of custom electronics! Remember the project 13-in-a-Box? Well, now 6 extra features got added, including a buzzer, voltage measurement, servo control, DCF77 decoding, Neopixel LEDs control, DS1820 temperature decoding and more. It just keeps growing and growing…

http://po.st/8Qradio

All You Need Is… 8 Transistors

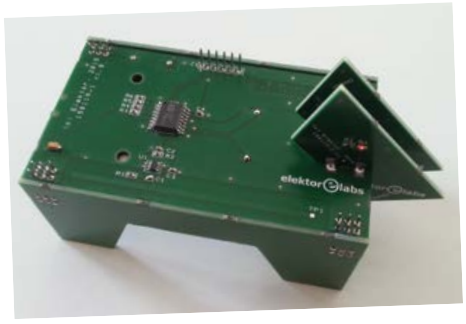
To build an AM radio, we mean, with lots of fun guaranteed during the process. This project details the assembly and testing of a standalone superheterodyne AM radio: it's an excellent way to get your hands on the world of RF. The first four transistors comprise the reception stages (incl. the local oscillator), and the other four are a simple 3-stage amplifier. The circuit can be powered at 6 V by means of four AA-size batteries, making the project suitable for portable designs.



http://po.st/TTLclock

It Was TTL Time

This is the story of a digital clock that wanted to be analog, but could only operate with TTL ICs. Instead of the expected hour and minute hands the clock boasts a circular set of LEDs. MCUs were obviously not an option (too easy, too mainstream!), but that didn't mean it was not user-configurable. Instead, a set of DIP switches was intended to change the LED patterns according to a coding system. A digital clock with an analog soul, and zero microcontrollers… but 100% cool!



http://po.st/hacktor

Bark at the Moon

Remember Hacktor, the "convertible" PCB that turns into a dog — he's now ready to bark! Hacktor features an Atmel QTouch proximity sensor which detects when you're near, playing a barking sound with an audio playback IC and a speaker located in its "belly," flashing its LED eyes as well. It's built to hack, so it can be programmed via the onboard SPI header, and incorporated easily in your own projects. Besides, it looks great on your desk, don't you think? ⏮

(150338)

Experimenter's Transistor Tester

Elektor 2/2015 (March & April), page 56 (130544)

ADDENDUM. Scarce mention is made in the article of the functionality of transistor T1. After building the unit as specified in the schematic, this component has no special significance to the user. T1 is used for self-calibration under control of port line PA3, and its operation is evident from the project source code written in BASCOM AVR.
On another note, there's also no reference to the frequency of the quartz oscillator of the Platino board, which has to be 8 MHz (this can also be checked in the project source code). [www.elektormagazine.com/130544]

Err-lectronics

Corrections, Updates and Addenda to published articles

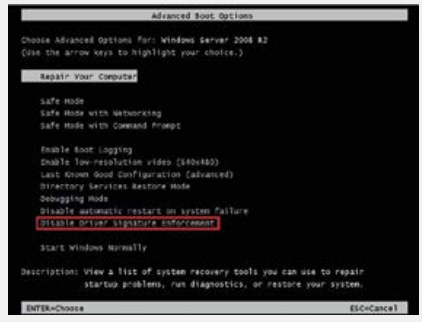
Compiled by **Jaime González-Arintero**

Typos, plain errors, postscripts … even erroneous historical facts this time! For sure that's not our particular walk of shame. Although our target is to release flawless designs, that's not always possible. Luckily, we have our keen, eagle-eyed readers (you!) to check everything. Not once, but twice, and even three times. Thanks everyone for the contributions!

UART/RS-232 Data Logger

Elektor 4/2015 (July & August), page 52 (140126)

UPDATE. Using Windows 8 and higher versions it may be impossible to install the USB driver for the Data Logger. The problem is Window's 'Driver Signature Enforcement' (DSE) that blocks the driver (which is not signed). To work around this problem you should disable the "Driver Signature Enforcement," which unfortunately is a rather complicated process. Luckily, there are many tutorials on the Internet that explain how to proceed. Please consider the patch linked to below. With DSE disabled, the driver should install without any problems.
[http://po.st/disableDSE]



J²B Synthesizer: an open-minded digital music platform

Elektor 1/2015 (January & February), page 10 (140182)

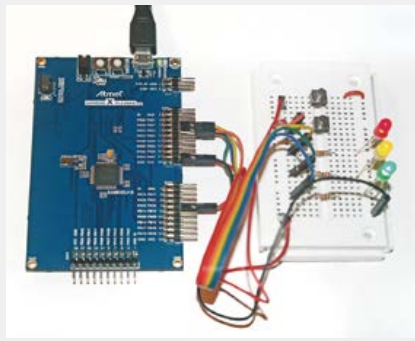
UPDATE. Elektor Labs Commander Clemens Valens has expanded the software of his popular J²B Synthesizer. Now there are three new firmware versions available in total, and one of them turns the synth into a (rudimentary) drum machine. These are 'ports' from Soulsby's Atmegadrum, Atcyclotron and Delayertron. Together with the original Atmegatron firmware there are now four unique synthesizers in one instrument. You can download the complete archive with all four firmware files via the original project at Elektor Labs, or via the link [http://po.st/J2Bv3].

USB Hub feat. RS-232/422/485

Elektor November 2014, page 10 (140033)

CORRECTION. In the published schematic, the pin numbers 2 and 3 of the LM2937 regulator (IC2) are exchanged. The correct pinout is: 1 (INPUT), 2 (GND), 3 (OUTPUT). The fixed schematic can be downloaded via the link below. The error does not affect the manufactured boards, or the rest of the files.
[http://po.st/140033schematic]

From 8 to 32 bits: ARM Microcontrollers for Beginners (3)



Elektor 3/2015 (April & May), page 13 (150041)

CORRECTION. A single letter can make a big difference! Have a look at the Listing 1: In the configuration functions for the WDT and the EIC, you may find two identical instructions (both in the second part of the code, with a few lines in between). Regrettably the port letter is wrong in both of these: 'B' is required instead of 'A'. Thus, both instructions should look as follows:

```
config_extint_chan.gpio_pin_mux = MUX_PBxxA_EIC_EXTINTx;
```

Please note that the x's correspond to the pin number and the channel number. Instead of typing it to implement the correction, you can download the corrected code from the web page with this article. This error was spotted by the author; thanks **Viacheslav Gromov**!
[www.elektormagazine.com/150041]

USB-to-Multi-Protocol Serial Converter

Elektor 3/2015 (April & May), page 92 (130542)

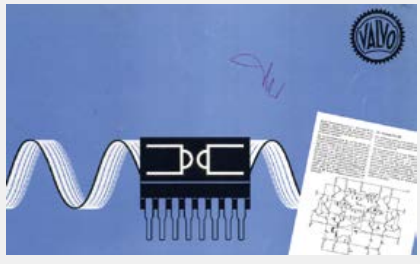
CORRECTION. The text box 'What's a serial port anyway?' mentions that "connecting the terminals to the computer in an economical way and allowing a certain distance finally resulted in the well-known RS-232 standard." However, the RS232-C standard was originally specified as an interface between DTE (Data Terminal Equipment, i.e. end instruments, such as terminals or mainframes, servers, etc.) and DCE (Data Communication Equipment, i.e. data transmission devices, such as modems). This allowed two DTEs to be connected directly by means of a crossover cable.
A big thank you to **Ernst Stippl** from Vienna for his (historical) rectification! :-)

Simple AVR port toggle (Tips & Tricks section)

Elektor 3/2015 (April & May), page 36 (150027)

ADDENDUM. This trick is relatively old, since the first AVR microcontrollers bringing this possibility were already released before 2005 approximately. Digging into the ATtiny13 datasheet shouldn't be too time consuming, since you can easily go to "I/O Ports," then to "Ports as General Digital I/O", and then check the section "Toggling the Pin" (sometimes also titled "Reading the Pin"). The Pin Registers may normally be found in the first 32 I/O registers (having an address under 0x20), and thus are still accessible at bit level. The expression PINB.5=1, used in the code, is converted by BASCOM into an SBI command. If the AVR MCU has a flash memory larger than 8 K, with a 2-word-wide IVT (Interrupt Vector Table) it's possible for example to toggle ports directly, by means of an interrupt in the IVT itself. If the pin is set as an input, the internal pullup can be enabled and disabled through the same procedure. Nevertheless, in some AVR series devices (approx. 2007 – 2010) the pullups could not be set via the data direction registers or the port registers, but via their own pullup enable registers. If an MCU pin is independent from the registers due to another function (such as a reset), but it's still accessible (working with the ATtiny4/5/9/10, for instance). In this way we can still perform bit-wise XORs, for example to calculate or check a parity bit.
Stefan Rosenthal, thank you so much for pointing it out!

TCA580 Integrated Gyrator



Elektor 3/2015 (April & May), page 35 (150024)

ADDENDUM. Our kind reader **Christoph Kessler** had the German application note for the TCA580 integrated gyrator, together with an awesome 5-page datasheet, so he sent us the scanned files. These were released back in 1974 so if you're feeling nostalgic you should definitely give them a look. Both files are available for downloading via the link below. And once again, thanks Christoph!
[http://po.st/TCA580DE]

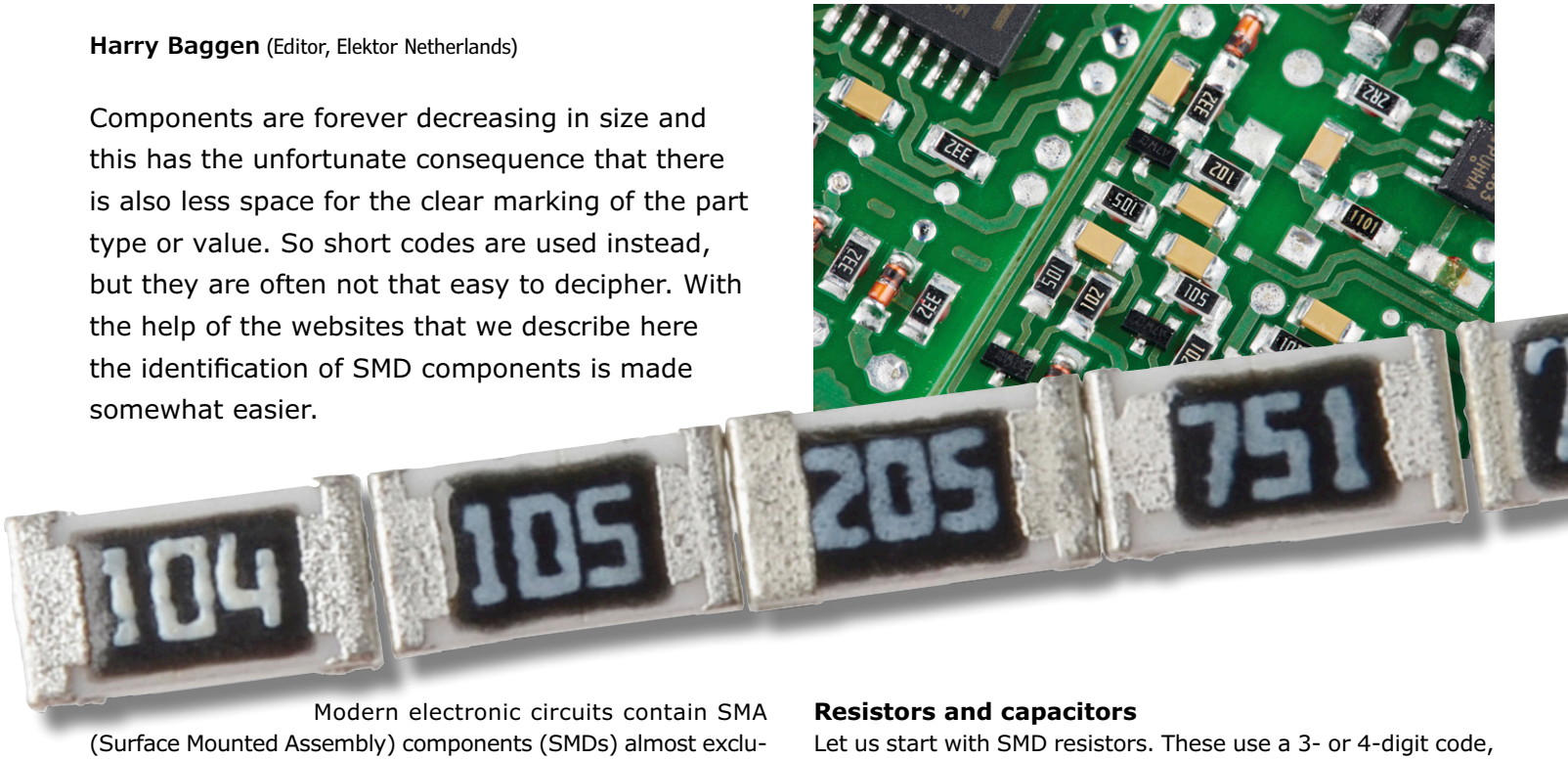
(150288)

SMD Codes Revealed

What's that under the magnifying glass?

Harry Baggen (Editor, Elektor Netherlands)

Components are forever decreasing in size and this has the unfortunate consequence that there is also less space for the clear marking of the part type or value. So short codes are used instead, but they are often not that easy to decipher. With the help of the websites that we describe here the identification of SMD components is made somewhat easier.



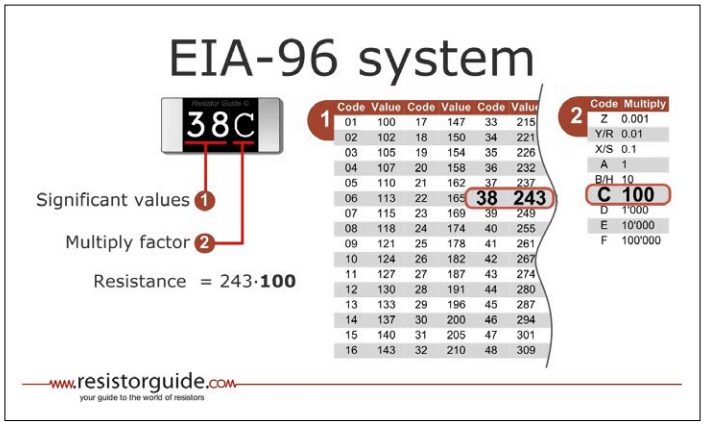
Modern electronic circuits contain SMA (Surface Mounted Assembly) components (SMDs) almost exclusively to reduce production costs and to decrease the size of the device. Although leaded components are still available, provided they are fairly standard parts, they are nevertheless becoming increasingly harder to obtain. Semiconductor manufacturers typically introduce (new) parts into the market only in SMD packages and then as a developer or hobbyist you have no choice: you have to work with SMD parts.

When SMDs started to play in increasingly important role in the 1990's in the electronics industry, many electronics enthusiasts feared that this would be the end of building small production runs or prototypes. But it wasn't all as bad as that. With some patience, some experimenting and the purchase of some new tools it turns out that working with SMD components by hand is quite possible. In the meantime the parts have become even smaller, but this is (still) not a limitation when building a prototype or repairing an electronic circuit. Because of the tiny dimensions of SMD components, many electronics enthusiasts do require stronger glasses, a magnifying lens or even a microscope when working with these smaller parts.

A problem with SMD parts is the device marking. Because there is so little space on the package, the manufacturers have resorted to all kinds of cryptic markings to indicate the type number and these are not straightforward to decode. When you order new components for a project, then the packaging will always clearly indicate the part number. But individual SMD components, and when repairing boards with SMD parts, it is difficult to establish what a particular component exactly is. To help with this we searched for a number of websites which will explain the abbreviations.

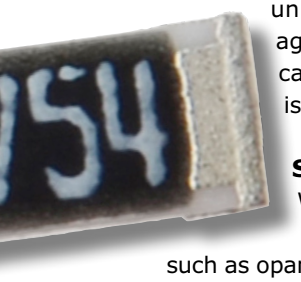
Resistors and capacitors

Let us start with SMD resistors. These use a 3- or 4-digit code, depending on the tolerance of the resistor. The design has a strong resemblance to that of leaded resistors: the difference is that numbers are used instead of colors. The first 2 or 3 digits indicate the value and the final figure is the multiplying factor. For 1%-resistors the EIA-96 code system (3 characters) is often used these days, where the characters no longer correspond to the value; a letter is used for the multiplying factor. On the website of *Your Hobby-Hour* [1] you will find a short explanation and a table for the EIA-96 values and importantly also an 'SMD resistor code calculator' which will translate the printed code into a sensible resistance value. On the website resistorguide.com [2] there is also a clear explanation about resistor codes, including a video which briefly explains the important details.



Because there is so little space on the package, manufacturers have resorted to all kinds of cryptic markings

With capacitors the situation is already much more complicated. The larger capacitors, such as electrolytics, are not a problem; there is usually sufficient room on the component to print the capacitance value and working voltage. With smaller Cs the same coding (3 numbers) as for resistors (see [3]) is sometimes used, but more often than not there is no marking at all (this is also true for small inductors)! The only solution then is to unsolder the component and measure its value with a capacitance meter. The maximum operating voltage remains an unknown. Based on the color of the package it is often possible to estimate in which capacitance range this particular capacitor is, without having to measure this [4].



Semiconductors

With smaller SMD parts that have 2, 3 or more pins (diodes, transistors, small ICs such as opamps, voltage regulators and reset-circuits) determining the correct part number is much more difficult. Most SMD parts are printed with a two or three character code, often complemented with a few smaller characters which indicate the production date and batch number.

Because only so few characters can fit on a component and there are so many different components, it is not possible to uniquely identify a particular component based on this short code. On various websites there are overviews and tables which contain a large number of character combinations and the potential type numbers and manufacturers that they indicate. Certain combinations have up to 20 possibilities. It then becomes necessary to carefully compare the package shape with the datasheets for all the components with these markings and possibly also study the schematic of the circuit that contained the component in order to discover what kind of component it is/was.

The SMD Codebook [5] is an 80-page long document (PDF) that after a few pages of general introduction is full of tables with codes. For each code it indicates the (potential) type number, the manufacturer, the package type and the technical details. The Codebook can be found on several websites,

Codes beginning with 'C'

Code	Device	Manufacturer	Base	Package	Leaded Equivalent/Data
C	BB565	Sie	I	SCD80	uhf varicap 2-20pF
C white	BAT165	Sie	I	-	40V 750mA sw Schottky
C	KV1832E	Tok	I	URD	uhf varicap 4-17pf
C0	HSMS-2820	HP	C	SOT23	HP2835 schottky
C0	HSMS-282B	HP	C	SOT323	HP2835 schottky
C1	HSMS-2821	HP	K	SOT23	HP2835 schottky
C1	BCW29	Phi	N	SOT23	BC178A
C1p	BCW29	Phi	N	SOT23	BC178A
C1t	BCW29	Phi	N	SOT23	BC178A
C1	BFQ51C	Phi	CX	SOT173	pnp BFR90A complement
C2	BCW30	Phi	N	SOT23	BC178B
C2p	BCW30	Phi	N	SOT23	BC178B
C2t	BCW30	Phi	N	SOT23	BC178B
C2	BFQ32C	Phi	CX	SOT173	pnp 4.5GHz 15V 100mA
C2	HSMS-2822	HP	D	SOT23	dual HP2835 schottky
C2	HSMS-282C	HP	D	SOT323	dual HP2835 schottky
C2	SST112	Tem	F	SOT23	J112 analog sw n-ch jfet
C2A	ZDC833A	Zet	B	SOT23	dual cc 28V varicap 15pF @2V
C3	HSMS-2823	HP	A	SOT23	dual HP2835 schottky
C3	BFQ23C	Phi	CX	SOT173	pnp complement BFP91A
C3	SMBT4126	Sie	N	-	2N4126
C3	SST113	Tem	F	SOT23	J113 analog sw jfet
C4	BCW29R	Phi	R	SOT23R	BC178A
C4	HSMS-2824	HP	B	SOT23	dual HP2835 schottky
C5	MMBA811C5	Mot	N	-	2N5086 pnp hfe 135-270

including also at [5]. If you prefer to use an HTML version then you are accommodated at *GM4PMK's SMD Codebook* [6] by Roger Blackwell. We haven't compared these completely, but the contents of both appear to be identical.

Another helpful source is the website of *S-Manuals* [7]. Under the heading 'Marking SMD' we find a table with all the 2-character combinations that can appear at the beginning of the SMD marking. Clicking on a combination brings us to a page with an overview of all the possible types that could be related to this combination. Here too, the package shape, type number and manufacturer are shown for each. And what is really handy: you can go straight to the corresponding datasheet!

With larger ICs, those with several tens of pins, we fortunately don't have to go through this much effort, there is usually sufficient space for a (clearly) readable type number.

Good luck with your search for the correct type number of that unknown SMD component!

(150339)

Correction: The web scouting installment in the May & June 2015 edition (Nosing Around Hackaday) has an incorrect name for the developer name of the project 'Using the Red Pitaya as an SDR'. The project developer is Pavel Demin.

Web Links

[1] www.hobby-hour.com/electronics/smdcalc.php
[2] www.resistorguide.com/resistor-smd-code/
[3] www.radio-electronics.com/info/data/capacitor/smd_capacitor.php
[4] https://en.wikipedia.org/wiki/Surface-mount_technology
[5] www.sos.sk/pdf/SMD_Catalog.pdf
[6] www.marsport.org.uk/smd/mainframe.htm
[7] www.s-manuals.com/smd

Preco 39 Years On

Better than ‘Hi-Fi’ all the while

(and how I broke its noise habit)

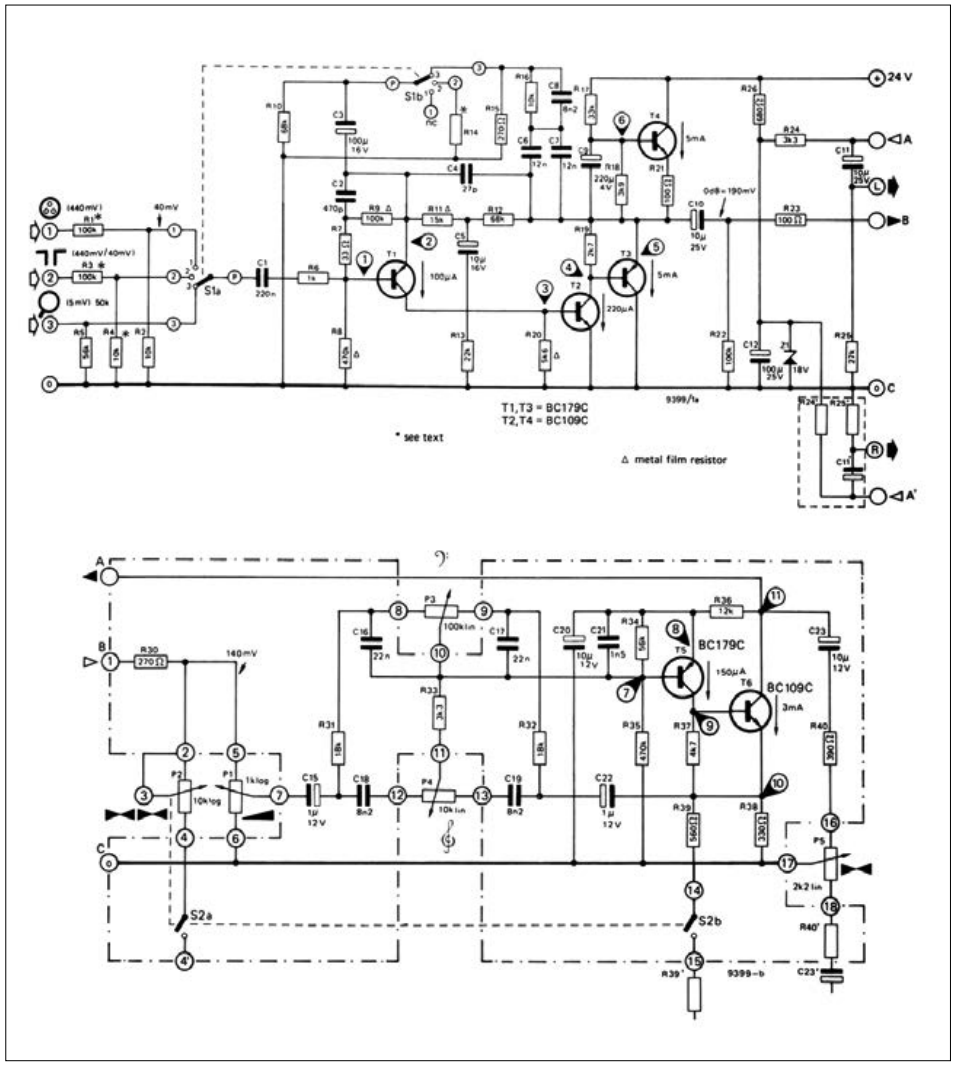


Figure 1. Reproduction of the original Preco schematics.

The German DIN 45500 industrial standard, which as far as I know is no longer in force, specified minimum requirements for ‘Hi-Fi’ audio equipment. It did not exactly reflect the state of the art even at the time, and specialist magazines dissatisfied with its laxity set their sights a little higher. In the case of Elektor, the result was the Equa standard.

One unusual design choice in the 1976 Preco project was that it was divided into two parts: the input stage and a variable amplifier, see **Figure 1**. The latter offered, in addition to the conventional controls such as volume, tone and balance, an adjustable ‘stereo width’ control. This could be set from approximately zero (i.e. mono) to twice the effective distance between the loudspeakers. It was far from just a toy feature: depending on the loudspeaker arrangement and the audio material it could deliver a definite improvement in the stereo image. The technique of deliberately adding out-of-phase crosstalk is still found today, although usually lacking Preco’s finesse: for example, a modern television might have a ‘stereo width’ or similar control buried in one of its sub-menus, but often the only possible settings are ‘on’ and ‘off’, and the effect, to my ears at least, is generally overdone and so rather annoying.

In 1976, stereo width control was de deluxe feature though and Elektor

By **Jürgen Friker** (Germany)

The Preco preamplifier by T Meyrick dates back to the April and May 1976 issues of Elektor. Naturally up to Elektor's in-house standards it was considerably more 'Hi-Fi' than the minimum required by the German standard. That does not mean Preco can't be improved though, or get a 21st century application.

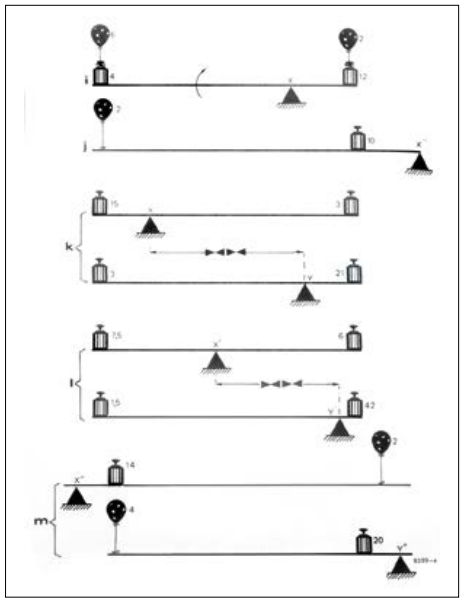


Figure 2. What better way to explain stereo bandwidth control than with a virtual balance, weights and the odd balloon.

Figure 3. Preco remote control.

devoted a small lecture to the principles using a unique drawing reproduced here for your amusement (**Figure 2**).
For the combination of a preamplifier and a control unit the Elektorians in good 1970's fashion coined the name *Preco*.

Flexible design

Usually a standard preamplifier will have an input stage for a moving-magnet cartridge, including an equalizer implementing the standard RIAA curve. The preamplifier/equalizer circuit can be implemented using, for example, one low-noise operational amplifier per channel. In high-end applications this will not be an off-the-shelf chip, but rather will be constructed from discrete components. The input stage for the Preco follows the conventional pattern, but with one difference: the path for all input signals is the same, and in particular this means that all signals go through the preamplifier stage. To cope with the wide range of possible input signal levels without clipping the preamplifier's input sensitivity is adjustable, and it should even be possible to use it as a microphone amplifier, although I have not tried this out as my tape recorder has one built in.
In summary we can say that the Preco was (and still is) a universal preamplifier with a wide range of applications, especially for those who want to use sources with low output signal levels such as record players with magnetic pick-ups.

Remote Control

Another special feature of the Preco is its remote control, the entire variable amplifier being located in an external box (**Figure 3**). This not only means that I can adjust the sound (including balance and stereo width) from the comfort of my armchair, but more importantly means that I can mount the whole system, including preamplifier, radio and output amplifier (the latter incidentally being an 'Elektornado') in a radio cabinet from the period before the appearance of pushbuttons and a nationwide AC power grid, without having to deface it by drilling extra holes. If the remote control option is not used, the two boards of the Preco can be fitted in a single enclosure and then it will look from the outside like any other preamplifier.
Figures 4, 5, and 6 show Preco fully integrated into my Lumophon tube radio. No apologies for the dust…this is Retronics.

The idea of having a separate input stage with adjustable amplifier does however require the audio signal to make the lengthy trip to the remote control box

Figure 4. Preco preamp mounted inside my Lumophon tube radio.

Figure 5. Side view of Preco preamp section.

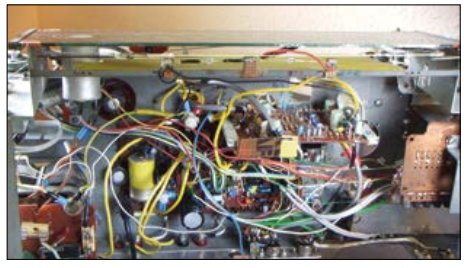


Figure 6. Under-chassis view showing Preco integrated into the tube radio's electronics and mechanics.

and back, which is rather odd. One might wonder whether a high-quality preamplifier can tolerate such long signal paths, in particular in view of the capacitance of the cable and (despite its screening) its ability to act as an antenna to pick up unwanted signals.

The capacitance of the cable, which of course depends on its construction and its length, forms an RC lowpass filter with the output impedance of the amplifier. In the case of the Preco this impedance (at least according to the 1976 article) is so low that even a long cable will have only a negligible effect on the frequency response of the system, and any external interference is very unlikely to be audible; and this is borne out in practice. Furthermore, the low output impedance means that a medium-impedance pair of headphones can also be successfully driven directly from the remote control box: via DC blocking capacitors, because, as we shall see, the output of the variable amplifier also carries power.

Caution: DC!

The connection between the input stage and the variable amplifier requires just five wires (four plus screens), and suitable cables, designed for stereo interconnections allowing both record and playback, and often fitted with five-pin DIN plugs, are readily available (they go by the odd name *Diodenkabel* in Germany). However, a consequence of this arrangement is that the power supply to the variable amplifier (at 18 V) has to be carried on the signal wires: more on this below.

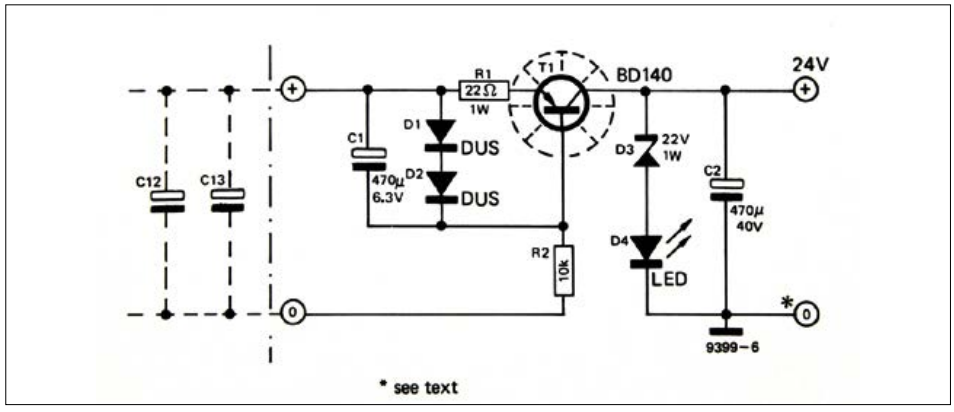


Figure 7. Reproduction of the original Preco power supply schematic. Hey *DUS*, good to see you back!

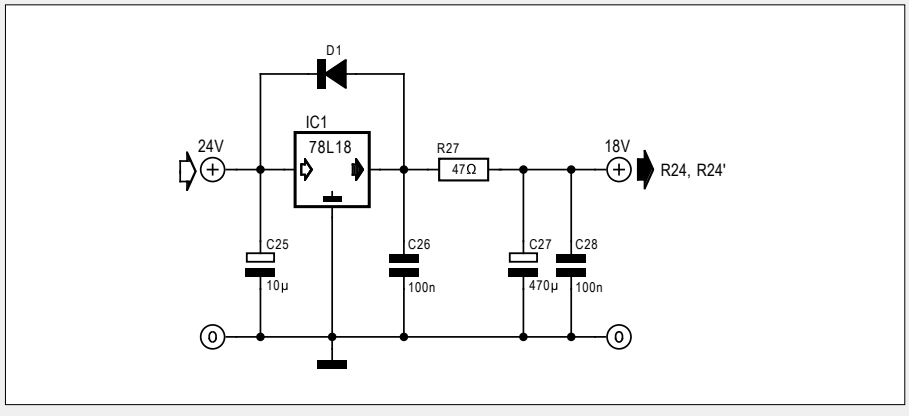


Figure 8. A suggested 24-to-18-V stepdown regulator.

C25 = 10µF (tantalum used to be used for its low inductance, but now for ethical reasons aluminum foil capacitors are preferred. For sure a modern 0.33-µF foil capacitor should do the job too)
C26 = 100nF
C27 = 470µF, 25V
C28 = 100nF (not critical; a larger value was used in the prototype, but it is even possible to do without this device entirely)
R27 = 47Ω
IC1 = µA78L18
D1 = 1N4002 (or any similar device available to hand)

What's that noise?

The original article recommended that certain resistors be metal film types rather than carbon film types because of their supposedly better noise performance; to me this seemed to indicate that it had already occurred to somebody that the Preco, although generally performing well, was noisier than it should be for a high-grade preamplifier (or at least for a preamplifier aiming at better performance than the DIN standard). Moreover, if the Preco really is such a high-end unit that we need to worry about the thermal noise from the material from which the individual resistors are made, then it is hardly likely to be realistic as a DIY project; and in any case our (non-digital) audio sources will of course not be free of noise. Notwithstanding all that, I fitted the recommended metal film resistors and also low-noise transistors from the BC414/BC416 series, and so it was all the more annoying to find that there was still a noticeable amount of noise on the output in the absence of any input signal. So, having eliminated the resistors and the transistors as likely sources of noise, I decided to try to find out what was going on.

Listening more carefully, and making use of the ability to adjust the stereo image width, I got the impression that the noise was coming from a fixed position in the stereo field. That was quite a surprise, as the stereo noise at the output of an FM receiver, for example, which is especially prominent when reception is poor, does not appear in a fixed position: whether one uses loudspeakers or headphones, it is not possible to say which direction the noise comes from. However, in this case the noise in the two channels is not independent, but synchronous; what matters is that the noise is out of phase between the two channels, so when the receiver is switched to mono operation by adjusting the stereo width control and the two channels are added together, the noise practically disappears. Contrariwise, this means that in a unit where the stereo noise appears to have a definite position, it is likely that one channel has been connected back-to-front, and as a result the

stereo experience will be rather unsatisfactory until the problem is fixed.

The internal noise of the Preco, however, is unaffected by the stereo width control; in fact any adjustments made on the remote control have remarkably little effect. On even closer listening it became apparent that not only was the noise fixed in its stereo position, it was exactly in the middle of the stereo image: the noise was in mono!

This was even more of a surprise. 'Mono' means that the same noise signal is appearing on both channels in the same phase. For an agreement like this between the two channels to occur in noise (which is made up of random and hence unpredictable voltage fluctuations) by chance is to say the least highly improbable; 'impossible' is not a word favored among probability theorists in a situation like this!

In other words, for two independent random signals to agree like this from time to time is inevitable; but for such agreement to happen two times on a row (depending on exactly what we mean by 'in a row') is much less likely. And if it continues to happen for more than a couple of microseconds, it is time to check your lottery tickets too.

So here we are dealing with monophonic noise, and either this comes about by dint of some mysterious power happening to affect both channels in synchrony, which seems highly implausible, or there is a single source for the noise affecting both channels. Now, turning to the components that make up the Preco, there really are not so many possible candidates: perhaps resistor R26, capacitor C12 or Zener diode Z1, which together comprise the power supply (**Figure 7**), the output of which feeds the variable gain circuit and the remote control circuit for both channels. The finger of suspicion started to point in the direction of the Zener diode.

It was also becoming clear why the noise was relatively unaffected by adjustments to the settings on the adjustable amplifier: even without this in circuit (and hence with no signal) the noise voltage was being carried directly to the output of the Preco via resistor R24 and capacitor C11 (and via R24' and C11'), and hence to the input of the power amplifier.

Now of course, at the same time as I was selecting special low-noise transistors for the circuit, I could also have purchased a low-noise Zener diode (and such devices do exist), but still the basic problem of breakthrough between the supply and the output would have remained and noise would still have been present.

Tips for anyone who might be inspired to experiment with the Preco design.

- If you are using the remote control, it is best not to work on the connection between the two parts of the Preco while it is in operation. This will not harm the Preco, but because of the DC levels on the connections you may carry out an inadvertent load test on your bass loudspeaker and it will make quite a lot of noise! This will happen even if the volume control is set to zero.
- For the same reason (in conjunction with Murphy's Law), it is a good idea to select connectors for the remote control which are of a type not used anywhere else on the equipment.
- For optimal matching between the magnetic pickup input and the pickup itself, R5 and R5' on the printed circuit board can be replaced by 100 kΩ types and an extra phono socket can be added in parallel with each input (regardless of what connector type it uses). A dummy plug containing a resistor matched to an individual pickup can then be constructed and plugged into each new socket. If desired, a small parallel capacitor can also be included in the dummy plug for even better (and easy to adjust) matching.

Having found the culprit I looked around for possible remedies. There was already a smoothing capacitor present in the circuit in the form of C12, which should help to reduce noise although it clearly was not doing a good enough job. I could increase its value significantly, but this is not a good idea: the heftier and higher-quality the capacitor placed in parallel with a noise source, the greater the smoothing currents flowing in and out of it arising from the random voltage fluctuations, which in turn can lead to a reduced life for the Zener diode.

What alternative ways are there to obtain 18 V from 24 V? **Figure 8** has the answer: with a 78L18. This is a great improvement over the original circuit, although it does draw about 2 mA more current. (Fortunately the day is yet to come when preamplifiers, like washing machines, are subject to energy efficiency labeling requirements, but even then perhaps the Preco might be granted an exemption as a vintage design!) Moreover, along with the necessary capacitors to suppress ripple, the TO-92 package fits on the existing printed circuit board.

In my search for perfection, I decided not to listen to the result of this change immediately, but rather to make a further change to improve the performance of the power supply still further: adding a lowpass filter in the form of R27, C27 and C28. This reduces the output noise of the voltage regulator even further (it can't of course be removed altogether) in comparison to the Zener diode circuit. D1, a protection diode, is also added, as there can easily be a large spike in the secondary voltage when the primary voltage is applied: in some cases this can defeat the otherwise very effective self-preservation features of the voltage regulator. It is easy to see that, with a resistance of 47 Ω, it is easy for the current through R27 considerably to exceed the 100 mA output capability of the 78L18 and hence push it into current limiting. This does in fact not pose a risk to the device, which can easily survive such stress repeatedly; and in any case it is not a problem in my design as, in keeping with the vintage enclosure I am using, the entire preamplifier is powered from an AC supply whose output voltage rises only slowly in order to simulate the warm-up behavior of tube circuits. But that is a story for another time. ⏮

(150343)

ESTD 2004

www.elektor.tv

Retronics is a monthly section covering vintage electronics including legendary Elektor designs.
Contributions, suggestions and requests are welcome; please telegraph editor@elektor.com

Compiled by **Beatriz Sousa**

Stronger logistics

We never thought the Elektor Store would be such a success, but it is! And if the writing wasn't on the wall with the Crazy X-mas sales, the Cool Summer Deals hammer it home. The success prompted us to beef up our warehouse and logistics department, so starting this month our delivery times will improve and the department will operate smoother. We apologize if you fell victim to delays in your order fulfillment with any of our highly popular promotions.

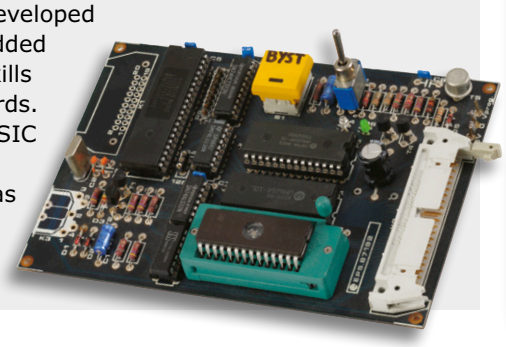
Robotcamp

Recently Elektor Expert Bart Huyskens rolled out a robot workshop for kids in his home country, Belgium. With some triggering boards from Elektor LABs and Bart's inimitably enthusiastic method of teaching electronics, the kids we're sure are hooked for life as (junior) Elektor members ;-)

READ ONLY MEMORY

Elektor magazine and its parent publishing company boast a long and rich history. In this space we picture a gem from the past.

Intel's **8052AH-BASIC microcontroller** was the Arduino during the eighties of the previous century. Elektor products like the 8052AH-BASIC Single-Board Computer (1987) and the 8032/8052 Single-Board Computer (1991) were immensely popular projects, and many readers developed their first embedded programming skills using these boards. The 8052AH-BASIC board and its famous BYST was Retronicized in March 2007.



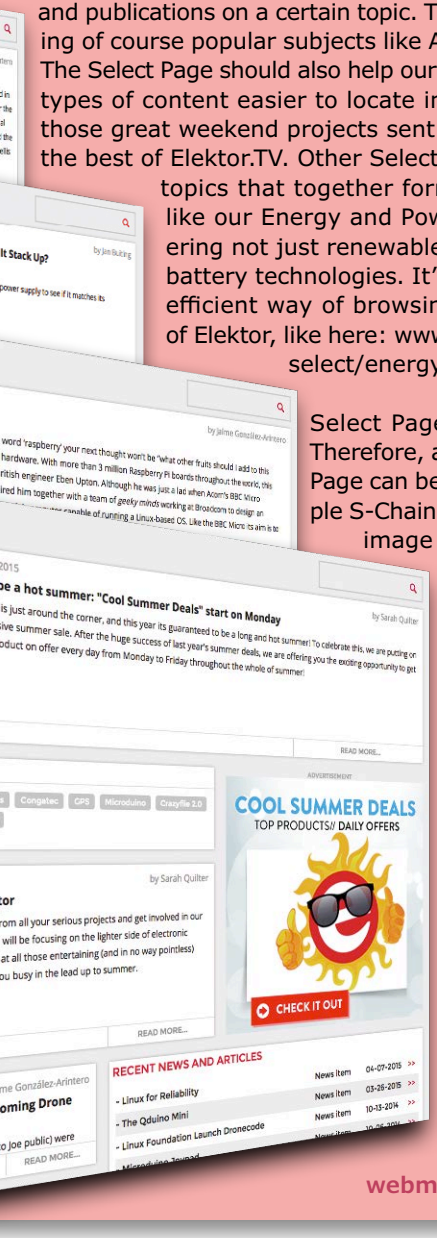
PEOPLE NEWS • CJ Abate from sister publication **CircuitCellar** lends a hand in launching Elektor's the US and Canada • Muhammed Söküt, Member Manager on the German team, will offer Elektor like DRONE communities in the German language • Margriet Debeij, Client Manager Germany, is publish 'special reports' in the next edition of **Elektor Business** .....

STORE FORUM LABS EN

Select Pages

that the platform offers. Apart from being a new home to all daily news items, magazines and magazine articles, there is a lot more coming your way on this brand new platform. Among the most recent additions are the Select Pages.

Select Pages are web pages where we compile information and publications on a certain topic. These topics vary, including of course popular subjects like Arduino or Raspberry Pi. The Select Page should also help our readers to make certain types of content easier to locate in a single place, like all those great weekend projects sent through "dot-POST" or the best of Elektor.TV. Other Select Pages combine certain topics that together form an area of expertise like our Energy and Power special edition covering not just renewable energy topics but also battery technologies. It's a completely new and efficient way of browsing and surfing the best of Elektor, like here: www.elektormagazine.com/select/energy.



Select Pages are built from Tags. Therefore, a Tag that holds a Select Page can be recognized from a simple S-Chain icon like is shown in the image (1).

Also, launching right from each article or news item that's part of the Special on the magazine website, there is a small banner that will lead you to the Select Page. It is easy and comfortable to use. We are open for any suggestion from our readers and would like to get input on themes, subjects or types of content you would like us to Select for you!

Let us know your suggestions through webmaster@eimworld.com!

Cool Summer Deal Campaign with extra intensity in memberships and products in peripheral networks inviting companies inside and outside Germany to

EXPERT PROFILE

Elektor works closely together with more than 1,000 experts and authors for the publication of books, articles, DVDs, webinars and live events. In each installment of Elektor Word News we put one of them in the limelight.

Name: **Jason Long**

Age: **37**

Education: **Electrical Engineer (B.ScEE)**

Publications: **Large library of online tutorials**

Training: **PIC, ARM, ANT+, BLE, LabVIEW, low power embedded design and intrinsic safety**

Who is Jason Long?

I'm a Canadian who absolutely loves embedded systems. I've celebrated that passion by teaching others as a volunteer for the last 15 years and started Engenuics Technologies in 2010. I have two young boys already fascinated by blinking LEDs.

What will be the most key electronics development?

I think Personal Area Networks through NFC and very low power radio protocols like ANT+ and BLE are going to drive the explosion of the Internet of Things. Embedded developers fluent with hardware and firmware design will be in big demand.

What was THE development of electronics?

Over the last few years the ARM Cortex family has been hugely significant, especially with devices like the Nordic nRF51 that puts everything you need together with low power wireless (I don't work for Nordic, I just love that part!).

What makes Canada different from the US in terms of electronic innovations?

Maybe innovation in Canada is driven by people's passion, while in the US it's capitalism? I'll buy beer if I just offended anyone. But really, innovation anywhere comes from a desire to connect to people through great products & service.

Suppose you get $500 to buy stuff in the Elektor Store: what's it going to be? Why?

Oh man — that's the greatest gift an engineer could get. One of the top things on my to-do list involves quadcopters, so your Crazyflie 2.0 looks pretty sweet.

Who is your most admired developer of the world, living or dead? Why?

I'm a bit weird because I've never identified with particular individuals. The people I admire are the hobbyists and engineers with the passion and commitment to figure stuff out and make things.

What's the project you are most proud of? Why?

I've evolved our "Razor" / "Blade" development platform over many years and I think it's really solid. I think it fills a gap for engineers as they move from the hobby world or academics into product design in industry.

(150344)

Hexadoku The Original Elektorized Sudoku

With summer vacation or camp well behind us, we can return to business as usual. As far as your spare time is concerned, the technology in this edition and a fresh Hexadoku puzzle should keep you busy for a good eight weeks. 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 book 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 Book Vouchers worth **$70.00 / £40.00 / €50.00 each**, which should encourage all Elektor readers to participate.

Participate!

Ultimately October 30, 2015, supply your name, street address and the solution (the numbers in the gray boxes) by email to: **hexadoku@elektor.com**

Prize winners

The solution of Hexadoku installment 4/2015 (July & August) is: **981B4**.

The €50 / £40 / $70 book vouchers have been awarded to: Gerard Lodder (Netherlands); Karl-Josef Wernet (Germany); Olivier Chaufouraux (Belgium); Pascal Jordil (Switzerland) and Dirk Petig (Germany).

Congratulations everyone!

	0	3				8	4	E	1				7	2	
F	5			C	9	6			D	A	7			8	0
					7		0	4		C					
	6	7				1			8				4	A	
		E	2	B		A	1	6	9		5	4	C		
	C		4	5	F	0			B	3	1	7		D	
0		5	D	6	8	2			7	4	F	B	1		E
							C	8							
							7	9							
1		D	7	E	6	4			3	5	C	9	2		A
	4		5	8	0	C			F	1	A	E		6	
		6	3	1		5	A	D	0		8	F	B		
	D	A				3			5				6	7	
					A		5	1		7					
5	2			7	B	F			E	8	4			3	9
	1	0				E	6	3	A				F	B	

3	8	C	A	4	0	6	E	2	D	9	F	7	5	B	1
D	E	9	7	5	F	A	C	0	1	6	B	8	2	4	3
6	F	1	5	B	7	D	2	3	4	A	8	C	0	9	E
B	2	0	4	8	1	3	9	C	E	5	7	F	6	A	D
E	1	3	B	F	2	7	A	4	5	D	9	0	C	6	8
C	D	2	6	9	8	1	4	A	F	0	3	B	E	5	7
4	5	7	9	0	C	E	D	B	2	8	6	1	3	F	A
0	A	F	8	6	3	B	5	E	7	1	C	9	4	D	2
2	0	8	F	A	9	4	6	5	3	7	E	D	1	C	B
5	4	D	E	C	B	8	7	6	9	2	1	A	F	3	0
1	6	A	C	D	5	2	3	F	8	B	0	E	9	7	4
7	9	B	3	1	E	F	0	D	A	C	4	5	8	2	6
8	B	4	1	E	A	C	F	7	6	3	5	2	D	0	9
F	C	6	0	2	D	9	8	1	B	4	A	3	7	E	5
9	7	E	D	3	6	5	B	8	0	F	2	4	A	1	C
A	3	5	2	7	4	0	1	9	C	E	D	6	B	8	F

The competition is not open to employees of Elektor International Media, its subsidiaries, licensees and/or associated publishing houses.

Professional Quality @ Discount Prices!

reichelt elektronik .co.uk

SUBSCRIBE NOW!

Newsletter

Receive weekly fresh information about

- product innovations
- specials
- Price reductions

PROXIMITY SWITCH · POSITION SENSOR · HUMIDITY · FLOW · PRESSURE · HALL EFFECT · ULTRASOUND · FORCE

- More than 45 years of experience
- 24-hour shipping
- More than 50,000 products

Hall effect sensors

Radiometric sensor, linear — Honeywell

- Operating voltage: 4.5 … 10.5 VDC
- Power consumption: 8.7 mA (@ 5 VDC)
- Operating temperature: –40 … +150 °C
- Linearity: 1% typ.
- Response time: 3 µs

Housing: TO-92,
Output: 0.2 V/1.5 mA

Order number			Magnetic range	Sensitivity
SS 496 A1	2,75 €	(~ 1,93 GBP)	±84 mT	2,5 mV/G
SS 495 A	1,40 €	(~ 0,98 GBP)	±67 mT	3,125 mV/G
SS 495 A1	2,65 €	(~ 1,86 GBP)	±84 mT	3,125 mV/G

Radiometric sensor, linear, SMD — Honeywell

- Operating voltage: 2.7 … 6.5 VDC
- Power consumption: 10 mA
- Operating temperature: –40 … +100 °C
- Linearity: 1% typ.
- Response time: 3 µs

Housing: SOT-89
Output: 1.0 V/1.5 mA

Order number			Magnetic range	Sensitivity
SS 59 ET	1,40 €	(~ 0,98 GBP)	±65 mT	1,0 mV/G

Hall effect sensor, digital, SMD — Honeywell

Temperature-compensated digital hall effect sensors

unipolar, housing: SOT-89
Output: 0.4 V/20 mA

Order number			Magnetic range
SS 543 AT	1,40 €	(~ 0,98 GBP)	7,5 … 18,0 mT
SS 549 AT	2,65 €	(~ 1,86 GBP)	23,5 … 39,0 mT

Hall sensors — SIEMENS

Housing: P-SSO-3-2

Order number			Magnetic range	
TLE 4905L	0,60 €	(~ 0,42 GBP)	±17 mT	unipolar
TLE 4935L	0,63 €	(~ 0,44 GBP)	±20 mT	bipolar/latch

Inductive proximity switch — CONTRINEX

Excellent inductive sensors for contactless detection of metal parts for daily use in automation solutions. Ideal for use in demanding ambient conditions.

Excerpt from our product range.
For the entire product spectrum please visit: http://rch.lt/kJ

NO contact/PNP

Order number			Ø x L	Switching distance	Installation
DWAD 623 03	66,80 €	(~ 46,79 GBP)	Ø 3,0 x 22 mm	1,0 mm	2 m cable flush
DWAD 623 04	51,40 €	(~ 36,00 GBP)	Ø 4,0 x 25 mm	1,5 mm	2 m cable flush
DWAD 623 M5	51,40 €	(~ 36,00 GBP)	M5 x 25 mm	1,5 mm	2 m cable flush
DWAD 503 M12	47,90 €	(~ 33,55 GBP)	M12 x 50 mm	6,0 mm	2 m cable semi-flush
DWAS 623 M8 001	32,90 €	(~ 23,04 GBP)	M8 x 45 mm	2,0 mm	M8 plug, 3-pin semi-flush
DWAS 613 M8 001	30,80 €	(~ 21,57 GBP)	M8 x 45 mm	2,5 mm	M8 plug, 3-pin not flush
DWAS 513 M12	47,80 €	(~ 33,48 GBP)	M12 x 60 mm	10,0 mm	M12 plug, 4-pin not flush

Light barrier Ø x L 18 x 50 x 50 mm — CONTRINEX

Through-beam light barrier, 2 m cable

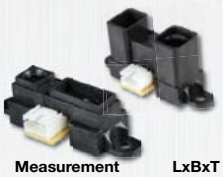
Order number			Switching distance	Switching method
LLK 5050 000	23,30 €	(~ 16,32 GBP)	15000 mm	Transmitter
LLK 5050 003	35,99 €	(~ 25,21 GBP)	15000 mm	antivalent, PNP

Reflective light barrier, 2 m cable

LLK 5050 103	42,40 €	(~ 29,70 GBP)	800 mm	antivalent, PNP

Distance Sensors — SHARP

Very high reliability and higher precision compared to conventional Sensors



GP2Y0A

Order number			Measurement range (cm)	LxBxT (mm)
GP2-0215	6,70 €	(~ 4,69 GBP)	20 ... 150	29,5 x 13,0 x 21,6
GP2-0430	5,10 €	(~ 3,57 GBP)	4 ... 30	37,0 x 18,9 x 13,5
GP2-1080	5,65 €	(~ 3,96 GBP)	10 ... 80	29,5 x 13,0 x 21,6

Order number			
DMC01-SC150	2,15 €	(~ 1,51 GBP)	3-pin, Connection cable

Humidity sensor 0 ... 100 % rF, TO 39 — B+B SENSORS

Digital humidity sensor with IC interface in pressure-resistant TO39 housing (up to 16 bar), suitable for dew point measurement.

Order number		
HYT 939	29,90 €	TO 39

Pressure sensors, UsV 4.75–5.25 — freescale semiconductor

Order number			PR kPa	Sens mV/kPA	Lin %
MPX 5010DP	11,60 €	(~ 8,12 GBP)	0 - 10	450	±5
MPX 5050DP	14,90 €	(~ 10,44 GBP)	0 - 50	90	±2,5
MPX 5100DP	14,90 €	(~ 10,44 GBP)	0 - 100	45	±2,5
MPX 5500DP	9,35 €	(~ 6,55 GBP)	0 - 500	9	±2,5

Ultrasonic sensors

Ultrasonic ceramic transmitter and receiver for 40 kHz

Ø 9,9 mm, H 7,1 mm

Order number			
MUS-40E	3,05 €	(~ 2,14 GBP)	Receiver
MUS-40S	3,05 €	(~ 2,14 GBP)	Transmitter

Daily prices ! Price as of : 27.07.2015
Prices in € incl. statutory VAT, plus shipping costs
reichelt elektronik, Elektronikring 1, 26452 Sande (Germany)

Payment Methods: SOFORT BANKING · VISA · amazon payments · PayPal

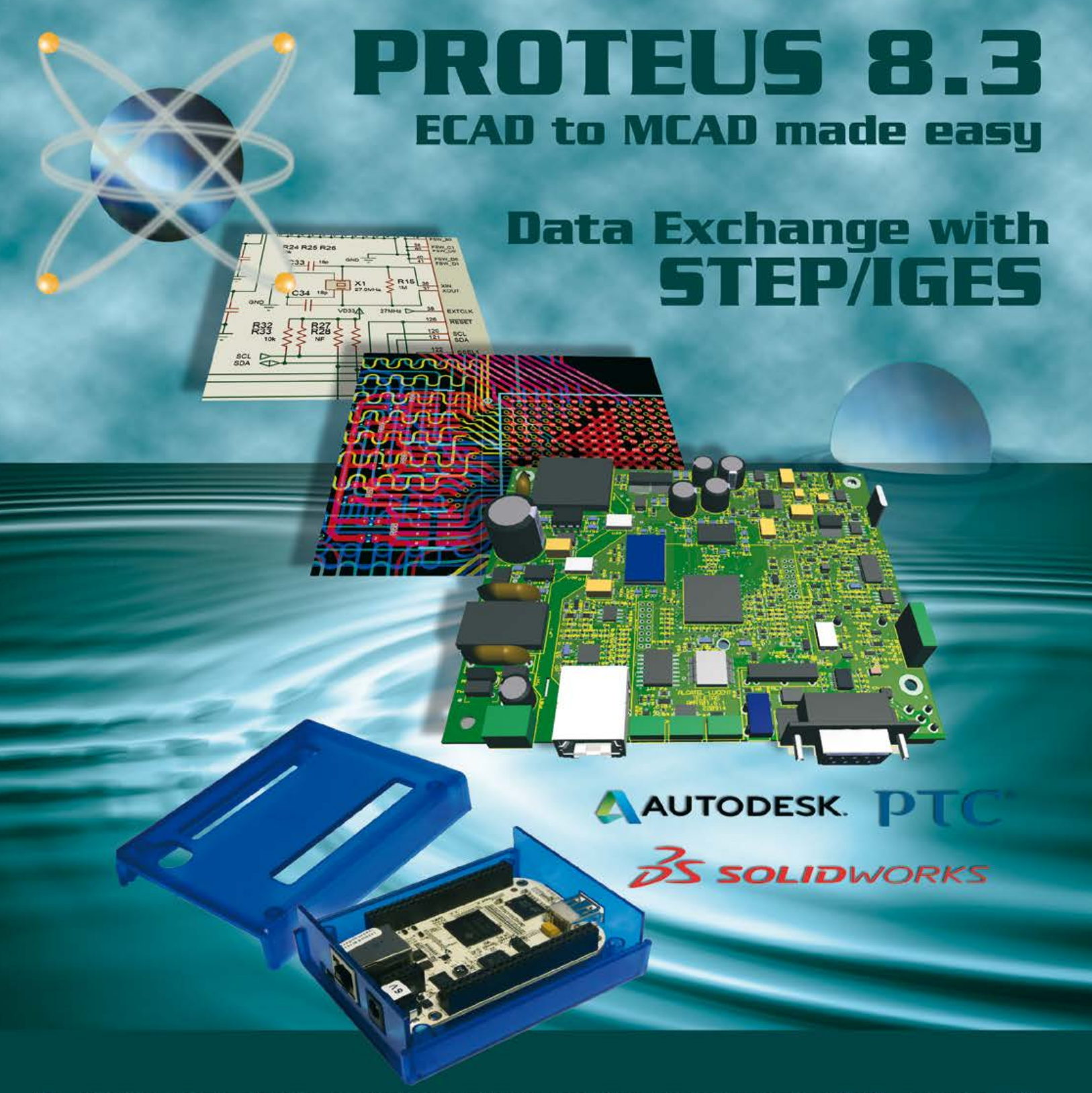
Order now! **www.reichelt.co.uk**

Order Hotline: **+49 (0)4422 955-360**

PROTEUS 8.3

ECAD to MCAD made easy

Data Exchange with STEP/IGES



The Proteus Design Suite now includes full support for data exchange with Mechanical CAD packages via the STEP/IGES file formats. This allows you to better visualise your design and helps quickly solve fixtures, fittings and casement problems.

Import 3D STEP/IGES models for your parts and visualise inside the Proteus Design Suite. Export your completed board to Solidworks or other MCAD software.

Visit www.labcenter.com
Tel:+44 01756753440 E-Mail info@labcenter.com

Labcenter Electronics Ltd, 21 Hardy Grange, BD23 5AJ

Labcenter Electronics www.labcenter.com