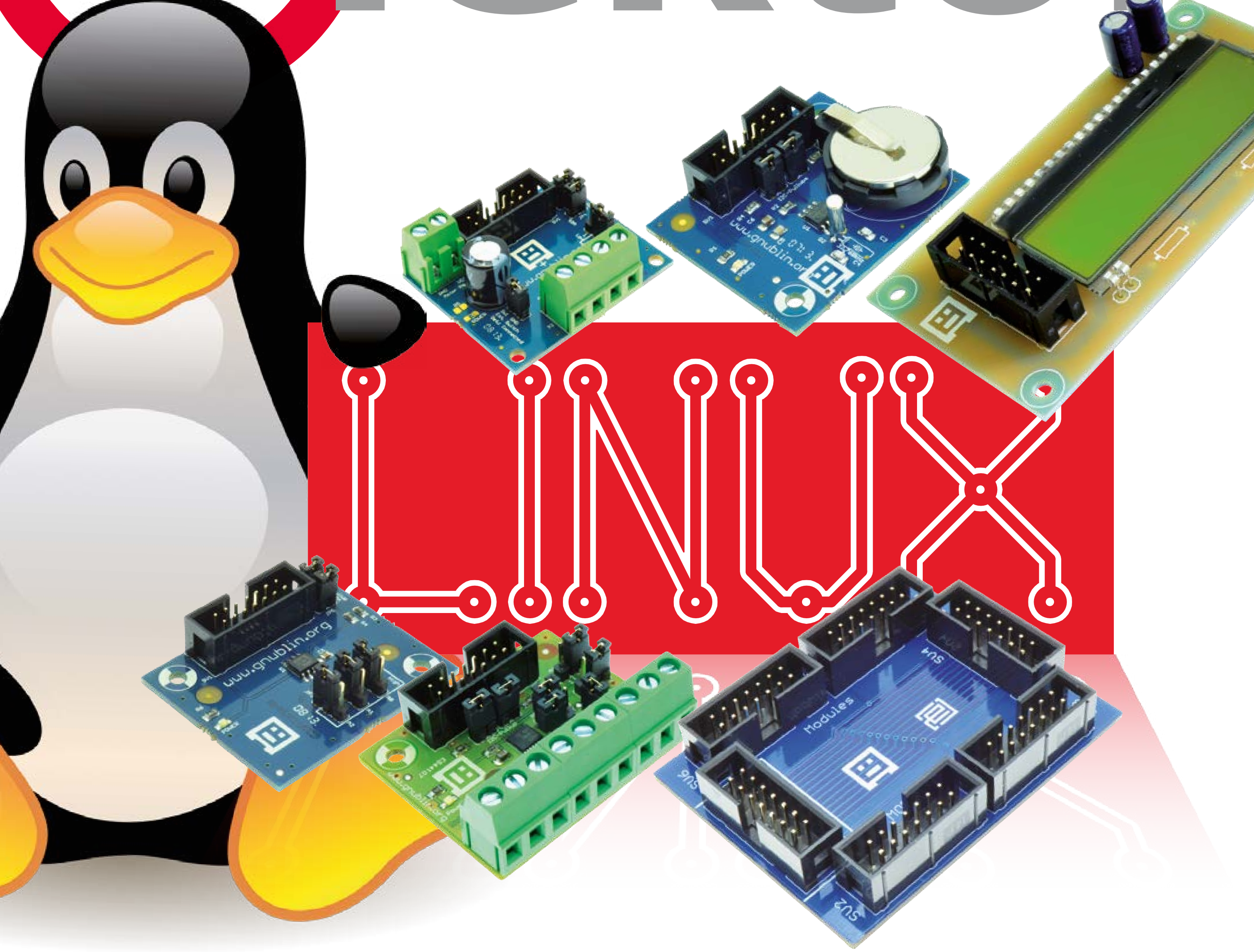
www.elektor-magazine.com

magazine

September 2013

elektor



Q-Watt 200 watts Monoblock | Manchester Coded RF Link | Gnublin Extension Boards | **Elektor Cardioscope Software** | From Basic to Python (3)

Dot-Labs Tips & Tricks **Blood pH/CO_2/O_2 Meter**

US $9.00 - Canada $10.00

09

An Exclusive Offer in your Email Inbox each Tuesday

CRAZY WEEKS @ ELEKTOR!

JUNE 21 – SEPTEMBER 21, 2013

BOOKS

KITS

SAVE UP TO 50%!

CD/DVDs

MEMBERSHIPS

MODULES

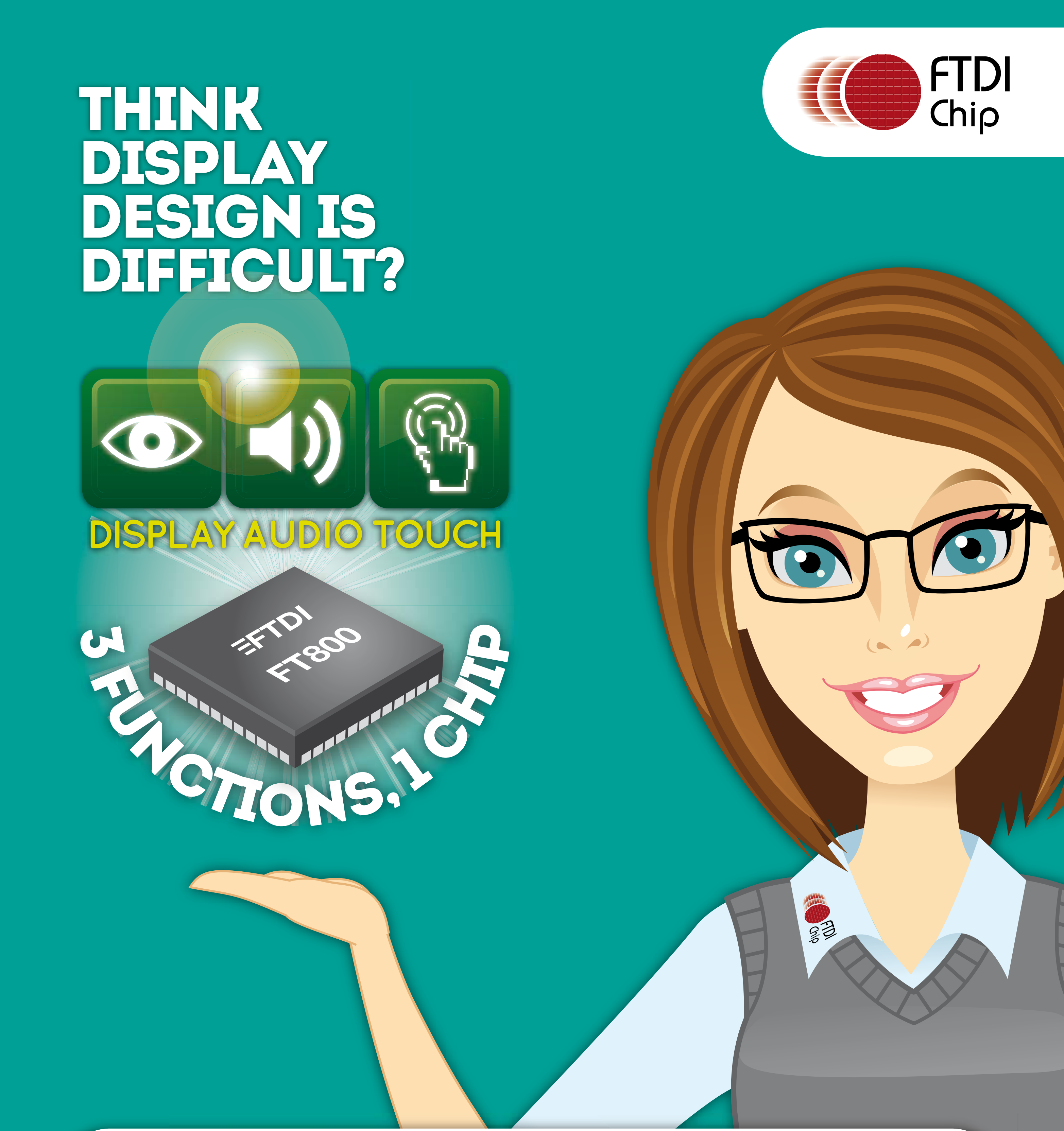
AND MORE...

TAKE ADVANTAGE OF OUR WEEKLY SUMMER DEALS!

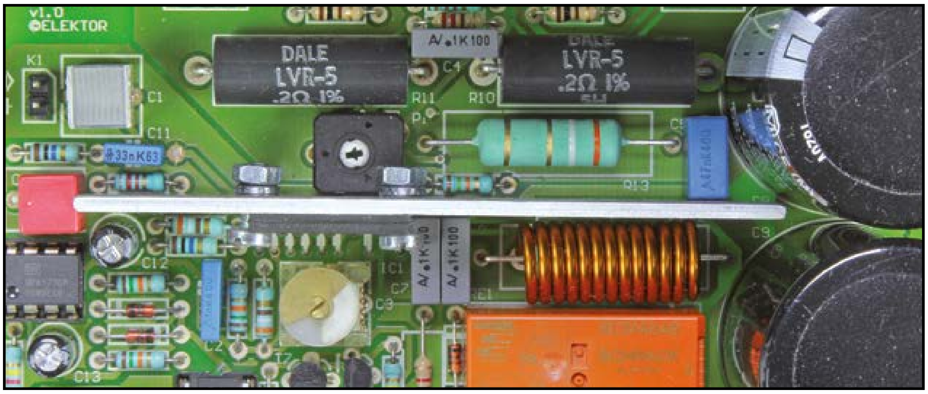
Register for our FREE Elektor.POST newsletter to receive our weekly Summer Deals*

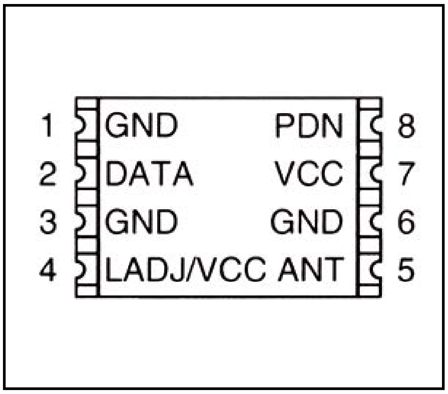
www.elektor.com/newsletter

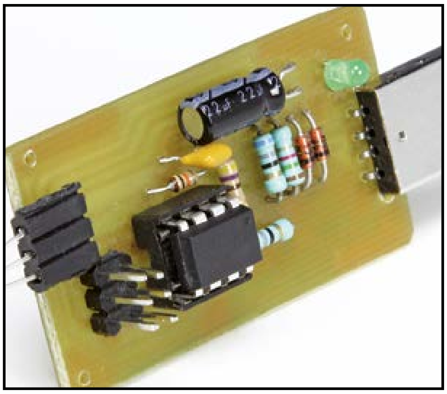
*If you already receive Elektor.POST you don't have to act. You'll automatically receive our Summer Deals.


FTDI Chip
THINK DISPLAY DESIGN IS DIFFICULT?
DISPLAY AUDIO TOUCH
FTDI
FT800
3 FUNCTIONS, 1 CHIP
FTDI Chip

Contents



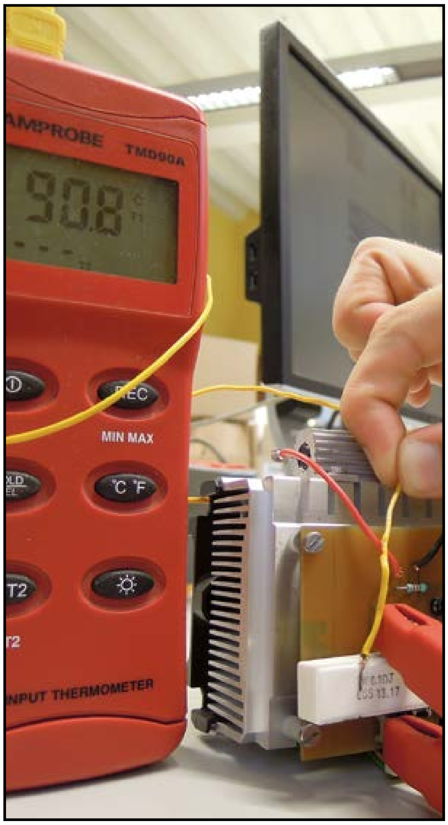


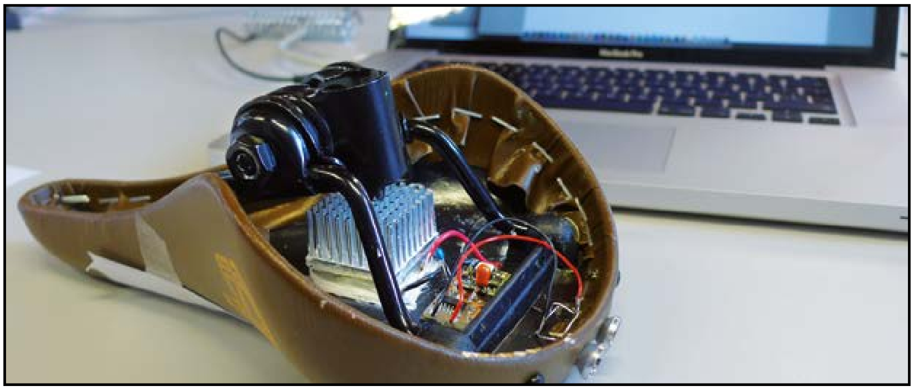


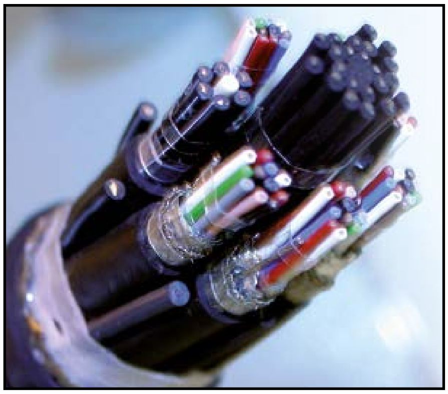
Community

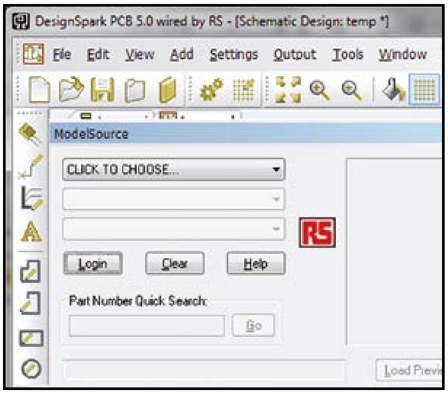
DesignSpark

Projects

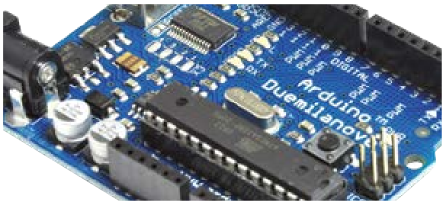







DesignSpark PCB 5.0 wired by RS - [Schematic Design: temp *]
File Edit View Add Settings Output Tools Window
ModelSource
CLICK TO CHOOSE...
RS
Login Clear Help
Part Number Quick Search:
Go
Load Previ

Labs



Industry

Tech the Future

Magazine

Volume 39, No. 441
September 2013

ISSN 1947-3753 (USA / Canada distribution)
ISSN 1757-0875 (UK / ROW distribution)
www.elektor.com

Elektor Magazine is published 10 times a year including double issues in January/February and July/August, concurrently by

Elektor International Media
111 Founders Plaza, Suite 300
East Hartford, CT 06108, USA
Phone: 1.860.289.0800
Fax: 1.860.461.0450

and

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

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

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

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

USA / Canada Advertising:
Peter Wostrel
Phone: 1.978.281.7708
E-mail: peter@smmarketing.us

UK / ROW Advertising:
Johan Dijk
Phone: +31 6 15894245
E-mail: j.dijk@elektor.com

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

Red Fruits, Italian Heroes, Penguins, and Texan Bones

Ever since Steve Wozniak assembled a 6502 based microprocessor system, and Steve Jobs literally created a market for it, we techno-inclined folks have delighted in creating and hearing product names with a disarming, if not charming, ring: apple, raspberry, acorn, penguin, Captain Zilog, KIM, Junior. I'm convinced a good number of the names given to microprocessor systems and platforms from the early days of computing have helped significantly to unnerdify the craft of programming and staring at command lines for hours on a 15-inch CRT screen propped up by pizza boxes.

The Linux community in particular has set the bar in creative product naming with every new release of "their" operating system. Where the "men in suits" simply put the next higher number behind the product name, a letter "b", or a year, the followers of Tux the Penguin came up with names you'd expect from a Tolkien book.

The main embedded platforms with clearly defined entry levels and educational aims are Raspberry Pi and Arduino, and both are covered extensively in Elektor. Magazine and Elektor.POST. However, in good engineering tradition there's more to choose from in a diversified market. Elektor's Embedded Linux board is linked this month to a range of extension boards through its "Gnublin" connector (that'll be a young goblin running *GNU's Not Unix*). The same boards, we're proud to say, also connect seamlessly to the Raspberry Pi and—as we've just discovered—to the BeagleBone Black. Have a look at the article on page 28 to see how our modules for controlling relays, displays, stepper motors, I/O devices and temperature sensors can be connected to the latest embedded microcontroller systems running Linux as the abstraction layer. Keeping abstraction and fantasy apart was never easier though as Penguin, Beagle and Gnome seem to get along very well, at least in this edition of Elektor. Let's hope no unadapt, badly named creatures appear on stage, like T-Roll or CEO2B.

More creatures and creations in this issue!

Jan Buiting, Editor-in-Chief

The Team

Editor-in-Chief:	Jan Buiting
Associate Editor:	Thijs Beckers
Publisher / President:	Hugo Van haecke
Membership Managers:	Shannon Barraclough (USA / Canada), Raoul Morreau (UK / ROW)
International Editorial Staff:	Harry Baggen, Eduardo Corral, Wisse Hettinga, Denis Meyer, Jens Nickel
Laboratory Staff:	Ton Giesberts, Luc Lemmens, Tim Uiterwijk, Clemens Valens, Jan Visser
Graphic Design & Prepress:	Giel Dols, Jeanine Opreij, Mart Schroijen
Online Manager:	Daniëlle Mertens
Managing Director:	Don Akkermans

USA
Hugo Van haecke
+1 860-289-0800
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
+35 19 14 46 55 77
j.martins@elektor.com

Portugal
João Martins
+35 19 14 46 55 77
j.martins@elektor.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

elektor

VOICE COIL

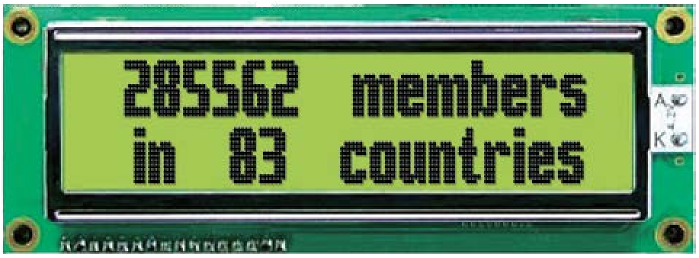
CIRCUIT CELLAR

hifi.nl MUZIEK - FILM - LIFESTYLE

audioXPRESS

techthefuture

Connects you to



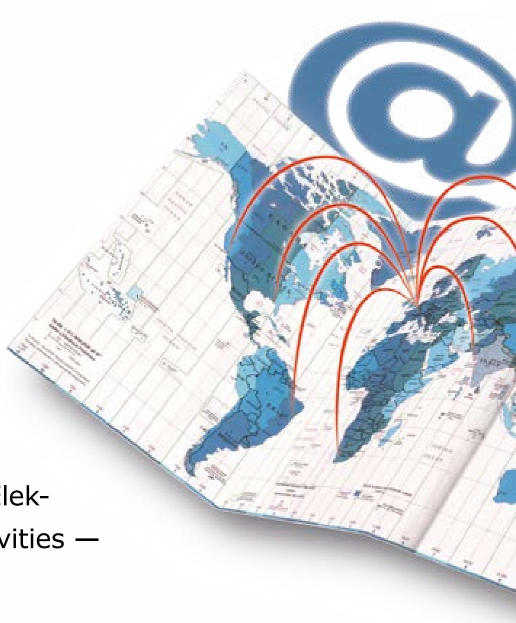
Supporting Companies

AP Circuits	www.apcircuits.com	57
Beta Layout	www.pcb-pool.com	57
Cleverscope	www.cleverscope.com	73
Crystalfontz	www.crystalfontz.com	61
DLP Design	www.dlpdesign.com	61
ExpressPCB	www.expresspcb.com	63
EzPCB	www.ezpcb.com	73
Fabstream	www.fabstream.com	27
FTDI	www.ftdichip.com	3
HuMANDATA	www.hdl.co.jp/EL/	65
International SoC Conference	www.socconference.com	84
Pololu	www.pololu.com	69
Saelig	www.saelig.com	57

Not a supporting company yet?

Contact Peter Wostrel (peter@smmarketing.us, Phone 1 978 281 7708,
to reserve your own space in Elektor Magazine, Elektor•POST or Elektor.com

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 these events and activities — for fun and business.

The CAN Man Came

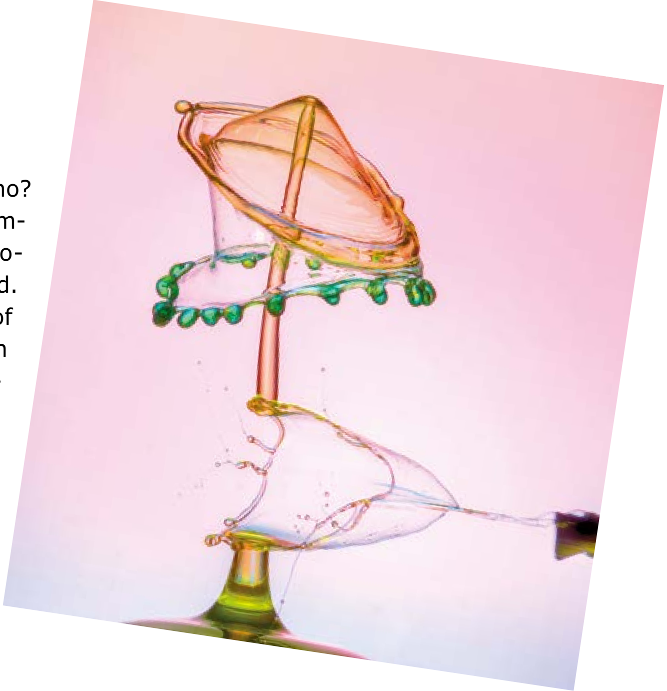
In preparation of a project on CAN, Jan Visser, member of the Elektor Labs team, invited its developer, Hugo Stiers, onto our premises to discuss some issues with the design. This visit ended up as an afternoon dedicated to the World of CAN! Hugo is an expert in the field of CAN and previous instructor with DAF Trucks. He is the type of technician who only believes what his hand can touch, so he brought in a *truck*load of boxes, PCBs, cables and notebooks to assist in field testing.

Watching Hugo and Jan work together on this project proved a great pleasure. In no time they created their own 'world of wires' where time does not seem to exist. The result of the many hours of tinkering: all is working as it should! Jan worked out the final details on the CAN project, which is published elsewhere in this issue. Thanks Hugo for visiting Elektor Labs.

Get the Picture

Are you still pondering on what you can do with Arduino? Huib Theunissen, partner of one of our sales team members, surprised us with a series of 'single shot' photographs —all timed and triggered with an Arduino board. He uses all six outputs of the board to trigger drops of differently colored liquid, a gun firing a bullet, the flash light of his camera and the shutter release. After timing these events meticulously down to a split second a beautiful photograph is taken.

Huib selected this photograph for us, 'The Speed of Life', which won him first prize in a Nikon challenge. Congrats Huib! Find more of his work on www.facebook.com/druppelfotos. Now it is your turn to think up something nifty to do with Arduino.

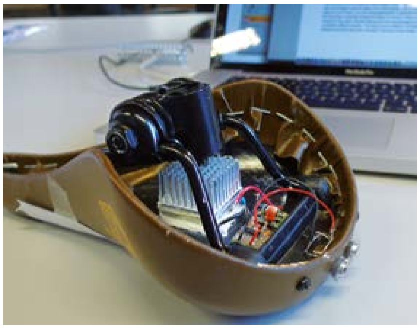


Draught Version 1.0

We are at The Kite in Oxford, UK, a local pub just near the railway station which offers some rooms upstairs for weary travelers. Chief Client Officer Johan Dijk and I had just wrapped up an enjoyable meeting with RS Components representatives and we were discussing how the draft proposal we sketched would end up in a signed contract. The lady bartender couldn't help overhearing us and offered to help us out: "I can fix you both up with a good draught (pointing at the beers on tap) and then take a picture of you holding it as proof". ...Happy to oblige!

The Ghost in the Castle

We were a bit surprised to find some people from LPI (League of Paranormal Investigators) running around in Elektor House... on a ghost hunt! They were looking for unearthly remains of French soldiers who died in the castle and the spirit of Entgen Luyten, the last 'official' witch in Holland. The story goes that she hung herself in the castle's cellar. Mart, our in-house photographer, followed the ghost hunters around and shot some photographs... and they turned out a little unusual. Using the camera flash he has been able to capture the right hunting spirit..."there, right in front of you"!



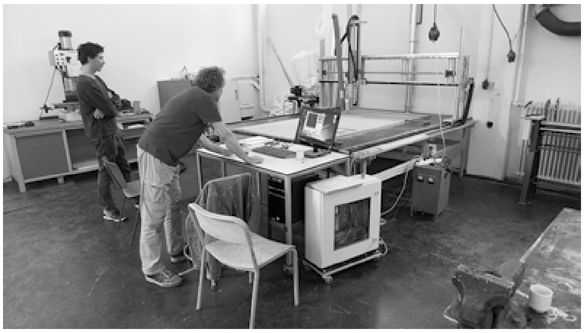
From the Pedal to the Saddle

The Dutch are proud to be a bicycle loving nation. During the day everything is fine, but towards nightfall they struggle to get the lights of their beloved bikes going, or avoid the police and a hefty fine. Most bikes are equipped with small dynamos. That way pedal power is used to generate the electricity for your lights. But there's a big chance something is broken, and you are in constant fear of being stopped by the police. Wouter Eisema from Hanze Highschool for Engineering came up with a completely different solution. The heat of your saddle (or more precisely: your butt) is converted to electricity by Peltier elements which drive LEDs mounted in the back of the saddle. Now isn't that clever!? We are planning a publication on the full project in the near future.

From DIY Plotter to JVE CNC

In 1987 Elektor published a DIY three color plotter, called Mondriaan. Among the many people building this project was Jonas Vos, a young artist. It was his first project with Elektor and a first step in making art using machines. Now a teacher at the Jan van Eyck Academy in Maastricht he decided to build his own CNC machine—a big one! This machine mills all sorts of materials in an XYZ area of 70x94x31 inch (180x240x80 cm). Students are allowed to use this machine to mill huge objects. In the picture you can see Jonas with his machine, as he is operating it.

More info on the Jan van Eyck academy: www.janvaneyck.nl



Q-Watt Audio Power Amplifier

Lots of power with low distortion

Good news for all audio enthusiasts: we are proud to present yet another fully analog circuit developed entirely in house. Despite the simple design of this audio power amp with just one pair of transistors in the output stage, *Q-Watt* can deliver over 200 quality watts into 4 ohms with exceptionally low distortion thanks to the use of a special audio driver IC.

By **Ton Giesberts**
(Elektor Labs)

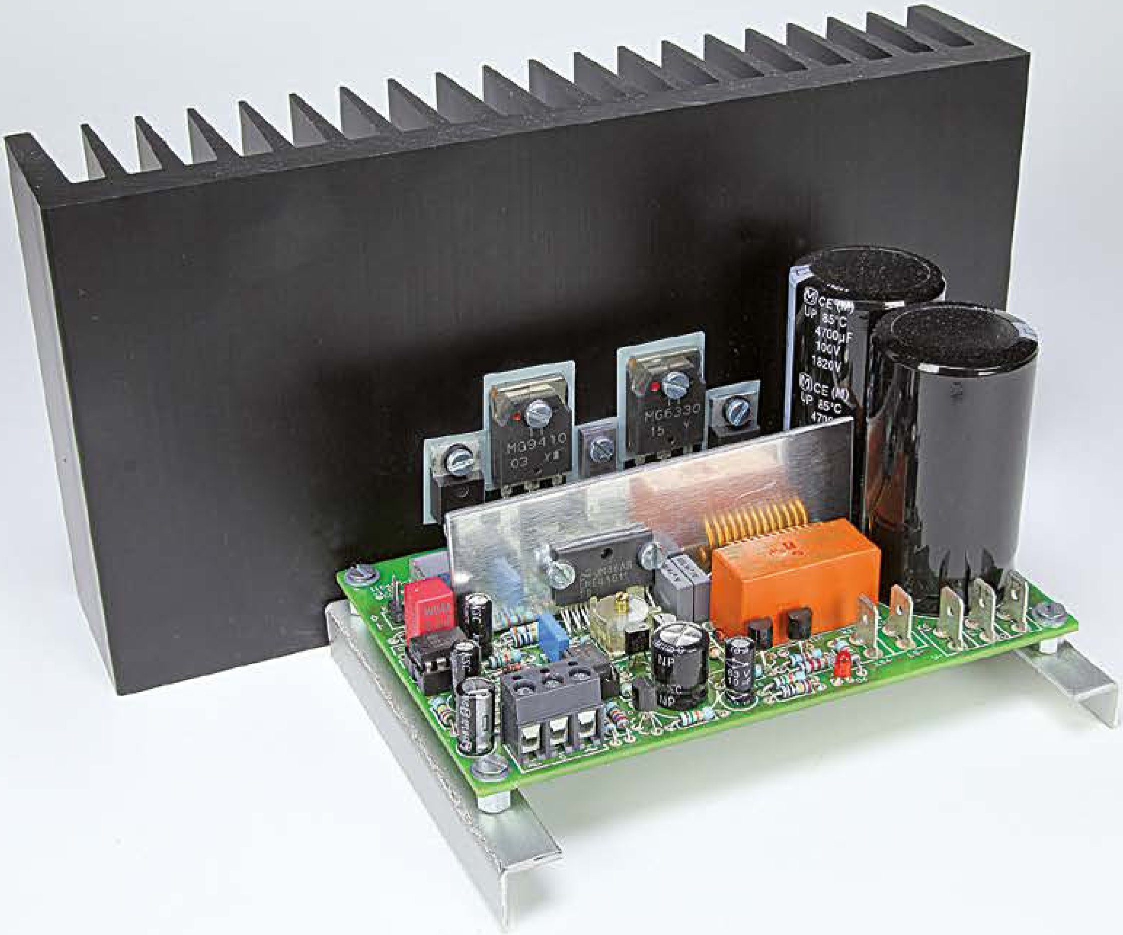
Elektor has a long history with audio power amplifiers. A couple of examples of our "golden oldies" are the Edwin, Ekwin and Crescendo amplifiers from the 1970s, which thousands of audio enthusiasts cut their teeth on. In recent years things have been quieter in this area, but that shouldn't be taken to indicate a lack of interest. Quite the opposite—many people are rediscovering the pleasure of soldering circuits themselves and putting together top-notch amplifiers with outstanding sound quality.

Since it's hardly possible to come up with something original in the realm of audio power amplifiers built with discrete components (except paralleling a few dozen NE5532 opamps...), this time we decided to take the semi-discrete route. This

Q-Watt Measured Performance

(Measured with a power supply consisting of a 500 VA power transformer with two 40 V secondaries (Nuvotem type 0500P1-2-040) and four external 10,000 µF, 100 V buffer capacitors)

• Input sensitivity:	0.88 V (137 W / 8 Ω, THD+N = 0.1%)
	0.91 V (145 W / 8 Ω, THD+N = 1%)
• Input impedance:	15 kΩ
• Continuous output power:	137 W into 8 Ω (THD+N = 0.1%)
	145 W into 8 Ω (THD+N = 1%)
	220 W into 4 Ω (THD+N = 0.1%)
	233 W into 4 Ω (THD+N = 1%)
• Peak/music power:	218 W into 8 Ω (THD+N = 10%)
(DC supply voltage ±56.8 V)	175 W (8 Ω, THD + N = 1%)
	165 W (8 Ω, THD + N = 0.1%)
	395 W (4 Ω, THD + N = 10%)
	316 W (4 Ω, THD + N = 1%)
	299 W (4 Ω, THD + N = 0.1%)
• Power bandwidth:	2.1 Hz to 125 kHz (50 W / 8 Ω)
• Slew rate:	26.7 V/µs
• Risetime:	2.4 µs
• Signal to noise ratio:	> 94 dB (linear, B = 22 Hz to 22 kHz)
(reference 1 W / 8 Ω)	> 97 dBA
• Harmonic distortion plus noise:	0.0033% (1 kHz, 1 W / 8 Ω)
(B = 80 kHz)	0.0006% (1 kHz, 50 W / 8 Ω)



	0.006% (20 kHz, 50 W / 8 Ω)
	0.0047% (1 kHz, 1 W / 4 Ω)
	0.0009% (1 kHz, 100 W / 4 Ω)
	0.009% (20 kHz, 100 W / 4 Ω)
• Intermodulation distortion:	0.002% (1 W / 8 Ω)
(50 Hz : 7 kHz = 4 : 1)	0.0009% (50 W / 8 Ω)
	0.003% (1 W / 4 Ω)
	0.0026% (100 W / 4 Ω)
• Dynamic IM distortion:	0.0033% (1 W / 8 Ω)
((3.15 kHz square wave) +	0.0022% (50 W / 8 Ω)
15 kHz sine wave:)	0.0045% (1 W / 4 Ω)
	0.0027% (100 W / 4 Ω)
• Damping factor:	560 (1 kHz / 8 Ω)
	311 (20 kHz / 8 Ω)
• Efficiency:	70.6% (8 Ω, THD+N = 0.1%)
(DC supply)	72.5% (8 Ω, THD+N = 1%)
	68.5% (4 Ω, THD+N = 0.1%)
	70.5% (4 Ω, THD+N = 1%)
• DC protection:	+0.55 V / -0.86 V
• DC output offset:	0.2 mV (max. 0.6 mV)
• Switch-on delay:	6 s

has the advantage of easy DIY construction, and it results in a very compact design. With a careful choice of components, it's possible to create a power amplifier with outstanding specs and sound quality with this approach.

Background

It all started with the *Measurement Filter for Class-D* (amplifiers) we published in our July/August edition in 2011 [1]. We developed this filter at Elektor Labs so that we could measure

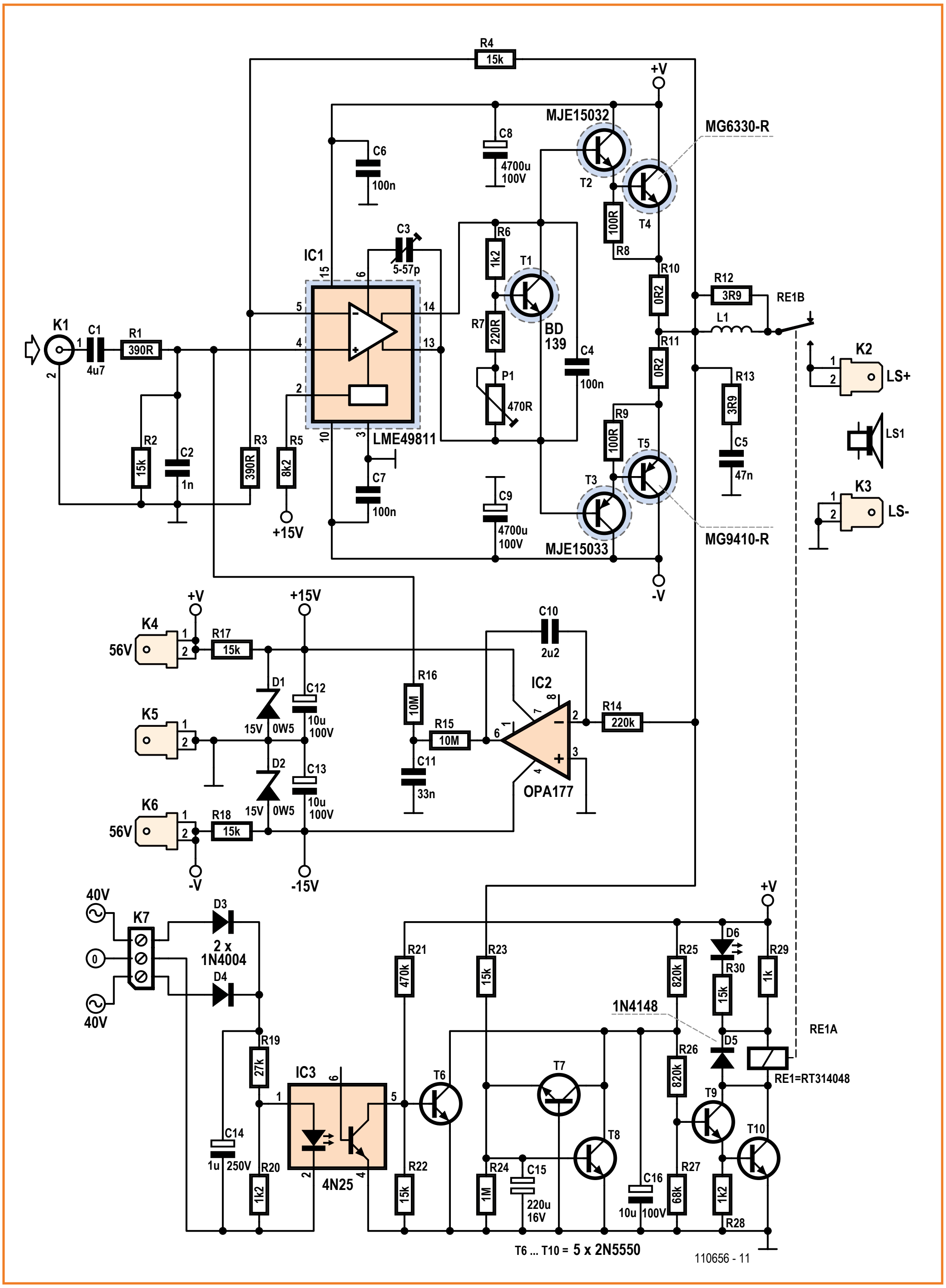


Figure 1.
Schematic of the Elektor Q-Watt audio power amplifier. Despite the simplicity of the design, the specs of this amplifier are truly excellent.

the output voltages of class-D amplifiers up to 70 V_{rms}. However, we never managed to test the filter with voltages at this level due to the lack of a suitable power amplifier. When there's a problem, you can always trust Elektor designers to come up with a solution, so they started working on the design of a fully discrete high-voltage amplifier with 23 high-voltage transistors (MJE340, MJE350, MPSA42 and MPSA92), which was intended to operate from a balanced ±110 V supply. The design turned out to be extremely complicated, and things got a bit out of hand. Although a PCB was designed for an initial prototype, we had to ask ourselves whether it was worth spending so much effort just to test a filter. The design specifications for the amplifier were truly impressive. It had to be able to deliver an output signal of 70 V_{rms} up to 20 kHz with extremely low distortion. The minimum impedance of the measurement filter is 1 kΩ, resulting in peak output current requirement of 100 mA (preferably even more).

We accordingly decided to look for a simpler alternative, such as an IC that could deliver such a high output voltage with sufficient power. Our search turned up the LME49811 from Texas Instruments. The title of the datasheet, "Audio Power Amplifier Series—High Fidelity 200 Volt Power Amplifier Input Stage with Shutdown", sounded very promising. The stated specifications were excellent, but it wasn't clear to us whether the measured performance figures on the datasheet were obtained with or without an external power stage. However, it certainly appeared to be worthwhile to develop an amplifier based on this IC.

The right transistors

The next step was to select the power transistors (T4 and T5) for the power amplifier. One of the key characteristics of power transistors for use in audio amplifiers is a large safe operating area (SOA). We ultimately found a couple of very nice devices at Semelab: the MG6330-R (NPN) and the complementary MG9410-R. These devices can handle more than 600 mA collector current at a collector–emitter voltage of 200 V. This condition occurs when the amplifier is driven to maximum output amplitude with no load. This allows the amplifier to be configured for class-AB operation with a relatively large class-A region. The DC gain of these power transistors is fairly linear up to several ampères (slightly less with the PNP version), which is a good starting point for a linear output stage. Similar requirements apply to the driver transistors (T2 and T3). The selected types—MJE15032 (NPN) and MJE15033 (PNP)—are suitable for voltages up to 250 V, and here as well the DC gain characteristic is fairly linear. The driver and output transistors have fairly high transition frequencies: 30 MHz for the MJE devices, 60 MHz for the MG6330-R and 35 MHz for the MG9410-R. The quiescent current setting is handled by an ordinary BD139.

Audio version

When one of our foreign editors saw the design, his first question was whether it could be adapted for use as a 'normal' audio amplifier. That would attract a much larger audience than a measurement amplifier for high output voltages. The answer was that it was certainly possible, and in fact it wouldn't require many changes to the original design. Some of the component values would have to be adjusted, and the supply voltage would have to be reduced. The end result is the schematic diagram shown in **Figure 1**. With a lower supply voltage (±56 V, provided by a transformer with two 40 VAC secondaries), the power amplifier can deliver a lot of power with just one pair of complementary output transistors—more than 300 watts of music power into 4 ohms.

In addition to the LME49811 (IC1), the power amplifier consists of four transistors (T2–T5), a quiescent current control network with one transistor (T1), and a few glue components.

The negative feedback network R4/R3 is dimensioned to provide an input sensitivity of approximately 1 V_{rms} for a maximum output amplitude of ±55 V with a supply voltage of ±60 V. This input voltage can easily be provided by any modern preamplifier. The resistor values are chosen to ensure that the dissipation of R4 remains just below 0.25 W at maximum output power. The values of resistors R1 and R2 are the same as those of R3 and R4 to maintain the best possible common-mode rejection at the input of the LME49811. The resulting input impedance is approximately 15 kΩ. The bandwidth of the input signal is limited at the low end by capacitor C1 (with a theoretical corner frequency of 2.2 Hz) and at the high end by C2. In addition to suppressing any HF noise that may be present, this limits the slew rate to prevent the amplifier from experiencing problems with excessively steep input signals. Only one capacitor (C3) is needed for the frequency compensation of the IC. To make it easy for users to experiment with the amplifier, we use a trimmer with PTFE (Teflon™) dielectric for this purpose (Teflon is an excellent choice for audio circuits). The PCB is also suitable for silver mica capacitors with a lead pitch of 5.9 mm. During testing a trimmer setting of one-third of the rated value (approximately 18 pF) yielded the best measurement results.

A feedback loop built around IC2 stabilizes the DC output voltage of the amplifier. It compares the output voltage to the ground reference and corrects it by injecting a very low current into the non-inverting input of the LME49811 (pin 4). The non-inverting input is used for this correction because the impedance at this input is higher than at the inverting input, whose impedance is largely dependent on the value of R3 (which is only 390 Ω). The response time is a few hundred milliseconds. We choose an OPA177 for the control amplifier on account of its outstanding DC specs (maximum bias current 1.8 nA, maximum offset 60 μV). The resulting maximum theoretical offset voltage at the output of the power amplifier is 0.6 mV, which is negligible for the connected loudspeakers. The output offset voltage of our prototype was just 0.2 mV.

The opamp in the DC correction circuit has its own ±15 V supply voltages tapped off from the main supply rails with the aid of a few resistors and Zener diodes (R17, R18, D1 and D2). The values of R17 and R18 must be adjusted if a lower supply voltage is used. In this connection an additional current of 1.5 mA drawn from the +15 V rail by pin 2 of IC1 must also be taken into account.

A Zobel network (R13-C5) is included at the output of the amplifier. It ensures that the amplifier remains stable with an inductive load or no load. Coil L1 provides additional protection against capacitive loads, and resistor R12 attenuates any oscillations or overshoots. On the PCB, R12 is fitted inside L1 to save space.

Two large buffer capacitors (4700 μF each) are also fitted on the circuit board. The selected types have low equivalent series resistance (ESR). The circuit additionally requires an external power transformer, bridge rectifier and four power supply capacitors rated at 10,000 μF, 100 V each. We chose a transformer with two 40-V secondary windings. For the prototype at Elektor Labs we used a low-cost 500 W transformer, with the result that the output voltage drops a fair amount with a relatively large load. Somewhat higher power output than stated in the specifications would be possible if you use a transformer with better voltage stability.

Supply lines

Very high peak currents occur in power amplifiers. To buffer the supply voltage, two electrolytic capacitors with low ESR are mounted on the PCB adjacent to the output transistors, in addition to the external capacitors in the power supply.

With an audio amplifier, it is essential that the supply lines to and on the board do not cause magnetic field interference, which can increase distortion by inducing currents in the negative feedback loop and other parts of the amplifier. One way to suppress this undesirable effect is to route the supply lines as close together as possible and to decouple them as close as possible to the output stage. Due to the class AB configuration of this amplifier, only unidirectional currents flow through the supply tracks on the board. Routing the positive and negative supply tracks as close together as possible causes the resulting magnetic field to be nearly sinusoidal, so it causes less distortion. With a double-sided board, these two tracks can be placed on opposite sides of the board exactly aligned to each other.

These design considerations are very important for power amplifiers with very low distortion figures. Single-point grounding is also very important in this regard. Here the ground point is located next to C5. The ground lines of the input, negative feedback, Zobel network, loudspeaker output and power supply all meet at this point.

The PCB is specifically designed for use as a monaural amplifier (monoblock). For a stereo amplifier you can simply build two of the boards and mount them in an enclosure together with the power supply. You should preferably use two separate power supplies (one for each channel).

Protection

We naturally hope that the amplifier will always work properly, but any electronic circuit can fail (especially audio power amplifiers, as we know from experience). Especially at full output power, the temperature of the output transistors can rise sharply (to above 70 °C), which can dramatically shorten the lifetime of these semiconductor devices. Our experience is that when transistors fail, they usually fail shorted. If a fuse doesn't blow somewhere in this case, a hefty DC voltage will be present at the amplifier output, which is naturally not the right way to treat your precious loudspeakers. For this reason, DC protection is actually indispensable in any audio power amplifier.

After the amplifier is switched on, it takes a few seconds for the DC voltage at the output to stabilize. As usual, the loudspeaker is connected to the output through a relay. This relay may close only when the supply voltage for the amplifier is present, and there is no DC voltage at the output of the amplifier. In this design, only the positive supply voltage is monitored by using it as the supply voltage for the protection circuitry built around T6 to T10. If there is no supply voltage, it is simply impossible to energize the relay coil.
DC protection is provided by a pair of transistors and a low-pass filter (R23/C15) with a time constant of 3.3 s. That may seem like a fairly long time, but the time required for T7 or T8 to start conducting and discharge C16 decreases with a decreasing DC voltage at the output. If there is a positive DC offset of more than 0.55 V at the output, T8 will conduct and disengage the relay via T9/T10. Transistor T7 responds similarly if there is a negative DC offset greater than 0.85 V.
In addition, both transformer secondary voltages are monitored so that the relay can be disengaged immediately when the power transformer is switched off, or a fuse blows. To avoid creating a ground loop, the secondary transformer voltages are monitored using optocoupler IC3, which feeds its output signal to T6 in the protection circuit. Diodes D3 and D4 in combination with capacitor C14 act as a full-wave rectifier for the LED in the optocoupler. The voltage divider R4/R3 is dimensioned so that the LED goes dark immediately if either of the transformer voltages drops out.
Capacitor C16 in combination with resistors R25 and R26 determines the time delay for engaging the relay after the supply voltage is switched on (approximately 6 seconds).
The relay used here has a rated coil voltage of 48 V. It is connected to the 56 V supply rail via a 1 kΩ series resistor (R29). If you have trouble finding a 48 V relay, you can use a 24 V relay instead. In that case R29 must be a 1 W type with a value of 2.2 kΩ.

The protection circuit is dimensioned for a supply voltage of ±56 V. If you use a lower supply voltage, some of the resistor values will have to be

Cooling

Adequate cooling must be provided for the driver transistors, output transistors and IC1. For the IC this consists of a piece of 2-mm aluminum sheet metal measuring 2.5 x 8 mm, which is mounted on the IC with a pair of screws and nuts. This heatsink is sufficient to handle the 2 W or so dissipated by the IC with a supply voltage of approximately ±56 V.
Choosing the heatsink for the output transistors involves a tradeoff between the size of the heatsink and the estimated average output power of the amplifier. A very large heatsink or forced air cooling would be necessary to handle continuous full output power, but this occurs very rarely in practice. We therefore decided to look for a heatsink that is big enough to handle the full output power for a short while (several minutes). We found a good match in a heatsink from Fischer Elektronik, Germany. It's not exactly small, but there's no getting around a low thermal resistance if you want to avoid overheating with high output power. The selected heatsink has a height of 10 cm and a thermal resistance of 0.7 K/W. To give you an idea of what this means, with a regulated supply voltage of ±56.8 V the amplifier can deliver nearly 300 W into a 4 Ω load with 0.1% distortion. With an efficiency of 68.5%, this means that it must dissipate about 137 W. With a continuous sine-wave signal, at full output power the temperature would rise to more than 90 degrees above the ambient temperature. The emitter resistors R10 and R11 (5 W types) would also be on the ragged edge at this point. However, as already mentioned this will never happen in normal use with music signals. By the way, there is virtually no manufacturer of audio amplifiers that dimensions their heatsinks for continuous full power.

COMPONENT LIST

Resistors

(5%, 0.25W, unless otherwise stated)
R1,R3 = 390Ω
R2,R4,R17,R18,R22,R23,R30 = 15kΩ
R5 = 8.2kΩ
R6,R20,R28 = 1.2kΩ
R7 = 220Ω
R8,R9 = 100Ω
R10,R11 = 0.2Ω 1% 5W, low inductance (Vishay Dale LVR05R2000FE73)
R12,R13 = 3.9Ω 5% 5W
R14 = 220kΩ
R15,R16 = 10MΩ
R19 = 27kΩ
R21 = 470kΩ
R24 = 1MΩ
R25,R26 = 820kΩ
R27 = 68kΩ
R29 = 1kΩ
P1 = 470Ω trimpot, horizontal

Capacitors

C1 = 4.7µF 63V, MKT (metal/polyester), 5mm or 7.5mm pitch
C2 = 1 nF/400 V, MKT (metal/polyester), 5mm or 7.5mm pitch
C3 = trimmer 5-57pF 250V, horizontal (Vishay BC-components BFC280908003)
C4,C6,C7 = 100nF 100V, 5mm or 7.5mm pitch
C5 = 47nF 400V, 5mm or 7.5mm pitch
C8,C9 = 4700µF 100V, 10mm pitch, snap-in, 30mm diam. (Panasonic ECOS2AP472DA)
C10 = 2.2µF 63V, 5mm or 7.5mm pitch
C11 = 33nF 63V, 5mm or 7.5mm pitch
C12,C13,C16 = 10µF 100V, 2.5mm pitch, 6.3mm diam.
C14 = 1µF 250V, 2.5mm pitch, 6.3mm diam.
C15 = 220µF 16V bipolar, 5mm pitch, 10mm diam.

Inductor

L1 = 450nH: 13 turns 14AWG (1.5 mm) enameled copper wire, 7mm inside diam.

Semiconductors

D1,D2 = 15V 0.5W zener diode
D3,D4 = 1N4004
D5 = 1N4148
D6 = LED, red, 3mm
T1 = BD139
T2 = MJE15032
T3 = MJE15033
T4 = MG6330-R
T5 = MG9410-R
T6–T10 = 2N5550
IC1 = LME49811TB/NOPB
IC2 = OPA177GPG4
IC3 = 4N25

Miscellaneous

K1 = 2-pin pinheader, 0.1’’ pitch
K2–K6 = Faston (blade) plug, PCB mount, 0.2’’ pitch
K7 = 3-way PCB screw terminal block, 5mm pitch
RE1 = relay, PCB mount, SPCO, 16A, 48V coil, 5.52kΩ (TE Connectivity/Schrack type RT314048)
TO-220 isolating washer for T1, T2, T3, Kapton MT film, 0.15mm, 6kV
TO-3P isolating washer for T4,T5, Kapton MT film, 0.15mm, 6kV
TO-220 3-mm isolating bush for T2,T3
Heatsink, 0.7K/W (e.g. Fischer type SK 47/100 SA)
Heatsink for IC1, dim. 25x 80 mm, 2 mm thick aluminum
PCB # 110656-1, see www.elektor.com/110656

Power Supply (for one amplifier)

Power transformer: sec. 2x40V, 500VA (e.g. Nuvotem 0500P1-2-040 for 230 VAC mains)
Bridge rectifier: 200V, 35A (e.g. GBPC3502) (Fairchild)
Four 10,000µF, 100V electrolytic capacitors (2 in parallel on each supply rail)

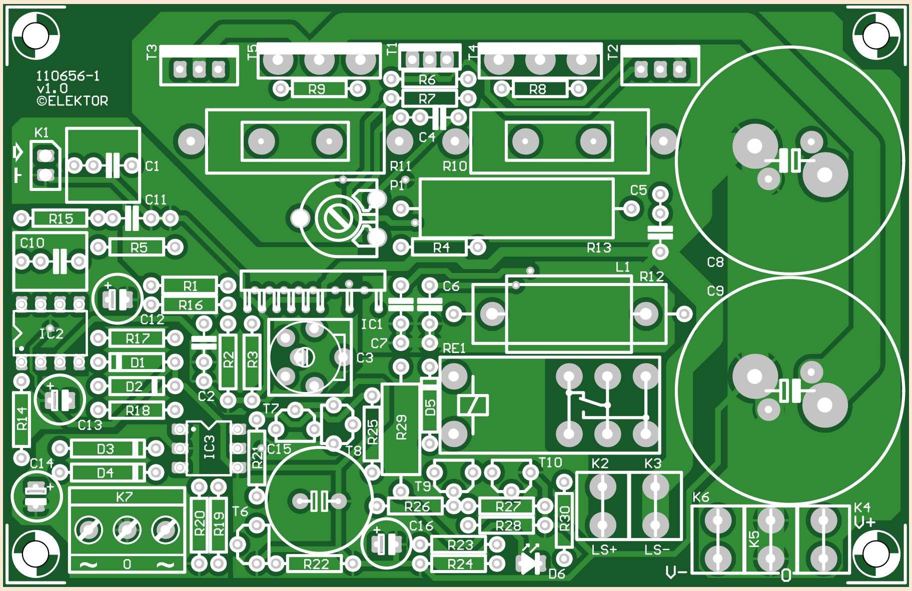


Figure 2.
The PCB holds the entire power amplifier, including buffer capacitors and protection circuitry.

adjusted. This also applies to the resistors in the negative feedback network if you want to maintain an input sensitivity of around 1 V. Bear in mind that the gain factor of the LME49811 must be at least 20 (26 dB).

Construction

Figure 2 shows the circuit board layout designed for this amplifier. As promised by the title of this article, everything has been kept nice and compact.

Building the board is certainly not difficult, but there are a few points that require attention. Most of the components can be soldered directly on the board, with the exception of T1–T5, IC1 and the supply capacitors C8 and C9. Blade connectors (Faston 6.3 x 0.8 mm) are soldered to the PCB for connecting the supply voltage and the loudspeaker.

Coil L1 consists of 13 turns of AWG #14 (approx. 1.5 mm) enameled copper wire wound on a 7-mm drill bit. Leave the ends long enough to allow the coil to be mounted a small distance above the board. The coil leads must be bent to point directly away from the middle of the coil. Place resistor R12 inside coil L1 and bend its leads so they line up with the corresponding holes in the PCB. Fit these two components on the board at the same time, and when soldering the leads ensure that the coil is raised a bit above the board surface and the resistor is properly located in the middle of the coil (see **Figure 3**).

Before going any further, you need to know what enclosure you will be using, so that you can determine how to secure the heatsink and the circuit board in the enclosure. The most convenient solution is to attach two aluminum brackets to the heatsink and use them to secure the circuit board. This way you can still perform tasks on the PCB after the transistors have been mounted on the heatsink.

The circuit board must be mounted on the heatsink so that the leads of the transistors are as close as possible to the corresponding pads on the board. Using needle-nose pliers, form the leads of T1–T5 into shallow S shapes so that the leads project slightly and fit into the holes in the PCB without any mechanical stress. Make the first bend as close as possible to the package. Never bend the leads directly; always place a small metal plate against the pins next to the package to prevent the formation of microcracks in the package. Make the second bend at the level of the holes in the PCB. **Figure 4** shows what it should look like. The insulator pads for the transistors can be temporarily placed between the transistors and the heatsink to determine the exact position of the second bend. Actually this is not critical unless you use ceramic insulator pads. Secure the transistors firmly to the heatsink (with the insulator pads in place) before soldering the leads to the board.

Next comes IC1. Start by attaching a heatsink plate, consisting of a piece of 2-mm aluminum sheet metal measuring 2.5 x 8 mm, to the IC with a pair of screws and nuts. Mount the heatsink so that it is a bit above the board when the IC is fitted, to avoid contact with R1, R4 and R5. **Caution:**

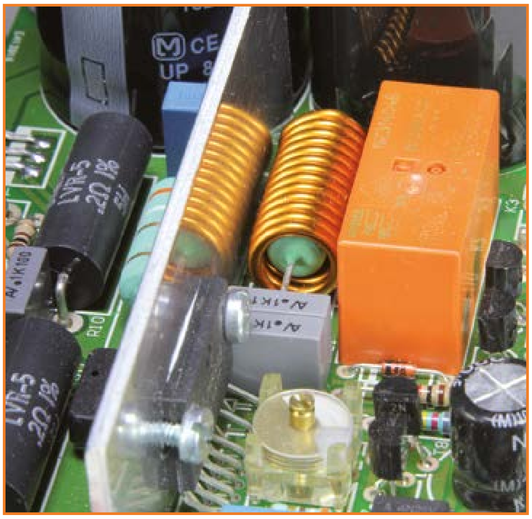


Figure 3.
Detail of output coil L1 with power resistor R12 fitted co-axially.

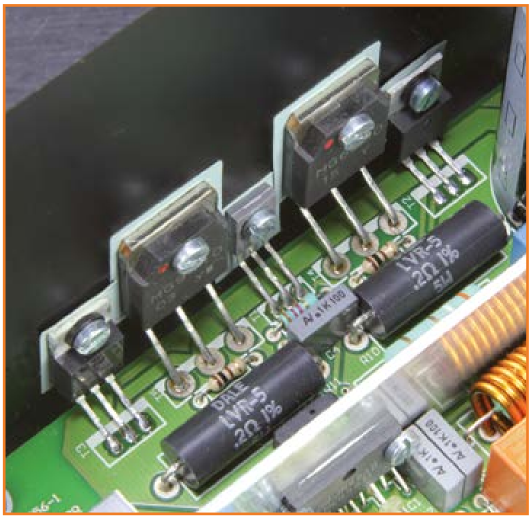


Figure 4.
The leads of all transistors mounted on the heatsink are formed with two bends so they fit precisely in the corresponding holes in the PCB without mechanical stress.

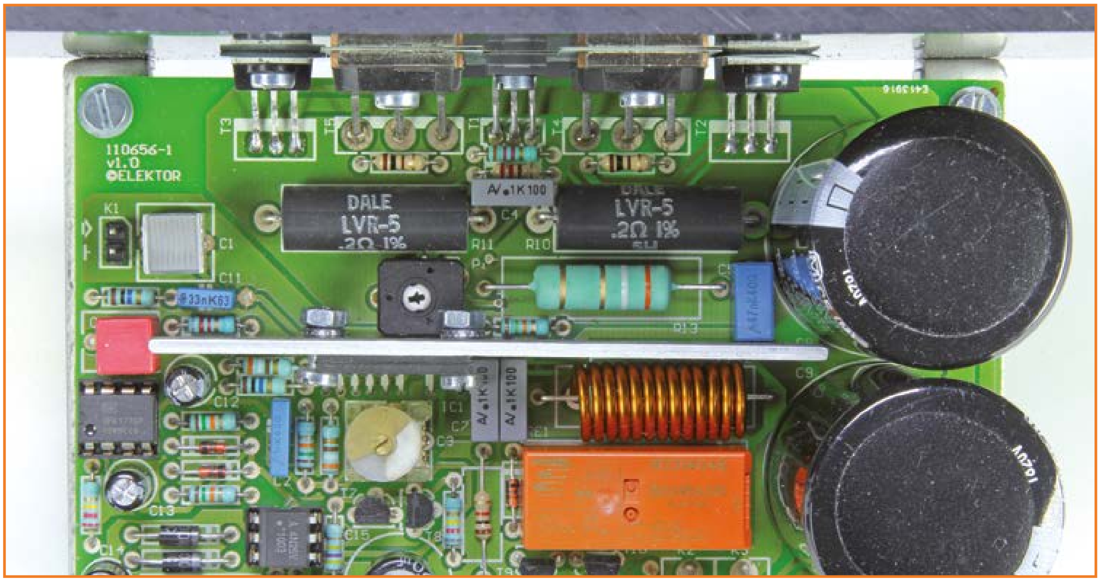


Figure 5.
There's just enough space on the circuit board for the heatsink that has to be mounted on IC1.

the metallic rear surface of the IC is connected to the negative supply voltage. This means that if you do not use insulated mounting hardware, the heatsink will be at the negative supply voltage. For safety, we recommend using insulating mounting hardware here. Then solder the IC to the PCB. Just enough space for the heatsink has been kept free on the board (see **Figure 5**). Bend L1 slightly away from the heatsink.

The final task is to mount the two large buffer capacitors C8 and C9. That way they don't get in the way of earlier activities.

Q-Watt on test run

Before you connect your *Q-Watt* amplifier directly to the power supply, you have to set the quiescent current of the output stage. To do this, first connect two 47 Ω, 5 W power resistors in series with the positive and negative supply voltage terminals. This prevents damage to the amplifier circuit if something is wrong, such as a short somewhere. The worst that can happen is that the two power resistors go up in smoke. Another option is to use a regulated power supply with current limiting, but most of us don't have this sort of supply available for ±56 V. Connect an ammeter in series with the positive supply line. Before switching on the supply voltage, turn P1 fully counterclockwise, and remember to connect the secondary windings of the transformer to terminal block K7 on the circuit board. After the power is switched on, the current through the positive supply line should be approximately 30 mA when the output relay is engaged. Slowly turn P1 to the right (clockwise) until the current increases by 30 mA (60 mA total). This relatively low quiescent current is more than adequate. The quiescent current will rise slightly as the heatsink temperature increases. However, it will normally remain below 90 mA. At very high output power levels, the junction temperatures of the two output transistors will rise much faster than the temperature of the heatsink, so the quiescent current transistor will not be able to fully track the change. This causes the quiescent current to rise briefly to several hundred milli-amps, but it declines quickly when the temperature drops again. That's actually a nice extra feature with this amplifier, since the class A range of the power amplifier effectively increases with the output power level.

We hope you have a lot of fun building this compact power amplifier, and a lot of listening pleasure afterwards.

(110656-I)

More details about this power amplifier are available at

www.elektor-projects.com/project/110656-simple-audio-power-amplifier.13247.html

Internet Reference

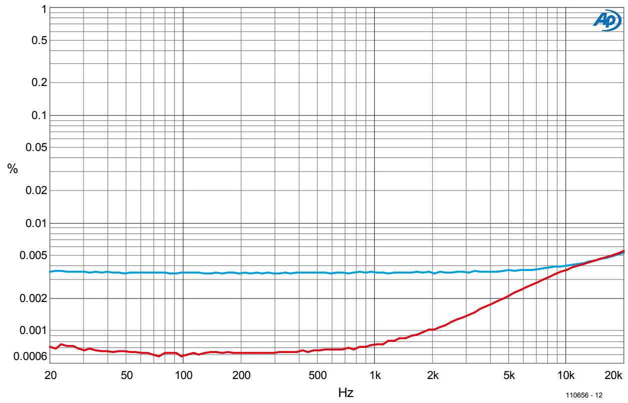
[1] Measurement Filter for Class D, Elektor July & August 2011, www.elektor.com/100540

Q-Watt Measured Characteristic Curves

Test equipment: Audio Precision System Two Cascade Plus 2722 Dual Domain

Plot A

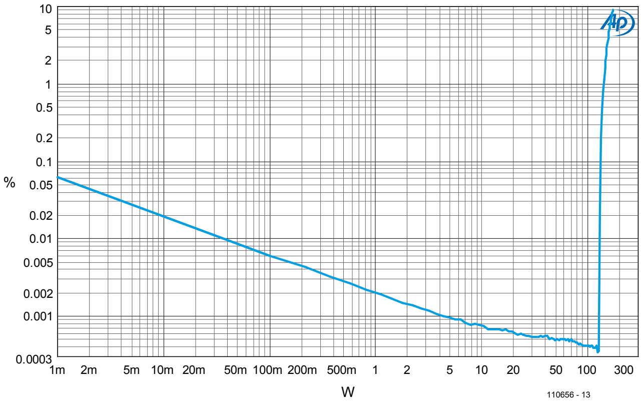
THD+N at output power levels of 1 W / 8 Ω and 50 W / 8 Ω, *B* = 80 kHz. The 1 W curve consists primarily of noise (THD+N = 0.0034%). The distortion does not rise above the noise until just before 20 kHz (THD+N = 0.0052%). At 50 W (which is exactly 20 V, so the results can be compared readily to the performance figures on the LME49811 datasheet) the noise floor is much lower relative to the output voltage. Here you can see that the distortion increases earlier at higher frequencies. At 1 W the distortion is still under the noise. The distortion above 10 kHz is nearly the same as the 1 W curve. The curve for 100 W is not shown here because it is virtually the same as the 50 W curve. The distortion is very low at all output power levels until just before the clipping level.



A

Plot B

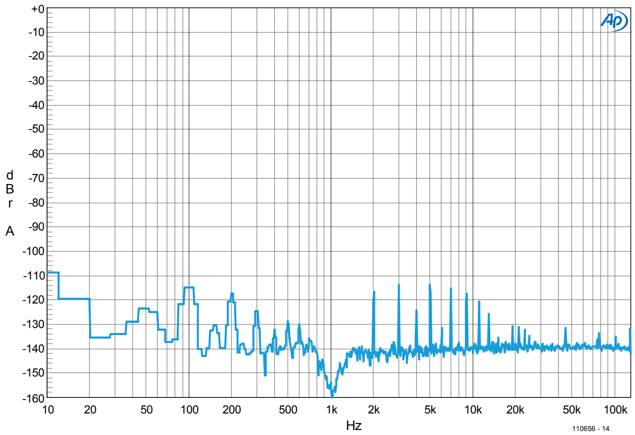
THD+N versus output power (1 kHz / 8 Ω, B = 22 kHz). The measurement bandwidth was reduced here to improve the visibility of the rise in distortion. Here again you can see that the distortion remains very low, while the noise floor drops as the output voltage rises. The clipping point is reached at 127 W, and the distortion rises rapidly above this point. At 137 W the THD+N reaches 0.1%, which is still a usable level for good sound quality. If you really overdrive the amplifier, it can deliver as much as 174 W with 10% distortion. Here it should be noted that with the low-cost power transformer used for the prototype, the supply voltage drops significantly at full output power (at 10% THD it falls to ±51.5 V). Even more output power would be possible with a transformer that provides a more stable supply voltage.



B

Plot C

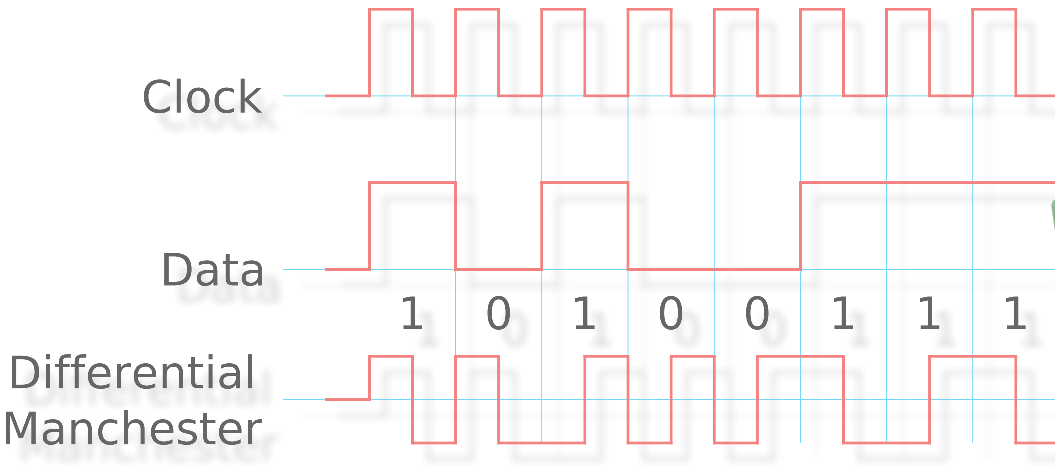
FFT of a 1 kHz signal at 50 W / 8 Ω (20 V rms). The levels of the residual harmonics in the supply voltage and the harmonics of the 1 kHz signal are very low, and in practice they would be inaudible. The third harmonic is at –113 dB, equivalent to just 0.0002%. The THD+N at this power level is 0.0006% (B = 80 kHz).

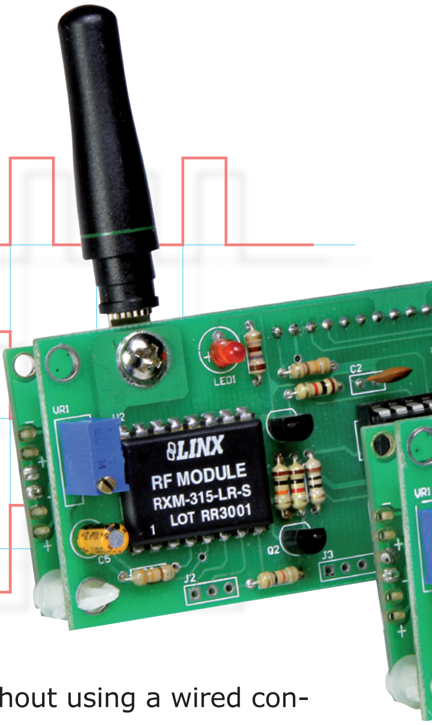


C

Modular RF Link using Manchester Code (1)

Part 1: Hardware





By
Marcelo Maggi (USA)

When there is a need to send data to a distant point without using a wired connection, either because the location's infrastructure does not allow a twisted pair to link both extremes, or just because one or both of the connected circuits are intended to be portable, or simply wireless, there are several ways to solve this, each of them with its own pros and cons. Let's investigate, solve and solder.

In this article we will describe a method to send data via Radio Frequency (RF), in the 315 MHz or 433 MHz ISM bands, at a maximum bit rate of 5000 bps, using low cost but highly reliable components, and implement an RF-friendly protocol, the Manchester Code, reaching distances in excess of about 600 feet (200 m).

Two general purpose units have been designed, one for transmission (TX) and one for reception (RX), ready to be used in any application, just by adapting the code of the included microcontroller.

Introducing TX, RX and antennas

Creating an RF link involves hardware and software (or firmware), both at the TX and RX end. This first installment of the article will describe the hardware.

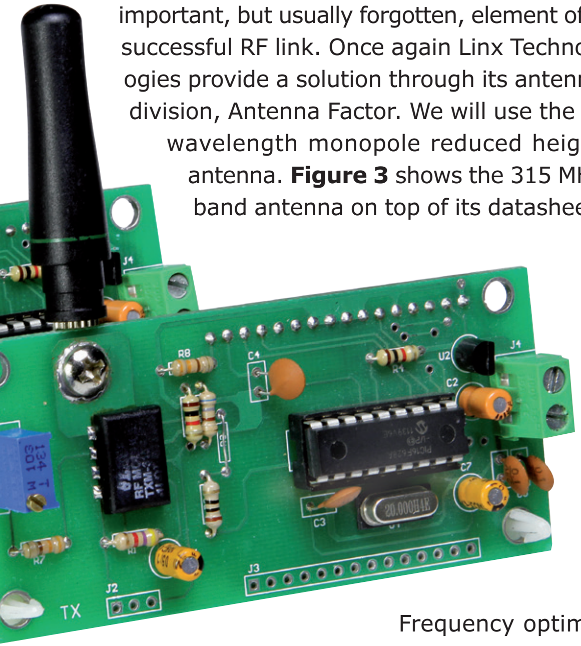
Designing a reliable RF circuit with discrete components is not an easy job, and the results are usually far from the expectations—much worse, that is. Fortunately, Linx Technologies have already taken care of this difficult task by offering complete RF modules encapsulated in a hybrid package. A wide range of used frequencies is available, but we will focus on the 315, 418 and 433 MHz bands for the purpose of this article, as these are the main free bands available, depending on where you live.

Figure 1 shows the transmitter module and its pin assignments, as shown in the part's datasheet (available from [1]). **Figure 2** is a mimic of the previous, but now depicting the receiver module. There are few active pins. The receiver has more pins, though most are not connected

Transmit 5000 bps across >600 ft

(NC). According to the datasheet you only need an antenna for basic operation, besides the obvious regulated power supply.

Speaking of antennas, this is another very important, but usually forgotten, element of a successful RF link. Once again Linx Technologies provide a solution through its antenna division, Antenna Factor. We will use the ¼ wavelength monopole reduced height antenna. **Figure 3** shows the 315 MHz band antenna on top of its datasheet.



Frequency optimization is indicated by a colored band on the body of the antenna; a green band indicates 315 MHz, while blue and red represent 418 MHz and 433 MHz respectively.

Designing the transmitter (TX)

The TX module simply transmits whatever signal is put onto the DATA pin, with a couple of restrictions of course, but there is no intelligence included, i.e. no data synchronization, no code protection, etc. The user needs to provide/implement this.

Have a look at the TX circuit's schematics, shown in **Figure 4**. The circuit is fairly simple; there's only one connection between the two main components, the microcontroller and the TX module: pin B0 (RB0/INT, pin 6) of the microcontroller connects to the DATA input of the TX module. The remainder of components is required for proper operation of the two main ones, but they play no active role in the actual data transmission. Components C1, C2, C3, C7 and C8 for instance are ceramic 0.1 µF decoupling capacitors. R5 is intended to keep the microcontroller running (pin 4 is the $\overline{\text{reset}}$ pin). The oscillator is formed by a standard 20 MHz quartz crystal.

The TX module works at 3 V, so the output of our (external) 5-V power supply is first applied to a fixed-voltage regulator type LP2950-30LPR, which provides a steady 3.0 volts to the module. C1, C5, C6 and R4 are added for additional stability, as recommended in the module's datasheet. Two additional resistors are connected to the TX module. R3 connects the PD pin to the 3 V supply rail, keeping it High. If Low, this pin will put the TX in a low-current state, in which it is unable to transmit. R1 is a simple wire jumper (0 Ω). However, when the transmit power is higher than you are allowed to use according to local regulations for ISM-band approved transmitters, it is possible to lower the transmit power by adjusting this resistor. With 0 Ω the maximum transmit power is selected, but increasing the resistance

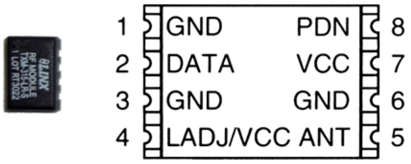


Figure 1.
The Linx TX Module with its pinout from the datasheet. There's just a few connections needed to get this module to work.

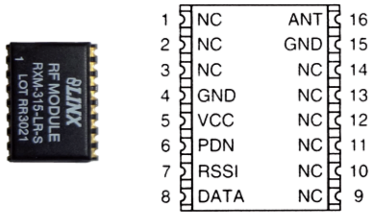
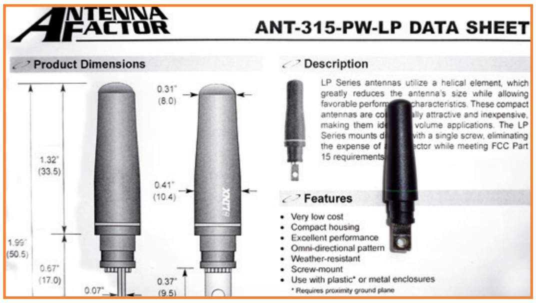


Figure 2.
The Linx RX Module with its pinout. A lot of pins aren't connected


ANTENNA FACTOR
0.31" (8.0)
1.32" (33.5)
0.41" (10.4)
1.99" (50.5)
0.67" (17.0)
0.07"
0.37" (9.5)
LINX

ANT-315-PW-LP DATA SHEET

Product Dimensions

Description

LP Series antennas utilize a helical element, which greatly reduces the antenna's size while allowing favorable perform characteristics. These compact antennas are co ally attractive and inexpensive, making them id volume applications. The LP Series mounts d with a single screw, eliminating the expense of a ector while meeting FCC Part 15 requirements.

Features

- Very low cost
- Compact housing
- Excellent performance
- Omni-directional pattern
- Weather-resistant
- Screw-mount
- Use with plastic* or metal enclosures

* Requires proximity ground plane

Figure 3.
The antenna is a ¼λ type from Antenna Factor.

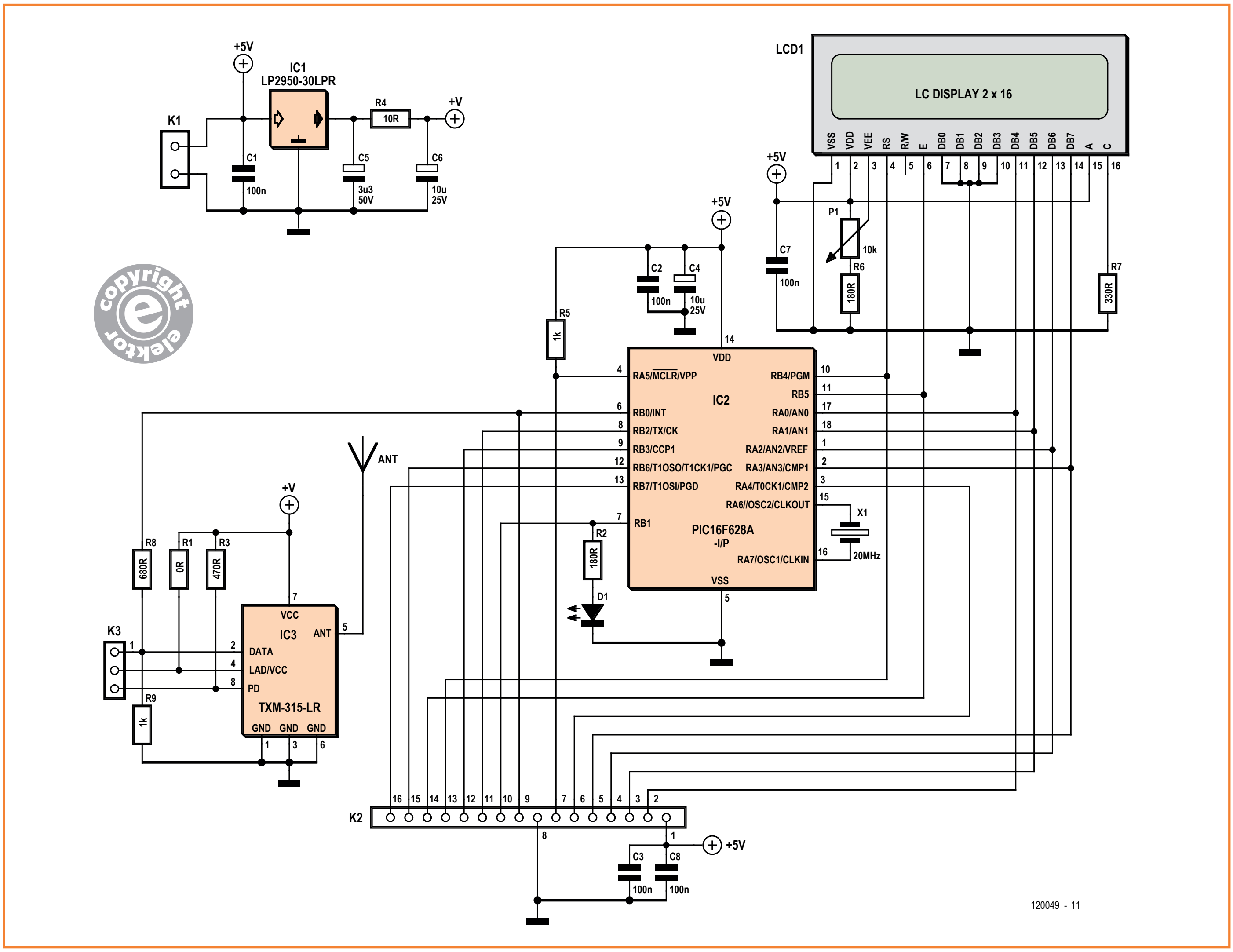


Figure 4.
The transmitter schematic, with a PIC controlling the LCD and data connection.

will reduce the power as shown in **Figure 5**. Since the transmitter operates at 3 V, the voltage on its data input pin must not exceed that level. Hence R8 and R9 are added, effectively forming a voltage divider.

For the microcontroller, plenty of options are available. The selected Microchip PIC microcontroller is a member of the subfamily of 18-pin PICs. The PIC16F628A is a 3.5 K version, but you may just as well use the PIC16F648A, a 7 K version, with larger SRAM and EEPROM, which can hold a larger program. Even the ‘old faithful’ PIC16F84A can be used—a real advantage when you happen to be familiar with it.

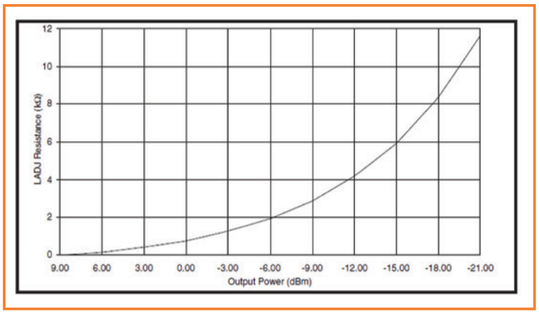


Figure 5.
Transmit Output Power can be trimmed using R3.

As can be seen in the schematic, access to most of the important pins of the microcontroller and the TX module is by way of K2 and K3, enabling full control over the operation if required, e.g. reset the microcontroller, input external data to be sent by the transmitter, etc. On the TX module, the transmit data and -power and the low-current state can be user controlled. K3 even allows a totally different microcontroller to be used, like the ones

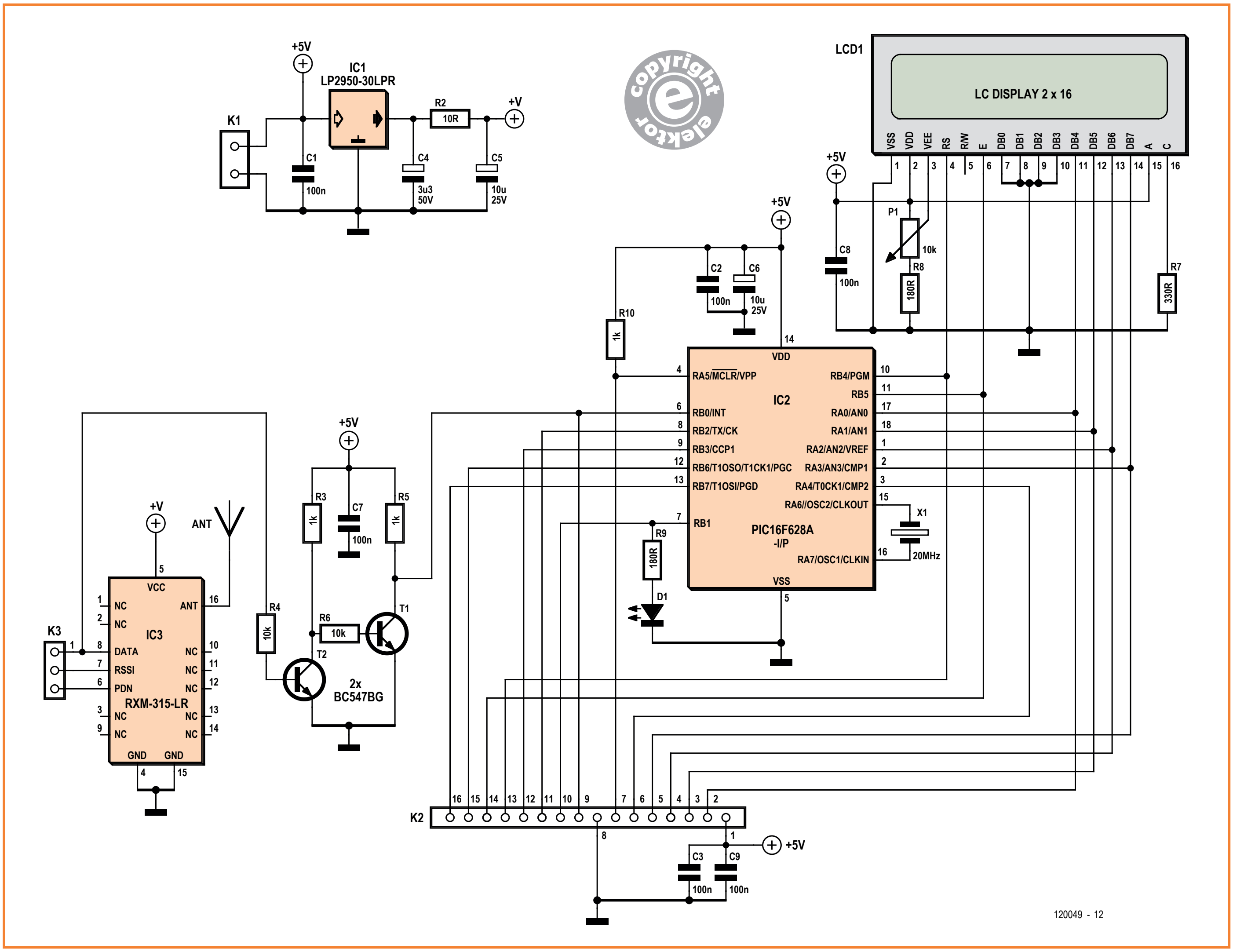


Figure 6.
The receiver schematic resembles that of the transmitter a great deal.

from Atmel. Just take the PIC out of its socket and drive the TX module directly using K3 (remember to use 3 V swing). Be sure to provide the 5 V supply needed for operation and everything has been taken care of to achieve reliable TX operation.

A final note: anyone with a bare knowledge of RF circuitry avoids testing any 10+ MHz RF circuit on a breadboard. The results—if any—will be wayward at best. However, this unit mounted on its PCB will be perfectly breadboard friendly, since all the RF related stuff is taken care of by the TX module.

Designing the receiver (RX)

Take a look at the schematic in **Figure 6**. Many already 'familiar' configurations can be seen; plenty of decoupling capacitors (C2, C3, C7, C8 and C9), a 3 V power supply similar to the one presented in the TX diagram (the RX module also works on 3 V), a noise canceling capacitor wired to the microcontroller's power pins (C6), R10 preventing the microcontroller from resetting, and a quartz crystal oscillating at 20 MHz. When we discuss the data handling program in the second installment, we will show the importance of having the microcontrollers run at the same frequency on both the TX- and RX side. The connectors for accessing the circuitry from the outside have a similar layout. The only difference is that in the RX module an RSSI (received signal strength indicator) output is available. The analog RSSI signal (useful to implement a squelch circuit) is routed to the middle pin of K3 instead

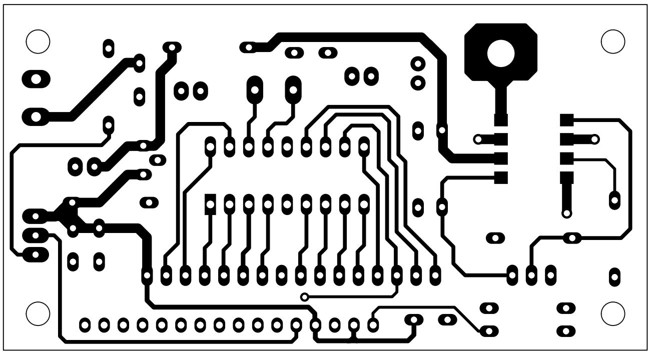


Figure 7. Top copper layout of the TX PCB.

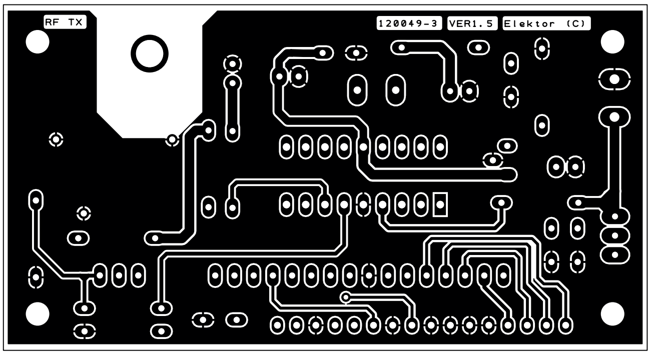


Figure 8. Bottom copper layout of the TX PCB.

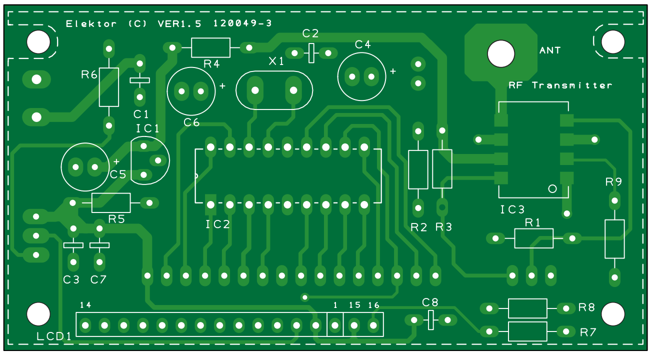


Figure 9. Component layout of the TX PCB.

of the transmit power adjustment pin (logically not available on the receiver module).

Parts R9 and D1, connected to pin 7 (RB1) of IC2, allow a quick check to be run whether the link is working or not. A very simple program lets the transmitter send the command to activate pin RB7 at the receiver side. If the command is transmitted and received correctly, the LED will light up. We will review this in detail when discussing the software.

Transistors T1 and T2 effectively form a non-inverting level converter from the 3 V output of the RX module to the 5 V input of the microcontroller. Two notes:

- Yes, the microcontroller could have been operated at 3 V. But in order to keep the design universal, it is designed for 5 V microcontrollers, to enable the use of older PICs that run on 5 V only.
- One transistor and a couple of resistors could have been saved had an inverting level converter been implemented, but then the signal inverting had to be dealt with in software. We didn't want to complicate things.

COMPONENT LIST

Transmitter

Resistors

R1 = 0Ω
R2,R6 = 180Ω
R3 = 470Ω
R4 = 10Ω
R5, R9 = 1kΩ
R7 = 330Ω
R8 = 680Ω
P1 = 10kΩ multiturn preset

Capacitors

C1,C2,C3,C7,C8 = 100nF
C4,C6 = 10µF 25V
C5 = 3.3µF 50V

Semiconductors

D1 = LED red, 5 mm pitch
IC1 = LP2950-30LPR
IC2 = PIC16F628A-I/P
IC3 = TXM-315-LR, Linx Technologies (418 or 433 MHz version as appropriate)

Miscellaneous

ANT = ANT-315-PW-LP, Linx Technologies
K1 = 2-pin PCB screw terminal block, 5mm pitch
K2 = 16-pin SIL connector, 0.1'' pitch
K3 = 3-pin SIL connector, 0.1'' pitch
LCD1 = 2x16 characters, DEM16217, Elektor Store #120061-71
X1 = 20 MHz quartz crystal
PCB #120049-3

Admittedly, if one were to design for mass production, the CFO would baulk at our solution.

Like with the TX module, you may use whatever microcontroller you like, just by removing the original PIC from its socket and using the three RX module lines. The PCB is 100% breadboard friendly and connects to your other designs in a snap.

Building the transmitter (TX)

Linx Technologies manufactures the TX modules in three frequencies, which are pin-compatible, so they can easily be interchanged. As stated before, RF design requires special precautions in order to obtain the desired performance. Although Linx Technologies have made a big effort to provide reliable and very stable modules, we must follow their recommendations regarding the PCB layout in order to achieve maximum performance. There are three key instructions to follow:

1. A ground plane on the layer opposite to the module must be implemented.

COMPONENT LIST

Receiver
Resistors
R1 = not fitted
R2 = 10Ω
R3,R5,R10 = 1kΩ
R4,R6 = 10kΩ
R7 = 330Ω
R8, R9 = 180Ω
P1 = 10kΩ multiturn preset

Capacitors
C1,C2,C3,C7,C8,C9 = 100 nF
C4 = 3.3uF 50V
C5,C6 = 10uF 25V

Semiconductors
D1 = LED red, 5 mm pitch
IC1 = LP2950-30LPR
IC2 = PIC16F628A-I/P
IC3 = RXM-315-LR, Linx Technologies (418 or 433 MHz version as appropriate)
T1,T2 = BC547B

Miscellaneous
ANT = ANT-315-PW-LP, Linx Technologies
K1 = 2-pin PCB screw terminal block
K2 = 16-pin SIL connector
K3 = 3-pin SIL connector
LCD1 = 2x16 character, DEM16217, Elektor #120061-71
X1 = quartz crystal, 20 MHz
PCB #120049-4

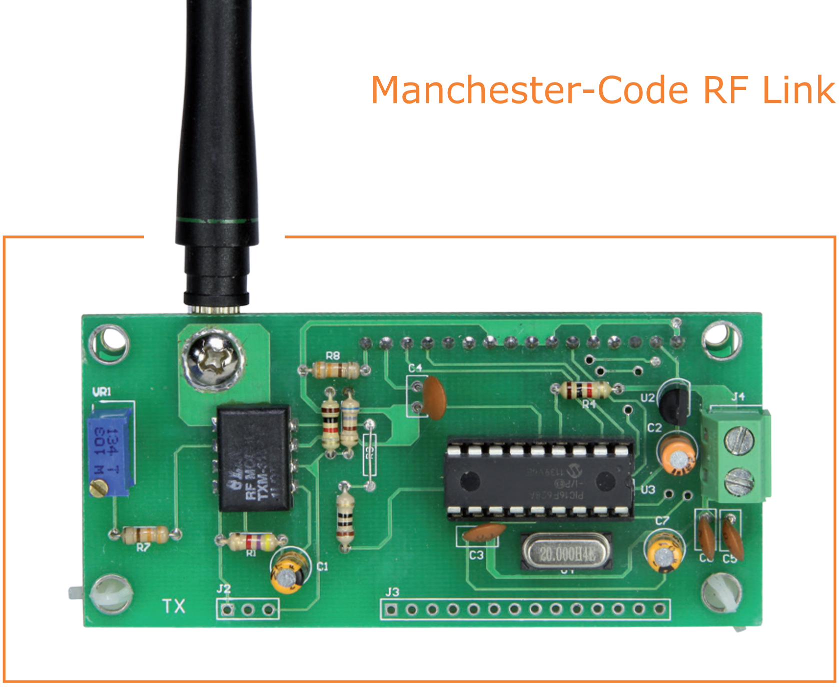


Figure 10. Our first prototype still needs some changes.

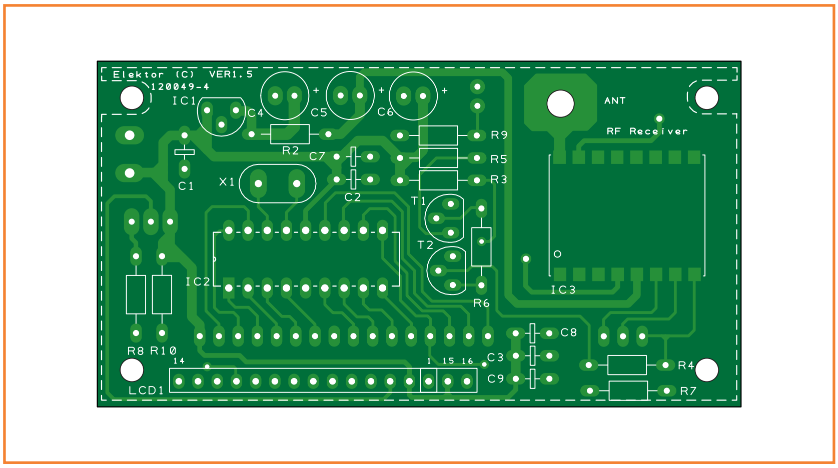


Figure 11. Component layout of the RX PCB.

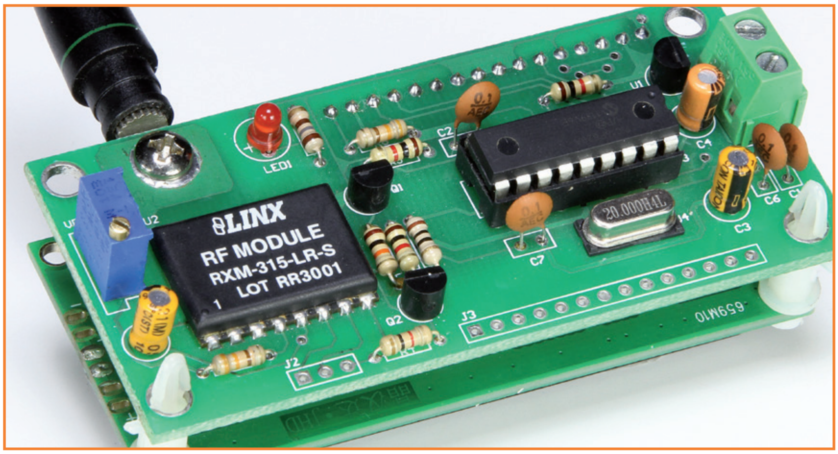
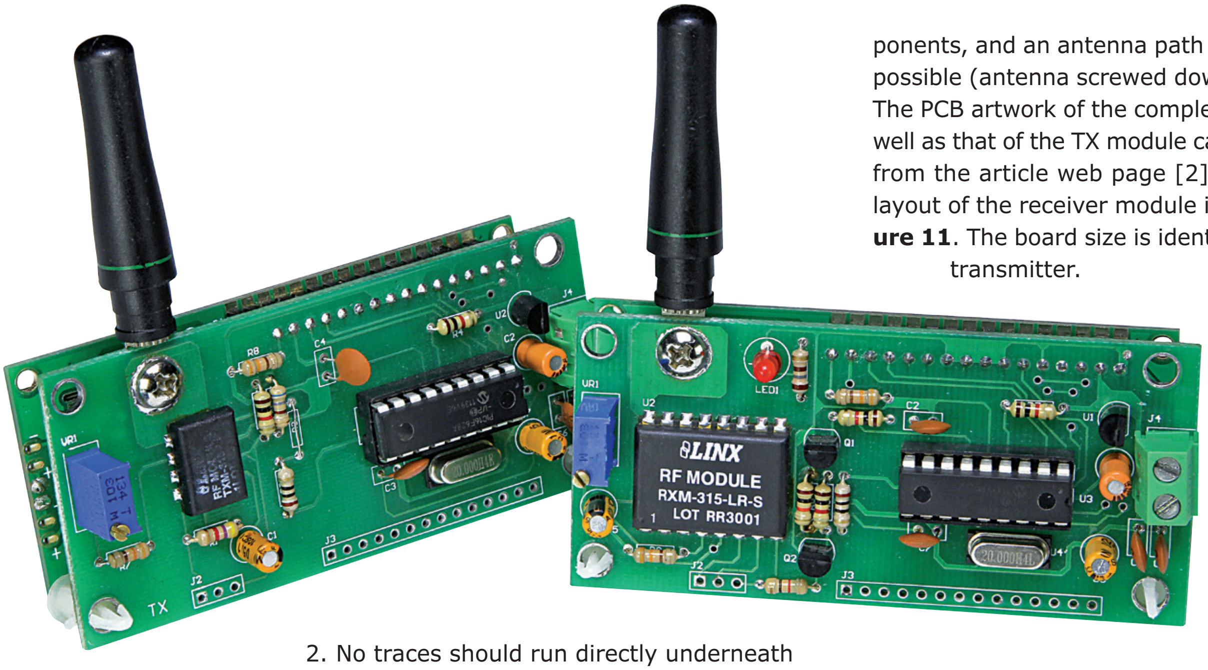


Figure 12. The LCD fits right onto the back of the Receiver Module prototype.



2. No traces should run directly underneath the module, and no conductive items should be placed within a 0.15 inch (3.8 mm) radius of the module’s top and sides.
3. The antenna path should be as short as possible.

With these guidelines in mind, we started the PCB design around the TX module area, and the rest of the components got accommodated accordingly. **Figure 7** shows the PCB layout for the complete TX circuit viewed from the component side, while **Figure 8** shows the copper (solder) side view. Note the isolated position of the TX module, the ground plane on the copper side and only a few traces on the component side. As recommended, the antenna path is very short. The antenna itself is fixed onto the PCB with a screw. A touch of solder is a good practice to secure electrical and mechanical robustness. The lead pitch of K2 and K3 is 0.1 inch (2.54 mm), so the circuit is easily plugged onto a bread board.
The component layout appears in **Figure 9**. The board measures 3.35 × 1.80 inch (85 × 46 mm). **Figure 10** shows a first prototype of the circuit. The microcontroller is mounted in a socket for easy removal in order to be able to easily reprogram it with updated firmware.

Building the receiver (RX)

All of the caveats stated during the assembling of the TX module apply to the RX module as well: a large ground plane at the copper side, reasonable separation of the module from the other components, and an antenna path that’s as short as possible (antenna screwed down and soldered). The PCB artwork of the complete RX circuitry as well as that of the TX module can be downloaded from the article web page [2]. The component layout of the receiver module is pictured in **Figure 11**. The board size is identical to that of the transmitter.

At 0.1 inch the lead pitch of the connectors is exactly the same as in the TX module and 100% breadboard compatible. As with the transmitters, Linx Technologies offers the receivers geared to three different UHF-band ISM frequencies.
In **Figure 12** our first receiver prototype is shown, again with the microcontroller socketed for easy reprogramming.

Although it may seem obvious, it will do no harm to emphasize this: for an RF link to work, all RF components: TX module, TX antenna, RX module and RX antenna, MUST be tuned to the same frequency. The antenna obviously does not discriminate between transmission and reception, so the same model is used for the TX and RX modules. A note here: while these antennas are pretty good, they are not perfect. At the transmitting side you may want to limit the power (and sometimes the efficiency of the antenna) to stay at or below the level permitted by local regulations. At the receiver end you may want to boost efficiency as much as possible. So feel free to experiment with straight ¼ wavelength monopoles (i.e. rods) in case you need a larger range.

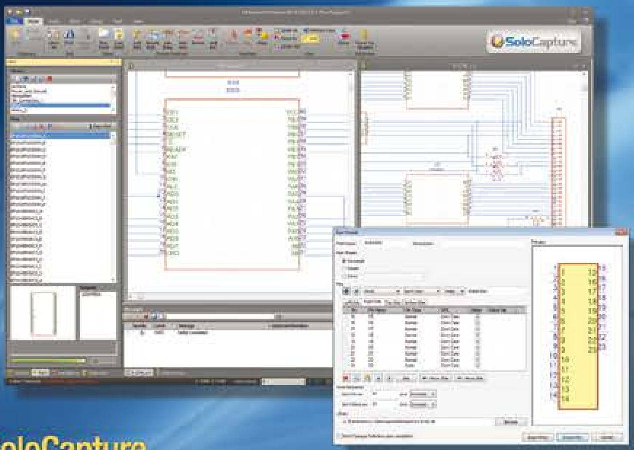
This concludes the first part of this article. In next month’s second and concluding part we will discuss the software, which has been developed as a true general-purpose solution.

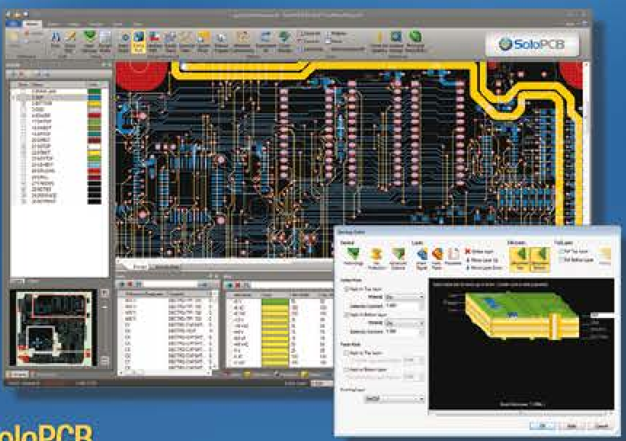
(120049)

Internet Links

[1] www.linxtechnologies.com

[2] www.elektor.com/120049





Gnublin Extension Boards

Plus command-line tools for the Elektor Linux board and Raspberry Pi

By
Benedikt Sauter [1]

Previously we published details of a relay board that can be connected to the Elektor Linux board, the Raspberry Pi, and other microcontroller boards. That was but one example of a wide range of expansion boards developed by the Embedded Projects team. The family is complemented by handy command-line tools and a C/C++ API to help you develop your own applications.

Once a connector specification has been agreed upon, it becomes possible to put together microcontroller boards and expansion boards in arbitrary combinations. We have previously described [2] the 14-way Gnublin connector on the Elektor Linux board, which will also feature as the 'Embedded Extension Connector' on the Xmega web server board that we will describe in the next issue. In this article we will describe a range of other expansion boards and in particular show how easy it is to control them under Linux. To allow for rapid testing the development team has written a small command-line program to accompany each module. A complete C/C++ API is also available to aid in application development. The API sits on top of the standard device drivers for I2C, SPI, GPIO, ADC and so on, and the application developer is spared having to deal with these drivers directly. Instead, he can control a device (such as a stepper motor) using straightforward function calls (see Figure 1). The team is currently also working on a Python API: more information on this at [4].

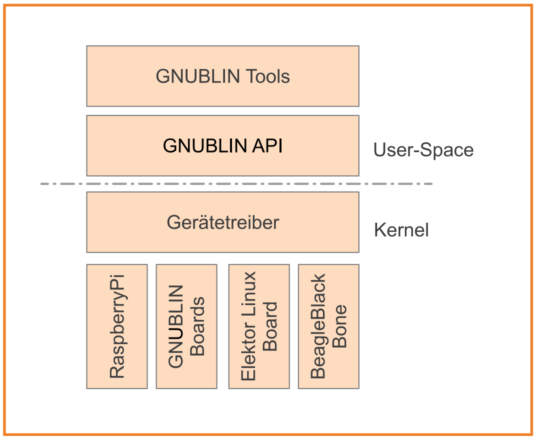


Figure 1. The Gnublin API spares you the need to deal with SPI, I2C and other device drivers.

The idea

Linux provides a very good abstraction layer that allows applications to be developed in a way that is independent of the target processor: one simply writes an application 'for Linux'. The new expansion boards, which are available via www.elektor.com/gnublin, extend this idea to cover projects that use motors, displays, temperature sensors, relays and much more besides. The boards are connected to the Elektor Linux board using a simple flatcable.

- Relay Module (controls eight relays) (130212-91, Figure 2)
- Temperature Module (temperature sensor) (130212-95, Figure 3)
- Display Module (4x20) (four-line text display) (130212-92, Figure 4)
- Step Module (stepper motor driver) (130212-93, Figure 5)
- I/O Expander Module (16 digital inputs and

outputs) (130212-94, Figure 6)

- - Extension Module (display, buttons, real-time clock, buzzer and port expander) (120596-91, Figure 7)

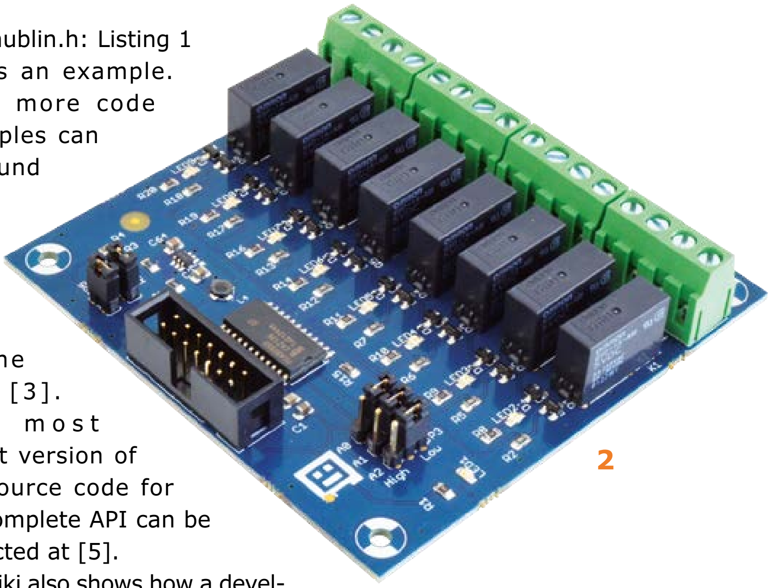
If you need to connect more than one expansion board you can use the Bridge Board (130212-71, Figure 8).
As we mentioned previously, an adapter board is also available to interface the expansion boards to the popular Raspberry Pi mini-computer (120212-72). A new development is an adapter board for the BeagleBone Black (130212-74).

Command-line tools

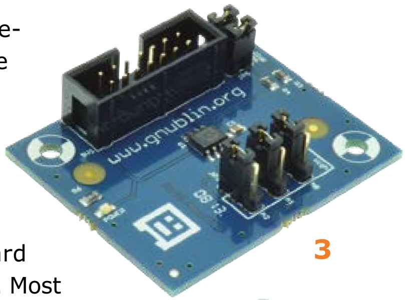
The command line is the sine qua non of working with the Linux board. From the command line you can launch and stop applications, administer the Linux system, read system messages and much more besides.
The development team has written a number of small command-line tools to help control the Gnublin expansion boards and some of the internal functions of the Linux board. Typing `gnublin-` at the command line and then pressing 'Tab' will produce a list of these mini-programs. Table 1 shows a sample.
The tools are very convenient for initial testing: you can easily determine whether the hardware is connected correctly. As many will know to their cost, it is easy to spend a long time fruitlessly hunting for bugs in software only to discover that the power supply is not connected!

The C/C++ API

After the module has been connected and tested we can get down to writing a real application. The C/C++ API mentioned above lets you use a range of easy-to-understand function calls, avoiding, for example, the use of pointers and structures. A separate software module is provided for each interface and expansion board (see Table 2). To use these functions with the Gnublin Elektor Linux board you simply need to include the header file gnublin.h: Listing 1 shows an example. Many more code examples can be found on the wiki [3]. The most recent version of the source code for the complete API can be inspected at [5].
The wiki also shows how a development environment can be created to simplify working with the API.



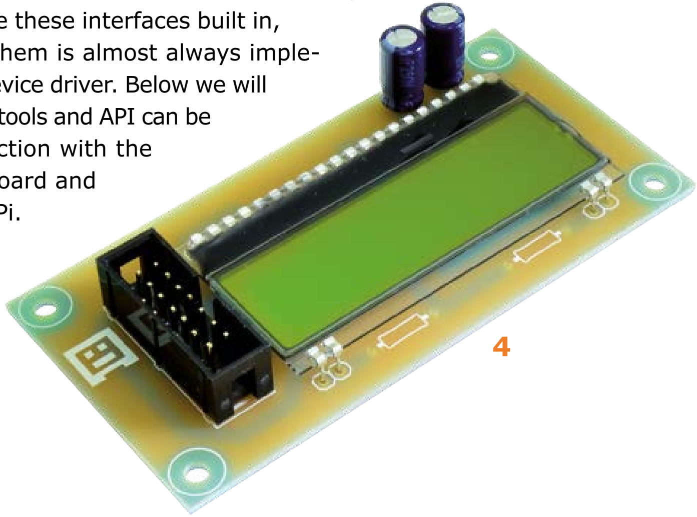
2



3

Installing the tools and API

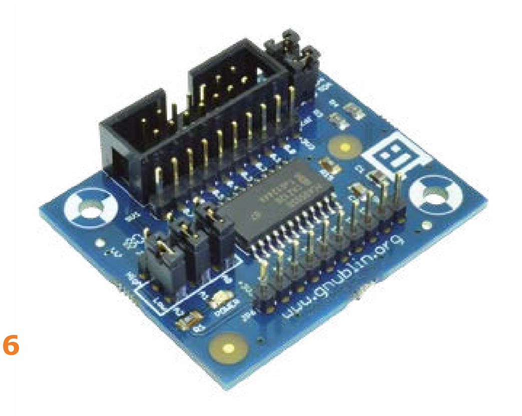
In principle the API can be used with any embedded Linux board that has drivers for I2C and SPI. Most processors have these interfaces built in, and access to them is almost always implemented via a device driver. Below we will look at how the tools and API can be used in conjunction with the Elektor Linux board and the Raspberry Pi.



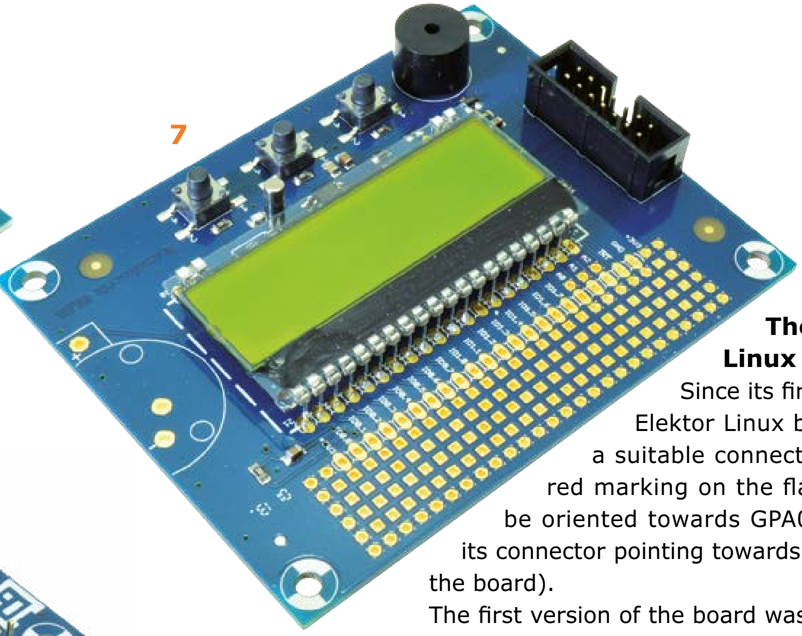
4

Table 1. Gnublin command-line tools (sample)		
Tool	**Example invocation**	**Description**
gnublin-lm75	(command takes no arguments)	Display temperature
gnublin-relay	gnublin-relay -p 1- o 1	Switch on relay 1
gnublin-adcint	gnublin-adcint -c 1	Read internal ADC, channel 1
gnublin-step	gnublin-step -p 3000	Move stepper motor to position 3000

5



6



7

Listing 1. Controlling an I²C device

```
#define BOARD_GNUBLIN
//#define BOARD_RASPBERRYPI

#include “gnublin.h”

int main()
{
   gnublin_i2c i2c;

   i2c.setAddress(0x42);        //i2c slave address

   char buffer[8];
   char RxBuf[8];

   buffer[0]=0x22;

   i2c.send(buffer,5);
   i2c.send(0x12, buffer, 2);   //send 2 bytes register 0x12

   i2c.receive(RxBuf, 3);       // read 3 bytes
   i2c.receive(0x23, RxBuf, 3); // read from register

}
```

The Elektor Linux board

Since its first version the Elektor Linux board has had a suitable connector fitted. The red marking on the flat cable must be oriented towards GPA0 (the key on its connector pointing towards the middle of the board).

The first version of the board was shipped with an ELDK filesystem, but this was subsequently replaced by a complete Debian image. The procedure for updating older memory cards is described at [6].

Compiling the expansion board tools on the Linux board itself takes a good five minutes. We have therefore built a Debian package that makes installing the tools much simpler.

First download the Debian package file on the PC. Then transfer the file to the SD card using the PC’s card reader.

If the board itself is connected to the Internet you can download the package file directly using the command line:

```
wget https://github.com/embeddedprojects/
gnublin-api/raw/master/gnublin-tools.deb
```

The tools are installed as follows:

```
root@gnublin:~# dpkg -i gnublin-tools.deb
```

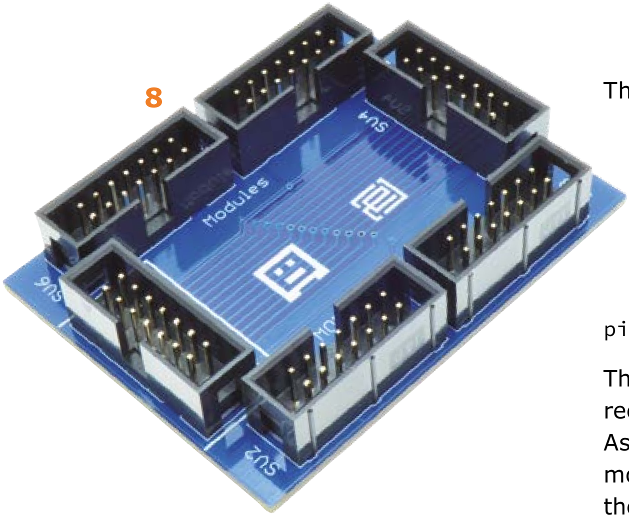
To remove the package at a later date, use the following command:

```
root@gnublin:~# dpkg -r gnublin-tools
```

Raspberry Pi

The easiest way to use the software module on the Raspberry Pi is to connect directly to the source code repository. Boot up the Raspberry Pi and make sure it has an Internet connection. The repository is cloned using the ‘git’ command: if this is not already installed on your Raspberry Pi, install it as follows:

8



```
pi@raspberrypi ~ $ sudo apt-get install
git
```

Now fetch a copy of the repository

```
pi@raspberrypi ~ $ git clone https://
github.com/embeddedprojects/gnublin-api.
git
```

switch to the source directory

```
pi@raspberrypi ~ $ cd gnublin-api
```

and compile and install the code, examples and the API:

```
pi@raspberrypi ~ $ make && sudo make
install
```

The API requires the following drivers to be activated:

```
      pi@raspberrypi ~ $ sudo modprobe
      spi-bcm2708
     pi@raspberrypi ~ $ sudo modprobe
  i2c-bcm2708
pi@raspberrypi ~ $ sudo modprobe i2c-dev
```

These drivers are already present in the most recent version of the Raspberry Pi distribution. As an alternative to the above commands, the module names can be entered permanently in the file '/etc/modules', one module per line:

```
spi-bcm2708
i2c-bcm2708
i2c-dev
```

The tiny command-line tools can now be used to test any expansion board connected to the Raspberry Pi.

(130212)

Internet Links

[1] sauter@embedded-projects.net

[2] www.elektor.com/130157

[3] http://wiki.gnublin.org/index.php/API

[4] http://en.gnublin.org/index.php/API_Python

[5] https://github.com/embeddedprojects/gnublin-api

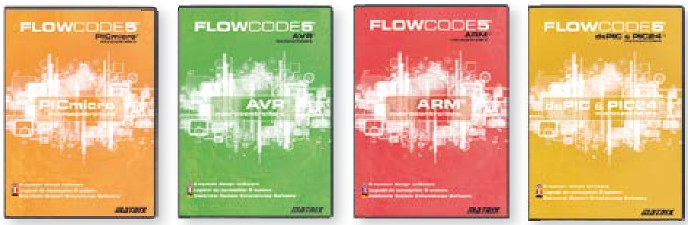
[6] http://en.gnublin.org/index.php/GNUBLIN-Elektor

Table 2. Software API objects (sample)

Modul	**Interface**	**Remarks**
gnublin_gpio	Internal	
gnublin_adc	Internal	Currently only available on Elektor Linux board (not Raspberry Pi)
gnublin_i2c	I^{2}C	Standard I^{2}C bus
gnublin_spi	SPI	Standard SPI devices
gnublin_pwm	Internal	Currently only available on Elektor Linux board
gnublin_module_lm75	I^{2}C	Temperature sensor
gnublin_module_relay	I^{2}C	Relay board
gnublin_module_pca9555	I^{2}C	Port expander with 16 digital inputs and outputs
gnublin_module_step	I^{2}C	Stepper motor
gnublin_module_lcd	I^{2}C	Display, 4x20 characters

Developing and learni

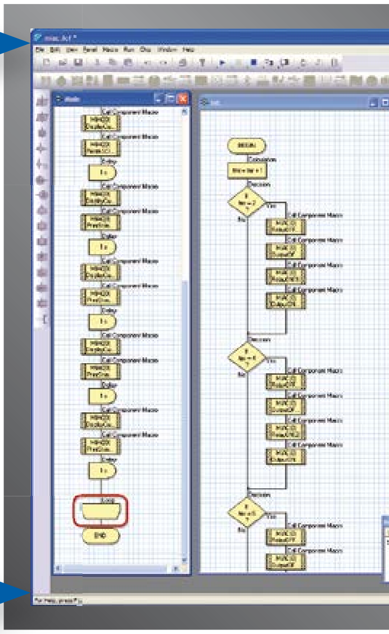
FLOWCODE5

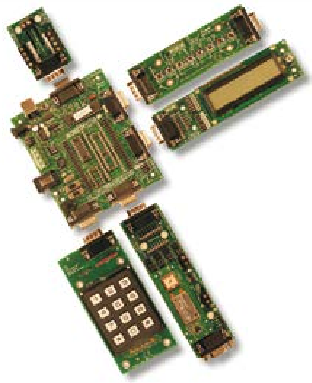


Flowcode 5 is one of the world's most advanced graphical programming languages for microcontrollers (PIC, AVR, ARM and dsPIC/PIC24). The great advantage of Flowcode is that it allows those with little to no programming experience to create complex electronic systems in minutes.

www.elektor.com/flowcode

… for electronics





E-Blocks are small circuit boards each of which contains a block of electronics that you would typically find in an electronic or embedded system. There are more than 40 separate circuit boards in the range; from simple LED boards to more complex boards like device programmers, Bluetooth and TCP/IP. E-blocks can be snapped together to form a wide variety of systems that can be used for teaching/learning electronics and for the rapid prototyping of complex electronic systems. Separate ranges of complementary software, curriculum, sensors and applications information are available.

… for industrial control

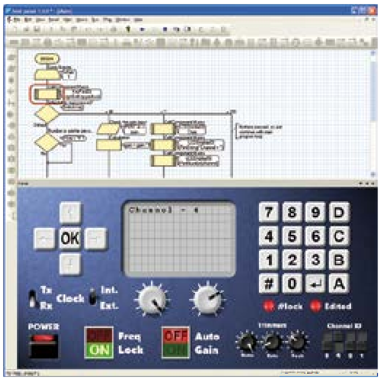
MIAC (Matrix Industrial Automotive Controller) is an industrial grade control unit which can be used to control a wide range of different electronic systems including sensing, monitoring and automotive. Internally the MIAC is powered by a powerful 18 series PICmicro device which connects directly to the USB port and can be programmed with Flowcode, C or assembly. Flowcode is supplied with the unit. MIAC is supplied with an industrial standard CAN bus interface which allows MIACs to be networked together.

Flowkit

Flowkit provides In Circuit Debugging for a range of Flowcode applications for PIC and AVR projects:

- Start, stop, pause and step your Flowcode programs in real time
- Monitor state of variables in your program
- Alter variable values
- In circuit debug your Formula Flowcode, ECIO and MIAC projects

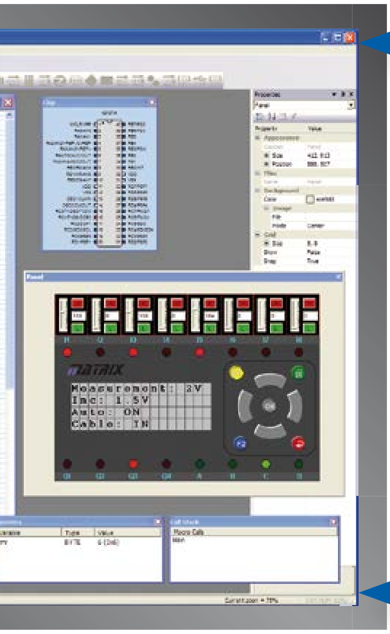
ng with Flowcode 5



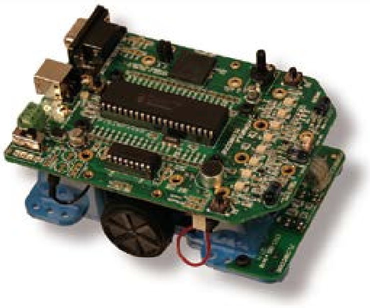
New features in Flowcode 5

Flowcode 5 is packed with new features that make development easier including:

- New C code views and customization
- Simulation improvements
- Search and replace function
- New variable types and features, constants and port variables
- Automatic project documentation
- New project explorer makes coding easier
- Implementation of code bookmarks for program navigation
- Complete redesign of interrupts system allows developers access to more chip features
- Compilation errors and warnings navigate to icons
- Disable icons feature
- Improved annotations
- Improved links to support media
- Support for MIAC expansion modules and MIACbus



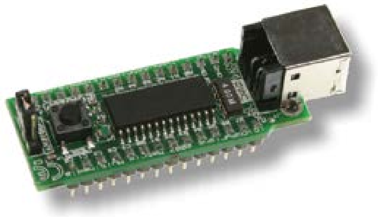
… for robotics



Formula Flowcode is a low cost robot vehicle which is used to teach and learn robotics, and to provide a platform for competing in robotics events. The specification of the Formula Flowcode buggy is high with direct USB programming, line following sensors, distance sensors, 8 onboard LEDs, sound sensor, speaker and an E-blocks expansion port. The buggy is suitable for a wide range of robotics exercises from simple line following through to complete maze solving. E-blocks expansion allows you to add displays, connection with Bluetooth or Zigbee, and GPS.

… for USB projects

ECIO devices are powerful USB programmable microcontrollers with either 28 or 40 pin standard DIL (0.6") footprints. They are based on the PIC 18 series and ARM 7 series microcontrollers. ECIO is perfect for student use at home, project work and building fully integrated embedded systems. ECIO can be programmed with Flowcode, C or Assembly and new USB routines in Flowcode allow ultra rapid development of USB projects including USB HID, USB slave, and USB serial bus (PIC only). ECIO can be incorporated into your own circuit boards to give your projects USB reprogrammability.



More information and products at:

www.elektor.com/eblocks

(Part 2) Android Elektor Cardi♥scope

Wireless, button-free: Bluetooth & touch screen

Marcel Cremmel
(France)
in co-operation with
Raymond Vermeulen
(Elektor Labs)

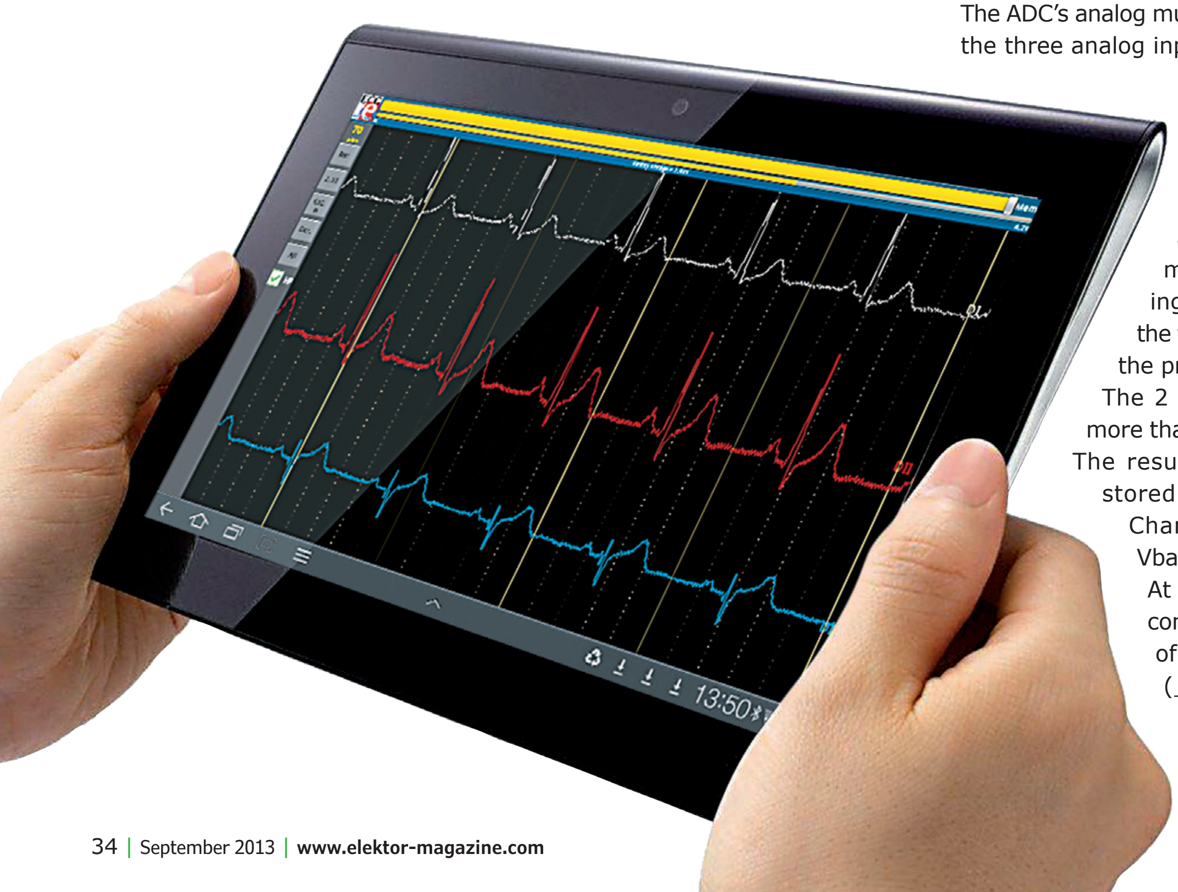
Following on from the description of the hardware for our new ECG interface on tablets or Android smartphones in the July & August 2013 double edition, we're coming back now to the PIC functions and how the program runs, before looking at the Android application. Without going into too much detail though—we're just going to say enough about this to encourage readers to get hold of the code and have a go at development under Android.

What does the PIC24 do?

Acquiring and transmitting the samples (Figure 5)

Three hardware modules included within the microcontroller are used:

- the 10-bit ADC and its analog multiplexer,
- the UART (*Universal Asynchronous Receiver Transmitter*) for communicating with the Bluetooth module (= BT),
- *Timer1* for producing the P2HZ and CAL signals.



The ADC's analog multiplexer allows us to convert the three analog inputs DI, DII, and BATT_LEV. The latter signal is produced by a resistive divider (R16/R17) that yields ½ of the battery voltage.

The ADC is configured in autoconversion and autoscan mode: it takes care of selecting, sampling, and converting the three inputs without involving the processor.

The 2 kHz sampling frequency is more than enough for an ECG signal. The results of the conversions are stored in three 16-bit variables: Channel_DI, Channel_DII and Vbatt.

At the end of each of the three conversions, i.e. at a frequency of 2 kHz, an interrupt (_ADC1Interrupt) performs the following processing :

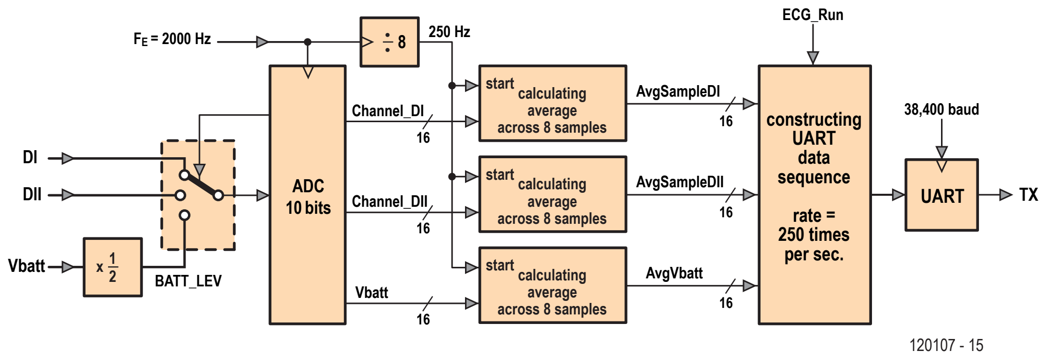


Figure 5.
Acquiring and transmitting DI and DII samples.

Hearts are trumps – that's just the PIC

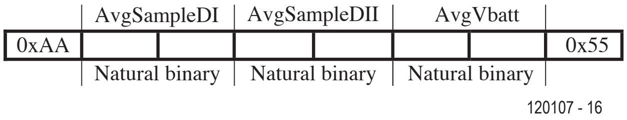


Figure 6.
UART data sequence format.

- Every 8 samples, i.e. at a rate of 250 Hz: calculates the average values **AvgSampleDI**, **AvgSampleDII**, and **AvgVbatt**. This processing makes it possible to reduce the effect of occasional interference.
- Construction and transmission of the asynchronous serial data sequence to the BT module.

Figure 6 shows the format adopted for this 8-byte sequence. The data are framed by the bytes 0xAA and 0x55. These will be used in the Android terminal to perform sequence synchronization and hence to identify and read out the samples. The sample values lie between 0x0000 and 0x03FF (10-bit conversion in natural binary), so mis-synchronization is impossible.

Selecting the auto-zero time-constants (Figure 7)

This software function constantly adapts the DI and DII signal alignment speed (see "Open-heart diagrams" paragraph in 1st article) so as to stabilize each ECG on the screen as rapidly as possible. To achieve this, the **MovingAverageCalc()** function calculates the moving average of the **AvgSampleDI** and **AvgSampleDII** digital signals over a period of 4 s. The **DI_Average** and

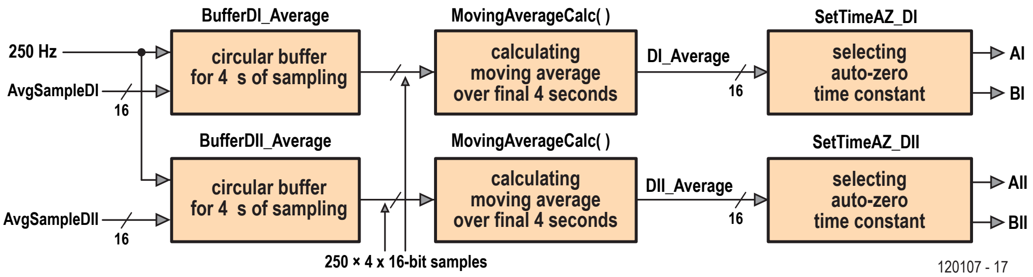


Figure 7.
Selecting the auto-zero time-constants.

NB: The numbering for the illustrations and links follows on from that in the first part of this article.

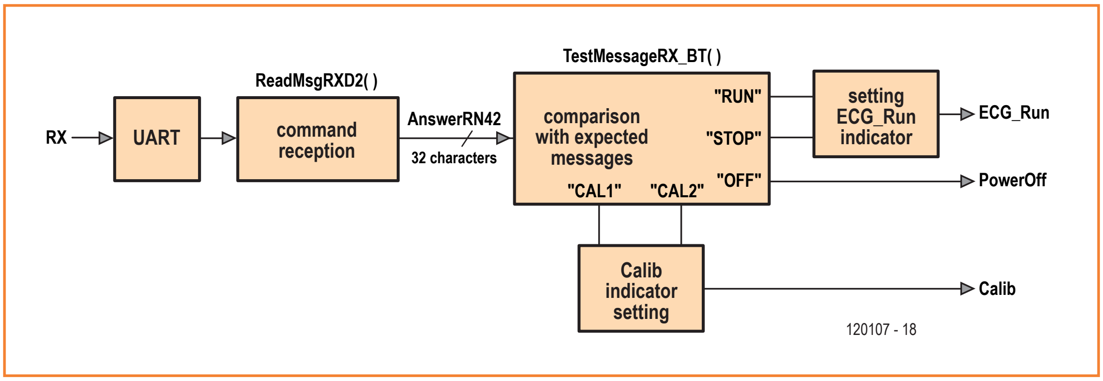


Figure 8.
Receiving orders from the Android smartphone or tablet.

DII_Average results are compared with the expected quiescent values in order to select, via AI and BI or AII and BII, an auto-zero time-constant that is faster when the difference is greater. Let's remind ourselves what is meant by the expression "moving average". The **AvgSampleDI** and **AvgSampleDII** samples are stored in a 4 s circular buffer, i.e. here 4×250 = 1,000 words of 16 bits. The **MovingAverageCalc()** function then calculates the arithmetic mean of the last 1,000 samples from the buffer sufficiently fast. The last sample corresponds to the moment of the calculation and hence moves over time.

Receiving orders from the terminal (Figure 8)

There are not many orders that come to the user from the terminal:

- a Run/Stop command to enable or block transmission of the data sequences.
- interface power-down Note that the interface can only be powered on via its On/Off button
- the **CAL0** and **CAL1** commands to control production of the calibration signals.

The UART module takes care of serial/parallel conversion for each byte of the message received. The byte reception functions provided in the *Microchip* libraries do not use interrupts. To avoid wait loops for these functions, which occupy the processor to no useful purpose, we are using the UART receive interrupt. The associated **_U2RXInterrupt** function stocks received characters in a buffer of sufficient size (256 bytes). These characters are promptly read by the **ReadMsgRXD2()** function.

The character string variable **AnswerRN42** is assigned as soon as a complete message is received (end sequence = CR-LF). The **TestMessageRX_BT()** function then compares this with one of the commands expected.

It affects accordingly the **ECG_Run** indicator validating the transmission of the data sequences (Figure 5), the $\overline{\text{PowerOff}}$ signal, and the **Calib** indicator confirming the production of the calibration signal.

Producing the calibration signals (Figure 9)

The P2HZ and CAL signals act on the analog multiplexer IC9 (Figure 3, F2) to periodically replace the voltages picked up by the electrodes with a calibration signal with an amplitude of 1 mV. The frequency of the P2HZ signal is 2 Hz with a duty cycle of 20 %, both close to those of an ECG signal. The signal produced by the microcontroller is attenuated by the network R21/R22/

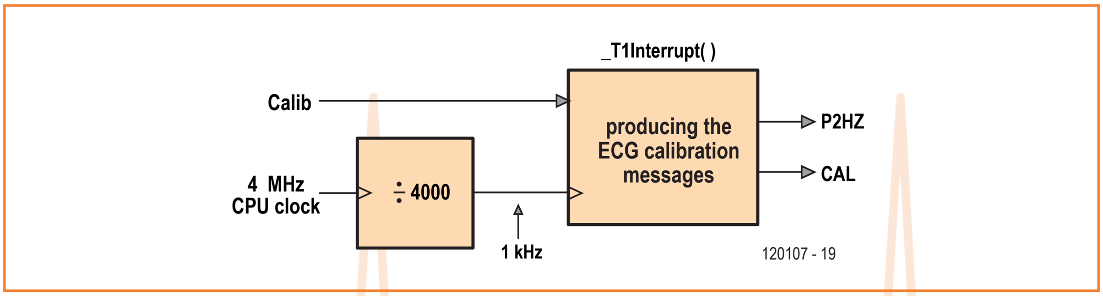


Figure 9.
Producing the calibration signals.

Record and view your own electrocardiograms on your smartphone or tablet!

R65 to obtain an amplitude of 1 mV and a mean value of zero.
The CAL signal goes high for 10 s every minute if the user has enabled it from the tablet or smartphone.

These signals are produced by a sequencer built in to the microcontroller. This comprises:

- a ÷4,000 frequency divider achieved using a hardware structure (*Timer1* module)
- a software interrupt function **_T1Interrupt** activated 1000 times per second. If the **Calib** indicator is set, counting variables are incremented and compared with constants to produce the P2HZ and CAL signals.

Bluetooth link status (Figure 10)

If there is no BT link, there is no point converting and transmitting the ECG signals. The STATUS signal produced by the BT module gives this information: link connected (1) or broken (0).
The **_CNInterrupt** interrupt function is activated whenever the STATUS state changes and accordingly affects the **ECG_Run** indicator and the ADON A/D converter validation bit (Figure 5).
The choice made to use an interrupt function obviates the need to interrogate the STATUS signal periodically and hence saves processor time.

PIC program sequence

The program architecture is conventional (unlike the Android application, as we'll be seeing). After initialization, the following operations are performed:

- initialization of the variables, the input/output ports, *Timer1* for producing the calibration signals (see above) and of the UART2 coupler for communicating with the BT module.
- configuration of the BT module to 38,400 baud
- initialization of the 10-bit ADC: sampling frequency 2 kHz, autoconversion and autoscan for the three analog inputs
- validation of the CN interrupt

the program then goes into an endless loop:

- call **TestMessageRX_BT()**: read and process any order received from the terminal
- call **MovingAverageCalc()**: calculate the **DI_Average** moving average
- call **SetTimeAZ_DI()**: selecting the auto-zero time-constants for the DI channel
- call **MovingAverageCalc()** and SetTimeAZ_DII() for the DII channel.

The average value calculation functions are placed in the endless loop, as the time to perform them is quite long (26,800 CPU cycles, i.e. 6.7 ms). In accordance with programming rules, one avoids using interrupt functions to handle lengthy processing. In point of fact, the other, lower-priority interrupt functions would not be executed during this time, which could lead to an error.
The execution frequency of the endless loop is around 75 Hz, frequent enough for calculating the moving averages and selecting the auto-zero time-constants.

Man/machine interface

It would be hard to find a more user-friendly (and cheaper) MMI than an Android terminal (or an *iPhone*). Elektor has already published a great many articles on this subject and even a book, the success of which confirms the great demand: *AndroidApps Programming Step by Step* by Stephan Schwark [4]. We invite readers interested in this subject to explore, extend, and even critique the Elektorcardioscope code available on the Elektor website [3]. It's impossible to describe the 1,900 lines of code in just a

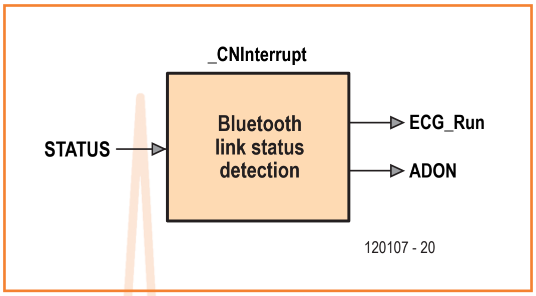


Figure 10.
Detecting the Bluetooth link status.

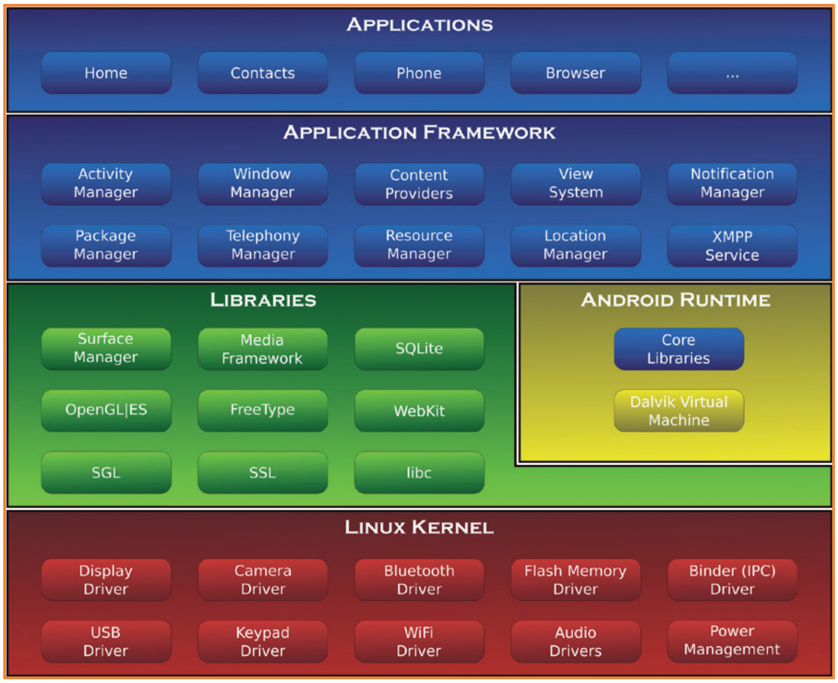
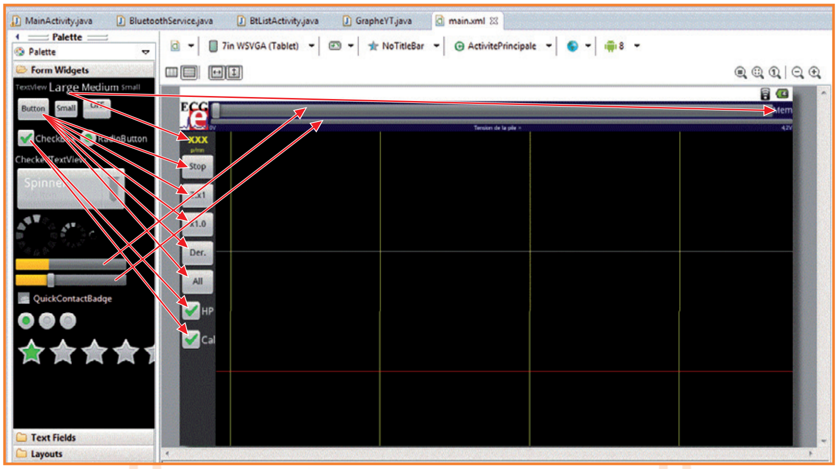


Figure 11.
The Android system architecture (this image, reproduced here at small scale just for information, can be downloaded as a high resolution file).

Figure 12.
All you have to do is drag-and-drop selected elements from the graphic components library in the palette (on left) to the screen (on the right).



few pages, we're going to give here just enough information to encourage our readers to delve into the source code to find the functions described. Experienced programmers will be able to make any modifications or improvements to it they like. And maybe it will motivate others to start developing Android applications too.

As the dynamic scrolling graph demands speed, but the graphic performance of applications developed using *AppInventor* are only mediocre, I had to give up the idea of using this free environment. But I would recommend it for other simpler applications, e.g. controlling a *Mindstorms* robot in BT, or for any of our readers who want to get a taste of programming.
Here, I've used Android SDK from *Google*, also free. The SDK tools (on PC, Mac, or Linux) are included in a popular, free IDE: *Eclipse*. It takes a long time to install it complete, but it's easy enough if you follow the procedure described by *Google*.

You need to be pretty familiar with Java and object-oriented language (like C++). Anyone who already knows how to write programs in C will be able to learn it if they are curious and enjoy making a little effort. There are some excellent available [5] as well as my own document on my website [6].

Developing for Android

Developing an application for an embedded operating system like Android requires good knowledge of its architecture (**Figure 11**). End users only directly access the applications installed on their terminal (the top layer in the illustration).

Developers can use these applications for their own application, but above all has access to a rich collection of APIs (Application Programming Interface) written in Java in order to exploit the tablet's resources. These are grouped together in the Application Framework. These APIs make use of libraries (in C and C++) which themselves rely on a Linux core.

The originality of Android is its execution engine, based on a Dalvik VM virtual machine (= VM). The principle is close to the Java virtual machine (JVM) used in PC and Mac: the Java compiler produces executable files in bytecode independently of the processor used. The VM, specific to each device, then executes the application bytecode which will behave in the same way regardless of the target computer.

Under Android too, the bytecode produced by the compiler can be executed on any terminals, whatever processor they use.
Each Android application runs in its own process, with its own instance of the virtual machine. Dalvik has been written in such a way that a device can run several VMs efficiently.

Composing screens

Composing application screens with the help of SDK Android before you have even written the least line of code – what a gratifying step! Programmers have access to a library of graphic components that they just have to arrange on the screen as they choose (**Figure 12**). The arrows represent some examples of drag-and-drop between the component palette and the screen. The screen already has its final appearance, but the activity doesn't exist, as at this stage not a single line of code has yet been written!

Events

An application in C always includes a *main()* function followed by an endless loop that calls in turn the main functions to be processed, while the architecture of an Android application is event-based. In Java under Android, functions are always executed following an event (touch on the screen, reception of a text message, etc.) and never include endless loops. Even the initialization function, when the application is run, ends at the end of its processing and hands over to Android. The execution engine is then able to take care of the other applications running. Events are managed by the operating system and are easy to use in a development environment.

Activities

An Android application includes as many activities as there are different screens displayed when it is run. Each of these screen is formed from buttons, texts, graphs, the associated processing for which is part of the activity. Since Android is multitasking, an activity can have several states:

- active: currently running
- suspended: paused following a higher-priority event (e.g. displaying a text message)
- stopped: focus shifts to another activity.

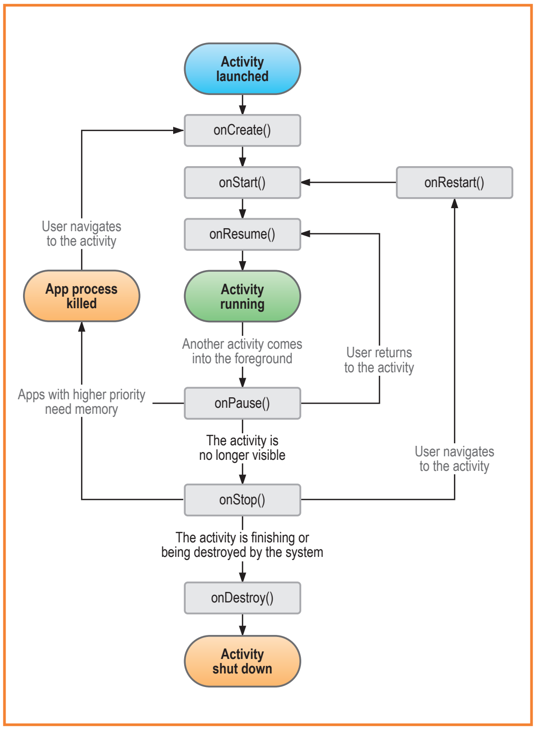

Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Figure 13.
Life-cycle of an activity.

The system remembers its state so it can go back to it, but it can happen that it ends the application to free up system memory.

Figure 13 illustrates the life cycle of an activity, typical in a multitasking system.
Our ANDROECG application comprises three activities:

- **MainActivity** start the application running. It displays the main screen and the control buttons (see screenshots) and sets up the services the application requires.

- **BtListActivity** runs on demand to displays the list of paired BT peripherals and select the one for our interface.
- **FileListActivity** is run on a request to save or read ECG data. It displays the list of existing files, along with an edit window for creating a file.

Services

These are background tasks that do not require either a screen or any action from the user. Services can communicate with activities via Intents. In the ANDROECG application, for example, the ***BluetoothService*** service looks after managing the BT module: establishing the connection,

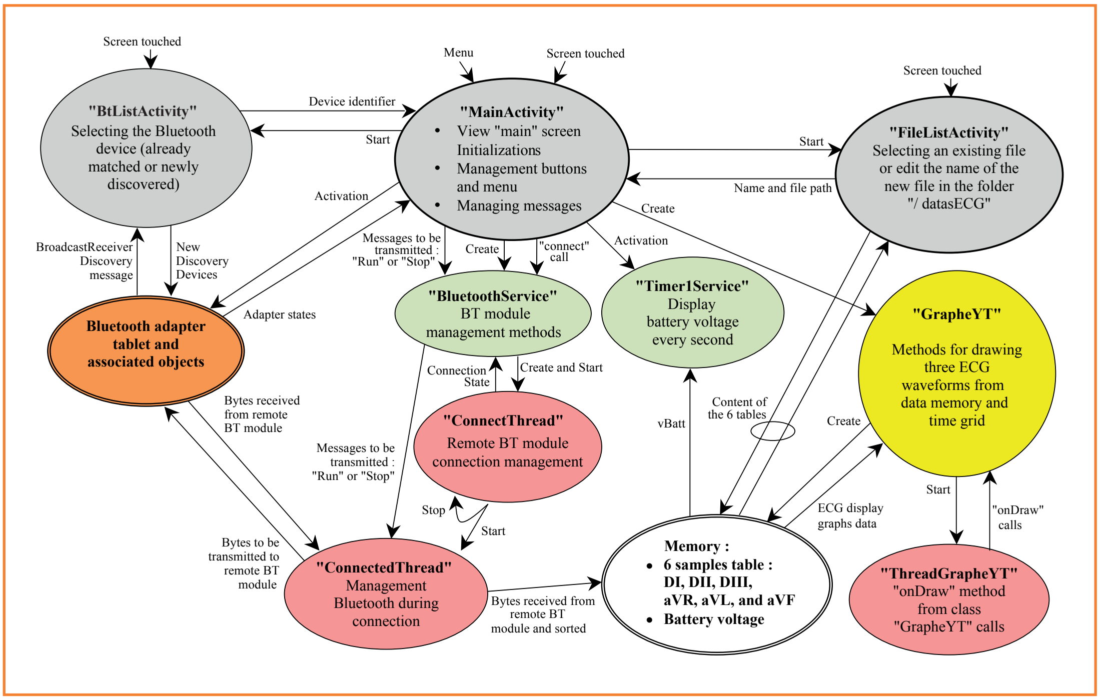

Screen touched
"BtListActivity"
Selecting the Bluetooth device (already matched or newly discovered)
Device identifier
Start
Menu
Screen touched
"MainActivity"
• View "main" screen Initializations
• Management buttons and menu
• Managing messages
Start
Name and file path
Screen touched
"FileListActivity"
Selecting an existing file or edit the name of the new file in the folder "/ datasECG"
Activation
Create
BroadcastReceiver Discovery message
New Discovery Devices
Messages to be transmitted : "Run" or "Stop"
Create
"connect" call
Activation
Adapter states
Bluetooth adapter tablet and associated objects
"BluetoothService"
BT module management methods
"Timer1Service"
Display battery voltage every second
"GrapheYT"
Methods for drawing three ECG waveforms from data memory and time grid
Connection State
Create and Start
"ConnectThread"
Remote BT module connection management
Bytes received from remote BT module
Messages to be transmitted : "Run" or "Stop"
vBatt
Content of the 6 tables
Create
Stop
Start
Start
ECG display graphs data
"onDraw" calls
Bytes to be transmitted to remote BT module
"ConnectedThread"
Management Bluetooth during connection
Bytes received from remote BT module and sorted
Memory :
• 6 samples table : DI, DII, DIII, aVR, aVL, and aVF
• Battery voltage
"ThreadGrapheYT"
"onDraw" method from class "GrapheYT" calls

Figure 14.
Organization of the activities, services, and threads in our application

sending and receiving data, and shutting down the link. The ***Timer1Service*** service is a periodic task responsible for displaying the battery voltage every second.

Under Parameters on your Android phone, the Applications menu gives a list of the services running at any time.

Threads

The thread, or task, is the basis of concurrent programming, which consists in developing an application in which the tasks, from the user's point of view, are running simultaneously. Each task reacts to events (screen click, reception of a BT message, etc.) independently of the others and carries out the associated operations.

Each thread includes a *run()* method (function) which acts rather like the *main()* function in C, except in concurrent programming there are as many *run()*'s as there are threads A service, for example, runs in a thread. When an application is run, Android creates the UI (***User interface***) thread tasked with detecting all the events used by the activity (e.g. button pushes) and acts accordingly.

Each activity or service can create new threads in order to carry out specific processing in them. Our ANDROECG application includes the following additional threads:

- **ThreadGrapheYT** taking care of the scrolling ECG display. For smooth display movement, this is given a high priority.
- **ConnectThread** establishes the connection with the remote BT module.
- **ConnectedThread** manages the current BT link, in particular the reception and transmission of the data.

ANDROIDECG application organization

The organization of our application's activities, services, and threads, along with the links between them (Intents) is less complicated than one might fear at first sight (**Figure 14**). Refer also to the screenshots (**Figure 15**).

MainActivity: Android creates this activity when the application is runs and executes the ***onCreate()*** method (Figure 13). This performs all the necessary initializations and creates, among others, the *BluetoothService* and *Timer1Service* services. The other methods (or functions) of the activity take care of the actions on the touch buttons and the menu functions. The last two methods take care of the messages sent when the other activities close and by the services in order to react accordingly and/or inform the user (e.g. loss of BT connection).

BtListActivity: This activity is created when you press the "Paired BT Devices" menu button (Figure 15). It opens a new window, queries the terminal's BT adaptor, and displays a list of paired peripherals (**Figure 16**). There's a button to let you start a new search. The window and the activity close when the peripheral is selected, and a message including its identifier is sent to the main activity. The main activity then starts *BluetoothService* in order to establish the link with our ECG interface.

BluetoothService: This service is created by the main activity as soon as the BT adaptor is activated. It is responsible for establishing the link and then managing it. To do this, it creates threads:

- **ConnectThread** is started as soon as the peripheral is selected. This thread requests the BT adaptor to establish a connection using the SPP profile. When this is done (this can take several seconds), this thread is deleted before starting the next one.
- **ConnectedThread** remains active the whole time the connection with the ECG interface is open. It comprises in particular the *write* and *run* methods respectively handling transmission and reception of the data being exchanged via BT. The *run* method detects each sequence of samples in the received stream, transmitted by the interface at a rate of 250 Hz (Figure 6) in order to assign in real time each of the 6 sample tables. The size of these tables allow 10 min of cardiac activity to be recorded. The thread is deleted if the link is lost or when the application is closed.

GrapheYT: instanced (that's Java jargon…) in the main activity, this class comprises the variables declarations and methods needed to plot ECGs. We can mention in particular:

- The 6 tables used to store the 10 min of ECG traces
- The *onDraw()* method called periodically by the *ThreadGrapheYT* thread, responsible for plotting the selected ECGs, along with the axes (**Figure 17** in the box).

ThreadGrapheYT starts when the *GrapheYT* class is created, so when the application is launched. Within its *run* method, it includes the call to the *onDraw* method mentioned above. It

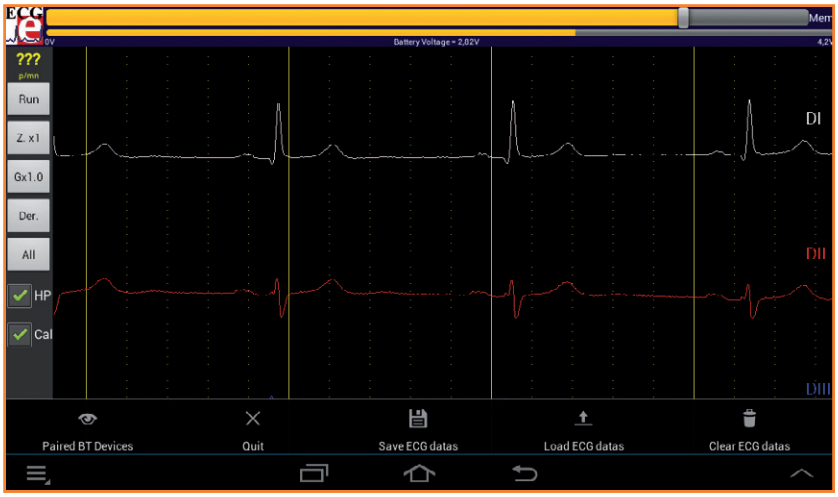
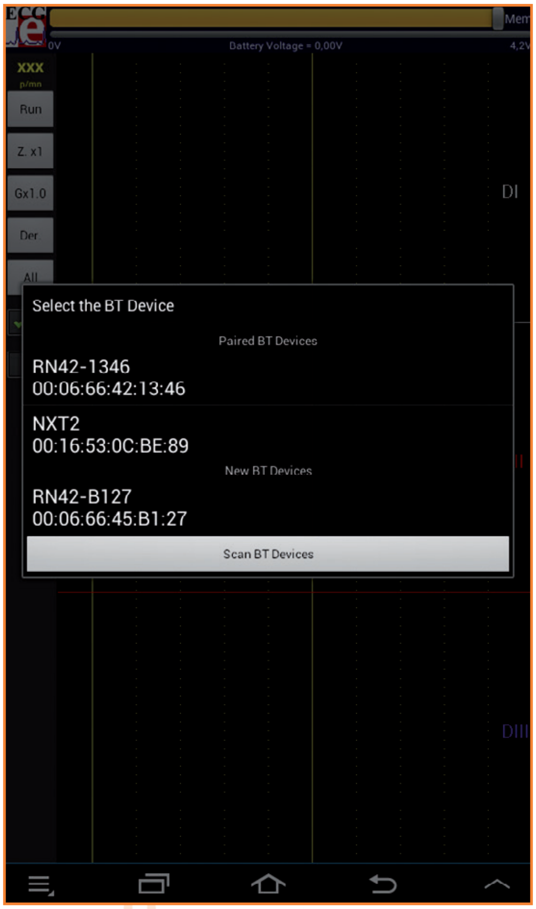


Figure 15.
The menu functions:
BT connection, quit, saver and load ECGs, erase ECG memory.



Select the BT Device

Paired BT Devices

RN42-1346
00:06:66:42:13:46

NXT2
00:16:53:0C:BE:89

New BT Devices

RN42-B127
00:06:66:45:B1:27

Scan BT Devices

Figure 16.
The BtListActivity displays a list of already-paired peripherals, and if so demanded finds new peripherals within range.

ECG graph scrolling refresh algorithm

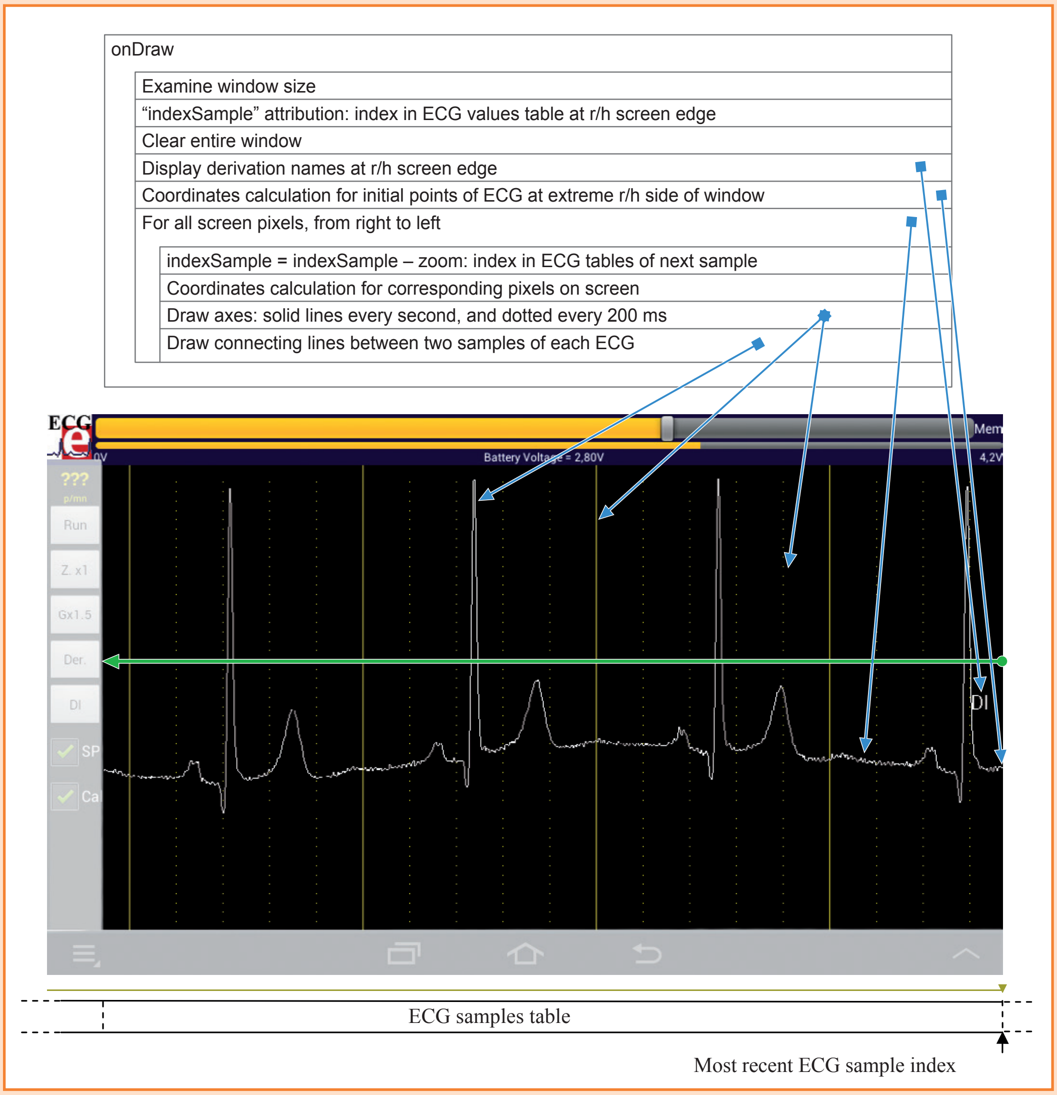
In order to understand the algorithm properly, you need to have firmly in your mind the way the tables storing the last 10 minutes of cardiac activity are used:

- each of the DI, DII, DIII, aVR, aVL and aVF derivations includes its own table with 10 min of samples;
- each of these is assigned by a new ECG sample to each data sequence received by the BT module, i.e. 250 times a second;
- in normal use (Mem cursor on right), the last sample acquired must always be displayed at the extreme right of the screen;
- in order to obtain a dynamic scrolling graph, the display function (*onDraw*) represents the last samples memorized in the tables, starting with the most recent. In a way, it's like going back in time.

Hence the scroll speed is 250 pixels per second (Zoom ×1).

What work are the Android terminal's processors asked to do to display a scrolling ECG? In this example, the size of the ECG graph is 722 × 403 pixels. Under these conditions, each time the *onDraw* method is called, we need to: erase the whole of the screen, i.e. the 722 × 403 = 290,966 pixels, plot the derivation names;

- plot the axes that move with the curves;
- plot up to three ECGs, i.e. for each of the 722 right-hand segments;
- calculate the cardiac rhythm, and display it!



```
onDraw
    Examine window size
    "indexSample" attribution: index in ECG values table at r/h screen edge
    Clear entire window
    Display derivation names at r/h screen edge
    Coordinates calculation for initial points of ECG at extreme r/h side of window
    For all screen pixels, from right to left
        indexSample = indexSample – zoom: index in ECG tables of next sample
        Coordinates calculation for corresponding pixels on screen
        Draw axes: solid lines every second, and dotted every 200 ms
        Draw connecting lines between two samples of each ECG
```

Figure 17. The graph refresh algorithm.

That's all... The number of instructions executed by the processor, helped in some cases by its graphics co-processor, is gigantic. What's more, to obtain smooth scrolling, the frequency at which the *onDraw* method is called must be significantly higher than 10 Hz! Just a few years ago, a large office PC would not have been capable of sustaining that sort of working speed. Today, one of these little wonders that fit in your pocket manages it easily, and also looks after the other active applications...

is given a high priority to ensure smooth movement of the traces. However, the execution frequency of its *run* method is determined by the Android system. If other threads are making heavy demands on the terminal's CPU, the traces may move jerkily.

Timer1Service: This class creates a service that performs a relatively simple task every second: displaying the interface battery voltage in a digital and graphical form (displayed at the top of the screen).

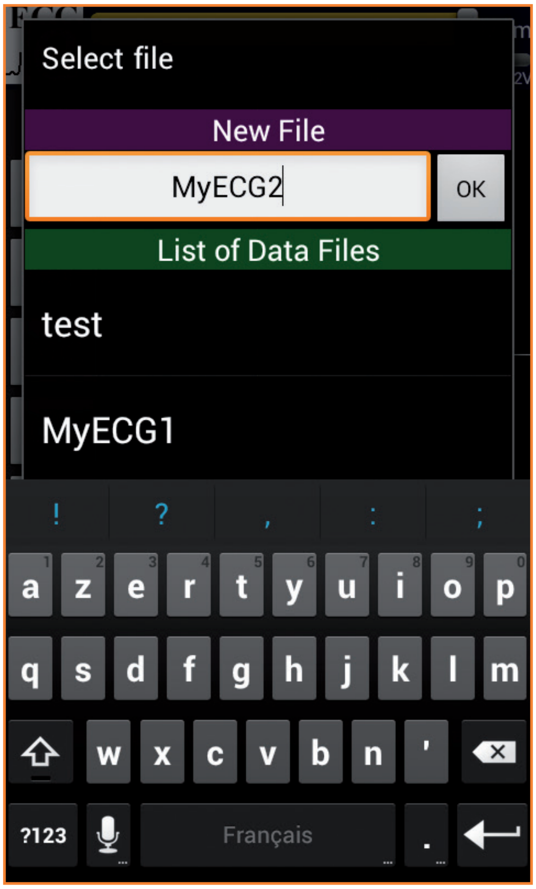
A project to your heart's content

FileListActivity: This activity is created when the user chooses to save or load ECG reports from the menu. It displays the list of existing files, along with an edit window for editing the name of a new file (**Figure 18**). The window and the activity are closed when the file is selected, after a sending a message including its name and the nature of the operation (*save* or *load*) to the main activity. The main activity then performs the operation requested.

My heart is going boom-boom

Without by any means exhausting the subject, we've come to the end of our description of the Elektorcardioscope. Next month we'll finally move onto the practical side with construction, adjustments, and some instructions. For the interface, this will be quite quick, as the finalized circuit board is now available from our ***Elektor PCB-service*** in the form of an assembled, ready-to-use module [7]. The adjustments won't require any special skills either, but we shall be taking a close look at the electrodes. For the intention of this sophisticated device is indeed to bring ECG within everyone's reach.

(130227)


Select file

New File

MyECG2

List of Data Files

test

MyECG1

Figure 18.
Selecting the file for saving or loading ECG reports.

Internet Links

[3] www.elektor.com/120107 and www.elektor.com/130227

[4] Android Apps | programming step by step, by Stephan Schwark www.elektor.com/android

[5] Le Site du Zéro http://www.siteduzero.com/informatique/tutoriels/apprenez-a-programmer-en-java or http://goo.gl/OVZQY (in French)

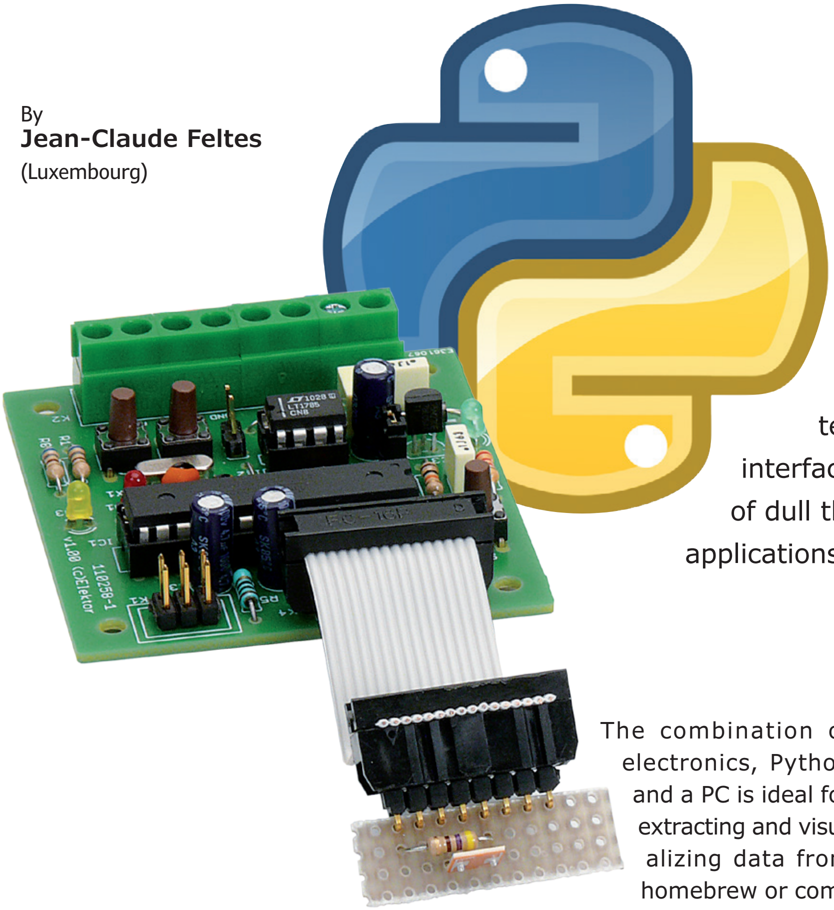
[6] Author's website: http://electronique.marcel.free.fr/

[7] www.elektorpcbservice.com/

From BASIC to Python (3)

Communicating with the ElektorBus

By
Jean-Claude Feltes
(Luxembourg)



In the first two parts of this series we looked at the differences between BASIC and Python in terms of mathematical amusements such as Fourier synthesis, plotting graphs, and graphical user interfaces. In this third part we will describe interfacing to the ElektorBus. We will steer clear of dull theory and concentrate on colorful practical applications.

The combination of electronics, Python and a PC is ideal for extracting and visualizing data from homebrew or commercial external hardware. And, with the data flowing in the opposite direction, we have the perfect way to control external hardware. The communication itself can be implemented using a common-or-garden serial interface or, if something a little more sophisticated is desired, over a bus. The ElektorBus makes for a good demonstration of how such as system can be put together.

The ElektorBus was described in an eleven-part series of articles [1] in Elektor magazine between January 2011 and January 2012. The hardware we will be using here is the experimental node, a board carrying a microcontroller, LEDs and buttons described in part 6 of the series [2]. The board can be connected to a PC using the USB-to-RS-485 converter that was also described at the time (**Figure 1**).

Hexadecimal help

First a little warm-up exercise: a script that provides an auxiliary function to display values in hexadecimal and that shows how reusable code can be encapsulated in a module. Data on the ElektorBus are conveyed in binary, and so some of the bytes in messages correspond to non-printing ASCII characters: this is not ideal for display in a terminal window. Instead, it is preferable to display received data bytes in hexadecimal format. The code in **Listing 1** from the file 'Hexfunctions.py' is what we need. The function implemented here can also be used by other programs, and it can of course be extended if desired. In traditional Python style the module includes its own test function. If it is run directly the variable '__name__' will have the value '__main__' and the second part of the code will be executed. The result is that an example string will be converted into hexadecimal and displayed:

```
HELLO
48 45 4C 4C 4F 0A
```

The example 'Test_hexfunctions.py' shows how the module can be used in another program:

```
from hexfunctions import *
s=”HELLO\n”
print s, translate2hex(s)
```

The imported module must be in the same directory as the main program or be found somewhere on Python’s search path. Putting functions in modules like this makes code easier to understand at a glance and easy to reuse.

A GUI with wxPython

Many roads lead to an attractive graphical user interface. However, Python does not really have anything as easy to use as Visual Basic’s ‘Forms Designer’, which gives full control over the appearance of the result and which works well when incorporated into the final program. I first tried to use ‘Boa Constructor’, but debugging seemed very tedious without a detailed background understanding. ‘Python Card’ is rather simpler, and very practical for straightforward applications. Unfortunately, however, it requires Python Card to be installed on each target computer.
‘Tkinter’ is the GUI library installed with Python. It is very simple, and includes fewer objects than wxPython. Since I needed to use the clipboard in my application, I settled on the more complete and powerful wxPython.

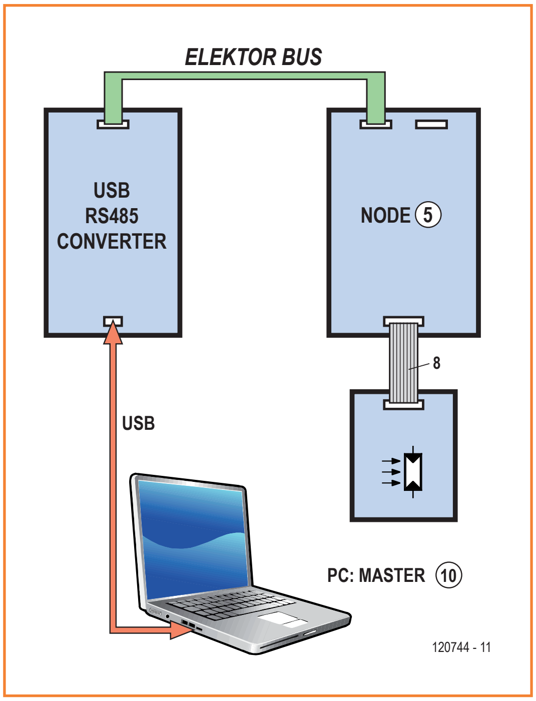


Figure 1.
The PC receives data over the (RS-485) ElektorBus from a small microcontroller board to which a light sensor is connected.

The code in **Listing 2** gives the basic framework for the graphical interface, which can be extended step-by-step into a complete application. Of course, you must install wxPython to run it.
The main window is defined, in object-oriented fashion, as a class ‘MyFrame’, which inherits all the properties from the class ‘wxFrame’, includ-

Listing 1: Hexfunctions.py

```
def translate2hex(c):
    “”” translate character string c to hex representation string
    e.g ABC -> 41 42 43 “””

    h=””
    for ch in c:                # iterate over all characters
        b=hex(ord(ch))          # get hex value
        b=b.replace(“0x”,””)    # take away leading “0x for better overview”
        b=b.upper()             # all in upper characters
        if len(b)<=1:
            b=”0”+b             # e.g. make “0A” out of “A”
        h=h+b+” “               # separate bytes by space

    return h

# test:
if __name__ == “__main__”:
    s=”HELLO\n”
    print s, translate2hex(s)
```

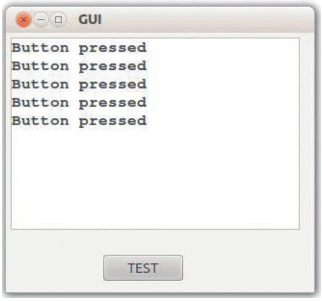
Listing 2: GUI_template.py

```
import wx

#  GUI
class MyFrame(wx.Frame):

    def __init__(self, **kwargs):
        # create frame
        wx.Frame.__init__(self, None, **kwargs)

        # text box with fixed width font for nice data representation
        self.textbox=wx.TextCtrl(self, style = wx.TE_MULTILINE,
            pos = (5,5),size=(300, 200))
        myfont = wx.Font(12, wx.MODERN, wx.NORMAL, wx.BOLD, False, u’Courier’)
        self.textbox.SetFont(myfont)

        self.button=wx.Button(self, -1, “TEST”, pos=(100,230))

        # Bindings
        self.Bind(wx.EVT_IDLE, self.OnIdle)
        self.Bind(wx.EVT_WINDOW_DESTROY, self.OnDestroy)
```

Listing 3: Serialthread class

```
class Serialthread(serial.Serial):
    def __init__(self, port, baud, **kwargs):
        # Initialization of port + baudrate
        serial.Serial.__init__(self)
        self.sCOM =serial.Serial(port)
        self.sCOM.setBaudrate(baud)

        # open port if not already open
        if self.sCOM.isOpen()==False:
            self.sCOM.open()
        if self.sCOM.isOpen()==True:
            print „connected to“, self.sCOM.port
        else:
            print „Error opening port“

        # Counter for received data blocks
        self.ctr=0

        # Create stop event (to terminate endless receiving loop)
        # and message queue for thread (to transmit received text to TextCtrl)
        self.stopevent=threading.Event()
        self.msgQueue=Queue.Queue()

    def disconnect(self):
        # set stop event so endless receiving loop can be interrupted
        self.stopevent.set()

    def connect(self):
        # create a new thread object that runs serial thread
        self.Bind(wx.EVT_BUTTON, self.OnButton)

    def OnIdle( self, event):
        # if nothing else to do, update text from message queue
        pass

    def  OnDestroy(self, event):
        print “Exit”

    def OnButton(self, event):
        self.textbox.AppendText (“Button pressed\n”)
#------------------------------------------------------------------------

# Main program
if __name__ == “__main__”:
    app = wx.App(redirect = False)
    frame = MyFrame(title=”GUI”, size = (320,270))
    frame.Show(True)
    frame.Centre()
    app.MainLoop()
```

```
        # to read serial characters
        self.serialthread = threading.Thread(target=self.readSerial)

        # clear stopevent and Connect thread
        self.stopevent.clear()
        self.serialthread.start()

    def readSerial(self):
        # endless receiving loop
        while not self.stopevent.isSet():
            data=““

            # read from port
            c = self.sCOM.read(1)

            # synchronize
            if ord(c) == 0xAA:
                self.ctr += 1
                rest = self.sCOM.read(15)
                data=c+rest

            # format c to 16 bytes output
            datastring=str(self.ctr) + „\t“ + translate2hex(data) + „\n“

            # update message queue
            self.msgQueue.put(datastring)
            wx.WakeUpIdle()         # wake up to update text

        # end serial thread
        print „disconnected“
        self.sCOM.close()
```



```
Button pressed
Button pressed
Button pressed
Button pressed
Button pressed
```

Figure 2.
The window created by the code in Listing 2.

ing functions to increase and reduce the size of the window, to move the window, and so on.
The properties of the window are specified in the function '__init__()'. This is where the graphical elements are defined: in this case a text box and a button. In the interests of simplicity the sizes and positions of these elements are fixed. With the help of so-called 'sizers' it is possible to create a dynamic layout.
The next step is to create three event handlers, although two of these will not be not needed until later. Traditionally the name of an event handler starts with 'On'. The 'Bind()' function attaches these functions to specified events: for example, 'OnButton()' is called in response to a button press.

With the class 'MyFrame' defined it can be used in the main program. This first creates a 'wx.App' object, which is mostly concerned with the processing of events. We then create an instance of our window, centered and made visible on the screen. Events are then handled in the infinite loop 'app.MainLoop()'.
When the program is run the window appears as shown in **Figure 2**. It already responds to the button being pressed and even supports extensive editing functions: pressing the right mouse button brings up a menu for selecting, copying, deleting and inserting text.
The program framework can easily be extended, for example to include a function for storing text in a file.

ElektorBus: read operations

The experimental node is connected to the PC via the USB-to-RS-485 converter. We fit an ATmega328 microcontroller to the small printed circuit board and program its flash memory with the hex file downloaded from [4]. A potentiometer or light-dependent resistor can be connected to the ADC0 pin on the expansion connector, as described in [2a] and [2b].
Before proceeding it is necessary to identify the correct serial port on the PC: in the Windows device manager look to see which new COM port appears when the converter is plugged in. Linux users can use the following command

```
  ls /dev/tt*U*
```

which will give a result similar to the following:

```
  /dev/ttyUSB0
```

The Python script for scanning the serial ports from part 1 of this series [3] can also be used.
If any of the scripts given below fails to work, the first thing to check is that the serial port is configured correctly. While experimenting it is possible under some circumstances for the operating system to surreptitiously change the port number. For example, this can happen if a script is using /dev/ttyUSB0 and the converter is unplugged and then plugged in again: the converter will then be assigned to /dev/ttyUSB1, and the script will not be able to receive data. In normal use this very rarely happens.
We can now start to extend the GUI framework in Listing 2. First we have to load all the modules required and set the various interface parameters:

```
COMport = “/dev/ttyUSB0” # adjust to suit
Baud = 9600

import threading, Queue
import serial
import time
```

The module 'threading' is needed because serial data reception is carried out in a separate thread from the main code. This in turn is because the receiver needs to run in an infinite loop and this would create a conflict with the main loop in 'wx'. Indeed, this is the most complicated aspect of the program.
We create a special class called 'Serialthread' to access the serial interface (**Listing 3**). An object in this class opens the interface, reads the incoming data bytes, formats them and sends the result via a message queue to the other parts of the program. This runs in an independent infinite

loop until the thread is stopped.

A ‘Serialthread’ object inherits all the properties and methods of the base class ‘Serial’: the port name, baud rate, functions for reading and writing, and so on. The procedure ‘__init__()’ creates a Serial object that implements all the port operations. The port is opened if it is not already open, and the baud rate is configured.

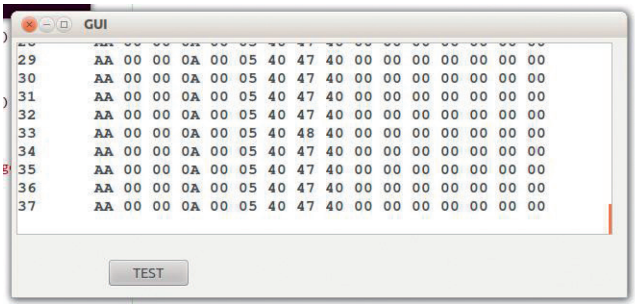
The counter ‘self.ctr’ is not important for now, but later will be used to number the data blocks. There are two further objects of interest used by the serial thread: a stop event to cause the thread to terminate and a message queue to send data to the GUI.

The external interface for starting and stopping the thread is via the methods ‘connect()’ and ‘disconnect()’. The ‘disconnect()’ method simply sets the stop event. The ‘connect()’ method launches a new thread in which the function ‘readSerial()’ is executed. This function in turn continuously reads bytes from the interface in an infinite loop until the stop event is received. The thread then terminates and the port is closed.

Within ‘readSerial()’ there is a mechanism for synchronizing to the incoming data. In the ElektorBus protocol each data packet starts with an 0xAA byte. When this value is seen, the data packet counter is incremented and a further fifteen bytes are read in. These are then formatted into a string and placed on the message queue. The function ‘wx.WakeUpIdle()’ is then called to signal to the GUI part of the program that there is an entry in the message queue that can be read when there is no other work to be done.

The ‘Serialthread’ class thus allows data to be read continuously without affecting the execution of other code. In effect it runs in parallel with the other parts of the program. The class must be instantiated and started from the main program, which means that we have to extend the code in ‘MyFrame’, which corresponds to the main window, created in the GUI framework script, by adding the following code (after the bindings) to the procedure ‘__init__()’:

```
class MyFrame(wx.Frame):

    def __init__(self, **kwargs):
        ...

        # Bindings
        ...
```



```
29        AA 00 00 0A 00 05 40 47 40 00 00 00 00 00 00 00
30        AA 00 00 0A 00 05 40 47 40 00 00 00 00 00 00 00
31        AA 00 00 0A 00 05 40 47 40 00 00 00 00 00 00 00
32        AA 00 00 0A 00 05 40 47 40 00 00 00 00 00 00 00
33        AA 00 00 0A 00 05 40 48 40 00 00 00 00 00 00 00
34        AA 00 00 0A 00 05 40 47 40 00 00 00 00 00 00 00
35        AA 00 00 0A 00 05 40 47 40 00 00 00 00 00 00 00
36        AA 00 00 0A 00 05 40 47 40 00 00 00 00 00 00 00
37        AA 00 00 0A 00 05 40 47 40 00 00 00 00 00 00 00
```

Figure 3.
Received data displayed in hexadecimal.

```
        # serial thread
        self.serialreceive =
Serialthread(COMport, Baud)
        self.serialreceive.connect()
```

We now have an object called ‘serialreceive’ which is connected to the port specified by the variable ‘COMport’ and which writes all incoming data to the message queue. To see the results we have to extract the text from the message queue and write it to the text box. It is at this point that we make use of the function ‘OnIdle()’ that we prepared earlier:

```
class MyFrame(wx.Frame):

    def __init__(self, **kwargs):
        ...

    def OnIdle( self, event):
        # if nothing else to do, update
text from message queue
        while not self.serialreceive.
msgQueue.empty():
            msg=self.serialreceive.
msgQueue.get()
            self.textbox.AppendText(msg)
```

The ‘pass’ command was simply a placeholder and can now be deleted. With these changes the incoming data should now be displayed in the text box with incrementing packet numbers as shown in **Figure 3**.

To avoid the appearance of unsightly error messages when closing the window, an extra touch is to add an ‘OnDestroy()’ procedure:

```
def OnDestroy(self, event):
    self.serialreceive.disconnect()
    time.sleep(1)
```

This will ensure that before the window closes the serial thread is forced to finish. This can take a moment, which explains the need for the ‘time.sleep(1)’ command. The end result of all the modifications we have described is the new program ‘Serialreceive1.py’, which, as usual, is available for free download from the *Elektor* web pages accompanying this article [4].

In the interests of modularity and code clarity the definition of the class ‘Serialthread’ can be stored separately from the module ‘Serialthread.py’, as soon as no further changes are planned. It can then be imported into the main program.

This requires a further small modification, as the modules required in turn by ‘Serialthread’ have to be imported from within that module rather than in the main program. The file ‘Serialthread.py’ therefore also needs to include the following code:

```
import threading, Queue
import serial
import time
from hexfunctions import *
import wx

class Serialthread(serial.Serial):
    def __init__(self, port, baud,
**kwargs):
        # Initialization of port +
baudrate
        serial.Serial.__init__(self)
        ....
        # end serial thread
        print “disconnected”
        self.sCOM.close()
```

The resulting main program ‘Serialreceive2.py’ now dispenses with the entire definition block for the class ‘Serialthread’. The ‘import’ lines are also no longer required, as they now appear in ‘Serialthread.py’.

It must be admitted that this division of the code is not ideal, as the module ‘Serialthread.py’ is written in a way rather specific to the project rather than being aimed at more general use.

ElektorBus: write operations

The experimental node includes a red LED. We would like to be able to turn this LED on and off using two buttons on the PC. The byte sequences required on the ElektorBus are as follows:

- Turn on: AA 00 00 05 00 0A 00 00 00 00 **60 01** 00 00 00 00
- Turn off: AA 00 00 05 00 0A 00 00 00 00 **60 00** 00 00 00 00

We will need to add a second button and a second event handler to the GUI program, and give the buttons appropriate labels. To do this we edit ‘__init__()’ in the ‘MyFrame’ class as follows:

```
    def __init__(self, **kwargs):
        # create frame
        ....
        buttonOn=wx.Button(self, -1, “LED
ON”, pos=(100,230))
        buttonOff=wx.Button(self, -1,
“LED OFF”, pos=(200,230))

        # Bindings
        ....
        buttonOn.Bind(wx.EVT_BUTTON,
self.OnButtonOn)
        buttonOff.Bind(wx.EVT_BUTTON,
self.OnButtonOff)
```

The event handler calls the functions ‘self.OnButtonOn()’ and ‘self.OnButtonOff()’. These have to be expanded:

```
    def OnButtonOn(self, event):
        self.textbox.AppendText (“LED
ON\n”)
        data=b”\xAA\x00\x00\x05\x00\x0A\
x00\x00\x00\x00\x60\x01\x00\x00\x00\x00”
        self.serial_thread.sCOM.
write(data)

    def OnButtonOff(self, event):
        self.textbox.AppendText (“LED
OFF\n”)
        data=b”\xAA\x00\x00\x05\x00\x0A\
x00\x00\x00\x00\x60\x00\x00\x00\x00\x00”
        self.serial_thread.sCOM.
write(data)
```

If you look at the source code for the ‘Serialthread’ class, you might wonder where the function ‘write()’ is defined. What is happening is that the base class ‘serial.Serial’ defines this function which the derived class inherits: it therefore does not need to be defined explicitly in the derived class. The result of these changes is the program ‘Receive_send.py’, which allows the red LED on the experimental node to be turned on and off from the PC. The software is not yet perfect: in particular it does not implement synchronization with the node. If the node and the PC happen to transmit at the same time, a collision occurs on the bus and data will be lost. This can be avoided by having the PC only transmit a message on the bus immediately after a packet has been received from the node (the ElektorBus ‘direct mode’: see the ElektorBus specification at [1]). For example, a flag could be set to indicate that a message is to be sent but the actual sending of the appropriate byte sequence would be delayed until the next data packet is received.

Graphs

Next we would like to convert the incoming data from textual to graphical form, so that we can for example visualize the output of the A/D converter on the node.

The standard library for creating graphical output is called ‘Matplotlib’, and you will need to install this library before proceeding. In the first part of this series we used the simple ‘pyplot’ interface which provides a quick and easy way to generate graphs. Things get more complicated if we want to embed a graph in a GUI as we have to use the object-oriented interface, which has rather more options. Indeed, the range of options offered can be overwhelming at first. The documentation for Matplotlib can be found at the project homepage [5], and an example of embedding it in a ‘wx’ interface can be found at [6].

Before embarking on changes to the ‘Serialthread’ module make a copy of the code under a different name, for example ‘Serialthread_diagram.py’. Now, the values from the A/D converter that we need are in bytes 5 and 6 of the received data packet. The values have to be extracted and then passed to the main program. There are several different ways this might be done: one approach is to add arrays ‘x’ and ‘y’ to the ‘Serialthread’ object as attributes to carry the values. The extra code in ‘__init__()’ is then as follows:

```
class Serialthread(serial.Serial):
    def __init__(self, port, baud,
**kwargs):
        ...

        ## init arrays and timer for data
        self.x=[]
        self.y=[]
        self.starttime=time.time()
        ...
```

This sets up two empty arrays for the x and y values. The additional attribute ‘starttime’ allows the total run time so far to be calculated, so that the x-axis of the graph can be labeled in seconds rather than by packet counts.

The receive function is modified to extract the A/D converter data, calculate the value and the timestamp, and place these in the ‘x’ and ‘y’ arrays:

```
    def readSerial(self):

        # infinite receiving loop
        while not self.stopevent.isSet():
            ...
            # synchronize
            if ord(c) == 0xAA:
                self.ctr += 1
                rest = self.sCOM.read(15)
                data=c+rest

                ## update x,y
                lbyte = ord(rest[6])
                hbyte = ord(rest[5]) & 7
                adc  =  lbyte + hbyte
*256
                t=time.time()-self.
starttime
                self.x.append(t)
                self.y.append( adc)
                print t,adc

                # format c to 16 bytes
output
                ....
                wx.WakeUpIdle()         #
wake up to update text
```

With these modifications the module provides the values for graphing in addition to the hexadecimal output.

Now let us turn to the changes in the main program. The name of the ‘Serialthread’ module has changed:

```
from serialthread_diagram import  *
```

To draw the graph we have to import the necessary libraries:

```
from matplotlib.backends.backend_wxagg
import FigureCanvasWxAgg as FCanvas
from matplotlib.figure import Figure
```

Take care with the indentation when copying from the listings: incorrect indentation will result in errors or strange program behavior.

Matplotlib works with a number of 'back ends' which actually carry out the drawing commands. Different back ends plot graphs on the screen, for example with wx, or output to printers, or to files. The first line above creates an object 'FCanvas' for wx on which Matplotlib can draw. 'FCanvas' inherits methods and properties such as size and position from 'wxPanel'.

The object 'Figure' is a container within which drawing occurs. However, 'Figure' is not the graph itself: instead, the graph is an 'Axes' object, and a 'Figure' object can contain one or more 'Axes' objects. This might seem bewildering, but it is all part of the power of Matplotlib.

Some changes are also needed in the layout and in text sizes to obtain a satisfactory appearance: details can be found in the Listing.

The graph window (see **Figure 4**) is constructed in the '__init__()' function of the 'MyFrame' class:

```
class MyFrame(wx.Frame):

    def __init__(self, **kwargs):
        # create frame
        wx.Frame.__init__(self, None,
**kwargs)

        # text box with fixed width font
for nice data representation
        self.textbox=wx.TextCtrl(self,
style = wx.TE_MULTILINE,
                pos = (5,5),size=(420,
200))
        myfont = wx.Font(10, wx.MODERN,
wx.NORMAL, wx.BOLD, False, u'Courier')
        self.textbox.SetFont(myfont)

        buttonOn=wx.Button(self, -1, "LED
ON", pos=(100,230))
        buttonOff=wx.Button(self, -1,
"LED OFF", pos=(200,230))

        ## diagram
        self.figure = Figure()
        self.axes = self.figure.
add_subplot(111)
        self.canvas = FCanvas(self, -1,
self.figure)
        self.canvas.SetPosition( (450,5))
        self.canvas.SetSize((300,250))
        ....
```

First we create a subplot within the 'Figure' instance. One 'Figure' can contain several subplots, for example if there are several time series to display. The syntax is as follows:

```
figure.add_subplot(numrows, numcols,
fignumber)
```

In our case there is only one graph (fignumber = 1) and hence only one row (numrows = 1) and one column (numcols = 1). The last three lines above the drawing surface for the wx back end is created using the 'FCanvas' object. It is then positioned and its size set. We are now in a position to draw a graph.

The second change affects the function 'OnIdle()'. This is called whenever the program has nothing else to do, in particular when new data become available.

```
def OnIdle( self, event):
        # if nothing else to do, update
text from message queue
        while not self.serial_thread.
msgQueue.empty():
            msg=self.serial_thread.
msgQueue.get()
            self.textbox.AppendText(msg)

            ## display values in diagram
            self.axes.plot(self.serial_
thread.x, self.serial_thread.y)
            self.canvas.draw()
```

This code plots the most recently received x and y values: recall that the 'x' array holds timestamps in seconds. The graph is finally displayed using the command 'canvas.draw()'.

The changes described above lead to the program 'Diagram.py'. In this simple example we have not included code to fix the scaling of the graph: instead the scaling continuously changes to match the data displayed. Another side-effect of the simple implementation is the continually-changing color of the curve each time a new graph is plotted. If you wish to modify the presentation to suit your particular requirements, you will need to immerse yourself in the Matplotlib documentation.

Conclusion

You should now have an initial impression of the basic principles involved in creating a GUI. The software we have described is far from perfect: error handling is rudimentary and if anything does go wrong the interpreter will simply report the error immediately.

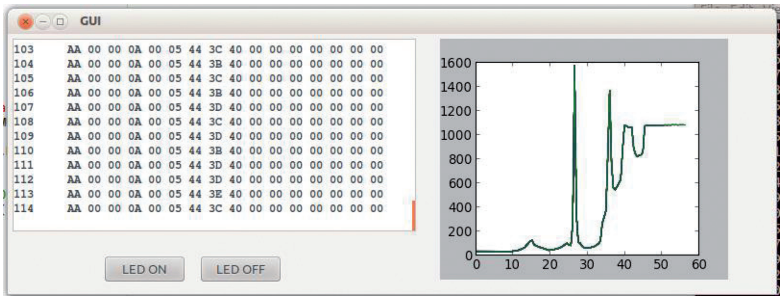
A further disadvantage is that the program can only run for a certain length of time. The problem is that the experimental node will continue to deliver data twice a second and eventually the 'x' and 'y' arrays will become full. Drawing the graph takes longer as more data points accumulate, and if new data points arrive while the graph is being drawn then Matplotlib will no longer be able to keep up. The window will go gray, indicating that the GUI thread is overloaded. It is interesting to note that the terminal window still displays the activity in the receiver thread, since this continues to run independent of the GUI thread.

It should be possible to work around this problem by modifying the program so that the graph is only redrawn every few seconds. Why not give it a try?

(120744)

The Author

Jean-Claude Feltes is an electronics engineer teaching at the Lycée Technique des Arts et Métiers in Luxembourg, a specialist technology and arts school that offers courses leading a range of vocational qualifications. In his spare time he keeps himself busy with electronics and programming projects (see [7]).



```
103       AA 00 00 0A 00 05 44 3C 40 00 00 00 00 00 00 00
104       AA 00 00 0A 00 05 44 3B 40 00 00 00 00 00 00 00
105       AA 00 00 0A 00 05 44 3C 40 00 00 00 00 00 00 00
106       AA 00 00 0A 00 05 44 3B 40 00 00 00 00 00 00 00
107       AA 00 00 0A 00 05 44 3D 40 00 00 00 00 00 00 00
108       AA 00 00 0A 00 05 44 3C 40 00 00 00 00 00 00 00
109       AA 00 00 0A 00 05 44 3D 40 00 00 00 00 00 00 00
110       AA 00 00 0A 00 05 44 3B 40 00 00 00 00 00 00 00
111       AA 00 00 0A 00 05 44 3D 40 00 00 00 00 00 00 00
112       AA 00 00 0A 00 05 44 3D 40 00 00 00 00 00 00 00
113       AA 00 00 0A 00 05 44 3E 40 00 00 00 00 00 00 00
114       AA 00 00 0A 00 05 44 3C 40 00 00 00 00 00 00 00
```

Figure 4.
Terminal window with packet counts and a graphical display.

Internet Links and References

[1] ElektorBus web site: http://www.elektor.com/elektorbus

[2a] Here comes the bus! (6): http://www.elektor-magazine.com/110258

[2b] Here comes the bus! (8): http://www.elektor-magazine.com/110428

[3] From BASIC to Python (1): http://www.elektor-magazine.com/110483

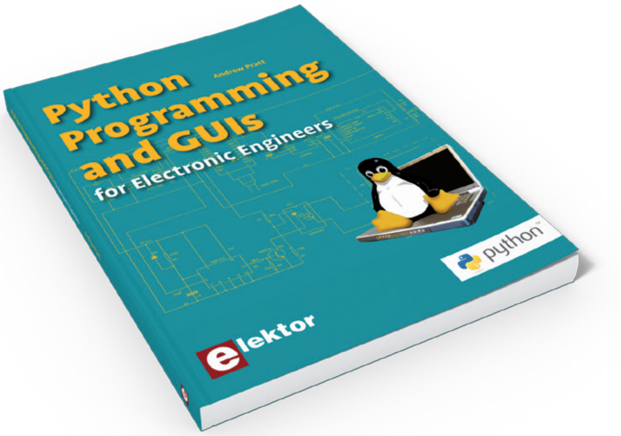
[4] From BASIC to Python (3): http://www.elektor-magazine.com/110483

[5] Matplotlib documentation: http://matplotlib.org/contents.html

[6] Sandro Tosi: Matplotlib for Python Developers

[7] Author's home page: http://staff.ltam.lu/feljc/home.html

[8] 'Python Programming and GUIs for Electronic Engineers' by Andrew Pratt: http://www.elektor.com/products/books/programming/python-programming-and-guis-for-electronic.1320886.lynkx



Using Libraries

By **Neil Gruending**
(Canada)

So far the previous DesignSpark Tips and Tricks articles have discussed how to setup and configure DesignSpark from a new installation perspective. In this installment we will look at how to use libraries to create a schematic and PCB design in DesignSpark.

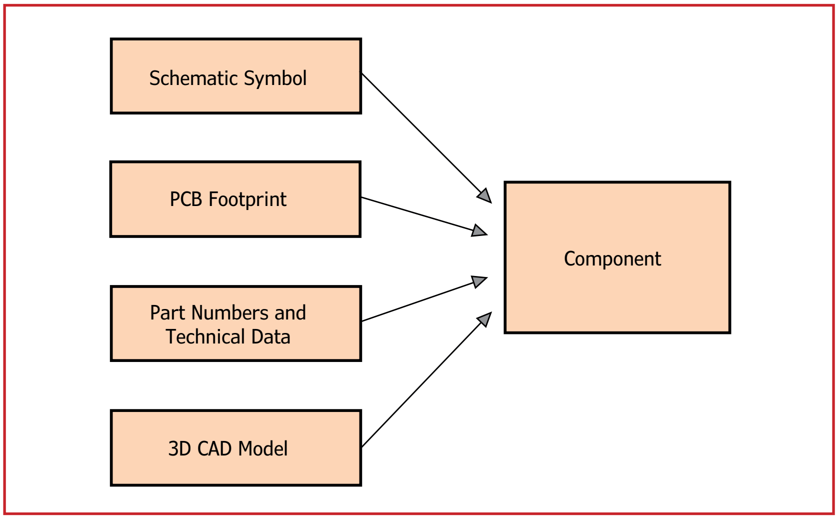


Figure 1.
Properties of a component the design software needs.

What are libraries?

When we created schematic title blocks we started by creating a schematic symbol in the schematic symbol library and then created a component in the component library that referenced that schematic symbol. That was an example of a schematic documentation component that didn't need any PCB design information associated with it. But normally you would probably want to create components that would incorporate the information as shown in **Figure 1**.

The schematic symbol is stored in a schematic symbol library file (*.ssl), the PCB footprint is stored in a PCB symbol library and the 3D CAD model is stored in a 3D view library file (*.pkg). The top level component is stored in a component library (*.cml) along with the part number and all the technical information. The component library also stores the references to the other libraries needed to complete the component.
DesignSpark uses different files for the different types of library data so that it's easy to reuse the design information in multiple components. For example, you could make multiple resistor components by creating one schematic symbol and then reusing it in the other components. Because all the components refer to the same symbol, any changes to it automatically propagate to the components that use it. The same is also true for PCB footprints and the 3D CAD models. The DesignSpark website has a good tutorial about libraries and how they're used, see [1].

Organizing them

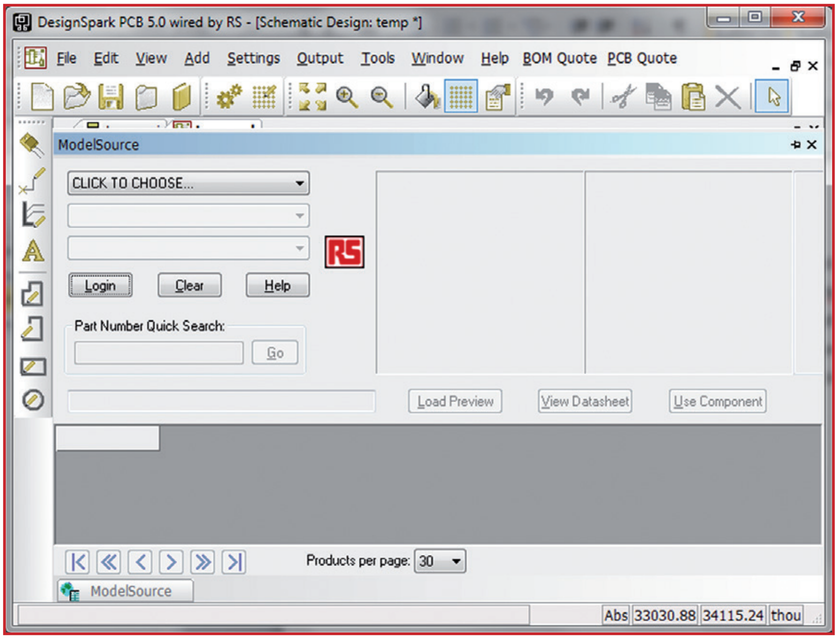
The libraries that come with DesignSpark are usually installed into 'C:\Users\Public\Documents\DesignSpark PCB 5.0\Library' and are a good example of how to organize a large component library organized by manufacturer. I prefer to organize my libraries by component type because I also use the libraries as a component part number database. For example I have a 2N3904 in my transistor library, but I have multiple manufacturer part numbers associated with it so I don't have to remember which transistor manufacturers I've used previously. I also try and reuse schematic symbols and PCB footprints so those go into generic libraries which make my library structure like this:

- Grouped component libraries (transistors, resistors, capacitors, etc)
- A generic schematic symbol library (resistor symbol, capacitor symbol, etc)
- A generic surface mount PCB footprint library (0603 footprint, LQFP footprints, etc)
- A generic through hole PCB footprint library (DIP footprint, 1/4W resistor footprint, etc)

So now that we've talked about libraries, let's learn how to use them starting with Modelsource.

ModelSource

If you haven't heard, ModelSource is an online database of components that's available to use in many different PCB software packages including DesignSpark. I like that DesignSpark directly connects to ModelSource so you can find components without leaving the application (a tutorial is available at [2]). It's also a great resource to find


DesignSpark PCB 5.0 wired by RS - [Schematic Design: temp *]
File Edit View Add Settings Output Tools Window Help BOM Quote PCB Quote
ModelSource
CLICK TO CHOOSE...
RS
Login
Clear
Help
Part Number Quick Search:
Go
Load Preview
View Datasheet
Use Component
Products per page:
ModelSource

Products per page: 30

Abs 33030.88 34115.24 thou

Figure 2.
The ModelSource screen just after opening it.

IPC compliant PCB footprints which meet standard manufacturing guidelines. To open ModelSource in DesignSpark, click on the ModelSource button or go into the 'View->ModelSource Bar' and you will see the ModelSource screen shown in **Figure 2**.

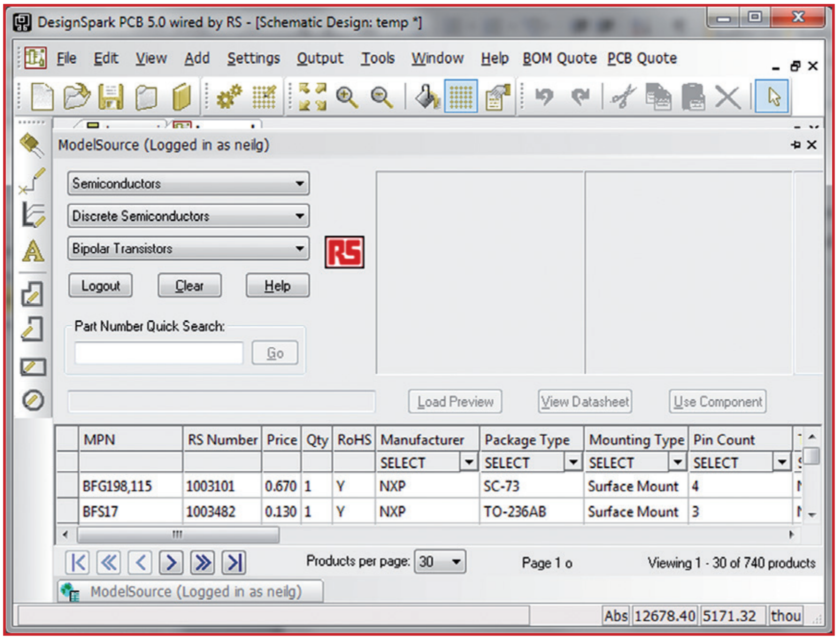
Let's search for a surface mount MMBT3904 NPN transistor using the parametric search engine. Click on 'CLICK TO CHOOSE' and login if necessary. To find a list of the available bipolar transistors by choosing 'Semiconductors->Discrete Semiconductors->Bipolar Transistors', which lists 740 different transistors like in **Figure 3**.

Now let's narrow down the search results by adding some filters to the columns of data. For 'Transistor Type' select 'NPN', 'Mounting Type' to 'Surface Mount', 'Package Type' to 'SOT-23' and 'Maximum Collector Emitter Voltage' to '40V'. The second transistor listed is an MMBT3904 which is exactly what we were looking for. After pressing the 'Load Preview' button you will get the following screen where ModelSource will show you the schematic symbol, PCB footprint and some key component design parameters (see **Figure 4**). You can also find components using the 'Part Number Quick Search' field if you already know a portion of the part number. Now that we've found our transistor click on 'Use Component' to use the component in your design and DesignSpark will download the component to a library in the downloaded libraries directory (you can find the full path in the Folders tab in the Library Manager). DesignSpark will tell you the name of the library after the component is downloaded. You can now add the transistor to your design by dragging it from the ModelSource window into your design or you can use the usual 'Add Component' toolbar button.

But what do you do when ModelSource doesn't have the part you want or you want to change something about the component? For example, I would change the MMBT3904 component we found to have a more conventional schematic symbol that shows the emitter. That's when it's time to use your own custom libraries.

Custom libraries

I always like to create my own set of libraries but that can be a lot of work. So I like to copy components from other sources when possible and then modify them. For our MMBT3904 example that would mean copying the downloaded component information into our own libraries using the Library Manager and then editing the component as required. This is also a good time to double check everything in case there's an error. The most important part of setting up your own libraries is to use common attributes for every component, so that it's possible to generate


DesignSpark PCB 5.0 wired by RS - [Schematic Design: temp *]
File Edit View Add Settings Output Tools Window Help BOM Quote PCB Quote
RS
Logout
Clear
Help
Part Number Quick Search:
Go
Load Preview
View Datasheet
Use Component
Products per page:

ModelSource (Logged in as neilg)

Semiconductors

Discrete Semiconductors

Bipolar Transistors

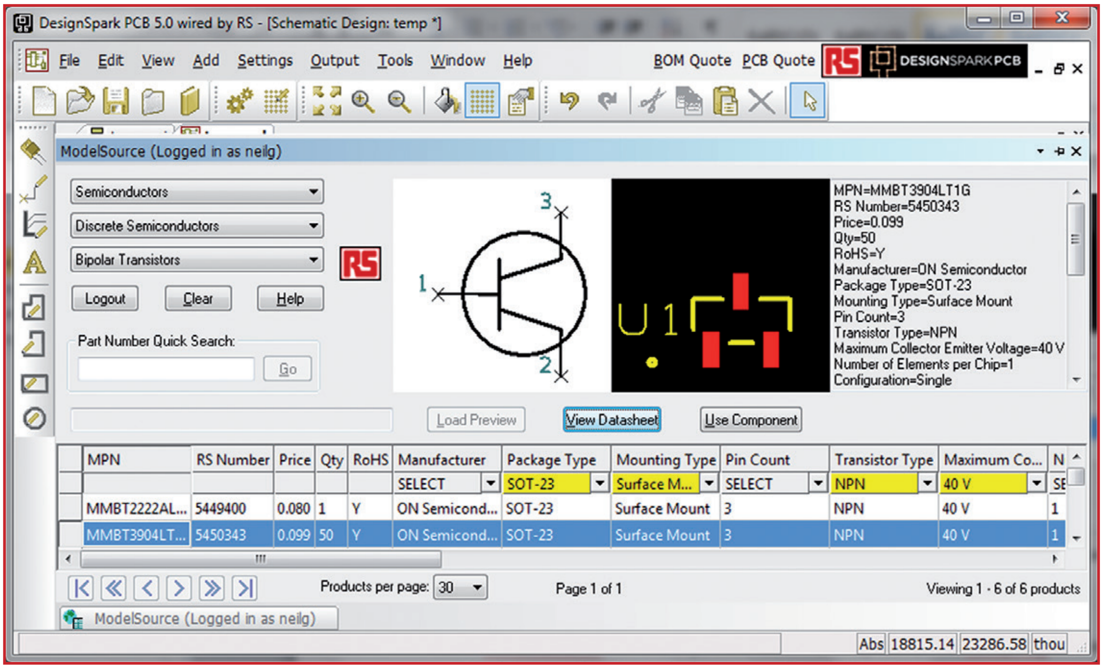
MPN	RS Number	Price	Qty	RoHS	Manufacturer	Package Type	Mounting Type	Pin Count
					SELECT	SELECT	SELECT	SELECT
BFG198,115	1003101	0.670	1	Y	NXP	SC-73	Surface Mount	4
BFS17	1003482	0.130	1	Y	NXP	TO-236AB	Surface Mount	3

Products per page: 30 Page 1 o Viewing 1 - 30 of 740 products

ModelSource (Logged in as neilg)

Abs 12678.40 5171.32 thou

Figure 3.
The ModelSource screen displaying search results.


DesignSpark PCB 5.0 wired by RS - [Schematic Design: temp *]

File Edit View Add Settings Output Tools Window Help BOM Quote PCB Quote

ModelSource (Logged in as neilg)

Semiconductors

Discrete Semiconductors

Bipolar Transistors

Logout Clear Help

Part Number Quick Search:

Go

MPN=MMBT3904LT1G
RS Number=5450343
Price=0.099
Qty=50
RoHS=Y
Manufacturer=ON Semiconductor
Package Type=SOT-23
Mounting Type=Surface Mount
Pin Count=3
Transistor Type=NPN
Maximum Collector Emitter Voltage=40 V
Number of Elements per Chip=1
Configuration=Single

Load Preview View Datasheet Use Component

MPN	RS Number	Price	Qty	RoHS	Manufacturer	Package Type	Mounting Type	Pin Count	Transistor Type	Maximum Co...	N
					SELECT	SOT-23	Surface M...	SELECT	NPN	40 V	SE
MMBT2222AL...	5449400	0.080	1	Y	ON Semicond...	SOT-23	Surface Mount	3	NPN	40 V	1
MMBT3904LT...	5450343	0.099	50	Y	ON Semicond...	SOT-23	Surface Mount	3	NPN	40 V	1

Products per page: 30 Page 1 of 1 Viewing 1 - 6 of 6 products

ModelSource (Logged in as neilg)

Abs 18815.14 23286.58 thou

Figure 4.
Detailed information of a transistor in ModelSource.

reports like bill of material (BOM) reports. For example, I prefer to store all of a component's manufacturing information in the component library. This means that I usually have multiple manufacturer part numbers, so I use the attributes as shown in **Figure 5** where I've added three additional manufacturer part numbers. Note that the other attributes are required if you want to use the DesignSpark BOM quoting function.

Conclusion

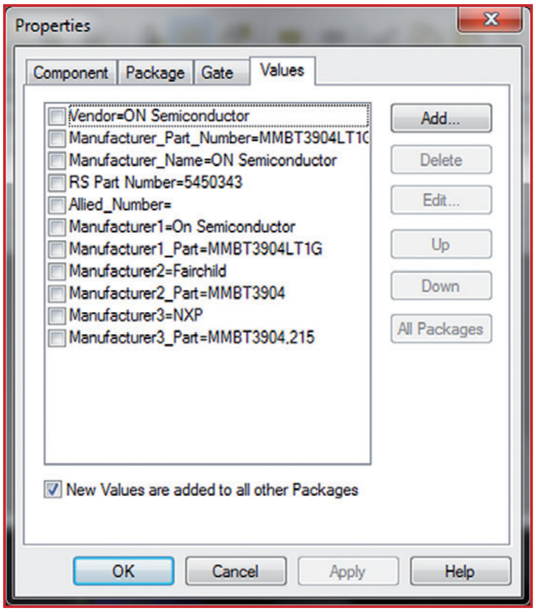
ModelSource and DesignSpark's libraries are a great resource when creating our own set of libraries and they can save a significant amount of time. At this point we can create a schematic and next time I will talk about some tricks when editing a schematic and how to generate a bill of materials.

(130207)

Internet References

[1] www.designspark.com/tutorial/components-library-structure-library-manager

[2] www.designspark.com/eng/tutorial/components-downloading-from-modelsource-building-up-libraries


Properties

Component Package Gate Values

□ Vendor=ON Semiconductor
□ Manufacturer_Part_Number=MMBT3904LT1G
□ Manufacturer_Name=ON Semiconductor
□ RS Part Number=5450343
□ Allied_Number=
□ Manufacturer1=On Semiconductor
□ Manufacturer1_Part=MMBT3904LT1G
□ Manufacturer2=Fairchild
□ Manufacturer2_Part=MMBT3904
□ Manufacturer3=NXP
□ Manufacturer3_Part=MMBT3904,215

Add... Delete Edit... Up Down All Packages

☑ New Values are added to all other Packages

OK Cancel Apply Help

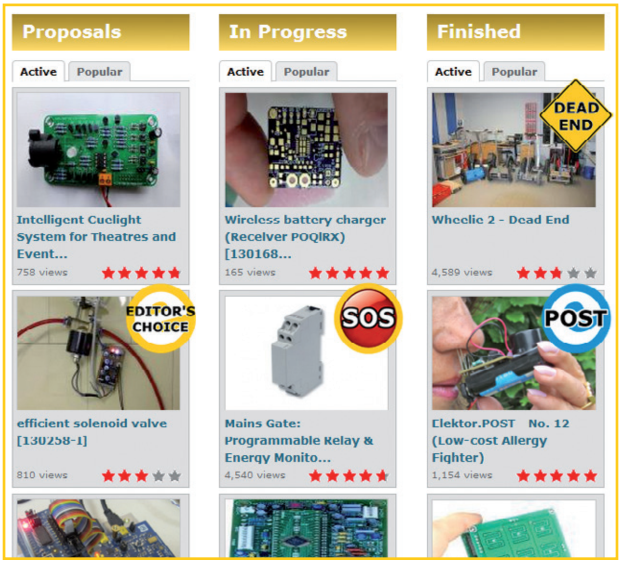
Figure 5.
Useful attributes of a component.

.Labs Tips & Tricks

elektor labs
Sharing Electronics Projects
Join a Project!
Numitron desktop clock - Using only classical components
Create a Project
In Progress
Finished

By Clemens Valens
(Elektor.Labs)

Summer is over, projects have been wrapped up, now is the time to document it all on Elektor.Labs. Here are some tips and tricks to get more out of your online publications.


Proposals
In Progress
Finished
Active
Popular
DEAD END
Intelligent Cuelight System for Theatres and Event...
758 views
Wireless battery charger (Receiver POQIRX) [130168...
165 views
Wheelie 2 - Dead End
4,589 views
EDITOR'S CHOICE
SOS
POST
efficient solenoid valve [130258-1]
810 views
Mains Gate: Programmable Relay & Energy Monito...
4,540 views
Elektor.POST No. 12 (Low-cost Allergy Fighter)
1,154 views

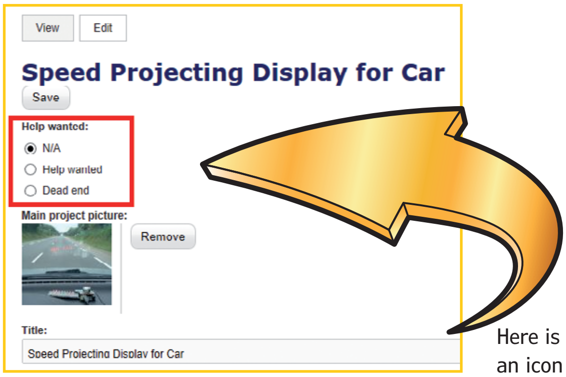
The Elektor.Labs homepage now boasts icons.

Icons

We keep enhancing the Elektor.Labs website by adding useful features. It is an evolutional process, and features are added as soon as we discover that we need them. One of the latest additions is in the form of icons that you probably have noticed on the homepage. Currently there are four icons available for drawing attention to a project. Two of these icons, the blue Post icon and the yellow Editor's Choice icon are controlled by Elektor editors. The Dead End and SOS icons are available to all posters. They can be activated to let other users know that you need help or that you are stuck.

Please note that when you activate the Dead End icon, your project may be moved to the Finished column, so use it with care.

EDITOR'S CHOICE
DEAD END
SOS
POST



Speed Projecting Display for Car

Help wanted:
● N/A
○ Help wanted
○ Dead end

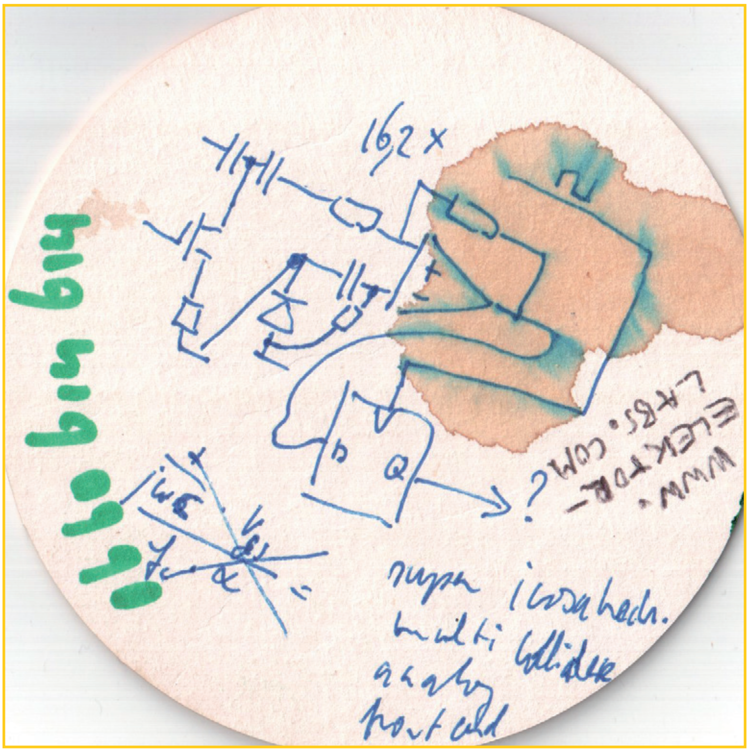
Main project picture:

Title:
Speed Projecting Display for Car

Here is where you activate an icon for your project.

Project visibility

Even though project header illustrations are optional, I highly recommend that you upload a photo or drawing anyway for the simple reason that nicely illustrated projects have a higher priority and end up higher in the lists. Every time you update your project—when you click the Save button—it is moved to the top of the list *if it has a project header photo*. The website provides a nice default picture of a dirty scribbled-on coaster (or "beer mat" if you prefer), but it is ignored by the rating mechanism.



Make sure to replace the default project header photo by a nice photo of your own project. It will improve the visibility of your project.

www.elektor-labs.com

Active users

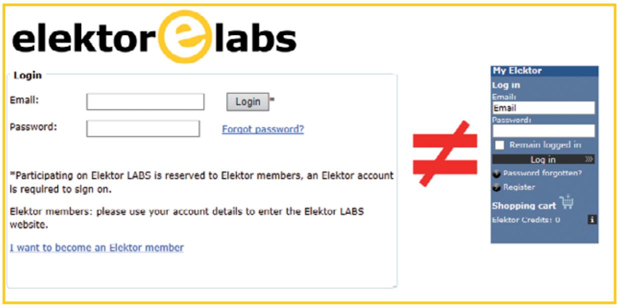
Some people are really prolific on Elektor.Labs and we like that a lot. Because these users are important to us, we have devised a scoring mechanism that allows us to identify active posters in an objective manner. Every once in a while Elektor staff have some hardware, books or other goodies to give away and the active users will be first in line to receive these freebies. The scoring system is simple: posting a project is worth four points, a contribution is two points and every comment earns you one point. The next step is of course to make the scores visible on the website—we are working on it.

P.S. Note that scoring is not completely automated, real people are involved, so posting rubbish projects, contributions or comments – spam in short – will not get you anywhere. We may even decide to block your access.

Passwords and email addresses

This is an awkward subject and, understandably, we keep receiving questions about it. The main thing for you to know as an Elektor Member is that, for historical reasons, we currently have two independent systems with two different login domains. We are working on unifying it all, but, unfortunately, that takes time. Your Elektor.com, Elektor.Post and Elektor.Store account is not the same as your Elektor.Labs and Elektor.Magazine account. You can make things simple for yourself by using the same email address and password for both accounts, but that's not mandatory.

If you want to change your email address just contact us by sending your old and new email addresses to service@elektor.com or labs@elektor.com and we will do it for you.



elektor labs

Login

Email: [ ] Login*

Password: [ ] Forgot password?

*Participating on Elektor LABS is reserved to Elektor members, an Elektor account is required to sign on.

Elektor members: please use your account details to enter the Elektor LABS website.

I want to become an Elektor member

≠

My Elcktor

Log in

Email: Email

Password: [ ]

□ Remain logged in

Log in

Password forgotten?

Register

Shopping cart

Elektor Credits: 0

Two accounts require two sets of login credentials.

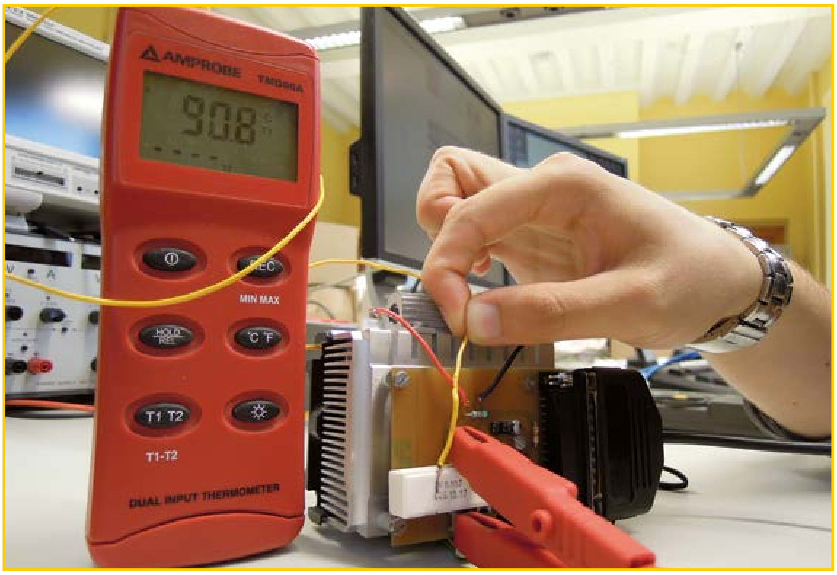
90 Degrees and Rising

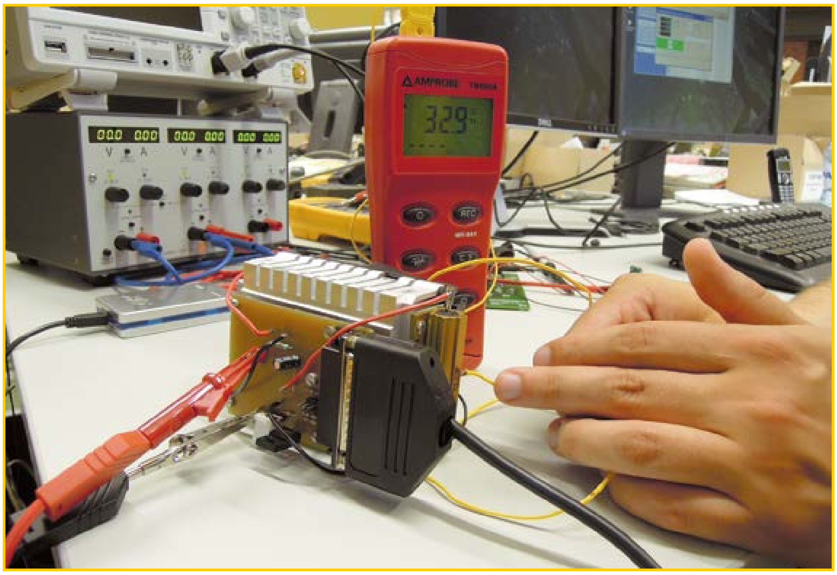
By **Thijs Beckers**
(Deputy Editor)

It wasn't just the temperature outside that rose significantly this summer @ Elektor House. While testing his prototype of the soon to be published battery testing circuit, lab worker Tim Uiterwijk was surprised to measure the temperature of a 7-watt series resistor (the big white one in the picture with the thermometer sensor held on it) at well over 90 ºC!

This wasn't expected, as the calculated dissipated energy (P_{diss}) was well below the 7 watts the power wire wound resistor is allowed to dissipate according to its specifications. The dissipated power could be calculated using the well-known formula:

$$P_{\text{diss}} = I^2 \times R.$$





While the upper limit of the current through this resistor was calculated to be 8 A, during a test it had been limited to 4.5 A. So in this case (with I = 4.5 A and R = 0.1 Ω), with just ($4.5^2 \times 0.1 \approx$) 2 watts this relatively big ceramics clad resistor got very hot *fast*. Too hot, actually. In general, any component exceeding a body temperature of 80ºC is regarded bad practice in our labs, so a solution had to be found.

As can be seen in the pictures, the circuit is mounted on a heatsink. This heatsink is a 'standard' CPU cooler (with fan) intended for an Intel P4 processor. Mostly due to their active air flow, this type of heatsink boasts a very low thermal resistance to air; of the order of 0.40 K/W. So they're excellent for dissipating lots of heat — depending on the model a PC CPU can easily generate 125 W or so — so the heatsink obviously has to be able to "digest" that. Another shunt resistor has already been mounted on the heatsink (on top in the top photo), but there's enough room for a second one.

Tim swapped the ceramics covered resistor with an aluminum housed wirewound power resistor rated at 50 W, and he mounted both power resistors on the sides of the heat sink where the airflow was highest, see the bottom photo. This solution proved adequate, with the previously hot headed power resistor now reaching only 33ºC under identical circumstances. With a maximum (software limited) current of 8 A, the resistor now heats up to about 50 degrees C, satisfying the Nothing-Hotter-Than-80 rule and even providing a little headroom. Theoretically, the 50-W resistor should be able to withstand currents of up to 22 A and temperatures of up to 250 ºC(!), but those extremes will never be met in our application.

Problem solved. Now keep an eye on our upcoming editions—an article with the full schematic and description of the circuit will be published soon.

(130055)

NT Series Transceiver now Available in 868MHz for European Market

The new 868 MHz NT Series RF transceiver module from Linx makes it simple to send and receive digital data in the 863 to 870 MHz band. The module offers the option of UART or True Transparency™ interfaces which lets the user create a wireless wire for use with nonstandard data rates, custom protocols, or encodings such as PWM and Manchester. The module features best-in-class receive sensitivity (up to –113 dBm) and low power consumption (only 19.2 mA in receive mode and 16 mA in transmit mode at 0 dBm).

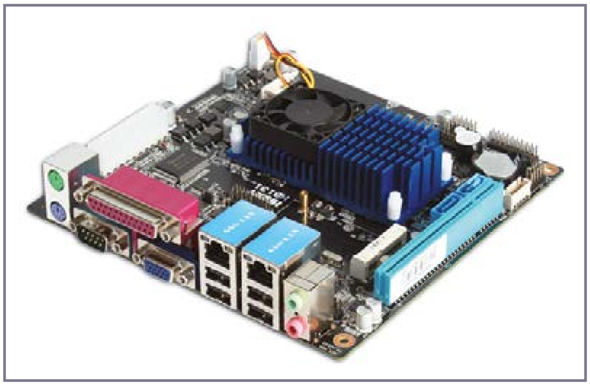
Unlike so-called transparent modules that require a UART interface and buffering scheme, the True Transparency™ feature of the NT Series reduces latency by transmitting raw data. All the user has to do is connect circuitry to the input line of the sending transceiver and the output line of the receiving transceiver. The module is completely hardware-configurable up to 8 channels. Although programming is not required, the NT Series has a UART interface that enables more advanced configuration with a microcontroller, RS-232 interface or USB interface. The 868 MHz version offers 68 channels, which lets the user implement a frequency agility scheme such as frequency hopping or listen before talk.

The module has a maximum output power of +12.5 dBm and a receiver sensitivity of –113 dBm. This gives the module a typical line of sight range of up to 2.5 miles (4 kms) at the maximum output power with typical monopole whip antennas. Regulations in the country of operation dictate the maximum legal output power, so the final system range may be less depending on the country of operation. Typical range at 0 dBm is up to 3,000 feet.

To aid rapid development, the NT Series transceiver is available as part of a master development system. The system comes with two development boards for benchmarking and prototyping, each of which is populated with a transceiver. The boards include lights that show you which areas are active as well as other enhancements that simplify the development experience.

To help you get started, the system also includes antennas, a daughter board with a USB interface, demonstration software, extra modules and connectors.

www.linxtechnologies.com (130285-II)



Mini-ITX-board for Intel Atom

The HB131 is a small form factor mini-ITX board based on low power Intel Atom Cedar Trail platform. The dual-core Atom D2550 processor, which is offered with Intel’s NM10 chipset, is primarily known for its lower power consumption & graphics enhancements compared to earlier Atom processors. The Mini-ITX board is equipped with dual Gigabit LAN ports and rich I/O. This environmentally responsible embedded mini-ITX board exceeds its predecessors in both performance speed and graphics capability all to offer greater scalability in a smaller footprint fanless PC solutions. The mini-ITX board features rich I/O interfaces: 8 USB 2.0, 6 COM ports, 2 LAN, 1 VGA up to 1920×1200, 1 LVDS up to 1440×900 (24bit). On board are 1 PCI and 1 mini-PCIe slots for richer connectivity, 1x SO-DIMM supports DDR3 800/1066MHz RAM up to 4GB, 2 SATA II. It is designed to be a high performance, reliable, secure and easy to manage board. Making it an ideal platform for point of sale, self-service terminals, queue machines and digital signage. Utilizing the small footprint of 170mmx170mm, the HB131 offers greater connectivity with dual Gigabit Ethernet with options of expansion via one Mini PCIe slot.

www.habeyusa.com (130285-IV)

0-120ml/min Liquid Flow Sensor

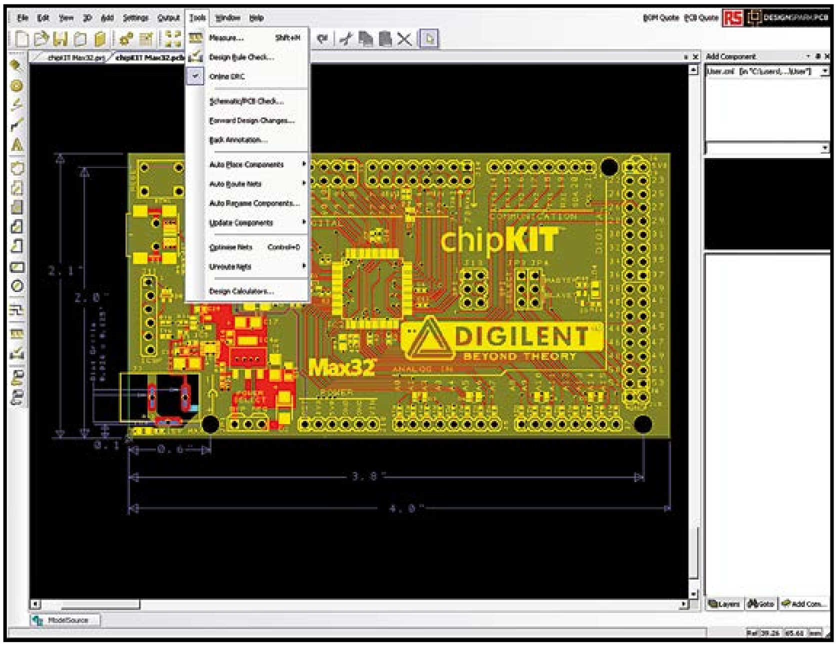
The newest liquid flow sensor from Swiss sensor manufacturer Sensirion excels thanks to its ultra-pure materials and outstanding precision. The small SLQ-QT500 sensor is designed for the needs of the semiconductor industry specifically.

The SLQ-QT500 covers flow rates from 0 – 120 ml/min. As with all Sensirion liquid flow sensors, its flow channel is absolutely straight and has no moving parts. The sensor is based on the patented CMOSens® Technology. The microthermal flow measurement is performed through the flow channel wall, which separates the chip from the measured liquids. Therefore, only the PFA tubing and the quartz flow channel are in direct contact with the liquid. This guarantees that the sensor has a superb chemical resistance. Thanks to these features as well as the RS485 digital interface, the sensor is able to achieve an exceptionally reliable measurement with a sample rate of up to 1 ms.

With this unique technology even liquids with a very high viscosity (100,000 cP and more) are not a problem. Andres Laib, Director of Sales Liquid Flow Products says: "The sensor is suitable for measuring hydrocarbon-based solvents such as photoresists, as well as water-based liquids such as TARC and H2O2. With the SLQ-QT500, liquids with virtually any viscosity as well as liquids which contain particles can be measured. This makes the sensor unique in the liquid flow sensor industry."

www.sensirion.com/slq-qt500 (130285-III)

DesignSpark PCB Version 5.0 is Here



RS Components (RS), the trading brand of Electrocomponents plc, has unveiled the latest release of DesignSpark PCB, the company's award-winning professional software for schematic capture and PCB layout. DesignSpark PCB Version 5.0 integrates two additional features within the free design tool — online Design Rule Checking and buses — which have been introduced to further reduce design times for engineers and to minimize errors during the design process.

This new release builds upon the previous version of DesignSpark PCB announced in October 2012, which provided access to the industry-leading ModelSource component library, PCB quote service and BOM quote functionality.

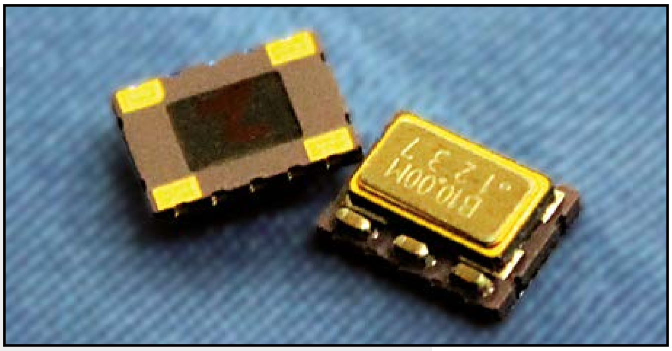
Design Rule Checking determines whether the physical layout of an integrated circuit satisfies a series of recommended parameters called design rules. DRC is an important step during the physical verification of a design, which is normally carried out in the post-design phase. Online DRC is an advancement of this process that enables the engineer to detect errors in real-time during the design stage, highlighting any issues before the design is finalized and layout completed.

The inclusion of online DRC within DesignSpark PCB Version 5.0 is a significant addition to the software's functionality, introducing considerable time saving benefits to the user as any errors can be rectified immediately with minimum knock-on effect to the rest of the design, ensuring maximum yield and reliability.

The increasing adoption of digital design has resulted in the use of on-chip buses, or bundles of related wires, to transfer data, enabling the engineer to combine multiple data signals into a bus in multiples of 8 (e.g. 8, 16, 32). Instead of drawing and/or labeling the individual wires in a schematic, a single 'bus wire' can be used to represent related wires, thus simplifying the final schematic. This also enables simulation and debugging to be carried out with minimum errors. Buses have been added directly into DesignSpark PCB Version 5.0, allowing the engineer to refine the schematic during the design phase.

www.designspark.com (130167-III)

Ultra Stable Surface Mount TCXO

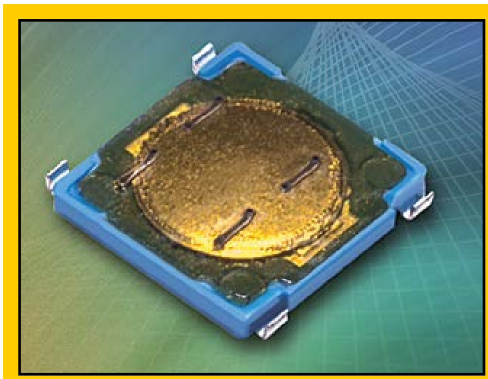


Bliley Technologies' has released industry standard 5 mm x 7 mm TCXO in an SMD hermetically sealed package. This series of TCXOs threatens OCXO Frequency vs. Temperature stability without the power consumption or cost. The frequency vs. temperature stability starts as low as ±50 ppb.

Options available for customer selection include supply voltage; +3.3 VDC or +5 VDC, HCMOS, LVCOS, or clipped sine wave output, and operating temperatures as wide as −40 °C to +85 °C. This series can be offered as a TCXO or can be configured with electronic frequency control range of ±5 ppm.

In addition to outstanding frequency vs. temperature stability this series of TCXO offers excellent phase noise at 10 MHz of −125 dBc at 100 MHz offset and a floor of −150 dBc. The T85H Series products are well suited for all network timing and general precision applications where optimum reliability and performance meet excellent price targets.

www.bliley.com (130167-VI)



Miniature Switch Saves Valuable PCB Space

C&K Components has developed the new PTS 530 Series ultra low-profile top actuated SMT switch that is ideal for consumer electronic applications. Utilizing C&K's unique symbol line identification system, design engineers can quickly and easily identify actuation force ratings, particularly valuable for designs that incorporate multiple switches. The marking system defines the model number and actuation force on the switch itself, which eliminates the need for designers to rely on confusing schematics.

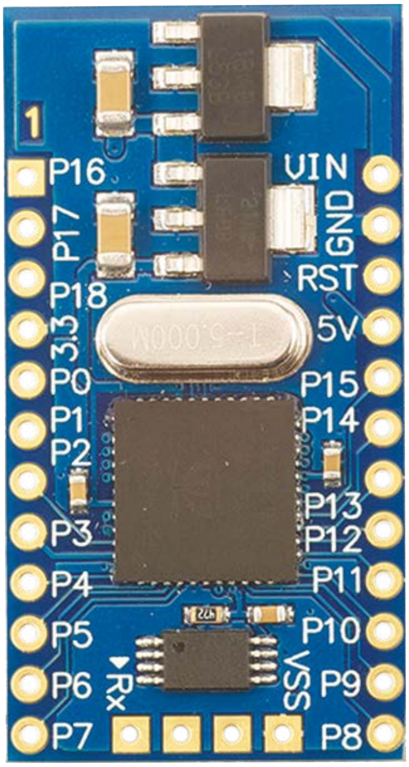
The PTS 530 Series SMT switch is only 4.5 mm x 4.5 mm with a 0.55 mm thickness (0.65 mm for high force versions), saving valuable PCB space. The small size combined with its extensive lifespan of up to 1,000,000 operations makes the PTS 530 Series satisfies multiple needs that other miniature switches cannot.

The momentary action, top actuated SMT switch features a gullwing or J-type terminal and can be soldered via infrared reflow in accordance with the IEC61760-1 standard. The PTS 530 Series switches are available in six different varieties, dependent on actuation force.

The PTS 530 Series SMT switch has a maximum voltage of 12 VDC. The operating temperature range is –40 °C to +85 °C.

www.ck-components.com (130167-IV)

Propeller Mini

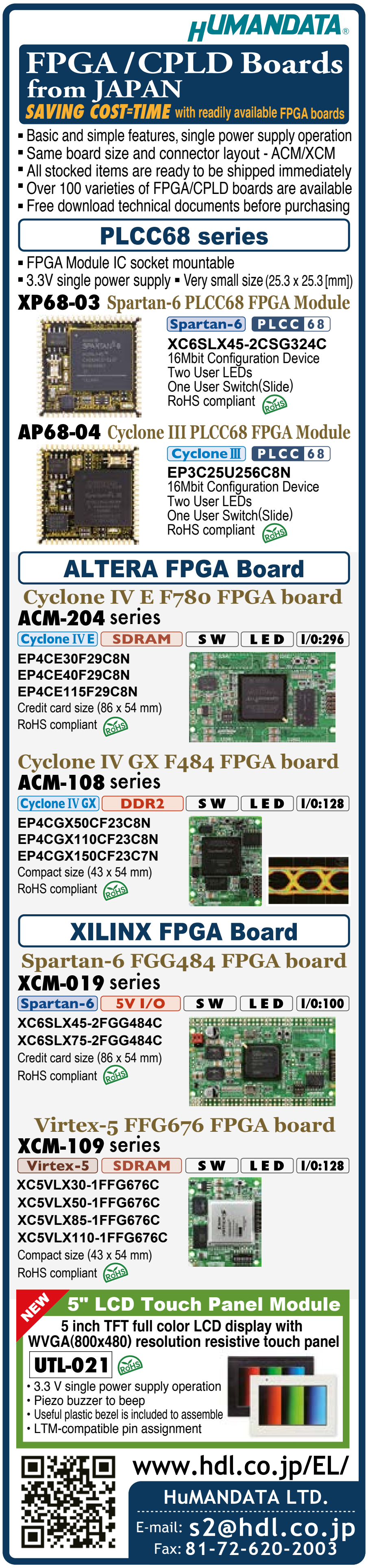


The Propeller Mini is a low-cost solution for embedding a multi-core microcontroller system in those hard-to-reach places or small sized projects where a full-sized development board is not practical. The board is small in size and component count, all while having the necessary features that you would expect from a control board.

You can solder the included header onto the Propeller Mini and be ready for breadboarding out of the bag. You also have the option of soldering your project's wire leads directly to the through holes on the board, to keep the control system for your project small. Or, solder sockets onto the Propeller Mini so it can plug into a proto board containing your sensors and other components. With any of these choices, you can have a complete prototyping system or project while maintaining a small footprint.

On the Parallax website search "Propeller Mini." (32150) Price: $24.99

www.parallax.com (130285-I)



Internet @ the Physical Layer

By
Tessel Renzenbrink
(Elektor TTF Editor)

The Internet consists of some 40,000 administratively separate networks linked together. How does this system-of-systems operate at the physical layer? Is it flakey and unreliable as some corporations hanging on to their private connections seem to think? Can it handle the persistent growth of data volumes? Is it expanding to reach the billions of poorly connected people in developing countries? Let's ask the specialists.

I discussed these questions with Henk Steenman, CTO of the Amsterdam Internet Exchange (AMS-IX) and James Cowie, co-founder and CTO of Renesys, an Internet measurement and analytics company, in an interview.

Internet exchange

In the early 1990s much of the local European Internet traffic was routed over submarine cables across the Atlantic to Virginia, USA. There MAE-East, one of the world's first Internet Exchanges (IXs), hosted physical connections to route traffic from one network to another. For many small European Internet Service Providers (ISPs) it was the only exchange point available.

In 1997 twenty competing ISPs and carriers established AMS-IX to interconnect their networks locally [1]. AMS-IX brought down the cost of data exchange, reduced latency and eased traffic congestion on the heavily overloaded American hub. Henk Steenman has been part of the Dutch nonprofit organization from the beginning. With his help AMS-IX has grown out to be one of the largest Internet exchange points in the world. Constantly in a close race for first place with DE-CIX in Frankfurt, AMS-IX currently comes second with 595 participating networks, and traffic peaking at 2.3 Tb per second.

Internet intelligence

Renesys is an American company that collects and analyzes data about both the logical and the physical structure of the Internet [2]. "The logical map tells you how the Internet believes it should be routing traffic", says James Cowie. "It basically says if you needed to reach this person and you were someplace else, what chain of organizations would help you carry it there.

The physical map is more detailed and involves figuring out which IP-addresses, which routers, are connected to each other and which ones of those are actually the most useful in moving traffic closer to its destination. We take active measurements of millions points of hundreds of places worldwide to make an accurate map of what the Internet is doing.

"We use that information for customers who need to figure out how to use the Internet effectively as a tool of business. People tend to study their own part of the Internet very carefully. But nobody worries about what is over the horizon. One of the things we provide is that big picture because more and more companies have a global interest. The Internet is not a managed system so we provide some of that missing transparency."

Physical layer

"It's interesting what we learn about the physical layer from the logical grid and these performance readings on the sensors", says Cowie. "There is a good example of a case where we saw a number of networks in Iran and Iraq disappear simultaneously. We thought this was strange so the next day we looked at the media to see what this might have been. It turned out there is a gas pipeline that goes from Iran across the frontier into Turkey and then on to European markets for the energy. Pipelines take a lot of effort to negotiate because you have to get the rights of way, secure it and bury the pipeline. So when people do that they typically also put some fiber optics next to the pipeline because it's basically zero marginal cost. That must have been the case here because that day there was a roadside bomb that had broken the pipeline."

"The heartening thing is that the Internet did not permanently get impaired by that because the Internet works around things like that all the time. There was probably another fiber route that could be used and was failed over to. So in our data you see a problem and then a recovery. The Internet is much more resilient to damage even than it's fabled."

Government regulation

Governments the world over increasingly want to regulate the Internet at the end user level. I asked the two specialists if they see the same trend at the infrastructural level.

Cowie: "The ITU, the UN agency responsible for global telecommunications regulations, took their eye off the ball during a critical window of inflation where the Internet stepped beyond something that could be lightly regulated. Which was—in my opinion—an excellent stroke of luck because it now becomes much more difficult to retroactively put things back in the bottle.

It is always possible government intervention will cause a mounting regulatory burden, you're always operating in some jurisdiction. But I think that governments realize that the Internet's fluidity makes it possible for IT services to go anywhere. The people are going to be reluctant to do things that will cause their local market to look less favorable from an investment standpoint.

I often am asked by people if their part of the Internet can be taken offline as happened during the blackouts in Egypt and Syria. I think that in Western Europe and the United States there aren't really many threats left to the Internet. The Internet has grown so wonderfully diverse in these places that in terms of being attacked or people taking the Internet offline it really can't happen anymore. It's beyond that stage."

The AMS-IX CTO isn't keen on increasing regulations either: "Currently the Dutch regulators are keeping their distance with respect to AMS-IX, but that could change. If regulations and bureaucracy were imposed on us, it would be at the cost of the flexibility and simplicity with which we run our operation. One of our most

Henk Steenman, CTO of AMS-IX

James Cowie, CTO of Renesys.

important qualities is that we're a carrier-neutral IX which means that any ISP can connect to exchange traffic. We'd like to propagate our neutrality as much as possible, and I am afraid that if the Government steps in we will lose some of that."

Data flood

AMS-IX deals with a doubling of traffic volume roughly every two years. The challenge for Henk Steenman and his colleagues is to find technical solutions to deal with that growth. "We're now implementing 100 Gb/s Ethernet equipment which became available last year", says Steenman. "Up until then we used the 10 GbE standard so we're increasing the throughput rate of our network by a factor 10. As one of the largest exchanges we're always running up against the limit of what is technically possible. We're participating in the IEEE standard body where the next standard is being developed which is going to be 400 GbE. The data rate of each new Ethernet standard has always been increased by a factor 10 but the technology simply isn't ready to make the jump to 1 Tb. Although in terms of growth we could really use that. On the other hand, growth is a two-way street, traffic can't grow faster than the available infrastructure allows, so I don't foresee any serious shortage."

James Cowie isn't worried about capacity either. "If you look at the total amount of subsea fiber that interconnects the various continents, a very small portion of that is actually lit and available for use. There is enormous amounts of reserved bandwidth. And inside continents, especially in Europe, there is just amazing amounts of available bandwidth that could be lit if the traffic grows. I don't think that is ever going to be a problem."

Digital divide

In most developed countries a well-established Internet infrastructure provides fast and cheap connections. In developing countries, however, infrastructure has lagged behind, causing a digital divide. Is the gap closing?

James Cowie: "The trend is that the countries that had the least Internet do the best as soon as the Internet does arrive. East Africa is a perfect example. The only Internet that had been available was very limited, much of it was over satellite connections which are very slow and super expensive. Until the submarine cables came on shore. Within three months we could see the whole market turning upside down. People were canceling their satellite contracts and they were moving all of their things onto this cable, and the data rates went from tens of kilobits per second to a gigabit network. Exactly like that in the space of weeks and months.

What happened there ultimately was that people were leapfrogging generations of technology. Maybe they never get a desktop computer—they just move right ahead an get a smartphone. Not having had the continuous evolution through all the stages that Western Europe went through, they get to cherry pick the very best technology at the very lowest prices. So it is actually very positive. The digital divide is still quite deep but the Internet is a great leveler."

Henk Steenman: "Now that the networks are rapidly growing in East Africa the next thing needed in terms of infrastructure are regional IXs. In Kenya, for instance, much of the local traffic destined for neighboring countries is routed via Europe for lack of a well-established regional exchange point. They are facing the same problem that motivated us to establish AMS-IX in the 1990s. So we thought: "we've done this before, why not do it again?" We're now developing an IX in Mombasa in collaboration with the Kenyan Telecommunications Service Providers Association of Kenya (TESPOK) to improve regional connectivity."

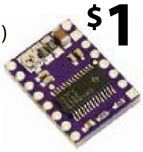
When I asked James Cowie where he thought the Internet would go from here he answered: "Yeah that's the part where I stop making predictions. The only thing we can be sure of is that it will be unexpected. It will be something completely different. We're always wrong. I am guessing the innovation is going to come from all these people in East Africa who are on the Internet and have real need. And it will be something that would have never occurred to us because we don't need things, really. We have most of our needs met. So these folks will figure this out."

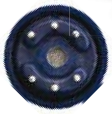
(130130)

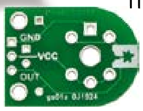
Internet References

[1] www.ams-ix.net

[2] www.renesys.com







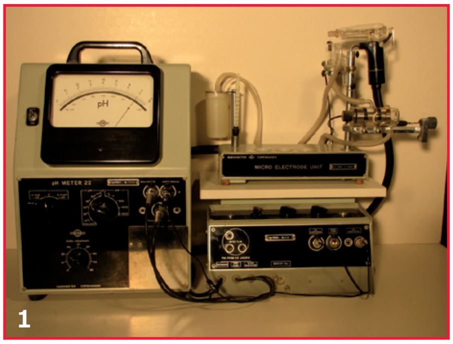
Radiometer PHM22 / PHA928a Blood pH / O_2 / CO_2 Analyzer System

Please wait for the results of your blood test

By **Seppo Lindeman** (Finland)

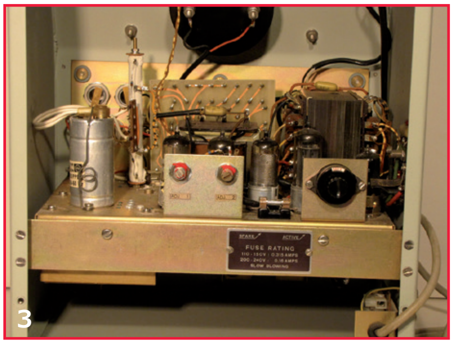
Is it a bright green instrument? No, more like greenish grey. Must be made by Radiometer, Copenhagen then, as that was their favorite (and only?) color for over 50 years of producing high quality electronic test gear back in the 20th century. The first time I came across Radiometer equipment was in 1961, at my first job at the Helvar TV and radio works in Helsinki. There were many green Radiometer vacuum tube voltmeters and signal generators around at the place—easily noticed of course as they have a tendency to flock in small groups near AC outlets. It wasn't until later that I noticed that all Radiometer gear was "that" green, be it medical-analytical or electronic test instruments.

It was at a time when:



1

2



3

- transistors entered competition commercially with vacuum tubes, and nobody had ever heard of ESD (electrostatic discharge);
- Germanium transistors often broke down "mysteriously";
- vacuum tubes lovers kept joking that the single advantage of transistors over tubes might be formulated as: "taking less space in the bin when defective".

That time, roughly.

Muppet Beaker's delight

The hematic pH/O_2/CO_2 analyzer discussed here is not a single instrument—it consists of many individual assemblies. The first models appear to have been made in the 1950s. The equipment discussed here however is mainly 1960s. These instruments appear in various guises, like bench models or 'mobile' for moving around on carts (trolleys). The instrument constellation shown in **Figure 1** consist of a blood pH meter type PHM22t with glass and calomel electrodes, a Micro Electrode Unit, a type PHA928a Oxygen Monitor with a pO_2 electrode and a D616 thermostat controlled cell, a tonometer and two humidifiers made from glass, a type VTS13 thermostat, a meter scale expander and two gas bottles.

pH Meter 22 (v. 1966)

The heart of the analyzer system is the PHM22t (model year 1966) pictured separately in **Figure 2**. There are five vacuum tubes in the meter (**Figure 3**). The circuit diagram pictured in part in **Figure 4** allows seven sections to be identified roughly: input amplifier, chopper, AC amplifier, demodulator, meter, power supply and compensation voltage divider. Three .jpg files constituting the complete schematic can be downloaded from [2].

The output voltage rate of the glass electrode is 61.54 mV pH^{-1} at 37 °C (approx. 100 °F). The meter has a resolution of .001 pH, equating to 61 microvolts. A pure DC amplifier built from vacuum tubes is not easy to keep stable at such low voltages. Also, the glass electrode output current is so low as to call for an amplifier input impedance in the range from 50 to 500 megohms. If you look at **Figure 5**, there's the principle of operation of an older Radiometer pH meter type PHM12. It incorporates vacuum tubes configured as a high gain amplifier. There is also a 'Weston Pair' type reference element ('Normal'). Before starting any measurement, users should press the Test button and register the analog measurement needle with a mark on the scale. This is to ascertain that any drift observed is not due to the amplifier.

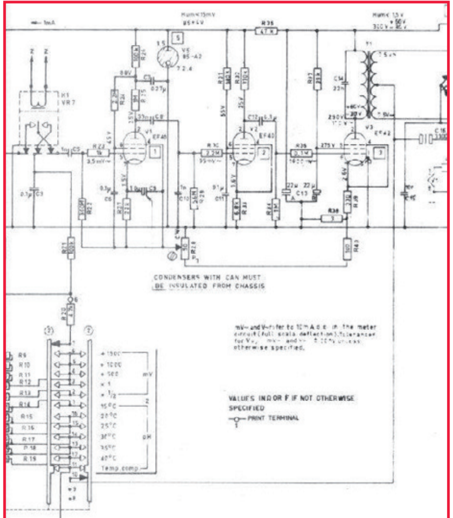
Inside PHM22t

The input amplifier of the pH meter comprises a chopper circuit. The chopper converts the input DC signal into an AC signal, which can be amplified more easily and drift-free. The chopper is similar to the Brown Converter pictured in *Retronics*, April 2013 [1]. The original chopper of this meter was mechanical, but later it got changed to a photo chopper (VR7). Sadly, humming away at 50 Hz a mechanical chopper can be relied on to fail after a few years due to contact problems.

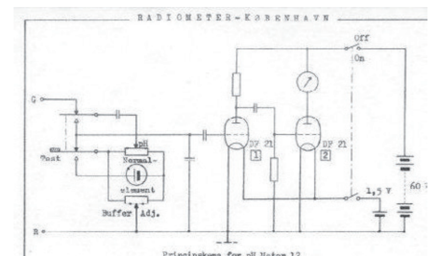
The AC amplifier is a 3-stage vacuum tube amplifier with feedback from the cathode of the output stage to the cathode of the input stage. The output of the AC amplifier is transformer-coupled to the demodulator circuit. The demodulator converts the AC signal into a DC signal virtually proportional to the direct voltage at the input of the pH meter (**Figure 6**).

One interesting thing to note concerns V1, the first tube in the PHM22t (not shown in Figure 4). Its filament voltage is lower than 6.3 V due to a 3-Ω series resistor, R48—and V1 is a rectifier. The intention is to raise the tube's input impedance by keeping the cathode of the tube 'colder' than normal. On the down side, reducing cathode emission is likely to shorten the tube's lifetime owing to cathode poisoning. When I opened the rear cover of the meter for the first time, I noticed that one tube glowed dimmer than the others, and mistakenly thought it was defective.

The DC supply voltage to the amplifier comes from a full-wave vacuum tube rectifier. The input stage operates off an 85-V voltage stabilizer tube.

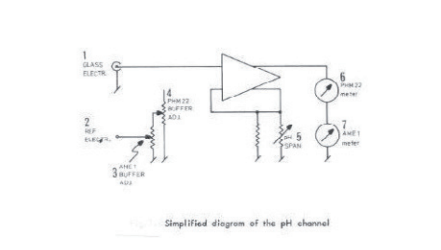


4



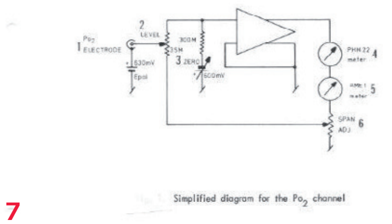
Principskema for pH Meter 12

5

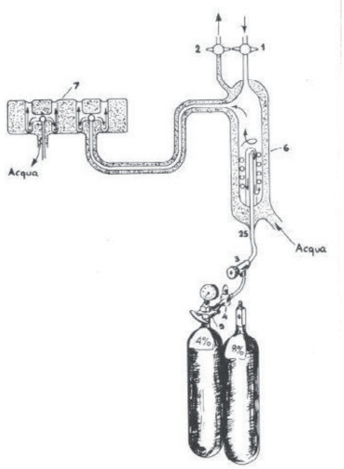


Simplified diagram of the pH channel

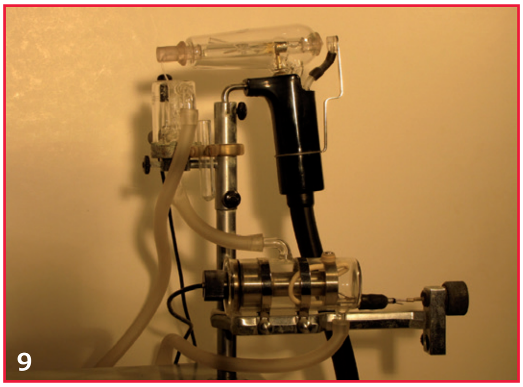
6



7



8



9

PHA928a oxygen monitor

The oxygen monitor is an all-passive instrument containing only feedback circuits that get connected into the amplifier and meter circuitry of the PHM22 pH meter and the external meter.

The pO_2 electrode is operated at a polarization voltage of 630 mV generated off a 1.35-V mercury battery. The circuitry around it is all low impedance, low current. **Figure 7** shows the simplified diagram of the pO_2 channel. Figures 6 and 7 show how a single pH meter may be used for two different measurements.

Carbon dioxide levels (pCO_2)

The instrument constellation does not feature a pCO_2 electrode. The CO_2 levels in blood are measured indirectly using three blood samples. The first measurement is direct pH. The other samples are placed in separate chambers of the tonometer, which also accept a feed of 4% CO_2 gas and 8% CO_2 gas (**Figure 8**). After about 4 minutes of equilibrating, the pH of both samples can be measured. The pCO_2 value can be obtained from Astrup`s (*et al.*) method using Siggaard-Anderson's nomogram correlating arterial pH and pCO_2 [3]. There's a lot more to glean from the same nomogram in terms of human physiology, but we are not medical students here and the matter is well beyond the scope of this article.

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

Micro Electrode Unit

The main cause of inaccurate measurement results is drift. The electrodes are sensitive to temperature and static buildup due to the very high impedance. The blood samples leave stains on the glass and also degrade (to a degree) the polypropylene membranes of the electrodes. The electronic parts also cause drift. All electrodes are protected with glass jackets and surrounded by thermostat-controlled water. The water is lightly saline, creating a liquid shield around the electrode—like a metal cover protects a sensitive amplifier (**Figure 9**).

Because the electrode sensitivity is subject to change, calibration is necessary using reference solutions for pH, and zero and saturated liquids for pO_2.

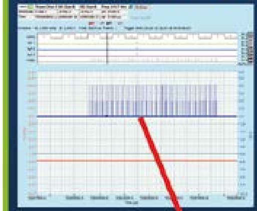
Fifty years on

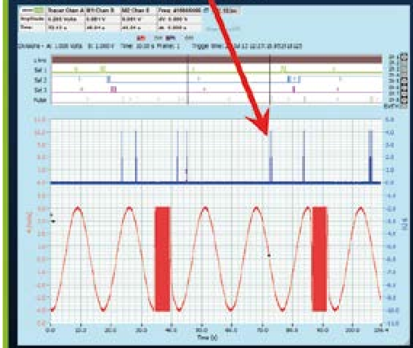
Today, hematology lab workers have an array of control ampoules at their disposal, and have the comfort of massive quality check systems. Nothing of the sort was available in the 1950s and 60s when the Radiometer PHM22/PHA928a kit was used in hospitals. One hospital in Helsinki reportedly ran their very own quality check and calibration "Retronics Style": if there was so much of a hunch of incorrect measuring results from the PHM22/PHA928a equipment, laboratory staff were quick to turn their brand new Rolodex, telephone a certain lady on the hospital cleaning staff, and wait for her shift to start. She was strong, stout and permanently healthy—in a word: *Known Good*. A nurse would simply take samples of the good lady's blood and have them analyzed. If the lab staff thought the pH, pO_2 and pCO_2 results made sense then the entirety of Radiometer blood analyzer gear was declared *beyond suspicion* and good to go for most patients—except possibly for one Jackson, Michael Joseph (*b.* 1958).

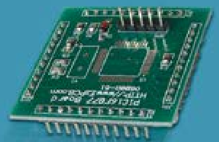
(130132)

Internet References

1. *The Curse of the Collector*, Elektor April 2013. www.elektor.com/120753
2. www.elektor.com/130132
3. www.anaesthesia.med.usyd.edu.au/resources/lectures/acidbase_mjb/description.html







Hexadoku Puzzle with an electronic touch

It's irrelevant if it takes you 20 minutes or 3 weeks to solve our popular Hexadoku puzzle—what matters is the achievement of actually solving this brain teaser. If you believe you are successful at finding the solution in the gray boxes, submit it to us 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 (£80.00)** and three Elektor book vouchers worth **$60.00 (£40.00)** each, which should encourage all Elektor readers to participate.

Participate!

Before October 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 June 2013 Hexadoku is: **F9407**. The Eurocircuits $140.00 (£80.00) voucher has been awarded to Ciril Zalokar (Slovenia). The Elektor $60.00(£40.00) book vouchers have been awarded to Arne Jansson (Sweden), Gerard Yvraut (France), and Philippe Monnard (Switzerland).

Congratulations everyone!

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

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

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

Join the Elektor Community

Take out a GOLD Membership now!

elektor
magazine
SUPRA PLATFORM PROGRAMMING
Pull your code from the Embedded Firmware Library
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
elektor
GOLD
membership

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

Projects
Wi-Fi Controller Board
Control RGB LED strips, motors, relays & stuff, but no wires

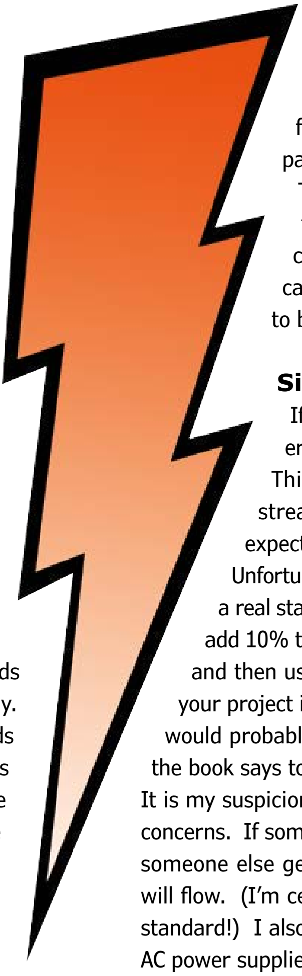
ALSO AVAILABLE:

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

ConFused

By Gerard Fonte (USA)



The lowly fuse (or circuit breaker) is an important part in the safety of any electrical device. Primary fuses (for the AC mains) protect against fire, shock and destroyed equipment. Secondary fuses (in the DC power supply) are generally used to safeguard the device. Unfortunately, there seems to be considerable confusion about these basic parts.

Ratings

There are two numbers associated with a fuse: a current and a voltage. Nearly everyone knows the current controls the fuse. More current than specified and the internal link melts and the fuse blows. The fuse is a current device, so it doesn't matter what the voltage is. It could be one volt or 100 volts (as long as it is below the rated voltage).

However, it can take a couple of hours for slow-blow fuse types to open at 135% of rated capacity. At 200%, it's typically a couple of minutes. The fast-acting fuses may still take up to an hour at 135% of rated current but only a few seconds at 200%. Most people are quite surprised at this much delay. At ten times the current, the blow times are about 0.1 seconds (slow) and 0.01 seconds (fast). For modern electronics, this is still plenty of time to do significant internal damage. (The speed specification is provided in the data sheet as I^2t. The lower the value, the faster the fuse.)

The voltage specification is the confusing one. It means that the blown fuse will present an open circuit at the rated voltage. That is: the gap in the link will be physically large enough so that there will be no arcing between the internal remains. Obviously this is a very important consideration. If there is arcing between the fuse elements, then the fuse is not really protecting anything. Current is still flowing into the device.

Application

For AC mains, it is absolutely essential that the fuse be placed in the hot lead rather than the neutral lead. This makes sense when you stop and think (always a good idea). A blown fuse in the hot lead stops the current before it gets any farther into the device. If the fuse is in the neutral lead, there is still voltage present in the device, even if the fuse is blown. So if a different ground path is available—like a test lead or your finger—current will flow. What's more, if the fuse is in the neutral lead and there is a short to ground (rather than neutral), the fuse won't blow at all. In this case, the fuse is not in the current-carrying circuit and is useless.

A fuse in the secondary of a transformer can be in either lead. If there are multiple transformer windings, or if the transformer has a center-tapped winding, several fuses may be required.

The standard fuse is a mechanical device and is susceptible to damage from shock and vibration. Mounting your fuse on a thin panel next to a big motor is just asking for trouble.

The fuse is also a thermal device. If you are using surface-mount fuses, or soldering fuses directly into your circuit (instead of using fuseholders) you must be very careful with the application of heat. It is quite possible to blow the fuse with improper assembly.

Sizing (mains power)

If you are wiring a house the fuses (actually circuit-breakers) are sized according to the appropriate building code. This code is often based on the thickness of the downstream wire rather than the actual current the circuit is expected to handle.

Unfortunately, for electronic projects there does not appear to be a real standard. My old ARRL Radio Amateur's Handbook says to add 10% to 20% more than the expected maximum current draw and then use the fuse with the next largest current rating. So, if your project is expected to draw one amp (of AC mains current) you would probably choose a 1.25 amp fuse. However, 200 pages later, the book says to fuse at 150% to 200% of maximum.

It is my suspicion that no one wants to be too specific because of legal concerns. If someone publishes a 'standard' (or even a suggestion) and someone else gets hurt using that standard or suggestion—lawsuits will flow. (I'm certainly not going to make any suggestion or define a standard!) I also suspect that many products that include stand-alone AC power supplies ('wall-wart') do so to eliminate AC safety testing and certification. It may be a bit more expensive to do so but there is much less legal exposure.

Of course, if you are making a project for your own use, you only have to consider your own risk. You can do whatever you want. But as soon as you start selling your device, you must consider the risk to the users.

PTC

PTC (Positive Temperature Coefficient) devices are very different from fuses. They are also thermal devices that operate about the speed of slow-blow fuses. However, one huge difference is that they do not actually interrupt the power. Rather, they change to a higher resistance and limit the current. Thus, some current still flows. The second big difference is that they 'self-reset' when they cool down. Usually, this requires the removal of AC mains power. However, if the short-circuit goes away when the current is reduced (fairly common for some solid-state designs), the power can come back on unexpectedly. This can be very shocking. Choose wisely when you select your protection.

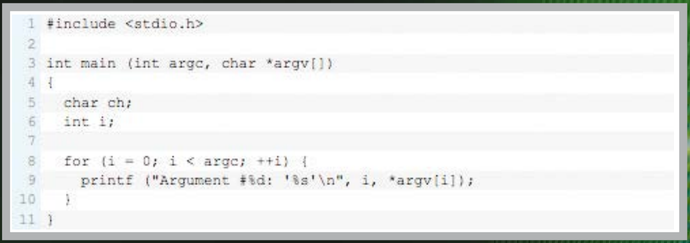
(130231)

CIRCUIT CELLAR

[The CC Weekly <Code/> Challenge is underway!]

Each week, you'll find a new snippet of **source code** that contains one error.

>>>



```
#include <stdio.h>

int main (int argc, char *argv[])
{
  char ch;
  int i;

  for (i = 0; i < argc; ++i) {
    printf ("Argument #%d: '%s'\n", i, *argv[i]);
  }
}
```

sample code only

If you can find the error, you could be a winner!

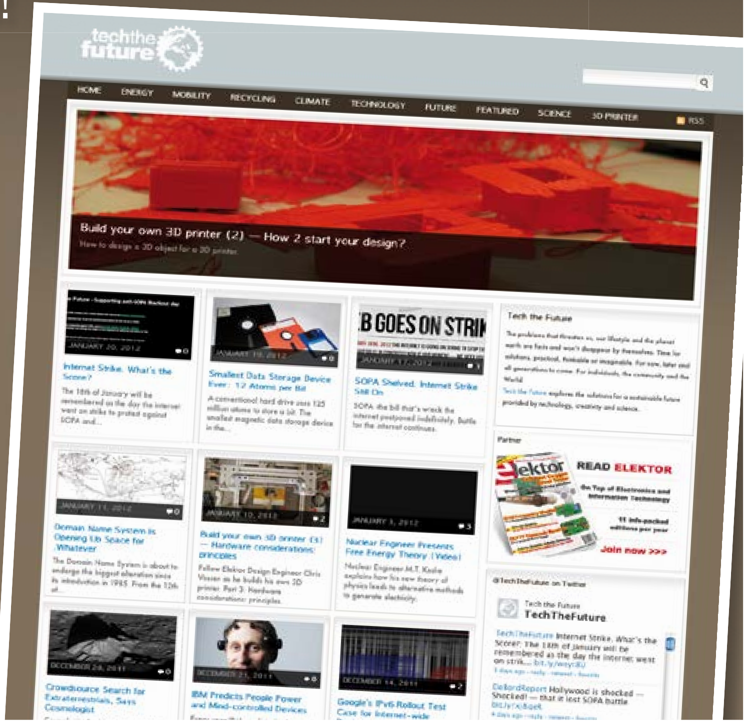
Follow *Circuit Cellar* on Facebook and Twitter for information about each week's challenge, prizes, and winners announcements.

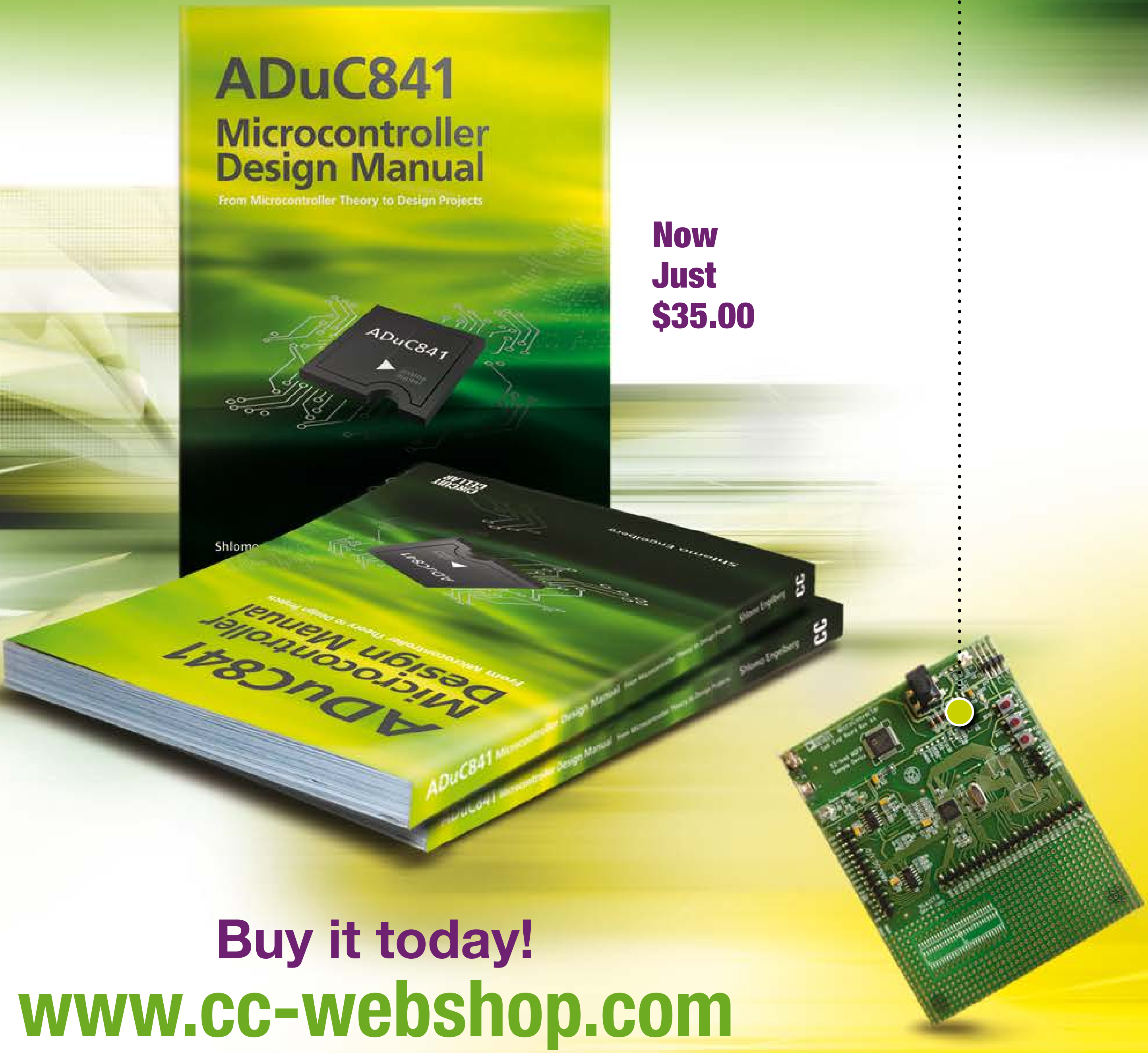
@circuitcellar
@editor_cc

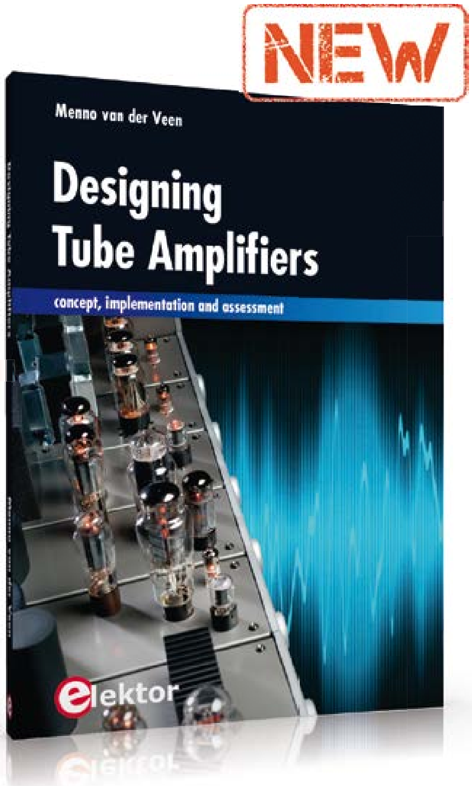
circuitcellar

For complete details, visit
circuitcellar.com/cc_weekly_code_challenge

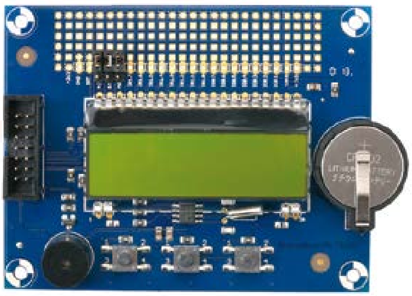
SPONSORED BY IAR SYSTEMS


techthe future
HOME
ENERGY
MOBILITY
RECYCLING
CLIMATE
TECHNOLOGY
FUTURE
FEATURED
SCIENCE
3D PRINTER
RSS

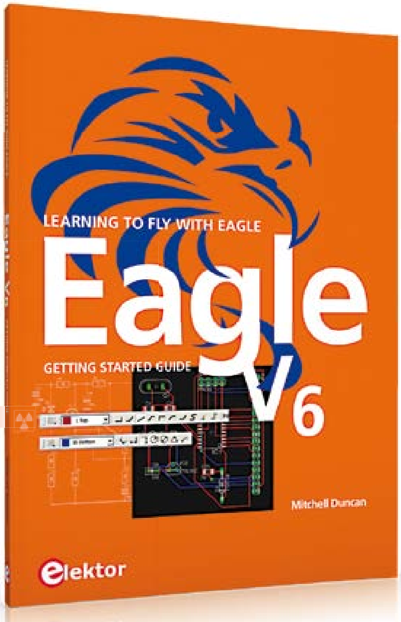

ADuC841
Microcontroller
Design Manual
From Microcontroller Theory to Design Projects
ADuC841
Shlomo Engelberg


NEW
Menno van der Veen
Designing Tube Amplifiers
concept, implementation and assessment
elektor

1



2



3

4

Limited Time Offer for GREEN and GOLD Members!
17% DISCOUNT + FREE SHIPPING
www.elektor.com/tubeamplifiers

Concept, implementation and assessment

1 Designing Tube Amplifiers

This book looks at tube amplifiers from more than just a theoretical perspective. It focuses primarily on the design phase, where decisions must be taken with regard to the purpose and requirements of the amplifier, and it addresses the following questions: How do these aspects relate to subjective and objective criteria? Which circuits sound the best, and why? If you want to develop and market an amplifier, what problems should you expect? What are the significance and meaning of measurements? Are they still meaningful, or have they lost their relevance? Thanks to the enormous processing power of computers, we can now measure more details than ever before. How can these new methods be applied to tube amplifiers? Menno van der Veen will give you all the answers!

188 pages • ISBN 978-1-907920-22-6
£29.50 • € 34.50 • US $47.60

Display, buttons, real time clock and more

2 Elektor Linux Board Extension

This extension board was developed to further propel our Embedded Linux series of articles and the matching GNUblin 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
£31.10 • € 34.95 • US $50.20

Learning to fly with Eagle

3 EAGLE V6 Getting Started Guide

This book is intended for anyone who wants an introduction to the capabilities of the CadSoft's EAGLE PCB design software package. This book will quickly allow you to obtain an overview of the main modules of EAGLE: the schematic editor; layout editor and autorouter in one single interface. You will apply your knowledge of EAGLE commands to a small project, learn more about some of the advanced concepts of EAGLE and its capabilities and understand how EAGLE relates to the stages of PCB manufacture. After reading this book while practicing some of the examples, and completing the projects, you should feel confident about taking on more challenging endeavors.

208 pages • ISBN 978-1-907920-20-2
£29.50 • € 34.50 • US $47.60

140 Minutes video presentation and more

4 DVD Feedback in Audio Amplifiers

In this Masterclass we address several aspects of feedback in audio amplifiers. The focus of this Masterclass, although not entirely math-free, is on providing insight and understanding of the issues involved. Presenter Jan Didden provides a clear overview of the benefits that can be obtained by feedback and its sibling, error correction; but also of its limitations and disadvantages. Recommended to audio designers and serious audio hobbyists!

ISBN 978-907920-16-5
£24.90 • € 29.95 • US $40.20

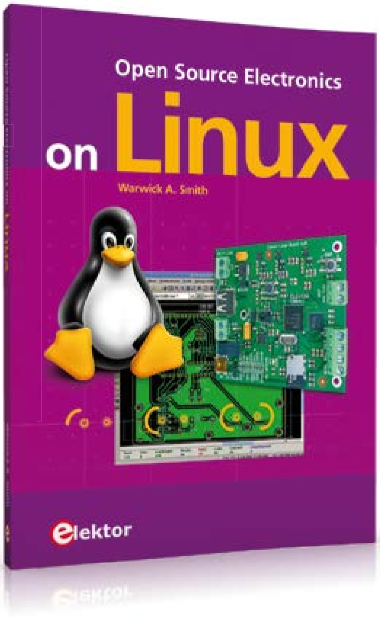
Ultrasensitive wideband E-smog detector

5 TAPIR Sniffs it Out!

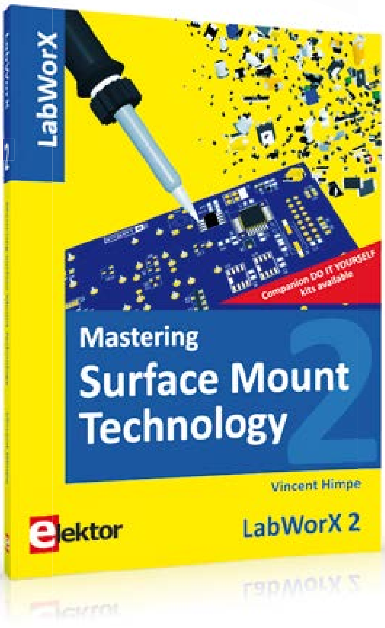
Attention boy scouts, professionals and grandfathers! This ultrasensitive wideband E-smog detector offers you two extra senses to track down noise that's normally inaudible. TAPIR — short for Totally Archaic but Practical Interceptor of Radiation — also makes a nice project to build: the kit comprises everything you need. Even the enclosure, ingeniously consisting of the PCB proper! Using the TAPIR is dead easy. Connect the headphones and an antenna and switch it on. Move it around any electrical device and you'll hear different

5

6



7



8

noises with each device, depending on the type and frequency of the emitted field.

Kit of parts, incl. PCB

Art.# 120354-71

£13.30 • € 14.95 • US $21.50

OS Hard- and Software for Electronics Applications

6 Open Source Electronics on Linux

If you have ever wanted to take advantage of the expanding field of open source software for electronics and everyday applications, this book is for you. Using the Linux OS, Warwick A. Smith guides you through the world of open source hardware and software, teaching readers to use EDA tools and software that is readily available online, free to download. The hardware projects inside can be built using easily obtainable parts, in the comfort of your own home, on single sided PCBs, or professionally manufactured with output files generated by you. Open Source Electronics on Linux is about changing today's electronics enthusiast into empowered, savvy, discerning engineers capable of building and modifying their creations, be it solely on Linux or in tandem with your current operating system.

272 pages • ISBN 978-1-907920-19-6

£29.50 • € 34.50 • US $47.60

LabWorX 2

7 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

£29.50 • € 34.50 • US $47.60

Ideal reading for students and engineers

8 Practical Digital Signal Processing using Microcontrollers

This book on Digital Signal Processing (DSP) reflects the growing importance of discrete time signals and their use in everyday microcontroller based systems. The author presents the basic theory of DSP with minimum mathematical treatment and teaches the reader how to design and implement DSP algorithms using popular PIC microcontrollers. The author's approach is practical and the book is backed with many worked examples and tested and working microcontroller programs. The book should be ideal reading for students at all levels and for the practicing engineers who may want to design and develop intelligent DSP based systems.

428 pages • ISBN 978-1-907920-21-9

£44.90 • € 49.90 • US $72.50

Further Information and Ordering: www.elektor.com/store
or contact customer service for your region

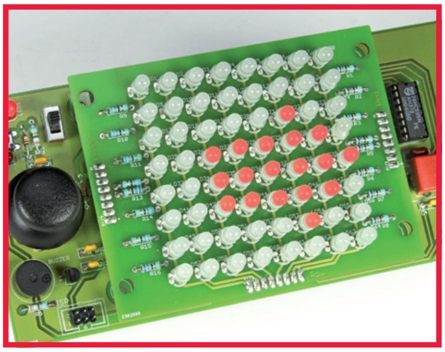
UK / ROW

Elektor International Media
78 York Street
London - W1H 1DP United Kingdom
Phone: +44 20 7692 8344
E-mail: service@elektor.com

USA / CANADA

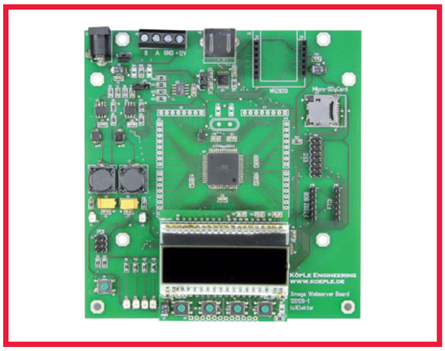
Elektor US
111 Founders Plaza, Suite 300
East Hartford, CT 06108 USA
Phone: 860.289.0800
E-mail: service@elektor.com

NEXT MONTH IN ELEKTOR MAGAZINE



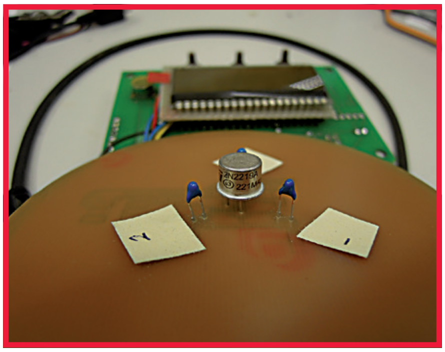
8×8 Two-color LED Matrix with an ATmega328P

This project aims to explain programming for Atmel microcontrollers in an easy way. The use of an 8×8 array of 2-color LEDs is mainly for the fun of it. Hopefully it also helps you understand the way the "bitshift" operation works for the purpose of LED driving. Plus we have a go at game programming! Some elementary knowledge of C/C++ programming is helpful here.



Xmega Webserver

Due to lack of space in the current edition, we've had to reschedule publication of this AVR powerhouse to the October 2013 magazine. In terms of I/O we have 4 LEDs, 4 pushbuttons and a (separately installed) display. For interfacing, you can choose between RS485 and various UART/TTL connectors, allowing our BOB USB-TTL converter to be connected, for example. The Embedded Extension Connector makes the board pretty versatile. The board also has a Micro SD connector, and there is room for a TCP/IP module that allows web server and other network applications to be realized.



Wind Speed and Direction Meter

The most widespread way to measure wind speed and direction is with a wind vane and an anemometer. In this project we take a different approach—without moving parts and using a circuit based on a thermal mass flow meter. A heating element heats the air, which depending on the wind direction and speed gets directed across sensors fitted around the element. Measured values are interpreted in software, which is no easy task.

Article titles and magazine contents subject to change; please check the Magazine tab at www.elektor.com.

Elektor October 2013 edition is published to Members on September 10, 2013.

See what's brewing @ Elektor Labs 24/7

Check out

www.elektor-labs.com

and join, share, participate!

elektor labs

Sharing Electronics Projects

Search

Home | News | Proposals ➡ In Progress ➡ Finished | Log in

get started with the LPC800

About Elektor.LABS

Proposals | In Progress | Finished

Active | Popular

3D Printer Head and Mat Temperature controller - u... 38 views

BLiVIT - A feature-rich LI+ battery charger / mana... 1,840 views

Android style capacitive sensing pattern lock [120... 3,224 views

editor's choice

elektor post

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 **not a member?** You can!

Click here to send a description of your project including a circuit diagram and a photograph for evaluation and maybe you will be granted free access!

ORDERING INFORMATION

To order, contact customer service for your region:

USA / CANADA
Elektor US
111 Founders Plaza, Suite 300
East Hartford, CT 06108
USA
Phone: 860.289.0800
E-mail: service@elektor.com

Customer service hours:
Monday-Friday 8:30 AM-4:30 PM EST.

UK / ROW
Elektor International Media
78 York Street
London W1H 1DP
United Kingdom
Phone: (+44) (0)20 7692 8344
E-mail: service@elektor.com

Customer service hours:
Monday-Thursday 9:00 AM-5:00 PM CET.

***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 for your region.*

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.

TERMS OF BUSINESS

Shipping Note:
All orders will be shipped from Europe. Please allow 2–4 weeks for delivery.

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, magazines, online publications and presentations. Elektor accepts no responsibility or liability for failing to identify such patent or other protection.

Copyright
All drawings, photographs, articles, printed circuit boards, programmed integrated circuits, discs, 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 educational 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

Membership renewals and change of address should be sent to the Elektor Membership Department for your region:

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

UK / ROW
Elektor International Media
78 York Street
London W1H 1DP
United Kingdom
Phone: (+44) (0)20 7692 8344
E-mail: service@elektor.com

Do you want to become an Elektor GREEN or GOLD Member or does your current Membership expire soon?

Go to www.elektor.com/member.

SoC 11th
INTERNATIONAL SYSTEM-ON-CHIP CONFERENCE

11th International System-on-Chip (SoC)

Conference, Exhibit & Workshops

October 23 & 24, 2013 – University of California, Irvine - Calit2

www.SoCconference.com

EARLY BIRD REGISTRATION IS NOW OPEN!

Don't Miss Out!

Platinum Sponsors

- FinFET Technology & Applications
- Analog & Mixed-Signal Designs
- Sub 14nm Designs & Beyond
- 3-D ICs Designs
- IC Security & Challenges
- Terahertz Power in CMOS
- SoC Design & Verification
- Biologic Battery
- SiGe Technology Trends
- Low-Power Design Techniques
- Charge-Trap Flash Memory
- Table-Top Exhibit (Free Passes)

Microsemi
ALDEC THE DESIGN VERIFICATION COMPANY
UCIRVINE | EXTENSION
Calit2
SSCS

- FinFET & Mixed-Signal SoC
- SOI vs. CMOS & Beyond CMOS
- Electromagnetic & CKT Co-Design
- FPGAs –Trends & Designs
- EDA for High-Speed Interconnects
- Smarter Mobile Devices
- Fully Depleted SOI Devices
- Network-on-Chips (NoCs)
- Informative Panels
- IEEE Student Design Contest
- Networking Opportunities
- And Much More. . .

Keynote Speakers

Xilinx, Inc.
Dr. Kees Vissers
Distinguished Engineer.

Nokia
Dr. John Paul Shen,
Nokia Fellow, Founding Head
of Nokia Research North
America Lab.

IBM
Dr. David Harame Chief
Technical Executive.

University of Pennsylvania
Dr. Nader Engheta
H. Nedwill Ramsey Professor
of Electrical and Systems
Engineering.

Microsemi
Jim Aralis,
Chief Technology Officer
(CTO),
and Vice President of R&D.

**Promote Your Company, Technology, Products & Services at
The Most Informative, Targeted & Affordably Priced
Chip Design Conference, Exhibit & Workshops of the Year!**

Selected Participating Companies & Universities

UC San Diego
Penn University of Pennsylvania
ALDEC THE DESIGN VERIFICATION COMPANY
Microsemi
intel
SAVANT COMPANY INC.
ST
Soitec
X-FAB
SYNOPSYS
IBM
XILINX
NOKIA
USC UNIVERSITY OF SOUTHERN CALIFORNIA
BROADCOM
NEC
GLOBALFOUNDRIES
BREKER
FUJITSU
AMD
elektor
UCDAVIS UNIVERSITY OF CALIFORNIA
Princeton University
UCLA
UCI
SPANSION
MICROCHIP
OPEN ALLIANCE
CIRCUIT CELLAR
THE UNIVERSITY OF TEXAS AT AUSTIN
sp3 diamond technologies
CRC Press Taylor & Francis Group

**For More Information or Questions, Please Contact the SoC Conference Organizing Committee at:
SoC@SoCconference.com or (949) 851-1714
www.SavantCompany.com & www.SoCconference.com**