

elektor

SUPRA PLATFORM PROGRAMMING

Pull your code from the Embedded Firmware Library

I/O LEDs LCD
ADC PWM SD MUX TCP/IP

- From BASIC to Python | **Embedded Firmware Library** | 500 ppm
- LCR Meter Part 3 | **FPGA Programming** | Linux Board Extension
- OTA Triangular Wave Gen** | BaroStick ● **PCB Soldering Pitfalls**
- Ceramic C's Off the Mark ● **1963 Tube PSU** ● The Invisible User Interface

US \$ 9.00 - Canada \$ 10.00



7 25274 24965 7

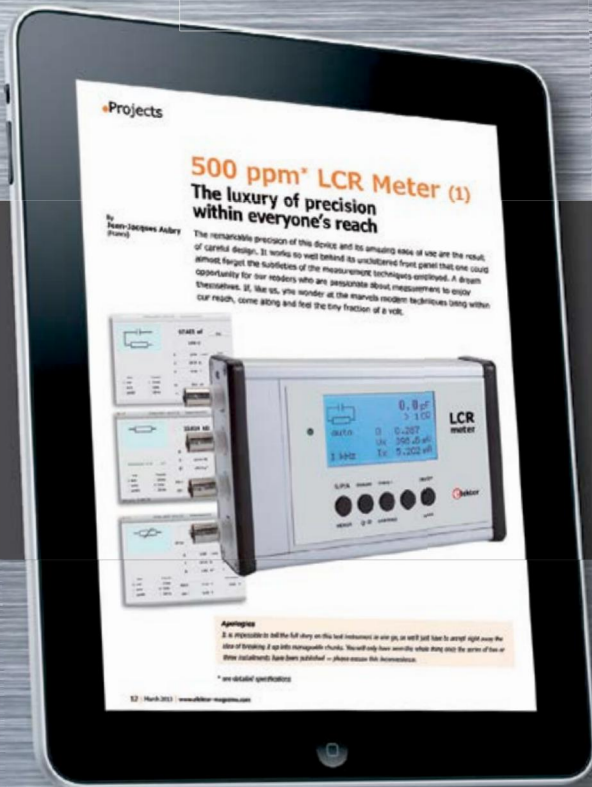
Join the Elektor Community

Take out a GOLD Membership now!



Your GOLD Membership contains:

- 8 Regular editions of Elektor magazine in print and digital
- 2 Jumbo editions of Elektor magazine in print and digital (January/February and July/August double issues)
- Elektor annual DVD-ROM
- A minimum of 10% DISCOUNT on all products in Elektor.STORE
- Direct access to Elektor.LABS
- Direct access to Elektor.MAGAZINE; our online archive for members
- Elektor.POST sent to your email account (incl. 25 extra projects per year)
- An Elektor Binder to store these 25 extra projects
- Exclusive GOLD Membership card



ALSO AVAILABLE:

The all-paperless GREEN Membership, which delivers all products and services, including Elektor.MAGAZINE, online only.



Take out your Membership now at www.elektor.com/members

sensors expo & conference

www.sensorsexpo.com

Sensing Technologies Driving Tomorrow's Solutions

Registration is Open for Sensors 2013

Sign up today for the best rates! www.sensorsexpo.com

Pre-Conference Symposia: June 4, 2013

Conference & Expo: June 5-6, 2013

Donald E. Stephens Convention Center • Rosemont, IL

Innovative Applications. Expert Instructors. Authoritative Content. Tomorrow's Solutions.

Register today to attend one of the world's largest and most important gatherings of engineers and scientists involved in the development and deployment of sensor systems.

CIRCUIT CELLAR **e**ektor
**Special
Subscriber
Discount!**

Register with code **A306L**
for \$50 off Gold and Main
Conference Passes.*

What's Happening in 2013:



"I come to Sensors Expo to see new things, learn about new technologies, and get my sensing questions answered."

—Nick Plessas, N.D.P.

Visionary Keynotes:



Connecting with the Emerging Nervous System of Ubiquitous Sensing

Dr. Joseph A. Paradiso

Associate Professor of Media Arts and Sciences, Responsive Environments Group
MIT Media Laboratory



Sensors in Space: The Robotic Exploration of Mars and its Environment

Dr. Raymond Arvidson

James S. McDonnell Distinguished University Professor, Washington University
Participating Scientist, NASA's Mars Curiosity Rover Mission

> Tracks:

- Big Data & Analytics
- Cutting-Edge Sensing Applications & Innovations **NEW**
- Energy Harvesting
- Gas Sensing **NEW**
- MEMS
- Novel Approaches to Measurement & Detection **NEW**
- Remote Monitoring
- Sensors @ Work **NEW**
- Wireless Sensing Solutions

> Full day Pre-Conference Symposia

> Technology Pavilions on the Expo Floor

- Big Data & Cloud
- Energy Harvesting
- Intelligent Systems **NEW**
- MEMS
- Wireless

> Co-location with Intelligent Systems Source **NEW**

> Best of Sensors Expo 2013 Awards Ceremony

> Networking Breakfasts

> Welcome Reception

> Sensors Magazine Live Theater

> And more!

Register at www.sensorsexpo.com or call 800-496-9877

OFFICIAL PUBLICATION:

sensors

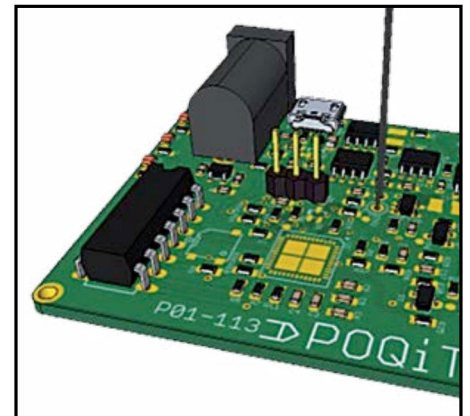
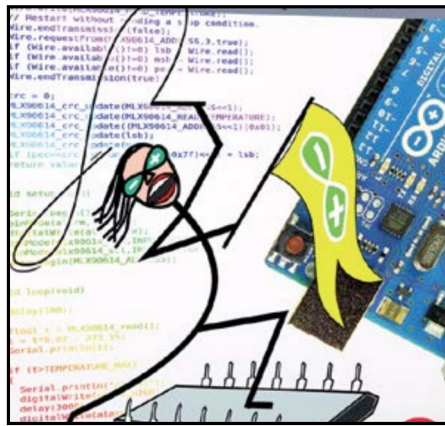
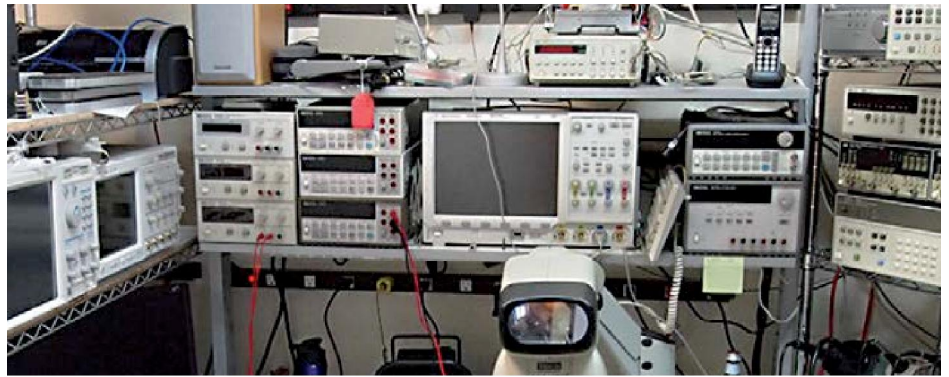
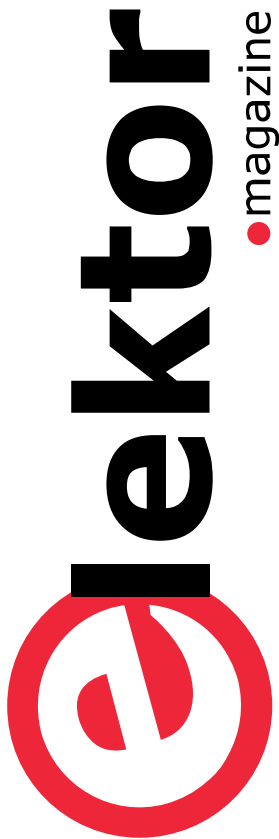
INDUSTRY SPONSORS:

CIRCUIT CELLAR **e**ektor

NEW CO-LOCATION:

Intelligent Systems Source

*Discount is off currently published rates. Cannot be combined with other offers or applied to previous registrations.



● Community

8 Elektor World

- Whip it Up
- Return of the Tapir e-smog meter
- Into the Nerd Cave
- Elektor Labs opens on Lamington Road, Mumbai
- Spark your Design

● Tech the Future

70 The Invisible User interface

There is a strong trend toward usability in industrial design and in human-computer interaction. That primarily revolves around efficiency, effectiveness, safety and ease of use. In short, the user interface.

● Projects

10 Embedded Firmware Library

The modular EFL (Embedded Firmware Library) lets you write applications and modules that can be easily transferred from one board to another. Meaning you are no longer locked in to a single microcontroller manufacturer but rather just write for embedded Linux in general.

18 500 ppm LCR Meter (3)

This month we get practical by covering the LCR Meter's circuit board assembly, parts lists, casing, and calibration in just 8 pages.

26 From BASIC to Python

Python is a high-level language particularly associated with small computer systems such as the popular Raspberry Pi. The language is marked by clarity and conciseness. Gone are the brackets and

semicolons which gave headaches in Pascal, C and Java, not to mention BASIC.

32 Taming the Beast (4)

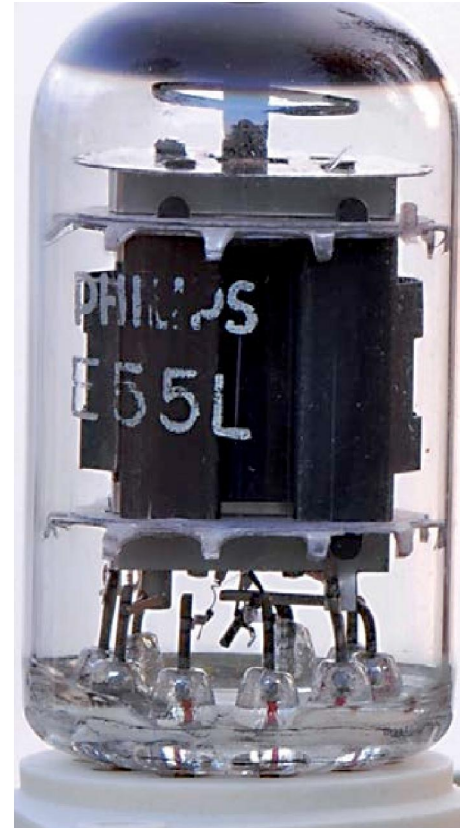
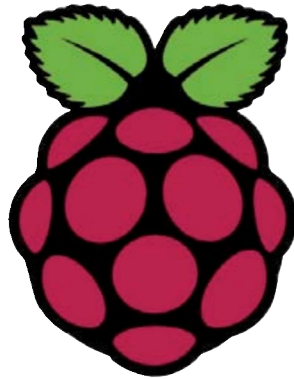
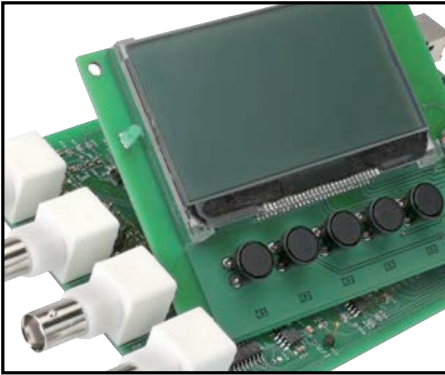
Once you start using virtually the entire FPGA chip for an application written in a hardware description language (HDL), examining the circuit inside the chip is difficult at best, so simulation is in order

40 Elektor Linux Board Extension (2)

In this second and closing installment we look at the hardware and software interaction between the buttons and the LC display on the board — all with a view to learning how to program a well thought out menu.

43 OTA-based Triangular Wave Generator

Although many applications of



● Industry

OTAs have been described, the basic circuit of a non-sinusoidal oscillator exploiting an OTA's specific advantages is a rare find. This design brief describes a combined triangular and rectangular wave oscillator with two OTA circuits.

46 BaroStick

This USB barometric pressure sensor uses a Bosch sensor to measure the air pressure and temperature, after which this data can be displayed graphically and stored using Windows software.

50 Program Like the Professionals

State diagrams see increasing use in software development. This article looks at how you can use them to make your programs more robust and free of bugs

58 Raspberry Pi: one year later, one million sold

An interview with Peter Lomas, a hardware designer behind the Raspberry Pi computer.

62 News & New Products

● Labs

54 Sorry about my English

Even if your English is fraught with gramatikall miztakez, do submit your project at elektor-labs.com.

56 PCB production pitfalls

Some issues to consider when wave soldering SMD components.

57 Ceramic C's off the mark

Here's an intriguing phenomenon concerning the value of 150-pF capacitors before and after soldering

● Magazine

74 Retronics: Wandel & Goltermann NE-171 MultiVoltage Tube PSU (ca. 1963)

This power supply for science labs and radio/TV repair outfits has just about everything you can desire or dream of when working with tubes. Series Editor: Jan Buiting.

78 Hexadoku

Elektor's monthly puzzle with an electronics touch.

79 Gerard's Columns: Credibility

A column or two from our columnist Gerard Fonte.

82 Next Month in Elektor

A sneak preview of articles on the Elektor publication schedule.

No. 53, May 2013
ISSN 1947-3753

Elektor Magazine is published 10 times a year including double issues in January/February and July/August at \$80 per year, Canada add \$15 per year; by

Elektor International Media LLC
111 Founders Plaza, Suite 300
East Hartford, CT 06108.

Phone: 860.289.0800
Fax: 860.461.0450
www.elektor.com

Elektor is also published in French, Spanish, German and Dutch. Together with franchised editions the magazine is on circulation in more than 50 countries.

Memberships:

Elektor USA
P.O. Box 462228
Escondido, CA 92046.

Phone: 800-269-6301
E-mail: elektor@pcspublink.com
Internet: www.elektor.com

Head Office:

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

Advertising:

Strategic Media Marketing
Peter Wostrel
2 Main Street
Gloucester MA 01930.

Phone: 978-281-7708,
Fax: 978-281-7706
E-mail: peter@smmarketing.us

Advertising rates and terms available on request.

Copyright Notice

The circuits described in this magazine are for domestic use only. All drawings, photographs, printed circuit board layouts, programmed integrated circuits, disks, CD-ROMs, DVDs, software carriers and article texts published in our books and magazines (other than third-party advertisements) are copyright Elektor International Media b.v. and may not be reproduced or transmitted in any form or by any means, including photocopying, scanning and recording, in whole or in part without prior written permission from the Publisher. Such written permission must also be obtained before any part of this publication is stored in a retrieval system of any nature. Patent protection may exist in respect of circuits, devices, components etc. described in this magazine. The Publisher does not accept responsibility for failing to identify such patent(s) or other protection. The submission of designs or articles implies permission to the Publisher to alter the text and design, and to use the contents in other Elektor International Media publications and activities. The Publisher cannot guarantee to return any material submitted.

© Elektor International Media b.v. 2013
Printed in the USA

Call for Papers

One of the most frequent questions I get, usually by email but occasionally by telephone or letter (!) too, is "can I contribute to your wonderful publication and if so, what are the requirements and the specific subjects you are interested in?" As of this year, the answer is: "If you have a project to present, please do it at www.elektor-labs.com". To this I usually add a few encouraging words and some help to get started on the .labs website. Now if this sounds like a straightforward approach to you, you should know that some of our competitors simply do not accept articles from anyone outside their circle of 'approved authors', which is another way of saying that Elektor is an independent publication with paying members and paid authors.

The idea behind using our .labs website as a project evaluation tool is that the community of Elektor readers and members can actively participate in enhancing the design, hopefully passing from *Proposal*, through *In Progress*, right up to *Finished* — meaning it gets elektorized in all respects.

Non-project contributions from companies, researchers and industry workers are also welcomed; these are given the usual evaluation by the team of editors, engineers and consultants.

Now, for the solemn bit: about half the article proposals reaching us through all international channels do not make it to publication in the magazine for a variety of reasons. Like uninventive use of components; the use of obsolete components; rehashing manufacturer's datasheets or old Elektor articles (!); vague circuits nicked from websites, and poor electronic design. The rest is gladly considered for publication on the .labs website to begin with, no matter if the piece is poorly written (see page 54) or the prototype built on prototyping board — in general we are good humored with a keen eye for originality. Even if it takes a while for us to get back to you due to the work load here at Elektor HQ, give us a try and eventually see your name (and circuit!) in print — it's by no means difficult, we're here to help.

Enjoy reading this edition of Elektor

Jan Buiting, Managing Editor



The Team

Managing Editor:	Jan Buiting
Publisher:	Hugo Van haecke
Membership Manager:	Shannon Barraclough
International Editorial Staff:	Harry Baggen, Eduardo Corral, Wisse Hettinga, Denis Meyer, Jens Nickel
Laboratory Staff:	Thijs Beckers, Ton Giesberts, Luc Lemmens, Clemens Valens, Raymond Vermeulen, Jan Visser
Graphic Design & Prepress:	Giel Dols, Jeanine Opreij, Mart Schroijen
Online Manager:	Daniëlle Mertens
Managing Director:	Don Akkermans



USA
Hugo Van haecke
+1 860-875-2199
h.vanhaecke@elektor.com



United Kingdom
Wisse Hettinga
+31 46 4389428
w.hettinga@elektor.com



Germany
Ferdinand te Walvaart
+49 241 88 909-17
f.tewalvaart@elektor.de



France
Denis Meyer
+31 46 4389435
d.meyer@elektor.fr



Netherlands
Harry Baggen
+31 46 4389429
h.baggen@elektor.nl



Spain
Eduardo Corral
+34 91 101 93 95
e.corral@elektor.es



Italy
Maurizio del Corso
+39 2.66504755
m.delcorso@inware.it



Sweden
Wisse Hettinga
+31 46 4389428
w.hettinga@elektor.com



Brazil
João Martins
+55 11 4195 0363
joao.martins@editorialbolina.com



Portugal
João Martins
+351 21413-1600
joao.martins@editorialbolina.com



India
Sunil D. Malekar
+91 9833168815
ts@elektor.in



Russia
Nataliya Melnikova
+7 (965) 395 33 36
Elektor.Russia@gmail.com



Turkey
Zeynep Köksal
+90 532 277 48 26
zkoksal@beti.com.tr



South Africa
Johan Dijk
+31 6 1589 4245
j.dijk@elektor.com

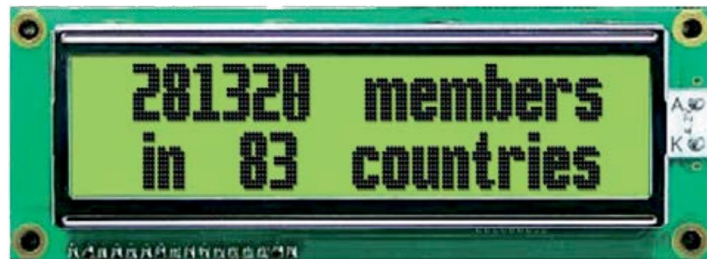


China
Cees Baay
+86 21 6445 2811
CeesBaay@gmail.com

Our network



Connects you to



Supporting Companies

	AP Circuits www.apcircuits.com39		Pololu www.pololu.com45
	Beta Layout www.pcb-pool.com63		Saelig www.saelig.com61
	Cleverscope www.cleverscope.com67		Sensors Expo 2013 www.sensorsexpo.com 3
	DLP Design www.dlpdesign.com63		
	ExpressPCB www.expresspcb.com65		
	EzPCB www.ezpcb.com39		
	Front Panel Express www.frontpanelexpress.com67		
	HuMANDATA www.hdl.co.jp/EL/64		

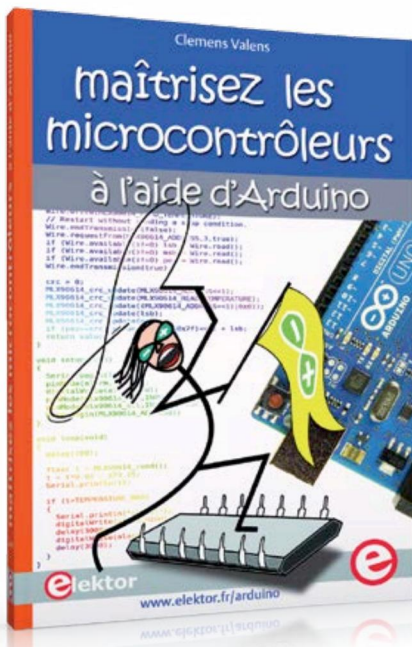
Not a supporting company yet?

Contact Peter Wostrel (peter@smmarketing.us, Phone 978-281-7708, Fax 978-281-7706)
to reserve your own space for the next edition of our members' magazine

Elektor World

Compiled by
Wisse Hettinga

Every day, every hour, every minute, at every given moment designers and enthusiasts are thinking up, tweaking, reverse-engineering and developing new electronics. Chiefly for fun, but occasionally fun turns into serious business. Elektor World connects some of the events and activities — for fun and business.

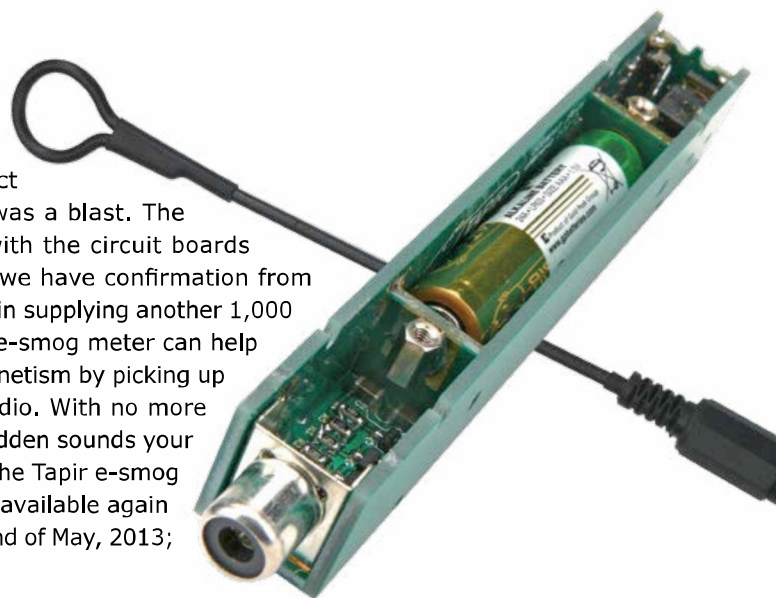


■ ■ Whip it up

If you follow our adventures you know the Elektor employees are not working from one place — the majority of us are located in Holland and we have colleagues in Spain, The United States, Italy, India, France — it's a long list. Clemens Valens is working from France and combines his Elektor Labs managing duties with... writing! In the quiet hours he has been working on a book with an 'edge'; an Arduino edge. In the book he explains how you can use the Arduino platform to 'tease your neighborhood', meanwhile learning to get a command of this interesting hardware platform. Admittedly, when we saw the book he surprised us with the cover — we see a microcontroller (so far, so good), but then there is a female figure using a whip, presumably to get the controller controlled properly. So, this really looks like an edgy book and we decided to have it translated in German and English as well. But that whip!? Kinky.

🇬🇧 Return of the Tapir e-smog meter

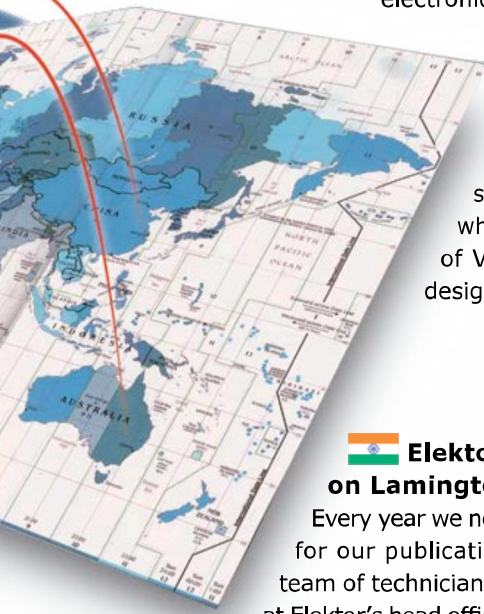
Last year we pushed a small project called the Tapir e-smog meter. It was a blast. The design was easy and remarkable with the circuit boards doubling as the housing. This year we have confirmation from Eurocircuits that they will participate in supplying another 1,000 pieces of that wonderful PCBs. This e-smog meter can help you discover the world of Electromagnetism by picking up stray signals and turn them into audio. With no more than an earpiece you can hear the hidden sounds your smartphone or tablet is producing. The Tapir e-smog meter boards and the DIY kit will be available again from the Elektor web shop from the end of May, 2013; follow www.elektor.com/tapir.



 **Into the Nerd Cave**

They are all over the world — attics, cellars, garages, small rooms, stuffed with electronic equipment. They are the treasure troves of electronics where many of you spend hours working, thinking and discovering new applications.

Forget the high-tech labs with mega money sponsoring — this is where the true spirit of electronic design dwells! Our sister magazine *Circuit Cellar* (what's in a name) started printing pictures of the 'nerd caves' where electronics enthusiasts toil. In the picture you see the setup of Vincent Himpe's workspace. Vincent is a veteran electronics designer and writer of Elektor's *LabWorX* series books.



 **Elektor Labs opens on Lamington Road, Mumbai**

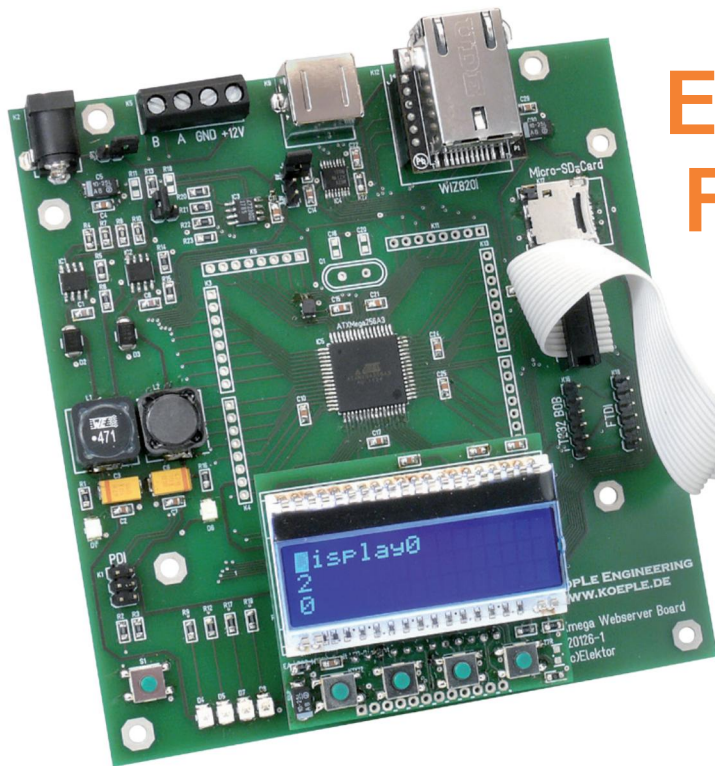
Every year we need around 200 project designs for our publications. Traditionally we have a team of technicians working close to the editors at Elektor's head office, but the growing number of projects demands more capacity. With the opening of a new Elektor Labs department on Lamington Road, Mumbai, right in the heart of India's biggest electronics market, we invest in capacity to complete our mission: test it first, then improve it, then publish. Lamington Road is packed with shops selling and stocking more electronics components than you can dream of. When our Mumbai designers need a component, they just pop 'round next door — what a great place to work! The Elektor Mumbai Labs team is pictured here. Left to right we have Krishna Chandran, Sunil Malekar, Clemens Valens, Nandini Singh, and Shreenivas G M Shree.



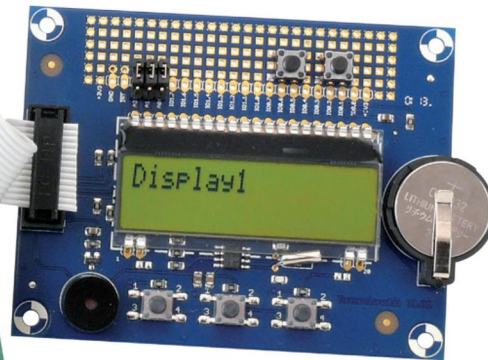
  **Spark your Design**

Elektor and RSComponents have entered into a partnership around the DesignSpark brand. First there is *DesignSpark* magazine. This is the rebranded version of the well-known title *eTech* and is published to the full RSComponents client database. The new magazine covers the latest developments on the new hardware platforms, and will also host a new series of Elektor projects. The other activity will focus on the launch of a special design-and-build section on the DesignSpark.com website. The project PCBs will be specifically designed using DesignSpark PCB, the free CAD software available at www.designspark.com. In *Elektor* magazine and on our website www.elektor-labs.com we will carry special pages and activities supporting this new initiative.

(130051)



Embedded Firmware Library



Build microcontroller projects faster!

By Jens Nickel

The modular EFL (Embedded Firmware Library) is not just another library aimed at ‘rapid application development’. It lets you write applications and modules in a hardware-independent fashion, so that they can easily be transferred from one board to another. This means you are no longer locked in to a single microcontroller manufacturer but rather just write for embedded Linux in general. Thanks to its wide range of interface functions covering A/D converters, PWM generators, displays, TCP/IP, SD cards and more, the library is ideal for beginners in (C) programming.

Features of the Embedded Firmware Library

- Portable in principle to any microcontroller for which an ANSI C compiler exists
- 32 KB flash and 2 KB RAM recommended
- Well-documented API
- Currently supported microcontrollers: ATmega328/324/664 and ATmega256A3
- Currently supported boards: ElektorBus experimental node, XMEGA web server board [9], Linux extension board [8]; Arduino Uno Rev. 3; further boards in preparation
- Simple to port to new boards
- Currently available higher-level libraries: LEDs and buttons, A/D converter and multiplexer, stepper motor, ElektorBus, display, WIZnet TCP/IP module, SD card (raw data)
- LGPL open source license

In 2012 we presented a C library for the ElektorBus microcontroller nodes [1]. One of its distinctive features is the separation between the hardware-specific part and the code module responsible for implementing the protocols. This makes it easy to adapt the library for other types of microcontroller and other boards: only the hardware-specific part needs to be changed, while the rest remains the same. However, there is more to a typical bus application than just composing messages and sending and receiving them: there are LEDs and relays to be turned on and off, analog values to be read and motors to be controlled. Beyond that it would also be desirable if the microcontroller could easily communicate with other chips, such as a TCP/IP interface module or an SD card.

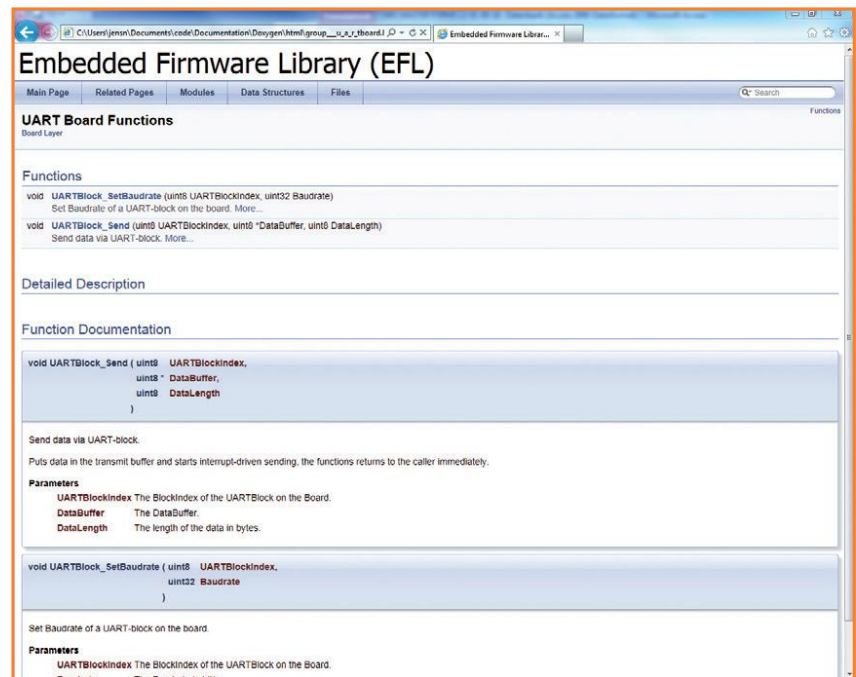
Parallel programming

In the course of programming the library the concept became better developed, and step by step it became more flexible and powerful, with capabilities well beyond what was required for the ElektorBus. I was fortunate to find in reader Michael Busser (a member of the ElektorBus Google group) a co-developer who contributed many clever code modules, functions and ideas. So we could work in parallel on the code in spare moments we uploaded it to the well-known project hosting site SourceForge [2] under the name 'Embedded Firmware Library (EFL)'. SourceForge is dedicated to open source projects, which fits well with the *Elektor* philosophy: we chose the LGPL license [3]. The most important feature of the SourceForge platform is its integrated SVN-compatible version control system [4] which can be used under Windows with (for example) TortoiseSVN [5]. A click of the mouse is all it takes to upload ('commit') one's changes or to download the newest version of the code ('update'). Over time we built up a considerable body of functions, and so we decided to use an automated documentation system. By simply adding comments in a prescribed format to the source code, we can use the open source program Doxygen [6] to automatically generate well-organized documentation for all the functions. The documentation is in HTML and so can be viewed in a browser (see **Figure 1**). Beautiful! The Doxygen EFL documentation can be downloaded from the project web page at [7].

Building blocks

The Embedded Firmware Library is designed as a set of building blocks, with the various modules in each layer being interchangeable with one another. There are three distinct layers: the microcontroller code resides in the bottom layer; above that is the 'board' layer and finally there is the 'extension' layer (see **Figure 2**). The layers correspond to the structure of a real project design: different (pin-compatible) microcontrollers can be used on the same board, or the same microcontroller can be used on different boards. A good example of this is the ATmega328, as used on a wide range of Elektor boards as well as on the Arduino Uno.

If we define an expansion connector and its pin-out then we can easily combine different microcontroller boards and different expansion boards. An example of this is the Linux extension board



described elsewhere in this issue [8]. This board can be used not only in conjunction with the *Elektor* Linux board, but also with the XMEGA web server board [9] which we will describe in the June issue (see large picture). The EFL can also be used in a system comprising a motherboard carrying various peripherals into which a range of processor boards can be plugged. There are three different code modules, corresponding to the microcontroller, the microcontroller board and the extension board. Each

Figure 1. The 'Doxygen' tool helps produce clear function documentation.

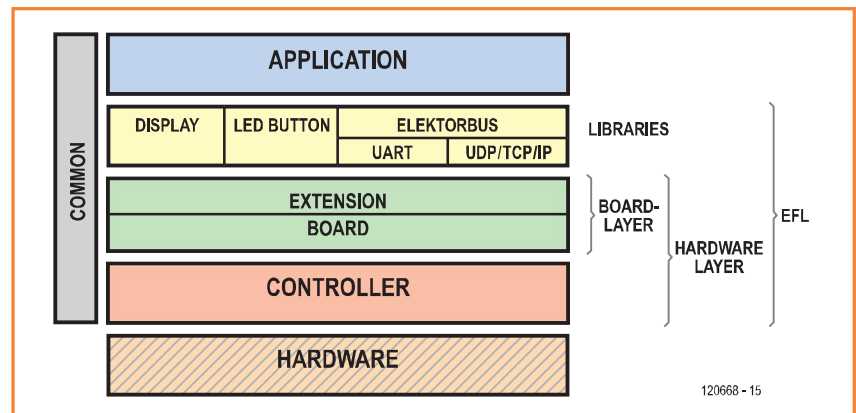


Figure 2.

Thanks to a clearly separated hardware-dependent layer an EFL-base project can quickly be ported to a new platform. This is very useful when hardware changes prove necessary during prototyping.

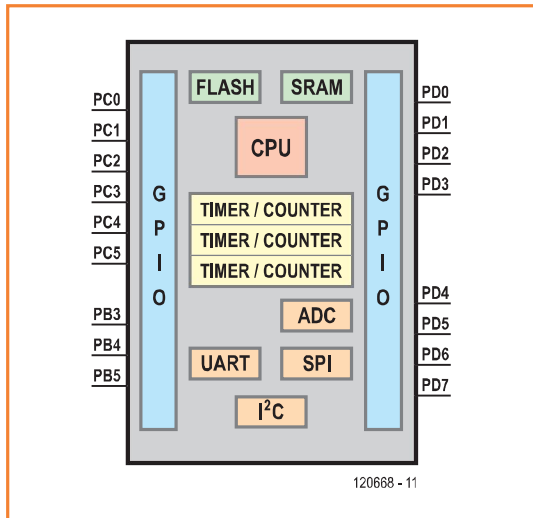


Figure 3. A greatly simplified block diagram of a typical microcontroller (ATmega328).

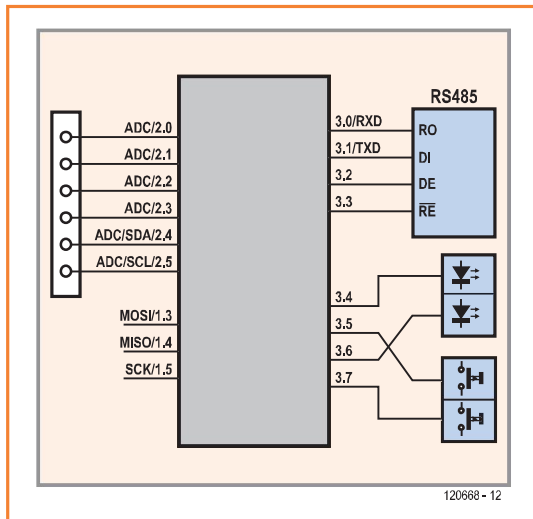


Figure 4. The board code developer can treat the microcontroller as a 'black box': only its port pin connections need to be known.

module consists of a '.h' header file and a '.c' code file. It is therefore necessary to add the files ControllerEFL.h/.c, BoardEFL.h/.c and ExtensionEFL.h/.c to your project. The file-names are the same for all the different types of microcontroller and board supported by the EFL. To avoid name conflicts the code downloadable at [11] is organized into different subdirectories named after the various microcontrollers and boards. More on this can be found in the additional documentation, also available at [7].

The black box principle

In the ideal case the three code modules could be written by independent developers: the developer of the microcontroller code should not need to know to what board the microcontroller will be fitted, and the microcontroller board code

developer should not need to be aware of what extension boards might be present.

The same goes in the other direction, from the highest level of the library to the lowest: the developer of an application should not need to be concerned with the details of how the code works at the next level down. Likewise the microcontroller board code developer should just need the circuit diagram of the board and a description of the microcontroller's port pins and their functions, and should not need to worry about the innards of the microcontroller itself (Figure 3), treating it instead as a 'black box' (Figure 4). The designer of an extension board should similarly only be concerned with the pinout of the expansion connector via which it communicates with the microcontroller board; and not have to worry about which signals are connected to which pins on the microcontroller (Figure 5).

Higher-level libraries

Above the hardware layer there are modules that provide higher-level functions, which we call 'libraries'. Examples of this are the module included in the current code base to drive HD44780-compatible alphanumeric LCD panels, a driver for a WIZnet TCP/IP module, a simple stepper motor driver and a module for reading and writing raw data from and to an SD card. The black box principle extends to these modules too: if you wish to change or extend the LCD library, the display (or perhaps more precisely the display controller) can simply be treated as a block (see Figure 6). There is no need to know which pins on the display are connected to which pins on the microcontroller; and it makes no difference to the developer of the display library whether the display controller is connected in four-bit mode or over SPI, leaving him to concentrate on the HD44780 command and data sequences.

Application code

The developer of applications has easy access to all the blocks on the board, and only needs to know which blocks are available. Now it is possible that there is a LED block available on both the microcontroller board and on the extension board, perhaps with two LEDs and four LEDs respectively (see Figure 6). The developer can write code which turns on the first LED in block 0 and the third LED in block 1 without having to worry about where they are physically located. Even if an LED, button, relay or other digital I/O

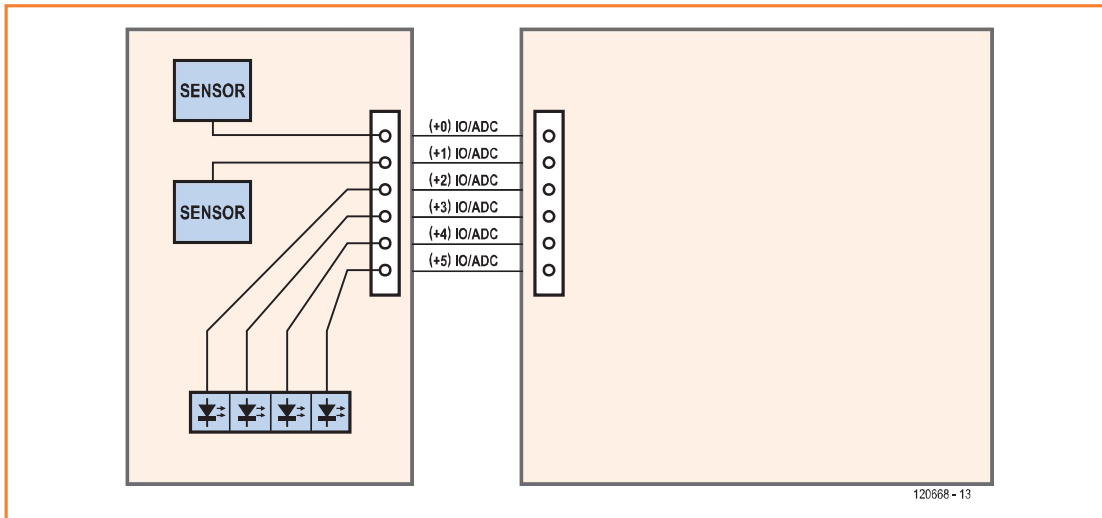


Figure 5.
The developer of the code for an extension board only has to know about the connections on the expansion connector and their possible uses.

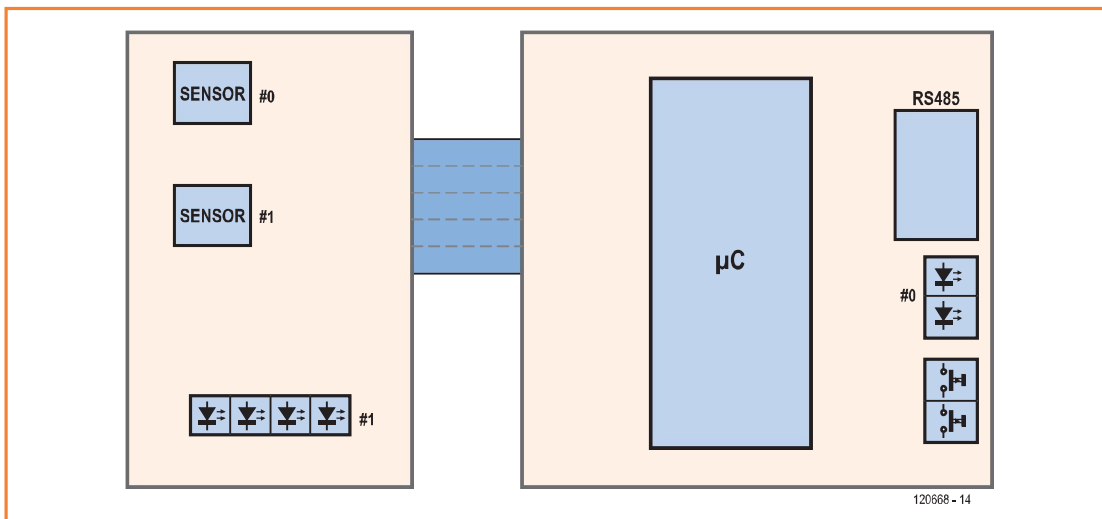


Figure 6.
The application developer only needs a block diagram of the system, and does not need to know the details of how the boards themselves are wired. The two groups of LEDs are addressed via their block numbers, 0 and 1.

device is connected via a port expander chip rather than directly to a pin on the microcontroller the same library functions can be used to interface to it. Buttons connected to analog inputs, as on the Linux extension board, can also be treated just like real ‘digital’ buttons (see large picture).

A consistent set of functions

We only have space here to describe the internal operation of the library as a whole in general terms (see text box). Further information on the hardware layer is available for download at [7]: this documentation will be of use to anyone who needs to modify the hardware layer to support a different board.

The underlying principle behind the library is that of a consistent set of functions (or API) exposed by each layer to the layer above it. The micro-

controller API always provides the same set of functions (or at least, to the extent that these functions are supported by the microcontroller in question). These comprise firstly set-up functions to initialize the peripheral units integrated into the microcontroller, such as the A/D converter, SPI and UART interfaces, counter/timers and so on, and then functions to set the parameters associated with these units (such as the baud rate of the UART) and finally functions to operate the unit: take readings from the A/D converter, send and receive bytes over the UART or SPI, and so on. For more details see the **Table**.

At the board and extension module level there are low-level functions for talking to the various individual functional units such as the display and stepper motor driver. These functions are needed in order to encapsulate the various pos-

Table: Microcontroller functions

(Version of March 2013, omitting set-up and parameter-setting functions; for updates see [11])

Function	ATxmega256A3	ATmega328
Enable/disable interrupts	X	X
Wait... (time delay)	X	X
Set pin	X	X
Read pin	X	X
Read A/D conversion result	X	X
Start A/D conversion	X	X
Set DAC value	X	-
Send UART data (interrupt driven)	X	X
Receive in circular buffer	X	X
SPI master (send/receive byte)	X	X
I2C master		
Read/write bytes (combined)	X	X
Start/stop timer	X	X
Count pulses on a pin	X	*
Timer-related:		
- increment	X	X
- call function	X	X
- toggle pin	X	X
- toggle at variable speed	X	X
- measure frequency	X	*
PWM output	X	*
Software PWM	*	*
X: implemented *: planned -: not available		

sible combinations of interface wiring and protocols, such as LCDs driven using a four-bit parallel interface or over SPI. The higher-level libraries access these low-level functions to provide a consistent interface to application-level code.

Example: display

As an example we shall look at controlling an alphanumeric LCD. The application code will put up a simple greeting on the first line of the display by calling the following functions in the display library:

```
Display_LibrarySetup();
Display_WriteString(0, 0, "Welcome!");
```

The first zero is the index number of the display to be addressed (as conventional in C code, we always count from zero in the EFL). In order to carry out the write command the display library

makes several calls to the function

```
void Display_SendByte(uint8
DisplayBlockIndex, uint8 ByteToSend,
uint8 DATABYTE_COMMANDBYTE)
```

at the board level to send the individual data or command bytes to the display controller. In the case of the XMEGA web server board the display is connected over an SPI bus. In this implementation the function

```
uint8 SPIMaster_TransceiveByte(int8
Handle, uint8 Databyte)
```

at the microcontroller level is used to send a byte to the display. The CS and RS inputs to the display also need to be driven appropriately.

Software download

The software download available from the web site accompanying this article [7] includes an application example. It is a similar project to [1]: the ElektorBus experimental node [10] is connected to an extension board that carries two sensors and four extra LEDs (**Figure 7**). These are built onto half of an Elektor Universal Prototyping Board (UPBS, a.k.a. ELEX), and the download also includes a LochMaster file showing the layout.

When the software is opened in Atmel Studio 6 the code files will appear in prescribed (virtual) subdirectories called 'Hardware', 'Common' and 'Libraries' (see **Figure 8**). You will also see the main code file (ExperimentalNodeEFL.c) that provides the basic framework for an EFL-based application. It is also possible to see which functions need to be called when the program starts up. The function ApplicationSetup() first provides all the configuration data for the bus (we no longer need the special configuration files ElektorBusNode.h/.c). Then we have the setup functions for the various library modules: usually it is not necessary to change any of their parameters. The call SwitchLED(1, 0, 0N); turns on the first LED on the extension board to confirm that the program has started up.

Pressing the first button on the experimental node toggles the red test LED and the third LED on the extension board. Under control of the bus scheduler the nodes send the readings from the two sensors to the PC. The second LED on the extension board blinks in synchrony with this activity. The fourth LED is controllable from the PC.

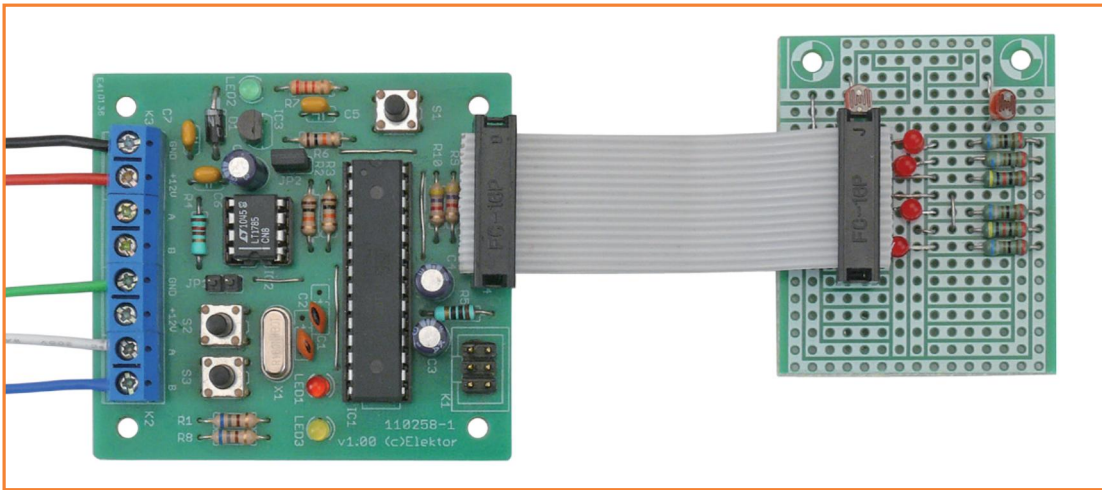


Figure 7. ElektorBus experimental node with extension board (two sensors and four LEDs). The board layer is easily adapted to another board, such as the forthcoming stepper motor interface board.

The file `ExtensionEFL.c` also includes the setup code for a small relay board: which part of the code is actually compiled depends on a `#define` directive in `ExtensionEFL.h`.

As mentioned above, the project download contains the extra documentation for the EFL and the Doxygen output as well as the complete code

base, including additional libraries for displays and the WIZnet TCP/IP module, all in source code form.

The future

This article has only been a brief introduction to the EFL. We are planning a step-by-step article

A deeper look inside EFL

Port pins

The functions associated with digital I/Os (for example to set the output level on a pin) are called with a parameter called 'Portpin', a two-byte unsigned integer ('uint16'):

```
void IO_SetPinLevel(uint16 Portpin, uint8 PinLevel)
```

The first byte of the Portpin parameter denotes the port (or group of I/O pins): in the case of an AVR microcontroller zero corresponds to port A, one to port B and so on. This lets the function work out which registers will be affected by the call. The second byte of the parameter specifies the pin within the port: on a typical eight-bit microcontroller this would be a value from zero to seven inclusive. The port pins can be labeled '0.0', '0.1', ..., '1.0', '1.1' and so on in the circuit diagram (see Figure 4).

Features, units and handles

The various peripheral functions offered by a microcontroller (such as A/D converters, UARTs, SPI interfaces and timer/counters) are called 'features' in the EFL. For each feature there can be several 'units': for example, in the ATxmega256A3 there are three SPI units each providing the SPI feature. Each unit is associated with particular pins on

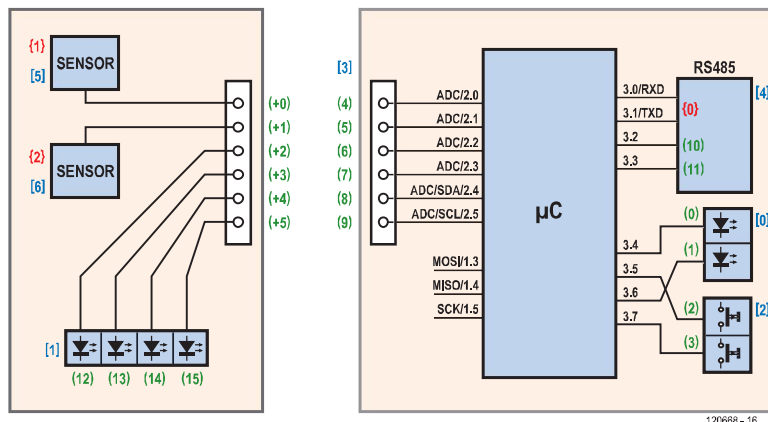
the microcontroller, as specified in the device's data sheet. The board code, however, cannot know anything about the internal numbering of the units or the registers that control the units. So, when a unit is set up the microcontroller-level function is passed the identity of one of the port pins that forms part of the interface. For example, in the case of an SPI unit, the set-up function is of the following form:

```
int8 SPIMaster_Setup(uint16 Portpin)
```

The set-up function in the microcontroller code determines the unit number of the SPI interface to be used from the specified port pin. From this it can in turn determine which registers need to be initialized. For subsequent access to the port the unit number and port pin are stored in an array called 'Map': the function `SPIMaster_Setup` returns an index into this array (from zero upwards) as a handle. The board code can now use the handle as required to specify the SPI interface unit when sending and receiving data:

```
uint8 SPIMaster_TransceiveByte(int8 Handle, uint8 Databyte)
```

Other features work in just the same way



Board pins and blocks

The individual port pins of the microcontroller are connected to various hardware units on the board, such as LEDs, buttons or an RS-485 driver. The port pins corresponding to these 'blocks' are stored in an array by the board-level code when it is initialized.

The indices to this array are called 'board pins'.

For example, board pins (0) and (1) are assigned to the LED block in Figure 4; board pins (2) and (3) are assigned to the button block; and board pins (4) to (9) are assigned to the expansion connector.

Board pin	Port pin
(0)	3.4
(1)	3.6
(2)	3.5
(3)	3.7
(4)	2.0
(5)	2.1
(6)	2.2
...	
(9)	2.5

The function `Extension_Init()` in the extension code module reserves board pins (12) to (15) for the extra LEDs on the extension board (see Figure 5). The corresponding port pins 2.2 to 2.5 are determined indirectly via the board pin entries (6) to (9) for the expansion connector:

```
(12)  2.2    // = (6)
(13)  2.3    // = (7)
(14)  2.4    // = (8)
(15)  2.5    // = (9)
```

The board pin array now contains two blocks for LEDs: (0) to (1) and (12) to (15). These ranges (or more precisely the first board pin in the range and the number of pins in the range) are stored in an array called 'Block' for each block, accompanied by a constant indicating the type of block:

BlockIndex	BlockType	First	Count
[0]	BLOCKTYPE_LED	0	2
[1]	BLOCKTYPE_LED	12	4
[2]	BLOCKTYPE_BUTTON	2	2
...			

If a LED library wants to control a particular LED in a particular block, it simply calls the function

```
void SwitchDigitalOutput(uint8 BlockIndex, uint8
Position, uint8 ON_OFF)
```

at the board level.

The function `SwitchDigitalOutput()` determines the board pin from the parameter `BlockIndex` and the position of the LED within the block, and from that determines the actual port pin involved.

The Block array also stores whether a high or a low logic level on the port pin is required to turn the connected LED on, as this can differ from board to board.

Resources

The great flexibility of the EFL comes at a price: the library takes up not only flash storage for the code itself but also RAM (adjustable from a few hundred to many hundreds of bytes). Performance (speed of program execution) also suffers, in particular for programs that rely on toggling output pins quickly. For this reason we have implemented special functions to switch digital outputs under timer control, removing the need to set the pins high and low with repeated function calls. But in any case performance is good enough for tasks such as dimming several LEDs.

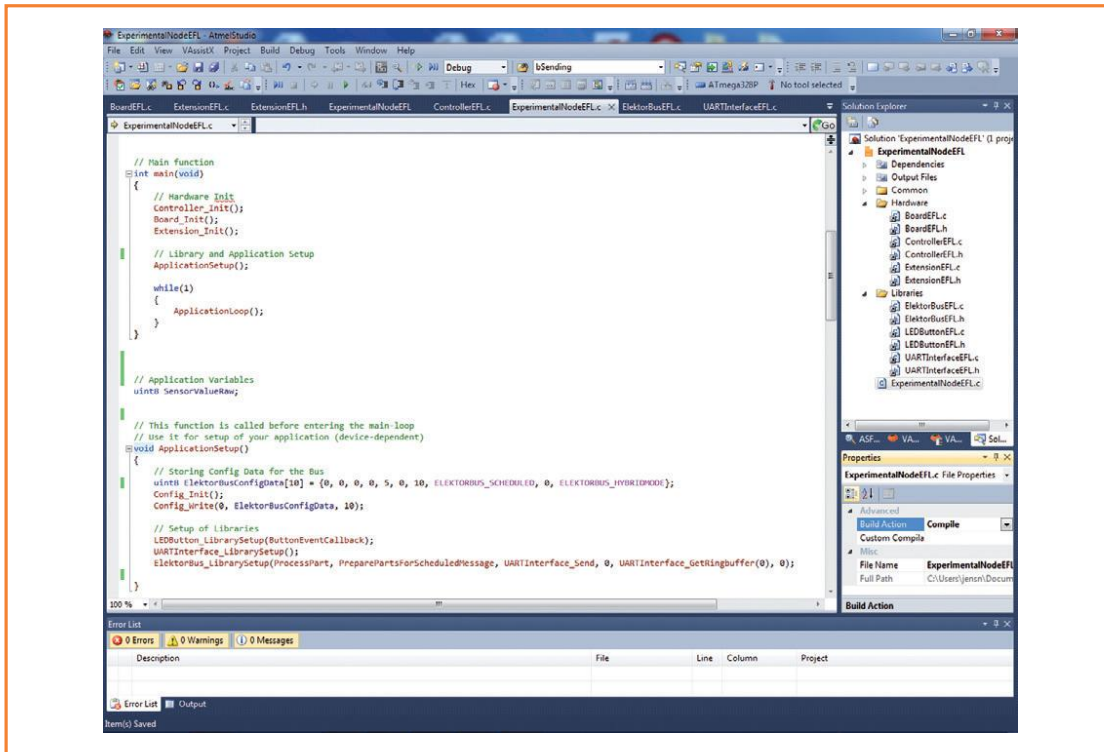


Figure 8. Screenshot of an example project in Atmel Studio 6. The code files comprising the project are stored in specified (virtual) folders, which makes it easier to understand its structure.

specifically aimed at beginners in C programming. We will also be using the EFL in future *Elektor* projects, starting with a stepper motor board for the *ElektorBus* and continuing a.s.a.p. with the XMEGA web server board.

Readers will perhaps be interested not just in using the EFL, but also in extending it and adapting it (see the additional documentation at [7]). The author welcomes any suggestions, in particular in the area of the microcontroller API, please email j.nickel@elektor.de, subject: EFL. Extensions to support further microcontrollers, boards and peripheral blocks would also be welcomed. Regular updates will be published on the *Elektor Labs* website [11].

(120668)

Internet Links

- [1] www.elektor-magazine.com/120582
- [2] <http://sourceforge.net/projects/embeddedlib/>
- [3] www.gnu.org/licenses/lgpl.html
- [4] <http://subversion.apache.org>
- [5] <http://tortoisesvn.net>
- [6] www.doxygen.org
- [7] www.elektor-magazine.com/120668
- [8] www.elektor-magazine.com/120596
- [9] www.elektor-labs.com/xmegawebserver
- [10] www.elektor-magazine.com/110258
- [11] www.elektor-labs.com/EFL

RS485, TCP/IP and wireless: *ElektorBus* and dedicated protocols

The hardware independence of the *ElektorBus* library has been improved by introducing a code module *ElektorBusEFL.h/c* which encapsulates the details of the protocol used. Now it is easy to send and receive messages not just over an RS-485 interface, but also over RS-232, a UART interface at TTL levels or even TCP/IP: we have already tested this with the forthcoming XMEGA web server board and a WIZnet WIZ820io TCP/IP module. Another

interesting option would be to use an ISM band (418 MHz; 433 MHz) wireless module.

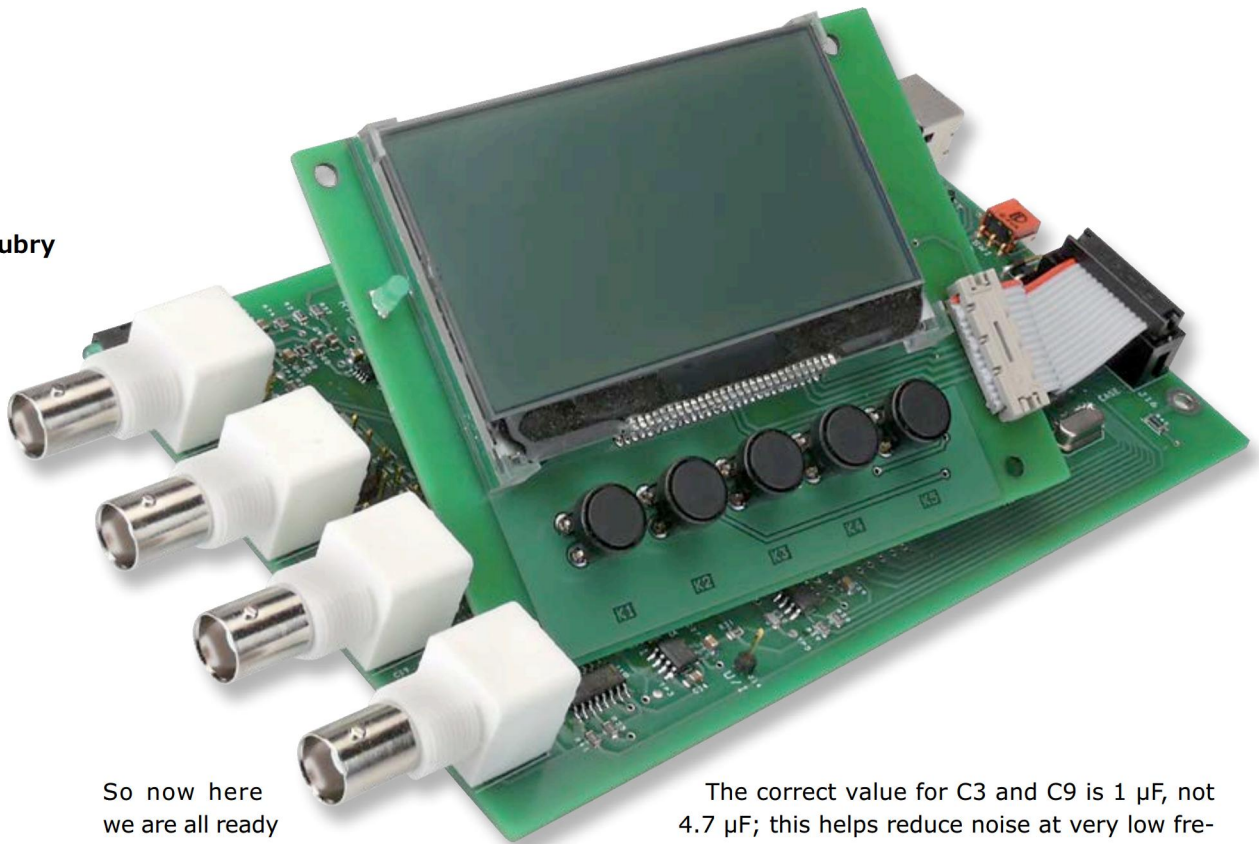
Likewise it is possible to write a replacement protocol library, which in turn would be independent of the communication medium, board and microcontroller. As you can see, there is a huge range of possibilities for experimentation.

500 ppm LCR Meter (3)

Part 3: PCBs, BOMs, assembly, calibration

After the flood of theory, design considerations and software descriptions in the first two installments we are ready to tackle practical matters.

By
Jean-Jacques Aubry
(France)



So now here we are all ready to start building the 500 ppm LCR meter. Most of this article is taken up describing the two boards, but before we finally get down to the nuts and bolts, we need to look again at the circuit diagram.

Inconsistencies and accuracy

We actually noticed a number of differences between the circuit published in the March 2013 edition and the final component list in this article. As so often, where there is a doubt, the component list is the one to believe, but as it hadn't yet been published, no-one was able to tell!

In **Figure 3** of the first installment [1], the value for crystal Y1 was missing: 24 MHz.

The correct value for C3 and C9 is 1 μF , not 4.7 μF ; this helps reduce noise at very low frequencies, attributed to the operation of the MAX7404 itself and revealed when measuring high-capacitance capacitors (1,000 μF and over): on the oscilloscope, looking at the ADC input signal, the mean value of low frequency fluctuates more and more as the capacitor value increases.

For U2, we need a normal MAX7404CSA (the extended temperature range of the MAX7404ESA is unnecessary here). For U10, we need an LM4040 (and not the LM4050 that I happened to have to hand, and which will also do). The amplifier used for U6 is an OPA365 and not an OPA354, which has too high an offset voltage. And lastly, for U19, it should be an FT232RL and not FT235RL (just a typo).

Specifications

The 500 ppm LCR Meter is an automatic impedance measuring bridge. The two guiding principles in designing this unit were maximum measuring accuracy and ease of construction. It measures resistance, capacitance, and inductance for components with impedances from 1 m Ω to 1,000 M Ω . Measurements can be made at three frequencies: 100 or 120 Hz (50 or 60 Hz power line), 1 kHz, and 10 kHz.

Two configurations are possible:

- Basic unit, with no display or keypad Works only when connected via USB to a computer running the user program. This program, developed from the *Qt* libraries, has been tested under *Windows XP*, *Windows 7*, *Linux (Ubuntu 11.04)*, and *MacOSX (Snow Leopard, Lion, and Mountain Lion)*.
- Basic unit + display & keypad extension. Also works stand-alone (without a PC) with an external 5 V supply via the USB cable (which can also be connected to a computer for the power).

Finally, we observed that to make one of the adjustments easier, a 10 Ω resistor must be fitted for R17, incorrectly shown on the circuit diagram as not fitted (NC). The other components marked NC should only be fitted if the calibration procedure requires it — we'll come back to this later.

Before plugging in your soldering iron, we suggest you make these corrections carefully on your circuit from the first article. A corrected XL version of this diagram will be put on line for download [3] along with the software and the other documents, including the PCB designs.

Construction

The layouts for the two PCBs that make up the LCR meter are given in **Figures 1–3**. Remember that **you only need the extension board for the standalone version of the device**; otherwise the main board is all you need. If you have the right tools and good experience in soldering SMDs manually, you can solder the components onto the bare boards yourself. You'll be all the more satisfied with the result. However, even though there's no special difficulty, it is a job for a specialist which you should only undertake if you know what you're doing.

Adhere scrupulously to the component list — not for the order of the components, of course, but for their values and tolerances. Apart from the four precision resistors, the rest are all standard 1% tolerance. R50 is indeed a 0 Ω resistor that makes it possible to link the digital and analog grounds in a single, precise spot; a solder bridge will also do! The tolerances of the various capac-

itors is stated in the component list. Make sure you adhere to the dielectric types specified, particularly for the NPO capacitors.

Alongside the crystal, you'll notice a point marked XTAL CASE on the board. This must be connected to the crystal case using a short piece of wire, as the metal screening is only effective if it is grounded. Above all, don't take too long with the soldering iron, to avoid damaging the crystal. Attention: If you solder in the components yourself, the microcontroller will be blank. You'll need to load it with the boot program; to do this, you'll have to use the USB DEBUG ADAPTER from *Silicon Laboratories* [3].

If you go for the standalone version, do pay attention as well to the full part number for components like the HE10 connector for the flatcable that is soldered directly into the display board (**Figure 4**). The height of a normal type would stop you mounting the module into the recommended case. The graphics display is soldered by a row of fine pins flanked by two normal-sized pins, but this isn't enough to keep it in place; to avoid its lifting accidentally, we secured it at the side opposite the pins with the help of a few spots of hot-melt glue.

In order to avoid any misunderstandings about the choice of components, we are making available to you on our site [3] the BOM (bill of materials) which supplements the component list given here. The size of the components and their pin spacing is what will determine the order in which you fit them. Don't forget the components mounted beneath the main PCB.

Those components marked NC will only be fitted (perhaps) later, for the calibration.

However, if you are not happy about soldering SMDs or if you don't have the time, all you need do is order the ready-to-use modules from www.elektorPCBservice.com, our professional production service: you will receive the modules through the mail, and all you'll then have to do is fit them into a case before calibrating them.

Like many electronic devices, an LCR meter is sensitive to electromagnetic radiation, in particular from the power line @ 60 or 50 Hz. So you'll need to use a metal case, e.g. the Hammond 1455L1601, which is perfect for a 160 x 100 mm board plus the extension, if used. Dimensioned mechanical drawings are available from our website [3].

Preparations

Drilling the holes for the BNC chassis sockets and LED D6 on one of the small sides presents no problem. For the USB connector and switch SW1 on the opposite side, you'll need a square-section tool... or a good file and some elbow-grease; in fact, for a tidy result, you'll need at least two files, a flat and a rat-tail, and possibly a triangular-section one too.

For grounding the main board to the case, you'll also need to drill a hole (3.2 mm Ø) in the bottom of the aluminum case, in line with the hole in the PCB located next to J16; use a 12 mm (0.5") M3 machine screw with washer and nut in conjunction with a 5.5 mm (0.2") spacer.

COMPONENT LIST Main circuit

Resistors

(default: SMD 0805 1%)

R1,R16,R17,R28 = 10Ω
 R2,R34 = 820kΩ
 R3,R5,R11,R13 = 8.2kΩ
 R4,R10,R47,R55,R56,R71,R72,R78,R81,R82,R94,R95 = 10kΩ
 R6, R58, R59 = 1.8kΩ
 R7,R100 = 5kΩ trimpot (Vishay-Sfernice TS53YJ502MR10)
 R8, R60, R62 = 16kΩ
 R9,R23,R25,R35,R38,R39,R41,R52,R68,R69,R70,R87,R88,R96 = 56Ω
 R12,R14 = 5.6 kΩ
 R15,R97 = NC (not fitted)
 R18 = 10kΩ 0.05% 10ppm (Panasonic ERA6ARW103V)
 R19 = 1Ω
 R20 = 1kΩ 0.05% 10 ppm (Panasonic ERA6ARW102V)
 R21 = 100Ω 0.05% 5ppm (Vishay-Dale TNPU0805100RAZEN00)
 R22 = 100kΩ 0.05% 10ppm (Panasonic ERA6ARW104V)
 R24,R26,R27,R29,R33,R36,R37,R40,R44,R57,R64 = 100Ω
 R30,R61,R76,R77,R80,R101 = 20kΩ
 R31 = 750Ω
 R32,R42,R49,R66,R93 = 100kΩ
 R43,R84,R85,R86,R89 = 2.2Ω
 R45,R73 = 39kΩ
 R46,R90,R91 = 680Ω
 R48,R51,R74 = 4.7kΩ
 R50 = 0Ω
 R53,R65 = 470Ω
 R54 = 1kΩ
 R63,R98 = 1.6kΩ
 R67 = 62Ω
 R75 = 5kΩ trimpot (Bourns 3266W-1-502LF)
 R79 = 4.3kΩ
 R81 = 7.5kΩ
 R83 = 30kΩ
 R92 = 430Ω
 R99 = 2kΩ

Capacitors

(default: SMD 0805)

C1,C2,C4,C10,C11,C12,C20,C26,C27, C28,C29,C31,C33,C34,C35,C36,

C37,C40,C41,C43,C44,C45,C46, C47,C48,C49,C50,C51,C53,C54,
 C61,C63,C65,C78,C80,C86,C87,C88 = 100nF 10% X7R 25V
 C3,C9,C62,C64,C90 = 1μF 10% 25 V X7R
 C5,C13,C18 = 15nF 5% 50V NPO, SMD 1206
 C6, C7, C14, C15, C16, C56 = 47nF 5% 50V NPO, SMD 1206
 C8, C73, C89 = 1nF 5% 50V NPO
 C17,C21,C23,C55,C60 = NC (not fitted)
 C19,C25 = 150pF 5% 50V NPO
 C22,C52,C58,C59 = 1.5nF 5% 50V NPO
 C24,C32,C69,C72,C74,C75,C81,C85 = 2.2μF 10% 16V X7R
 C30,C57 = 4.7nF 5% 50V NPO
 C38,C39,C83,C84 = 33pF 5% NPO
 C42,C70 = 33μF 6.3V, tantalum case A (Vishay Sprague 293D336X96R3A2TE3)
 C66,C71, C82 = 10nF 10% 50V X7R
 C67,C77, C79 = 4.7μF 10% 10V X5R
 C68,C76 = 470μF 6.3 V tantalum case D (Kemet T495D477K006ATE100)

Inductors

L1 = 20μH dual coupled (Bourns PM3602-20-RC)
 L2, L3 = ferrite bead, SMD 0805 (Murata BLM21PG221SN1D)

Semiconductors

(default: SMD)

D1,D2,D3,D4,D5 = BAV199 (SOT23)
 D6 = LED, green, 3mm, leaded, horizontal mounting (Dialight 551-0207F)
 D7 = MBR0520 (SOD123)
 D8, D9 = LED, red, SMD 0805 (e.g. Kingbright KP-2012SURC)
 D10 = BAT54A (SOT23)
 U1 = OPA725AIDBV (SOT23-5) (TI)
 U2 = MAX7404CSA (SOIC-8) (Maxim)
 U3 = 74HCT4052D (SOIC-16)
 U4, U20 = TLC2274AID (SOIC-14) (TI)
 U5 = INA128U (SOIC-8) (TI)
 U6,U11,U14 = OPA365DBV (SOT23-5) (TI)
 U7 = PGA103U (SOIC-8) (TI)
 U8 = 74HCT4053D (SOIC-16)
 U9 = C8051F061-GQ (TQFP-64) (Silicon Laboratories)
 U10 = LM4040D25IDBZ (SOT23) (TI)
 U12 = DAC8811CDGK (MSOP-8) (TI)

To accommodate the display & keypad extension, the extruded aluminum lid will have to be machined and marked up (silk-screened or engraved) with the functions of the buttons (**Figure 5**). Then use four M3 screws (12 mm / 0.5", countersunk) with nuts and washers, and four 7 mm (0.3") spacers.

Note: for fixing the extension, in the absence of spacers, you can use other nuts (and washers): 4 for fixing the 4 screws onto the lid, 4 for setting the 7-mm (0.3") spacing, and lastly 4 more to secure the extension board.

Two final remarks about the jumpers: Normally, the firmware is updated by the main program on the computer. Straight afterwards, the circuit

ought to work, but if for some reason the firmware didn't respond (e.g. in the event of a bug in the new firmware that's just been loaded), it will then be necessary to intervene at the bootloader stage and tell it that we want to perform an *unconditional* update. Jumper J17, accessible by removing the back panel, is used to give this information.

The update procedure is described in the downloadable documentation [3].

To make sure the measuring probe is firmly connected to the measuring point pins J4, J5, J14, it's perhaps not a bad idea to bend them at an angle.

All the other jumpers are described in the online documentation.

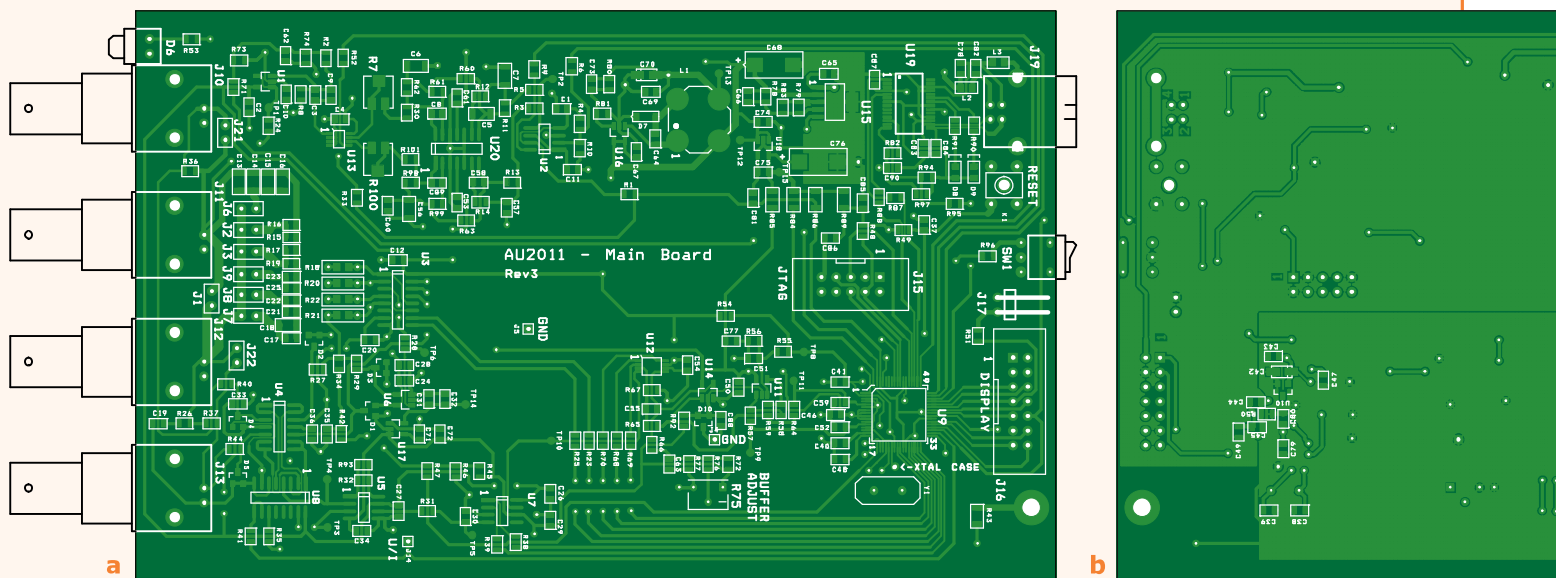


Figure 1. The LCR Meter's double-sided main board; component side (a). Don't forget the dozen or so components to be fitted **at the underside** of the board (b). The absence of silk screen print under the board is deliberate; it is not an oversight or an error.

U13 = SN74LVC2G53DCT (SM-8) (TI)
 U15 = REG102GA-A (SOT223-5) (TI)
 U16 = LT1611CS5 (SOT23-5) (Linear Technologies)
 U17 = TPS72325DBV (SOT23-5) (TI)
 U18 = TLV70030DDC (SOT23-5) (TI)
 U19 = FT232RL (SSOP-28) (FTDI)

Miscellaneous

Y1 = 24MHz quartz crystal, fundamental resonance (e.g. Euroquartz 24.000MHz HC49/4H/30/50/40+85/18pF/ATF)
 SW1 = rocker switch (RS Components 4US1R1020M6RNS, code 734-6934)
 J1, J2, J3, J6, J7, J8, J17, J21, J22 = 2-pin pinheader, 0.1" pitch (2.54mm)

J4, J5, J14 = 1-pin pinheader, bent (see text)
 J10, J11, J12, J13 = BNC socket, horizontal mounting, isolated (e.g. TE Connectivity 1-1337543-0)
 J15 = 10-pin pinheader HE10 (e.g. Multicomp MC9A12-1034)
 J16 = 14-pin pinheader HE10 (e.g. Multicomp MC9A12-1434)
 J17 = 2-pin pinheader, 0.1" (2.54 mm), bent (see text)
 J19 = USB connector, type B, horizontal (e.g. TE Connectivity 292304-1)
 K1 = pushbutton (Omron B3F-10xx series)
 PCB, bare, # 110758-1
 alternatively, preassembled module # 110758-91
 Case (Hammond 1455L1601 + machining + lettering)
 Screws, nuts, spacers, misc. hardware

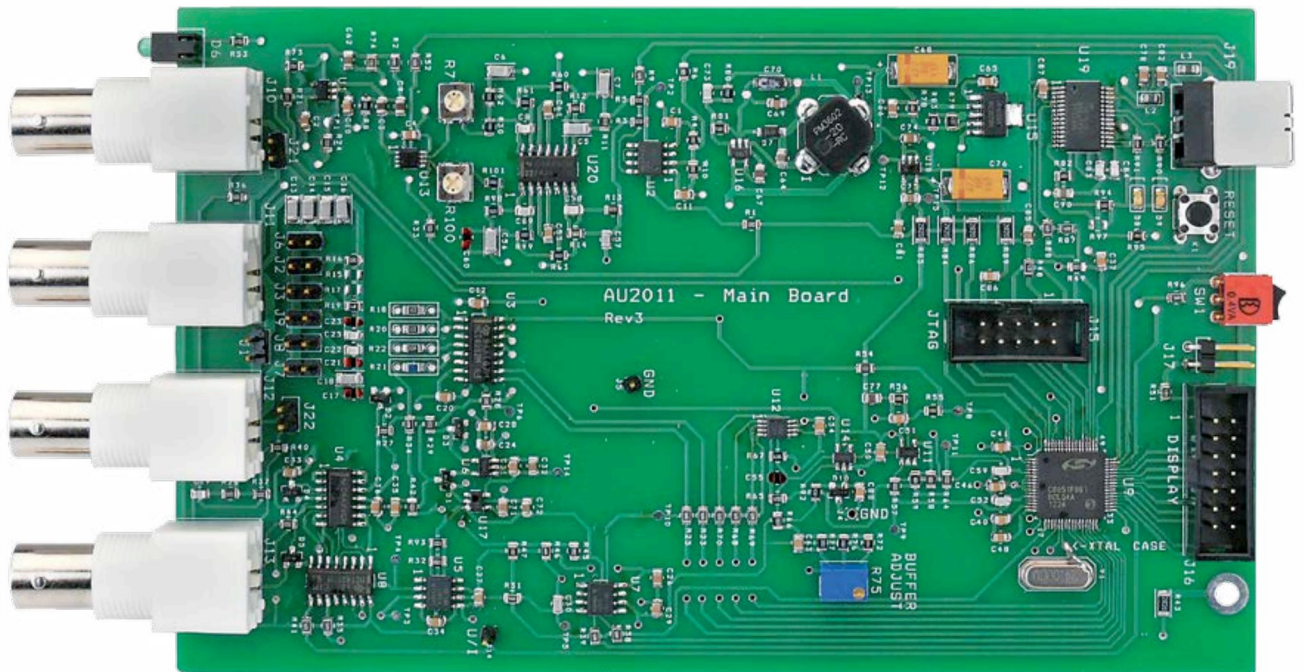


Figure 2.
Photo of the prototype.

Semi-automatic adjustment

Contrary to what we might be afraid of in the light of the device's remarkable accuracy, the adjustment procedure does not require any special skill or equipment. Apart from a multimeter

for checking the supply voltage, no measuring device is required! The software takes care of the semi-automatic calibration. The user has only to carry out a few adjustments themselves, for which they are guided step by step by the program that

COMPONENT LIST

Display / Keyboard module

Resistors

(SMD 0805 1%)

R1,R2,R3,R4,R10,R11 = 4.7k Ω

R5,R6,R7,R9 = 56 Ω

R8 = 1k Ω

Capacitors

(SMD 0805)

C1,C2,C3,C4,C5,C6,C7,C8,C9,C10 = 1 μ F 10% 25V X7R

Semiconductors

D1,D2,D3,D4 = BAT54A (SOT23)

D5 = LED, green, 3mm, leaded, (e.g. Kingbright L-424GDT)

Q1,Q2 = N-channel MOSFET (SOT23) (e.g. Fairchild FDU303N)

Miscellaneous

U1 = 64128M-FC-BW-3 graphic display (Displaytech)

J1 = 14-way HE10 transition header, **max. height 5.4mm** (Harting 09 18 114 9622)

K1,K2,K3,K4,K5 = pushbutton (Multicomp TS0B22)

14-way flatcable connector (e.g. Multicomp MC6FD014-30P1)

14-way flatcable, length (4 in. / 10cm) (3M 3365-14)

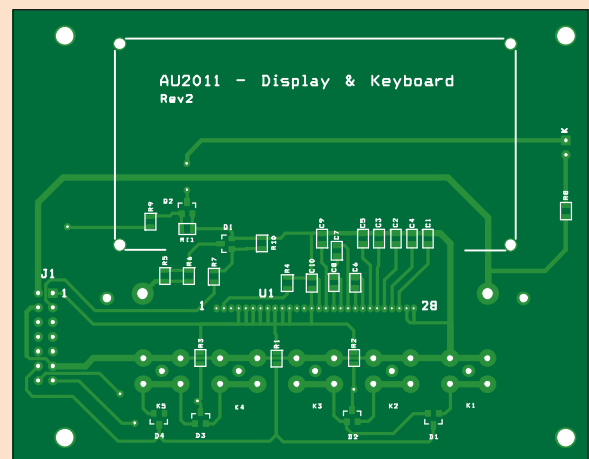


Figure 3. The design for the double-sided display extension PCB (optional).

PCB, bare, Elektor #110758-2

Alternatively, preassembled module, Elektor # 110758-92

performs all the measurements. The procedure is carefully described in a downloadable document [3], including fitting the NC components when necessary and the jumper configuration. All it takes is a little screwdriver, a little bit of judgment, and patience, as you must read the documentation right through and follow all the instructions to the letter, without skipping over any of them.

In the first article, we saw the importance of the measuring cables and their effect on the accuracy. So depending on what you want to use the LCR meter for, you may use Kelvin clips (**Figure 7a**) and/or a measuring box with four BNC connectors like the type TH26001A, "4 terminal test fixture" from TONGHUI (**Figure 7b**). They're easy enough to find via an Internet search (keywords LCR test clip, Kelvin clip, TH26001A).

Once you have the board you've built, you can power it using a USB supply and check the supply voltages. Since the microcontroller is blank, all its ports are at high impedance and it won't do anything! So there are no signals to measure. You can connect the board briefly to a PC and should see diodes D8 and D9 light: that's the FT232R (U19) communicating with the PC. Once the bootloader has been loaded into the microcontroller by the FlashUtil program using the DEBUG ADAPTER USB module (**Figure 8**) connected to J15, the microcontroller is operational. This operation and all those that follow are described in

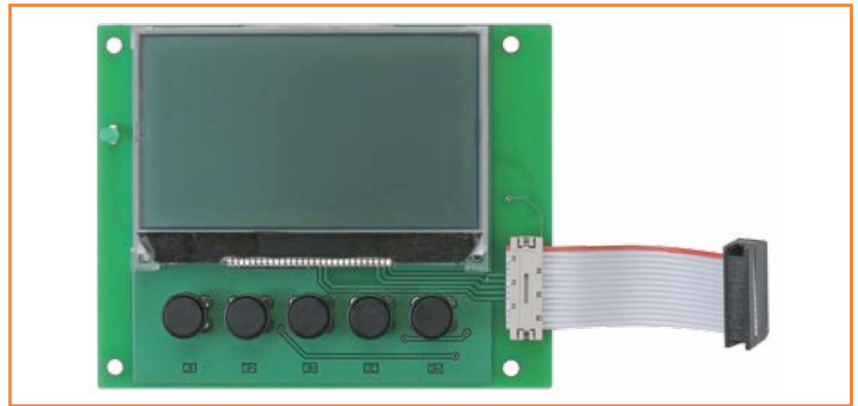


Figure 4a. Pay attention to the exact type of the HE10 connector for the display.

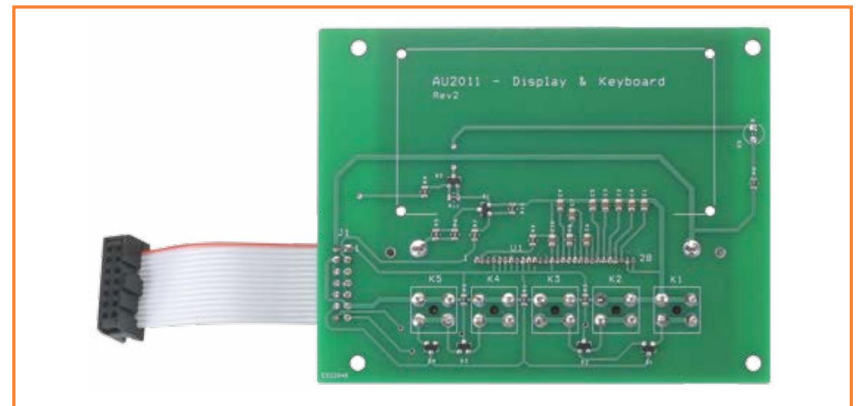


Figure 4b. The extension seen from below.

Figure 5. Dimensioned mechanical drawings.

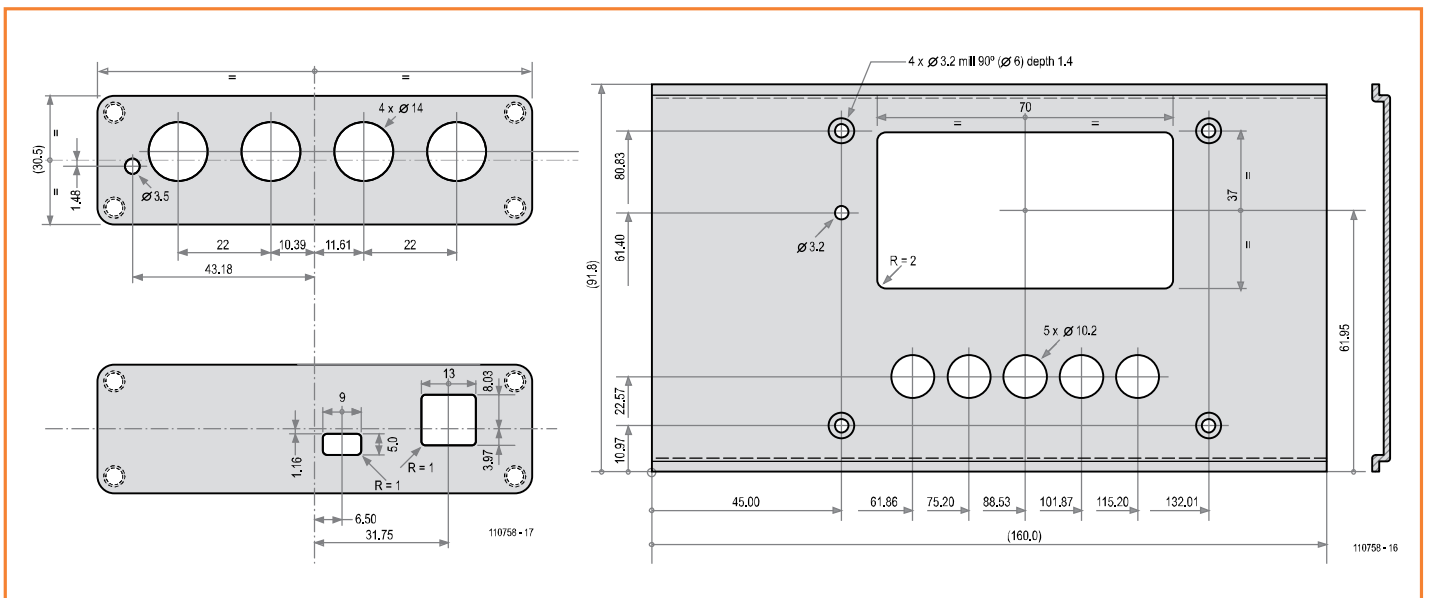
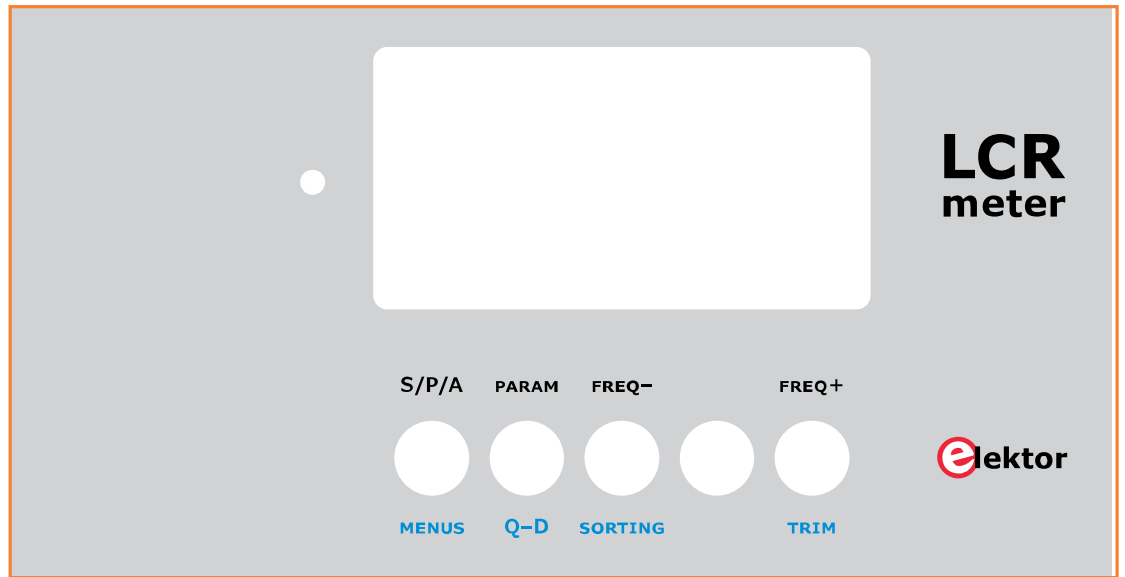


Figure 6.
Suggestion for front panel,
to be engraved or silk-
screened.



detail in the online documentation.
The AU2011 program installed on the PC must then establish the software link by opening the port; is it working? Then you're on the right track! All that remains is to load the *firmware* from

the PC via the AU2011 program's Tools/Program update menu.
After that, when you re-boot the LCR meter, you should see the (semi-)automatic offset adjustments taking place normally as described in the

Figure 7a.
This set of Kelvin clips with
four BNC leads can be found
for around \$10.





Figure 7b.
This test fixture fitted directly to the LCR meter case can be found for around \$50.

Set-up manual [3]. If yes, there's a 95% chance that everything's OK. Then you will run the measurements by clicking on the Start menu on the PC. If the parameters of an open circuit before TRIM are displayed (**Figure 9**), then you're almost 100% sure of success!

(130093)

Links & References

- [1] 500 ppm LCR meter Part 1
www.elektor-magazine.com/110758
- [2] 500 ppm LCR meter Part 2
www.elektor-magazine.com/130022
- [3] 500 ppm LCR meter Part 3
www.elektor-magazine.com/130093

Online documentation

- software (bootloader, firmware, and main program)
- designs for PCBs 1 & 2
- component overlay, top 1 & 2
- component overlay, underside 1 & 2
- complete BOM
- XL version of circuit diagram
- mechanical drawing (standalone version with extension)
- front panel (standalone version only)
- Set-up and Operating Instructions documents



Figure 8.
To load the program into a blank microcontroller, you'll need this accessory.

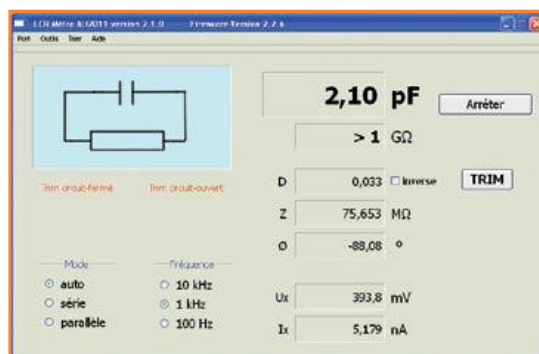
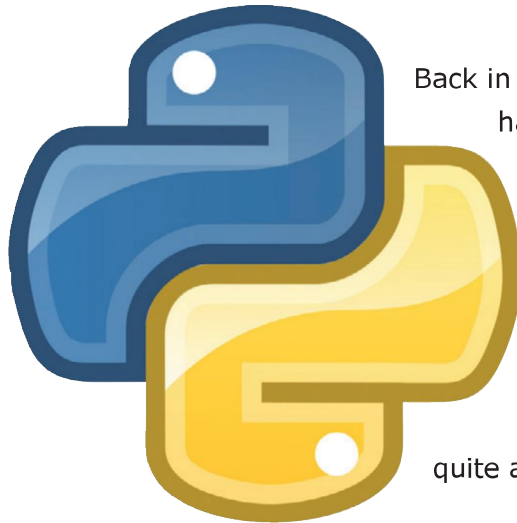


Figure 9.
If this window with the parameters of an open circuit before TRIM appears, it confirms that the circuit is working correctly.

From BASIC to Python (1)

A field report

By
Jean-Claude Feltes
(Luxembourg)



Back in the 80s the author's first PC, by chance, had QuickBASIC already installed. That sparked his interest in programming; he later went on to Visual BASIC. With his current Linux platform he went looking for a programming language to perform the same role as BASIC. That turned out to be a real step up; Python is a neat and concise language but also in many respects quite a different animal...

Regular readers may be thinking it a bit unusual to find a programming language review in Elektor magazine. Well the reason of course is that Python is a well-structured high-level language particularly associated with small computer systems such as the popular Raspberry Pi. The author was first introduced to Python by a colleague and was immediately intrigued by the language concepts and its contrast to Visual Basic with which he was more accustomed. A study of some source code examples impressed him with their clarity and conciseness. Gone were the brackets and semicolons which always seemed to give headaches in Pascal, C and Java. Further study uncovered many other differences.

BASIC versus Python

The first important thing to note is that Python is purely an interpreted language. This feature has both advantages and disadvantages. A Python program will generally run more slowly than an equivalent compiled program. The efficiency of the Python library routines however ensures that it is not too much of a disadvantage. The Interpreter along with any used library modules must also reside in the target system memory. An advantage of this approach is that the program can be easily modified when, for example, you need to switch the communication port to a

different interface. Quick and dirty hacks can be performed by doing away with any user interface code and simply changing the values of variable directly in the code. Interpreted code also allows you define functions at run time and can be useful when implementing such things as function plotters.

Python is very strongly object-orientated. For just a small program this feature is not so important. In the text and with example programs we will make good use of these features. If you are new to object oriented code then some of the concepts may seem difficult at first.

One thing that takes a little getting used to is the minimalist layout of the program: Code blocks are not delimited by parentheses or 'begin' and 'end' but instead just by indenting the source code. It can be seen from the short example shown in the box comparing C, BASIC and Python that Python produces very readable and concise code. Another difference is that unlike BASIC, Python code is case sensitive. An attractive feature for engineers is that Python can handle complex variables. Interpreters are available which run in Windows, Linux and OS X environments.

Installation

Once the Python Interpreter for your chosen OS has been installed it will also be necessary

C, BASIC and Python

C	Quick/Visual Basic	Python
<pre>#include <math.h> #include <stdio.h> int main(int argc, char *argv[]) { printf ("Hello World\n"); int i; int x; for (i=0; i<11;i++) { if(i%2==0) { x = pow(i,2); printf("%d ^2 = %d \n",i,x);} else { x = pow(i,3); printf("%d ^3 = %d \n",i,x);} } return 0; }</pre>	<pre>Print "Hello world!" For x = 1 To 10 If x Mod 2 = 0 Then Print x; "^2 = "; x^2 Else Print x; "^3 ="; x^3 End If Next x</pre> <p>For VB replace Print with Debug.Print, and the code must be in a Sub, for example Form_Load().</p>	<pre>print "Hello world!" for x in range(0,11): if x % 2 == 0: print x, "^2 = ", x**2 else: print x, "^3 = ", x**3</pre>

to download some useful additional modules. The main question concerning the Interpreter is whether version 2.x or 3 should be installed? Version 3 does unfortunately have some important libraries missing. Some backport improvements to version 2.7 have also been implemented which make this version more useful. **Table 1** shows some modules for software development along with their download URLs.

The set up in Windows is simplified by an installer program. For other systems:

- Download (Archive) into a temporary directory
- Enter the command line instruction:
python setup.py install

The Python script will now create and install all

Table 1: Software modules and download links

Python 2.7 The Interpreter	www.python.org/download/ <i>Python is already installed in Linux systems.</i>
Numpy Scientific and Mathematic functions etc.	http://pypi.python.org/pypi/numpy
Matplotlib Graphs	http://sourceforge.net/projects/matplotlib/files/matplotlib/matplotlib-1.1.0/
PySerial Gives access to the serial interface	http://sourceforge.net/projects/pyserial/files/
PyParallel Gives access to the parallel interface	http://sourceforge.net/projects/pyserial/files/
PyUSB USB Module	http://sourceforge.net/projects/pyusb/
WxPython GUI toolkit	www.wxpython.org/download.php
Geany Editor with syntax highlighting	www.geany.org/

Projects

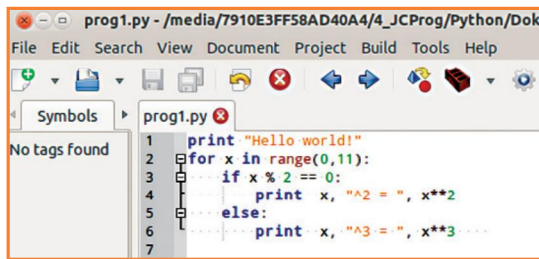


Figure 1.
Geany the free editor.

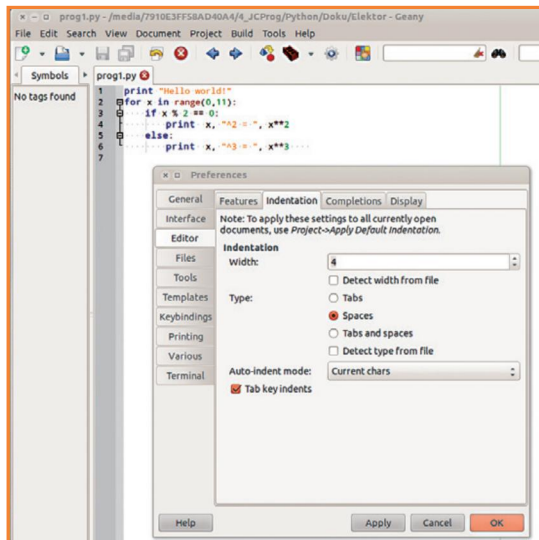


Figure 2.
The correct 'Pythonic'
format using a four space
indent.

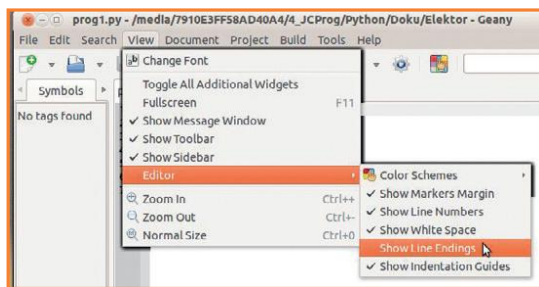


Figure 3.
Some more settings in
Geany.

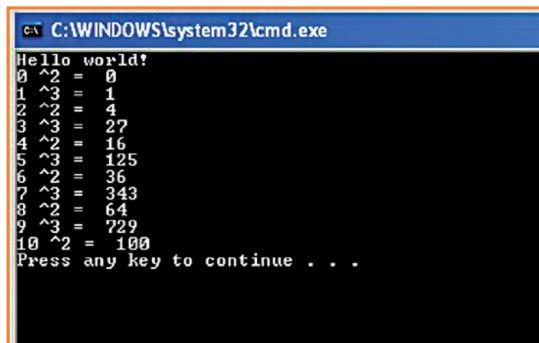


Figure 4.
The test program results in
Windows.

the necessary files into a subdirectory where they can be used by the Interpreter. This directory name and path depends on the type of operating system in use ('/usr/lib/python2.7' for Ubuntu or '\Python27\Lib' for Windows XP).

IDE or No-IDE?

At the start I missed the IDE that comes with VB to assist program development. Eventually however it did not prove to be too much of a hindrance. For simple programs all you really need is a good Editor, one favorite of mine is the freeware 'Geany' which runs under Windows and Linux. It performs automatic syntax highlighting for files that have a '.py' extension. It also supports code folding and allows a script to be started directly from the editor.

Users of OS X need look no further than the Apple Xcode programming environment which supports Python along with many other languages. The 1.65 GB package can be downloaded for free from the Apple App store. A double click on .py extension files will load them into the Xcode Editor.

A first program

The programs can be written using any text editor, **Figure 1** shows how it looks in Geany. In Python the source code layout is very important: you can either use tabs or spaces to generate the indents but do not mix them. The correct Python layout is four spaces per indent. **Figure 2** shows how this is set up in the Geany editor. For maximum convenience it is advisable to use the 'show markers' margin, line number display and indent support features as shown in **Figure 3**.

When the simple test program in **Figure 1** is run under Windows the result is shown in **Figure 4**. The program can also be run using the command line statement 'python test.py' if you do not wish to or can't start it in the Editor program. With GUI scripts a double click on the file with the .py extension will work but with text programs like test.py the window closes so quickly after the output that it's difficult to read.

When the program is used under Linux it is good practice to insert the following two lines before the start of the source code:

```
#!/usr/bin/env python  
  
# -*- coding: utf-8 -*-
```

The first line informs the operating system which

Interpreter is used and the second line indicates the character set. Programmers raised on BASIC should study the box **Python features** which compares how the same routine can be written in three programming languages.

The Python shell

Python can be started using a command line (use the DOS box in Windows or Terminal in Linux or OS X), you should then see the Python triple chevron prompt '>>>'. Now you can enter commands interactively and explore some of the language features:

```
>>> s = 'hello'
>>> s.upper()
'HELLO'
```

Library modules can be imported into the shell:

```
import time
```

Help for any imported module can be accessed by using 'help(module)' at the prompt. Using 'dir(module)' returns a list of all names (variables names and functions etc.) defined in the module.

Pitfalls for Python newbies

Anyone accustomed to other programming languages will most likely make the same simple errors when starting out with Python. One of the most common is to incorrectly indent the code. Python is particularly fussy, if the interpreter complains of an 'unexpected indent' then you can be sure you've got an incorrect number of spaces or tabs or you have mixed spaces and tabs. Each indent in Python should be four spaces. Pay close attention to how integers, floating point values and variables are written: $3/5 = 0$ and only $3.0/5.0 = 0.6$!

Python also states that for every subprogram and each Method requires a bracket at the end. Closing the serial interface using just 's.close' generates an error while 's.close()' is accepted.

External hardware

For engineers and developers one of the most crucial aspects of any programming language is how easily it works with the hardware. Python includes ready written modules to control the PC interfaces: *pyUSB*, *pySerial*, *pyParallel* and *pyI2C*. As a simple example we will use the serial interface: For communication the port is assigned an instance of the serial objects in the 'serial.

Listing 1: ScanSerial.py

```
import serial
def scan_serial():
    """ Scans for available serial ports """
    portnames = []
    # Windows
    for i in range(256):
        try:
            name = "COM"+str(i)
            s = serial.Serial(name)
            s.close()
            portnames.append(name)
        except:
            pass
    # Linux
    for i in range(256):
        try:
            name = "/dev/ttyS"+str(i)
            s = serial.Serial(name)
            s.close()
            portnames.append(name)
        except:
            pass
    # Linux USB
    for i in range(256):
        try:
            name = "/dev/ttyUSB"+str(i)
            s = serial.Serial(name)
            s.close()
            portnames.append(name)
        except:
            pass
    return portnames
#-----
# main
portnames = scan_serial()
for p in portnames:
    print p
```

Listing 2: ReadSerial.py

```
"""Read and print serial data from COM1 (9600baud)"""
import serial
# init serial port COM1 / ttyS0
sCOM1 =serial.Serial(0)
sCOM1.setBaudrate(9600)
if sCOM1.isOpen()==False:
    sCOM1.open()
# read lines of data until user presses <Ctrl-C>
while(1):
    line = sCOM1.readline()
    print line
sCOM1.close()
```

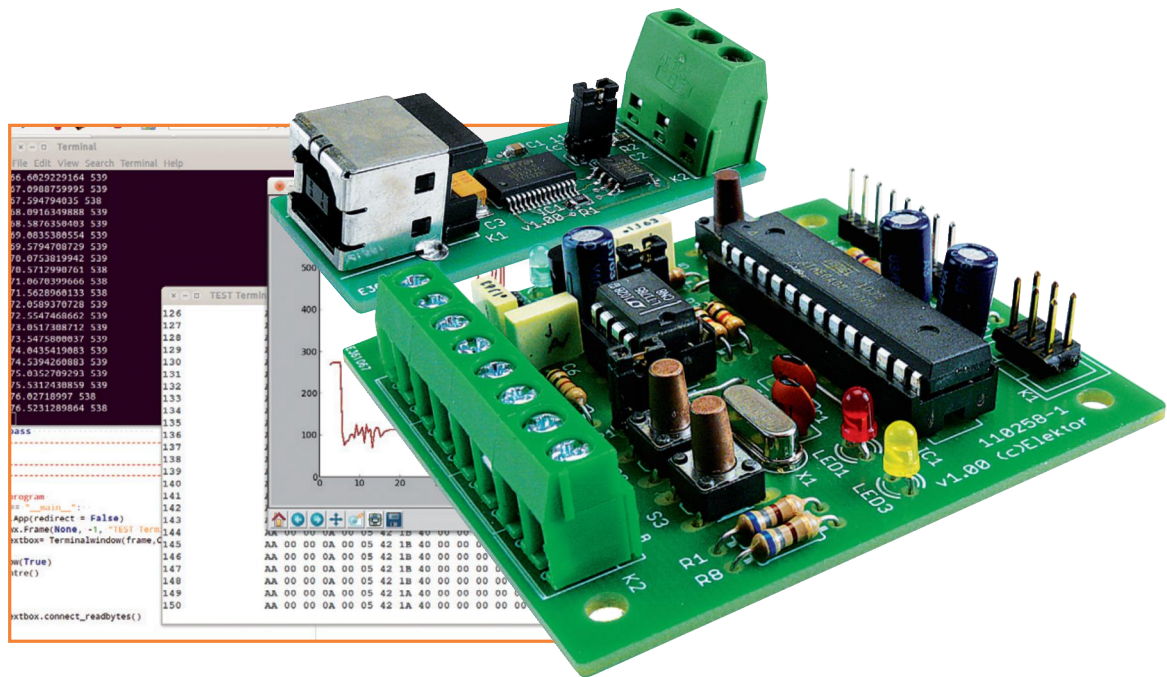


Figure 5. Python works well with the ElektorBus communication transferring measurement data.

py' module. Ports can be addressed by their number or name ('COM1' in Windows or '/dev/ttyS0' in Linux).

The simplest method to search for available ports is to try to instantiate a port (Listing 1). When it is successful it indicates that the port exists. Virtual USB ports will also be detected.

Listing 2 illustrates how data can be received through a serial port. The data may have originated from an external microcontroller. After the assignment of the object sCOM1 to Port COM1

the baud rate is specified (default values for parity and number of stop bits are used) and the port is opened. During software experimentation it often occurs that a program will leave a port open. Next time the program starts it cannot open the port. It is better only to open a port when it is not already open.

The program now executes an endless loop until it is terminated using Ctrl-C. A more elegant solution would be to poll a key and terminate the program when a press is detected. This is possible but not so simple because the operating system decides. The Function 'raw_input()' cannot be used in this instance because it will wait until the enter key is pressed. If you plan to add a GUI to your program using wxPython for example, it provides simple methods to detect and react to key presses and other events.

With the addition of a little more code (Listing 3) it is possible to make a data logger. Subroutines have been used here to improve code clarity. These are first defined using 'def...()' and the main program comes right at the end. Next in 'show_log()' it checks using (try / except) to find a LOG file, if it exists then the file is read and output. The 'file.read' reads all the text in the file. The serial port is opened and each line of received data is displayed and written to the file.

In the pipeline

Now that you've got a taste for Python you should be able to install the interpreter yourself along with some libraries and begin to write a

Python features

- Variables are declared implicitly by their assignment: `x = 5.0`
- The For-Next loop structure found in many other languages does not exist in Python. An iteration can be performed on the components of an object, like the characters in a string or lines in a file. In place of the classic 'For... Next' the 'for I in range' condition can be used:

```
for i in range (0,5):  
    print i
```

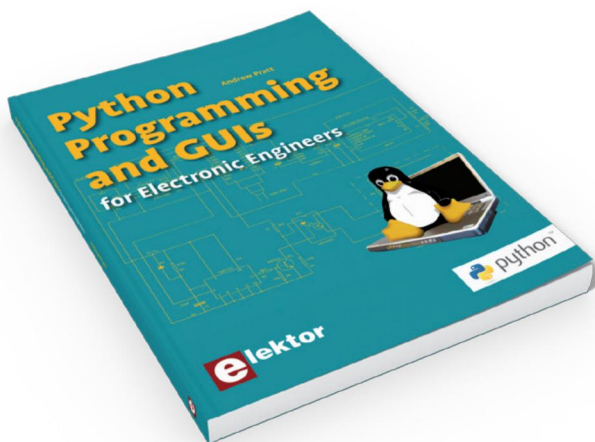
The iteration is performed on all the components of the object 'range (0,5)' i.e. on the list [0,1,2,3,4]. NB: The end value (5 in this example) isn't part of the list!

- Arrays (lists) may contain different objects:
`x = [0, 3.14, 'Ham' , 'Eggs']`
- Functions can return multiple values as a 'tuple':
`(x, y, z) = myfunction(v)`
- There are no SUBs, only Functions, as in C. These are declared with 'def...()' and can, as needed, return values or objects.

few simple programs. All the example listings shown here can be downloaded for free from [1]. In the second part of this series we will explore a few engineering niceties such as graphs, Fourier synthesis with a nice user interface. In the third part we switch to practical examples and use a small microcontroller board to send measurement data via the ElektorBus to a PC (**Figure 5**).
(110483)

Internet links & Literature

- [1] Listings etc.:
www.elektor.com/110483
- [2] The author's homepage:
<http://staff.ltam.lu/feljc/home.html>
- [3] Python documentation:
<https://pypi.python.org/pypi/RPi.GPIO>
- [4] Python Tutorials:
www.awaretek.com/tutorials.html
- [5] Introduction:
J.M. Hughes: Real World Instrumentation with Python



- [6] Andrew Pratt: Python Programming and GUIs for Electronic Engineers
www.elektor.com/products/books/programming/python-programming-and-guis-for-electronic.1320886.lynx

Listing 3: ReadSerial2.py

```

"""Read and print serial data from COM1 (9600baud)"""
import serial

def show_log():
    """ show result of last logging"""
    try:
        file = open("test.log", "r")
        text = file.read()
        file.close()
        print "CONTENTS OF LAST LOG FILE:"
        print text
        print "END OF LOG FILE"
    except:
        print "NO LOG FILE FOUND"
        pass

def open_port_log(port):
    """ open serial port and LOG file"""
    # init serial port COM1
    sCOM =serial.Serial(port-1)
    sCOM.setBaudrate(9600)
    if sCOM.isOpen()==False:
        sCOM.open()
    file = open("test.log", "w")
    return sCOM, file

def receive(sCOM, file ):
    """ read lines of data until user presses <Ctrl-C>"""
    while(1):
        line = sCOM.readline()
        print line
        file.write(line)

port=1
show_log()
ok = raw_input("NEW LOG (y/n) ?")
if ok== "y":
    s, f=open_port_log(port)
    receive(s, f)

# file and port closing done by interpreter at <Ctrl-C>

```

The Author

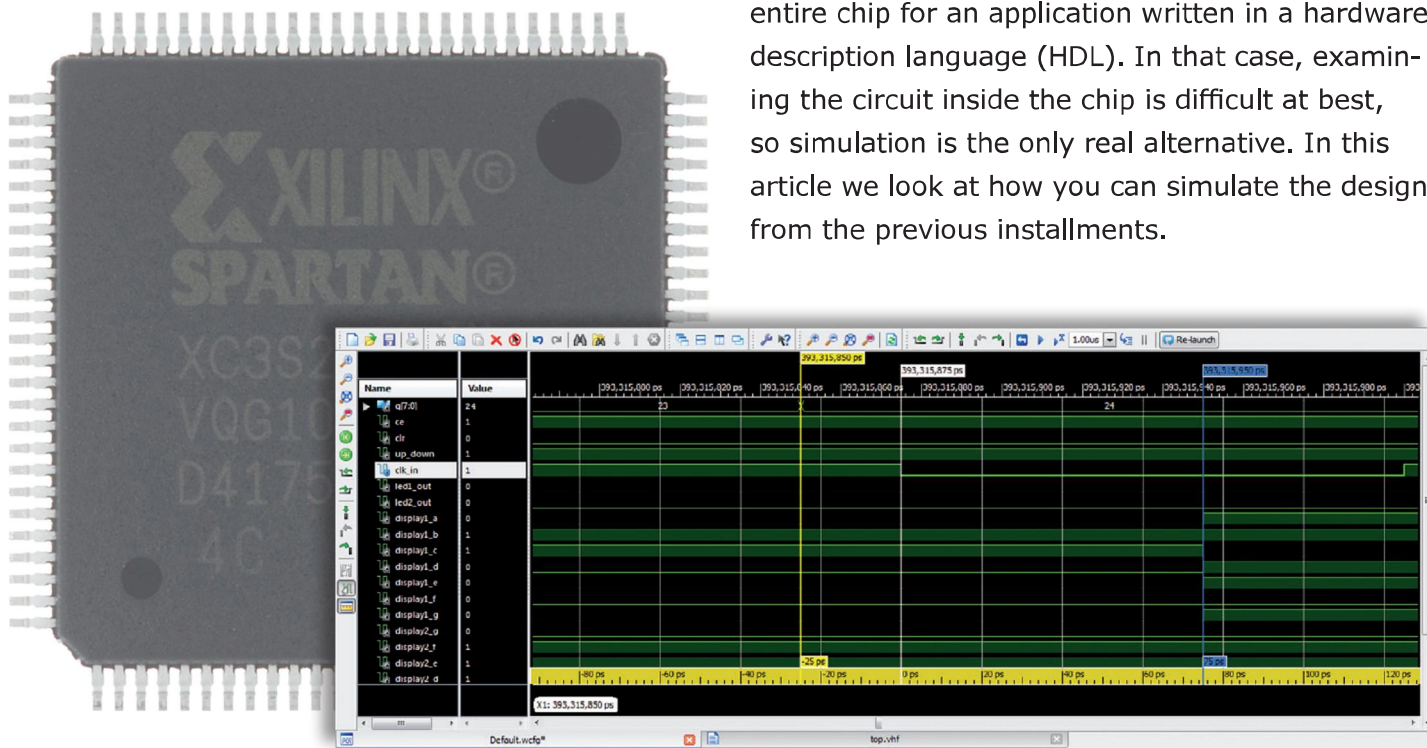
Jean-Claude Feltes lectures in electronics at the Lycée Technique des Arts et Métiers in Luxembourg. This college caters for both arts and technical students and provides professional qualifications for apprentices and technicians. He spends much of his free time pursuing his interests in electronics and programming [2].

Taming the Beast (4)

Make-belief with 250 Kgates

By **Clemens Valens**
(ElektorLabs)

Simulation is an important part of programming FPGA applications. If your application involves only a handful of gates, it isn't especially difficult to come up with a properly working design. However, things are different when you use virtually the entire chip for an application written in a hardware description language (HDL). In that case, examining the circuit inside the chip is difficult at best, so simulation is the only real alternative. In this article we look at how you can simulate the design from the previous installments.



The problems that arise when you are implementing a design in an FPGA are not all due to dumb design errors. The architecture of the chip and the shortcomings of the development tools can also play a role. Especially in applications that work with high clock rates, these factors become significant and sometimes unavoidable. Glitches can occur when the distances and propagation times in the chip become relatively large. For this reason, FPGA programmers run a lot of simulations to check whether the design does what it is supposed to do. They also look for signs that it does things it isn't supposed to do. Simulations are used at various stages in the development process. This starts with a functional simulation, which serves to indicate whether the basic design is sound. However, at this point there's still a lot to be done before the design is

finished. Each step of the process creates new opportunities for errors, so a simulation after each step is necessary to keep things under control. If the design passes the final simulation after the last implementation step, there's a reasonably good chance that it will work properly. However, there is no guarantee of this, since a simulation is just a simulation and real hardware can always throw an unexpected monkey wrench in the works.

Starting the simulator

To see what it's like, let's simulate the design of the two-digit up/down counter from the previous installments. Start by making a copy of the project from last time, which I named *part3* (see installments 3 [3]). I decided to name this copy of the project *part4* (can you guess why?). It's

a good idea to open the new project in ISE now, before you read any further.

You can open the ISE simulator at the top of the *Design* tab, where you see *View*. In the previous installments you were always working in the Implementation view, but now you need the Simulation view. After selecting the function by clicking *Simulation*, you will see that the content of the *Processes* pane has changed. If you look closely, you will also see that a new panel named *Behavioral* now appears above the *Hierarchy* pane. Since you're starting with an uncompiled virgin project, there is only one entry in the *Processes* pane: *Design Utilities*.

Select *top* in the hierarchy to display the *ISim Simulator* process in the *Processes* pane. Expand this entry by clicking the plus sign. Now select the process *Simulate Behavioral Model* and click the button all the way to the right, just below the green arrow. Alternatively, you can select *Run* in the pop-up menu that appears when you right-click the relevant entry, or you can double-click the entry with the left mouse button. ISE will now get busy, and after a little while (assuming all goes well) you will see a message in the *Console* pane telling you that the process completed successfully. You should also see that an ISim window has been opened next to the ISE window, with your signals displayed there on a sort of logic analyzer screen (**Figure 1**). In the *Console* pane at the bottom of the ISim window you can see that you are working with the

Lite version because you haven't acquired a full license yet. You can also see that the resolution of the simulation here is 1 picosecond (1 ps). At the top of the window you can see that the duration of the entire simulation is 1 μ s.

Now let's try a few things.

Manual simulation

Click the *Restart* button (the blue button with a white right-angle arrow facing left), or restart via the *Simulation* menu. Then right-click the *ce* signal in the signals list (there are two of them, but it doesn't matter which one you choose). Alternatively, you can click the signal on the graphic display. Make sure that only one signal at a time is selected. In the pop-up menu that appears, select *Force Constant*, since this signal is more or less constant – especially compared with a clock signal. In response, ISim opens the *Force Selected Signal* dialog, where you can set a number of parameters. Enter "1" in the field *Force to Value* and "1ms" in the field *Cancel after Time Offset*. The remaining fields can be left at their default values. Click *OK*. Now the signal *ce* will be high when the simulation starts, and it will remain high for 1 millisecond. Repeat this procedure for the signal *up_down* and the signal *clr*, but with the signal value forced to "0".

Next, use a similar procedure for the signal *clk_in*, but in this case select *Force Clock* instead of *Force Constant*. The *Define Clock* dialog that appears now is very similar to the *Force Selected*

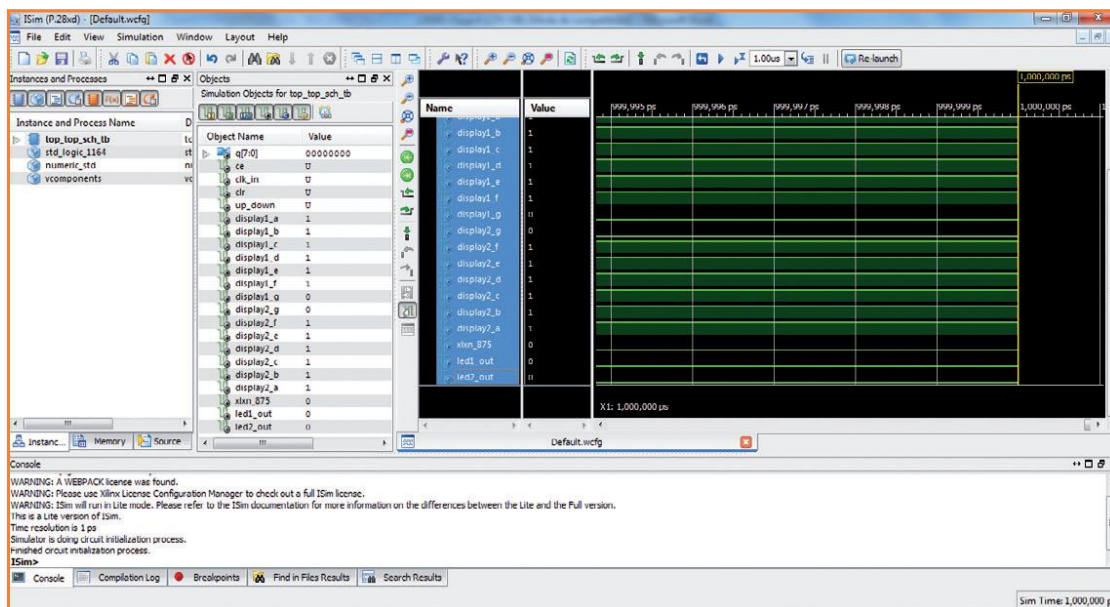


Figure 1. ISim in full glory. There's nothing in particular to see in the simulation pane here because we haven't defined any stimuli yet.

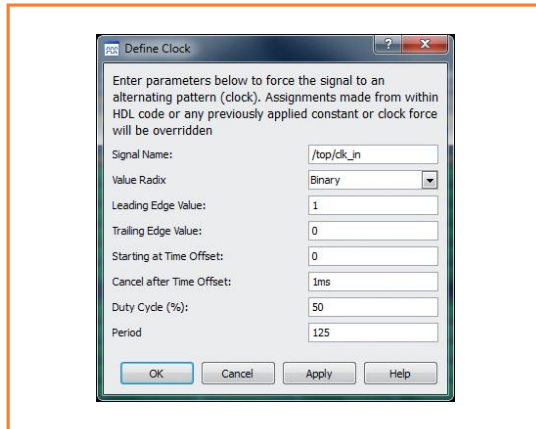


Figure 2. The *Define Clock* pane. Time intervals without specific units are picoseconds.

Signal dialog, but not exactly the same. Enter the values shown in **Figure 2** and click *OK*. These values cause the clock signal to start with a rising edge at time 0 and remain active for 1 millisecond, with a duty cycle of 50% and a period of 125. But 125 what? The answer here is 125 ps, since the time resolution is 1 picosecond. If you do not specify a unit, ISim uses the default unit. The clock frequency of the FPGA on the board is 8 MHz, which corresponds to a period of 125 ns, so you should enter “125 ns” here instead of “125”.

Now that you have defined **all** the input signals, you can start the simulation. If you do this by clicking the *Run All* button (or via the *Simulation* menu), the simulation will run until there are no more events or until a breakpoint is encountered. Since you have not defined any breakpoints, the latter possibility is rather unlikely. This means that the simulation will stop only after the defined stimuli expire at the end of 1 millisecond, since a clock signal constantly generates a series of events. You can interrupt the simulation at any time by clicking the *Break* button (the one with the pause symbol). Note that after a forced stop, ISim opens a VHDL source file (this usually *top.vhd*, but it can also be a test bench) and pushes the Waveform view to the background, where it can be recognized by the *wcfg* tab. Click this tab to display the Waveform view again.

You can also run the simulation in step mode with the *Run* button (the one with the triangle and hourglass symbols), which is next to the *Run All* button. Before doing this, you must enter the simulation interval in the box next to the button. For example, you can enter “10ms” here for an interval of 10 ms. After this, the simulation will

run for 10 ms each time you click the *Run* button. When the simulation is stopped, either by itself or forced, you can view the signals (**Figure 3**). You can zoom in or out, and you can scroll through the signals. You can also change the sequence of the signals by dragging them up or down, in order to group signals together. If you don’t see anything changing, check the colour of the four input signals. If any of them is orange instead of green, it is not defined and you have made a mistake somewhere.

Once you find the right zoom factor, you can see the clock signal oscillating away or see the other signals changing levels. The current states of bus signals (multi-bit signals, such as `q[7:0]`) are shown in binary notation. This is the default setting; it can be changed by selecting *Radix* in the same pop-up menu as *Force Constant*. This is a convenient way to see whether a counter is actually counting and in which direction. The signal states at the yellow cursor position are shown to the left of the waveforms.

On the test bench

Simulating a circuit in the manner described above is entertaining, but it’s not especially convenient. If the circuit works okay, this method is reasonable if you just want to look at something, but if you are iteratively modifying a circuit design, you have to define the signals again each time using time-consuming dialogs, and that gets old pretty quickly. If you’re a whiz with a keyboard, you can enter commands in the Console (for an example, look at the contents of the Console after you close a *Force Constant* dialog), but first you have to learn all these commands (see the document *plugin_ism.pdf*, which is available on the Xilinx website). A better way – in fact, the way Xilinx recommends – is to use a test bench to automate the simulation.

In this context, a test bench (or a test fixture) is simply a file that you add to the ISE project when the Simulation view is active. You do this in the usual way via *New Source* (right-click in the *Hierarchy* pane). This brings up the *New Source Wizard*, where you can select *VHDL Test Bench*. Enter a name (such as *test_bench*), check that the option *Add to project* is ticked, and then click *Next*. Now you have to associate the test bench file with a source file. In this case you want to test the entire circuit, so you should associate it with

Test bench in Verilog

Just as there are various computer programming languages, there are also various FPGA programming languages. In the main article we used VHDL because ISE apparently has a preference for this, but we could have just as easily used Verilog. For the sake of completeness, here we show you how the test bench can be built in Verilog

The clock process is implemented in Verilog with an always command (note that unlike VHDL, Verilog is case sensitive), and it fits on a single line thanks to the invert operator "!". The frequency is set with the "#" symbol, which introduces a delay. In this case the delay is 62.5 ns, since the default time unit here is nanoseconds (although the resolution is still 1 picosecond).

```
always
    #62.5 CLK_IN = !CLK_IN;
```

Here the level of `clk_in` is inverted every 62.5 ns. This works if you assign an initial level to `clk_in`. You could also use the following code to generate the clock signal in the same way as in VHDL:

```
always
begin
    CLK_IN = 1;
    #62.5;
```

```
CLK_IN = 0;
    #62.5;
end
```

With this code segment, there is no need to initialize `clk_in`. The stimuli are initialized in the initial code block, which is executed only once at the start of the simulation. The syntax is similar to the VHDL syntax, and here again we start with a 100 ns wait time.

```
initial
begin
    #100;
    CLK_IN <= 0;
    CE <= 1;
    CLR <= 0;
    UP_DOWN <= 1;
end
```

Here you can also use "=" in place of "<=". Just to be on the safe side, I initialized `clk_in` here. It may not be necessary, but in any case it doesn't hurt.

The always and initial code blocks can always be placed together at the end of the Verilog test bench generated by ISE, in place of the "ifdef auto_init ... endif" code block.

All clear? Start simulating!

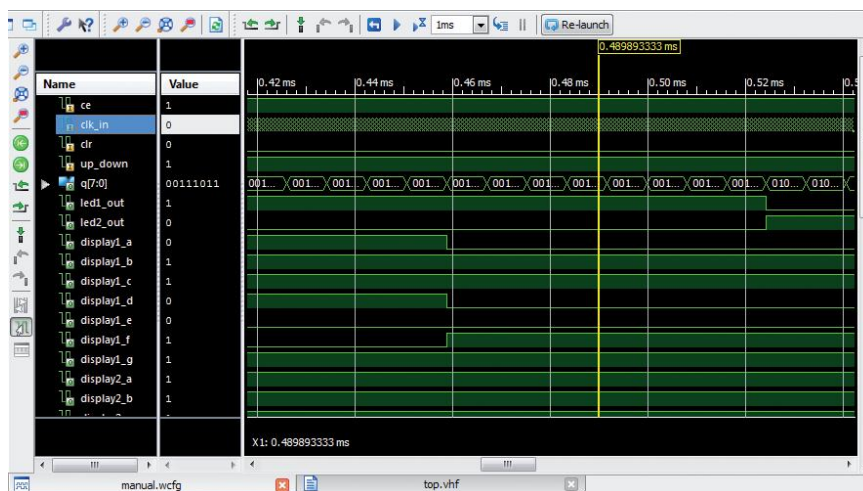


Figure 3. The waveform pane after a simulation. At the selected scale, the `clk_in` signal changes state so often that ISim shows it cross-hatched. You can right-click a bus signal to change the radix used to show the bus state value. In this figure it is shown in binary notation.

top. After you click *Next*, the usual summary is displayed. You can usually skip the details here, but in this case it's a good idea to check that the test file has been given the extension ".VHD". This means that it has been converted into a VHDL file. Click *Finish* to complete the procedure. Now you should verify that you have added the test bench in the Simulation view, not the Implementation view. If you accidentally get it wrong, you can correct your mistake by setting *View Association to Simulation* in the file properties (*Source Properties*).

Adding a test bench has more consequences than you might think at first glance. For instance, after a little while you will see a new entry in the *Hierarchy* pane between the chip and *top*, with the euphonious name "top_top_sch_tb". You will also see that *top* has changed to *UUT - top*, where *UUT* stands for *Unit Under Test*. The hierarchy thus clearly indicates what is tested and by whom.

In order to test the counter, you have to do the same thing in VHDL or Verilog that you previously did manually. You need a clock signal and several levels that are more or less constant. Let's start with the easy part: the constant levels.

At the end of the VHDL test bench listing, you will see the following code segment (note that the reserved VHDL commands are written here in capital letters, but this is not essential since VHDL is not case sensitive):

```
tb : PROCESS
BEGIN
    WAIT;
END PROCESS;
```

This is more or less the main routine of the test bench, or better put, the main process. This process, named *tb*, does nothing — it just waits. A process is a lot like a subroutine in a computer program, but unlike subroutines, processes are

Concurrent processing guarantees perfect multitasking

At this point the file `test_bench.vhd` has been opened in the Editor, and as you can see there are all sorts of things in it.

Just for fun, you can try adding another test bench, but this time in Verilog instead of VHDL so that you can see the differences between these two languages. The procedure for this is the same as previously described, except that you select *Verilog Test Fixture* as the source type. After you add this second test bench, you will see that an entirely new hierarchy has been added in the *Hierarchy* pane. This appears to suggest that you can use several test benches in the same project, but I couldn't manage to do this. ISE has a clear preference for the VHDL test bench. Note that the Verilog file has the extension ".V" and the icon is not quite the same as the icon for the VHDL file. Since the Verilog and VHDL files have different extensions, you can use the same file name for both test benches if you wish. Whether that's a good idea in practice is something you will have to decide for yourself.

Now for a bit of VHDL

The test bench only contains a skeleton at first, so you have to add the information necessary to activate the right signals at the right times.

executed concurrently (in parallel) instead of one after the other. You could regard this as a sort of perfect multitasking. A process is also executed over and over again unless it is interrupted, as you can see here with the infinite loop `WAIT`.

You need to add the signals `ce`, `clr` and `up_down` to this process and assign them levels. The resulting code looks like this:

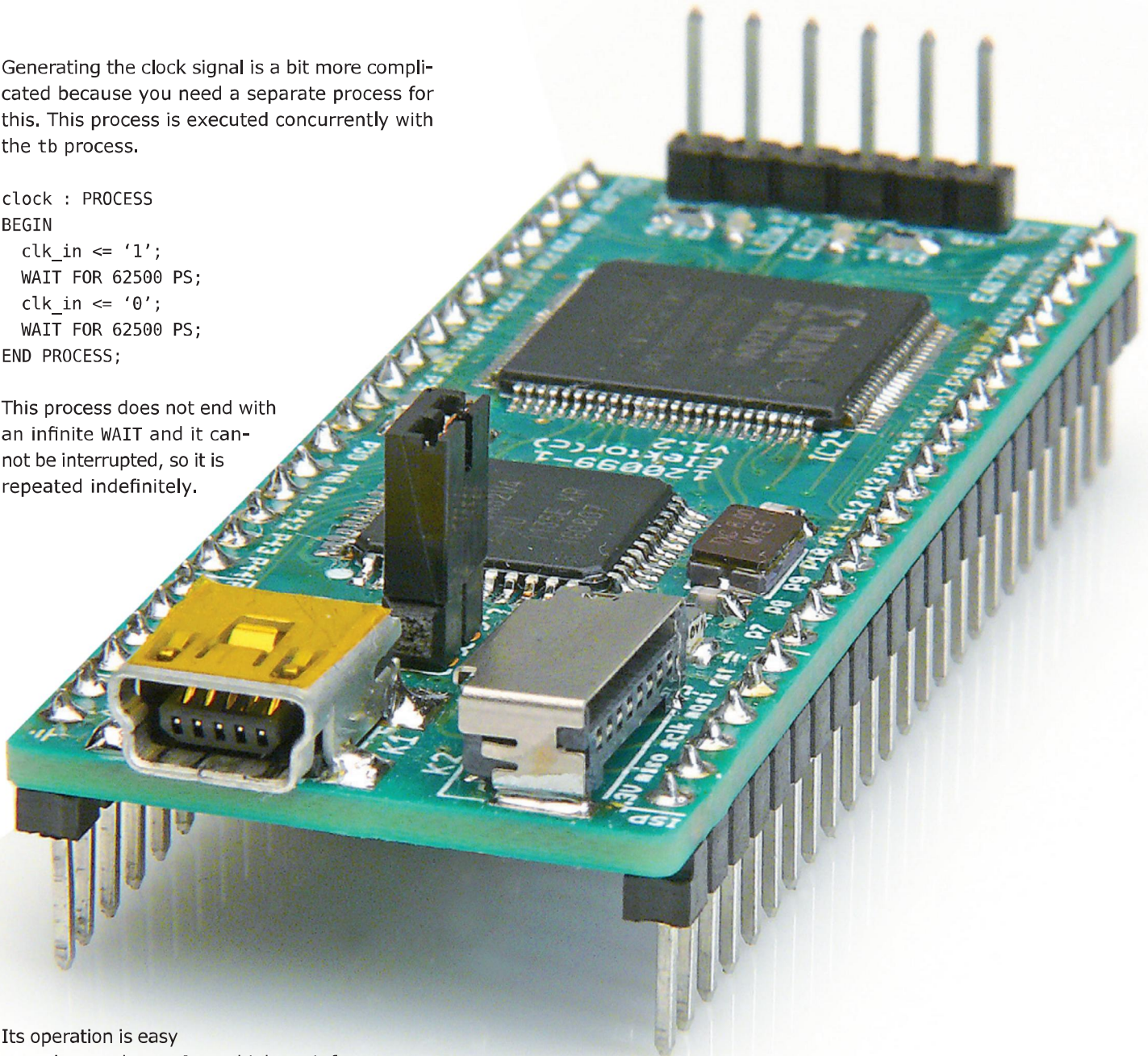
```
tb : PROCESS
BEGIN
    WAIT FOR 100 NS;
    ce <= '1';
    clr <= '0';
    up_down <= '1';
    WAIT;
END PROCESS;
```

Now the process sets the signals `ce` and `up_down` high and the signal `clr` low before it starts waiting. These instructions are executed concurrently instead of sequentially, so the order of the instructions here has no effect. The line `WAIT FOR 100 NS` causes the process to wait for 100 ns, giving it time for the reset signal. The *ISim Help* file recommends doing this, so we do so without asking any questions.

Generating the clock signal is a bit more complicated because you need a separate process for this. This process is executed concurrently with the tb process.

```
clock : PROCESS
BEGIN
  clk_in <= '1';
  WAIT FOR 62500 PS;
  clk_in <= '0';
  WAIT FOR 62500 PS;
END PROCESS;
```

This process does not end with an infinite WAIT and it cannot be interrupted, so it is repeated indefinitely.



Its operation is easy to understand: set `clk_in` high, wait for 62.5 ns, set `clk_in` low, wait again for 62.5 ns, and then repeat the process. The result is a signal with a duty cycle of 50% and a frequency of 8 MHz. Since the duty cycle is 50%, this can be done more elegantly by using a constant for the wait time. This constant should be added at the bottom of the SIGNAL list:

```
CONSTANT clock_half_period : TIME := 62500 PS;
```

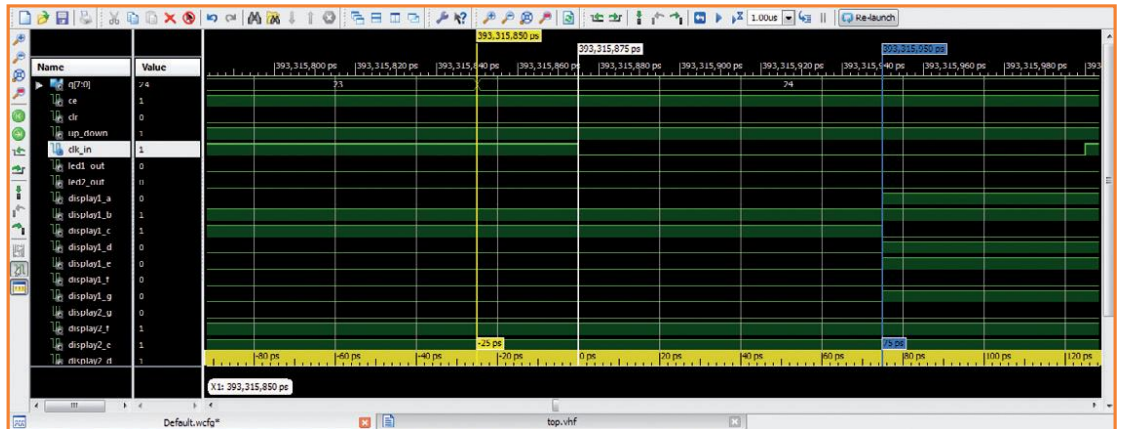
The constant `clock_half_period` is of type TIME and can be used with the WAIT command. The process code is now:

```
clock : PROCESS
BEGIN
  clk_in <= '1';
  WAIT FOR clock_half_period;
  clk_in <= '0';
  WAIT FOR clock_half_period;
END PROCESS;
```

Now the test bench is ready and you can start the simulation. Select the test bench in the hierarchy to display the ISim process. Expand this entry to display the following two options: *Behavioral Check Syntax* and *Simulate Behavioral Model*.

Figure 4.

You can use markers to measure time intervals. The clock period for this simulation was set 250 ps instead of 125 ns. When you select a marker, it becomes the zero reference point and ISim calculates the time intervals to the other markers. You can set the time unit by right-clicking the time axis.



Execute the first of these processes to check for typos. This must result in a green check mark before you can go any further.

Right-click *Simulate Behavioral Model* to open the simulation properties, and then tick the option *Run for Specified Time*. Now the simulation will run until you click *Break*. Otherwise the simulation will start immediately and run for the specified time. Close the properties window and launch the process *Simulate Behavioral Model*. This opens the ISim window, but nothing else happens. Click the *Run All* button, or else *Run (for the time specified on the toolbar)*, to start the simulation. Let the simulation run until the progress bar shows the desired duration. Then click *Break* to stop the simulation. Now you can view the signals in the same way as previously described with manual simulation. Bear in mind that the time axis may

have a very small time increment, which makes it look like all the signals are static. Use the *Zoom to Full View* button to display the full simulation, so that you can see the slowly varying signals properly. Then you can easily zoom in on the signals of interest (**Figure 4**).

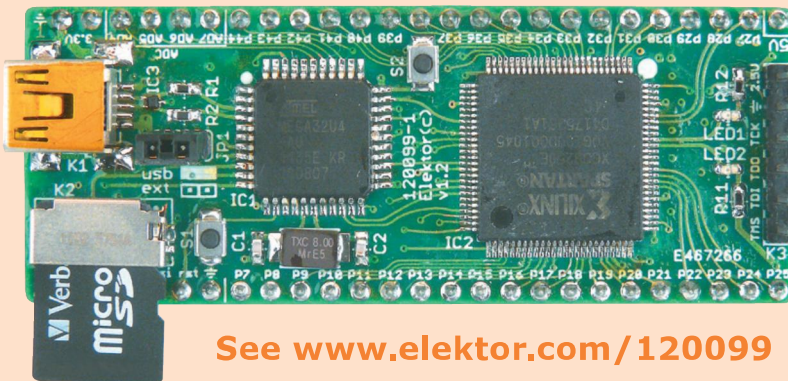
The signals in the waveform pane are shown in the same order as in the SIGNAL list. You can arrange this list to suit your taste, but make sure the commas are right. Remember that you have to close the ISim window each time you want to run a new or modified test bench, since ISE can't seem to do this for itself and generates an error message.

That's all folks!

This brings us to the end of our introduction to FPGA programming. You should now be able to set up your own simulations. There are a lot of things that we haven't discussed, and I recommend that you use the Internet and other resources to study this complex topic in more detail. Simulating a design is not especially difficult, but doing this properly requires sound knowledge of programmable logic, hardware definition languages and development tools. Go for it!

(130065-I)

The fully assembled and tested FPGA development board is available in the Elektor Shop for just \$66.89 plus shipping.



Internet Links

- [1] Part 1: www.elektor.com/120099
- [2] Part 2: www.elektor.com/120630
- [2] Part 3: www.elektor.com/120743
- [2] Part 4: www.elektor.com/130065



Professional PCB & PCBA Supplier

New Website Is Online!

Better Design
More Powerful
Easier to Use



Instant Quote & Pay

1 to 40 Layers
Prototype to Production
Amateur to Professional

order now

Prototype start at \$10/ea

2L 4"x4" each

Free Shipping!

<http://www.ezpcb.com>



AP CIRCUITS
PCB Fabrication Since 1984

As low as...

\$9.95
each!

Two Boards
Two Layers
Two Masks
One Legend

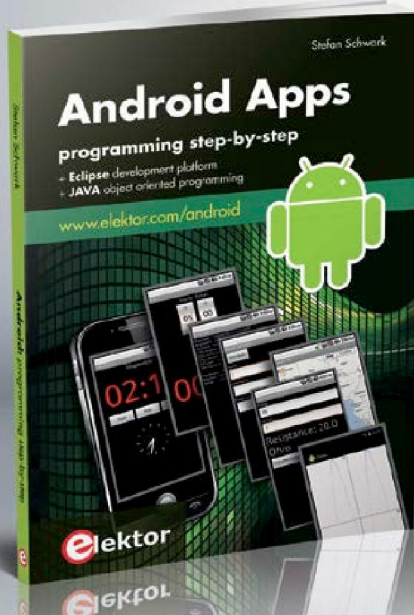
Unmasked boards ship next day!

www.apcircuits.com



Android Apps programming step-by-step

BEST-SELLER



When it comes to personalizing your smartphone you should not feel limited to off the shelf applications because creating your own apps and programming Android devices is easier than you think. This book is an introduction to programming apps for Android devices. The operation of the Android system is explained in a step by step way, aiming to show how personal applications can be programmed. A wide variety of applications is presented based on a solid number of hands-on examples, covering anything from simple math programs, reading sensors and GPS data, right up to programming for advanced Internet applications. Besides writing applications in the Java programming language, this book also explains how apps can be programmed using Javascript or PHP scripts.

ISBN 978-1-907920-15-8
244 pages • \$56.40

**10% OFF for
GREEN and
GOLD Members**

Further information and ordering at www.elektor.com/android

Elektor Linux Board Extension (2)

Implementing menus on a text display

By
Benedikt Sauter
(Germany) [1]

Even on a small display it is possible to show a surprisingly wide variety of parameters and settings by implementing a well-designed menu system. For ease of use the buttons should preferably be placed adjacent to the display, exactly as we have done on the Linux extension board. In this article we look at the required code.

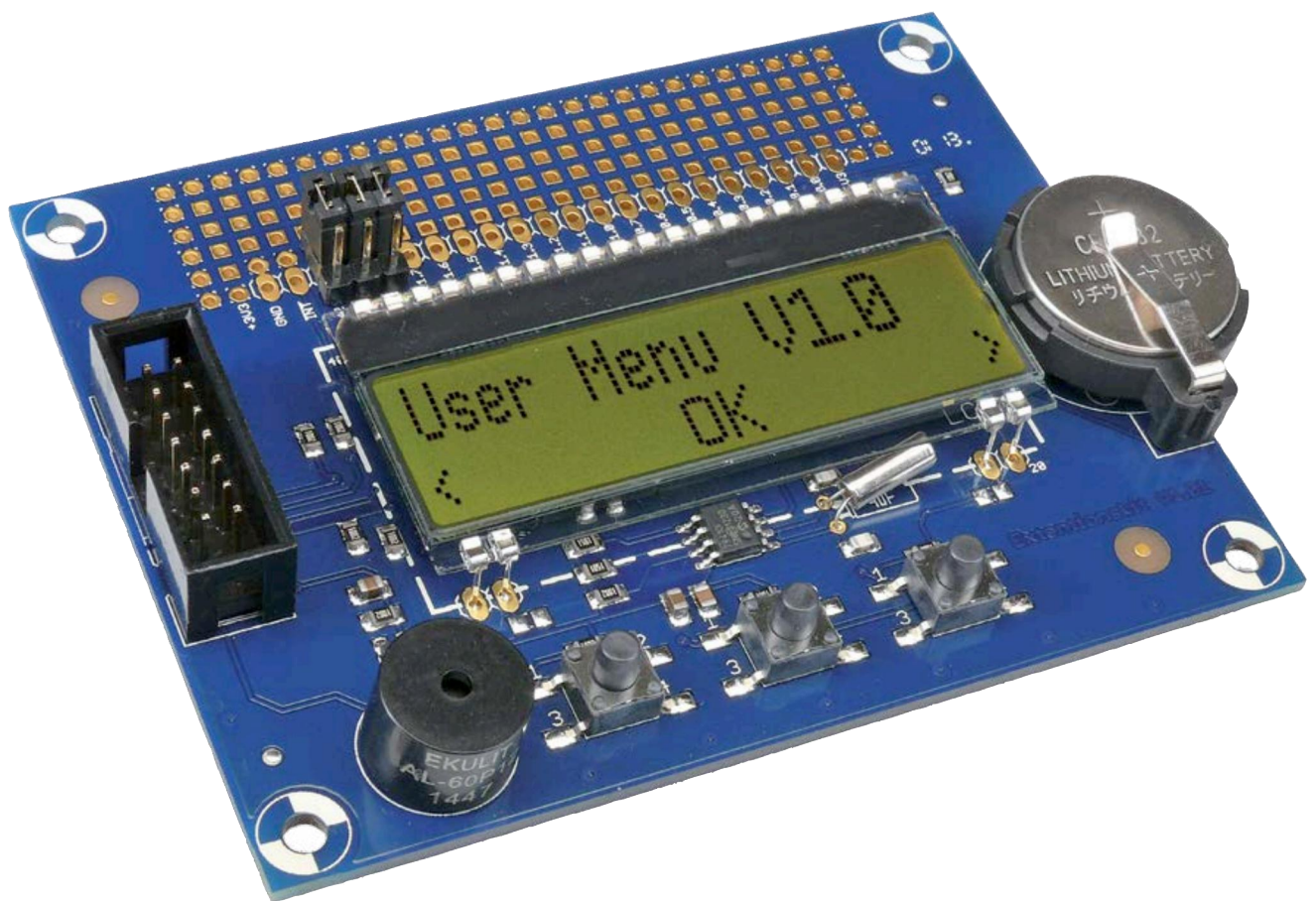


Figure 1.
The Elektor Linux Extension Board includes an alphanumeric display and three conveniently-located buttons.

Although in the smartphone and tablet markets there is a constant push towards higher and higher resolution screens, there are still many devices that only have a simple text display. Many printers, fax machines, NAS servers and mobile phones offer simple text-based menus navigated using buttons to adjust settings and view status information. In this article we will give an example of how you can implement something similar in your own projects.

Menu structure

The Elektor Linux extension board [2] includes a two-by-sixteen character display with three buttons placed immediately below it (see **Figure 1**). The board therefore makes an ideal hardware platform for our example.

First we have to make a list of the various menu items that we will want to adjust or view through the user interface. A simple menu system might resemble the following:

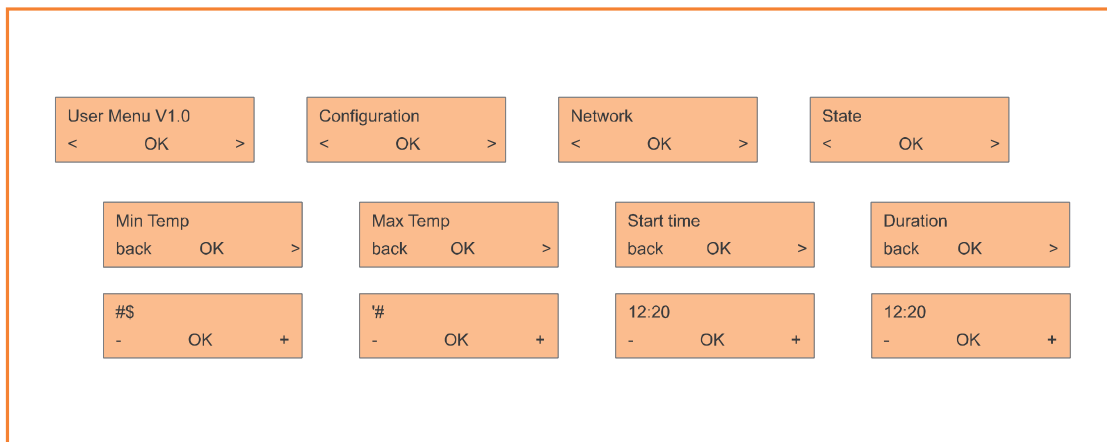


Figure 2.
Example menu structure.

- Configuration
 - Minimum temperature
 - Maximum temperature
 - Daily start time
 - Maximum duration
- Network configuration
 - IP address: dynamic or static
 - IP address
 - Network address
 - Gateway
 - DNS server
- Status
 - Current temperature
 - Processor load
 - Memory use
 - System uptime

The 'Configuration' menu allows the main settings to be adjusted. The minimum and maximum values can be set to control (for example) the switching behavior of the unit. A network connection also requires a few configuration parameters. The first thing to choose is whether the IP address for the connection is dynamically or statically configured: if static configuration is chosen the remaining parameters in the group have to be set. Well-designed software will of course check that all the IP addresses have the correct format.

Implementing the menu

When implementing the menu you can seek inspiration from devices you may already have: you can probably think of some more or less successful examples.

Figure 2 shows a sketch of how the menus that we want to implement might look. This is not

the complete set from the list above, but it does provide a representative sample.

Having decided what the user will see at each stage we need to decide on the logic that ties the various screens (including the sub-menus) together and on the types of operation that will be supported. Such operations might include:

- Static output
- Choice between options, with confirmation of selection
- Setting numerical parameters (up/down counter)
- Setting alphanumeric parameters
- Dynamic output (progress bars, live values, time etc.)

The easiest of these to implement are the static output and the straightforward choice (for example between static and dynamic IP addressing). Implementing parameter settings and continuously updating dynamic displays is more difficult. The naive approach of jumping in and writing code will typically result in the dreaded 'spaghetti'. The more satisfactory approach is to design a flexible and extensible solution, with the idea of making it easy to add new menu items to existing code without having to get too deeply involved in the software.

Our suggested approach

We looked at two possible ways of designing such a flexible and extensible solution. The first is a conventional library that allows the menu to be implemented along the lines of the pseudo-source-code shown in **Listing 1**. The functions

Listing 1. Pseudocode for constructing a menu.

```
configuration = MainMenu("Configuration")
network       = MainMenu("Network")
state        = MainMenu("State")

//Menu Configuration
mintemp = SubMenu(configuration,"Min Temp")
maxtemp = SubMenu(configuration,"Max Temp")
start   = SubMenu(configuration,"Start Time")
duration = SubMenu(configuration,"Duration")

//Menu Network
ipchoose = SubMenu(network,"IP Address: Dyn/Static")
ip        = SubMenu(network,"IP Address")
netaddress = SubMenu(network,"Network")
gateway   = SubMenu(network,"Gateway")
dns       = SubMenu(network,"DNS Server")

//Menu State
temperature = SubMenu(state,"Temperature")
cpu          = SubMenu(state,"CPU")
ram          = SubMenu(state,"RAM")
uptime      = SubMenu(state,"Uptime")

...

RegisterMenu(mintemp,MinTemperature)
```

Listing 2. Pseudocode for setting a parameter.

```
Function MinTemperature()
{
    mintemp = ConfigRead("mintemp")
    switch(button)
    {
        case "button_left":
            mintemp--
        case "button_ok":
            Back()
        case "button_right":
            mintemp++
    }
    ConfigWrite("mintemp",mintemp)
    Display("%s",mintemp)
}
```

MainMenu() and SubMenu() allow a tree structure to be built, and it is easy to see how this structure can be extended. When a new node is created a reference to it is returned, and this reference can subsequently be used to create new nodes beneath it in the tree. The function RegisterMenu() allows a function to be associated with a node, such that the function is called whenever the menu item is accessed.

Now we come to the implementation of these functions. When a menu sub-item, such as the minimum temperature setting, is selected a menu should appear that allows a numerical value to be adjusted. Pseudocode for this is shown in **Listing 2**. For this (and for any other function that allows parameters to be set) we need read and write access to a configuration file or database (ConfigRead() and ConfigWrite()). The button press is processed in a switch-case construct and the updated temperature value is displayed with a call to Display().

The web pages accompanying this article [3] include an example implementation in C that can be downloaded.

An alternative approach

An elegant alternative way to implement a menu system is using the (Linux) file system. A directory tree is created mirroring the desired menu structure. Each directory can contain a short piece of code that implements the menu item in question, and the output of this code can be directly shown on the display.

(130044)

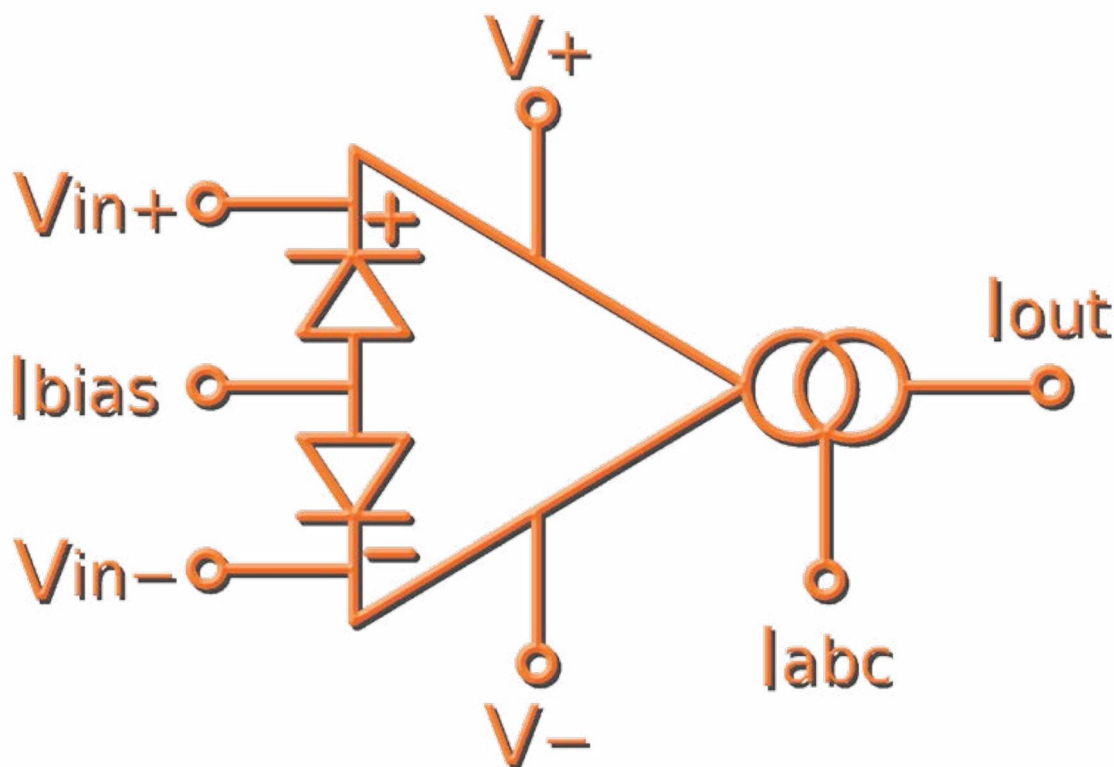
Internet Links

- [1] sauter@embedded-projects.net
- [2] www.elektor-magazine.com/120596
- [3] www.elektor-magazine.com/130044

OTA-based Triangular Wave Generator

“Do more with an operational transconductance amplifier”

Although many applications of OTAs have been described, the basic circuit of a non-sinusoidal oscillator exploiting an OTA's specific advantages is a rare find. This design brief describes a combined triangular and rectangular wave oscillator with two OTA circuits.



An OTA is a wonderful electronic component enabling a circuit to be designed whose parameters can be changed by setting the I_{abc} bias current without the need to change the parameters of the other components. Many applications of OTAs have been described and may be found in books and articles such as references [1], [2] and [3]. Here we set out to complement this body of knowledge with a generator circuit.

How it works

The circuit diagram of the generator is shown in **Figure 1**. Circuit OTA1 with capacitance C forms

an integrator. The capacitance C is supplied from the OTA1 output by a constant current I_{01} . Voltage V_C is linearly dependent on time t . Circuit OTA2 with resistors R_1 , R_2 forms a Schmitt trigger device. The output current I_{02} creates trigger voltage V_{R2} across the R_2 resistor. When voltage V_C reaches the threshold value, OTA2's output toggles to the opposite state, causing the output current to change its orientation to $-I_{02}$ and the V_2 output voltage polarity to toggle too. Using feedback the V_2 output voltage is fed to the input of the integrator. Its output changes its state to the opposite $-I_{01}$ by the V_2 voltage and the

By **Libor Gajdošík**
(Czech Republic)

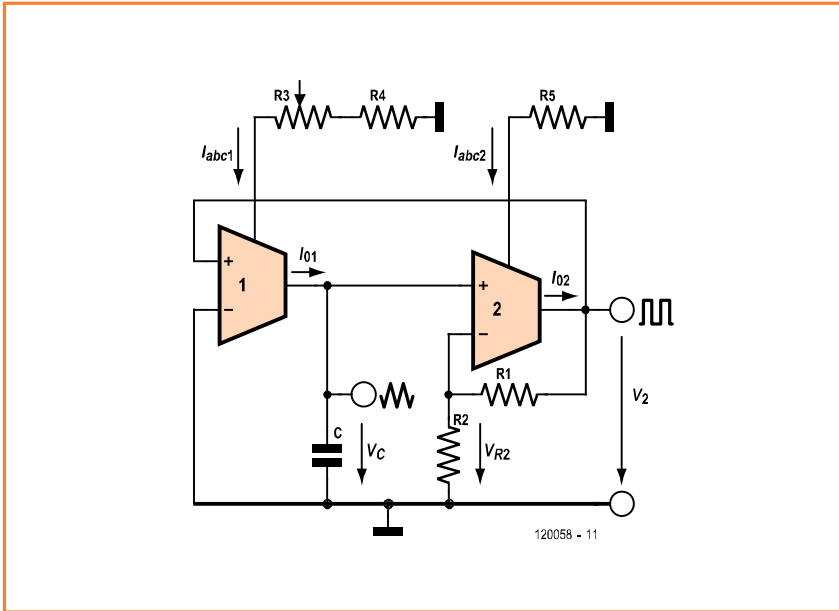
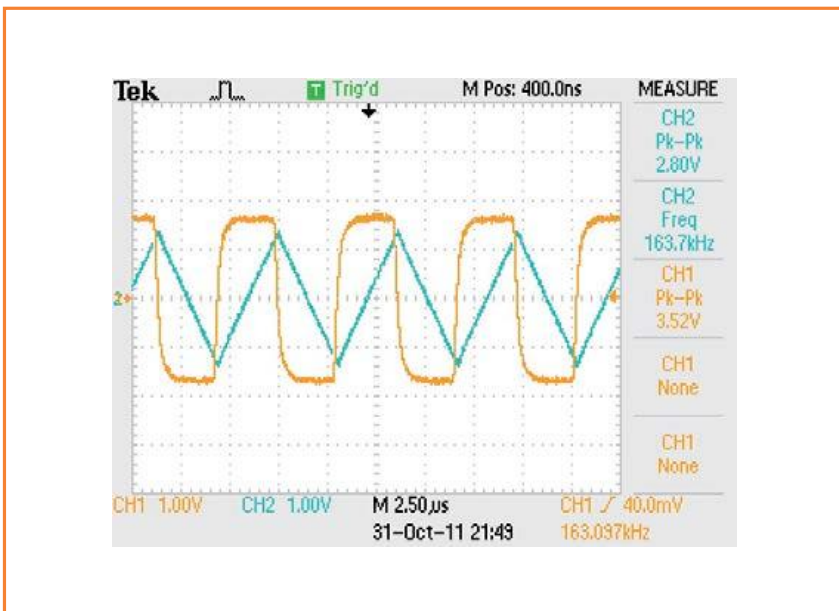


Figure 1. Basic elements that go into the making of the generator.

Figure 2. Rectangular and triangular waveforms produced by the OTA-based generator.



capacitance decreases down to the new value V_{R2} . The activity of the circuit repeats cyclically. The activity of the generator is based on the fact that under the given operational conditions the OTA output current has positive and negative saturation values, which appear when the input voltage exceeds the linear operational mode of the OTA. The range of the input voltage in the linear mode is usually small, 20 to 50 mV. In cases when the V_{R2} trigger voltage is much higher than the input voltage for the linear mode, this

input voltage can be ignored. The OTA input consists of the differential amplifier, hence both OTA input currents can be considered low and they are basically the same.

Within one period T the capacitance is charged from $t=0$ and $V_C=0$ until the time $t=t_1$, when V_C reaches the maximum value V_{Cm} . In an equation:

$$V_{Cm} = \frac{1}{C} \int_0^{t_1} I_{01} \times dt = \frac{I_{01}}{C} t_1 \quad (1)$$

This value V_{Cm} equals the trigger voltage V_{R2} , so we can write:

$$\frac{I_{01}}{C} t_1 = V_2 \frac{R_2}{R_1 + R_2} \quad (2)$$

Since V_2 is formed across the R_1 and R_2 resistors by the I_{02} current, the following is true:

$$V_2 = I_{02} (R_1 + R_2) \quad (3)$$

Let us insert (3) in (2) and calculate the time t_1 :

$$t_1 = \frac{CR_2 I_{02}}{I_{01}} \quad (4)$$

It is assumed that both values of the OTA output current, negative and positive, are the same. As the triangular waveform is a partly linear function and it is symmetrical both horizontally and vertically, the total period of the oscillations T equals $4t_1$, and for the frequency f the following formula is true:

$$f = \frac{1}{T} = \frac{I_{01}}{4CR_2 I_{02}} \approx \frac{I_{abc1}}{4CR_2 I_{abc2}} \quad (5)$$

For saturation and bias currents we have:

$$I_{01} \approx I_{abc1} \quad \text{and} \quad I_{02} \approx I_{abc2} \quad (6)$$

As the voltage divider on the output is not loaded, the following relation holds for the V_{Cm} amplitude of the triangular waveform:

$$V_{Cm} = V_2 \frac{R_2}{R_1 + R_2} = I_{02} R_2 \approx I_{abc2} R_2 \quad (7)$$

The simplifying assumptions for the derivation of formulas are not completely fulfilled. However, they can be used for an approximate estimation of the frequency and the voltage when designing the generator.

Practical results

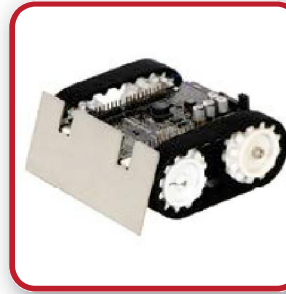
The measurements were performed with the following components: OTA type LM13700, resistors $R_1=820\ \Omega$; $R_2=2\ \text{k}\Omega$; capacitor $C=1\ \text{nF}$. The bias currents were realized by means of resistors connected between the bias input and the ground. For I_{abc1} the values $R_3=1\ \text{M}\Omega$, $R_4=10\ \text{k}\Omega$ were used. For I_{abc2} , the value $R_5=15\ \text{k}\Omega$ was used throughout, so as to keep a constant value of V_2 and V_{cm} for all measurements. The supply voltage was maintained at $\pm 10\ \text{V}$.

The output current I_{01} was found to be between $14\ \mu\text{A}$ to $0.87\ \text{mA}$ and the frequency, between $3.6\ \text{kHz}$ to $160\ \text{kHz}$. The output current I_{02} was measured at $0.57\ \text{mA}$. The measured waveforms are shown in **Figure 2**.

(120058)

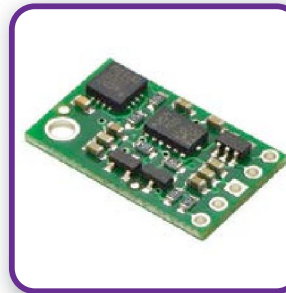
References

- [1] T. Parveen, *Operational transconductance amplifier and analog integrated circuits*, New Delhi, India, I.K. International Publishing House, 2009.
- [2] R.L. Geiger and E. Sanchez-Sinencio, "Active filter design using operational transconductance amplifiers: A tutorial", *IEEE Circuits Devices Magazine*, vol.1, pp.20 - 32, March 1985.
- [3] W.R. Grise. (1998, October). Operational Transconductance Amplifiers (OTA) for synthesis of voltage-controlled active-filter tuned oscillators. *TECHNOLOGY INTERFACE: The Electronic Journal for Engineering Technology*. Available online: <http://technologyinterface.nmsu.edu/fall98/electronics/grise/griseota.html>



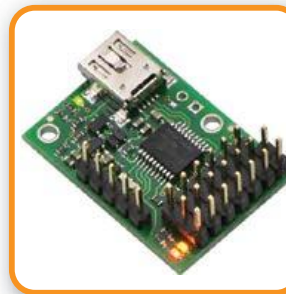
Zumo Robot

Arduino-controllable tracked robot small enough for minisumo (less than $10\ \text{cm} \times 10\ \text{cm}$) and flexible enough to make it your own.



MiniIMU-9 v2 Gyro, Accelerometer, and Compass

Provides 9 rotation, acceleration, and magnetic measurements that can be used to calculate absolute orientation.



Maestro USB Servo Controllers

Compact, versatile servo controllers that offer USB, serial, and internal scripting control.



Metal Gearmotors

Available in a wide range of motor types and gear ratios that let you choose the right combination of size, torque, and speed for your application.

Finding the right parts for your robot can be difficult, but you also don't want to spend all your time reinventing the wheel (or motor controller). That's where we come in: Pololu has the unique products — from actuators to wireless modules — that can help you take your robot from idea to reality.

Find out more at www.pololu.com

BaroStick

Measuring barometric pressure and air temperature with a USB stick



By
Ruud van Steenis
(Netherlands)

These days USB sticks are available for all kinds of applications, ranging from the storage of data to the measurement of temperature. USB sticks to measure barometric pressure are still quite rare however. The USB barometric pressure sensor described here uses a Bosch sensor to measure the air pressure and temperature, after which this data can be displayed graphically and stored using Windows software.

Every PC user is familiar with the handy USB memory sticks for storing files. The major advantages of such a USB stick are that it is easy to carry around, can be used on any PC and in most cases does not require a separate driver to make it work.

The USB stick concept is now also used for simple measuring equipment. For example, there is a Chinese manufacturer who makes USB sticks for measuring temperature and relative humidity. These are measurements that are of interest to many amateur meteorologists, among others. However, USB sticks to measure barometric pressure are thin on the ground. By using a pressure sensor made by Bosch it is nevertheless relatively easy to make a barometric pressure sensor yourself.

The pressure sensor used in the circuit, a Bosch BMP085 [1], has already been calibrated at the

factory (see the block diagram in **Figure 1**). The calibration values are stored in the BMP085 and have to be read out first, before making even the first serious measurement.

This is then followed by a calculation, using the measured value and the calibration values, before arriving at a sensible result.

The 'awkward' calculations are accommodated in a DLL [2], so that even the do-it-yourself programmer can easily build their own application around this barometric pressure sensor. Naturally, all the necessary routines are also accommodated in the accompanying applications for the measurement and logging of barometric pressure and temperature.

The circuit

The communications with the barometric pressure sensor BMP085 operates using the I²C pro-

to col. Care has to be taken with the fact that the I²C lines of the BMP085 are never allowed to be higher than 3.3 V. Since a standard USB port supplies a voltage of 5 V, a level-shifting circuit is therefore essential.

The 5-V power supply voltage of the USB port is, however, suitable for powering a small PIC-processor. In this design, the PIC18F14K50 in a SMD (SOIC) package is used for this.

In addition to an SMD crystal of 12 MHz, with accompanying capacitors, and a couple of capacitors for power supply decoupling there are otherwise very few components required, as the schematic in **Figure 2** shows. The power supply voltage for the BMP085 is supplied by a low-drop voltage regulator type AP1117.

The problem of the (bidirectional!) connection between the PIC, which is running off 5 V and the BMP085, which can tolerate at most 3.3 V, is solved as follows (see **Figure 3**).

Scenario 1: The line on the 3.3-V side is not logic Low. The gate and source of the FET in this line are both at 3.3 volts, so that the V_{gs} voltage is below the threshold value of the FET and the FET will therefore not be conducting. The 5-V side of the circuit is high because of the presence of a pull-up resistor.

Scenario 2: On the 3.3-V side the line is pulled Low. The source of the corresponding FET will now also be low, while the gate of the FET remains at 3.3 V. Voltage V_{gs} is now greater than the gate threshold value and the FET will conduct. The bus line on the 5-V side will now also be pulled low because of the conducting FET.

Scenario 3: The line is made low in the 5-V side. Firstly, because of the substrate diode in the drain, the 3.3-V side will also be low. Subse-

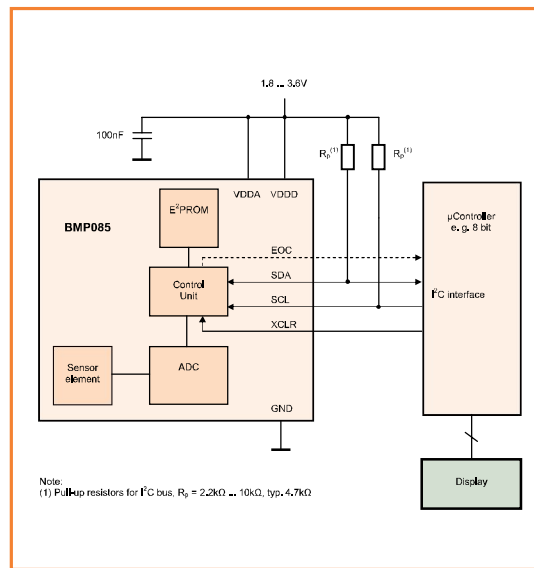


Figure 1. In addition to the actual sensor, the BMP085 contains an A/D-converter, an EEPROM with calibration details and an I²C interface controller.

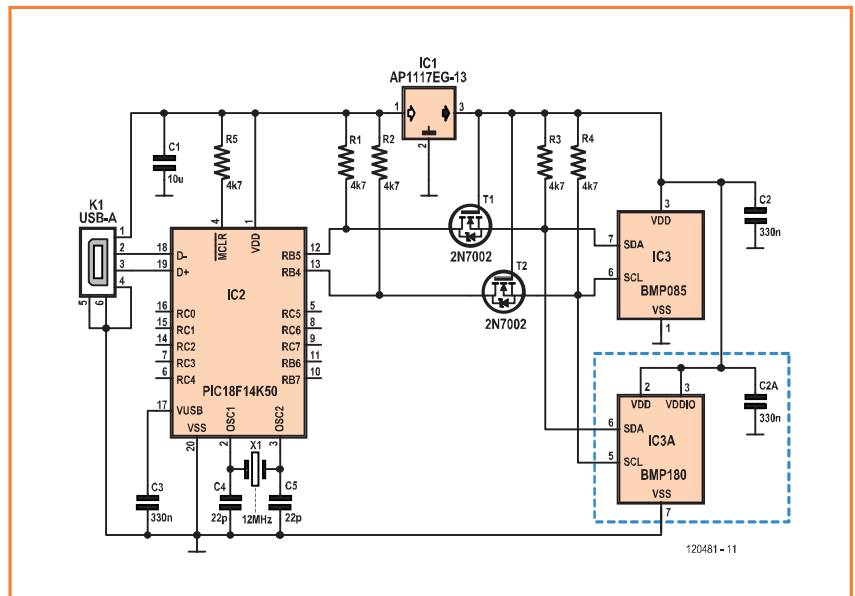


Figure 2. The schematic for the barometric pressure sensor and thermometer consists mainly of a Bosch sensor and a PIC microcontroller.

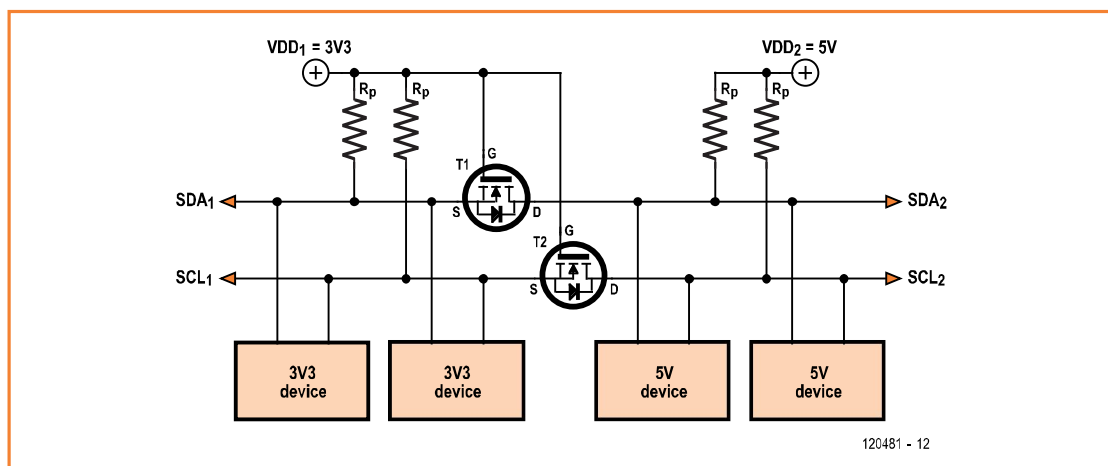


Figure 3. The level-shifting between the 3.3 V and 5 V sides is done with the aid of 2 FETs and a few pull-up resistors.

COMPONENT LIST

Resistors

(shape: 0603)
R1-R5 = 4.7kΩ

Capacitors

C1 = 10μF 10V (X5R)
C2,C3 = 330nF (0603)
C4,C5 = 22pF (0603)

Semiconductors

T1,T2 = 2N7002 (SOT232)
IC1 = AP1117E33G-13 (SOT-223)
IC2 = PIC18F14K50-I/SO, programmed, Elektor # 120481-41
IC3 = pressure sensor type BMP085 or BMP180 (Bosch Sensortec)

Miscellaneous

X1 = 12MHz quartz crystal, HC49US case
USB-A connector, surface mount, Lumberg type 2410 07
USB stick casing, type USB1SW, Conrad Electronics # 531275-89
PCB # 120481-1
Assembled and tested module, Elektor # 120481-91 [3]
Project source and hex code files, Windows software and DLL with source code, free download from [3]

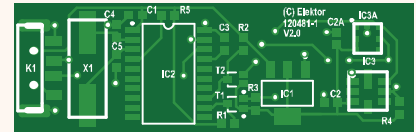


Figure 4. The circuit board has been designed so that it is suitable for 2 (compatible) types of sensor from Bosch.

quently, the FET will start to conduct, with the result that the voltage at both sides will be at the same potential.

In the schematic of **Figure 1** you can see that there are actually two sensors drawn, namely IC3 and IC3A (BMP085 and BMP180 respectively). The reason for this is that while the BMP085 is still readily available from several suppliers it has nevertheless been declared as 'obsolete' by Bosch. Its direct successor is the BMP180, which is completely compatible with the BMP085 with

respect to functionality, but has a different package and pin-out. The PCB for this project has been designed so that it is suitable for either type. Of course, you need to mount only one of them. At the moment, the ready-made modules supplied by Elektor contain the BMP085.

The construction of the circuit

A small circuit board was designed for this circuit, which fits exactly in a standard USB enclosure (see **Figure 4**). This circuit board has, as just mentioned, been designed in such a way that it is suitable for two different types of sensors from Bosch.

All parts, including the USB connector, are mounted on the copper side of the board. SMD components are used, but most of them are reasonably easy to solder by hand, if you have a little experience with this. The only really troublesome part to solder is the BMP085/BMP180 sensor. It is therefore recommended to start with this one. The best way is to start by applying a little solder to both those connections that are used on the BMP085 and the corresponding pads on the circuit board. After that, the sensor is positioned on the circuit board in the correct place and the corresponding copper traces of the board are heated one by one, if necessary adding a small quantity of (thin) solder. When using the BMP180, solder paste and a real reflow oven will be required, since the connections of the sensor are underneath the package. The other SMD components can easily be soldered using a normal soldering iron with a fine tip.

The author has made several prototypes without problems using this 'reflow' solder method. It is

Figure 5. The board is also available completely assembled, including the matching housing [3].



a good idea to check with an ohmmeter whether all the connections are indeed 'successful'. The PIC must be programmed first. Both the source code for the PIC (in PicBasic) as well as the resulting hex file which has to be programmed into the PIC, are available free from the web page for this project [3]. In order to program the PIC is not strictly necessary to use a special SOIC adapter. The author used a DIL-to-SMD adapter board [4] with gold-plated traces. During programming, the SMD PIC was simply pressed firmly against this adapter board. This did not give any problems in practice. It is still prudent however, to verify the contents of the program EEPROM of the PIC after programming to make sure that it is indeed the same as the contents of the hex file. Note that Elektor can also supply a pre-programmed PIC for those builders who have no programming experience. For those of you who do not want to do all this work themselves, but just want to use the BaroStick, Elektor can also supply a completely assembled and tested module, including the enclosure that you can snap on yourself [3].

The software

This project includes a relatively comprehensive Windows program that displays the measurement results (barometric pressure and temperature) graphically. The daily measurements can also be stored in a file and the measuring interval can be adjusted (see **Figure 5**). There are extensive options to set the language, colors, line widths, etc. to your personal preference. There is also the option of sending pictures of the 'meters' for barometric pressure and temperature to an FTP server at set times, for example for the purpose of a website showing weather information. For amateur meteorologists there is also the option of uploading the measurement results to the weather website 'weather underground' [5]. The Windows software can also be downloaded from [3]. For those who would like to do some programming themselves there is also a DLL (PSensor.dll) available. The usage of the DLL is explained in the (Delphi) test application DllDemo.exe. The source code for the DLL is also available.

Before connecting the USB stick to the PC for the very first time, it is a good idea to check the current consumption first by connecting the power supply connections of the USB connector

to a 5-V power supply. If the current consumption is around 10 mA then the USB-stick can be connected to the PC. The new hardware will be detected shortly after inserting the BaroStick. Since the BaroStick will behave as an 'HID Device', the installation of a driver is not necessary and Windows will announce after a short time that a 'Pressure sensor' has been found. Subsequently a new message will show that the new device has been configured. You can now use the application software and configure it as required.

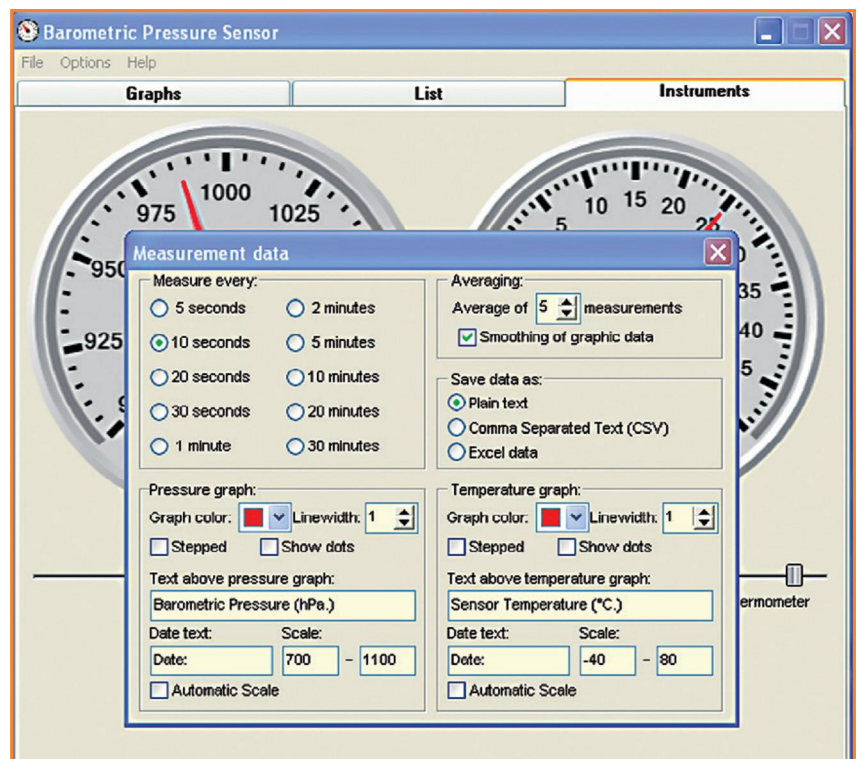
We wish you plenty of times with high barometric pressure!

(120481-1)

Internet Links

- [1] www.datasheetarchive.com/barometric-BMP085-datasheet.html
- [2] www.property-protection.nl/Temper
- [3] www.elektor.com/120481
- [4] www.voti.nl/common/pcb-12.jpg
- [5] www.wunderground.com/

Figure 6. The accompanying Windows program offers a wide range of configuration options.



Program Like the Professionals

Keep bugs at bay with state diagrams

By Peter Müller
(Germany)

Although originating in the world of hardware design, state diagrams are being used more and more in software development. This article looks at how you can use state diagrams to make your programs more robust and free of bugs.

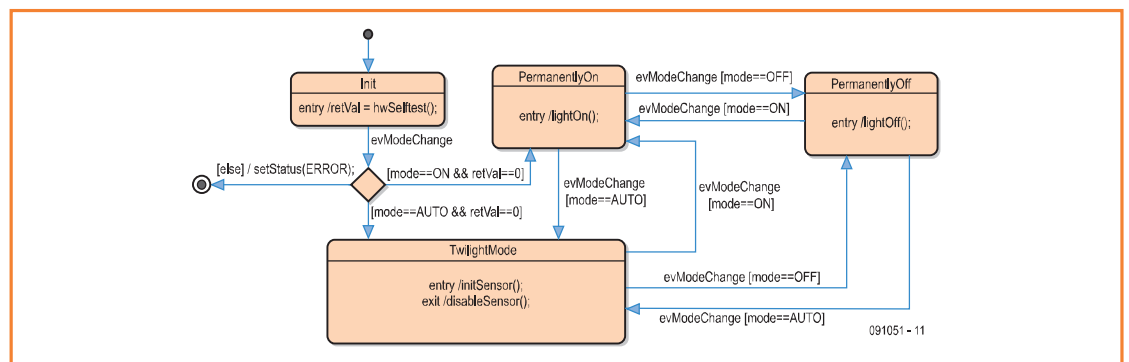


Figure 1. Simplified model of a twilight switch.

As the name implies, state diagrams can be used to set out an unambiguous description of the possible states of a system and the conditions under which this state changes. This is useful because in many cases devices have to be able to react to a wide range of internal or external events. However, the way the device reacts to an event is not always the same: it depends on the current state of the device. In other words, the history of events that have occurred in the past has a bearing on the way the device reacts to a new event.

ating mode (‘permanently on’, ‘party mode’ and so on).

We could add many further examples. Now, as soon as you start to analyze these systems as ‘state machines’ you will quickly see how powerful the approach is.

An example

As an example we will consider a twilight switch with a motion detector, as used outside many front doors. The system reacts to external events such as:

- changes to the operating mode (automatic, permanently off, permanently on);
- detecting a person;
- transitions from night to day, and from day to night;

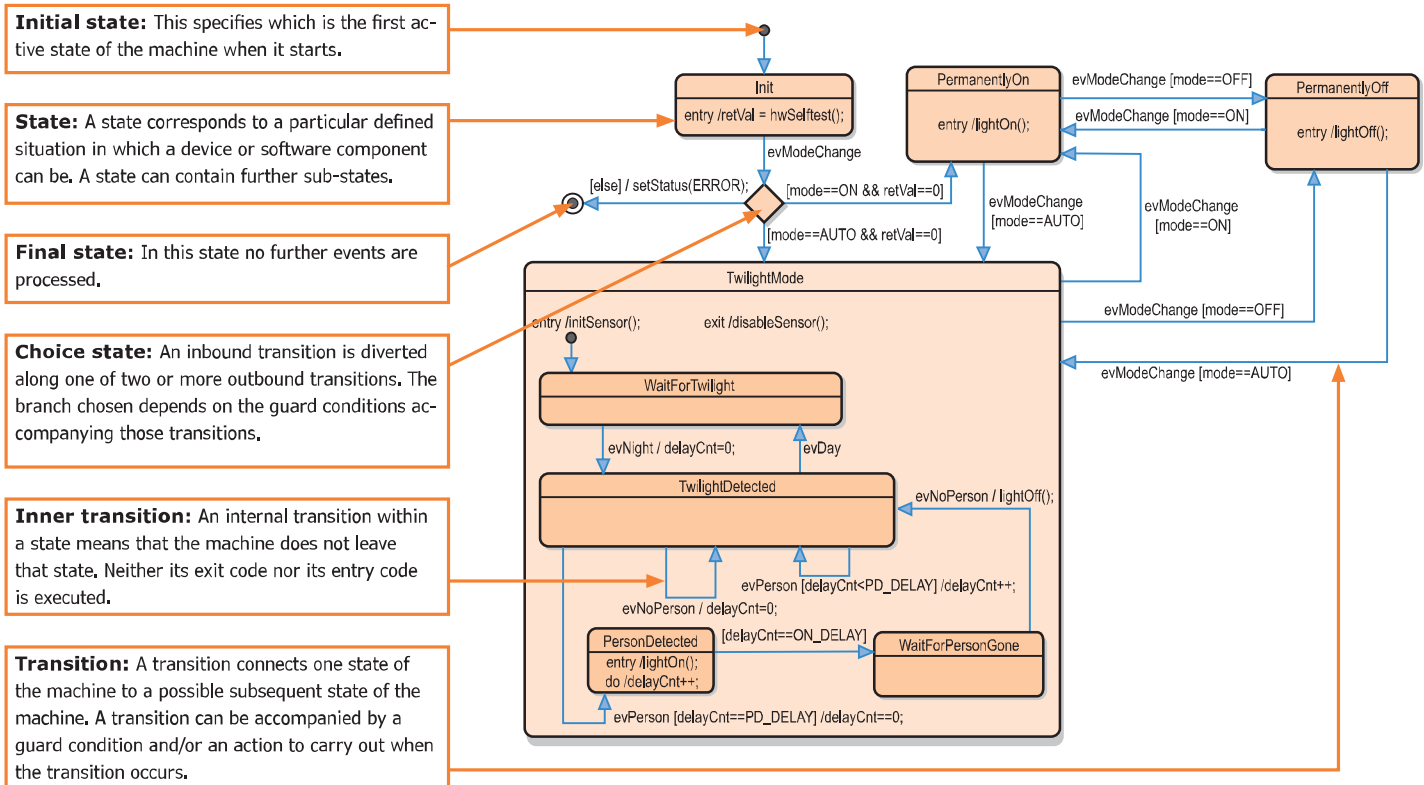
as well as internal events such as when a pre-set time-out occurs. In response to any of these events the device might turn the light on or off. Not every interaction is possible at any point in time: for example, a person is ignored if the light sensor indicates that it is daytime, whereas if a person is detected at night the light will be turned

Easy-to-understand

One of the great strengths of state diagrams is that even non-programmers can quickly learn to understand them. They therefore make a good basis for software reviews or for discussions with colleagues and customers.

Let us look at a few examples.

- An elevator reacts differently to the ‘open doors’ button depending on whether it is moving between floors or waiting at a floor.
- A drinks machine first asks for a drink selection, and then, once full payment is made, supplies the drink. However, the behavior changes if the selected drink is not available, or if the machine does not have enough change.
- An automatic twilight switch reacts differently to detecting people depending on ambient lighting conditions and on its oper-



Initial state: This specifies which is the first active state of the machine when it starts.

State: A state corresponds to a particular defined situation in which a device or software component can be. A state can contain further sub-states.

Final state: In this state no further events are processed.

Choice state: An inbound transition is diverted along one of two or more outbound transitions. The branch chosen depends on the guard conditions accompanying those transitions.

Inner transition: An internal transition within a state means that the machine does not leave that state. Neither its exit code nor its entry code is executed.

Transition: A transition connects one state of the machine to a possible subsequent state of the machine. A transition can be accompanied by a guard condition and/or an action to carry out when the transition occurs.

on. When power is first applied the twilight switch is in its so-called 'initial state'. Which states can be reached from this state can be determined at a glance from the state diagram. States are drawn as rectangles with rounded corners, and states are joined by arrows when a transition between them is possible. The text annotating the transition describes what event triggers the change from the state at the start of the arrow to the state at the end of the arrow. Whether an event actually causes the transition can be qualified by a 'guard' condition.

Figure 1 shows a simple state diagram for the twilight switch. The initial state of the machine is indicated by a small circle with an arrow coming from it. So, when the device is switched on, it goes into the state labeled 'Init'. A transition leads from this state via a lozenge-shaped 'choice' to one of three possible states, depending on the guard condition. It is possible to specify an action to occur:

- on entering a state ('onEntry');
- on leaving a state ('onExit');
- while remaining in a state ('do').

When the device changes state, first the 'onExit' code for the current state is executed. Then the 'onEntry' code for the new state is executed. So in this example when the twilight switch enters the 'Init' state it calls the self-test routine. If a fault is detected it is signaled to the user (via the 'else' branch of the choice). The circle with the solid black central disk indicates the final state of the machine. No further transitions are possible from here, and all events are ignored.

If the self-test is successful (returning a value of zero) the machine enters either the 'Twilight-Mode' or the 'PermanentlyOff' state, depending on the value of the variable 'mode'.

A transition can itself also be accompanied by an action (that is, a piece of code to execute). In this example the device status is set to 'Error' in the 'else' branch of the choice.

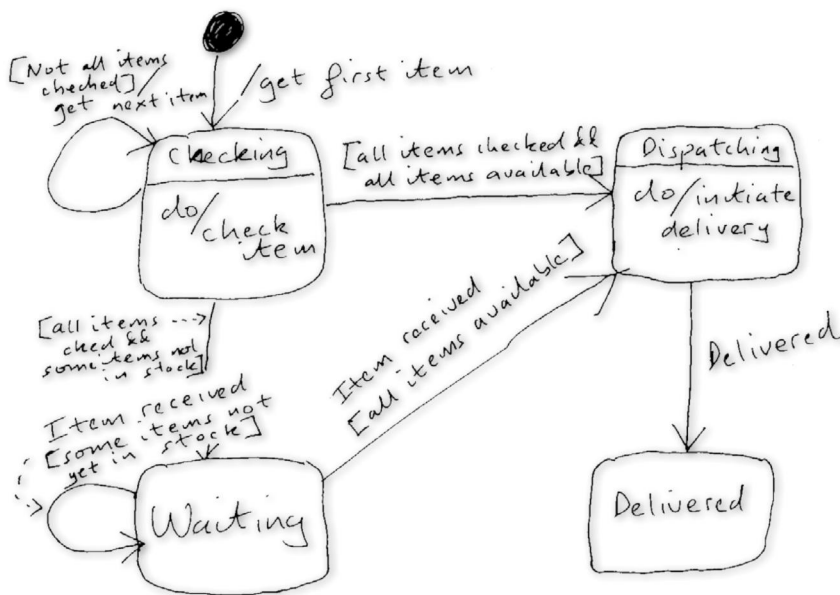
Refinements

State machines can be expressed hierarchically or in flat form. The machine shown in Figure 1 is flat: no state contains any sub-states. If we now proceed to refine the 'TwilightMode' state, we will make the machine hierarchical. **Figure 2** shows the refined version of the machine of Fig-

Figure 2. Diagram showing detail within the 'TwilightMode' state.

ure 1. When the machine goes into the state 'TwilightMode' it enters the initial state of the sub-machine, called 'WaitForTwilight'. If the machine receives the event 'evNight' it moves to the state 'TwilightDetected'. If a person is now detected the light will be turned on for a fixed period and then, assuming there is no longer any person present, turned off.

When looking at a state machine certain design decisions become clearly visible, whereas if the design were expressed purely in program code they would be hard to see. For example, the light remains on after the timeout has expired if a person is still being detected. If that possibility had not been foreseen then the light would briefly be turned off in this case. Also, at the transition from 'PermanentlyOn' to 'TwilightMode', the light is on. Is that sensible behavior or is another action needed?



Execution

State machines react to events. Depending on the system in question this reaction can take a time measured in milliseconds or perhaps seconds. For correct execution it is essential that an event is processed completely before the next event occurs. If this guarantee cannot be given, then a buffer, or 'event queue', must be used. In hierarchical diagrams events are first made available to the innermost active state. If the event cannot be dealt with there, it is offered in turn to each state up the hierarchy until the outermost state is reached. The event 'evModeChange', for example, can only trigger a transi-

tion from 'TwilightMode' if it has not already been consumed by a sub-state (which cannot in fact happen here). The basic idea is that inner states refine the behavior of outer states: that is why the state at the innermost level of the hierarchy must have the first chance to process an event.

State tables

Graphical presentation of state diagrams becomes unwieldy when there is a very large number of states. An alternative is to present the state machine in the form of a table. The rows and columns of the table are each labeled with all the possible states. The trigger for a particular transition is entered in the table at the intersection of the appropriate row and column.

Figure 3 shows an extract from the state table corresponding to the diagram of Figure 2. The tabular presentation is easy to understand. There are many empty cells, which is perfectly normal: undefined transitions simply correspond to events which are discarded. For example, in the state 'WaitForTwilight' the event 'evPerson' is not processed.

When reviewing a design it is a good idea to go through the cells in the state table systematically to check that the events, actions and guards have been specified correctly and, in the case of empty cells, that no transition has been forgotten.

Implementation

After the state machine has been designed and perhaps also reviewed it must be implemented. This is usually done using a switch/case construct.

[www](#) The document at [1] provides a listing of a program that implements the state machine of Figure 2 using a switch/case construct.

Turning non-trivial state diagrams into program code by hand is a rather tedious job. It is very easy to forget a transition or insert an entry action at the wrong point. More problematically, when changes are subsequently made, hard-to-track-down errors can be introduced very easily. Fortunately we have tools at our disposal which can create program code automatically from a state diagram. Ideally the whole system is generated automatically from the state diagram: this turns out to be a big advantage, as the design always remains consistent with the code and a great deal of tedious manual coding is avoided. In order to create directly executable code, a

	PermanentlyOff	PermanentlyOn	TwilightMode	WaitForTwilight	TwilightDetected	PersonDetected	WaitForPersonGone	Init	FINAL_0
PermanentlyOff		evModeChange [mode==OFF]	evModeChange [mode==OFF]						
PermanentlyOn	evModeChange [mode==ON]		evModeChange [mode==ON]					evModeChange [mode==ON && retVal==0]	
TwilightMode	evModeChange [mode==AUTO]	evModeChange [mode==AUTO]						evModeChange [mode==AUTO && retVal==0]	
WaitForTwilight					evDay				
TwilightDetected				evNight	evNoPerson evPerson [delayCnt<PD_DELAY]		evNoPerson		
PersonDetected					evPerson [delayCnt==PD_DELAY]				
WaitForPersonGone							[delayCnt==ON_DELAY]		
Init									
FINAL_0								evModeChange[else]	

few rules need to be obeyed when creating the design. These include the following.

- States must have unique names.
- State names must be valid identifiers in the target programming language. For example, in C, spaces are not allowed in variable names, and neither may they begin with a digit.
- Actions must either be simple function calls or valid program code.
- Guards must always evaluate to 'true' or 'false'.

Many design rules can be checked automatically by the code generator. Furthermore, other checks can also be carried out, such as the following.


- Are all states reachable? Are there any 'dead ends'?
- Are initial states defined wherever they are needed?
- Is a trigger specified for each transition?

These checks allow various infelicities in the design to be detected quickly and automatically. Moreover, the code generator will often come with additional features that help with debugging and analyzing the state machine.

Simulation and debugging

When developing a state machine it is very useful to be able to simulate the model. State transitions are triggered by simulated events, and this

lets you see what code is executed when and whether the machine is behaving as it should. In the next step, when the machine is running on a real device, it is very useful to be able to observe its operation. In the simplest case this might take the form of a simple textual display of the current state and incoming events.

 In the document at [1] there is an example that shows how you can write your own function to provide a tracing feature like this. Also, a second example shows how the tiny ASURO mobile robot [2] can be equipped with a bit of intelligence.

If you are interested in learning more about state diagrams, you can install a UML (Unified Modeling Language [3]) tool. In particular, the open-source tool ARGO UML [4] is recommended. Simple experiments with automatic code generation can be carried out using the demo version of Sinelabore [5].

(091051)

Internet Links

- [1] www.elektor.com/091051
- [2] www.youtube.com/watch?v=pIpuR_LLwY4
- [3] http://en.wikipedia.org/wiki/Unified_Modeling_Language
- [4] <http://argouml.tigris.org>
- [5] www.sinelabore.com

Figure 3. Presentation of the state diagram of Figure 2 as a state table (extract).

Sorry about my English

By Clemens Valens
(Elektor .Labs)

You are likely to have seen this phrase when searching desperately through online electronics discussion groups looking for that nugget of information oh so vital to make your new design work. When you finally find it in some obscure poorly written post three-quarters down the seventh page, you often read at the end: “sorry about my english” or words of similar intent. I have seen it in several places on the Elektor.Labs site too.



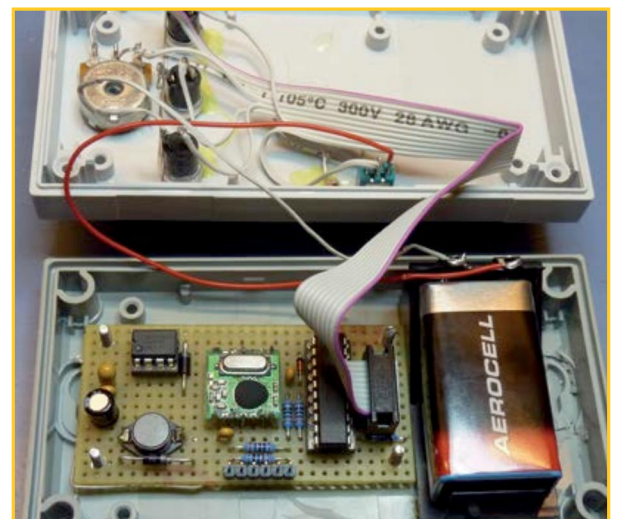
Dear Poster, let me tell you this, as the moderator of the .labs site I don't give a [♪♪♪] [♪♪♪] about your English, you just saved my [♪♪♪] day, maybe my [♪♪♪] job (sorry for my English ;-). So, please keep on posting the good stuff and stop worrying about your grammar and punctuation. I'm convinced most people will comprehend whenever it has their attention.



Brave Electronics Engineer Prevents Train War

When two competing train controllers unwillingly took control of each other's trains leading to severe incivilities between the two parties, electronics engineer waltro decided that it was enough, that things should stop there and that it was time to do something about it instead of trying to ignore the ongoing hostilities. So, one night, when both hostile parties were sleeping, our brave engineer broke into the trains and replaced their control systems by a more clever system of his own devising. The next morning peace was restored thanks to the multi-channel capabilities of the new control system. Well done, waltro!

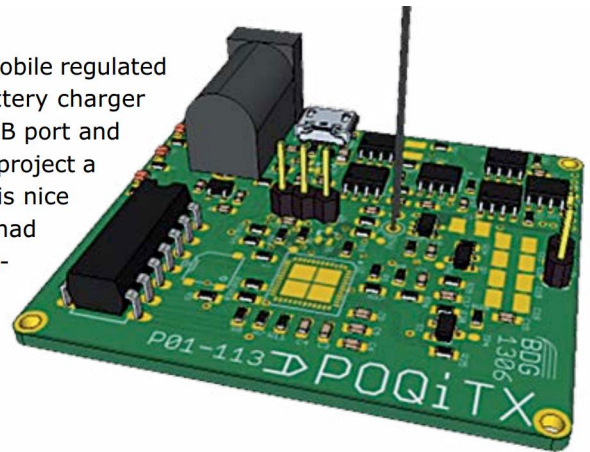
www.elektor-labs.com/node/3020



Perfectly Fine Project Killed By Bad Title

OP BifrostDevGrp was happily finishing his design of a small and clever mobile regulated 3.3 V / 5 V battery operated power supply including a multi-source battery charger capable of recharging its 950 mAh Li-ion cell from a wall wart, from a USB port and even wirelessly. The suddenly disaster struck: the OP decided to give its project a funny name. Not realizing what he just had done, the OP then posted his nice design on the Elektor.Labs website using an even funnier title. If only he had taken care that the first lines of his description were captivating, the damage might have been limited and his project might have been saved from oblivion. Unfortunately, he didn't. Luckily we are there looking out for such projects and trying to rescue them.

www.elektor-labs.com/node/2969



OP Launches Quest For Best Low-Power Tube Amplifier Ever

Passion, although too often for money, is key to achieving great things. OP Ken has a passion for great sounding audio, launching a quest to design and build the best low-power tube amplifier ever created. The OP has done a lot of advance work in selecting parts and materials but he hasn't yet nailed down all the details. To do so he requests your help. Do you get a kick out of audio? Would you like to be (part of) the best ever? Yes you would, wouldn't you? (Be honest, who wouldn't?) So give your opinion on transformers, capacitors and other sound determining design subtleties and Make This Happen!

www.elektor-labs.com/node/2949

Waste Collection Calendars Too Complicated

I had never given the waste collection schedule in my town much thought until it was changed recently and we started to forget putting the right bin out at the right time. What we needed was a reminder. And what do you know? MarkDonner posted an idea for just such a device on .Labs. And then amigaman joined in and gave a link to a German trash bin forgetters support group and I discovered that I was not alone. There are other people like me with the same disorder and now we also have our own support group. Are you looking for help too? Join us and participate in creating a fool-proof designer electronic put-the-bin-out reminder system doohickey thing.

www.elektor-labs.com/node/2946



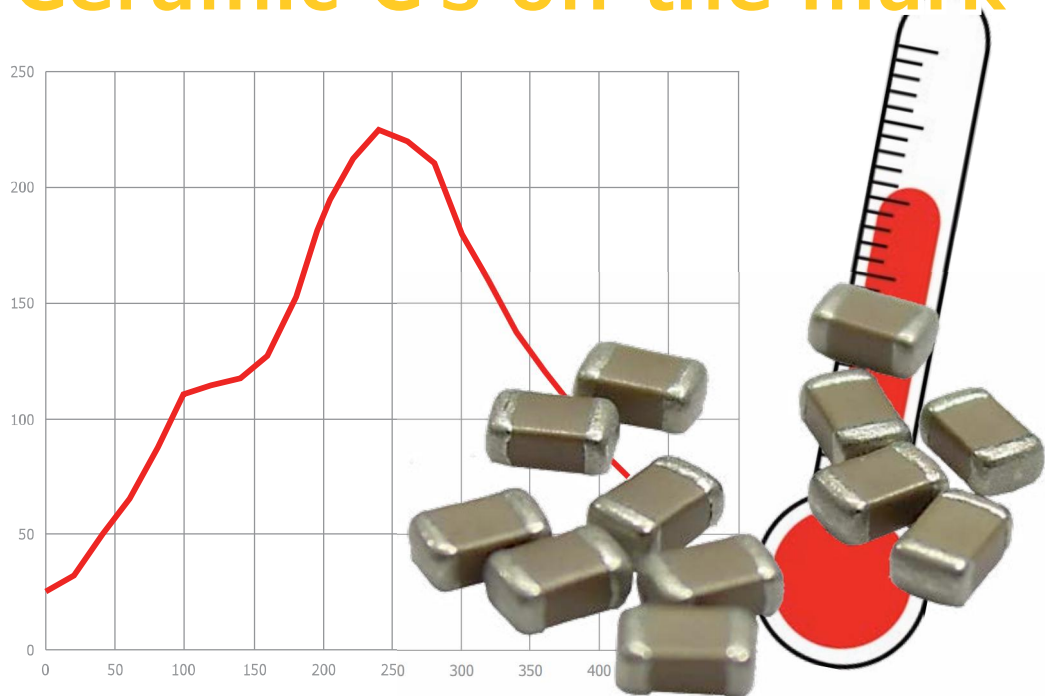
(130090-1)

www.elektor-labs.com

Note: OP stands for Original Poster, the person who started an online project or discussion. OPs who want to have a chance of appearing in the printed edition of Elektor must (regularly) check the email address they use to access Elektor.Labs. This is our only means of contact.

Ceramic C's off the mark

By Thijs Beckers
(Elektor Editorial)



Never a dull moment in Elektor Labs. While assembling a prototype of our latest *500 ppm LCR Meter*, lab worker Jan Visser ran into an easily overlooked property of ceramic capacitors. It happened when the assembly called for ceramic capacitors with a value of 150 pF. Freshly ordered, brand new out-of-the-box components were mounted on the PCB using our trusted SMD rework station. But during the calibration procedure of the LCR Meter some hiccups occurred. Jan wasn't able to get the device to complete the procedure correctly.

Suspicious of deviating component values, Jan measured the ceramic capacitors used by the LCR Meter for the calibration cycle. He measured 157 or so pF instead of the promised 150 pF. This turned out to be a little too high for the calibration procedure to finalize correctly. Investigating the suspected ceramics, Jan measured a few more from his fresh stock. The indicated value of 150 pF was exceeded by a large margin: all measured brand new capacitors showed values between 168 and 183 pF, instead of 150 pF \pm 5%. This prompted us to engage in a small test. We put a couple of capacitors through a standard reflow soldering cycle in our SMD oven and measured their values before and after the heating process. Before the ceramics were exposed to heat of the soldering cycle their values varied

between 170 and 180 pF. But after the cycle their values had dropped significantly — almost exactly to their proper value, that is. For example, a 178-pF cap dropped to 156 pF and a 173-pF cap dropped to 152 pF after being run through the reflow soldering process.

In our rush to meet the article deadline for this issue we were unable to pinpoint the exact reason for this phenomenon, but we suspect the dielectric to play a role in the process. If you think you know the details and care to bring us up to speed, do send us an email at editor@elektor.com; subject line: ceramicz. In return for your effort and a meaningful answer we may award you an mbed LPC11U24 kit* [1] and publish your clarification in a future edition.

Of course all of you seasoned engineers out there already know that there are certain (heating) protocols involved in the production process, but if you're a new kid on the block and/or don't have access to a professionally tuned production line, you may want to check on temperature critical components and give them the attention they need.

*courtesy of NXP

(130118)

Internet Link

[1] <http://mbed.org/handbook/mbed-NXP-LPC11U24>

PCB production pitfalls

Wave soldering is a technique often used to solder complete PCBs very quickly in production lines and facilities. It is best described as running the board over a 'wave' of liquid solder, causing all pads and components on one side of the PCB to be soldered quickly. The Internet provides tons of pictures and short videos showing how it works in practice — "a picture is worth a thousand words". This process has been used by the industry for many years now and should have reached its peak abilities.

But there are some things that just can't be avoided in this process. Take a close look at the first picture. Notice that the solder on one end of the SMD capacitor — the left side — has a much steeper ramp than the solder on the other end. That's not just because there's a via there and consequently no solder mask to keep the solder from flowing.

When the PCB is run through the wave of solder, its direction and that of the solder wave result in slightly less solder on the 'front' and a little more on the 'back' of the component. Just like a comet's tail indicates the direction of the sun, the solder 'tail' indicates the direction of the solder wave. This effect can also be seen with the SMD resistor on the far left. Apparently, in this case the solder wave passed the PCB left to right. Other than nice to know, this implies some restrictions to the design of a PCB as well. Take a close look at the second photograph of the same PCB. The wave soldering direction is identical to the direction in the first photograph — from left to right. You may already spot the problem.

On the left side of the inductor (upper, black component) and capacitor (beige component below the inductor) you can see the pads are connected to each other by (PCB) design. But on the right there's an obvious short between the pads as well, rendering the components useless and the circuit faulty (in this case a massive short from power to ground).

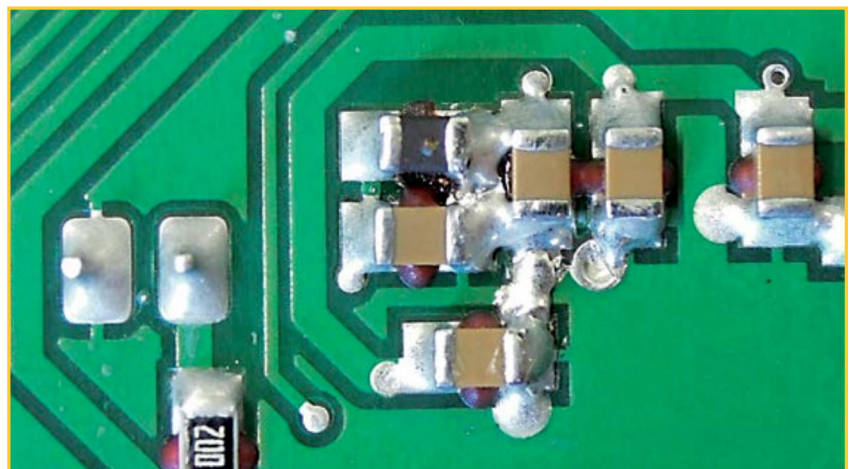
How could this have happened? Surely there's solder mask in between the pads, now buried underneath a lump of solder. But this didn't prevent the short. This shows the limits of the wave soldering process.

All would have been fine if it weren't for the capacitor placed directly adjacent to the LC-decoupling circuit acting like a wall for the sol-

der wave, trapping solder in between the three components. No amount of solder mask is going to prevent this from happening. The only solution is to redesign the PCB (moving the components away from each other), or switch to an alternative production process. Of course it would be best if the designer knew upfront which production process was going to be used so he or she could account for these kinds of traps (pun intended). ...The number of things you have to keep in mind when working on electronics! And in case you wonder, the reddish bulges sticking out from underneath the components are glue to hold the components into place during the wave soldering process. The side you see in the pictures is actually the bottom side during the wave soldering process (SMDs are always upside-down when wave soldering is used) so if the components are not glued to the board they're sure drop off or disappear with the solder wave. No worries, the employees at the PCB manufacturing facility will take care of that ;).

(130027)

By Thijs Beckers
(Elektor Editorial)



Raspberry Pi: one year later, one million sold

Interview by
Clemens Valens
Transcription by
Joshua Walbey



The April 2012 edition of *Elektor*^[1] featured an interview with Eben Upton, one of the founders and trustees of the Raspberry Pi Foundation. Back then the first production batches started to arrive, and since then almost a million boards were sold. We met Peter Lomas, the ‘R-Pi’ hardware guy, at the Embedded World 2013 trade show in Nuremberg, Germany, and found it an excellent opportunity to catch up with the Raspberry Pi Foundation.

C: Raspberry Pi, the phenomenon. It is quite amazing what happened.

P: *It is*, and lots of people keep asking me, why has Raspberry Pi done what it has done, what makes it different? I think it’s something we’ve really been trying to grasp. The first thing that happened with Raspberry Pi which I think is important, is that we had one of our very first prototypes on a UK blog for one of the BBC correspondents, Rory Cellan-Jones, and they made a little video, a YouTube video, and that got 600,000 hits. So I guess that if you look at it from one aspect that created a viral marketing, a very viral marketing campaign for Raspberry Pi. The other I think, the name, Raspberry Pi was key. And the logo that Paul Beach did for us is absolutely key because it has become iconic.

C: Yes, it’s very recognizable.

P: Very recognizable. If I show you that, you know exactly what it is, in the electronics circle. So I think the brand has been very important. But, you know, we shouldn’t forget the amount of

work that Liz Upton’s been doing with the blogs and on our website, keeping people informed about what we’re doing. Then, I think we’ve got the fact we are a charity with the aim of promoting and that is a big single point of difference; that we are focused on the education of computing and electronics and that’s our motive — *not* actually to make boards and to make money except to fund the foundation.

C: I had a look on the Raspberry Pi website, and it doesn’t look easy to me. You target education, children, and on the website it’s hard to find what Raspberry Pi exactly is; it’s not really explained. You have to know it. There are several distributions, so you have to know Linux and you have to program in Python...

P: Well, that’s true and, in a weird way, that’s part of its success, because you actually have to be active. In order to something with Pi, you can’t just get it out of a shiny box, put it on the desk and press ‘On’. You have to do some mental work. You have to figure some things out. Now, I

actually think that there's a bit of a benefit there, because when it actually works, you have some achievement. You've done something. Not 'we've done something'. YOU'VE done it personally and there is a gratification from doing it.

C: But it's not the easiest platform.

P: No, but with our educational proposition, the whole object now is to package that up in easier to use bundles. We can make the SD card boot straight to Scratch^[2], so Linux becomes temporarily invisible, and there's a set of worksheets and instructions. But we're never going to take away, hopefully, the fact that you have to put your wires in and I do think that is part of the importance and the attraction to it.

C: Because of all these layers of complexity and having to program it in English [Python is in English, ed.], for the non-English population it is yet another hurdle. That's why Arduino was so successful; they made the programming really easy. They had cheap hardware, but also the way to program it is very easy.

P: There's no doubt Arduino is a brilliant product and you are right, it allows people to get to what I call "Hello World" very easily. But, *in fact*, on a Raspberry Pi, after you've made those connections and plugged the card in, you can get to an equivalent "Hello World", but ours is the Scratch cat. Once you've moved the Scratch cat (**Figure 1**), you can go in a few different directions: you can move it some more, or you can use Scratch with an IO interface to make an LED light up or you can press a button to make the Scratch cat move. There are endless possibilities of directions you can go. I've found, and I think Eben [Upton] has experienced similar, that kids just get it. As long as you don't make it too complicated, *the kids just get it*. It's the adults that have more problems.

C: I saw that there are at least three different distributions for the boards. So what are actually the differences between the three? Why isn't there just one?

P: Well, they all offer subtly different features. The whole idea was to make Raspberry Pi as an Undergraduate tool. You give it to Cambridge University, hopefully Manchester University, and undergraduates can view the science *before* they start it. They have the summer; they can work on it, come back and say, "Look I did this on this

board". That's where it all started.

C: Okay. So, you were already on quite a high level.

P: Well, we were on a high level, that's true. We were on a high level so Scratch wouldn't have been on the agenda. It was really just Python - that's actually where the Pi comes from. What has really happened is that we've developed this community and this eco-system around Pi and so we have to be able to support the, if you like, "different roots" of people wanting to use Pi. Now we've got the RISC OS that you can use. And people are even doing bare metal programming. If we just gave one distribution I guess we're closing it up. I fully approve of having different distributions.

C: From the website, it's not clear to me what is different in these distributions. For the first one is written "If you're just starting out".

P: I think maybe we do need to put some more material in there to explain to people the difference. I have to explain: I'm the *hardware guy*. I'm the guy sat there connecting the tracks up, connecting the components up. My expertise with the operating systems, with the distributions that we have, is really limited to the graphical interface because that's what I use day in, day out.

C: Okay. Once you have chosen your distribution, and you want to control an LED, you have to open a driver or something, I suppose?

P: Well, you've got the library; you just have to make a library call. Again it's not easy. You have to go and find the libraries and you have to download them. Which is where things like Pi-Face^[3] [add-on board, ed.] come in because that comes with an interactive library that will go onto Scratch and you've got the Gertboard^[4] [another extension board, ed.] and that comes with the libraries to drive it and some tutorial examples and then you can wind that back to just the bare metal interface on the GPIOs.

C: So, the simplicity is now coming from the add-on boards?

P: Some of the add-on boards can make it simpler, where they give you the switches and they give you the LEDs you don't need to do any wir-



Figure 1. The cat from Scratch, a multi-media programming environment aimed at children.

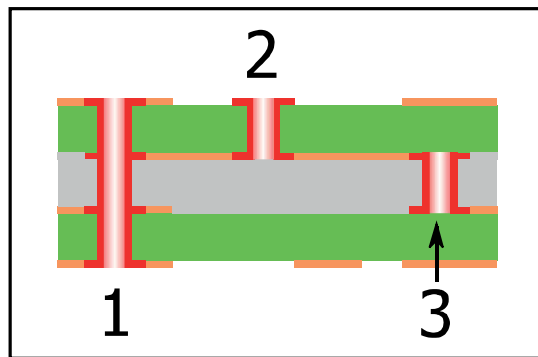


Figure 2. Three different vias: 1 – normal via; 2 – blind via; 3 – buried via. Imagine a six layer board with blind vias connecting layers 1 and 2. These two layers have to be manufactured as a separate doubled-sided plated through board before they can be bonded to the other four layers. This assembly is then once more drilled and plated. In total two drilling and plating processes are required instead of just one in the case of a standard board without blind or buried vias.

ing. My view is that I'm trying to make it like an onion: you can start with the surface and you can do something, and *then* you can peel away the layers. The more interested you get, the more layers you can peel away and also, the more

to "okay, well what if I try and do this?" and then that's the Nirvana. Certainly for the kids that's crucial. Because we're changing them from where they do what they're told, to start doing things that they think they might be able to do - and *try it*. That makes them into engineers.

C: Let's move on to the hardware of the board.
P: Sure.

C: So, you chose a Broadcom processor, because Eben, he worked at Broadcom?

P: He still works within Broadcom. It would be hard for me to argue that that wasn't an influence in the decision, because Eben said: "*Oh look, here's the bright shiny chip. It can do all the things that we want, why wouldn't we use it?*". The decision we made is we nailed our credentials and our reputations to the website by saying it will cost \$35 - it will cost \$25 for the basic one - and there was no way on earth any of us were going to go back on that. Unless, sometimes, as they say, *shit happens* and you can't do it. But we had a spreadsheet, the basic

A blob of solder and they all burst into life

different directions you can go in what you do with it. You *must* have seen the diverse things that can be done?

C: I've looked at some of the projects. I was surprised by the number of Media Centers. That's how RS are promoting the board. Aren't you disappointed with that? It seems to be, for a lot of people, a cheap platform to do a Linux application on. They just want to have a Media Center.

P: I know exactly what you mean and I suppose I should be disappointed that some people buy it, they make it into a media center and that's all it does. *But*, I think if only five or ten per cent of those people who make it into a media center will think: "*Well, that was easy, maybe I'll get another and see if I can do something else with it*", then it's a success.

C: It would be an enabler.

P: Getting the technology in front of people is the first problem. Getting the "*Hello World*" so they've got a sense of achievement is the second problem. Then turning them over from doing that

numbers looked plausible, we just had to do a lot of work to chop it down - to hone it, to get it tight so it would actually meet the prices. So, I think if we'd gone another way, like maybe with Samsung, that would have blown the budget.

C: Did Broadcom help in any way to make this possible?

P: Every semiconductor manufacturer helped the project by making the chips available and, also, the price point of the chips is important. I think some of the people that helped us took an educated gamble and gave us good pricing from day one. Because the big problem that you get with trying to bootstrap any project, is that if you don't know what your volume is going to be, you have to be conservative. So, initially, we priced up for a thousand boards, but quickly we priced up for twenty thousand boards, but nowhere in our wildest dreams did we think we were going to get to a two-hundred thousand board requirement on launch day *and* being so tantalizingly close to selling a million after our first year. So that's helped in a lot of ways, because obviously

it's driven the price of all the components down. I'm not going to pretend it doesn't please the vendors of the components that had faith in us from day one, because they've obviously made some money out of it.

We always had the rationale that we had to have a sustainable model where the foundation, our community that is buying the boards and our suppliers were all making a living and can feed themselves. It would have been a total disaster if someone like Broadcom had said: "Tell you what guys, let's give you the processors. We'll give you the first twenty-thousand." And so we could've added all sorts of extra bells and whistles to the design and then, when we would have sold these twenty-thousand boards, we're going to put the price of everything up by \$12. That would've been the end of Raspberry Pi.

C: If Eben and the others had not worked for Broadcom...

P: "Would we have used a different chip?" Well, I sort of speculated this and I went 'round and had a look and, at the time for the price point, we couldn't find anything that would've met our requirements as well as that chip. So I was comfortable that was the one that would allow us to get to where we wanted to be, and I think the big key crunch for that was the HDMI.

From a technical point of view, one of the challenges we had was getting the breakout under the BGA, because blind and buried vias on PCBs are very expensive (**Figure 2**).

C: Peter, it all went really fast

P: Oh yes, it's gone like a rocket!

C: Have you personally learned something valuable from it?

P: Well, I've learned lots of things. I think the most valuable, maybe not a lesson, but a reinforcement of something I already thought, is that education doesn't just exist in the class room. It exists all around us. The opportunity to learn and the opportunity to teach exists every day in almost every aspect in what we do. You know, there are people who spend their life trying to keep every secret, keep everything to themselves, but there are also people who just give. And I've met so many people who are just givers. I suppose I've learned there is a whole new system of education that goes on outside of the standard curriculum that helps people do what they want to do.

(130101)

Links & References

- [1] Interview with Eben Upton: 'What are you doing?'
Elektor Magazine, April 2012, p. 39.
- [2] Scratch: <http://scratch.mit.edu/>
- [3] Pi-Face: <http://pi.cs.man.ac.uk/interface.htm>
- [4] Gertboard: http://elinux.org/RPi_Gertboard

BEST SCOPE SELECTION & lowest prices!

IPHONE SCOPE

5MHz mixed signal scope adapter for the iPhone, iPad and iPod Touch!
The FREE IMSO-104 app is available for download from the Apple App Store.

IMSO-104 **\$297.99**



30MHZ SCOPE

2-ch, 250MS/s sample rate
30MHz scope with 8" color TFT-LCD, AutoScale & waveform mathematic functions.
Quality FREE carry case included.

SDS5032E **\$299**



60MHZ SCOPE

60MHz 2-ch scope with 500MSa/s rate & huge 10MSa memory!
8" color TFT-LCD & FREE carry case!

SDS6062 **\$349**



100MHZ SCOPE

High-end 100MHz 2-ch 1GSa/s benchscope with 1MSa memory and USB port • FREE scope carry case. Super low price!

DS1102E **\$399**



100MHZ SCOPE

100MHz 2-ch scope with 1GS/s sample rate and 8" color TFT LCD. Huge amounts of memory • FREE scope carry case.

SDS7102 **\$429**



100MHZ MSO

2-ch 100MSa/s scope + 8-ch logic analyzer. USB 2.0 and 4M samples storage per channel with advanced triggering & math functions.

CS328A **\$1359**



20MHZ HANDHELD

Fast & accurate handheld 20MHz 1-ch oscilloscope.
- 100 M/S sample rate
- 3.5 in. color TFT-LCD
- 6 hour battery life
FREE rugged, impact-resistant case!

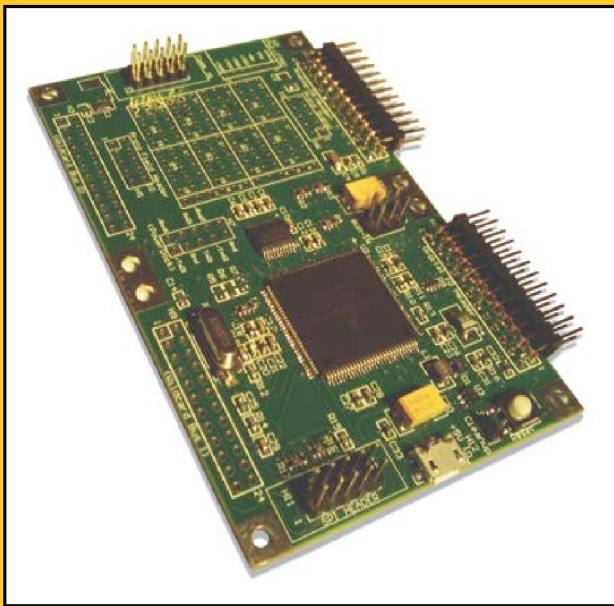
HDS1021M **\$269.95**



WWW.SAELIG.COM

Saelig .com
unique electronics





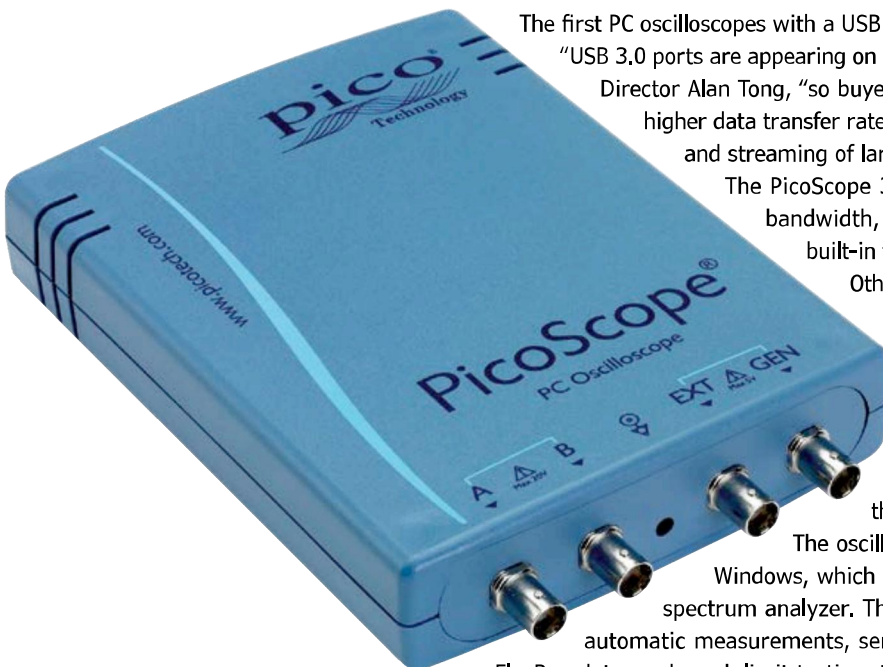
Low cost SBC simplifies instrument control

The PDQ Board Lite is a low cost single board computer and development board that hosts the Freescale HCS12/9S12 MCU and an embedded RTOS. This GNU C programmable instrument controller is well suited for data acquisition and control, PWM drive, I²C sensor interfacing, laboratory automation, scientific instruments, SCADA, instrumentation and automation.

A compact 2.5" x 4" board, this simple SBC provides all the I/O of the Freescale MC9S12A512 processor chip, including dual logic-level and standard RS-232 serial ports, 10-bit resolution analog inputs, I²C, dual SPI links, PWM, and timer-controlled digital I/O. The PDQ Board Lite is powered by +5 volts delivered via one of the I/O headers or from a standard micro-USB connector, the same type used on many cell phone chargers. The PDQ Board Lite is programmed using the C language. It contains an embedded RTOS in firmware and is programmed using a C integrated development environment (IDE). The Mosaic IDE+ is a comprehensive GNU environment that simplifies the coding of any multitasking application and allows users to edit, compile, download, interactively debug, and run application programs.

www.mosaic-industries.com (120749-III)

World's first USB 3.0 oscilloscopes launched



The first PC oscilloscopes with a USB 3.0 interface have been released by Pico Technology. "USB 3.0 ports are appearing on most new computers and laptops," explained Managing Director Alan Tong, "so buyers of USB oscilloscopes will expect to benefit from the higher data transfer rate. With the new USB 3.0 PicoScopes, large data captures and streaming of large data sets are now much faster."

The PicoScope 3207A is a 2 channel USB oscilloscope with 250 MHz bandwidth, 1 GS/s sampling rate, 256 MS buffer memory and a built-in function generator. Basic timebase accuracy is ± 2 ppm.

Other features include digital triggering for accurate, stable waveform display, and equivalent-time sampling, which boosts the effective sampling rate to 10 GS/s for repetitive signals. The PicoScope 3207B has 512 MS buffer memory and an additional 32 Ksample arbitrary waveform generator with 100 MS/s update rate. As the scope obtains its power from the USB port, there is no need for an external power adaptor.

The oscilloscopes are supplied with the PicoScope software for Windows, which turns your computer into a powerful oscilloscope and spectrum analyzer. The software includes many advanced features such as automatic measurements, serial decoding of RS-232/UART, SPI, I²C, CAN, LIN and FlexRay data, and mask limit testing, that are only available as expensive add-ons for most competing scopes. Software updates are free of charge.

A free software development kit (SDK) is also available for those who wish to write their own data-acquisition programs. Example code in a number of languages is included.

The PicoScope 3207A and 3207B USB 3.0 oscilloscopes are priced at only US\$1813 and US\$1978 including a set of two probes.

www.picotech.com (120749-II)

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

DLP-USB1232H: USB 2.0 UART/FIFO

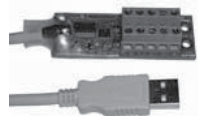
HIGH-SPEED
480Mb/s

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



DLP-IO8-G

8-Channel Data Acquisition



Only
\$29.95!

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

DLP-IOR4

4-Channel Relay Cable

DLP-TH1b

Temp/Humidity Cable

DLP-RFID1

HF RFID Reader/Writer

DLP-FPGA

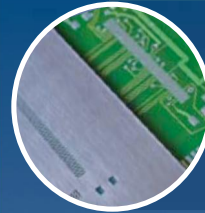
USB-to-Xilinx FPGA Module



www.dlpdesign.com



THE ORIGINAL SINCE 1994
PCB-POOL
Beta LAYOUT



FREE Stencil
with every prototype order



Embedded RFID

authenticate, track & protect
your product

www.magic-pcb.com

Call Tyler: 1 707 447 7744
sales@pcb-pool.com

PCB-POOL® is a registered trademark of

Beta
LAYOUT
create:electronics

www.pcb-pool.com

**Take out a FREE
membership to
Elektor.POST**

- The latest on electronics and information technology
- Videos, hints, tips, offers and more
- Exclusive bi-weekly project for GREEN and GOLD members only
- Elektor behind the scenes
- In your email inbox each Friday



Register today at www.elektor.com/newsletter

FPGA / CPLD Boards from JAPAN

SAVING COST-TIME with readily available FPGA boards

- Basic and simple features, single power supply operation
- Same board size and connector layout - ACM/XCM
- All stocked items are ready to be shipped immediately
- Over 100 varieties of FPGA/CPLD boards are available
- Free download technical documents before purchasing

PLCC68 series

- FPGA Module IC socket mountable
- 3.3V single power supply = Very small size (25.3 x 25.3 [mm])

XP68-03 Spartan-6 PLCC68 FPGA Module



Spartan-6 PLCC 68
XC6SLX45-2CSG324C
 16Mbit Configuration Device
 Two User LEDs
 One User Switch(Slide)
 RoHS compliant

AP68-04 Cyclone III PLCC68 FPGA Module



Cyclone III PLCC 68
EP3C25U256C8N
 16Mbit Configuration Device
 Two User LEDs
 One User Switch(Slide)
 RoHS compliant

ALTERA FPGA Board

Cyclone IV GX F484 FPGA board

ACM-024 series

Cyclone IV GX DDR2 SIF40

EP4CGX50CF23C8N
 EP4CGX75CF23C8N
 EP4CGX110CF23C8N
 EP4CGX150CF23C7N
 Credit card size (86 x 54 mm)
 RoHS compliant



Cyclone IV GX F484 FPGA board

ACM-108 series

Cyclone IV GX DDR2

EP4CGX50CF23C8N
 EP4CGX110CF23C8N
 EP4CGX150CF23C7N
 Compact size (43 x 54 mm)
 RoHS compliant



XILINX FPGA Board

Virtex-5 LXT FFG665 FPGA board

XCM-017 series

Virtex-5 SDRAM RocketIO SIF40

XC5VLX30T-1FFG665C
 XC5VLX50T-1FFG665C
 Credit card size (86 x 54 mm)
 RoHS compliant



Spartan-6 LXT FGG484 FPGA board

XCM-111 series

Spartan-6 DDR2 RocketIO

XC6SLX45T-2FGG484C
 XC6SLX75T-2FGG484C
 XC6SLX100T-2FGG484C
 XC6SLX150T-2FGG484C
 Compact size (43 x 54 mm)
 RoHS compliant



Universal Board (Type2)

ZKB-106

- One for general power(3.3V 3A max) and the Two variable outputs for Vccio(0.8V to 3.3, 3A max)
- For ACM/XCM-2 series FPGA boards
- Power Switch and LED
- Power input:DC5V/2.1[mm] Jack/ Terminal Block (option)
- Board size :156x184 [mm]
- 4 Layers PCB, Thru-hole



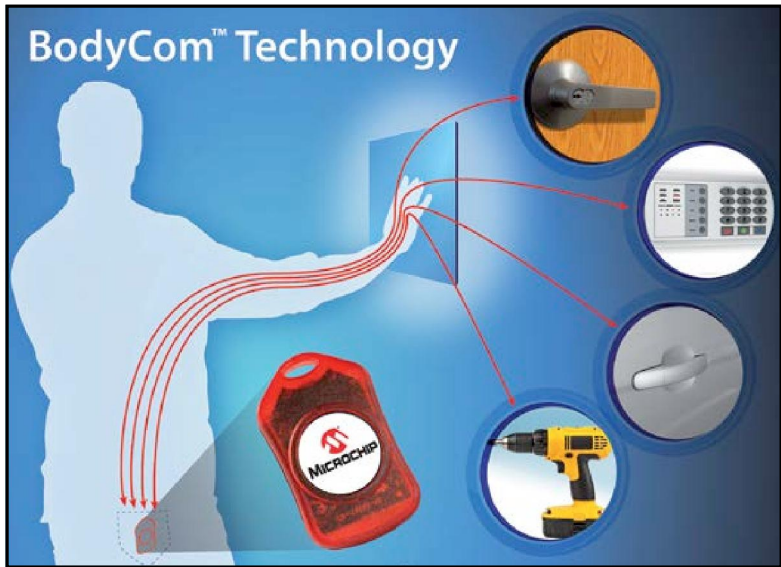
www.hdl.co.jp/EL/

HUMANDATA LTD.

E-mail: s2@hdl.co.jp

Fax: 81-72-620-2003

●Industry



Human Body as a secure, low-power communication channel

Microchip Technology Inc., announced from the Embedded World conference in Germany its BodyCom™ technology, which provides designers with the world's first framework for using the human body as a secure communication channel. Compared to existing wireless methods, BodyCom technology provides lower energy consumption, while further increasing security via bidirectional authentication. Because no RF antennas are required, BodyCom technology allows for simpler circuit-level designs and a lower bill of materials (BOM). All of this is enabled by the BodyCom Development V1.0 Framework, which is supplied through free software libraries that work on all of Microchip's more than 900 8, 16 and 32-bit PIC® microcontrollers.

BodyCom technology is activated by capacitively coupling to the human body. The system then begins communicating bidirectionally between a centralized controller and one or more wireless units. There are a broad range of applications where secure wireless communication is essential, and there is no more secure channel than the human body. This is especially true when you add bidirectional authentication that supports advanced encryption, such as KeeLoq® technology and AES. For example, BodyCom technology helps prevent the "Relay Attack" problem that is typical in automotive passive-keyless-entry security systems.

Most secure, short-range communication designs are battery powered and highly cost constrained. BodyCom technology significantly increases battery life by eliminating the need for a wireless transceiver or high-power inductive fields. It also simplifies development and lowers BOM costs by not only making antenna design unnecessary, but also by using a low-frequency framework with a common microcontroller and standard AFE frequencies (125 kHz and 8 MHz) — no external crystals are needed. And, because it complies with FCC Part 15-B for radiated emissions, BodyCom technology eliminates the cost and complexity of certification.

Additional example applications include Access Control (security systems, home/industrial door locks, pet doors); Personal Safety & Security (equipment access/disable, power tools, firearms, computer systems); Medical (patient monitoring, hospital-room access, equipment tracking); and Consumer (profile management for gaming consoles and exercise equipment). Microchip is also announcing the BodyCom Development Kit (part # DM160213). This kit is available today for \$149, and comes with a central controller unit and two wireless mobile units.

www.microchip.com/get/GA5E video: www.microchip.com/get/9TMM (120749-VII)

Uncovering Colossus: video now online

The video of Professor Brian Randell, seated in the heart of the Colossus Gallery at TNMOC, telling the story of how he uncovered the existence of Colossus in the 1970s and how the 30-year veil of secrecy surrounding the world's first electronic computer was lifted, is now online at www.youtube.com/watch?feature=player_detailpage&v=YI6pK1Z7B5Q

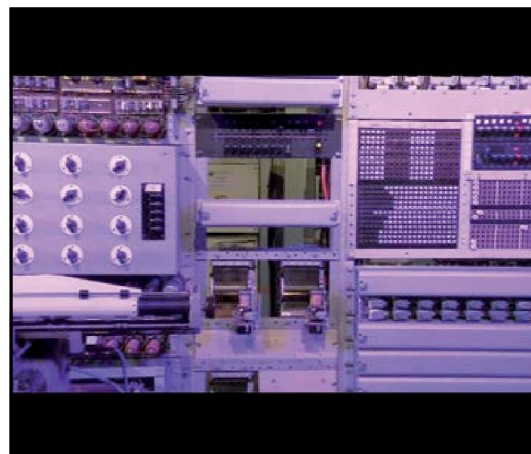
Tim Reynolds, Deputy Chair of TNMOC, said: "This video is essential viewing for anyone interested in the history of computing and we are delighted that Professor Brian Randell agreed to give his presentation in the new Colossus Gallery at TNMOC.

"The specially invited audience was captivated by Professor Randell's history of the uncovering of Colossus. It was a fascinating talk of machines, code-breaking, intrigue and politics. We are delighted to make this presentation available free for anyone who wants to learn about one of the great milestones in computing."

Margaret Sale, a TNMOC trustee and wife of the late Tony Sale who led the team that rebuilt Colossus, said: "Professor Randell inspired Tony to rebuild Colossus and now it stands in TNMOC on Bletchley Park as a wonderful celebration of Britain's codebreaking and engineering ingenuity during World War II. I thank Professor Randell from the bottom of my heart for starting that off and finding out so much for us."

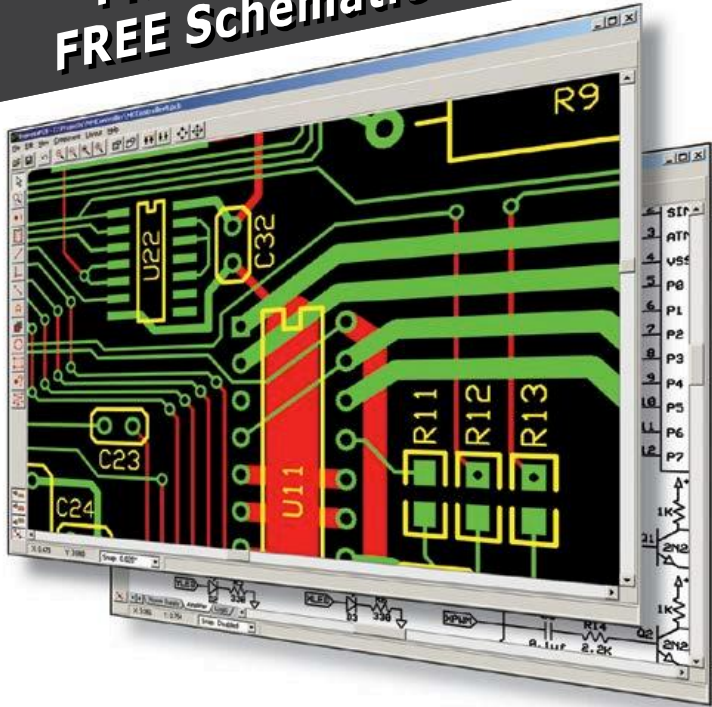
The rebuild of Colossus can be seen every day at The National Museum of Computing, located on Bletchley Park.

The National Museum of Computing, located at



\$51^{For 3} PCBs

FREE Layout Software!
FREE Schematic Software!



- 01 DOWNLOAD our free CAD software
- 02 DESIGN your two or four layer PC board
- 03 SEND us your design with just a click
- 04 RECEIVE top quality boards in just days

expresspcb.com

Bletchley Park, UK, is an independent charity housing the largest collection of functional historic computers in Europe, including a rebuilt Colossus, the world's first electronic semi-programmable computer. The Museum enables visitors to follow the development of computing from the ultra-secret pioneering efforts of the 1940s through the mainframes of the 1960s and 1970s, the rise of personal computing in the 1980s and beyond.

New working exhibits are regularly unveiled and the public can already view a rebuilt and fully operational Colossus, the restoration of the Harwell Dekatron / WITCH computer, an ICL 2966, one of the workhorse mainframes computers of the 1980s, many of the earliest desktops of the 1980s and 1990s, plus the NPL Technology of the Internet Gallery. Funders of the Museum include Bletchley Park Capital Partners, CreateOnline, Ceravision, InsightSoftware.com, Google UK, PGP Corporation, IBM, NPL, HP Labs, BCS, the Drapers' Foundation, Black Marble, and the School of Computer Science at the University of Hertfordshire.

www.tnmoc.org (130028-I)

ams: brighter, clearer images from mobile phone cameras

ams AG recently introduced a new intelligent LED driver for mobile phone cameras that maximizes the brightness of the flash without causing the phone's battery to fall below its minimum operating voltage.

The AS3649 LED driver uses an innovative "diagnostic pulse" – a burst of controlled high current lasting a few milliseconds – immediately before every flash operation. During this pulse the device measures the momentary voltage across the terminals of the phone's battery. On the basis of this measurement, it reports a value for the highest flash drive current the battery can sustain, up to a maximum of 2.5 A, without dropping below its minimum voltage and triggering the phone to reset itself during the main flash.

Drawing on advanced analog sensing technology developed by ams, the AS3649 measures the battery voltage and current with high accuracy, enabling it to precisely calibrate the optimal LED drive current under any given conditions.

Mobile phones that use the AS3649 can therefore generate the brightest possible flash light, without the need for a bulky auxiliary power source such as a super-capacitor. Users benefit from higher image quality and higher resolution. When taking pictures of fast-moving objects, a brighter flash enables the use of faster shutter speeds for sharper, clearer pictures.

The introduction of the LED driver AS3649 also allows mobile phone manufacturers to markedly reduce the engineering and software development effort involved

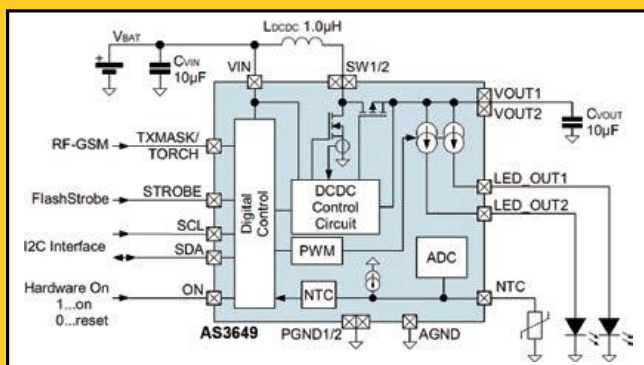
in flash LED implementation. Today, manufacturers exhaustively test the operation of each mobile phone model's LED flash system under all possible operating conditions, and at all operating voltages. The results of these tests are encoded in a software look-up table stored on the phone. Whenever the camera calls for the flash to be operated, the phone's processor must read from the look-up table an estimate for a safe drive current value.

The diagnostic pulse technique implemented by the AS3649 eliminates virtually all of this engineering effort, since it is able to measure the actual behavior of the battery at the time of use, instead of estimating it beforehand on the basis of sampled test results.

Supplying up to 2.5 A to a single LED or up to 1.25 A each to two LEDs, the AS3649 is well positioned for next-generation phone cameras using higher brightness LEDs. The device's current-source architecture provides for effective thermal management, and an on-board NTC (temperature sensor) automatically reduces the current to the LED if it exceeds a programmable temperature threshold.

The AS3649 intelligent flash LED driver is available in volume production now. A demonstration board for the AS3649 is available.

www.ams.com/Camera-Flash-LED-Driver/AS3649 (130028-II)



1-inductor, 60-V input, low IQ inverting DC/DC controller

Linear Technology Corporation's LTC3863 is a high voltage inverting DC/DC controller that uses just a single inductor to produce a negative voltage from a positive input voltage. Most low to medium power inverters utilize a coupled inductor topology, or transformer which increases the circuit size and complexity. The LTC3863 simplifies the design further with all of its interface signals being positive ground referenced. None of the LTC3863 pins are connected to a negative voltage allowing the output voltage to be limited only by the selection of external components.

The LTC3863, operating over a 3.5 V to 60 V input supply range, is designed to protect against high voltage transients, to operate continuously during automotive cold crank and cover a broad range of input sources and battery chemistries. This device helps increase the run-time in battery-powered applications with its low 70 µA quiescent current in standby mode with the output enabled in Burst Mode® operation. The LTC3863's output voltage can be set from -0.4V to -150V or lower at up to 3 A typical; making it well suited for 12 V or 24 V automotive, heavy equipment, industrial control, robotic and telecom applications.

The LTC3863 drives an external P-channel MOSFET, operates with a selectable fixed frequency between 50 kHz and 850 kHz and is synchronizable to an external clock from 75 kHz to 750 kHz. Its current-mode architecture provides easy loop compensation, fast transient response, cycle-by-cycle over current protection and excellent line regulation. Output current sensing is accomplished by measuring the voltage drop across a sense resistor. Additional features include programmable soft start or tracking, overvoltage protection, overcurrent and short-circuit protection, a power good output signal and FMEA (failure mode and effects analysis) verified for adjacent pin opens and shorts.

The LTC3863 is offered in 12-pin thermally enhanced MSOP and 3 mm x 4 mm QFN packages. The LTC3863E and LTC3863I versions operate from a -40°C to 125°C junction temperature. The LTC3863H grade is guaranteed to operate from a -40°C to 150°C operating junction temperature. The LTC3863MP is guaranteed to operate from a -55°C to 150°C operating junction temperature.

www.linear.com/product/LTC3863 (130028-III)

The Convenient All-in-One Solution for Custom-Designed Front Panels & Enclosures



ONLY \$90.24
with custom
logo engraving

You design it
to your specifications using
our FREE CAD software,
Front Panel Designer

We machine it
and ship to you a
professionally finished product,
no minimum quantity required

- Cost effective prototypes and production runs with no setup charges
- Powder-coated and anodized finishes in various colors
- Select from aluminum, acrylic or provide your own material
- Standard lead time in 5 days or express manufacturing in 3 or 1 days

**FRONT PANEL
EXPRESS**

FrontPanelExpress.com
1(800)FPE-9060

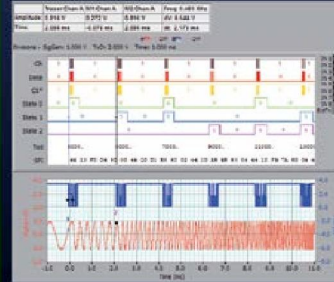
CS328A-XS



100MHz Mixed
Signal Oscilloscope
+ Signal Generator

Embedded Development

EASY AS!



- + Two mixed signal triggers
- + Protocol decoding
- + Spectrum analysis
- + Symbolic maths
- + Custom units
- + Copy & paste
- + Signal generator
- + USB or Ethernet
- + 8M samples storage
- + 100 MHz sampling
- + Dual 14 bit ADC
- + Ext Trigger, 8 Digital Inputs
- + 1.5 MSa/sec HD streaming

14 Bits 100 MSPS MSO



www.cleverscope.com

elektor
PCB Service

powered by Eurocircuits

25% Discount on new Elektor PCBs

Benefit now: Elektor PCB Service offers a permanent
90-day launch discount on new Elektor PCBs!

Check www.elektor.com/pcb for an overview of all Elektor PCBs

One-resistor tunable timer ICs

Touchstone Semiconductor announced it has added four new, high-accuracy, micropower timer ICs to its rapidly expanding family of “NanoWatt Analog™” timer integrated circuits. In addition to all other Touchstone Semiconductor ICs, these new analog timers are all in stock and ready to ship from Digi-Key, Touchstone’s authorized distributor.

These new, higher-single-supply voltage, second-generation timer ICs to the company’s very popular TS3001 and TS3002 operate over a wider supply voltage range (1.55 V to 5.25 V) and consume very little supply current (<2 μ A). To set the output frequency (or period) in these new timers, only a single resistor is needed. For the TS3003 and TS3006, the output frequency can be set from 9 kHz to 300 kHz. For the TS3004 and TS3005, an internal 3-bit counter was added that can program the output period from 3.3 μ s to 233 s (TS3004) or from 1.7 ms to 33 hrs (TS3005).

Three of the timers include a dc-voltage-controlled pulse-width modulation (PWM) output (the TS3003, the TS3004, and the TS3005). All four timers are available in space-saving, low-profile, exposed-backside-paddle TDFN packaging.

Key Specifications:

Ultra low supply current:

TS3003, TS3004, TS3006: 1.9 μ A at 25 kHz

TS3005: 1.35 μ A at 49 Hz

Supply voltage operation: 1.55 V to 5.25 V

All four timers use (1) single resistor to set F_{OUT} at 50% Duty Cycle

TS3003, TS3006 output frequency range: 9 kHz $\leq F_{OUT} \leq$ 300 kHz

Two timers with 3-pin-programmable extended ranges:

TS3004: User-programmable F_{OUT} period: 3.3 μ s $\leq t_{FOUT} \leq$ 233 s

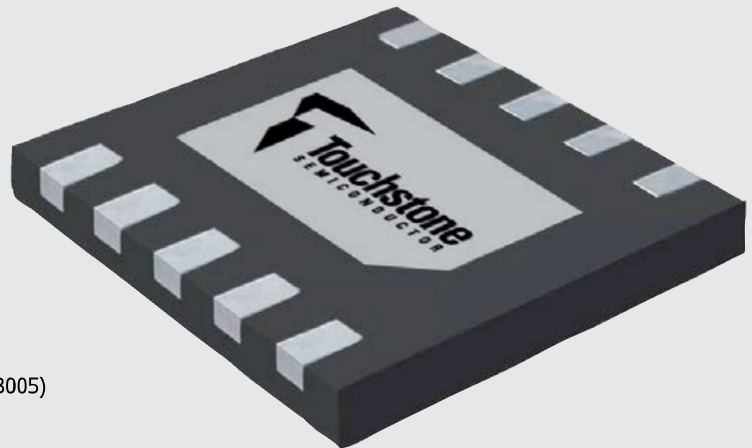
TS3005: User-programmable F_{OUT} period: 1.7 ms $\leq t_{FOUT} \leq$ 33 hr

F_{OUT} period accuracy: 3%

F_{OUT} period drift: 0.02%/°C

Separate PWM control and buffered output (TS3003, TS3004, and TS3005)

F_{OUT}/PWM_{OUT} output driver resistance: 160 Ω



All Touchstone timer ICs are fully specified over the -40°C to $+85^{\circ}\text{C}$ temperature range.

Engineers are invited to order free samples and free demo boards.

<http://touchstonesemi.com/products/timers> (130028-V)

Universal programmer

BPM Microsystems’ Model 2800 combines the unrivaled speed of Vector Engine Co-Processor® technology plus true universal device support, resulting in the fastest universal programmer in the industry.

The ultra-fast programming speed of the 2800 is attributed to BPM Microsystems’ Vector Engine Co-Processor, the same proven technology that established Flashstream® as the fastest flash-dedicated programmer. This technology uses a co-processor design to hardware-accelerate waveforms during the programming cycle. Faster speeds are

achieved through synchronous operations that eliminate the dead times so that the device under test no longer waits for the programmer. The result is programming near the theoretical limits of the silicon design — the faster the device, the faster the device is programmed

The 2800 supports all device technologies including high-density NAND Flash, NOR Flash, Serial Flash, Managed NAND Flash, EPROM, EEPROM, Flash EPROM, Microcontrollers and more with densities up to an 8Eb theoretical limit.

Model 2800 also uses BPM Microsystems’ cost-effective, efficient socket cards with receptacle-base socket option. Individual socket cards can be fully utilized and replaced without dramatically affecting programming capacity. With the receptacle-base socket option, customers can purchase only the consumable socket as needed. The design of BPM Microsystems’ socket cards increases manufacturing uptime, produces higher first-pass yield and saves replacement costs by as much as 75 percent.

www.bpmmicro.com (130028-VII)



ELEKTOR Preferred Suppliers

Tel. 1-978-281-7708

Fax 1-978-281-7706

Email ElektorUSA@smmarketing.us

bob www.decadenet.com
 basic overlay board for NTSC & PAL

- Superimpose graphics over NTSC or PAL video
- Overlay resolution: 480x240 / 480x288
- Standard fonts from 6x10 to 20x40 pixels
- Stores custom fonts and bitmap graphics
- Automatic text scroll and smooth crawl
- Fast 'TTL-232' and SPI control ports




DECADE ENGINEERING
 503-743-3194 Turner, OR, USA

Ultrasonic Distance Sensing Made EZ

www.maxbotix.com

HRUSB-MaxSonar®-EZ™

- Multi-sensor operation
- USB interface
- Easy integration
- 1 mm resolution
- MSRP \$49.95

I2CXL-MaxSonar®-EZ™

- Incredible noise immunity
- I2C interface
- 1cm resolution
- UAV's and robotics
- Automatic calibration
- Starting at \$39.95




HRXL-MaxSonar®-WR™

- IP67 rated
- Multi-sensor operation
- Great for tank and bin measuring
- Low power
- Easy to use
- MSRP \$109.95



INTEGRATED Ethernet PLCs for OEMs

Built-in Ethernet
 MODBUS TCP/IP
 Digital and Analog I/Os
 PWM/PID/Stepper Control

From \$119



TRI TRIANGLE RESEARCH INTERNATIONAL

Tel : 1 877 TRI-PLCS
 web : www.tri-plc.com/ek.htm

See what's brewing
 @ Elektor Labs 24/7

Check out
www.elektor-labs.com
 and join, share, participate!



elektor labs
 Sharing Electronics Projects

Home News Proposals In Progress Finished

Mall Navigation System

Proposals	In Progress	Finished
Active Popular	Active Popular	Active Popular
 KlikoMatic - Waste Reminder	 High-end propeller dock [120732]	 120296 USB I/O Cable

About Elektor.LABS

Elektor Labs

Create a Project

Create a new project or enter a proposal

Get help, feedback & votes from other visitors, and maybe you will get Elektorized too!

Not a member?

You want to post a project but you are

The Invisible User Interface

By
Tessel Renzenbrink
(Elektor TTF Editor)

“The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.” That’s what computer scientist Mark Weiser wrote in his famous article “The Computer for the 21st Century,” originally published in 1991.



Jelle Saldien (getting his teeth into Arduino) with colleague Jolien De Ville.

He predicted the emergence of “ubiquitous computing” — omnipresent information technology that fades into the background of the human environment. According to Weiser, this would allow people to ignore the technology and focus on new objectives. The user interface is a major stepping stone on the path toward computer invisibility.

“If you look at traditional computer interfaces, such as the keyboard and mouse, you see that people act as a cog in the process. Users have to learn to work with these interfaces because that is the only way to interact with the computer. The trend we see now is that people are looking more and more at the human aspect and how people communicate intuitively. What we are aiming for is to teach machines how to understand this human form of communication.”

The speaker is graduate engineer Jelle Saldien, a research coordinator and teacher of industrial design at the Industrial Design Center in Kortrijk, Belgium. He is also the project manager of the Tangible Intuitive Interactive Interfaces

(TIII) project [2]. The objective of the TIII project is to close the gap between the digital and physical worlds, particularly with regard to the user interface. Until recently the keyboard was the only way to get access to the digital world, but an upsurge of new interfaces has started to emerge. The breakthrough of touchscreen display on the Apple iPhone and the popularity of motion sensing game controllers, such as the Kinect and the Wii Remote, show that there is a market for them. However, interface development has primarily been a game for the big players in the technology sector. The TIII platform takes the approach that the current situation in the electronics world makes it possible for relatively small companies to participate in interface development. The platform aims to promote this by means of support and cooperation.

Usability

There is a strong trend toward usability in industrial design and in human-computer interaction. That primarily revolves around efficiency, effectiveness, safety and ease of use. “This means that people are regarded as part of the machine,” according to Saldien. “You focus on how people can perform their tasks as successfully and as efficiently as possible. This means that you look at speed, at how many mistakes they make, and how fast they can learn to click the right things in order to complete their tasks. All of that is part of usability. Ergonomics is also a major factor with products. For example, we look for the shape that best fits the hands of 90 percent of the population. People are generalized into a mathematical and psychological model with standard dimensions and reaction times. Then we try to build our systems so that everything goes as smoothly as possible with people as part of the system.”

User experience

The TIII project shifts the focus from usability to the user experience, which involves looking at all aspects of the interaction between users and their products. How is the product perceived? How do users learn how to use it? And how does their use of the product change over time? Another factor that is examined is who the user is: where does the user live, what does the user do, and what does the user want? This is a shift from efficiency to experience, and it is an emerging trend in the interface world. Saldien: “In the gaming industry they make interfaces with a focus on entertaining and surprising effects. Sometimes the interface is intentionally made more difficult to make the environment more challenging. If you constantly make the interface easier, simpler and more efficient, users may start to feel stupid. It’s like an operating system that tries to protect users to the point of annoyance by asking: Do you want to delete this file? Are you really sure?”

“The shift toward user experience is also visible in areas outside the high-tech sector. Car makers such as BMW and fashion designers such as Armani or Dolce & Gabbana are catering to user experience much more than before. They focus on aspects such as fun, surprise, image, beauty and esthetics, because these aspects have a major effect on how people experience a product or a fashion line.”

Emotions

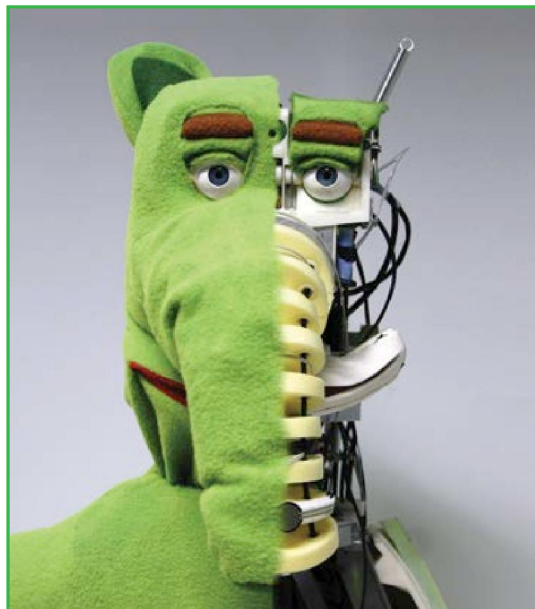
Jelle Saldien became interested in interfaces from his background in human-robot interaction. “At the Free University of Brussels we developed the social robot Probo for autistic children [3]. I have also done a lot of research on how to express emotions with a robot and how we can create social communication between people and machines. A lot of the interactions between people are non-verbal – we communicate with our tone of voice, our facial expressions, our body language and so on. That’s something you learn to understand emotionally as a person. Autistic children have difficulty with this.”

“With the social robot, we can express emotions in a controlled and gradual manner. For example, the robot can always laugh in the same way, or it can change its laugh bit by bit, very gradually. The robot expresses these emotions according to recognizable social scenarios that are explained to the children. This is a specific form of therapy, called social story telling. It is certainly not

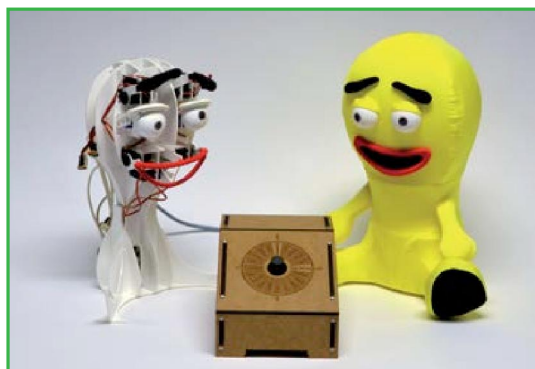
intended as a substitute for a human therapist – which I would like to emphasize, since that would not be possible – but the robot acts as an intermediary or a doll that can mimic social interactions for children.”

“By studying this, we discovered that there is still a lot to be done in the area of human-machine interaction. How people communicate with each other is very different from how people interact with machines. This can be seen with e-mail messages, where misunderstandings can arise very quickly. There we miss the social interactions, such as eye contact and emotions, which help us communicate the right meaning. This is why people came up with emoticons, such as the smiley and the sad face. Even though keyboards were never intended to be used to express emotions, we humans quickly devise ways to convey emotional content in our messages.”

“Now we are looking more at how we do this as

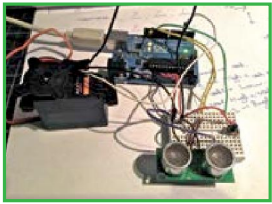


The social robot Probo.

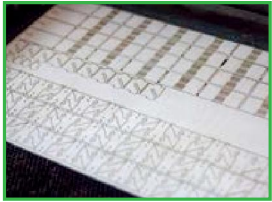


The social robot Ono [4], an open source and DIY-friendly project.

● Tech the Future



Playing with technology in the creative lab



Whist: an interactive lamp made from playing cards (inside)

Whist: an interactive lamp made from playing cards (outside).



people and how machines could pay more attention to human factors. Especially in the academic world, people are already looking at how we can let machines simulate emotions, and in the other direction, how machines can detect human emotions. For example, if my computer detects that I am angry, it can adapt the interface accordingly. People are also working on this in the car industry. It would be handy if a car could notice that the driver is drowsy and warn the driver. Or if the driver gets aggressive, the car could adapt by slowing its response to driver input.”

Limits to what is possible

Saldien thinks that if you are sufficiently creative, there aren't any technological limits to what is possible with the interface. This is partly due to the impact of the Arduino platform on interfacing. “Since this platform is so easily accessible, it allows artists, hackers and hobbyists to start tinkering and playing with technology,” he notes. “The software programs for Arduino are called sketches. You're actually sketching with software code. You copy and paste bits of code and try things out. The focus is much more on interaction — what you can get the product to do — than on software engineering. This is really prototyping with electronics. That makes it possible to actually start prototyping, shaping, testing and exploring the interface between people and the product, which is very important.”

Education

“Learning to play and be creative with technology is also what we want to emphasize in the TIII project. As an educational institution, we believe that education should be more oriented toward this. A large part of innovation consists of creativity and daring, but there is no room at all for these traits in traditional education. Conventional education is mostly about solving problems. You are presented with questions that have only one right answer. The answers are in the back of the book, but you aren't supposed to look. You also aren't allowed to talk with your fellow students, since that's called cheating — but in normal life we call it cooperation. That has to change.”

“As a teacher, I notice that freedom is very important for students. You have to give them a good environment and the right tools, so they can get started quickly. But you must also help them get in contact with companies and users quickly. The social aspect is important; you have to get users

fully involved in the development process. Talk with them, do tests with them, make something, solder something together, put it on your head, put it on and take it out on the street. Don't keep sitting in front of the computer making models and running simulations to see how it could be, because that's a dead end.”

Ubiquitous computing

“The underlying idea of ubiquitous computing is that in the near future we will have computers all around us without noticing them. This ties in with the advent of the Internet of Things (IoT). All of us are already virtually present on the Internet. Each of us has an e-mail address, an IP number and an online profile. With the IoT, every product will also have a virtual presence on the Internet. Everything will be linked together then. The interface vision is a very important aspect of this, since you can't use a keyboard to interact will all those products. If the products in your surroundings can respond to the input they obtain from your gestures or facial expressions, the boundary between the digital and physical worlds disappears.”

“This semester we will start having our students integrate sensor for electronics into fabrics. This is an important step in interfacing with electronics. Now we carry our smartphones in our pockets, but we think that in future electronics and interfaces will be incorporated in the products we already have and like, such as clothing, wristwatches, jewelry, tables, chairs, windows and mirrors.”

“With a bit of luck the future will be nicer, with fewer boxes and cables, fewer new devices, and more integration into existing products. And hopefully with more cooperation between designers, engineers and social scientists. That's a good recipe for making nice, attractive and pleasant products.”

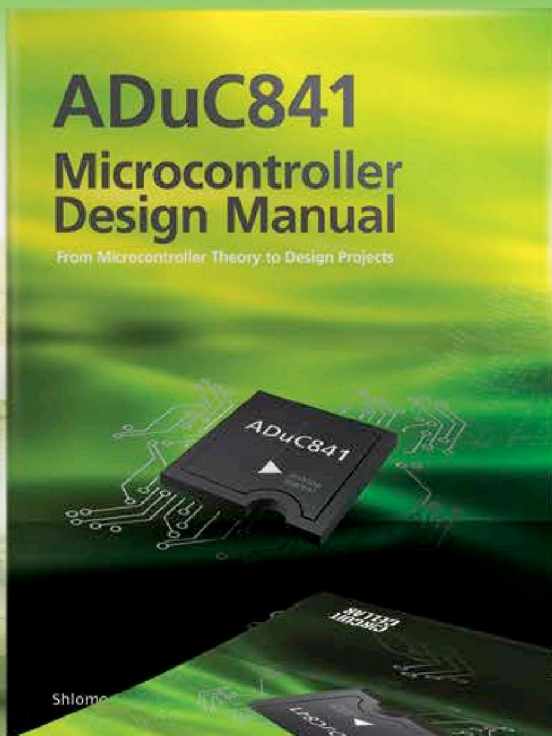
(130029-I)

Internet Links

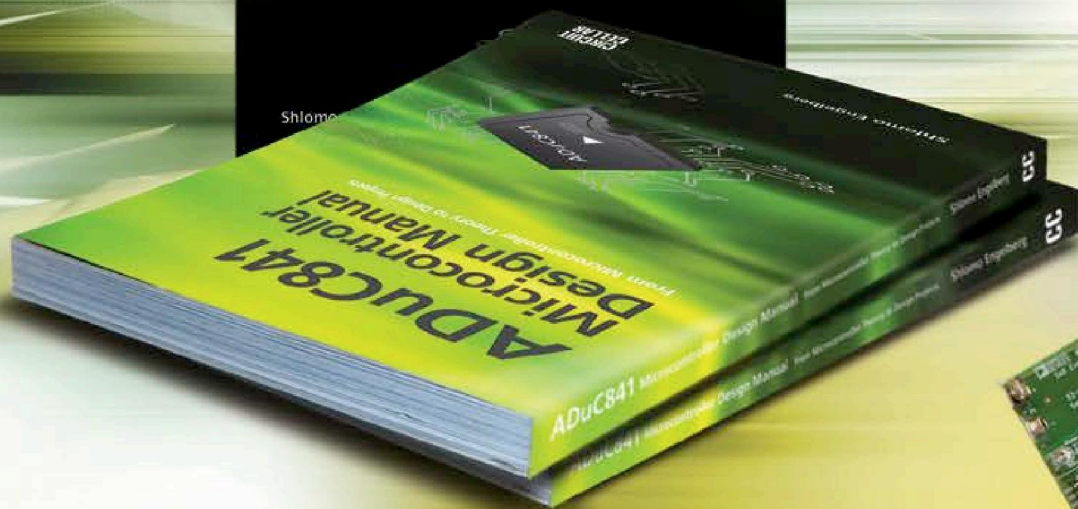
- [1] www.cse.nd.edu/~cpoellab/teaching/cse40463/weiser.pdf
- [2] www.tiii.be/
- [3] <http://probo.vub.ac.be/>
- [4] <http://io.workspace.howest.be/edubots/ono/>

ADuC841 Microcontroller Design Manual: From Microcontroller Theory to Design Projects

If you've ever wanted to design and program with the ADuC841 microcontroller, or other microcontrollers in the 8051 family, this is the book for you. With introductory and advanced labs, you'll soon master the many ways to use a microcontroller. Perfect for academics!



**Now
Just
\$35.00**



Buy it today!

www.cc-webshop.com

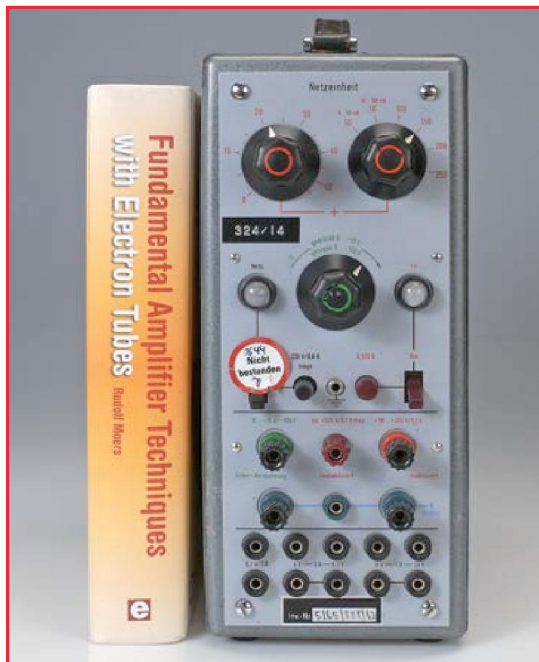
Wandel & Goltermann

NE-171 MultiVoltage Tube PSU (ca. 1963)

It's every radio repair guy's dream

By Jan Buiting
(Managing Editor)

Give me a tubed radio built between 1920 and 1965 and it's likely to have problems in the power supply department, the switches, and/or the volume control. The problems in the PSU include worn out rectifiers (tubes or selenium), overheated transformers, blown fuses, dried out electrolytic caps, spider nests, mice droppings, and bunt series or bleeder resistors. But you can't solve a problem before you have located it, which at the same time reveals the cause of the malfunction.



Just as with most algebra, problems in old radios (or any bit of electronics for that matter) get solved by replacing suspect bits by **known-good** bits and then **checking**. That's easy if a power transformer's primary measures milliohms, or if a burnt and broken power resistor is spotted or smelled out, but the real problems are with intermittent operation of a power supply, like "this radio will play just fine for a good twenty minutes, then drops silent, then starts to hum, then smoke ... and only on Wednesdays and Sunday evenings

when Aunt Mary is not around." These problems, and less grave ones too, require complete disconnection of the power supply in a radio or tube amplifier, and all of its functions taken over in one fell swoop by a **known-good** device. The good news is that silencing and effectively uncoupling the radio's built in PSU (or what's left of it) is a matter of unsoldering a few wires (typically the heater voltage and the plate voltages). If you can't locate these wires, do not attempt to repair anything 'tubed & vintage' and stay with more contemporary electronics like Rpi's and iPhones.

The bad news: you need a known-good power supply for all voltages and currents required: 50-300 VDC plate, 6.3 VAC or 4 VAC filament, -10 VDC to -100 VDC negative grid — in other words, a Multi Talent. On a positive note again, such equipment exists, and the specifications listed in the **inset** are not a dream. They pertain to the NE-171 Multi-voltage Adjustable Power Supply with Electronic Stabilization manufactured by the German firm *Wandel u. Goltermann*. I was given an NE-171 as part of an electronics laboratory clearance recently, specifically for discussing on these *Retronics* pages.

Made in Germany

If Beamers and Mercs have been benchmarks for quality "wheels" from Germany for decades, the name *Wandel & Goltermann* equals top class professional electronic test and measurement equipment from the post WW1 years right into the

IC age (i.e. before being absorbed by Wavetek in 1999). W & G equipment used to be in the top price segment — and still is as prices go on Ebay, which are said to indicate a high collectability factor.

Two manuals

Although some freestyle information on the NE-171 may be found on the Internet at the exquisite Jogi's Röhrenbude [1], nothing beats an original manual in your hands. In luck, I received two manuals with the instrument. The oldest is time stamped 27 June 1958. It covers the 'C' production batch, hence my conjecture that the original design of the NE-171 is older. This handbook is a crude production, apparently stencil copied from a typewritten original. A later version of the manual dated September 1962 by W & G, has typeset characters on the pages, which

passed") warning sticker, I wasn't in the least concerned that the unit was faulty. For four reasons, basically:

1. It's a genuine Wandel & Goltermann product.
2. It came from an environment where good care and maintenance was exercised by former and current readers of *Elektor*.
3. I have an adjustable fully isolated transformer on my workbench to give the instrument a soft start after years in storage.

I switched on, breaking the sticker. And indeed the unit worked spot on. Here's two scenarios for a No-Go sticker applied on a working instrument.

1. In week 44, 2006, someone from Approval Stickers Dept. duly connected his (her) 2012 Fluke DMM to the 50–300 V adjustable voltage output, and switched on the NE-171. Noticing a voltage surge, the test person switched



are glossy and printed rather than stenciled. This bright orange colored booklet covers production batches F, G and H. My NE-171 is no. 16432H, putting its date of manufacture in 1962, possibly 1963.

“Nicht bestanden” (in June, 2006)

Despite its age, my NE-171 is in mint condition on the outside, and just a bit dusty inside. Although the lever of the Power switch was covered by a conspicuous “Nicht bestanden” (“Not

off instantly and duly applied the red & white sticker. After the annual equipment status report was filed to the accounts department, the NE-171 was declared ‘faulty’ and put in storage, waiting to be destroyed or sent to a landfill. Fortunately, after just 8 years it was spotted by a lab worker and *Retronics* fan of a certain age. I got an email.

2. As 1. above, but: “Noticing no immediate voltage reading, ...”
- The plot is explained further on.

Compact and powerful

For sure many high-voltage adjustable power supplies exist from various manufacturers covering roughly 1950-1970, all designed for use with tube equipment, be it experimental in the lab, or for repair jobs. They're all bulky instruments though, like the Van der Heem 8619 tube PSU I discussed in the April 2007 edition of Elektor [2]. Compared to that boatanchor, the W & G NE-171 is a compact lightweight, noting of course it supplies slightly less output current and voltage on the HT adjustable output. I love the form factor of the NE-171 — it's like a thick book, and the perfect companion to many of my repair activities on tube radios.

EL156 "2 the max"

From an electronics point of view, the design

of the adjustable HT section of the NE-171 is conventional. Only a section of the schematic is shown here. We have a tube series regulator (Rö1), a regulating amplifier tube (Rö2), and an array of resistors and a pot to set the output voltage. The -85 V reference voltage furnished by a type 85A2 stabilizer tube is applied to the common rail of the lower resistor array.

The crux is not in the basic design but in the choice of the components. Where many other adjustable HT supplies have a pair of EL34 (6CA7) tubes as the series regulator, here a single EL156 is used in combination with a very 'steep' pentode type EF804S. That EL156 gets high marks throughout: cathode current: 180 mA max.; anode dissipation; 40 W max.; anode voltage: 800 V max. The device is one of the most pres-

NE-171 Specifications	
Direct Voltage 1 stabilized, floating, 6 ranges and continuous adjustment.	50 V - 300 V
Max. output current (DV2 = 0)	100 mA
Internal resistance (175 V; 20-200 mA)	≤ 2 Ω
Output voltage accuracy	≤ 1% ± 2 V
Hum (300 V; 100 mA)	approx. 0.1 mV _{rms}
Output voltage change (±10% line fluctuation; 300 V; 100 mA)	approx. ±300 mV
Direct Voltage 2 (50 mA max.) Unstabilized, floating, negative terminal common to DV1	approx. 520 V
Max. output current (DV1 = 0)	50 mA
Internal resistance	approx. 1 kΩ
Ripple voltage (50 mA)	approx. 0.5 V _{rms}
Direct Voltage 3 stabilized, floating, positive terminal on DV1 negative terminal.	0 - -10 V / 0 - -100 V (selectable)
Max. output current	
0 - - 10 V	up to short circuit (approx. 1.5 mA)
0 - - 100 V	up to short circuit (approx. 3.5 mA)
Hum (1 mA)	approx. 20 μV _{rms}
Regulation (±10% line fluctuation)	approx. ±0.1%
Alternating Voltages	
3 separate, floating, heater outputs	4 V / 6.3 V 3 A
	6.3 V 3 A
	18 V / 20 V 1 A
Tube complement	EL156, EF804S, ECC82, 85A2
AC line input	220 V, 45-60 Hz
Power consumption (fully loaded)	approx. 140 VA
Dimensions	140 x 315 x 249 mm
Weight	approx. 22 lbs (10 kg)

tigious tubes developed by the Telefunken company, and a culmination of their awesome expertise — just marvel at that datasheet [3]. The EL156 is a great audio power output tube too. The output voltage on the NE-171 is set using a combination of a 6-position range switch with 50-V increments and a pot that gives 0-50 V on top of the range value. In my case, the range switch appears stuck at the 150 V position so I can only set values between 150 and 200 V. I will look into freeing the spindle. Internally, the switch, cam and spring loaded arm appear functional.

The 50-300 V adjustable output has a separate (red) on/off switch and a resettable fuse rated at 125 mA. The later NE-171 manual warns users not to switch on the 50-300 V output until the



instrument has been on for a minute or so, on penalty of voltage surges, or no voltage at all. So now you know why the “Nicht bestanden” sticker got applied, probably by someone not too familiar with tube equipment (“Nicht verstanden?”), and definitely not by anyone on that kind lab team. No problems were encountered with the three multi-section electrolytic capacitors in the instrument. I found that three type BY179 bridge rectifiers were fitted instead of B250C150 devices as indicated by the schematic. The BY179 is conspicuously 1970s green!

Finally, I should not forget a big asset of the NE-171: all outputs float with respect to the chassis.

NE-171 double-team

The HT adjustable supplies in two NE-171s can be connected in parallel or in series, to obtain higher output currents (0-200 mA) or higher output voltages (100-600 V) respectively. The parallel configuration requires a plug at the back of the instruments to be pulled from its socket, and a special equipment linking cable to be inserted. The cable can be home made provided you can get your hands on the special 5-way plugs.

Practical use

Whenever a faulty tube radio lands on my desk, I rigorously disconnect ALL of its power supply circuitry, electrolytics included, and power the radio electronics from the NE-171 — after consulting the radio schematics of course. First, I allow the radio tubes to wake up by applying the heater voltage(s) only for 15 minutes or so. This will immediately reveal filament wiring faults and intermittent tube filaments.

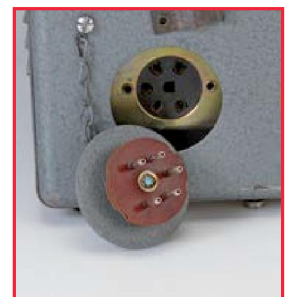
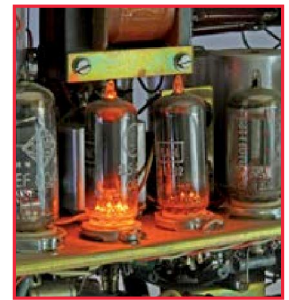
Next, the plate voltage is applied, starting carefully at 50 volts and increasing to the nominal voltage (usually 250 volts). In 8 out of 10 cases, the radio will play to an extent. Next, the electrolytics are reformed, starting at 50 volts and slowly ramping up to 300 volts, fire extinguisher ready ☺. Then I work my way back right up to the rectifier input(s). In case of doubt, a milliammeter is inserted in the central HT line.

The whole procedure can be carried out with total confidence of a known good power supply, which is a reassuring factor. The repairs on the radio’s AF, IF and RF sections finished, I’m always a bit sad when it’s time to disconnect the NE-171 and redo the connections to the radio’s own supply section — step by step, of course, and measuring voltages with one hand in my pocket.

(130030)

Internet Links

- [1] www.jogis-roehrenbude.de/, search NE-171
- [2] Adjustable high voltage supply, Retronics, *Elektor* April 2007, www.elektor.com/075036
- [3] www.hifitubes.nl/weblog/wp-content/telefunken-el156.pdf



ESTD 2004

Retronics is a monthly section covering vintage electronics including legendary Elektor designs. Contributions, suggestions and requests are welcome; please telegraph editor@elektor.com

Hexadoku Puzzle with an electronic touch

For optimum performance the human brain needs regular exercising and stimulating. Although the technical articles in this edition of Elektor should prove pretty effective in that respect, an extra challenge in the form of a puzzle is beneficial. In the puzzle below, enter the right numbers or letters A-F in the open boxes, find the solution in the gray boxes, submit online, and you automatically enter the prize draw for one of four 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 one Eurocircuits PCB voucher worth **\$140.00** and three Elektor book vouchers worth **\$60.00** each, which should encourage all Elektor readers to participate.

Participate!

Before June 1, 2013, supply your personal details and the solution (the numbers in the gray boxes) to the web form at www.elektor.com/hexadoku

Prize winners

The solution of the March 2013 Hexadoku is: **48C57**. The Eurocircuits \$140.00 voucher has been awarded to Yves Printemps (France). The Elektor \$60.00 book vouchers have been awarded to Peter Raue (Germany), Arwin J. Vosselman (Netherlands), and Torsten Clever (Germany).

Congratulations everyone!

9	4	E			5	6			1	3			2	B	A
B		0		1		E	2	4	9		D		8		6
	1		3	0		C	8	6	A		B	4			F
C			8	3							7	E			1
	9			2							0				D
	7	3		9							E		B	6	
	2			4	B	5			6	9	A				7
	5	B	C									1	E	4	
					0		4	8		2					
		5			2	3			E	7				F	
E		A	4	7	C					B	6	5	3		8
		C				B			5				0		
A		1			6					F			4		D
F	E					2			7					1	3
		D	9	E							C	A	7		
4				D	3		A	0		1	5				C

6	F	2	8	C	3	D	E	9	0	1	A	7	4	5	B
0	7	9	C	F	5	A	1	B	6	4	2	D	E	8	3
1	3	D	E	B	2	4	8	C	5	7	F	0	6	9	A
4	A	B	5	6	7	9	0	8	3	D	E	C	F	1	2
A	8	C	9	0	B	1	3	E	F	2	5	4	7	D	6
7	D	3	2	4	F	C	5	A	8	0	6	1	9	B	E
B	0	F	1	7	E	8	6	D	9	3	4	A	2	C	5
E	4	5	6	9	A	2	D	1	7	B	C	F	0	3	8
2	E	0	4	8	C	3	7	F	A	9	D	B	5	6	1
3	5	6	F	1	D	B	4	7	2	C	0	8	A	E	9
8	B	7	D	2	9	5	A	4	E	6	1	3	C	F	0
9	C	1	A	E	6	0	F	3	B	5	8	2	D	7	4
C	1	A	3	D	8	6	9	0	4	E	7	5	B	2	F
D	2	8	7	A	0	E	B	5	1	F	9	6	3	4	C
5	6	E	0	3	4	F	C	2	D	8	B	9	1	A	7
F	9	4	B	5	1	7	2	6	C	A	3	E	8	0	D

The competition is not open to employees of Elektor International Media, its business partners and/or associated publishing houses.

Credibility

By Gerard Fonte (USA)

It is most frustrating to have a great idea and not be believed. The most common reason for this is lack of credibility. That is, you are not perceived as having the skills nor the experience to be successful. And while you should seriously consider this as being true, you shouldn't blindly accept this supposition, either.

Call for Backup

Probably the easiest approach is to enlist the aid of someone who does have credibility. Experts in the field have demonstrated their experience and skills and have considerable credibility. Lawyers call "technical experts" as witnesses to present their side — often with different experts testifying to completely opposite things. "Nine out of ten doctors agree...." shout advertisers. Perhaps the most famous example is physicist Leo Szilard's recruitment of Albert Einstein to write a letter to President Roosevelt in 1939. This helped initiate the formation of the Manhattan Project (the atomic bomb). It should be noted that Einstein was a theoretical physicist and had considerable credibility in scientific circles in that area. But the Manhattan Project was experimental, or applied, physics and Einstein had little scientific credibility there. Roosevelt apparently wasn't aware of the distinction. Szilard did know the difference but was a shrewd student of human nature and understood that the perception of credibility was sometimes more important than actual credibility. (Einstein played no direct role in the development of the atomic bomb.)

Unfortunately, the expert is the one who will usually get credit for the successful project. You will only get credit for calling in the expert. This is probably not the outcome that you were hoping for.

The Deep End

Suppose you are an engineer in a company and you can clearly see that the production floor layout is incredibly inefficient. So you take upon yourself to develop a completely new production floor layout. You spend a lot of time creating the optimal design. You ask for, and are granted, a meeting with senior staff. There you present your idea for the complete re-structuring of the physical layout of the whole production floor. After the end of your presentation, the vice president of production says something like: "Bob, this is a really impressive design. I can see that you spent considerable effort working on this. But for the time being I think we'll keep things as they are. After all, they've been working well for 15 years."

Obviously you have no credibility in the development of production floors. That is not your area of expertise and trying to tackle a major project right from the start just makes things worse. No one really heard what you had to say. They were turned off before the meeting even started. Consider this, you hire a plumber to fix your hot water tank and he says that he

can completely re-design your living room for better space utilization. What would you think?

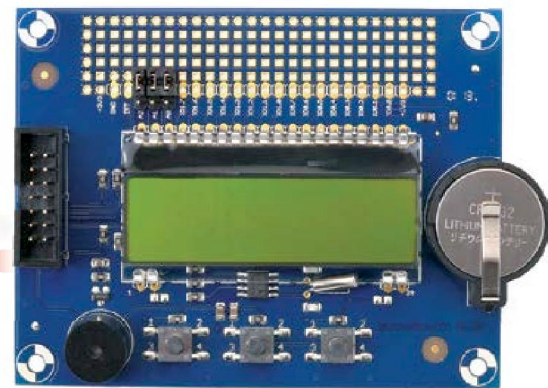
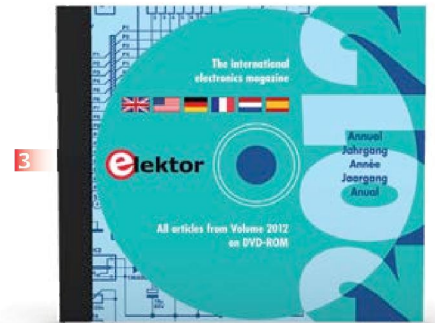
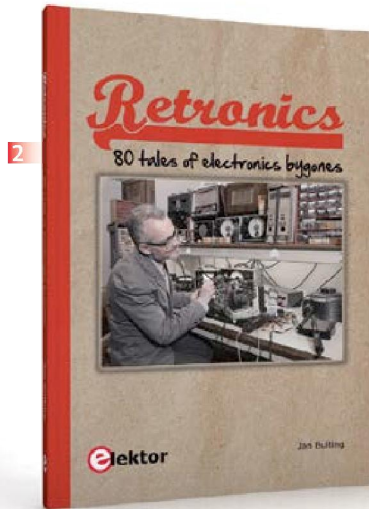
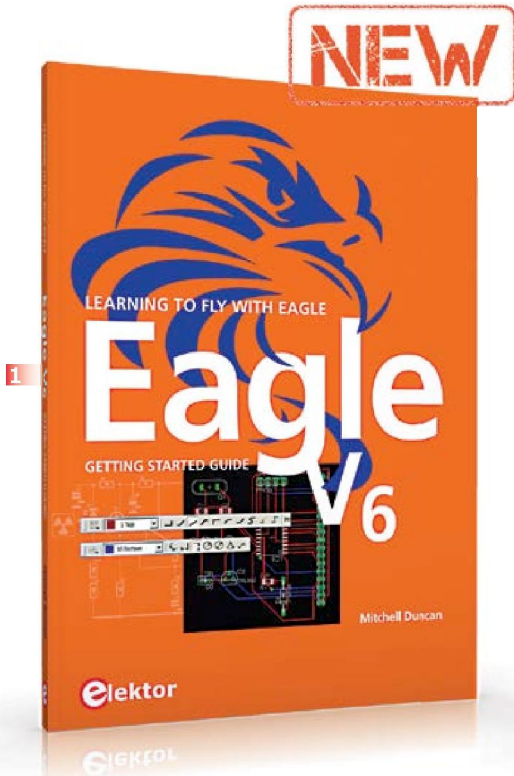
Step by Step

It's difficult to stop when you have a good idea. You want to develop it as far as you can. That is, the project becomes bigger and bigger. But, it's much easier to sell a small idea rather than a big one. And being successful with small projects gives you credibility that you can leverage to larger projects. Here's a different approach to re-designing the production floor. The first thing to do is take a course in production floor design. Then go to George (the production floor manager) with a very small change that can make a fairly large change in efficiency. But approach him deferentially and refer to your course. "George, I just finished a course in production floor design and it seems to me that if those two work-stations were swapped, the workflow would be improved. What do you think?" Assuming you are right, he would probably agree. It's a simple and obvious change. (Although he may provide you with information why that is impossible. In this case, you haven't really lost anything and have gained a better understanding of the production system.) Wait for a month after the change and visit George. If things are working out as you expected, ask him to write a note to your boss about your help. Keep up an ongoing relationship with George and make further very small suggestions for improvement. It doesn't matter if these improvements will be changed in the final grand re-design. They only have to show incremental progress. The main purpose of these changes is to develop your credibility in this area. The secondary purpose is to get George as an ally. When the time comes to present your full production floor layout, George will be listened to--he has credibility. If he agrees to your ideas they are much more likely to be accepted. In fact, you should enlist George as a co-developer of your idea. While it's nice to get the full credit, it's really not all that important. You will still be recognized as the originator of the idea. And George is very likely to be able to contribute to the design. After all, he's been working there for years and his experience is clearly meaningful. By including George you are acknowledging his importance in moving the project forward. You needed his support, so it's proper to recognize this. The critical aspect is to understand that credibility does not happen overnight. It is developed and cultivated over time.

Credibility is the spawn of success.



(130138)



Limited Time Offer for GREEN and GOLD Members!
13% DISCOUNT + FREE SHIPPING
www.elektor.com/may

Learning to fly with Eagle

1 EAGLE V6

EAGLE is a user-friendly, powerful and affordable software package for the efficient design of printed circuit boards. This book is intended for anyone who wants an introduction to the capabilities of EAGLE. The reader may be a novice at PCB design or a professional wanting to learn about EAGLE, with the intention of migrating from another CAD package. This book will quickly allow you to obtain an overview of the main modules of EAGLE, learn to use some of the basic commands in the schematic and layout editor modules, apply your knowledge of EAGLE commands to a small project, learn more about some of the advanced concepts of EAGLE and its capabilities and much more. A free version of EAGLE is available to enthusiasts for their own use. After reading this book while practicing some of the examples, and completing the projects, the reader should feel confident about taking on more challenging endeavors.

208 pages • ISBN 978-1-907920-20-2
 \$47.60

80 tales of electronics bygones

2 Retronics

Quite unintentionally a one-page story on an old Heathkit tube tester in the December 2004 edition of Elektor

magazine spawned dozens of 'Retronics' tales appearing with a monthly cadence, and attracting a steady flow of reader feedback and contributions to the series. This book is a compilation of about 80 Retronics installments published between 2004 and 2012. The stories cover vintage test equipment, prehistoric computers, long forgotten components, and Elektor blockbuster projects, all aiming to make engineers smile, sit up, object, drool, or experience a whiff of nostalgia. Owners of this book are advised to not exceed one Retronics tale per working day, preferably consumed in the evening hours under lamp light, in a comfortable chair, with a piece of vintage electronic equipment close and powered up.

193 pages • ISBN 978-1-907920-18-9
 \$40.00

A whole year of Elektor magazine on a single disk

3 DVD Elektor 2012

The year volume DVD/CD-ROMs are among the most popular items in Elektor's product range. This DVD-ROM contains all editorial articles published in Volume 2012 of the English, American, Spanish, Dutch, French and German editions of Elektor. Using the supplied Adobe Reader program, articles are presented in the same layout as originally found in the magazine. An extensive search machine is available to locate keywords in any article. With this DVD you can also produce hard copy

of PCB layouts at printer resolution, adapt PCB layouts using your favorite graphics program, zoom in / out on selected PCB areas and export circuit diagrams and illustrations to other programs.

ISBN 978-90-5381-273-0 • \$37.90

Display, buttons, real time clock and more

4 Elektor Linux Board Extension

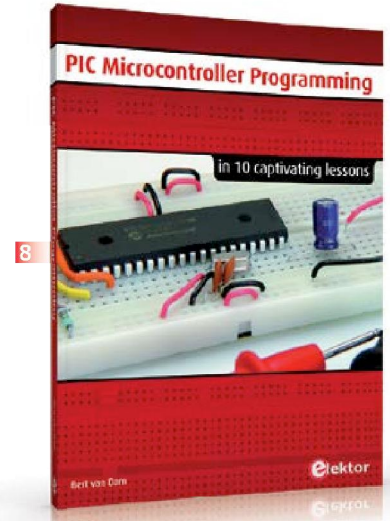
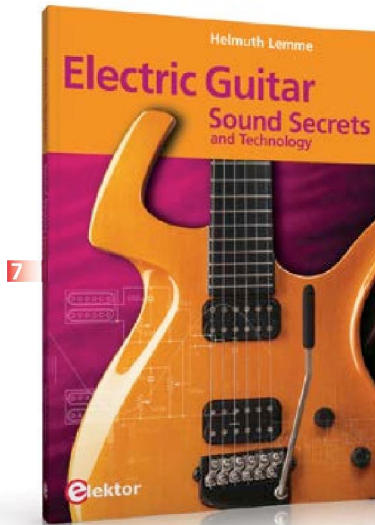
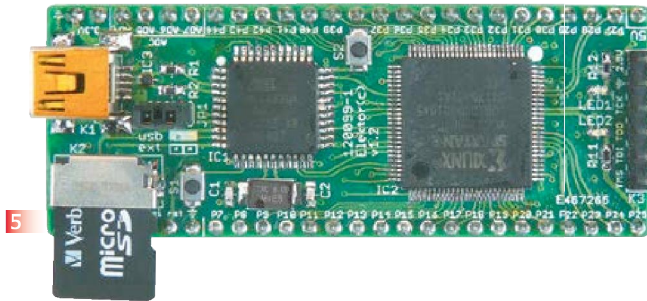
This extension board was developed to further propel our Embedded Linux series of articles and the matching GNU/Linux board. It has a display, buttons, a real time clock and 16 GPIOs. Linux devotees, switch on your solder irons. The Linux extension board includes everything needed to provide the user interface for a wide variety of projects!

Module, SMD-populated and tested board, incl. LCD1, X1, K1-K4, BZ1, BT1 for home assembly
 Art.# 120596-91 • \$50.20

Taming the Beast

5 FPGA Development Board

FPGAs are unquestionably among the most versatile but complex components in modern-day electronics. An FPGA contains a maze of gates and other circuit elements that can be used to put together your own digital circuit



on a chip. This FPGA development board (designed in the Elektor Labs) shows how easy it is for any electronics enthusiast, whether professional or amateur, to work with these programmable logic devices.

Module, ready build and tested Art.# 120099-91
See www.elektor.com/fpgaboard

LabWorX 2

6 Mastering Surface Mount Technology

This book takes you on a crash course in techniques, tips and know-how to successfully introduce surface mount technology in your workflow. Even if you are on a budget you too can jumpstart your designs with advanced fine pitch parts. Besides explaining methodology and equipment, attention is given to SMT parts technologies and soldering methods. Many practical tips and tricks are disclosed that bring surface mount technology into everyone's reach without breaking the bank. A comprehensive kit of parts comprising all SMT components, circuit boards and solder stencils is available for readers wishing to replicate three projects described in this book.

282 pages • ISBN 978-1-907920-12-7
\$47.60

Sound Secrets and Technology

7 Electric Guitar

What would today's rock and pop music be without electric lead and bass guitars? These instruments have been setting the tone for more than forty years. Their underlying sound is determined largely by their electrical components. But, how do they actually work? This book answers many questions simply, in an easily-understandable manner. For the interested musician (and others), this book unveils, in a simple and well-grounded way, what have, until now, been regarded as manufacturer secrets.

The examination explores deep within the guitar, including pickups and electrical environment, so that guitar electronics are no longer considered highly secret. With a few deft interventions, many instruments can be rendered more versatile and made to sound a lot better – in the most cost-effective manner.

287 pages • ISBN 978-1-907920-13-4
\$47.60

10 captivating lessons

8 PIC Microcontroller Programming

Using the lessons in this book you learn how to program a microcontroller. You'll be using JAL, a free

but extremely powerful programming language for PIC microcontrollers. Assuming you have absorbed all lessons you should be confident to write PIC microcontroller programs, as well as read and understand programs written by other people. You learn the function of JAL commands such as include, pin, delay, forever loop, while loop, case, exit loop, repeat until, if then, as well as the use of functions, procedures and timer- and port interrupts. You make an LED blink, build a time switch, measure a potentiometer's wiper position, produce sounds, suppress contact bounce, and control the brightness of an LED. And of course you learn to debug, meaning: how to spot and fix errors in your programs.

284 pages • ISBN 978-1-907920-17-2
\$47.60

Further information and ordering:

www.elektor.com/store

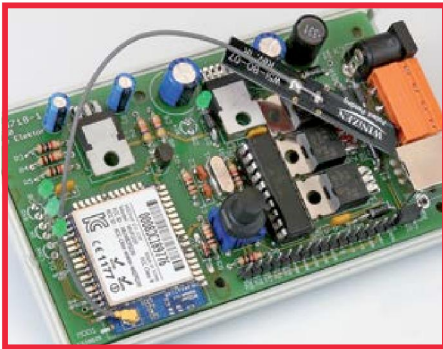
Elektor US
111 Founders Plaza, Suite 300
East Hartford, CT 06108
USA

Phone: 860.289.0800

Fax: 860.461.0450

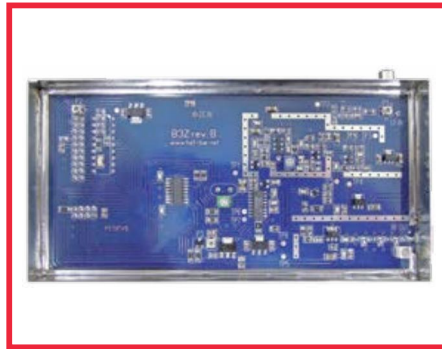
E-mail: order@elektor.com

NEXT MONTH IN ELEKTOR MAGAZINE



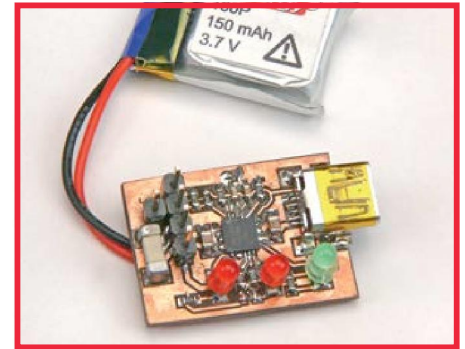
Universal Wi-Fi Controller Board

Although LED control systems exist in many shapes and sizes, surely there's room for one more from Elektor. Next month we'll publish a handy circuit that allows an RGB LED strip to be controlled through a Wi-Fi network. Those of you tired of seeing yet another RGB LED circuits will like the extra features built into the project, like a boot loader, an expansion port and protective diodes. The circuit is easily adapted for Wi-Fi based control of a garage door, a bunch of relays, a robot, a (model) car or even a train.



70 cms 130 mW Wideband FM Transmitter

This exciter/driver outputs 130 milliwatts of ultra-clean wideband FM at a user selected frequency within the 430-440 MHz section of the 70 cms amateur radio band. Legal restrictions and your national band plan allowing, this TX lets you set up high quality over-air audio links, even in duplex mode as the 70 cms band should provide enough room to accommodate several channels using directive antennas for point-to-point links.



Lithium-ion Battery Charger

Properly recycling Lithium-ion batteries and battery packs from junked equipment is not easy because the batteries are usually charged internally, i.e. by the equipment itself. Consequently there is no separate charger that can be recycled also. But then, it is not difficult to build a charging circuit for used (or new) Li-ion cells. This circuit is microcontroller free, and instead employs a special charger IC from Maxim. An array of LEDs indicates the charging progress.

Article titles and magazine contents subject to change; please check www.elektor-magazine.com

Elektor USA June 2013 edition published May 13, 2013.

See what's brewing
@ Elektor Labs 24/7

Check out
www.elektor-labs.com
and join, share, participate!

The screenshot shows the Elektor Labs website interface. At the top, the logo "elektor@labs" is displayed with the tagline "Sharing Electronics Projects". Below the logo is a search bar and a navigation menu with links for Home, News, Proposals, In Progress, and Finished. The main content area features a large banner for a "Mall Navigation System" project. Below this, there are three columns of project listings under the headings "Proposals", "In Progress", and "Finished". Each listing includes a thumbnail image, a title, and a star rating. For example, the "Proposals" column shows a "555 Class D Power Amplifier [130144-I]" with a 4-star rating. The "In Progress" column shows a "8x8 2-Color Led Matrix with ATmega328P [Arduino co...]" with a 5-star rating. The "Finished" column shows a "Switched 7805 Replacement THT" with a 4-star rating. On the right side of the page, there are several informational boxes: "About Elektor.LABS", "Create a Project" (with a link to "Create a new project or enter a proposal"), and "Not a member?" (with a link to "Click here" to send a description of your project).

ORDERING INFORMATION

To order contact customer service:

Phone: 860.289.0800

Fax: 860.461.0450

Mail: Elektor US

111 Founders Plaza, Suite 300
East Hartford, CT 06108
USA

E-mail: order@elektor.com

On-line at www.elektor.com/store

Customer service hours: 8:30 AM-4:30 PM EST Monday-Friday. Voice mail available at other times. When leaving a message please be sure to leave a daytime telephone number where we can return your call.

PLEASE NOTE: While we strive to provide the best possible information in this issue, pricing and availability are subject to change without notice. To find out about current pricing and stock, please call or email customer service.

COMPONENTS

Components for projects appearing in Elektor are usually available from certain advertisers in the magazine. If difficulties in obtaining components are suspected, a source will normally be identified in the article. Please note, however, that the source(s) given is (are) not exclusive.

PAYMENT

Orders must be prepaid. We accept checks or money orders (in US \$ drawn on a US bank only), VISA, Mastercard, Discover, and American Express credit cards. We do not accept C.O.D. orders. We also accept wire transfers. Add \$20 to cover fees charged for these transfers.

TERMS OF BUSINESS

Shipping Note: All orders will be shipped from Europe. Please allow 3-4 weeks for delivery. Shipping and handling via airmail: \$20.00 per order.

Returns

Damaged or miss-shipped goods may be returned for replacement or refund. All returns must have an RA #. Call or email customer service to receive an RA# before returning the merchandise and be sure to put the RA# on the outside of the package. Please save shipping materials for possible carrier inspection. Requests for RA# must be received 30 days from invoice.

Patents

Patent protection may exist with respect to circuits, devices, components, and items described in our books and magazines. Elektor accepts no responsibility or liability for failing to identify such patent or other protection.

Copyright

All drawing, photographs, articles, printed circuit boards, programmed integrated circuits, diskettes, and software carriers published in our books and magazines (other than in third-party advertisements) are copyrighted and may not be reproduced (or stored in any sort of retrieval system) without written permission from Elektor. Notwithstanding, printed circuit boards may be produced for private and personal use without prior permission.

Limitation of liability

Elektor shall not be liable in contract, tort, or otherwise, for any loss or damage suffered by the purchaser whatsoever or howsoever arising out of, or in connection with, the supply of goods or services by Elektor other than to supply goods as described or, at the option of Elektor, to refund the purchaser any money paid with respect to the goods.

MEMBERSHIPS (US & CANADA ONLY)

Order memberships on-line at www.elektor.com/members

All memberships begin with the current issue. Expect 3-4 weeks for receipt of the first issue. Membership renewals and change of address should be sent to:

Elektor US

P.O. Box 462228

Escondido, CA 92046

E-mail: elektor@pcspublink.com

Memberships may be paid for by check or money order (in US \$ drawn on a US bank only). We accept Mastercard, VISA, Discover and American Express credit cards.

For gift memberships, please include gift recipient's name and address as well as your own, with remittance. A gift card will be sent on request. Memberships may be cancelled at any time for a refund of all unmailed issues.

Does your membership expire soon?

Renew it on-line at www.elektor.com/members



CIRCUIT CELLAR

Subscribe Now!



Subscribe now to the leading computer applications magazine specializing in embedded systems and design!

**12 issues
per year for just**

Print OR Digital: **\$50** :: Combo (Print + Digital): **\$85**

www.circuitcellar.com/subscription