www.elektor-magazine.com

.magazine

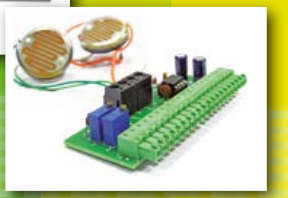
July & August 2014

elektor

Project Generator Edition

100+ Pages of Electronics & Embedded



Dice Clock
Animal Friendly Mousetrap
Smart Power Bar • Analog Discovery from Digilent
IO-Warrior Expansion Board • Opto-isolated PTT/CW Interface
Farmer-Goat-Wolf-Cabbage Game • Chip-8 Video Games Emulator
Soldering On With the Weller • Brushed-to-Brushless Motor Conversion
Power Outlet tester with LEDs • Acupunctural NiCd Battery Conditioner
My First Shield • WIN a Rohde & Schwarz 100MHz Digital Oscilloscope
TH LED Matching & Sorting Accessory • Microcontroller BootCamp
Blown Fuse Indicator • 500-ppm LCR Meter: Feedback
Hewlett Packard 71B Number Cruncher (1984)
Arduino is a Tool • LM317 Turns 78xx
IR Tester with Solar Cell Power Supply
Stage Tuner for Guitars
& More.





US $14.00 - Canada $14.00

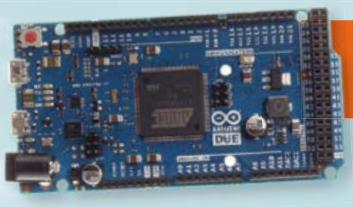
07

0 56698 24965 8

Arduino

Now Available @ Elektor!

10% OFF for GREEN and GOLD Members



ARDUINO DUE

32-bit power thanks to an ARM processor

Features	
Microcontroller	AT91SAM3X8E
Operating Voltage	3.3V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 12 provide PWM output)
PWM Channels	12
Analog Input Pins	12
Analog Outputs Pins	2 (DAC)
DC Current per I/O Pin	130 mA
DC Current for 3.3V Pin	800 mA
DC Current for 5V Pin	800 mA
Flash Memory	512 KB (all available for the user applications)
SRAM	96 KB (two banks: 64KB and 32KB)
Clock Speed	84 MHz

£46.95 • € 52.90 • US $72.00

ARDUINO LEONARDO

Especially good for USB applications

Features	
Microcontroller	ATmega32u4
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	20 (of which 7 provide PWM output)
PWM Channels	7
Analog Input Pins	12
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (of which 4 KB used by bootloader)
SRAM	2.5 KB
EEPROM	1 KB
Clock Speed	16 MHz

£21.95 • € 24.90 • US $34.00

ARDUINO ETHERNET

Networking has never been easier

Features	
Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 4 provide PWM output)
Arduino Pins reserved	10 to 13 used for SPI 4 used for SD card 2 W5100 interrupt (when bridged)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (of which 0.5 KB used by bootloader)
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

£46.95 • € 53.90 • US $73.00

ARDUINO MEGA

Like the Uno but with more memory and I/O

Features	
Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB (of which 8 KB used by bootloader)
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

£46.95 • € 52.90 • US $72.00

ARDUINO UNO REV.3

The most popular board with its ATmega328 MCU

Features	
Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Channels	6
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (of which 0.5 KB used by bootloader)
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

£23.95 • € 27.50 • US $38.00

ARDUINO YÚN

Two processors

Features AVR Arduino microcontroller	
Microcontroller	ATmega32u4
Operating Voltage	5V
Input Voltage	5V
Digital I/O Pins	20
PWM Channels	7
Analog Input Channels	12
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (of which 4 KB used by bootloader)
SRAM	2.5 KB
EEPROM	1 KB
Clock Speed	16 MHz

Features Linux microprocessor	
Processor	Atheros AR9331
Architecture	MIPS @400 MHz
Operating Voltage	3.3V
Ethernet	IEEE 802.3 10/100Mbit/s
WiFi	IEEE 802.11b/g/n
USB Type-A	2.0 Host/Device
Card Reader	Micro-SD only
RAM	64 MB DDR2
Flash Memory	16 MB
PoE compatible 802.3af card support	

£60.95 • € 69.95 • US $95.00

ARDUINO

elektor

Further Information and Ordering at www.elektor.com/development/arduino

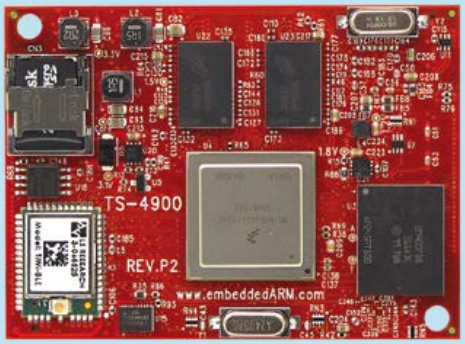
SUPERIOR EMBEDDED SOLUTIONS

Technologic Systems

DESIGN YOUR SOLUTION TODAY
CALL 480-837-5200

www.embeddedARM.com

Computer-on-Modules
State of the Art Embedded Design



pricing starts at
$139
qty 1
$99
qty 100

- Up to 1.2GHz ARM w/ 2GB RAM
- Fanless, low power designs
- All parts soldered on, no moving parts
- DoubleStore SD for reliable storage
- User-programmable FPGAs
- -40 to +85C Industrial temperature range
- Easy development w/ Android, Debian and Linux
- Sub-second boot to Linux
- COTS carrier boards available or design a custom solution with reduced design time and complexity

Computer-on-Modules include:

- TS-4200: Atmel ARM9 w/ super low power
- NEW! TS-4600: 450MHZ low cost w/ 2 Ethernets
- TS-4710: Up to 1066MHz PXA168 w/ video
- NEW! TS-4712: like TS-4710 + 2 Ethernets
- NEW! TS-4720: like TS-4712 + 4GB eMMC Flash
- TS-4800: 800MHz FreeScale iMX515 w/ video
- NEW! TS-4900: 1.2GHz QuadCore i.MX6 w/ WiFi

NEW!

Single Board Computers
TS-7680 Industrial Computer

pricing starts at
$203
qty 1
$159
qty 100

Features can include:

- Up to 454 MHz ARM CPU
- Up to 256 MB RAM
- 24V AC Power Input
- WiFi and Bluetooth Radio
- 2GB Flash Storage
- mSD Card Socket
- 2x Ethernets
- 6x 30V DIO Ports
- 2x CAN Ports
- 1x Modbus Port
- 2x 3A Relays
- 2x RS-485 Ports

Benefits:

- Converter-less power direct to 24V AC voltage
- Solid screw terminals for demanding environments
- Highly connected with WiFi/Bluetooth and Dual Ethernet
- -40°C to 80°C, 100% soldered down components
- Easy development w/ Debian and Linux 2.6
- Boots to Linux shell in under 5 seconds
- Guaranteed availability until 2025

LONG LIFE — We've never discontinued a product in 30 years

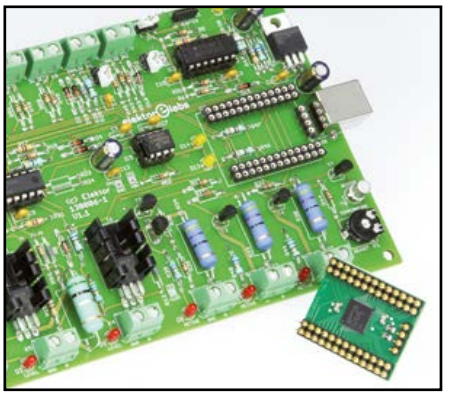
RUGGED — Embedded systems that are built to endure

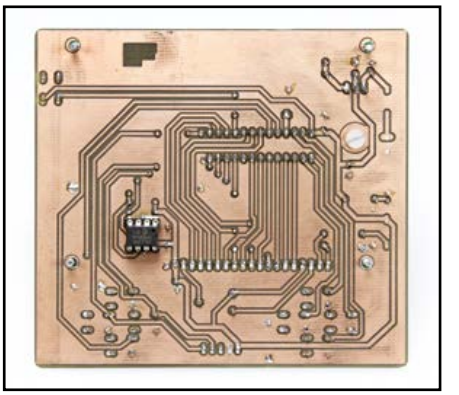
OPEN — Support every step of the way with open source vision

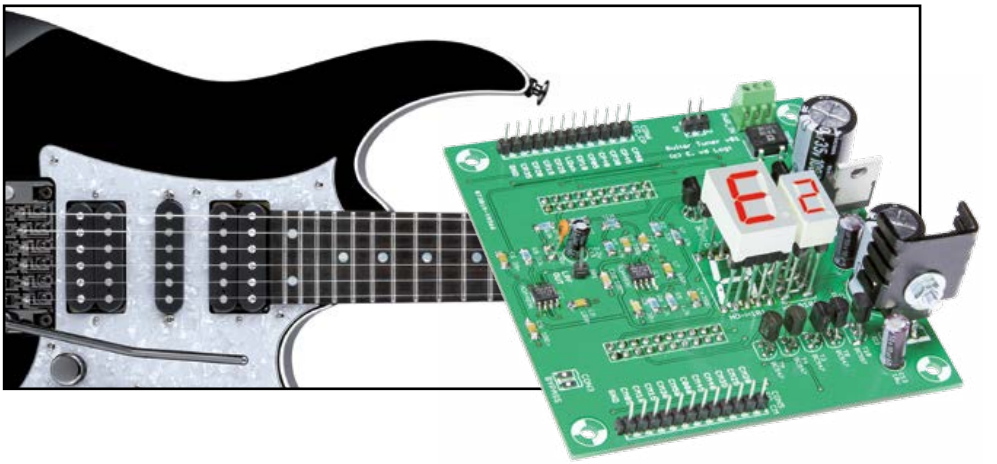
ORIGINAL — Unique embedded solutions add value for our customers

Contents

elektor magazine

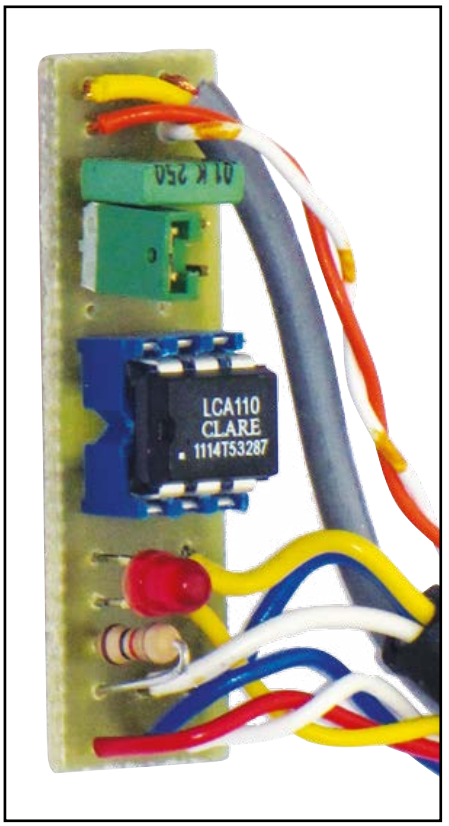


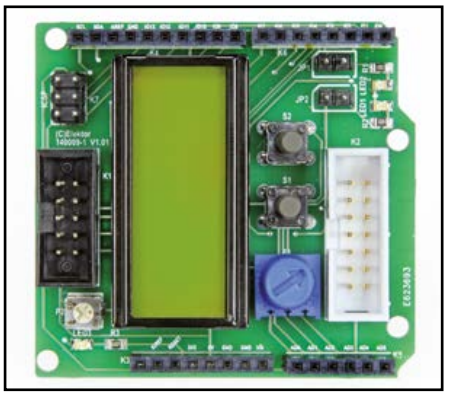


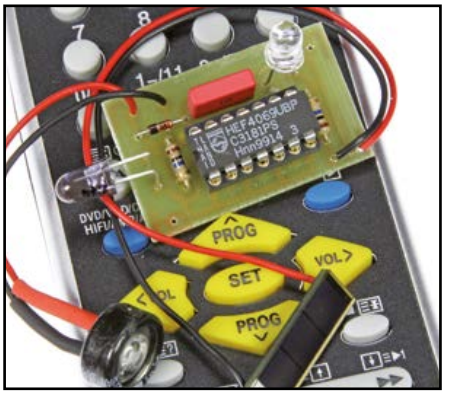


Advertorial

Project Generator







Labs

Industry

Regulars

DesignSpark

Volume 40, No. 451 & 452
July & August 2014

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

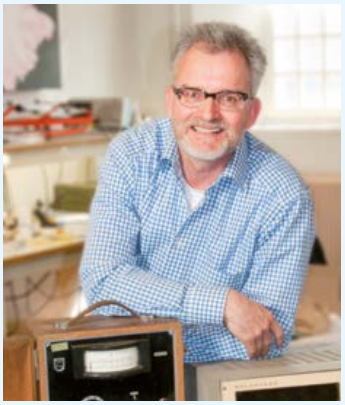
USA / Canada Advertising:
Peter Wostrel
Phone: 1.978.281.7708
E-mail: peter@smmarketing.us

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.

Printed in the USA Printed in the Netherlands

The Eye of the Beholder



Assuming you start reading this edition right here (which frankly I have seen few engineers do), there's a true melting pot of hands-on electronics on the pages ahead. In good Project Generator Edition (PGE) tradition, the assortment of articles and projects is as diverse as I was able to make it in terms of complexity and length. This year I am again happy to respond positively to recurrent requests from readers for Sunday afternoon projects with a fun character but still educational and with an Elektor twist. On the other hand, there are also 'mega' projects on these pages to challenge all of you ambitious programmers and fans of embedded electronics. Both types of article, and some medium-sized ones too, I am assigned to line up on—what I dream of—any number of pages. However, my colleagues from the advertising department also want their share, and the printers and accountants tell me there's an absolute limit of 132 pages bound & wrapped to arrive by mail at your home or office, not a day late.

While I would agree to the limited scientific significance of modding an LM317 to look & act like a 78xx (page 99), or building a clock from CD4000 ICs (page 22) I am sure both articles contain a little something to trigger the electronics engineer in you, if only that one design aspect I have forgotten to highlight or describe in full. Although publishing about electronics and reading about it may seem the two extremes of a lengthy process, they have Discovery in common. The ability to discover should be respected and remain a key element in every sense, in electronics across the entire range. In practice, one man's e-debris in brownish plastic is another man's gem (page 122), likewise one LED and a triac (page 73) as opposed to a 54-pin IO-Warrior (page 10). It's true—the articles in this PGE are best read in random order.

Happy reading,

Jan Buiting, Editor-in-Chief

The Team

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

USA
Don Akkermans
+1 860-289-0800
d.akkermans@elektor.com

United Kingdom
Don Akkermans
+44 20 7692 8344
d.akkermans@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
Ferdinand te Walvaart
+31 46 43 89 444
f.tewalvaart@elektor.nl

Spain
Jaime González-Arintero
+34 6 16 99 74 86
j.glez.arintero@elektor.es

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

Sweden
Carlo van Nistelrooy
+31 46 43 89 418
c.vannistelrooy@elektor.com

Brazil
João Martins
+31 46 4389444
j.martins@elektor.com

Portugal
João Martins
+31 46 4389444
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

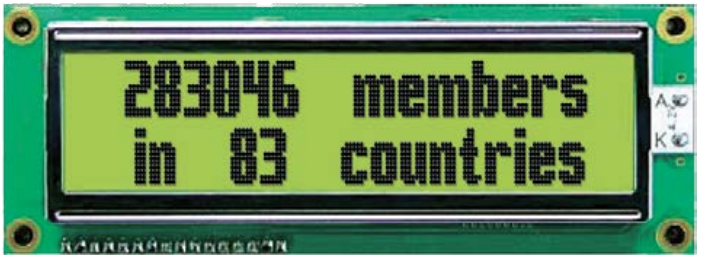
South Africa
Johan Dijk
+31 6 1589 4245
j.dijk@elektor.com

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



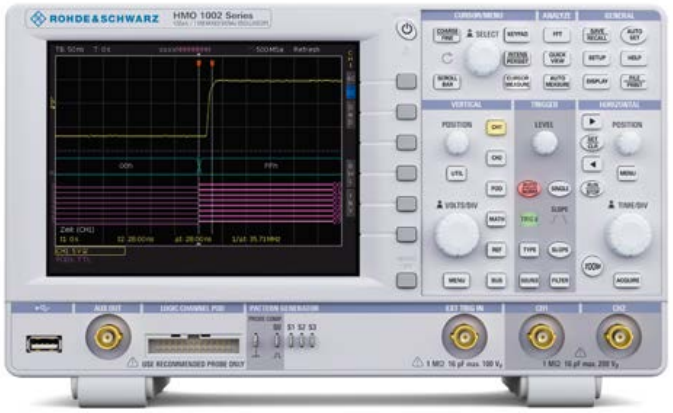
Supporting Companies

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

New Digital Oscilloscope from Rohde & Schwarz

High quality — Best value!



Rohde & Schwarz will be giving away a top-spec R&S®HMO1002 oscilloscope (with the maximum bandwidth!) to ten lucky Elektor readers! To be in with a chance just register today at www.elektor.com/hmo1002 it could be your lucky day! Win a brand new R&S®HMO1002, developed by Hameg Instruments and the latest member of the *Scope of the Art* family!

The new digital Oscilloscope type R&S®HMO1002 from **Rohde & Schwarz** is available with bandwidths up to 50 MHz, 70 MHz and 100 MHz. The design stands out for its fast Waveform-Update-Rate (screen waveform capturing) and high vertical-input sensitivity. The fan-less design offers a 1 GSample/s sample rate and a memory depth of 1 MSamples. In common with all oscilloscopes in the R&S®HMO series it offers the versatility of mixed signal functionality.
Its host of additional functions makes the oscilloscope an ideal diagnostic and teaching tool for a wide range of applications for students, service technicians and for developers of embedded systems.

A built-in *three digit DVM* function simplifies the service technician's work and cuts down on work bench clutter while giving the R&S®HMO1002 the ability to make simultaneous voltage measurement on two analog channels.
Its *function generator* is particularly useful in educational environments, producing all elementary waveforms at frequencies up to 50 kHz. Trainees, graduate and undergraduate students will find it useful to solve a variety of measurement tasks.
An integrated *pattern generator* makes it easy for developers of embedded systems to configure and generate protocol telegrams at speeds up to 50 Mbit/s. This gives developers the freedom to program customized waveforms together with the supported predefined serial communication protocols.
Integrated in the R&S®HMO1002 is an optional hardware-accelerated serial bus signal trigger and decoder with support for some of the most popular communication protocols (I^2C, SPI, UART, CAN or LIN).
Thanks to its 128 Kmeasurement points the HMO1002 *FFT-Analyzer function* can hold its own against larger oscilloscopes, its representation of the time signal, measurement window and analysis region of the FFT with on-screen results greatly simplifies spectra measurements.

To make full use of its MSO capabilities a separate eight channel logic probe type R&S®HO3508 is necessary. The probe is compatible across the complete HMO-range of equipment and gives the R&S®HMO1002 complete time-domain, logic, protocol and frequency analysis functionality in the same device, making it a member of the Rohde & Schwartz 'Scope of the art' family of devices. Recommended retail prices range from 798 euros for the economy model with 50 MHz bandwidth to 998 euros for the instrument with 100 MHz bandwidth. US dollar or UK pound prices are to be announced at the time of writing *(Ed.)*.

Main Features

- Models available with 50 MHz, 70 MHz and 100 MHz Bandwidth
- 1 GSample/s Sampling Rate, 1 MSample memory depth R&S®
- High vertical sensitivity up to 1 mV/Div
- Fast error signal detection rate at 10,000 Waveforms/s
- Wide choice of automated measurement functions
- Quickview: all important signal parameters available with one press
- Mixed-Signal-functionality as standard
- Targeted trigger on signal events
- Versatile bus analyzer option for isolating specific data packets
- FFT: the simple path to spectral analysis with 128k measurement points
- Digital Voltmeter for simultaneous measurement of two analog channels
- The right signal, where you need it: 50 Mbit/s pattern generator or 50 kHz function generator.

HOT CHIPS 26

ADVANCE PROGRAM

August 10-12, 2014

A Symposium on High-Performance Chips

Flint Center for the Performing Arts-Cupertino,CA

http://www.hotchips.org

HOTCHIPS brings together designers and architects of high-performance chips,software, and systems. The tutorial and presentation sessions focus on up-to-the-minute developments in leading-edge industrial designs and research projects.

Sunday Aust 10

Tutorial 1: Emerging Trends in Hardware support for security

- Security Overview — **Princeton**
- Mobile HW Security — **Arm**
- Secure Systems Design — **AMD**
- Hardware assists for introspection — **Intel**

Tutorial 2: Internet of Things

- Powering the Internet of Things — **TI**
- Ultra Low Power Design Approaches for IoT — **National University of Singapore**
- Connecting the IoT — **Qualcomm**
- IoT Systems Architecture: It's not just "embedded with a radio" — **ARM**

Monday August 11

High Performance Computing

- SC-ACE Processor: NEC's Brand-New Vector Processor — **NEC**
- SPARC64 XIfx: Fujitsu's Next Generation Processor for HPC — **Fujitsu**
- Anton2:A 2nd-Generation ASIC for molecular Dynamics Simulation — **D.E. Shaw Research**

Keynote 1 **Power Constraints: From Sensors to Servers**
Peter Hutton — **ARM**

Mobile Processors

- NVIDIA's TegraK1 System-on-Chip — **NVidia**
- Aplying AMD's "Kaveri" APU for Teterogeneous Computing — **AMD**
- NVIDIA's Denver Processor — **NVidia**

Technology

- HBM: Memory Solution for Bandwidth-Hungry Processors — **SK Hynix Inc**
- Improved 3D chip stcking with ThruChip wireless connections — **Thruchip Communications**
- CMOS Biochips for Point-of-Care Molecular Diagnostics — **Insilixa**

Arm Servers

- The AMD Opteron "Seattle": A 64b ARM Dense Server Processor — **AMD**
- ARM Next-Generation IP Supporting LSI's HIgh-End Networking — **ARM, LSI Logic**
- X-Gene2: 28mm scale-out processor — **Applied MIcro**

Tuesday August 12

FPGAs

- Design of a High-Density SOC-FPGA at 20nm — **Altera**
- A Reconfigurable Fabric for Accelerating Datacenter Services — **Microsoft**
- Xilinx FPGAs case study: High capacity and Performance 20nm FPGAs — **Xilinx**
- SDA: Software-Defined Accelerator for Large-Scale DNN Systems — **Baidu**

High PErformance ASICS

- Hardware Accelerated Text Analytics — **IBM**
- Myriad2 "Eye" of the Computational-Vision Storm — **Movidius**
- Goldstrike 1: A 1st Generation Cryptocurrency Processor for Bitcoin Mining — **Cointerra**
- RayChip: Real-time Ray Tracing Chip for Embedded Applications

Keynote 2 The Internet of Everything: What is it? What's driving it? What comes next?
Rob Chandhok — ***Qualcomm***

Dense Servers

- SCORPIO: 36-Core Shared-Memory Processor with a Coherent Mesh — **MIT**
- Next Generation Oracle SPARC Processor — **Oracle**
- Oracle's Next Generation SPARC Cache Hierarchy — **Oracle**
- Atom C2000 Microserver: Power Efficient Data Center Processing — **Intel**

Big Iron Server

- Performance Characteristics of the POWER8 Processor — **IBM**
- Unchaining the data center with Open Power: Reengineering a server ecosystem — **IBM**
- IvyBridge Server: Delivering Performance from Workstations to Mission Critical — **Intel**

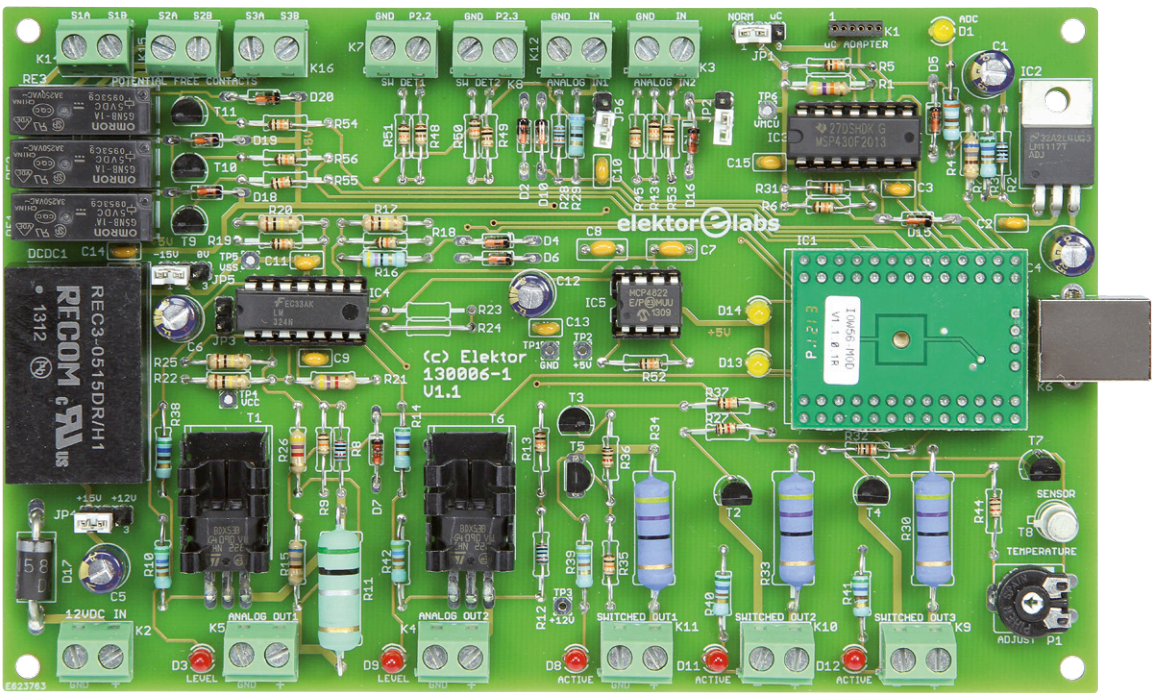
A Symposium of the Technical Committee on Microprocessors and Microcomputers of the IEEE Computer Society and the Solid-State Circuits Society

IO-Warrior Expansion Board

Measurement and control using your (old) PC

By
Franz-Peter Zantis
(Germany)

Don't throw out your old PCs and notebooks or leave them gathering dust in the basement! They can be a useful resource: by adding this universal interface card an old PC can be pressed into service as a measurement and control hub. An IO-Warrior module on the I/F board takes care of USB communication, and source code is available that works with the free version of Visual Studio.



There are many measurement and control interface cards available, each with their own particular advantages and disadvantages. The card described here has many positive aspects: it is simple to control using familiar programming languages such as Visual Basic, C# and C; sensors and actuators can be wired directly to the terminals on the board without using special connectors; and the hardware is easy to modify and adapt for a wide range of different applications.

The basics

First we shall look at a few basic requirements for an interface card of this type. It should be a standard Eurocard size (100×160 mm; 3.94x6.30 inch) and able to be connected to the **USB** port on any PC. A chip called the 'IO-Warrior56', from the German company Code Mercenaries, is used to handle the USB protocol on the board, which removes the need for complex programming. The chip is very versatile and is easy to talk to using a Visual Basic program.

The card should have **outputs** that can be connected directly to low-voltage loads (up to 12 V) without additional hardware, and which can supply power, derived from an external power supply. Two of the outputs can be set to any voltage from 0 V to 12 V under program control. There are also three switching outputs of which one switches to ground. Loads can be connected to these outputs either directly or via a relay. AC line (115 V or 230 V) loads can be controlled via three further potential-free switching outputs.
Two digital **inputs** allow switch settings to be read, and two analog input voltages can be read using an A/D converter. The interface card also has a built-in temperature sensor using a silicon transistor in an old-style metal package as a sensor.

Putting it into practice

The upshot of the above requirements is the circuit shown in **Figure 1**. We need a programming language to talk to the interface card, and in the following examples we will use Visual Basic. More specifically we will use Microsoft Visual Studio Express, which supports Visual Basic as well as C## and C++, and which can be downloaded for free from Microsoft's website [1]. Programming can equally well be done under Linux as under Windows. Introductions to programming in Visual Basic can be found on the Internet and elsewhere.
The USB chip [2] comes in two variants: as an SMD in an MLFP56 package and as a ready-made module (**Figure 2**), which connects directly to a USB cable. In order not to have to deal with soldering the 56 tiny pads on the SMD, we have chosen to use the module, which is simply mounted upside down in a socket.
The interface board is powered over its USB connection, with the help of an on-board DC/DC converter to generate the symmetrical +15 V and −15 V rails needed for the operational amplifiers from the USB's +5 V. Strictly speaking the negative supply is not really necessary as the type LM324 operational amplifier used can drive its output to within a few millivolts of ground. Jumpers JP4 and JP5 allow the supply voltages to be set at ±15 V, or at +12 V/0 V from the external supply. In the second case the maximum voltage available on the analog outputs will only be approximately +9 V, since the output of the operational amplifier can only swing to within about 3 V of its positive supply; on the other hand, the DC/DC converter can then be omitted.

Two yellow LEDs (D13 and D14) are connected to pins on the IO-Warrior and can be used to indicate the status of the system.
Overall the interface board is clearly laid out and easy to construct. We have avoided using SMDs in the interests of making the design suitable for those who have less experience with the soldering iron. All the ICs (with the exception of the regulator) and the USB module are fitted in sockets. It is important to use high-quality header sockets for the module to avoid possible problems with unreliable connections from the outset.
The unpopulated board (Elektor Store no. 130006-1) is available from the Elektor Shop [3]. To simplify matters for constructors, we have also made the IO-Warrior module and the TI microcontroller, programmed to act as an A/D converter, available there (nos. 130006-91 and 130006-41 respectively). The project software (no. 130006-11) is available for download: as well as library modules the download also includes example code and a description of an experimental set-up for measuring temperature.

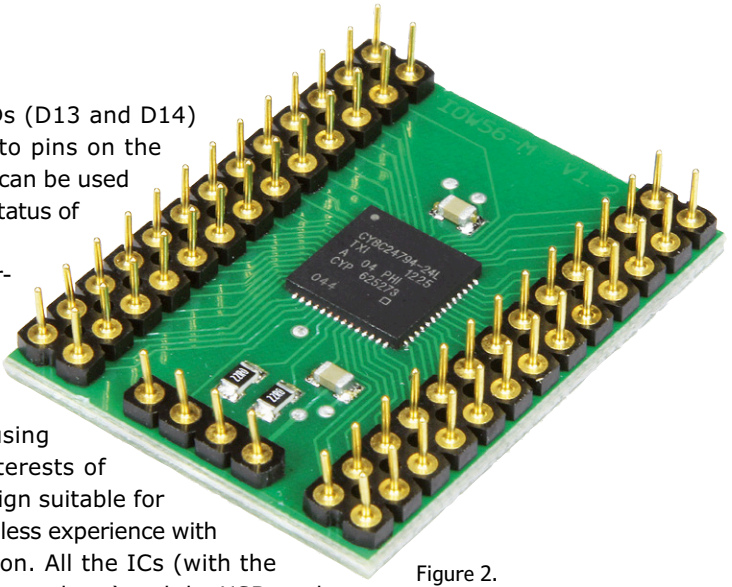


Figure 2.
The IO-Warrior in module form is easy to use.

Switching outputs

Once the job of populating the board is complete there are a few more steps to take before embarking on writing a program. The library `iowkit56.dll` in the download ([3]) contains the driver for the IO-Warrior module: it forms the glue between the application program and the module itself. To ensure that the library can be accessed it must either be placed in a directory on one of the standard system paths or be included in the Visual Basic project. The latter option has the advantage that the finished program can be copied as a single directory to another computer and run there without further fuss. **Figure 3** shows the arrangement of files within a Visual Basic project, here called 'Motorsteuerung' ('Motor control').
The next step is to add a Visual Basic module to the project which includes declarations of functions and subroutines. Doing this saves a lot of typing and searching through documentation.

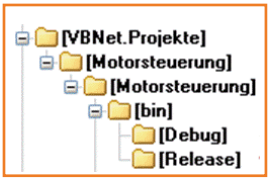


Figure 3.
Structure of a Visual Basic project.

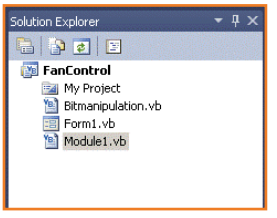


Figure 4.
Including Module1.vb in the project.

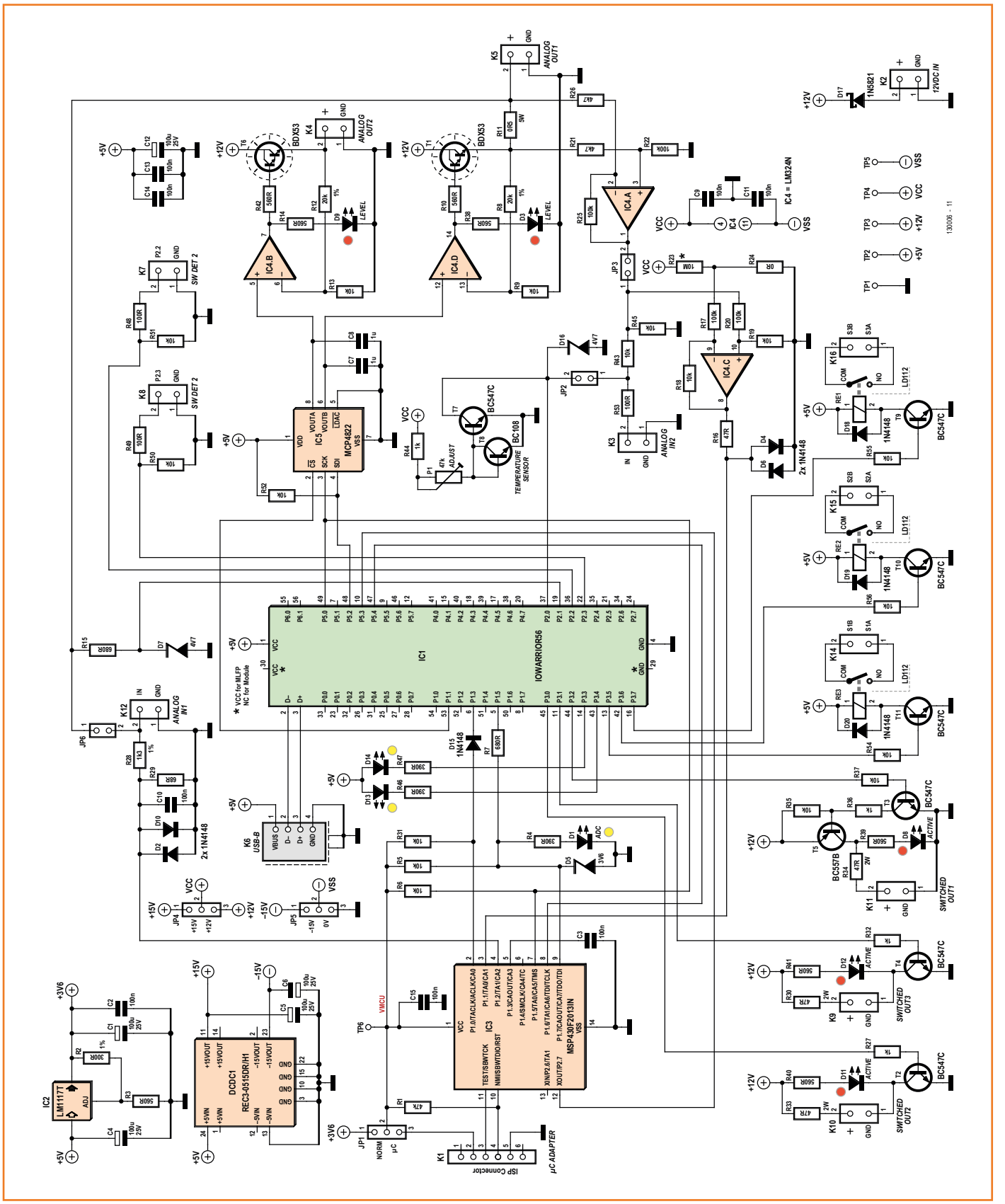

IC1
IOWARRIOR56
IC3
MSP430F2013IN
IC4
IC4 = LM324N
IC5
MCP4822
IC2
LM1117T
DCDC1
REC3-0515DR/H1
ANALOG OUT1
ANALOG OUT2
ANALOG IN1
ANALOG IN2
12VDC IN
SW DET 2
TEMPERATURE SENSOR
ADJUST
ISP Connector
µC ADAPTER
SWITCHED OUT1
SWITCHED OUT2
USB-B
VCC
VSS
130006 - 11

Table 1. Parameters to the function `IowKitWrite()`	
`intret`	An integer variable that will contain the return value from the function call after its execution, giving the number of bytes transferred.
`IOhandle`	The handle number as obtained from an earlier call to `IowKitOpenDevice()` (see text for more information).
`0`	For operation in standard mode this parameter is always zero.
`IOdata(0)`	The prepared array and the starting index within it from which data are to be transferred.
`8`	The number of bytes to be transferred: always 8.

Figure 4 shows the source code files for the project 'FanControl': note the presence of the module `Module1.vb` in the project.
The function `IowKitOpenDevice` is called to determine whether the IO-Warrior is connected to the computer. The function tries to open a connection to the IO-Warrior and returns an integer 'handle'. If the handle returned is zero then no device was found: in this case an error can be reported to the user (see **Listing 1**).

It is possible to call this function from time to time as the program is executing, to check that the hardware is still connected. If the USB connection is even briefly interrupted communication between the software and hardware will stop. The next call to `IowKitOpenDevice` will then re-open the connection and communication can continue. One possibility is to provide a button in the user interface labeled 'Connect Hardware' to trigger a call to the function.

With that we have completed our preparations and can move on to the programming proper.
First we will test the switching outputs. The IO-Warrior has six ports (Port 0 to Port 5) with eight I/O pins (labeled Px.0 to Px.7) each, and a further port (Port 6) with two I/O pins. Pins P5.0 to P5.4 are reserved for the SPI bus.
We use Port 3 for switching loads and for controlling LEDs D13 and D14. To begin with we will drive the two LEDs: in doing this we will show how the individual pins of the IO-Warrior can be controlled from within Visual Basic. The IO-Warrior supports standard I/O operation and a special (SPI and I²C) mode. We will use standard mode to operate the I/O pins in this example. The LEDs are powered from the 5-V USB supply and driven from ports P3.3 and P3.4 via 390-Ω resistors.
A byte array is used to specify which pins of each port are set to a high or a low level, with the states for the eight pins of one port packed into a single byte for sending to the IO-Warrior. An additional byte at the beginning of the array is used to indicate that normal mode is to be used. The total size of the array is therefore eight bytes. Since the array is indexed from zero its declaration is as follows:

```
Dim IOdata(7) As Byte
```

The elements of the array can be set to whatever values are required. To light the LEDs we must set their port pins to logic high: the corresponding bit pattern is 00011000. This is 24 in decimal of 18 in hexadecimal (written &H18 in Visual Basic). **Listing 2** shows how the elements of the array are set. Note that Port 1 (`IOdata(2)`) is set to all ones: this is because this port is used for the chip select (CS) signals for devices connected on the SPI bus, and chip select is an active-low signal. The array can now be sent to the IO-Warrior. For this we use the function `IowKitWrite`:

```
intret = IowKitWrite(IOhandle, 0,
IOdata(0), 8)
```

The meaning of the arguments to the function is explained in **Table 1**.
To turn the LEDs off, we need to take pins P3.3 and P3.4 to ground. The example program in **Listing 3** includes a loop that executes a fixed number of times: when the start button is clicked the LEDs should blink one hundred times at approximately 0.5 Hz.
The other switching outputs are controlled in exactly the same way. To switch on a load con-

Figure 1. Circuit of the interface card with its multitude of inputs and outputs.

Listing 1. The function IowKitOpenDevice().

```
IOhandle = IowKitOpenDevice()
  If IOhandle = 0 Then
     MessageBox.Show("No hardware found!", "ERROR", MessageBoxButtons.OK, _
       MessageBoxIcon.Error, MessageBoxDefaultButton.Button1)
     End
     Else

- Code to be executed when hardware recognized -

  End If
```

Listing 2. Array of port configuration settings.

```
IOdata(0) = 0     'Byte for parameterizing, in Normal-Mode always 0
IOdata(1) = 0     'Byte for Port0
IOdata(2) = 255   'Byte for Port1; all at 1, Port used for Chip-Select
IOdata(3) = 0     'Byte for Port2
IOdata(4) = &H18  'Byte for Port3; connectors for LEDs D13 and D14 to logic high
IOdata(5) = 0     'Byte for Port4
IOdata(6) = 0     'Byte for Port5
IOdata(7) = 0     'Byte for Port6
```

Listing 3. Pressing the button generates 100 LED flashes.

```
Dim IOdata(7) As Byte  'Array containing the port states
Dim IOHandle As Integer'variable for the Handle Number
Dim intret As Integer  'variable for Integer Return-Value
Dim n As Integer    'variable for counter

Private Sub BtnStart_Click(sender As System.Object, e As System.EventArgs) _
Handles BtnStart.Click
IOhandle = IowKitOpenDevice()
       If IOhandle = 0 Then
         MessageBox.Show("No hardware found!", "ERROR", MessageBoxButtons.OK, _
MessageBoxIcon.Error, MessageBoxDefaultButton.Button1)

           End 'program termination, if the hardware is not connected
Else
      For n = 1 to 100 'For-Next-loop
     IOdata(4) = &H18    'LEDs D13 and D14 on +5V: LEDs are off
       intret = IowKitWrite(IOhandle, 0, IOdata(0), 8)
            System.Threading.Thread.Sleep(1000) '1000 ms to wait

    IOdata(4) = 0      'all Pins of Port 3 on GND: LEDs are on
```

```
    intret = IowKitWrite(IOhandle, 0, IOdata(0), 8)
    System.Threading.Thread.Sleep(1000) '1000 ms to wait
  Next n
       End If
End Sub
```

Listing 4. Driving a remote-controlled mains switch.

```
Dim IOdata(7) As Byte  'Array with port states
Dim IOHandle As Integer'variable for handle-number
Dim intret As Integer  'variable for integer return-value

Private Sub BtnStart_Click(sender As System.Object, e As System.EventArgs) _
Handles BtnStart.Click
IOhandle = IowKitOpenDevice()
       If IOhandle = 0 Then
         MessageBox.Show("No hardware found!", "ERROR", MessageBoxButtons.OK, _
MessageBoxIcon.Error, MessageBoxDefaultButton.Button1)

          End 'program termination, if no hardware found
Else
  IOdata(4) = &H20      'P3.5 on logical 1; Relay is active
      intret = IowKitWrite(IOhandle, 0, IOdata(0), 8)
          System.Threading.Thread.Sleep(500) '500 ms to wait
  IOdata(4) = 0       'all Pins of Port3 to GND; Relay de-energized
  intret = IowKitWrite(IOhandle, 0, IOdata(0), 8)
       End If
End Sub
```

nected to OUT1, OUT2 or OUT3, the corresponding port pin (P3.0, P3.1 or P3.2) should be taken high. OUT1 switches to ground; OUT2 and OUT3 switch to +12 V. Schottky diode D17 protects against reverse polarity connection of the +12 V supply.

Logic Highs on pins P3.5, P3.6 and P3.7 cause the potential-free contacts brought out to K14, K15 and K16 to close. One interesting idea for using these is in conjunction with a radio-controlled mains switch, allowing mains appliances to be controlled without having to worry about high voltages. Open up the remote control and connect the potential-free contacts on the interface board across its pushbuttons. Using the program shown in **Listing 4**, the contacts will close briefly (for about 500 ms) when the Start button is pressed.

Analog outputs

Any voltage between 0 V and +12 V can be generated at K4 and K5 (ANALOG OUT1 and ANALOG OUT2). The circuit uses a type MCP4822 dual D/A converter controlled over the SPI bus. When the device receives a 16-bit word over the bus, it interprets the lower twelve bits (bits 0 to 11) as the desired output level and the top four bits (bits 12 to 15) as control information: see **Table 2** for details.

Output VOUTA is connected to operational amplifier IC4.B, which multiplies the voltage by a factor of 3. Feedback resistor R12 is connected directly to the terminal of K4, and so the operational amplifier also compensates for the base-emitter voltage drop in the output transistor T6. As a result the voltage on K4 should be exactly three times the voltage at VOUTA. The maximum possible voltage is 4.096 V multiplied by three, or

Table 2. Control bits for the D/A converter	
Bit 15	0: data are sent to DAC A, with output voltage at VOUTA
	1: data are sent to DAC B, with output voltage at VOUTB
Bit 14	not used
Bit 13	0: voltage range at output is 0 V to 4096 mV
	1: voltage range at output is 0 V to 2048 mV
Bit 12	0: output buffer is disabled: output is high impedance
	1: output buffer is enabled

12.288 V. To achieve this the voltage provided at K2 must be rather higher (for example +15 V) as there will be a drop of up to 2 V between the collector and emitter of T6 when it is in saturation.

In some circumstances the transistor can dissipate a considerable amount of power and it is therefore fitted with a small heatsink. The smaller the output voltage and the higher the output current, the higher the power dissipation in the transistor will be. Of the voltage supplied at K2 around 400 mV is dropped across D17 and the remainder is dropped across the output transistors. Note that the outputs are not protected against short circuits and so it is a good idea to use a current-limited power supply. In any case it is a good idea to proceed with care when using these outputs. LEDs D9 and D3 change in brightness according to the voltage on the outputs.

The voltage on each output (VOUTA or VOUTB) is determined by a number, from zero to 4095, presented to the corresponding D/A converter. The number multiplied by three gives the output voltage in millivolts, and so the output resolution is 3 mV. There are five steps to carry out when using the D/A converters, as follows.

Prepare a 16-bit integer value

The 16-bit value that is sent to the D/A converter comprises four control bits and the twelve data bits that correspond to the desired voltage. The easiest way to construct the word is to put the voltage value in a variable and then set the top four control bits as required. SPI communications from the USB chip happen one byte at a time, and so the integer value has to be split into a high byte and a low byte. The module `Bitmanipulation.vb` in the software download includes functions to set, clear, toggle and extract individual bits in a byte, as well as a function for splitting a 16-bit integer into a high byte and a low byte.

When the program given in **Listing 5** is run, the voltage on output K4 (ANALOG OUT2) should be set to 3.6 V. The value sent to the D/A converter is 3600 mV/3 = 1200. In this example the two bytes are stored in the array `msblsb`.

Configure the SPI bus

Listing 6 shows how the SPI bus is initialized. The clock frequency and a few other parameters regarding the clock and data signals need to be configured: the parameters are stored in a 64-byte array and then sent to the chip.
When the SPI bus is used the IO-Warrior has to be operated in its 'special' mode. In this mode the parameters to the functions `IowKitWrite` are different from those used when configuring normal I/O ports. `IOhandle` has the same meaning; the next parameter is 1, which enables the special mode; then follows the array with the index starting from which bytes are to be sent; and then the final parameter (64) is the count of bytes to be sent.

One point to note regarding `w56init(4)`: the USB chip is run from a 5 V supply, while the microcontroller that acts as an A/D converter (see below) runs on 3.6 V. To ensure that excessive voltages are not presented to the microcontroller the resistors on the SPI lines in the IO-Warrior that pull the signals up to its supply rail must be disabled.

Activate chip select

Next communications with the D/A converter are enabled by activating its chip select signal (pin 2 on the converter, connected to P1.1 on the IO-Warrior): see **Listing 7**.

Send the 16 bit integer

The data to be sent must be stored in a 64-byte array as before. As **Listing 8** shows, three further configuration bytes are needed, which include the number of bytes to be sent over the SPI bus.

Listing 5. Preparing values for the D/A converter.

```
Dim curspgch2 As UShort 'variable for desired voltage value (16-Bit Integer)
Dim vmsblsb(1) As Byte  'Array for receiving high-byte and low-byte
Dim curspgch2 As UShort = 1200     'desired output-voltage in mV divided by 3
vmsblsb = findHbyteLbyte(curspgch2)'separation in high-byte and low-byte with the help of
                                   "Bitmanipulation.vb"
'now follows the setting of the control bits for the DAC as shown in Table 3.3
msblsb(1) = ClearBit(msblsb(1), 7)   'Bit15: data are sent to DACA
msblsb(1) = ClearBit(msblsb(1), 6)   'Bit14: this is not relevant
msblsb(1) = ClearBit(msblsb(1), 5)   'Bit13: output voltage range to 0...4096
msblsb(1) = SetBit(msblsb(1), 4)     'Bit12: 1=OutputBuffer active
```

Listing 6. Initializing the SPI bus for the D/A converter.

```
Dim w56init(63) As Byte  'Byte-Array with 64 places

'SPI-Initialization
w56init(0) = &H8      'for SPI-Modus
w56init(1) = &H1      'SPI activated
w56init(2) = &H0      'SPI parameter to conform with the DAU MCP4822
w56init(3) = 119      'clock frequency 24MHz / (119+1) = 200 kHz
w56init(4) = &H1      'deactivate pull up resistors
retval = IowKitWrite(IOhandle, 1, w56init(0), 64)  'sending parameter to IO-Warrior
```

Listing 7. Activating Chip Select for the D/A converter.

```
'ChipSelect of IC5 (P1.1) from High to Low
w56data(2) = ClearBit(w56data(2), 1)
retval = IowKitWrite(IOhandle, 0, w56data(0), 8)
```

Listing 8. Sending data to the D/A converter.

```
Dim w56SPIdata(63) As Byte  'Byte-Array with 64 positions

'sending the data via SPI to the DAU
w56SPIdata(0) = &H9 'with SPI always &H9
w56SPIdata(1) = &H2 'number of Bytes to send; here two (high-byte and low-byte)
w56SPIdata(2) = &H0 'Flags; here 0
w56SPIdata(3) = vmsblsb(1)  'loading the high-byte in the Array to send
w56SPIdata(4) = vmsblsb(0)  'loading the low-byte in the Array to send
retval = IowKitWrite(IOhandle, 1, w56SPIdata(0), 64)
```

Listing 9. Deactivating Chip Select for the D/A converter.

```
'ChipSelect IC5 (P1.1) back to logic High
w56data(2) = SetBit(w56data(2), 1)
retval = IowKitWrite(IOhandle, 0, w56data(0), 8)
```

When this code is executed, 3.6 V should appear on connector K4.

Deactivate chip select

To complete the transaction it is necessary to deactivate the chip select signal on the D/A converter: see **Listing 9**.

It would be a good idea to integrate the code described above into a single function, that simply takes the desired output voltage as a parameter. Capacitors C7 and C8 prevent any spikes that may appear on the outputs of the D/A converters from appearing on the analog outputs of the board. If it is desired to be able to change the output voltage quickly, the values of these capacitors can be reduced, for example to 100 nF. The example code in the listings controls ANALOG OUT2; the same approach is used to control ANALOG OUT1 (on K5). The only change is that bit 15 of the word sent to the D/A converter should be set rather than cleared.

Reading switch status

K7 and K8 are connected via protection resistors to P2.2 and P2.3 respectively. It is possible to determine whether a high or a low level is present on each of these inputs. With its supply at 5 V the IO-Warrior is guaranteed to report a high level when the voltage on its input port pin is 3.25 V or more, although usually 2.25 V is enough.

Likewise it is possible to determine via port P2.1 whether a voltage is present on K5, as long as it exceeds the necessary threshold. This can be useful when K5 is used for output voltages greater than about 4 V. Resistor R15 and diode D7 protect the P2.1 input pin of the USB chip from excessively high voltages.

To read the state of the inputs the relevant pins are first set to their high-impedance (logic high) state. Each pin is internally pulled up to V_{CC} (+5 V) via a resistance of between 4 kΩ and 8 kΩ and driven from the collector of a transistor whose emitter is grounded. With the transistor turned off the level on the pin can be read by the software. One function that can be used to do this is `IowKitReadNonBlocking`. As the code in **Listing 10** shows, first the required variables are declared, then all pins on port 2 are set high, and finally the read function is called.

The result is placed in the array `w56data`. By reading values from this array and carrying out suitable comparisons the program can determine which pins are at a logic high level and which pins are low. The code always reads the state of all the pins, even though in this case we are only interested in detecting the state of pins in port 2. The code therefore only needs to inspect the value stored in `w56data(3)`.

Reading analog values

In order to read analog values the interface board makes use of a microcontroller (an MSP430F2013), programmed to act as a 16-bit analog-to-digital converter. The sample rate is 488 Hz. This approach allows the characteristics of the converter to be modified by changing the firmware in the microcontroller, but we shall not discuss this possibility further here.

The A/D converter has two inputs (pin 3 and pin 4). The analog input voltage can vary between 0 V and 600 mV, corresponding to digital codes of 0 to 2^{16}–1 (0 to 65535). Pin 3 is connected to the output of operational amplifier IC4.C. R16, D4 and D6 protect the input of the A/D converter from voltages in excess of about 700 mV. R16 has a sufficiently low value that it does not significantly affect the results of the A/D conversions while still providing protection. IC4.C is wired as a differential amplifier with the inverting input taken to ground. If desired, the values of resistors R23 and R24 can be changed to provide a DC level that will be subtracted from the voltage to be measured that appears on the non-inverting input. This can be helpful, for example, to make

Listing 10. Reading the state of inputs.

```
Dim retval As Integer
Dim w56data(7) As Byte
w56sdata(0) = 0   'always 0 for "simple mode"
w56data(3) = 255  'all Pins of Port2 at 1 (in high impedance-mode)
retval = IowKitReadNonBlocking(IOHandle, 0, w56data(0), 8)
```

fuller use of the A/D converter's 600 mV input range. The gain of the amplifier is 0.1.

The non-inverting input of the differential amplifier can be connected to different sources using jumpers JP2 and JP3.

To measure the output current at ANALOG OUT1 (on K5), fit JP3 and leave JP2 open. Like IC4.C, IC4.A is wired as a differential amplifier, and measures the voltage across shunt resistor R11. If a current of 500 mA flows, for example, the output of IC4.A will be 5 V; IC4.C will bring this

Component List

Resistors

R1 = 47kΩ
R2 = 300Ω 1%
R3,R10,R14,R38,R39,R40,R41,R42 = 560Ω
R4,R46,R47 = 390Ω
R5,R6,R9,R13,R18,R19,R31,R35,R37,R43,R45,R50,R51,R52,R54,R55,R56 = 10kΩ
R7,R15 = 680Ω
R8,R12 = 20kΩ
R11 = 0.5Ω 5W
R16 = 47Ω
R17,R20,R22,R25 = 100kΩ
R21,R26 = 4.7kΩ
R23 = 10MΩ (not fitted)
R24 = 0Ω (wire link)
R27,R32,R36,R44 = 1kΩ
R28 = 1.3kΩ 1%
R29 = 68Ω
R30,R33,R34 = 47Ω 3W
R48,R49,R53 = 100Ω
P1 = 47kΩ trimpot

Capacitors

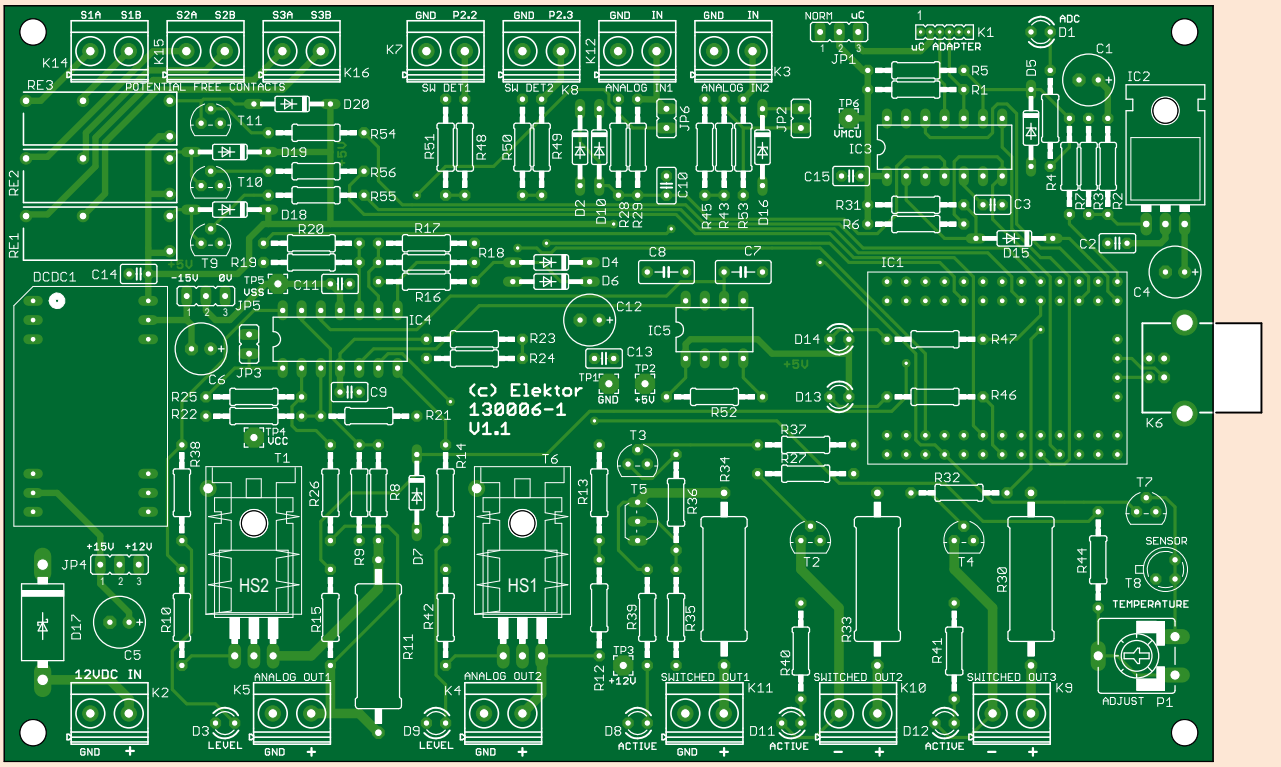
C1,C4,C5,C6,C12 = 100µF 25V
C2,C3,C9,C10,C11,C13,C14,C15 = 100nF
C7,C8 = 1µF ceramic

Semiconductors

D1,D13,D14 = LED, yellow, 3mm
D2,D4,D6,D10,D15,D18,D19,D20 = 1N4148
D3,D8,D9,D11,D12 = LED, red, 3mm
D5 = 3.6V zener diode
D7,D16 = 4.7V zener diode
D17 = 1N5821
T1,T6 = BDX53B
T2,T3,T4,T7,T9,T10,T11 = BC547C
T5 = BC557B
T8 = BC108
IC1 = IO-Warrior56-MOD (Version 1.1.0.1 or higher)
IC2 = LM1117T-ADJ
IC3 = MSP430F2013IN
IC4 = LM324N
IC5 = MCP4822-E/P
DCDC1 = AM3N-0515D or REC3-0515DR/H1

Miscellaneous

JP1,JP4,JP5 = 3-pin pinheader, 0.1"pitch
K1 = 3-pin pinheader, 0.05"pitch
K6 = USB B socket, right angled
JP2, JP3, JP6 = 2-pin pinheader, 0.1"pitch
K2,K3,K4,K5,K7,K8,K9,K10,K11,K12,K14,K15,K16 = 2-way PCB terminal block
DIP8 IC socket
DIP14 IC socket
26-way DIL socket strip, 0.1" pitch, for IC1, premium quality
4-way SIL socket strip, 0.1" pitch, for IC1, premium quality
TO220 style heatsink, 21K/W
RE1,RE2,RE3 = 5V relay, SPST-NO
Jumper
PCB # 130006-1 v. 1.1



down to 0.5 V, which can then be converted to digital by the microcontroller. The measurement range can easily be adjusted by making suitable changes to the values of resistors R21 and R26. If these are changed to 10 kΩ then the output of IC4.A at a current of 500 mA will be just 2.5 V, and hence the output of IC4.C will be at 250 mV.

To read the analog voltage at ANALOG IN2, leave JP3 open. The analog input voltage is halved by resistors R43 and R45 and then divided by ten by IC4.C. If 10 V is present on K3 the output of IC4.C will be 500 mV, suitable for the A/D converter. R53 protects T7 when a voltage is present on K3 and JP2 is inadvertently fitted.

To measure the voltage on ANALOG OUT1 (K5) JP6 must be fitted. While R15 and D7 allow the detection of voltages above 4 V on this output, jumper JP6 allows the exact value to be read. When JP6 is fitted the voltage appears across the potential divider formed by R28 and R29, which give a division ratio of 1:20. If there is 10 V on K5, then a voltage of 500 mV will appear across R29, which is connected to the second input of the A/D converter (pin 4). This input is also protected against excessive voltages by two diodes, D2 and D10.

An external voltage presented on ANALOG IN1 (K12) can also be measured by leaving jumper

Listing 11. Initializing the SPI bus for the A/D converter.

```
'SPI-initialization
w56init(0) = &H8      'configuration of SPIMode
w56init(1) = &H1      'activating SPI
w56init(2) = &H4      'SPI parameter for communication with the µC MSP430F2013
w56init(3) = 119      'clock frequency 24MHz / (119+1) = 200 kHz
w56init(4) = &H1      'deactivating the pull-up resistors
retval = IowKitWrite(IOhandle, 1, w56init(0), 64)  'sending the initialization
data to the IO-Warrior
```

Listing 12. Activating Chip Select for the A/D converter.

```
'activate ADU ChipSelect
IOdata(2) = ClearBit(IOdata(2), 3)
retval = IowKitWrite(IOhandle, 0, IOdata(0), 8)
```

Listing 13. Receiving four data bytes.

```
ldata(0) = &H9 'indicator for using SPI (always &H9)
ldata(1) = &H4 'number of Bytes to send
ldata(2) = &H0 'Flags
retval = IowKitWrite(IOhandle, 1, ldata(0), 64)
retval = IowKitRead(IOhandle, 1, ldata(0), 64)
valuePin4 = ldata(2) * 256 + ldata(3) 'value at Pin 4 of the ADU
valuePin3 = ldata(4) * 256 + ldata(5) 'value at Pin 3 of the ADU
```

Listing 14. Deactivating Chip Select for the A/D converter.

```
'resetting ChipSelect of the ADU (to logic high)
IOdata(2) = SetBit(IOdata(2), 3)
retval = IowKitWrite(IOhandle, 0, IOdata(0), 8)
```

JP6 open. As in the case of ANALOG IN2 (K3) the potential divider gives a division ratio of 1:20 between the input voltage and the voltage seen by the A/D converter.
There are thus many ways in which analog values can be acquired. Each request to the A/D converter yields a pair of numbers: one corresponding to the voltage on pin 3 and the other to the voltage on pin 4. Again the readings are transferred over the SPI bus, with pin 2 of the converter acting as its chip select signal.
The A/D converter internally generates samples of the analog values at 488 Hz but only reports them on receiving a suitable request. When this happens, LED D1 changes state, giving an indication of A/D converter activity. R7 and D5 are required because the the USB chip and the A/D converter operate from different supply voltages.

The following steps are required to fetch data from the A/D converter over the SPI bus.

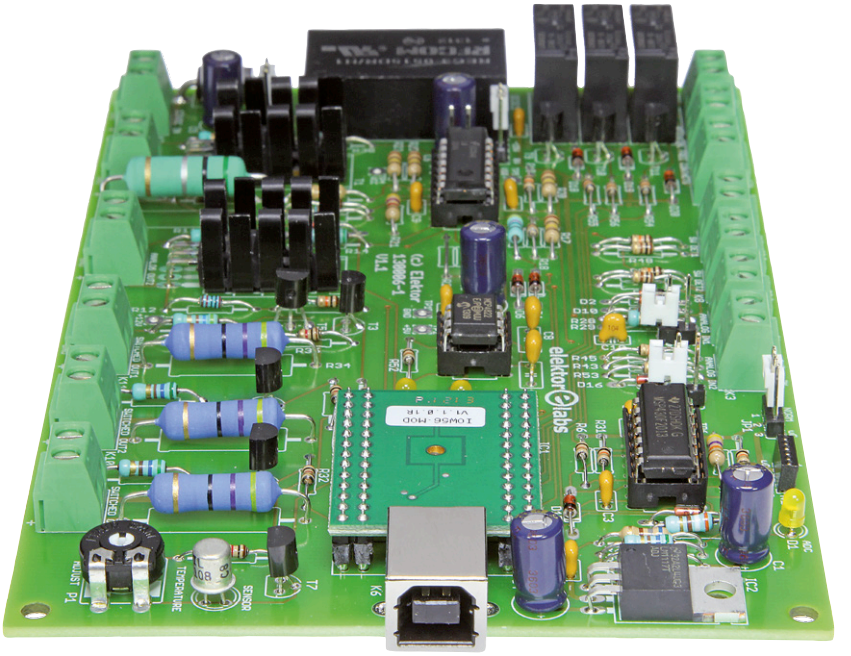


Figure 5.
The USB cable is connected directly to the IO-Warrior module.

Configure the SPI bus
This first step is similar to the configuration of the SPI bus in the case of controlling the analog outputs described above. The only difference is in `w56init(2)`, which must in this case be set to 4: see **Listing 11**.

Activate Chip Select
Again, this step is similar to the case of controlling the analog outputs. The pin that is driven in this case is P1.3: see **Listing 12**.

Read in two 16-bit integers
Two 16-bit values, one for pin 4 and one for pin 3, must be received, making a total of four bytes. As is conventional when using the SPI bus, a write command must be issued before the read command. The end result is four bytes in the array, at indices 2 to 5. The bytes must be assembled into the 16-bit integer values: see **Listing 13**.
The received values need to be interpreted correctly. The full-scale range of input voltages to the A/D converter is 600 mV, and this range corresponds to the 2^{16} = 65536 possible output codes. The voltage (in millivolts) on pin 4 of the converter is thus related to the conversion result by the formula

$$V_{pin4} = 600\ mV \times (\text{conversion result}) / 65536$$

and the same goes for pin 3. To calculate the actual voltage at the input terminals the calculation must be modified to take account of the voltage divider and amplifier circuit.

Deactivate Chip Select
The final step is to deactivate Chip Select: see **Listing 14**. With that the process of acquiring the two samples is complete.

The software download includes a document with source code that shows how to use the temperature sensor in the board.

(130006)

Web Links

[1] www.microsoft.com/en-gb/download/details.aspx?id=40787
[2] Code Mercenaries IO-Warrior56 generic universal I/O controller for USB, V1.0.2, September 22 2011 for chip version V1.1.0.1: www.codemercs.com/uploads/tx_sbdownloader/IOWarrior_56_Datasheet_01.pdf
[3] www.elektor-magazine.com/130006

Dice Clock

With CD4000 ICs and an innovative display

By **Joost Waegebaert**
(Belgium)

Digital clocks are a traditional and rewarding subject for a small electronics project that can be realized in a spare moment. The trigger for this specific design was a discussion in several electronics fora about the generation of an accurate 1-Hz signal. This was the provocation for the author to design an accurate clock himself, using only standard ICs (and therefore without microcontroller).

In the various discussions that the author was following, the solution almost invariably evolved into the direction of a microcontroller with all sorts of software tricks to obtain a sufficiently accurate timing. Although the reflex of a software solution to an electronics problem is these days usually justified, it is not necessarily the best method for a clock. This article describes a digital clock, following the classic design, but with a novel display to catch the eye. In addition, this clock is extremely frugal and, with the aid of a small battery, continues to keep time when there is a power failure.

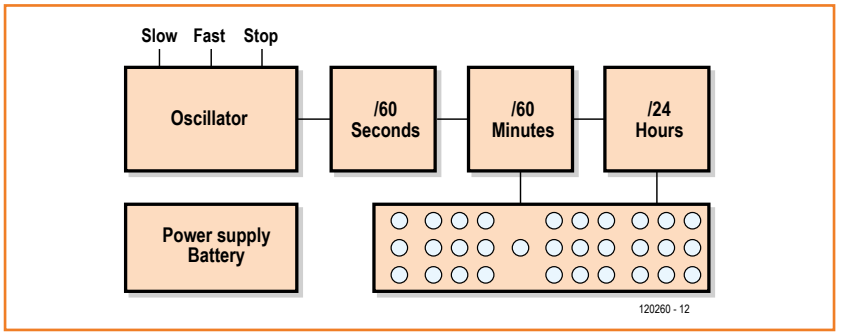


Figure 1.
The block diagram of the clock with a dice-like display.

Block diagram

A digital clock is not much more than a counter which is, every second, incremented by 1 (**Figure 1**). The counter is split into several parts, with the individual sections dealing with the seconds, the minutes and the hours respectively. The counters each drive their part of the display.

Oscillator

At the heart of each clock is, of course, a stable oscillator. Any electronics engineer will automatically arrive at a crystal-driven implementation. The frequency is in principle an arbitrary choice—a suitable divider will then ultimately deliver 1 pulse per second.

A sensible choice of this frequency will, however, have to take into account the effect on the current consumption. For crystal oscillators, obtaining a low power consumption is easier at lower frequencies.

A second requirement is that it should be easy to divide the frequency down to 1 Hz. A power of 2 is then always very convenient. A divider is then no more complicated than a series chain of a sufficient number of flip-flops. Since time imme-

morial, 32.768-kHz crystals have been available for this purpose. 32,768 equals 2^{15} and it therefore suffices to chain 15 flip-flops one after the other to generate one pulse per second.

A third function that this oscillator has to fulfill is the simple generation of a higher pulse speed for the purpose of setting the clock.

At low frequencies, discrete digital logic has a frugality champion in the form of the CD4000 family. Decades old, but even by modern standards still extremely low power and furthermore has no special requirements when it comes to the power supply voltage. Several single-chip oscillator/divider combinations are available and the choice here fell for the CD4536 (see the schematic in **Figure 2**).

In the CD4538 chip, the oscillator is followed by 24 flip-flops, more than enough for the application as a 2^{15} divider. The binary value at inputs A through D determines which flip-flop output from the divider chain is connected to the output. In normal operation, output 15 (1110) is selected by R2 through R5 for a divider ratio of 32,768 and 1 Hz at the output.

For setting the time of the clock it is easy to choose a faster output by changing the binary code on the selection inputs. Pushbutton S2 selects output 7 (256 Hz) for the slow setting speed. Pushbutton S1 selects output 3 (4096 Hz) for the high setting speed. The final pushbutton (S3) stops the clock for as long as it is held down to allow the clock to be accurately synchronized to, for example, the hour beep of a radio station. When S3 is released the seconds counter will start from 0, so that the next minute will begin exactly 60 seconds after the button is released. The exact frequency of the seconds signal is adjusted with C3. This should be done only after

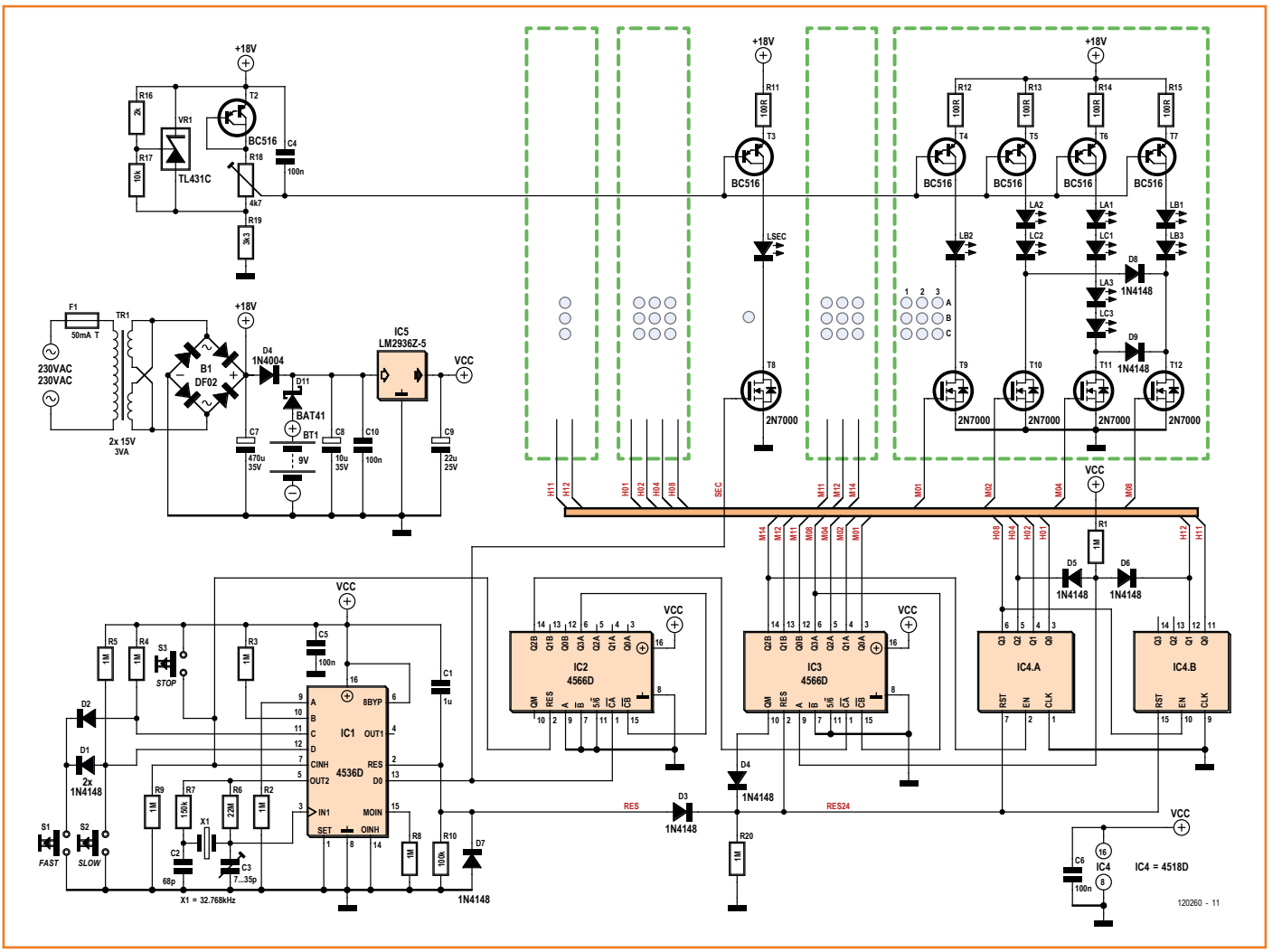


Figure 2.
The complete schematic for the clock. The complete LED driver section is only drawn for the final dice.

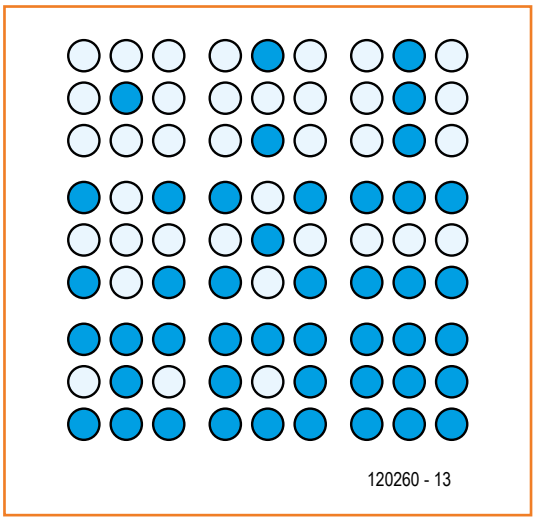


Figure 3.
This is how the numbers 0 through 9 are represented on each dice.

the clock has been assembled in its permanent enclosure. Turn the clock on for half an hour, so that it will reach thermal equilibrium before starting the adjustment. This is a simple job with an accurate frequency counter: The signal on pin 5 of IC1 is adjusted to 32,768.00 Hz. Without a frequency counter you need to check the time daily at a fixed time and adjust C3 accordingly. After a week of adjustments the accuracy will be very good.

Counter

A handy chip for use in a digital clock is the CD4566. It contains two BCD counters, the first of which counts from 0 to 9 and the second counts to 5 or 6. Here we obviously choose to divide by 6 (pin 11 of IC2 and IC3) for counters from 0 to 59. That finishes the minutes and seconds counters. Nothings else is necessary.

The 24-counter for the hours is built from two BCD counters in IC4. These are reset at the counter value of 24 by generating a short pulse via D5 and D6. A nice monostable multivibrator is available in the CD4566, which will generate a clean reset signal (on pin 10 of IC3) from the short pulse at the node with R1.

For a 12-hour clock it is enough to connect diodes D5 and D6 to outputs Q1 of IC4A and Q2 of IC4B respectively. The reset pulse will then occur at a counter value of 12.

Display

The display for this clock comprises a number of LEDs, which are arranged in the shape of the dots of a number of dice. This has some likeness to the so-called binary clocks (which show a binary code directly), but the display pattern used here is definitely much easier to understand by less technically inclined people.

A number is displayed on a 'dice' by illuminating the corresponding number of LEDs (0 through 9). This uses a variation of the patterns of dots as used on a dice (see **Figure 3**).

The LEDs are controlled via current sources. These follow the conventional design with the combination of a transistor (for example T3) and an emitter resistor (R11). The voltage at the base of T3 is constant with respect to the +18 V, which is the other side of R11. Assume that U_{BE} is constant, therefore there is also a constant voltage across R11. From this follows that there is a constant current in the emitter. The collector current which will flow through the LED (LSEC) is therefore this same value minus the base current. T3 is a Darlington type, which means that the base current is extremely small and its effect can be ignored.

The base voltage can be set with potentiometer R18 and with this also the brightness of the LEDs. The additional U_{BE}-junction (T2) in series with potentiometer R18 provides temperature compensation of the current sources. A change in the U_{BE}-voltage of, for example, T3 will be approximately compensated by a similar change of the reference voltage at the base.

VR, an integrated voltage reference type TL431, is set to 3 V with R16 and R17. V_{BE} of the Darlingtons is about 1.4 V. R18 therefore adjusts the voltage across the current source resistors (R11, etc.) between 0 and 1.6V, corresponding to LED currents up to 16 mA.

Because of the current control, all the LEDs in the display segments have the same brightness, independent of the number of LEDs in series.

The on/off switching of a segment is done with a MOSFET (for example T8). In addition to its switching behavior, this MOSFET is also essential for isolating the clock from the display driver during a power outage. It is very important that the powered-down driver stage does not draw any current from the battery-powered clock section. The gates in the MOSFETs are ideal for this since in the steady-state there will flow only a very small leakage current (10 nA). A bipolar transistor would work just as well as a switch in this application, but will always require a base current, even in standby mode and is therefore unsuitable. The exact type of the MOSFETs is not critical: 2N7000, BSS100 and BSS123 are

all suitable.
The decoding of the binary value to the correct display pattern is very simple by the clever combination of the LEDs in the series connections. Only with bit 3 of the input code (M08) are 2 additional diodes required.
Because of the current control it is also possible to give each digit its own color LEDs. The current through the LEDs remains constant after all, independent of the different voltage drops across the LEDs of different colors. The difference in efficiency between the different colors can be compensated for by adjusting the current for each color. A smaller or larger current for a particular color can be obtained by changing the emitter resistors for that color.
The circuit from block M01 through M08 is repeated 3 more times for the remaining digits of the clock. H11 through H12 requires only 2 current sources, since the highest number that needs to be display is '2'.
A flashing seconds LED is mounted between the hours and minutes digits. This is driven directly from the 1 -Hz clock pulse.

Power supply

The power supply for the clock provides an unregulated voltage for the display section and an uninterruptable supply for the oscillator and the counters.
The required transformer voltage depends on the voltage drop across the LEDs (from 1.7 V for red LEDs to 4 V for some white LEDs). The display segments with 4 LEDs in series determine the minimum voltage required. For the voltage drop across the current source (emitter resistor + V_{CEsat} of the Darlingtons) a value of about 3 V needs to be taken into account. Therefore, four white LEDs require 4x4+3 = 19 V. Add to that the 1.4 V for the diodes in the rectifier bridge B1, which results in a minimum transformer voltage of approximately (19+1.4)/√2 ≈ 14.5 V.
The power supply voltage to the current sources does not need to be regulated since the current through the LEDs is already accurately controlled. The LM2936 (IC5) looks like an ordinary linear regulator, but is quite an exceptional chip because of its extremely low quiescent current.
In the prototype, the current consumption of the counters and oscillator together amounted to 28 µA. The LM2936 delivers this current and manages to add barely 10 µA to the energy budget of the clock. In the absence of the line voltage the complete clock therefore consumes only 38 µA at 9 V!
In addition, IC5 has a drop-out voltage of less than 0.1 V at these low currents. The backup battery will therefore be drained to the very last drop. A typical 9-V alkaline battery is specified with a capacity of 550 mAh when discharging to 5.4 V. For this clock this means 0.55/0.000038 ≈ 14470 hours or more than a year and a half of standby use. A rechargeable battery is also a possibility by adding a trickle-charge resistor across D11, but provides hardly any advantage in this circuit.

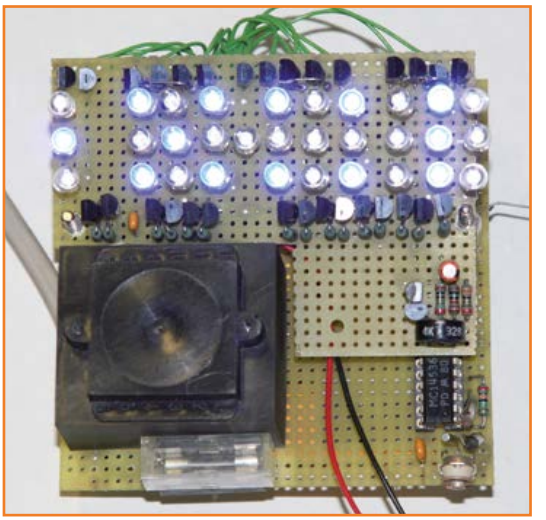


Figure 4.
The author built the entire circuit on two pieces of prototyping board.

Finally

This little clock illustrates that using classic digital building blocks it is possible to build elegant circuits in a simple way. Particularly the extremely low current consumption will be a surprise to many.
Various enhancements can be conceived. For example, by making the reference voltage to the current sources dependent on ambient light, the LEDs can be dimmed when it is dark.
A purely battery powered device is also possible if the transformer is replaced by a step-up voltage regulator which powers the current sources from the battery via a 'display on' pushbutton.
It is also possible to add two mode LED dice to display the seconds information from IC2.
The CMOS 4000-series is still manufactured by various companies (TI, NXP, OnSemi, ...). The type names often have different prefixes. Chips with type codes like CD4566, HEF4566, HCF4566 and MC14566 are all largely compatible.

(120260-I)

Opto-isolated PTT/CW interface

Serial communication: avoid the pitfalls

By
Christophe Bourrier,
F4EZC (France) QRV@
wanadoo.fr

My opto-isolated interface allows you to transmit PTT (Push To Talk) or CW (continuous wave or Morse) command signals from a computer to a transmitter, while keeping the computer circuits isolated from the radio circuits.

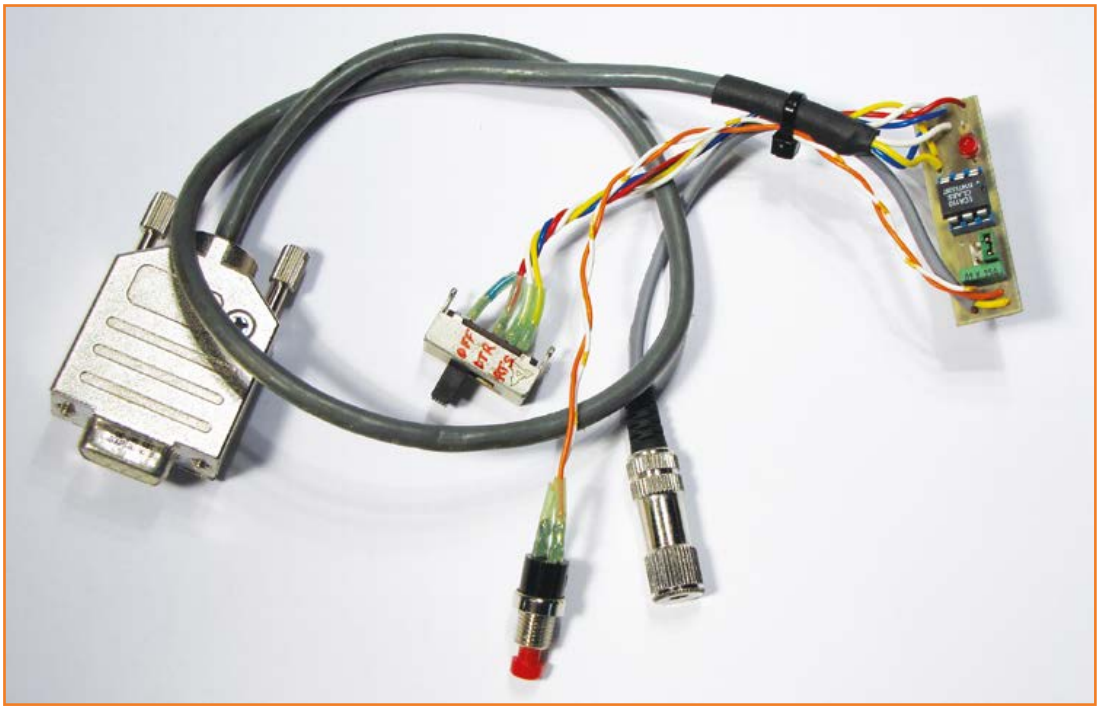
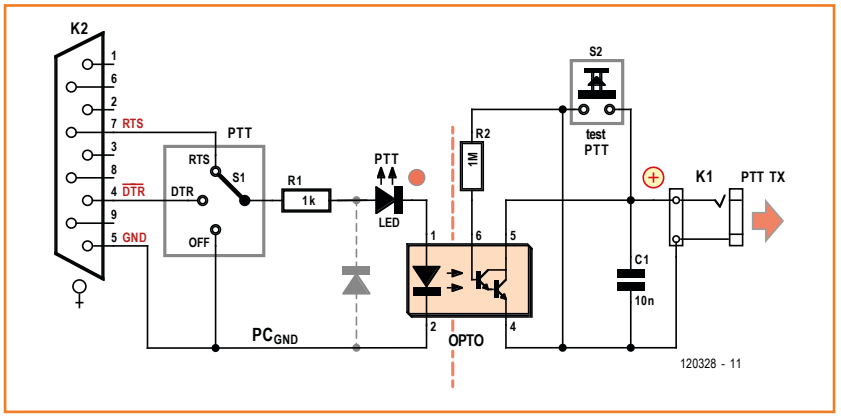


Figure 1.
An optocoupler allows us to ensure electrical separation between a PC and the radio it is controlling.



The electrical isolation between radio and computer makes it possible to avoid interference from the computer being conducted via the serial interface to potentially cause interference in the receiver. It likewise avoids any RF from the transmitter getting fed back to irreparably destroy the computer… These sorts of dangers are mainly a threat when there is a breakdown, a fault—or sometimes a bit of finger-trouble: an incorrect setting, an antenna problem, a coax connector, defective ground or electrical wiring, can all cause serious trouble.
If, for example, the transmitter goes into trans-

this ham radio interface doubles for other applications driven by RTS or DTR

mit, even for only a few seconds, while it is connected to the computer (itself often connected to the supply line), leakage or RF feedback currents would then pass down the computer and LF connexion cables, electrical circuits, and… bang! The higher the transmitter power, the great the risk is too. So protecting the computer from that sort of risk has to be the start of any protection. In the absence of appropriate precautions, despite its robustness, at best the computer's serial port will be destroyed... and at worst, the motherboard! Not only is this board simple, it also offers characteristics that I hope will convince you how useful it is: choice of RS232 command signals, LED indicator, and use of optoMOS possible. An OFF position lets you disable the circuit to avoid any unwanted commands.

The basic circuit shown in **Figure 1** is conventional, apart from the fact that it lets you choose the RTS or DTR command signals coming from the computer's RS232 socket. The OFF position prevents any commands being sent to the radio transmitter; this is handy when the transmitter is not being used, but above all while the software is initializing—whether Windows or the peripherals. Resistor R1, the red LED and the optocoupler are fitted in series in the input circuit via selector S1 which lets you choose the signals to be used. The LED lights when a command (PTT or CW) is sent by the computer.
Because of its low cost, the phototransistor optocoupler in Figure 1 is perfectly suitable in the majority of cases, as long as we can put up with the fact that the emitter/collector junction is polarized and that it causes a voltage drop of sometimes over 1 volt.
Resistor R2 is optional, but by biasing the base of the phototransistor, it avoids certain oscillation phenomena. Capacitor C1 makes it possible to damp certain possible output oscillations.
The button lets you directly test the radio transmitter command.

Single-pole optocoupler solid-state relays known as optoMOS™ [1] (a process patented by Clare, whose trademark it is) used in the circuit in **Figures 2a and 2b** don't suffer from the drawback of the emitter/collector voltage drop in a phototransistor. They switch less fast, but still more than enough for radio applications. In either case, R2 will be left out. Using optoMOS in universal AC configuration—and hence non-polarized—(**Figure 2a**) the voltages switched will be able to be negative or positive. Using the DC configuration (**Figure 2b**), pin 4 is the positive pole and the switching resistance is lower than in the AC configuration. Depending on the exact type, it may be under 10 Ω. Watch out—in this configuration, the polarity is reversed with respect to that in Figure 1 with a phototransistor.

I am suggesting a small (42.5 mm × 11 mm; 1.67 x 0.43 inch) single-sided PCB design (**Figure 3**),

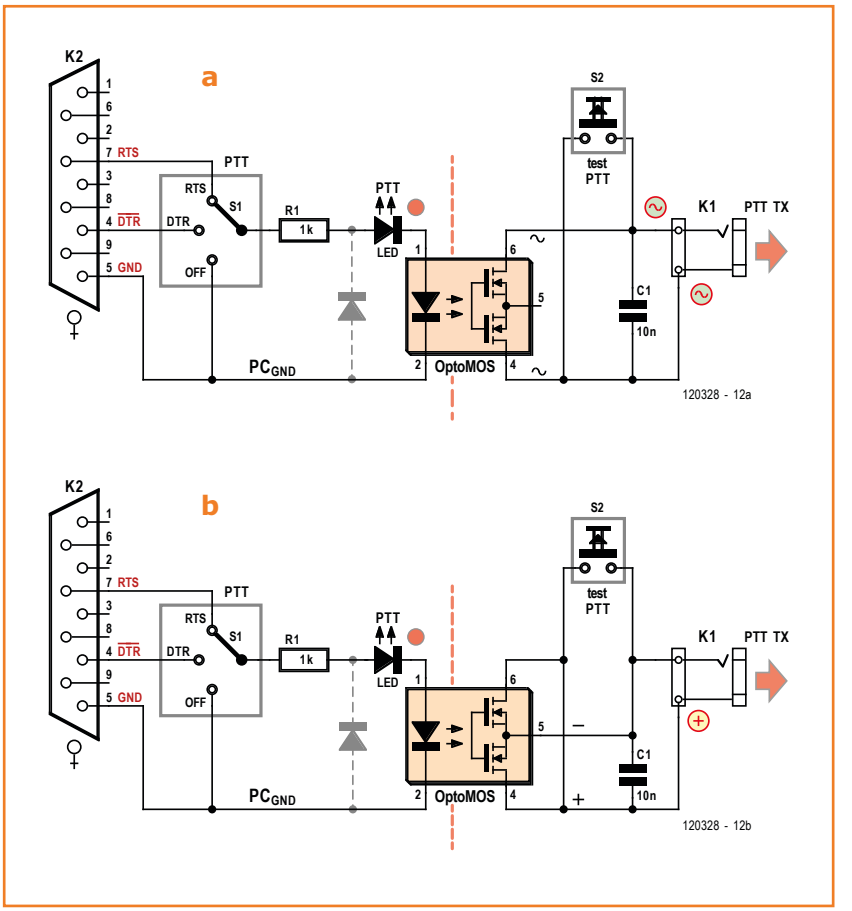


Figure 2.
Using an optoMOS, it's easy to change the configuration (DC/AC).

Component List

Resistors
R1 = 1kΩ 0.25W
R2 = 1MΩ 0.25W (optional)

Capacitors
C1 = 10nF

Semiconductors
LED = LED, red
Optoelectronics (see text)
version with phototransistor: 4N25, 4N26, 4N27, 4N32, 4N33, 4N37, CNY17-3, SL5500, TIL111
version with optoMOS : LCA110, OMA160 (CLARE)
6-pin DIL socket

Miscellaneous
S2 = pushbutton (optional)
S1 = 3-position slide switch
K1 = mono 3.5mm jack socket (preferably isolated)
K2 = 9-way sub-D socket

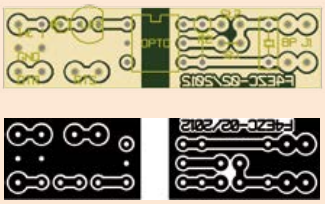


Figure 3.
PCB design and component layout.

distinctly separated into two halves, along with its layout diagram. Using a conventional (phototransistor) optocoupler and the optional resistor R2, the jumper JP1 is fitted but JP2 stays open. Using an optoMOS in AC configuration (**Figure 2a** circuit), JP1 is left out, JP2 fitted, and R2 absent. Using an optoMOS in DC configuration (**Figure 2b** schematic), the jumper JP1 is fitted and JP2 is open, while R2 is replaced by a short wire strap. Watch out for the polarity of the mini-jack socket K1!! If you fit it into a metal case, preferable use an **insulated-chassis** type: you will have the benefit of all the polarity options possible without having any voltage on the case. In order to be able to change the optocoupler easily, I recommend fitting it into a DIL IC-socket.

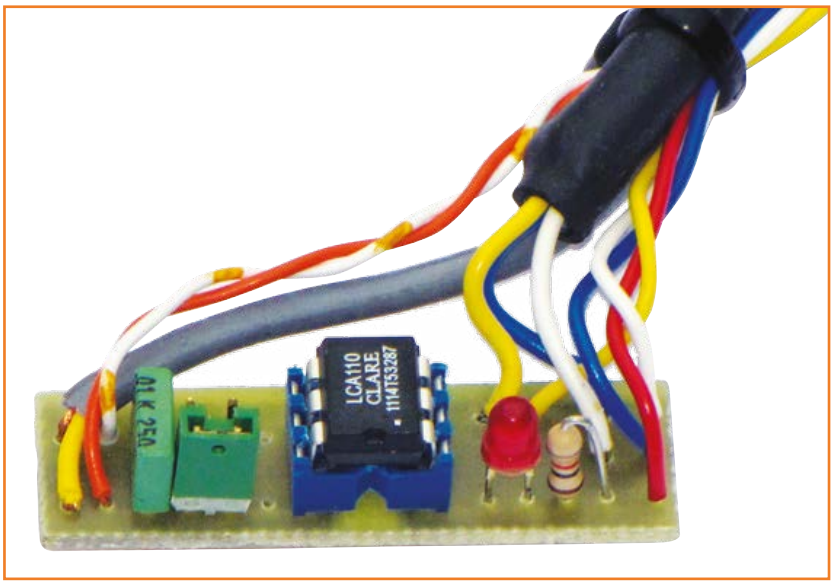


Figure 4.
The prototype of my interface using a Clare optoMOS devivce.

All the types of optocouplers mentioned in the components list have been tested.

Use

For the first tests, don't connect up the radio transmitter. Connect the circuit's RS232 socket to one of the computer's serial ports or via a USB/serial converter. Try your radio transmission software (PTT or CW). Select the option RTS or DTR in the software (when this is possible), as well as on the selector switch S1. Check that the LED does indeed follow the transmit commands. If you are using CW mode, it should flicker at the same rate as the Morse being handled. Connector K1 must be connected to the transmitter's CW socket.

In the other digital modes, it's the PTT command that is used. This lets you switch the radio transmitter to transmit. In this case, K1 must be connected to the transmitter's PTT input.
Button S2 makes it possible to briefly check the transmitter action (without lighting the LED), to ensure that the wiring is correct and the transmitter is configured correctly.

And lastly, when the circuit is not being used, do always remember to put the switch to the OFF position.

(120328)

Web Link

[1] LCA110 datasheet: http://goo.gl/Dlgbqo

Super Mobility Week is North America's largest forum for the mobile innovations to power your connected life —
Business. Home. Health. Money. Auto. Retail. Media. M2M. Networks.

Only CTIA can convene the most influential mobile marketplace, bringing together, for the first time ever, the leading authorities on the connected life.

Strive for Success. Live for **Super.**

Super Mobility Week
powered by CTIA

September 9, 10 & 11, 2014 Sands Expo & Convention Center Las Vegas #SuperMobility

Analog Discovery from Digilent

All the bells & whistles in one basket

By
Dr. Thomas Scherer
(Germany)

You may be wondering what's so interesting about a USB oscilloscope module with several additional functions and an analog bandwidth of 5 MHz. However, this multipurpose device has a lot to offer. Its additional functions make it an all-round 'lab in a box' that is ideal for educational purposes or for learning about electronics on your own (the name says it all).

Of course, 'real' oscilloscopes with screens and knobs are not all that expensive any more, and they have significantly higher analog bandwidths of 25 to 40 MHz. And even the low-cost models also have extra functions. So what's the advantage of this little plastic module from Digilent? What can it do better than a low-cost stand-alone oscilloscope? And 5 MHz bandwidth—isn't that something for the kids?

The short answer is that if you're only looking for a low-cost USB oscilloscope, Digilent's offering may not be your best choice. But if you're a student, a committed analog enthusiast or a hobbyist with a passion for test equipment, you should keep on reading—you may find that this little device with its bundle of leads can do a lot more than you might expect.

What's in the package?

This multipurpose module, which you can use to test, measure and control just about anything to do with electronics in the audio to low-RF realm, is available for US $239 from the website of Digilent, a manufacturer of all sorts of electronics accessories and modules. Developed in collaboration with the well-known semiconductor manufacturer Analog Devices, this little device boasts a wealth of functions (see **Technical Data**), which makes it an excellent hardware platform for training electronics engineering students in entry-level courses. Accordingly, this target group gets a special discount. As a bonus feature, for US $70 you can buy an accessory parts kit. It contains a number of commonly used opamps and CMOS ICs in DIL packages, as well as a bunch of modern SMD ICs from Analog Devices on small PCBs which can be plugged directly into the solderless breadboard included in the kit. If you buy the package, you get a US $20 discount on the price of the parts kit.

Technical Data

Oscilloscope

- 2 channels with differential inputs
- 5 MHz analog bandwidth
- 100 MS/s ADC sampling rate
- 14 bit ADC resolution
- 1 MΩ / 20 pF input impedance
- Vertical: 0.5 mV – 5 V per division, max. ±25 V
- 16 KS buffer
- Cross-triggering with other functions

Function generator

- 2 channels with single-ended outputs
- 5 MHz analog bandwidth
- 100 MS/s DAC sampling rate
- 14 bit DAC resolution
- Sine, triangle, square wave, etc.
- User-defined waveforms
- Sweeps, envelopes, AM and FM modulation

Logic analyzer

- 16 digital channels
- 100 MS/s sampling rate
- 3.3 V logic level, LVCMOS compatible
- 16 K transitions buffer capacity
- Trigger: pin change, bus pattern, etc.
- Interpreter for SPI, I²C, UART, parallel bus

Other functions

- Voltmeter: DC, AC with True RMS
- Adjustable output voltage
- ±5 V / 50 mA output voltage
- Spectrum analyzer with many modes
- Digital I/O with all 16 pins individually controllable
- Digital pattern generator – up to 100 MS/s
- Network analyzer – 1 Hz to 10 MHz
- Data transfer and power via USB 2.0
- Waveform generator output on 3.5 mm audio jack
- Signal data can be exported
- MATLAB support

Figure 1 shows what you get when you order the Analog Discovery & Parts Kit package. When you open the green box on the left, you see a USB cable and a fairly small black plastic module measuring 82 x 63 x 19 mm (3.22 x 2.48 x 0.75 inches) with a bunch of colored leads attached (**Figure 2**). A few extended six-pin headers are thrown in for free. All of the parts contained in the transparent, compartmented parts kit box are listed on the lid. For anyone who does not have a well-stocked components drawer, the kit definitely has several interesting items in store (**Figure 3**). Along with the usual opamps, transistors, resistors, trimpots, ceramic capacitors and electrolytic capacitors, there are some unusual devices such as a voltage reference IC, a voltage to frequency converter, comparators, and a variety of sensors for measuring current, magnetic fields, acceleration, sound, temperature, vibration, brightness and infrared radiation. Digilent has also thought about the details: along with a handy solderless breadboard, there are a bunch of jumper leads and even a small screwdriver. What's that for? Well, it's perfect for levering components free from the breadboard without damaging them.

What it can do

Everything comes without a single sheet of paper—obviously no trees were sacrificed for the documentation—so the first thing you have to do is visit the website listed on the back of the package [1]. That turns out to be Digilent's order page, but you can also download the necessary software there, which goes by the name "WaveForms". As we all know, nowadays hardware is useless without software. WaveForms is available for Windows PCs, but there is also a version that runs under Linux on i386 and AMD64 processors.

Figure 1.
What you get when you order the Analog Discovery & Parts Kit package.

Figure 2.
The Analog Discovery package contains a USB cable and a small plastic module with a bunch of leads.

Installation is a simple double-click operation. The software looks for matching hardware and then links to it. However, on my PC running Windows 7 this takes a little while; approximately 20 seconds each time after WaveForms is launched. When it's up and running you see the selection dialog shown in **Figure 4**, where you can run the basic functions as well as the other functions listed under "Technical Data". The voltage on the USB bus and the current consumption are also indicated.

Figure 5 shows the window of what is probably the most important instrument: the oscilloscope. This is a full-fledged two-channel storage scope with all sorts of nice extra features. In the figure the oscilloscope is displaying the signal from the function generator, which is producing a 200-kHz sine wave with frequency and amplitude modulation. The spectrum of this signal is also shown in the lower part of the window.

Figure 6 shows the window of the arbitrary function generator, which is the source of the signal measured by the oscilloscope. As you can see, the instruments are all modular and can be operated independently in separate windows, just like real stand-alone instruments. You can also see from the setting options that many features are available. The features are certainly as good as what you get with a low-cost function generator, which can easily lie in the same price range as the multipurpose Digilent device even if is made in a low-wage country. The function generator is called "arbitrary" because it can output user-defined waveforms in addition to the usual predefined waveforms. The only tool you need for this is MS Excel.

Figure 7 shows the window of the spectrum analyzer. There you can see the typical spikes of a 100-kHz triangle waveform (first, third, ninth harmonics and so on), whose amplitudes drop with increasing frequency. The setting options here are fairly simple. A somewhat irritating feature is that the frequency range settings can only be selected from predefined values, with no option for user-defined settings. This is obviously not a high-end analyzer, but its capabilities are certainly adequate for relatively simple analog tasks. In any case, it's fine for checking audio filters and the like.

Now let's look at the other functions mentioned in the technical data. Checking out circuits with relative simple microcontrollers, for example, is no problem. Even if you can't measure signals at high clock frequencies, a 16-channel logic analyzer is certainly useful. The counterpart of the logic analyzer is a pattern generator, which can output parallel digital signals at a respectable 100 MS/s and can utilize the full buffer capacity for the generated sequences. An especially attractive feature is that the digital I/O pins can be individually configured as either inputs or outputs, allowing them to be used to read or generate digital logical levels. Last but not least, the two analog outputs are connected to a 3.5-mm audio jack in addition to the appropriate leads on the front of the module. This means that you only need a standard audio cable to test a sound system.

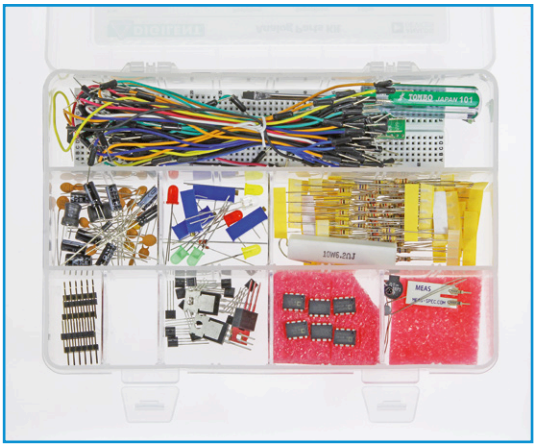


Figure 3.
In addition to standard components, the parts kit includes some interesting sensors, a solderless breadboard and other small parts.

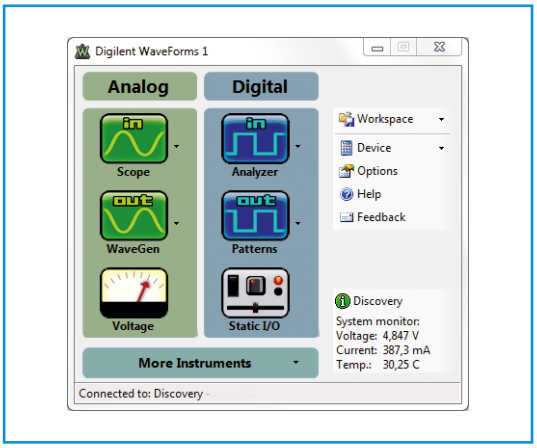


Figure 4.
The selection dialog of the WaveForms program gives you access to the main functions.

Selling points

By itself, the multipurpose module together with the parts kit would not be of much use in a school lab or on a hobby bench for self-study. For that reason, five "getting started" videos are available on the website [1], along with a complete university course in basic electronics tailored to

the device. However, the course is not recommended for would-be electronics specialists who don't have a mind for math.

That leaves the question of why this little multipurpose instrument could be a suitable bit of equipment for a small electronics lab in certain areas, even for non-students. The answer is that the Analog Discovery module has a lot of strong advantages compared to low-cost stand-alone oscilloscopes. The first, of course, is the small footprint—you already have a PC anyhow, and the device is really small and multipurpose. In the sub-professional realm it can replace a whole set of individual instruments that would collectively not only take up more space but also cost a good deal more.

Another important consideration is that a PC has a much larger screen. Even if a low-cost stand-along oscilloscope boasts a 7-inch screen with a "massive" resolution of 800 by 600 pixels, it doesn't come anywhere close to this integrated

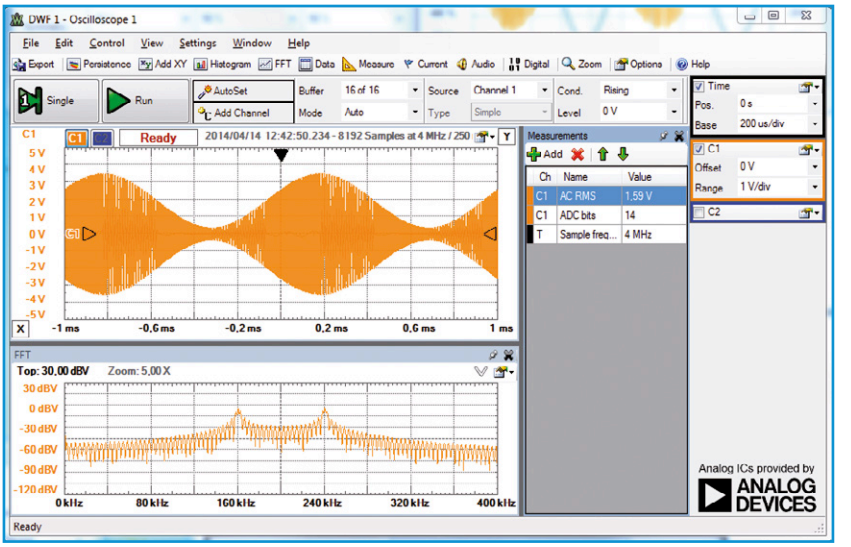
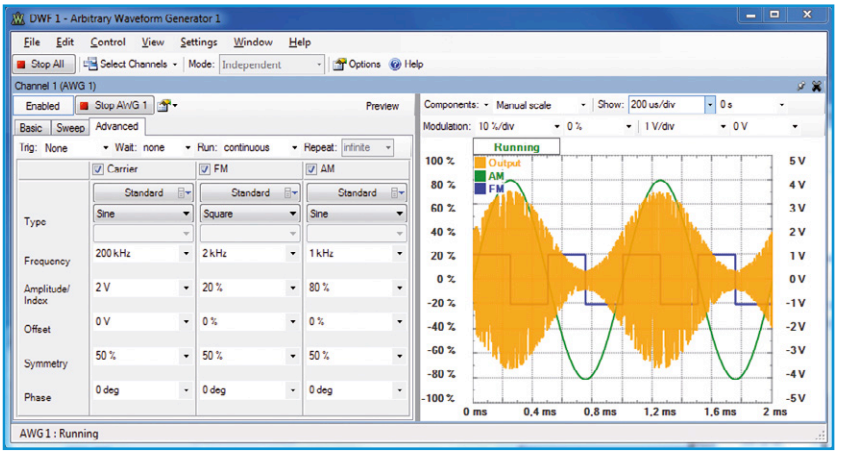


Figure 5. The oscilloscope window of the WaveForms program. A modulated sine wave signal generated by the Analog Discovery module and its spectrum are shown here.



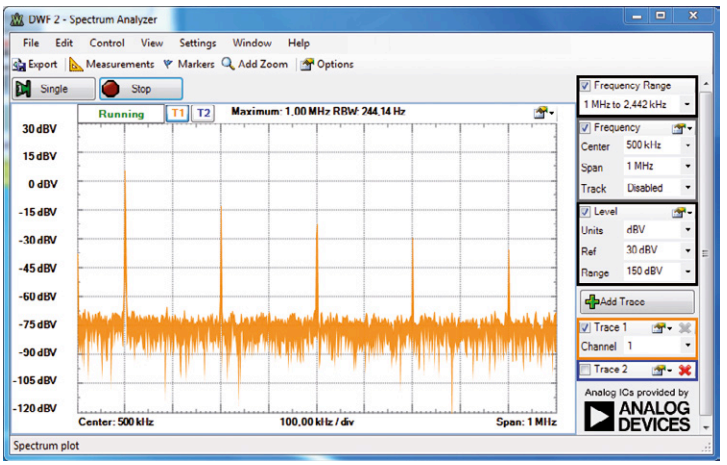


Figure 7. The spectrum analyzer only offers a limited number of functions. The spectrum of a 100-kHz triangular-wave signal is shown here.

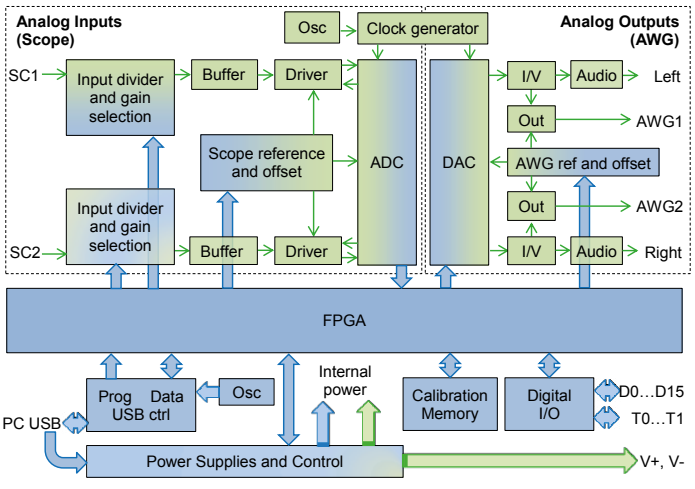


Figure 8. Internal block diagram of the Analog Discovery module. All of the circuit details are described in a PDF document.

Figure 6. The arbitrary waveform generator produces the 200-kHz sine wave signal measured in Figure 5, which is both frequency modulated and amplitude modulated.

solution. That's because low-cost oscilloscopes usually have ADCs with a resolution of just eight bits. As a result, the vertical amplitude of a signal is limited to 256 pixels unless pixel replication is used, which does nothing to improve the waveform presentation. With a resolution of 14 bits, the Analog Discovery is clearly playing in a different league. What's more, the window of the WaveForms program can be expanded or contracted as desired. Finally, the software for low-cost stand-alone scopes is often rather rudimentary, even when a USB port or Ethernet port is available, and not especially user-friendly. Here as well, WaveForms comes out on top. Another thing: even if a stand-along scope has more bandwidth, with the Analog Discovery module you get an arbitrary function generator and much more as a bonus. Another plus point is that you can download a PDF document with a detailed description of the inner workings of the Analog Discovery module [1]. Where else can you find something like that?

Of course, there are two sides to every coin. The most serious shortcoming is doubtless that the Analog Discovery module is not galvanically isolated. This is not important for analog signals with potential differences of a few volts, since differential inputs are available. However, the digital I/Os are referenced to a ground level that is directly connected to the PC ground via the USB cable. This means you need to know what you're doing when you work with digital signals.

(120585-I)

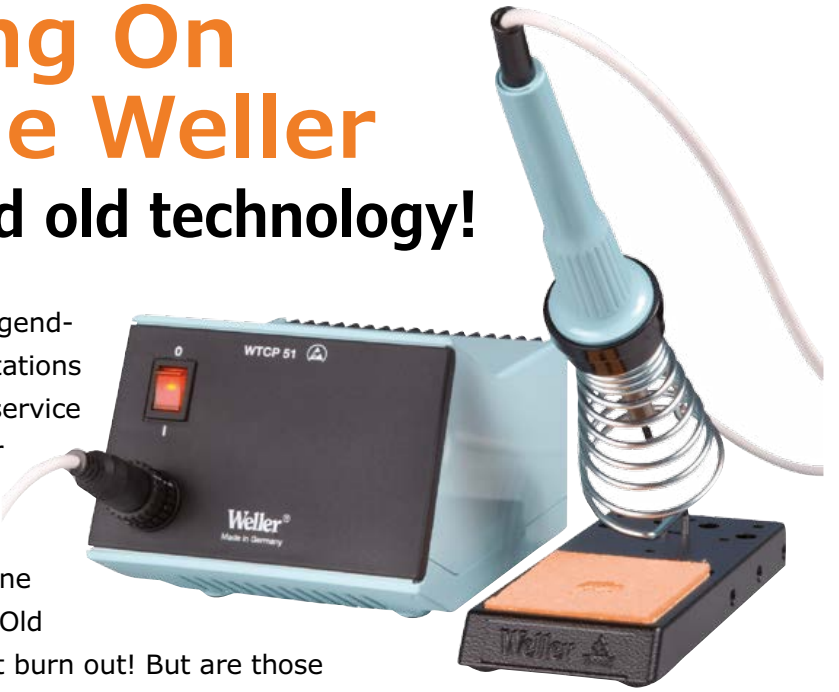
This interesting kit is now available from the Elektor Store, www.elektor.com.

Web Link

[1] More information, user guides and software: www.digilentinc.com/AnalogDiscovery

Soldering On With The Weller

Don't discard old technology!



By **Ingo Burret**
(Germany)

Weller's (Cooper Tools') legendary Magnastat soldering stations have performed valuable service in labs and elsewhere over many years. Gradually, however, they get long in the tooth and develop one minor ailment or another. Old flames never die, they just burn out! But are those grounds for pensioning off faithful old servants?

Nowadays even the most basic soldering station contains elaborate control electronics. But in Weller's revered turquoise-colored WTCP 51-LT, WTCP-S, WTCP 50, WTCP 51 [1] (and previous) soldering stations, the matter of temperature regulation was managed mechanically. In a number of countries Weller products are sold under the brand name Cooper Tools.

Figure 1 shows the internal construction of the soldering iron. Reading upwards, the tip (1) is linked mechanically to a metal cap (2), which forms the temperature sensor in this system. The cap is ferromagnetic and in its cold state attracts a permanent magnet (3). A push rod transfers the movement arising from this action to a contact bridge (4), which enables the heater current supply (5) to switch in.

The numeral on the ferromagnetic cap is a code for the regulated temperature (from 260°C to 480°C). As the iron heats up and the temperature rises towards the required value, the Magnastat loses its ferromagnetic characteristics abruptly, thanks to the Curie Effect [2]. It can then no longer retain the permanent magnet. The magnet drops off and operates the switch, cutting off the power supply to the heating element. When the tip cools, the temperature sensor attracts the permanent magnet once more and the whole sequence begins afresh.

This snap action of the Magnastat is very consistent and reliable, with no wear and tear caused by ageing or fatigue of the materials. In addition, the Curie Values of the soldering tips display very little variance. A further advantage of this arrangement is that changing the tip switches off the iron, meaning that the heating element does not burn out because the tip is missing.

Hot contacts

All the same, it frequently occurs that the magnetic switch seizes up. Then you have to knock or shake the iron in the hope of dislodging the switch. The soldering station itself is still OK otherwise and you probably have numerous soldering tips of different shapes and temperature values in the drawer. Over the years you must surely have become fond of the device, meaning you should not throw it out on account of such a minor flaw. Of course you can replace the magnetic switch (the replacement part comes as a single item with the push road and permanent magnet), but it's expensive. And if you then establish that the heater winding is burnt out as well, it's doubly annoying.

So how can you unburden the magnetic switch without undue effort? As the two-conductor wiring delivers 24 VAC to the heater (a third wire

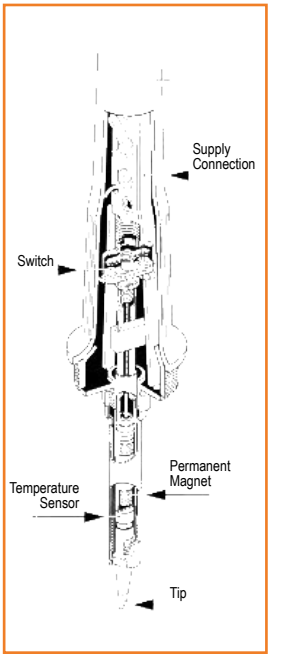


Figure 1.
Cutaway drawing of the Magnastat soldering iron (Weller illustration).

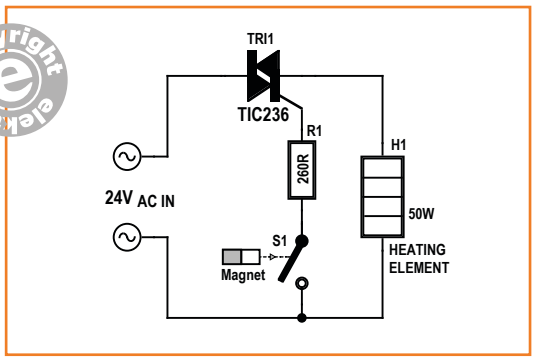


Figure 2.
A triac and a resistor ensure only meager current flow though the magnetic switch.

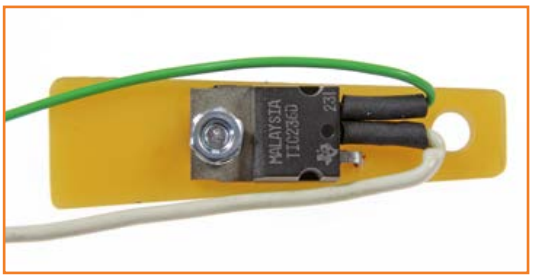


Figure 3.
The triac and resistor can easily be mounted inside the soldering iron handle on a small piece of PCB.

is provided for potential equalization) a mosfet or bipolar transistor will not offer a simple solution. Relays would be too large to fit inside the soldering iron. The solution is a triac, which is conveniently small and wired almost like a transistor (**Figure 2**). The "electronics" comprise just a resistor for reducing the trigger current for the triac and hence the current through the switch to a tolerable value.
The triac fits nicely onto a wedge-shaped scrap of PCB material fixed inside the handle of the soldering iron (**Figure 3**). So the magnetic switch soldiers on forever, even unto grim death...

(130008)

Web Links

[1] www.weller.de/products/product.php?pid=431 (set language to English)
[2] https://en.wikipedia.org/wiki/Curie_temperature

Animal Friendly Mousetrap

Non-lethal pest control

By **Manfred Haemmerle** (Austria)

Mice are really cute little creatures—except when they enter your home, where they can cause considerable damage and leave droppings that can jeopardize our health. This leaves us with no option: if there's a mouse in the house, we've got to catch it and take it outside. Dead or alive!

A couple of years ago we had a crisis: a mouse we had caught in a snap trap was still twitching. Somehow the truly lethal blow had not dispatched the creature to the afterlife. This left me no other option than to grab an axe and deal with the poor mouse. No, let's drop the subject

there. But afterwards I had to promise faithfully to my animal-lover wife that I would never, ever again murder mice using such beastly, blood-thirsty methods.

There had to be another solution and not one of those so-called live-catch traps, which seldom catch anything (people claim to have seen whole families of mice dancing round them, laughing their little heads off). Some of the most outlandish solutions devised by mankind are to be found in the Mousetrap Museum in my country [1]. As a seasoned electronics enthusiast I realized immediately that outsmarting the mouse demanded an ingenious combination of sophisticated hard and software. And that's something you definitely won't find at any Mousetrap Museum!

Figure 1.
This is how we fix the trapdoor to the cage.

Trap shut, mouse alive

At the supermarket I discovered some storage baskets made of expanded metal lattice. These come in various sizes and one of them (measuring 20x12x12 cm) would serve as my mousetrap, fastened to a wooden base with hinges and inverted with the base of the basket on top. The small hand-grip hole on one end would serve as the entry point (the other handle opening was blocked afterwards with the printed circuit board). A trap door was fixed over the way into the cage. You can use a small piece of thin wood or circuit board material, suspended on some stiff wire or rings and kept open by a pin passing through a metal loop (**Figure 1**). As soon as a mouse looking for food breaks the infrared (IR) beam, a magnet pulls the pin out of the loop, the trap-door falls and closes the escape route. No way out for the mouse!

Figure 2 shows the electronics of the mousetrap. Power comes from four (rechargeable) batteries. The voltage of NiMH batteries is just right for

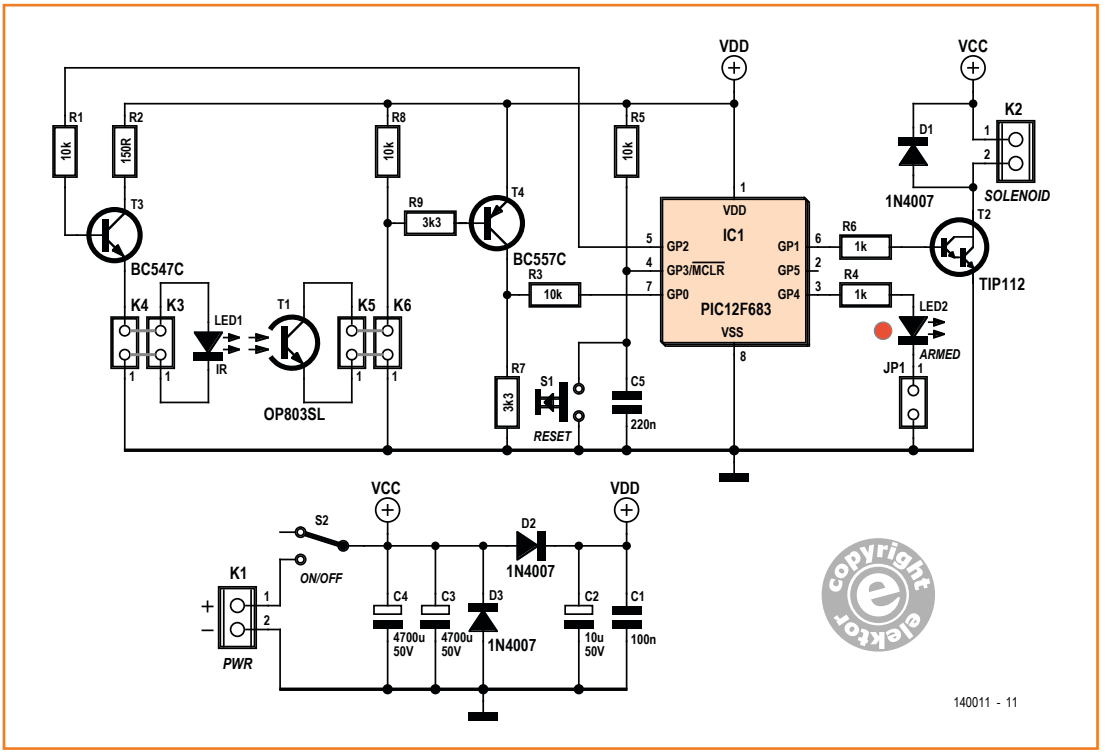


Figure 2.
The electronics for our mousetrap.

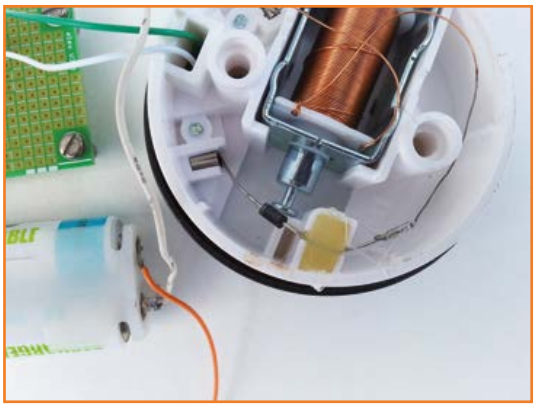


Figure 3.
A magnet from a toaster is used here. The flyback diode is connected direct to the inductance.

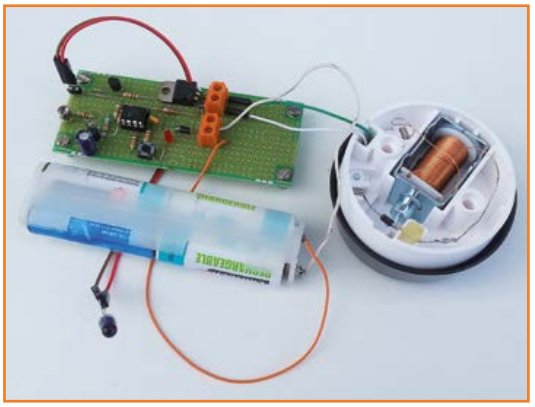


Figure 4.
The electronic section of the mousetrap fits on a scrap of perfboard, as made in Elektor Labs.

the PIC, whilst if you use alkaline batteries the polarity protection diode D2 drops the voltage to 5.4 V, which is still OK for the controller. D3 also provides some polarity protection, although if called into action, it would rapidly succumb to a sizzling death.

At top right you can see the connection to the magnet (K2), for which you could use (for example) one from an electric bell (its clapper serving as the release pin) or as seen in **Figure 3** one from an electr(on)ic toaster. Maybe you already have a small electromagnet or solenoid in your junk box. The current-hungry magnet (it takes more than 1 A for a short instant) is energized directly from U_{bat} via K1. As the batteries lose their charge over time, I provided two large 4700-µF capacitors C3 and C4 for energy storage in parallel with the batteries. This guarantees that the magnet will still attract even when the batteries are almost exhausted. The magnet is driven by Darlington transistor T2, which must be capable of switching at least 1.25 A. I used a TIP112 for this.

The infrared beam consists of an IR LED and an IR phototransistor T1 (OP803SL) fixed to the other side of the basket as receiver. Both the LED and T1 are linked to the microcontroller via transistors (T3 as amplifier, T4 as level inverter). A PIC12F683 is the centerpiece of the circuit. All functions are achieved using timing loops. Every tenth of a second output GP2 of the controller produces a pulse lasting just a few microseconds. The software checks whether this pulse reappears at GP0. If this does not occur (following multiple error checks), the magnet is released via GP1. If the jumper JP1 is fitted and the check LED (LED2) flashes slowly (every 2 seconds), the trap is primed. On the other hand, if it blinks rapidly and jumpily, this indicates that too much ambient light is falling on the phototransistor. These control signals would perturb a mouse of even average intelligence (not to mention the additional current consumption). Accordingly this function is used only at switch-on for checking, after which JP1 should be pulled.

Normal current consumption lies generally in the region of just 660 µA, so a set of fresh batteries should keep the circuit active for plenty of days. If the mouse enters the trap, however, the PIC12F683 enters Sleep Mode and the circuit then consumes only 40 µA. This saves the batteries to the extent that turning off the electronics is hardly necessary in fact. But the general idea is not to let the mouse starve to death and according to the laws on animal protection (in Germany at least), live-catch traps need to be checked daily without fail. To reactivate the trap all you need do is to operate the reset button. If the trap is not required for a while, you switch off the electronics altogether at the main switch S2 (or just remove the batteries).

The well documented C program is written using the CCS C Compiler, compiled in the MPLAB environment and loaded into the controller using a PICKit3. All this is unproblematic. The controller firmware can be downloaded free as source code and Hex code at [2], along with the printed circuit board layout.

The circuitry was constructed on perfboard in Elektor Labs (**Figure 4**), where a “printed” version (**Figure 5**) was also devised. Next the small sections containing the IR diode and the phototransistor were separated off and the populated

Figure 5. The layout of the mousetrap printed circuit board, with the two cut-outs for the photoelectric beam.

main PCB attached to the rear side of the cage, so as to block off the second handle opening. The phototransistor is now attached to one side of the cage and the IR LED to the other side, so that the mouse breaks the light beam on entry. Screw terminals are provided for connecting the battery pack and magnet. The IR LED and the phototransistor, as seen in **Figure 6**, are fixed simply in a couple of header pin sockets. The fly-back diode D1 in parallel with the coil is fastened directly to the magnet.

Rapid success

Shortly after commissioning the mousetrap, it demonstrated its capabilities in an actual test. A mouse started living rent-free in our cellar, so now it was time to test out the construction. A home-made coconut cookie was concealed at the back of the mousetrap, in the hope that our new resident could not resist sampling it.

Just one day later a dainty little mouse was now inside the trap. For her sake alone constructing the trap had been worthwhile. And the following winter, when a complete mouse family of eight members landed one after another in the trap, we set them all free in the woods some way from our house. That was proof that there was no need to kill uninvited houseguests.

(140011)

Web Links

[1] www.mausefallendorf.de/museum/ (enjoy the illustrations even if you don't read German)

[2] www.elektor-magazine.com/140011

[3] www.abacom-online.de/uk/html/lochmaster.html (free demo)

Component List

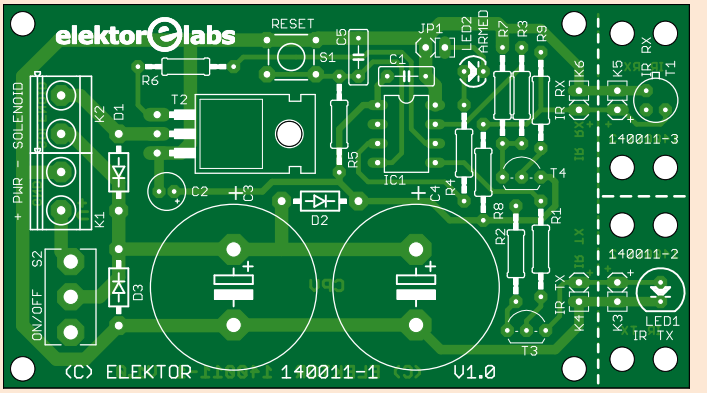
Resistors
R1,R3,R5,R8 = 10kΩ
R2 = 150Ω
R4,R6 = 1kΩ
R7,R9 = 3.3kΩ

Capacitors
C1 = 100nF
C2 = 10µF 50V, 2mm pitch
C3,C4 = 4700µF 50V, 10mm pitch
C5 = 220nF

Inductors
L = solenoid (see text)

Semiconductors
D1,D2,D3 = 1N4007
LED1 = IR ED, 940nm
LED2 = LED, red
T1 = OP803SL or PBY62IR phototransistor
T2 = TIP112
T3 = BC547C
T4 = BC557C
IC1 = PIC12F683-E/P, programmed, Elektor Store# 140011-41 [2]

Miscellaneous
S1 = pushbutton
S2 = miniature switch, 1 contact
K1,K2 = screw terminal block, 0.2"pitch
JP1,K3,K4,K5,K6 = 2-pin pinheader, 0.1" pitch
Battery holder for four batteries or cells
1 jumper
PCB # 140011-1 [2]



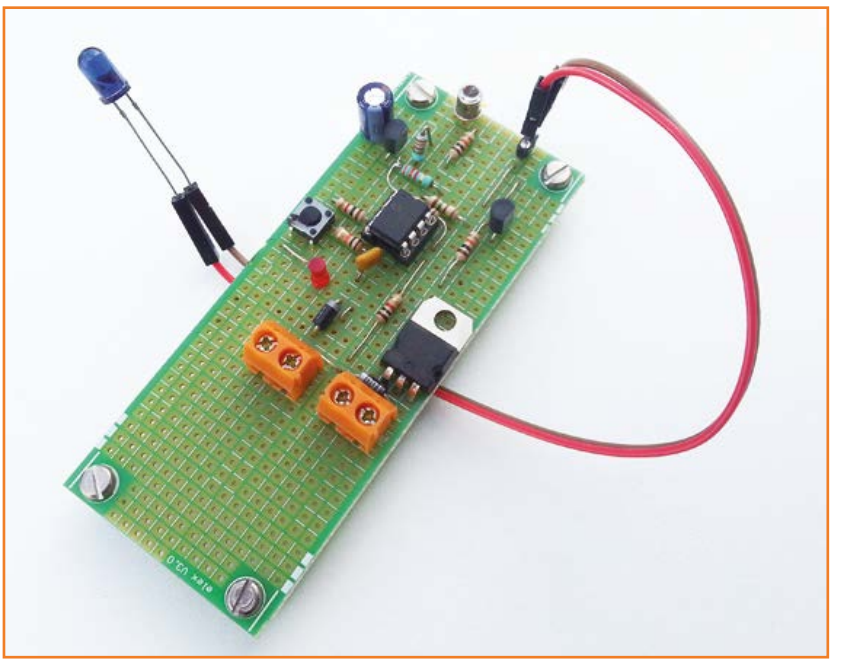


Figure 6. The IR-LED is connected using a long lead that is installed on the other side of the mousetrap.

Brushed-to-Brushless Motor Conversion

Chip tuning for RC models!

by **Jörg Prim**
(Germany)

Exchanging a standard motor in an RC model for a brushless equivalent involves some alterations. But the circuit described makes the task simply child's play.

Plenty of manufacturers of RC airplanes offer keenly priced 'ready to fly' (RTF) models. RTF means that no tinkering is required; the models can be flown straight out of the box (sort of). The only preparation necessary before your new model makes its maiden flight is to pop some batteries in the RC transmitter and charge up the plane's battery. These models are a great fun and ideally suited for beginners too.

All the same, people often feel a desire for a more powerful engine after a few trial flights. That's because most RTF models are equipped, for cost reasons, with standard, brush-type, motors. They are also fitted with a combined 'nerve center' PCB that incorporates a 2.4 GHz or other LPR band receiver, the drive controller and frequently also the servos.
An upgrade should not pose problems; surely all that's necessary is to exchange the weakling motor for a more powerful one? Frequently, however, anything that sounds simple is condemned to failure. Here it's because the new, more powerful motor is heavier and takes more current, making a bulkier battery necessary. As a rule even this is no help, because the sturdier motor is unable to defeat the excess weight of the model.

Lighter, stronger, further

Here's an ideal application for a brushless motor. With no extra weight overhead these motors offer higher performance along with lower wear and tear, because they have no commutator. Standard (brush) motors are controlled by pulses (a few kilohertz in frequency) produced by a drive controller, so as to generate motor control pulses with a repletion rate of around 3 kHz. When the input pulse width is 1 ms the motor does not move, whilst at 2 ms the motor runs at full tilt.

A brushless (BL) motor on the other hand is driven with three-phase alternating current, for which the previous drive controller is not designed. In a BL motor the rotational speed is proportional to the three-phase frequency. The brushless drive controller is much more complex in its construction, since it is tasked not only with generating the desired frequency but also with keeping count of the motor revolutions, in order to keep the motor and power source in step. Fortunately the trade offers plenty of suitable BL-type drive controllers at modest prices, so no problem here.

Only one minor problem remains to be addressed: the BL drive controller expects to see a servo signal on its input (**Figure 1**). This involves pulses of 0.9 to 2.1 ms length repeated every 20 ms. In this case 0.9 ms corresponds to 'motor off' and 2.1 ms to 'full throttle'. The receiver module of the model provides no servo output for the motor, however, since the main PCB already provides a drive controller suitable for standard motors. The output of the receiver unit looks in principle as shown in **Figure 2**.

A power MOSFET is provided with a PWM signal. An N-channel MOSFET is used mostly, driven by a signal that is referenced to ground, in other words switching the motor with positive-going pulses. The PWM signal is a square wave of several kilohertz. In this way the pulse width is varied from 0 to 100 % (often only from 20 to 90 %). In the motor the pulsed drive is transformed into kinetic energy: the greater the pulse width, the higher the rotational speed of the motor. At 100 % the MOSFET is conductive all the time and the motor runs at maximum power.

Servo signal from the microcontroller

The circuit described here now transforms the PWM signal from the receiver module into a servo signal for the drive controller (**Figure 3**). Input K2 is connected to the receiver unit in place of the standard motor. Here only Pin 1 is assigned, deriving its ground connection through the MOSFET of the receiver module. R3 is a relatively low-value pull-up resistor, to ensure that the PWM signal reaches the microcontroller input PB4 with its pulse edges as steep ('fast') as possible.

With D1 we can monitor whether the controller is working. A three-wire servo cable is connected

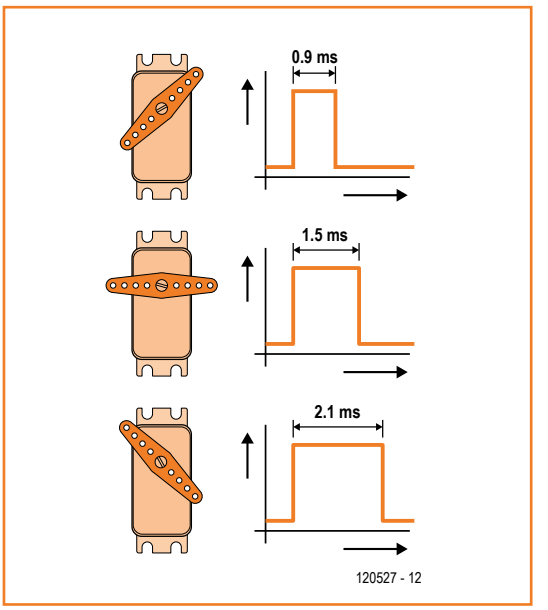


Figure 1.
Here's how a servo signal looks.

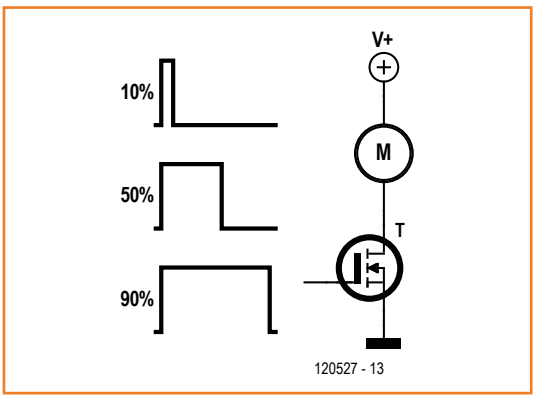


Figure 2.
Typical drive controller configuration for standard motors.

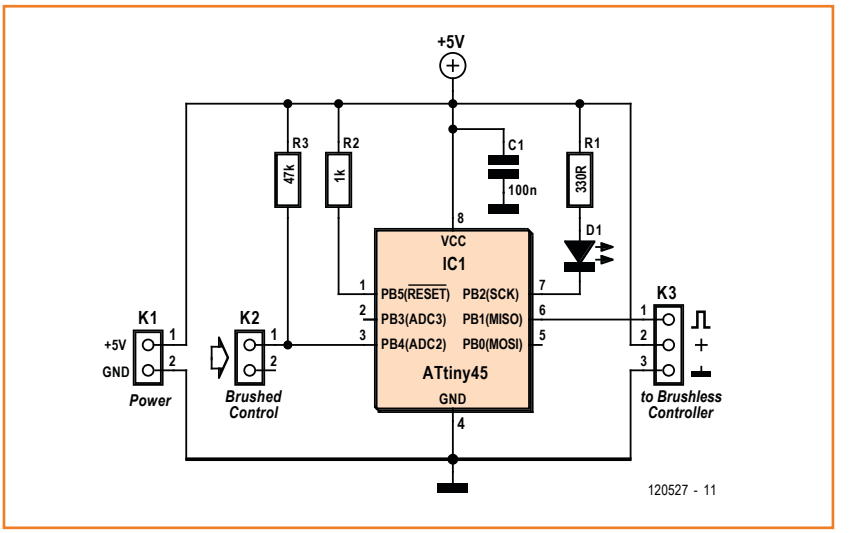


Figure 3. A small circuit with a small microcontroller!

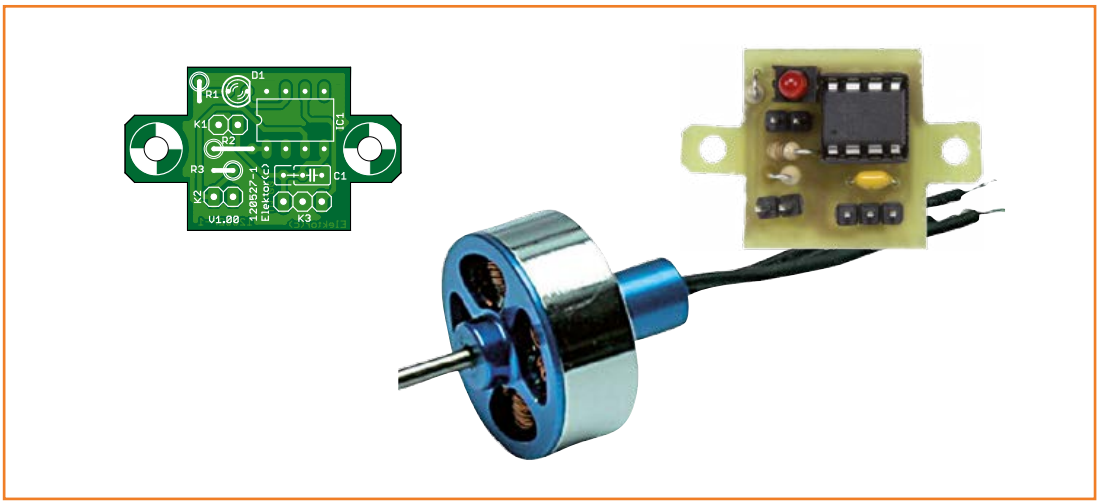


Figure 4.
And a particularly compact PCB...

to the BL drive controller at K3. Pin 3 is ground and Pin 2 the positive operating voltage (5 V) for the drive controller. Pin 1 carries the servo signal from the microcontroller to the drive controller. The software in the microcontroller evaluates the signal at PB4 and is tolerant of PWM frequencies from around 500 Hz up to 10 kHz. The servo signal generated on output PB1 is proportionate to the pulse width of the PWM signal. If no suitable PWM signal is detected, the software always delivers a 0.9 ms 'motor stop' servo signal at the output.

The software is self-calibrating, requiring the user to give a single short burst of full throttle at switch-on for the software to 'learn' the maximum PWM pulse length. Naturally you need to hang on like the grim death to the model during this calibration process. The LED flashes twice briefly after power-on—indicating operational readiness. When the motor is idle the LED is dark but it is illuminated as soon as the 'engine' is turning.
Construction of the small, single-sided PCB in **Figure 4** is child's play. If you provide the 8-pin ATtiny controller from Atmel with a socket, this must hold the IC firmly during even the most intrepid flight maneuvers. The controller is theoretically programmable in-circuit, although for space reasons we had to omit the necessary interface connector on the PCB. If you have no facility (or desire) to program the controller yourself, you can acquire it ready-made, just as you can the PCB [1]. The software (source code and hex code) is also available of course.

Please note that the circuit is designed only for receivers using motors switched to ground and for PWM frequencies between 500 Hz and 10 kHz.

(120527)

Internet Link

[1] www.elektor-magazine.com/120527

Component List

Resistors
R1 = 330Ω
R2 = 1 kΩ
R3 = 47kΩ

Capacitors
C1 = 100nF

Semiconductors
D1 = LED, 3mm, red
IC1 = ATtiny45-20PU, programmed, Elektor Store # 120527-41

Miscellaneous
8-pin IC socket
K1,K2 = 2-pin pinheader, 0.1'' pitch
K3 = 2-pin pinheader, 0.1'' pitch
PCB # 120527-1

Farmer-Goat-Wolf-Cabbage Game

"Old riddles never die"

The "Crossing the River" riddle effectively trains a number of mental skills young children (and some adults) need to develop, like abstracting, strategizing, reasoning, combinatory logic, and Potential Problem Analysis (PPA), all in *set curriculum* at HBS (Harvard Business School).

By **Mark Donners**
(Netherlands)

The riddle comes in several guises depending on region and folklore, but boils down to a virtual problematic situation with no opponent really but your own mind.
The Farmer-Goat-Wolf-Cabbage type is one of the oldest and best liked in the author's country, although native wolves have not been sighted there for at least a century, and there's also the odd problem with goats (something to do with Q but unrelated to James Bond or flipflops). Cabbages fortunately are aplenty in Holland.

Imagine

Imagine this farmer returning from Riddles Market where he bought a Q-free goat, a living wolf and an eco-friendly grown green cabbage. To reach his farm in the *polder* he has to cross a river by a boat that's so small it allows him to take only one of three pieces of cargo at a time.
At this point the TV zapper and MP3 generations should know that the wolf (*canis lupus*) when unattended on a river bank will eat the goat (*capra hircus*), and likewise the goat when unattended will feast on the green cabbage—all to the detriment of the farmer & his wife. Although Wikipedia does not yet request a citation, a wolf can be assumed to not eat cabbage. Also, neither goats, cabbages nor wolves can row a boat successfully.

In the past, the Farmer-Goat-Wolf-Cabbage riddle was played out by standing up in the classroom and reasoning your proposed solution out loud for all to jeer at. Later the graphics-assisted

approach flourished involving a blackboard or crayons. Today, the game can be played using intelligized silicon and a mere hundred thousand bits being moved around invisibly.

Electronics

Looking at the schematic in **Figure 1**, assuming the circuit is powered via S8 and IC4, the program stored in the ATtiny2313 microcontroller first starts an initialization loop to check if the LEDs in the circuit are operational, i.e. correctly fitted and soldered. Next, the game's begin situation is represented by the three LEDs representing the goat, cabbage and wolf, and the center LED row to indicate the farmer's rowing activity on the river. The goal is obviously to get the goat, cabbage and wolf to the other side of the river without any harm befalling to the cabbage or the goat.

The wolf, cabbage and goat each have two LEDs driven by the AT micro (via current limiter resistors) to indicate which river bank they are on (Left or Right). For example the wolf LED pair is LED9 (left bank) and LED11 (right bank). The rowing action by the farmer is visualized by a row of LEDs LED1–LED8 driven by a 74LS138 1-of-8 demultiplexer (IC2) driven by ATtiny port lines PB0, PB1 and PB2.

In this application the speed requirement on the microcontroller is neither high nor exact, allowing the ATtiny2313 to safely operate at about 8 MHz using its internal clock oscillator.

Build & Program

The component overlay of the double-sided printed circuit board designed for the game is shown in **Figure 2**. The project is SMD-Free so assembly and soldering should be easy. All LEDs, switches and even the buzzer are contained on the board so no wiring is necessary except perhaps for the 9-V battery or an alternative power source.

For the circuit to work the AT microcontroller should be programmed before fitting into its socket (*Elektor Labs PPA rec. 13-JV$_{bis}$, rev. B0.1*). Ready-programmed microcontrollers are avail-

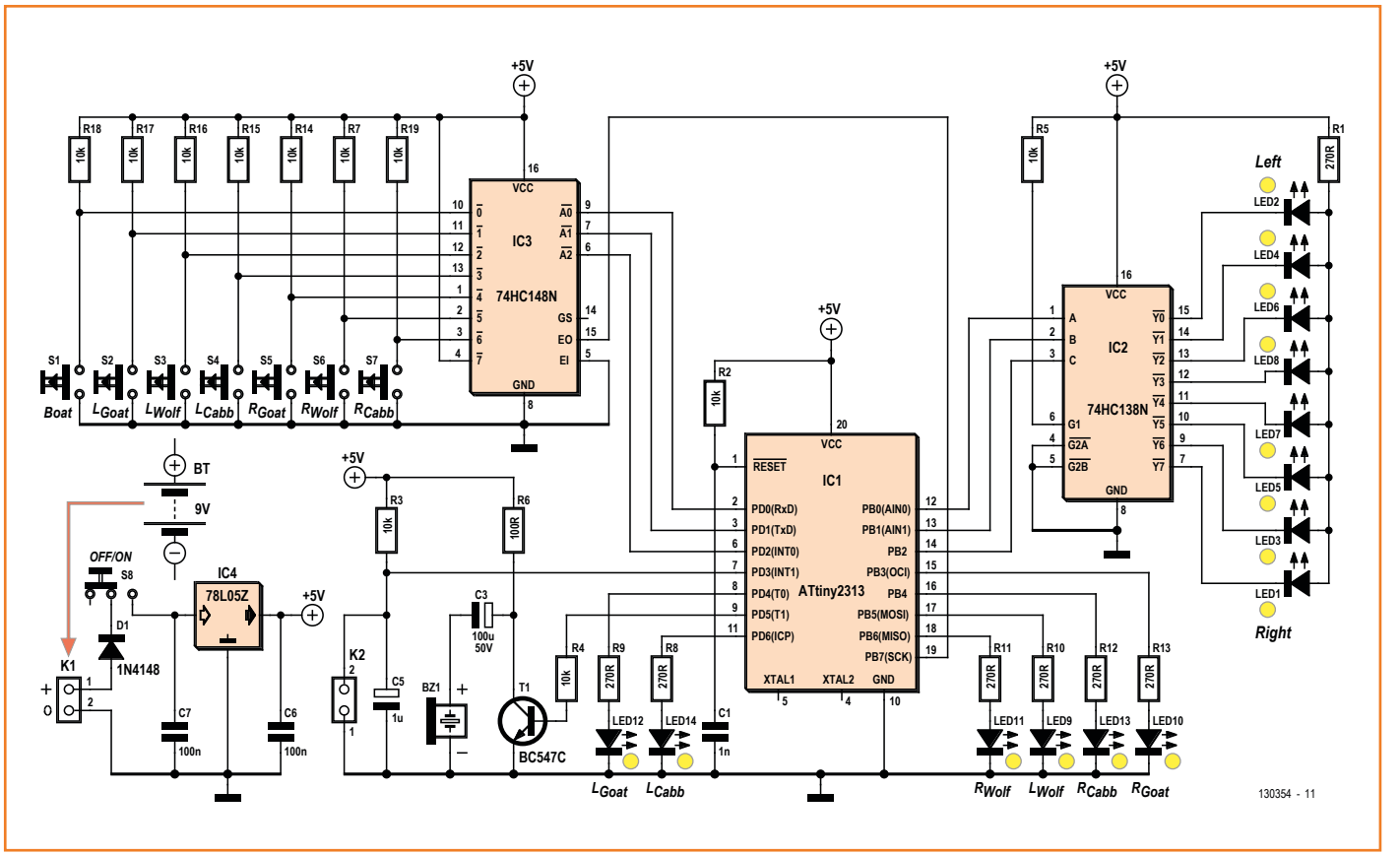


Figure 1.
The game is played using an ATtiny2313 microcontroller, pushbuttons and a bunch of LEDs.

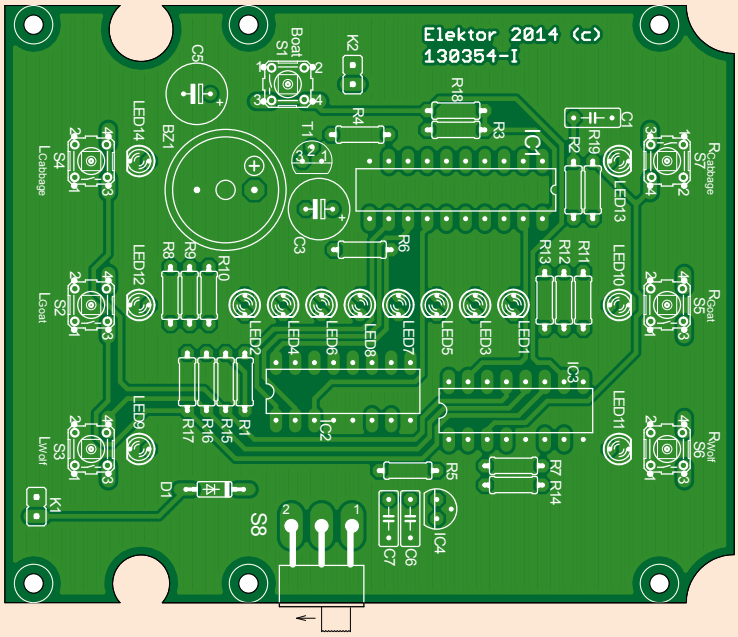


Figure 2. Component layout of printed circuit board designed for the game.

Component List

Resistors
(5%, 0.25W)
R1,R8–R13 = 270Ω
R2–R5,R7,R14–R19 = 10kΩ
R6 = 100Ω

Capacitors
C1 = 1nF
C6,C7 = 100nF
C3 = 100µF 50V radial
C5 = 1µF 50V radial

Semiconductors
IC1 = ATtiny2313-20PU, programmed, Elektor Store # 130354-41
IC2 = 74HC138N
IC3 = SN74HC148
IC4 = MC78L05Z
LED1–LED14 = LED, 3mm
D1 = 1N4148
T1 = BC547

Miscellaneous
BZ1 = buzzer, Kingstate (15027377)
S1–S7 = pushbutton, PCB mount, Omron B3F-1070, 6x6x9.5mm (959698)
S8 = slide switch, SPDT, PCB mount (674357)
K1 = 2-pin pinheader
K2 = not used, optional
9V battery clip + wires
PCB # 130354-1

Numbers in round brackets are Newark/Farnell order codes

able from the Elektor Store as o/n 130354-41. Pinheader K2 was used during the project development phase—it has no function in the current version.

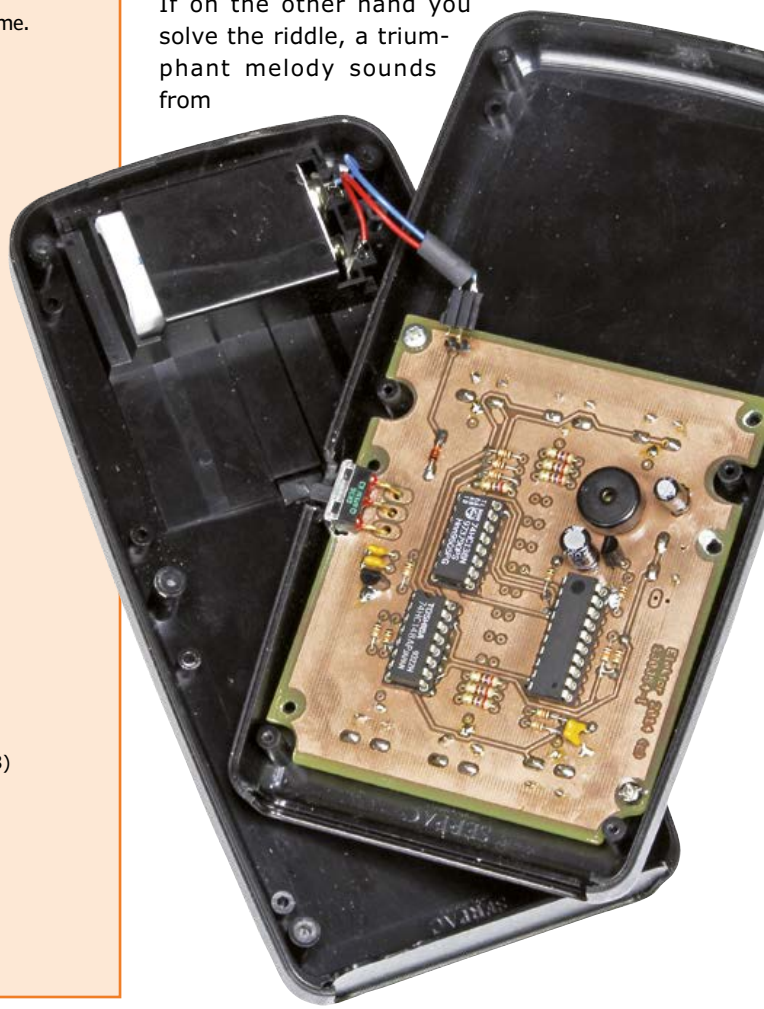
A suggestion to give the enclosure a pastoral/artistic as well as child friendly look is given in **Figure 3**. Where we talked about Left and Right above, it's actually North and South for the river banks.

Play

Either animal or the cabbage can be selected for boating across the river in the desired direction by pressing pushbuttons S2-S7. If you want to get the farmer to row across the river with no cargo, press the Boat button S1.

If you fail the game (cabbage munched or goat devoured on a river bank), eater and victim will start to blink to indicate what HBS people say is "subject to serious reviewing". Also, a sorrowful sound is heard, and the game will start over.

If on the other hand you solve the riddle, a triumphant melody sounds from



Listing 1. Program extract.

```
#include <avr/io.h>
#define F_CPU 8000000UL
#include <util/delay.h>

#define LGoat PD4
#define LCabb PD6
#define LWolf PB5
#define RGoat PB3
#define RCabb PB4
#define RWolf PB6
#define Buzzer PD5

#define statLGoat 0b00000001
#define statLCabb 0b00000010
#define statLWolf 0b00000100
#define statRGoat 0b00001000
#define statRCabb 0b00010000
 #define statRWolf 0b00100000
  #define statLBoat 0b01000000
   #define statRBoat  0b10000000

  /*

Inputs are as follows:

PIND&7 (0b00000111)

7 = S1 (Boat)
6 = S2 (LGoat)
5 = S3 (LWolf)
4 = S4 (LCabb)
3 = S5 (RGoat)
2 = S6 (RWolf)
1 = S7 (RCabb)
0 = not connected

*/
void setLeds(void);
uint8_t readInput(void);
void boatStart(void);
void boatContinue(void);
void playSound(uint8_t Sequence);
```

buzzer BZ1. Also the farmer gets very excited and starts racing across the river in celebration. We surmise a slight problem there arising from his activity but no matter—the game will start again. The program running inside the ATtiny micro is available for free downloading from Elektor's Magazine website [1]. It's supplied for you to feast on the C code and optionally program your own ATtiny2313 chip. An extract appears in **Listing 1**.

(130354)

Web Link

[1] www.elektor-magazine.com/130354

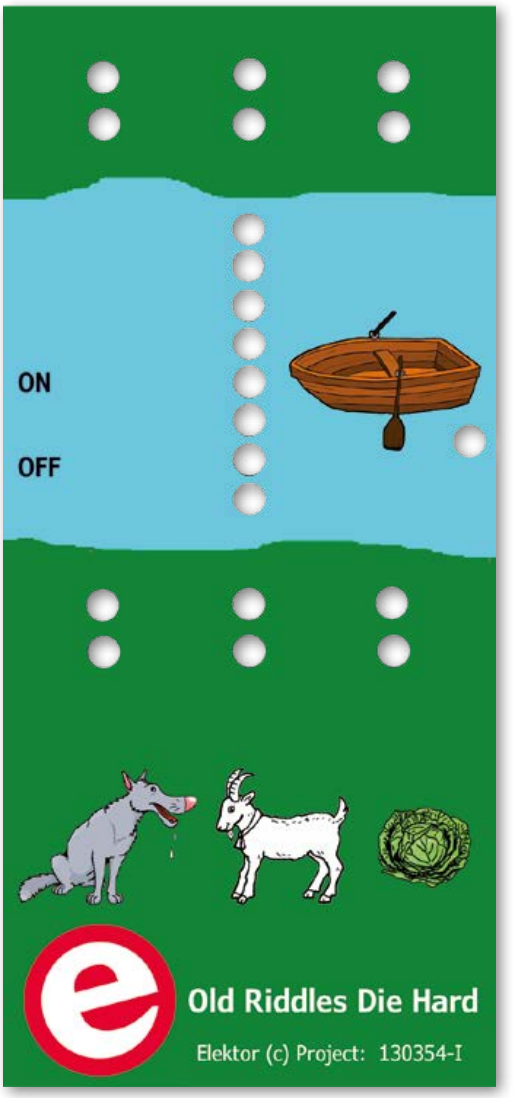
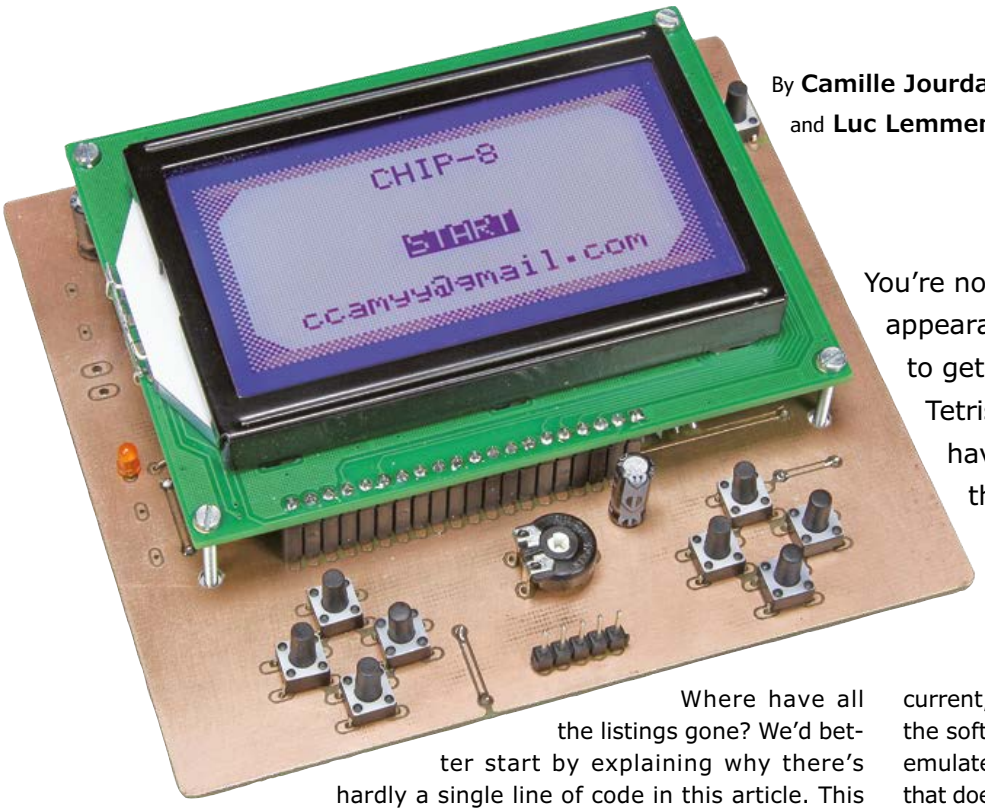


Figure 3.
Suggested front panel layout designed for the Serpac H75 9V case.

SAME: Chip-8 Video Games Emulator

Single Arcade Machine Emulator: a virtual machine on a PSoC



By **Camille Jourdan Gay** (Japan) (ccamyy@gmail.com)
and **Luc Lemmens** (Elektor Labs)

You're not going to believe me, but despite appearances, the aim of this article is not to get you playing, neither Pong nor Tetris, but rather to encourage you to have a go at programming. Happily the one doesn't rule out the other.

Where have all the listings gone? We'd better start by explaining why there's hardly a single line of code in this article. This absence, apparently in contradiction with the subject, is deliberate: rather than approach it through an examination that would necessarily have to be superficial in an article of just a few pages, the full description of the software has been brought together in a very comprehensive and clearly-structured document that Elektor members can download free of charge. Obviously the full source code is available too.

Chip-8 in 2014?

The Chip-8 language originally designed by Joseph Weisbecker for RCA's 1802 microprocessor will soon be 40 years old! [1] In the late '70s and early '80s, it made it possible to create video games without having to worry about their hardware context. This is the idea, still very current, of virtual machines, like Java or others: the software imitates or simulates, or as we say emulates, the functions of an item of equipment that doesn't actually exist. In this way, the Chip-8 video games, written for a virtual platform, are able to run on a real platform with which they have not the slightest affinity. By programming a virtual machine there, we will transform any embedded system into a video games console: a touch-screen mobile phone, a computer, a refrigerator or a washing machine—it doesn't much matter, as long as their microcontroller is capable of emulating a (simple) virtual machine. We just need a microcontroller, a display, a few buttons, and an interpreter for running the Chip-8 video games code that is easy to find on the web. Ideal for students, the exercise is highly educational and very instructive for anyone who wants to gain experience in system architecture programming: registers, stack, memory space, program counter, etc. This lets you get a proper

grasp of how a basic computer operates, in an entertaining way.

The heart of the Chip-8 virtual machine resembles that of a real microcontroller, with its processor, its registers, its stack, its memory, and its two timers. It is capable of executing 35 16-bit instructions (called opcodes) with which the video games are programmed, without requiring any other resource. Everything happens between a few buttons for entering data, and a display for the eyes and possibly a sounder for the ears.
The interpreter is a program that, on an embedded system, executes the code for the Chip-8 video games, recovered ready-made (e.g. `Tetris.ch8`). It is written in the language of the embedded microcontroller. It translates the 35 virtual opcodes into a code that is executable by the real microcontroller. In the code for a game, the opcode `00E0` for example gives a clear screen command in a single line, while in the embedded system code it will be several lines, the number and exact nature of which will depend on the hardware used.

The part of the embedded system memory used by the interpreter is divided in two: it has its own workspace for its own variables, for the interfacing with the hardware, etc.; and then a space for the virtual variables of the Chip-8 architecture, for example, the program counter or stack pointer. The code for the game itself is stored in an array of bytes that the interpreter reads in order to translate each of the opcodes into an equivalent function executable by the microcontroller to which it is addressed. In C, for handling these opcodes, we use quite simply a switch/case structure for layered switching. For example like this one:

```
while(1)
{
  opcode = gameMemory[programCounter];
  switch(opcode)
  {
    case 0001: ExecuteOpcode0001();
      break;
    case 0002: ExecuteOpcode0002();
      break;
    //...
  }
  programCounter++;
}
```

Undefined space

The workspace size is undefined and depends on the hardware and the programmer's choices. For example, the opcode currently being executed (16 bits) does not have a memory space defined by the Chip-8 architecture; it's a combination of the program counter and the virtual memory space (8 bits).
The programmer can choose either to make it into a macro

```
#define OPCODE ((READ_PROGRAM_MEMORY(PC) << 8) |
READ_PROGRAM_MEMORY(PC+1))
```

or to use a variable in the workspace

```
short OPCODE = ((READ_PROGRAM_MEMORY(PC) << 8) |
READ_PROGRAM_MEMORY(PC+1))
```

The first case will limit the use of the microcontroller's memory, the second case, the use of the CPU in case of frequent calling.

Certain of the 35 virtual codes are easy (e.g. ADD or SUB), others can cause quite a headache, particularly those that allow us to draw on the display or the one that produces a random number—a very useful function in games. It's not possible to go into detail here, but nothing of this interpreter will be hidden from you; everything is carefully described in a 31-page downloadable document which we can't recommend too highly to read [2]. There you will find all the detail for programming each code.

It seemed to us pointless and frustrating to show here only certain aspects, which would still be incomprehensible because of the lack of an overall view. On the other hand, this article describes the construction of a real embedded circuit that lets you play Chip-8 video games. This principle is applicable to any other microcontroller circuit with the necessary inputs/outputs. When, thanks to the downloadable documentation, you have seen how I've configured my system (timers, inputs, etc.) and programmed the interpreter, it may be that my approach and my dodges will be useful and will inspire you. Even if you take a different route to arrive at the same result, you will undoubtedly encounter the same difficulties as I did, like for example the lack of memory or the too few inputs available.

Virtuality materialized

We've talked enough about the virtual, now let's take a look at the hardware. Here's what we're going to need:

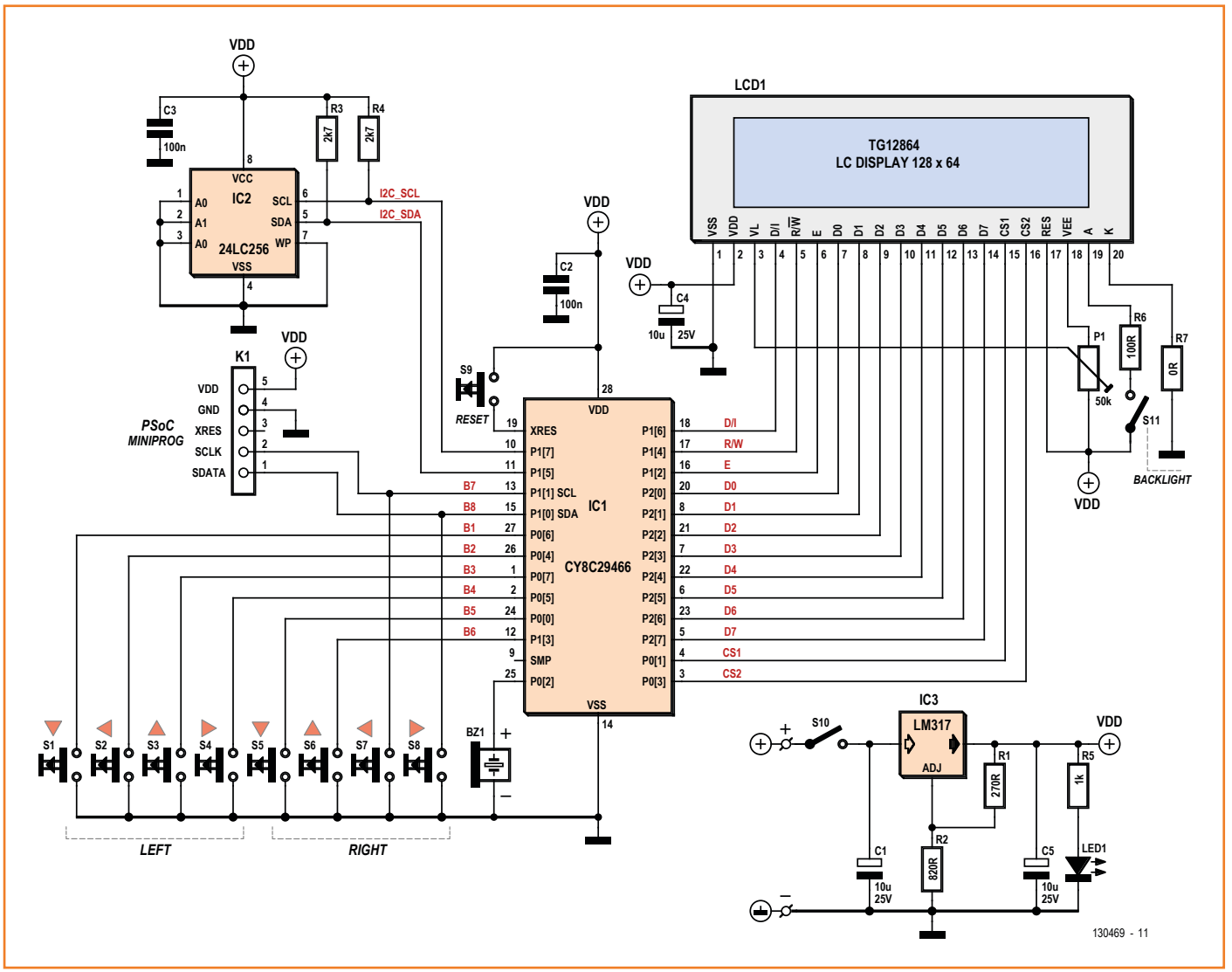


Figure 1.
The simplicity of the emulator circuit is deceptive: within the PSoC IC1 hides a programmable arsenal that runs the Chip-8 virtual machine.

- a matrix display of 64×32 dots or more
- 16 buttons (in practice, 3 or 4 are enough for most games)
- a sounder (unless you prefer silent games)
- a microcontroller capable of emulating the Chip-8 virtual machine, with these specifications:

  - 35 16-bit instructions (opcodes)
  - 4 kB of memory
  - 16-level 16-bit stack, used for the return addresses
  - 16 8-bit general-purpose registers and 3 special registers (memory pointer, program counter, stack pointer).
  - 2 60 Hz 8-bit counters: Delay Timer and Sound Timer.

It's not a lot. I've also made provision for my system to be compatible with a more recent version of Chip-8, called Super Chip (SCHIP), which allows a 128×64 dot display and handles a few additional codes and registers.

If you were expecting a complicated circuit, you're going to be disappointed by **Figure 1**. Apart from the buttons, the liquid crystal display, and the voltage regulator (IC3), there are only two integrated circuits: a 29466 PSoC, which has the status of microcontroller (IC1), and an I²C EEPROM. Oh, and I was forgetting Bz1.

I'd initially conceived this circuit for a single game, resident in the microcontroller memory. While on the way, I had such fun that the idea expanded

An entertaining and educational exercise for gaining experience in system architecture: registers, stack, memory space, program counter, etc.

to house eight games in a separate EEPROM. All that it took was to add a selection menu so the player can choose from the games available. This is, moreover, how the reset button S9 appeared, which is not needed in the monogame version.

A PSoC (Programmable System on Chip) combines on the same chip a microcontroller and miscellaneous digital (counters, convertors, etc.) or analog (amplifiers, filters, etc.) accessories which can be freely combined with the help of the software. My choice of the 29466 PSoC is based on the one hand on its price (€ 30 for the essential programmer and € 5 for the actual PSoC itself) and on the other, on the presence of several digital sub-assemblies that enable us to have two timers, a PWM signal generator (for the sounder) and a built-in random generator. For the buttons S1–S8, there's not even any need for biasing resistors on the corresponding PSoC inputs, as these are internal and configurable.

To create the two 8-bit timers mentioned above, I used several of the PSoC's modules. These can be configured either by laboriously writing the values given into the registers provided for the purpose, or by using a special configuration tool, which is much more convenient: the PSoC Designer, provided for the purpose with a graphical interface, using which all you have to do is arrange the modules as you see fit before establishing the desired connections between modules and between ports using logical operators. This configurator also lets you define global variables like the microprocessor frequency, port configuration, etc. just by clicking. PSoC Designer automatically creates functions for direct interaction between the microcontroller and the modules (e.g. `Timer1_Start()`, `Timer1_Stop()`, etc.)

The ports connected to the buttons are configured as inputs, fitted with internal pull-up resistors. The other ports are configured as outputs. **Figure 2** illustrates the module configuration. The names of the `DELAY_TMR` and `SND_TMR` timers leave us in no doubt as to their function; the second auto-decrements and makes the sounder sound when its value is other then 0. Thanks to an AND gate and `SND_PWM`, this is the generator for the 4 kHz PWM signal made audible by Bz1. PRS8 delivers the random numbers for the Chip-8 RND opcode.
Here are a few of the PSoC's specific features.

Buttons: The state of S1–S8 is read by a function that returns a variable in which each bit corresponds to one of the buttons. It's simple, but when you examine the software and my documentation, you'll see that I did have to cheat because of the internal resistors, whose high bias has to be established each time before reading the port. Otherwise the input stays low even once the button has been released.

Display: There's a huge choice and the display drive techniques vary. Here we use a KS0108 driver chip which uses parallel bit-banging. The DI pin distinguishes the instructions from the data, the RW line read from write, the two CS1/CS2 pins the right side of the screen from the left (128×64 = 2×64×64), and lastly the EN signal for synchronization. The 8 pins on port 2 are used for the data being read or written. In my doc-

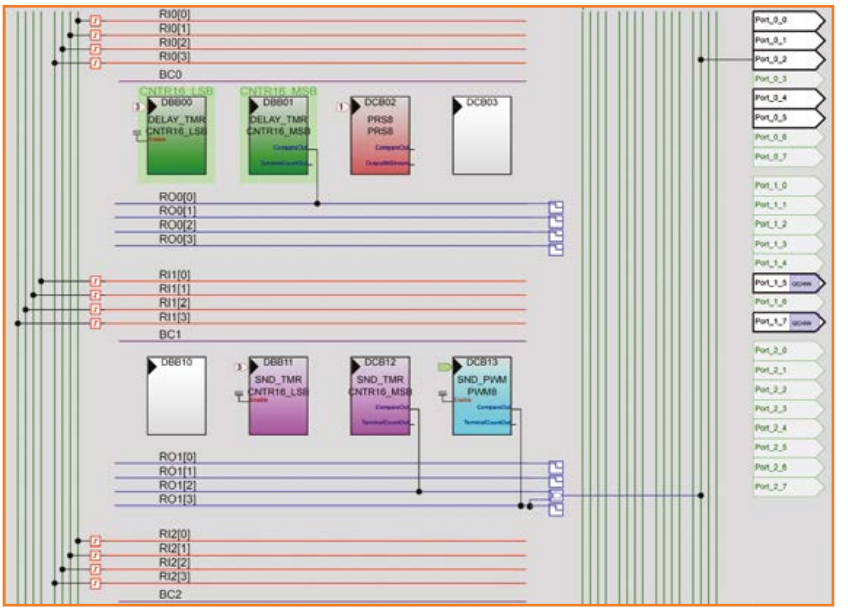


Figure 2. Configuration of the PSoC modules.

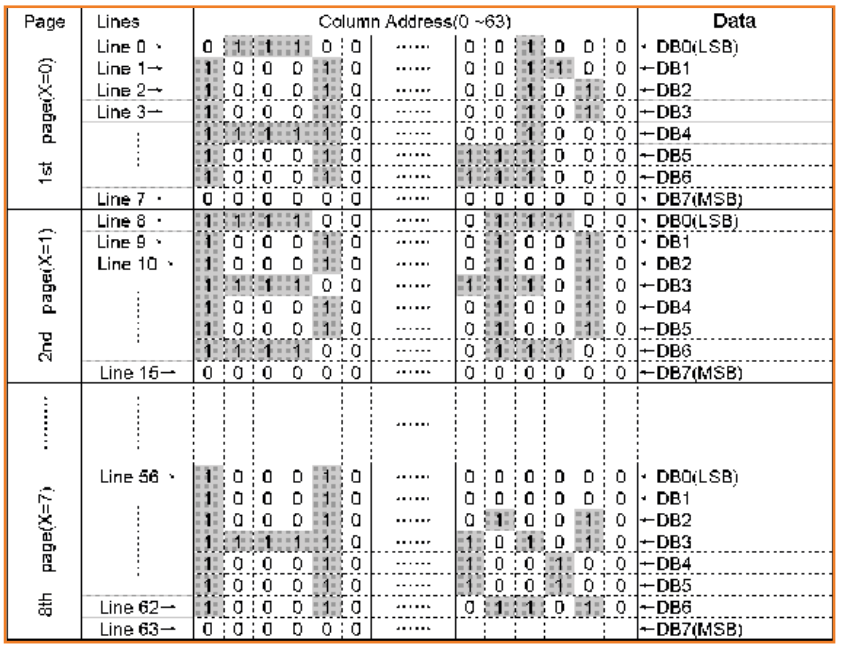


Figure 3.
The pixel configuration for the display is nothing like the way they're organized in the Chip-8 architecture.

umentation [2], you'll find a suitable macro for changing the PSoC data port into inputs (HIGHZ) or outputs (STRONG).

Timers: The 29466 PSoC offers three oscillators (VC1, VC2, VC3) cascaded from the system clock (24 MHz). Here the dividers are configured such that VC1 = 1.5 MHz, VC2 = 93 kHz, and VC3 = 360 Hz. This last frequency, the lowest we can obtain from the PSoC clocked at 24 MHz, will be divided again by 6 to obtain the 60 Hz timing specific to Chip-8. The documentation explains how to do this using a virtual timer. You'll also find there some explanations about the subtle logic combinations of the counter output and of the PWM module, all within the PSoC.

This is not a games console

This is not a games console—don't expect 2014 levels of sophistication. It's normal, for example, for the Pong ball to be as difficult to make out on the liquid crystal screen as if it had been fired at you on a real tennis court by John McEnroe—back in 1982 of course.
The EEPROM is divided into blocks of 4 KB, which corresponds to the Chip-8 RAM, with one game per block. The maximum size of a Chip-8 game is 3584 bytes, which leaves 512 bytes for the header—only 64 bytes of which are actually used: two for a pseudo checksum, one for the game speed, 16 for the keyboard, and the rest for the game title, i.e. eight games in 32 KB of EEPROM. The average size of a Chip-8 game is 256 bytes (1 KB for S-Chip), so the EEPROM could hold many more.
After initialization, the PSoC reads the headers for the games present in the EEPROM and displays their titles so the player can choose. Then they load the chosen game and apply the keyboard configuration and speed parameter, also stored in the game header.

Interpreter

The major business of the interpreter is to simulate the Chip-8 architecture processor in the microcontroller memory. For the registers already mentioned, only a few bytes are required—but you do have to be careful not to get in a muddle. You need to allocate 4 KB to the Chip-8 game memory. But how can we do that with a PSoC that has only 2 KB of RAM? Here again, in the documentation you'll find a programming trick that's really interesting, but whose finesse is outside the scope of this article. Don't miss studying it with a clear head—you won't be disappointed.

Without going into further detail here, we'll just point out that use is also made of one of the remarkable characteristics, and thus one of the advantages of PSoCs over other programmable ICs, which is the possibility of dynamically reprogramming the PSoC ROM via I²C commands as if it were an EEPROM. This is what makes it possible to load a new game into the PSoC without having to reprogram it. Anyone replacing the PSoC by a microcontroller without dynamic reprogramming would still be able to emulate a 4 KB space in 2 KB of RAM—but loading a game dynamically into the ROM from an EEPROM no, that's more difficult!

We've reached the point where the interpreter, after having prepared the memory, can start simulating the Chip-8 code cycles. We've already seen that this mainly consists of a while(1) loop that scrolls through the game's opcodes using a switch/case, executes them, and usually increments the program counter (except e.g. in the case of a jump). Our interpreter's other tasks are scanning the buttons, detecting any errors (e.g. by stack overflow or memory access fail) and other needs like e.g. introducing a delay to avoid certain games running too fast. On this subject, you will have noticed that as we have 16-bit opcodes in an 8-bit array, the incrementing is done in steps of 2.

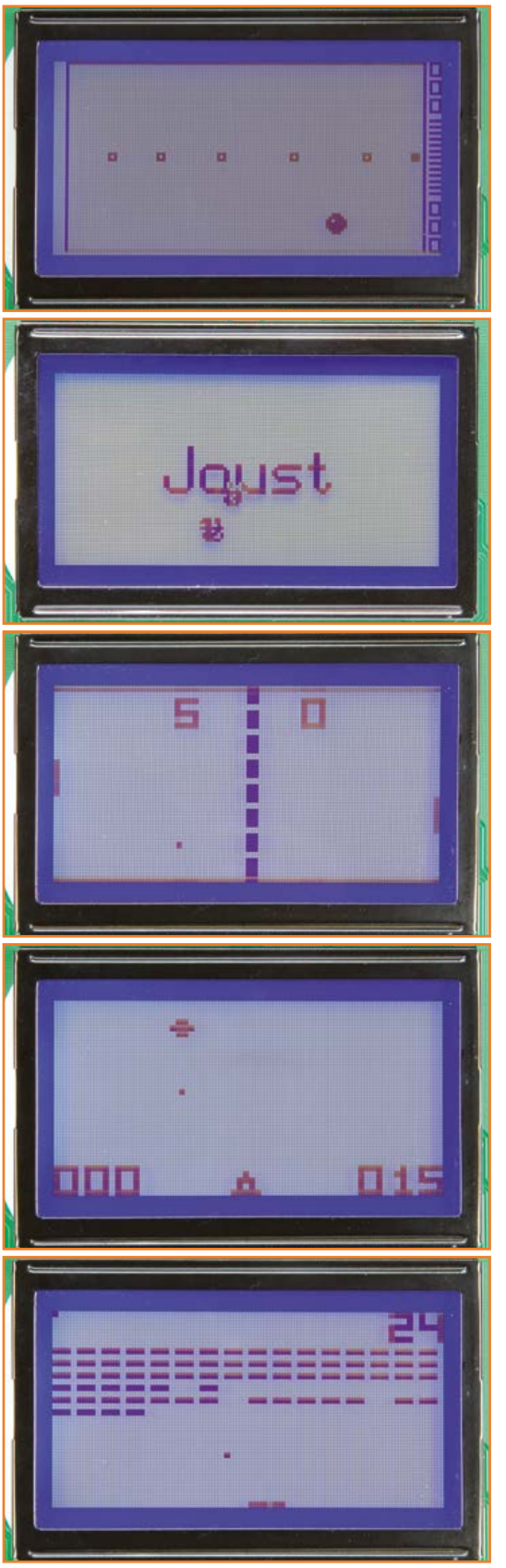
All in all, the skeleton of the interpreter is not very imposing:

```
while(1)
  {
    //Clear the flags
    OS_FLAGS = 0;
    ERROR = ERROR_NO_ERROR;
    input_read(&buttons);
    //fetch the opcode (16 bits)
    OPCODE = ((READ_PROGRAM_MEMORY(PC) <<
8) | READ_PROGRAM_MEMORY(PC+1));

    switch((OPCODE >> 12)& 0xF)//we
decode the opcode by groups of 4 bits
      {
      /*OPCODE DECODING AND EXECUTION*/
      }
    if(ERROR)
    {
    /*ERROR HANDLING*/
    }

    if(!((OS_FLAGS >>
PC_NOT_INCREMENT_FLAG)&1))
    {
      PC += 2; // Increment the PC except
if told not to
    }
  }
```

Here we are now arrived at the 35 opcodes, among which we'll be able to spot:

- the arithmetic operators like ADD and SUB which most microcontrollers know
- the branch operators like JMP and CALL – apparently simple, but which, because they affect the virtual architecture, need to be handled with care
- the operators that affect the memory, which are no problem for microcontrollers with plenty of memory, but which oblige us to use dodges if the microcontroller's memory space is tighter than that of the Chip-8 architecture.
- the operators that affect the hardware and depend directly on the hardware configuration

All this is described through the menu in the on-line documentation, as are the opcodes specific to the timers, buttons, and random numbers. We'll conclude here with one special category,



the Chip-8 graphics opcodes. We've seen that the standard display format is a 64×32 matrix. As we have a 128×64 dot screen, one Chip-8 pixel corresponds to 4 actual pixels. The origin (0,0) is the top left corner. The graphics codes are by far the most difficult to transpose – among other things, because the logic of our display is totally different from that of the Chip-8 architecture, which writes in "lines" of 8 pixels, while the LCD display writes in "columns" of 8 pixels. Chip-8 draws the pixels from co-ordinates in fixed-width groups (8-bits) and variable height. The 128x64 display is divided into two 64x64 screens, in turn divided into 8 pages, themselves divided into 64 columns (**Figure 3**). Hence we are obliged to write our pixels in column of 8 within one of the 2*8*64 boxes, which implies a fixed height for these pixel groups. We're only going to be able to use the fixed Y co-ordinates (lines 0, 8, 16, 24, 32, 40, 48, and 56) and each write operation will also affect certain of the neighboring pixels, which requires a certain degree of gymnastics. The on-line document explains how to set about this and gives other valuable explanations about managing the display.

Construction

After all that virtual stuff, here we come now to the concrete bits.

The suggested PCB layout (**Figure 4**) is double-sided, but for the sake of a few straps, you can easily produce your own single-sided one—especially as only through-hole components are used. The 128×64 pixel display with its KS0108 chip won't be any problem to source, but do watch out for its pinout. We ordered a TG12864B-02 (Vatronix), but received a TG12864B**-03,** on which pins 19 and 20 for the anode and cathode of the backlight panel are reversed. So watch out for that suffix! Whence the presence of the very

Figure 4.
The PCB designed as double-sided is easy to make single-sided.

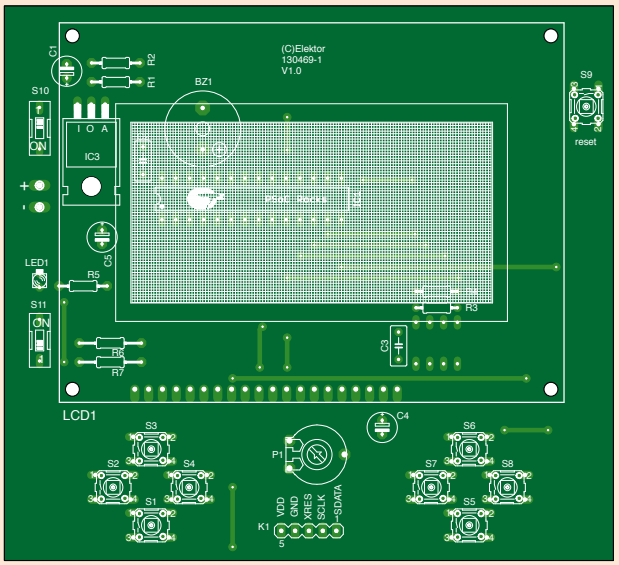
Component List

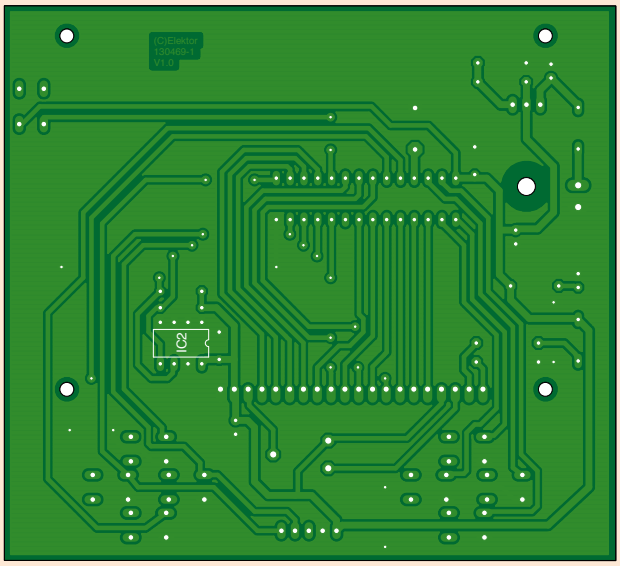
Resistors
R1 = 270Ω 0.25W 5%
R2 = 820Ω 0.25W 5%
R3,R4 = 2.7kΩ 0.25W 5%
R5 = 1kΩ 0.25W 5%
R6 = 100Ω .25W 5 %
R7 = wire link (see text)
P1 = 47kΩ trimpot

Capacitors
C1,C4,C5 = 10µF 50V radial
C1,C2,C3,C4,C5,C6,C7 = 100nF 50V 20%

Semiconductors
LED1 = LED, red, 3mm
IC1 = CY8C29466, Cypress PSoC, programmed, Elektor Store # 130469-41
IC2 = 24LC256 EEPROM
IC3 = LM317

Miscellaneous
S1–S9 = pushbutton, PCB mount, 24V 50mA, 6x6 mm
S10,S11 = on/off switch
8-way DIL IC socket for IC2 (see text)
LCD1 = 128x64 pixels, e.g. Vatronix TG12864B-03 or Midas MC128064A6W (see text)
Bz1 = 5 V buzzer





philosophical 0 Ω resistor R7 (in fact a simple wire strap) for changing the configuration of these two pins according to which model of display you have. Where necessary, cross over R6 and the wire strap (R7) so that pin 19 is grounded via R7 and pin 20 connected to V_{DD} via R6.

Farnel/Newark offer the MC128064A6W (Midas), but if you want to find cheaper, you can take a look on eBay.
Although it's not obligatory, the backlight is undeniably convenient for the player. It only uses 20 mA.
The EEPROM is fitted underneath the PCB (solder side) to make access easier; if you want to add or change games, it's easier to remove the EEPROM from its socket to plug it into the programmer than if it was under the display! Note in **Figure 5** the special type of socket to make up yourself in order to be able to solder it on the copper side of the PCB!

The regulator IC3 is fitted flat to the PCB. If you make a double-sided PCB like the one in the photos, the metal surface of the LM317, which is at V_{DD} (+5 V), must not come into contact with the copper of the surface, even if this has no voltage on it. As a precaution, it's best to arrange sufficient spacing and fix the regulator using a nylon nut and bolt (**Figure 6**).

The PSoC is programmed using a PSoC MiniProg interface connected to K1, directly from the PSoC Designer programming environment (we used version 5.4). During programming, the whole circuit can be powered via this interface and the normal supply is not needed.

Firmware

Elektor supplies the (source) code for programming the games into the EEPROM and the code for the PSoC emulator, but not for the actual games themselves! The programmed PSoC available from the Elektor Store [8] (# 130469-41) contains only the firmware. You won't have any trouble downloading Chip-8 games code [3]. Some are in the public domain, others not. Be sure to respect any copyright; this remains in force even after decades.

If you want to program the firmware in the PSoC flash memory yourself, connect a MiniProg (Cypress) programmer to K1.

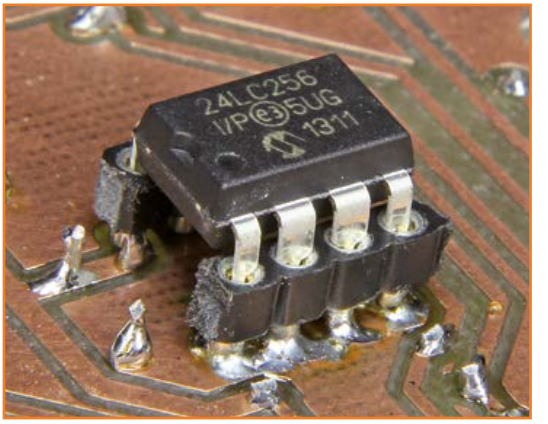


Figure 5.
IC2 must be fitted into a socket that you'll have to make up yourself from socket headers, in order to be able to solder the pins to the PCB tracks.

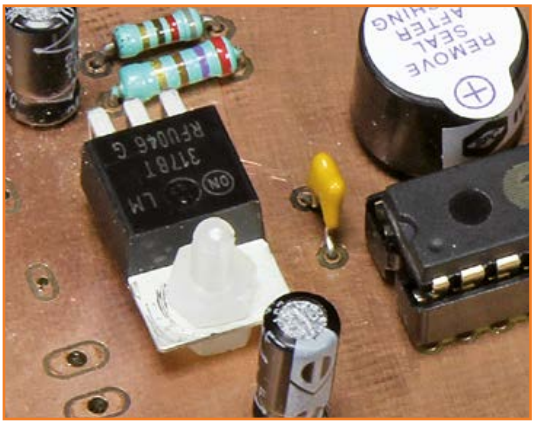


Figure 6.
We can see the regulator fitted a small distance from the copper surface.

Here are the three ways to load a game:

- insert the game into the EEPROM using a third-party tool that lets you copy a file to an EEPROM;
- insert the game in the EEPROM using the tool [9] that I've made available; in this case, the emulator board (Figure 4) is used for flashing the EEPROM;
- insert the game directly into the PSoC flash memory. In this case, you'll need to modify the default values of the interpreter vari-

Figure 7.
Menu for selecting the games the PSoC finds in the EEPROM.

About the author

Following his Electrical Engineering studies at the INSA in Lyon (France) and an exchange year at the University of Tokyo, Camille Jourdan is currently in Japan under an international business volunteer scheme with Ichikoh Industries, who manufacture automotive lighting. He has worked in electronics, image processing, and in software development in Tokyo and Paris.

This Chip-8 project was designed and built on a personal initiative during his free time, in Japan, using equipment from Akihabara, Tokyo's electronics district—a place of pilgrimage he recommends to Elektor readers!

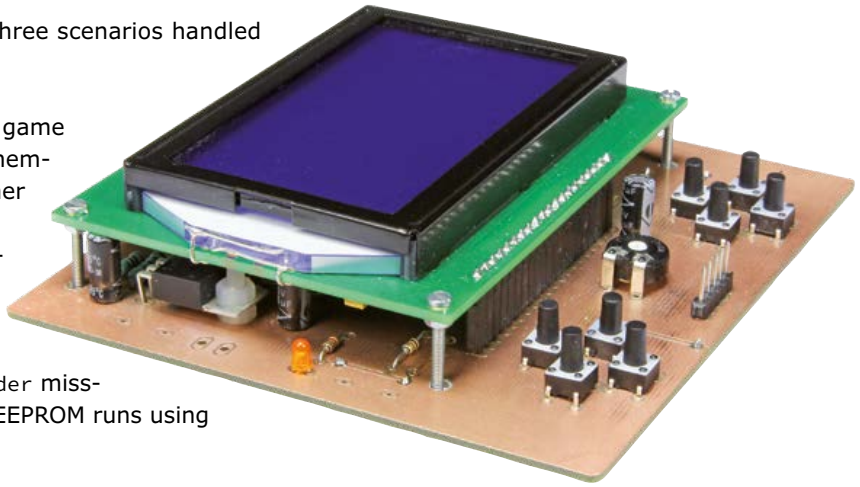
Camille is grateful to Matthieu Denoual for advising him to publish this project, together with his Laboratory Director Yoshio Mita, and of course Luc Lemmens at Elektor Labs!

ables in order to be able to insert the game there. Hence the code must be located in the `PROGRAM_MEMORY` array just after the typefaces (`global.c`).

To close, here are the three scenarios handled by the interpreter:

- EEPROM missing: the game present in the PSoC memory will be run (whether it was copied from an EEPROM that had previously been connected or from the PSoC programmer)
- EEPROM present, `header` missing: the game in the EEPROM runs using the default settings
- EEPROM present with `headers`: selection menu runs, then the game selected using the settings indicated.

(130469)



Web Links

[1] General description of instructions, http://en.wikipedia.org/wiki/CHIP-8

[2] Full documentation for the SAME Chip-8 emulator, www.elektor-magazine.com/130469

[3] Commented games code and documentation by David Winter, www.pong-story.com/chip8/

[4] Details of the Chip-8 instructions, http://devernay.free.fr/hacks/chip8/C8TECH10.HTM

[5] Information about Super-Chip and a good example of an interpreter (in French) http://blogs.wefrag.com/mrhelmut/2012/01/14/recette-creer-son-premier-emulateur/

[6] Useful libraries (fonts) for programming LCD display screens, http://en.radzio.dxp.pl

[7] Tool for converting game code into hex arrays, www.fourmilab.ch/xd/

[8] the PSoC in the Elektor Store, www.elektor.com/psoc_chip8

[9] Instructions for programming the EEPROM, www.elektor-magazine.com/130469

TH LED Matching & Sorting Accessory

By **Henry Morizot**
(France)

This project is a spinoff from, and an accessory to, the magnificent *Precision Adjustable Current Source* published earlier this year [1]. The simple tool shown here can help to sort or select through-hole LEDs by visually comparing their color, brightness and/or radiation angle.

Though they may all look identical in your storage box, LEDs can look completely different when lit, especially when mounted close to each other (like in an LED bar or panel).

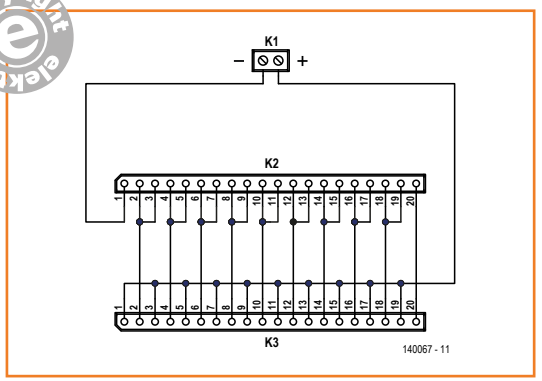
In the "early LED days" color and brightness differences occurred with LEDs from the same manufacturer, and sometimes even from the same type and production run.

Production processes may have improved, but most of you should own up to having at least one box marked "LEDs" containing devices of undetermined brand and type numbers. With this accessory LEDs can easily be connected in series by inserting their pins into a socket strip. Next, the precision current source can be set to different values to see what the light bar looks like. Even small differences in color, brightness and radiation angle can be spotted with confidence.

With our *Precision Adjustable Current Source* the maximum clamping voltage is about 20 V, meaning we can power roughly up to 10 'normal' LEDs in series. This number depends on the type and color; a red one will typically have a forward voltage of some 1.8 volts whereas blue may be in the 3-volts region.

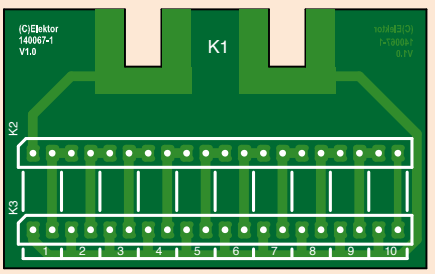
Set the clamping voltage of the DC current source to maximum value (fully clockwise), and the current to the 1 mA – 20mA range. Start with the reference LED in the left-most socket position, with the cathode pointing to the left, and insert the other ones in neighboring positions to the right. A jumper must be set on the two pins in the same position in front of the rightmost LED. All LEDs should be lit then, the reading on the LCD of the current source showing the total forward voltage of all the LEDs in series. You can switch off one or more LEDs by moving the jumper, that way you can also determine the forward voltage of each individual LED. If the clamping indicator on the front panel of the DC current source lights, the forward voltage of the LED chain exceeds the range of our DC current source (you did set that clamping to max—right?).

(140067)



Component List

K1 = 2-way connector
(pads and clearance at PCB edge)
K2 = 20-way SIL socket, 0.1" pitch
K3 = 20-pin SIL pinheader, 0.1" pitch
JP = Jumper, 0.1" pitch
PCB # 140067

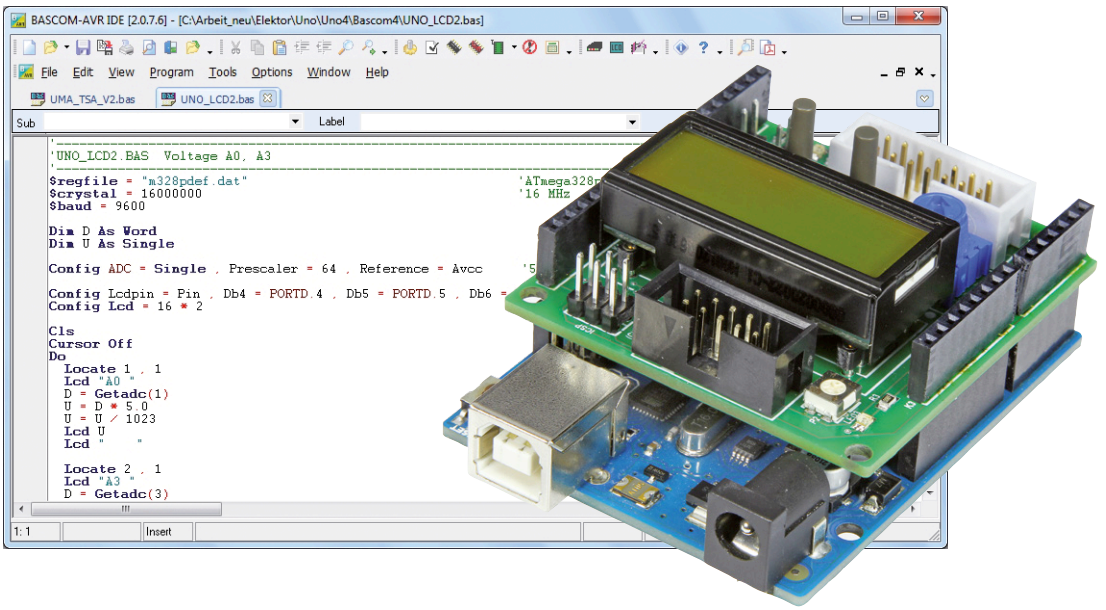


Reference and Web Link

[1] Precision Adjustable Current Source: Elektor April 2014, www.elektor-magazine.com/130287

Microcontroller BootCamp (4)

User interfaces



By **Burkhard Kainka** (Germany)

Serious microcontroller applications usually have some sort of interface between humans and the machine. You can do a lot with a just a small display, a few buttons, a potentiometer and a couple of LEDs. What's more, you can hit the ground running with the Arduino shield developed by Elektor, which is described elsewhere in this edition.

First let's see what it takes to drive a display. Various types are available, including graphic and text displays. Text displays range from tiny with a single line of 8 characters, to large with four lines of 20 characters. As the saying goes, small is beautiful, and in many cases you don't have to display a lot of characters. That's why the new Elektor shield has a small two-line text display with eight characters per line. In any case, that's enough room for messages such as "V=3.3V" or "I=8.2mA". What more do you need on your electronics bench?

LCD connection

Figure 1 shows how the display module on the shield is connected to the ATmega328 on the Arduino board. You can also see the other peripheral components there: two LEDs, two pushbuttons and a potentiometer. If you wish, you can build all this yourself on a piece of prototyping board. You don't even have to use exactly the same type of display module. One with two lines of 16 characters would work just as well. If you like to keep things simple, you can order the new shield on the web page for this article [1], where as usual you can also download a file containing the code for the programs described here.

The display occupies six pins on port D: D2 to D7. That's handy, because it leaves exactly two spare for the serial interface: D0 (RXD) and D1 (TXD). Ports B and C remain free for whatever strikes your fancy. All standard display modules

of this sort can optionally be operated in 8-bit mode, which means that the data is sent to the display over eight lines in parallel. However, 4-bit mode has become customary to reduce the number of port pins used on the microcontroller. In this mode each data byte is sent to the display in two steps. For example, if you want to write an upper-case A you have to send the hex value &H41 (binary &B01000001, or 65 in decimal notation), as specified in the ASCII code table. In fact you first send &B0001 and then &B0100. However, you don't actually need to know all these details or how to go about initializing the display, since Bascom does all the hard work for you. If you simply write `Lcd "A"` in your program, the letter A will appear on the display.

Our first example program (Listing 1) shows how it all works. First you need a `Config` instruction to tell the compiler how the display pins are connected to the port pins. Next you have to initialize the display with the `Config Lcd = 16 * 2` instruction. You can choose from several formats here. The format 8 * 2 is not supported because the display controller in the LCD module is designed for two lines of 16 characters (16 * 2), even with this small display. That's not a problem in practice because the program keeps on running without an error if you accidentally write up to 16 characters in a line. However, when you see the truncated output on the display you will have to consider how you can shave a few more characters off the message.

After you initialize the display, you should clear everything with the `Cls` (Clear Screen) command. Everything written after this lands on the first line, starting at the left end. Here it is the word "Elektor" – seven characters, short and sweet. If you send another character to the LCD now, it will be written to position 8 on the first line, but what you actually want is to start with the second line. For this you use the `Locate` command. Here `Locate 2 , 1` causes the next character to be written to the leftmost position of the second line. The program outputs an incrementing number at this position. The count variable `N` is incremented once per second, so you can use the program to measure time. For example, it's

Listing 1: Text and number output.

```
'-------------------------------------
'UNO_LCD1.BAS  Text Output
'-------------------------------------
$regfile = "m328pdef.dat"
'ATmega328p
$crystal = 16000000             '16
MHz
$baud = 9600

Dim N As Word

Config Lcdpin = Pin , Db4 = Portd.4 ,
Db5 = Portd.5 , Db6 = Portd.6 , Db7 =
Portd.7 , E = Portd.3 , Rs = Portd.2
Config Lcd = 16 * 2

Cls
Lcd "Elektor"
'Cursor Off
Do
  Locate 2 , 1
  Lcd N
  N = N + 1
  Waitms 1000
Loop
```

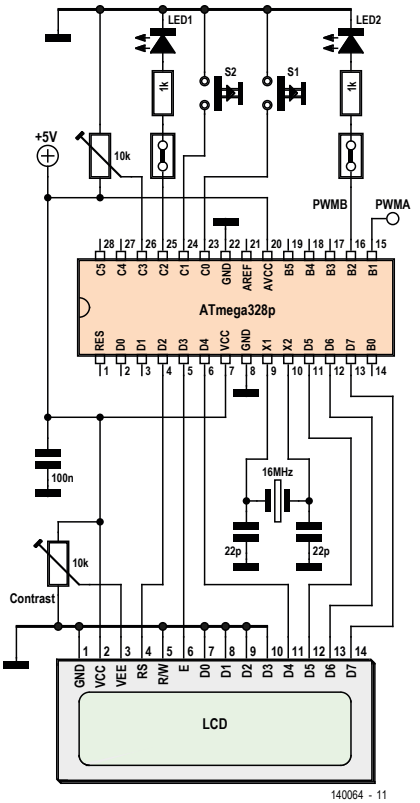


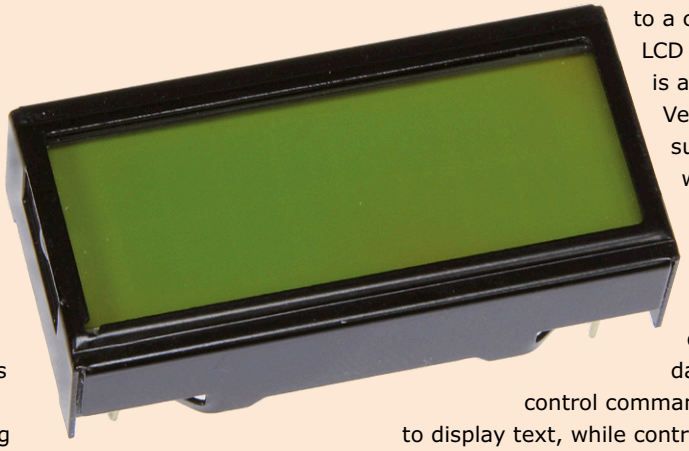
Figure 1.
Basic diagram: how the LEDs, buttons and display on the Elektor shield are connected to the ATmega328 on the Arduino Uno board.

taken me 1200 seconds to write these lines since I started the program. That's 20 minutes, which means I'm pretty slow today.

There are two `Lcd` instructions in this example program, and it's important to understand that they do two completely different things. The pure text output instruction simply copies the characters set in quotation marks by the programmer. By contrast, the instruction `Lcd N` converts the numerical value in the variable `N` into text and then sends this text to the display. In this case `N` is an integer, but in other cases it could be a real number with decimal places. Bascom decides on its own what has to be done, which is very convenient for the user. In other languages the programmer has to do more in these situations. On the display you will see an underscore after the number that was output, which represents a cursor. The cursor shows where the next character will be written if and when it comes. This may be very helpful when you're busy typing something in, but in our case the cursor is irritating. Fortunately, we can do something about this with the `Cursor Off` command, which is initially commented out in the code. If you delete the comment character, recompile the program and download it to the microcontroller, you will

Liquid Crystal Displays

Liquid crystal displays (LCDs) are very popular because they are easy to read and don't need much power. They can be operated without a backlight, although the backlight on the Elektor shield is always on. This still results in much less power consumption than an LED display with the same number of characters.



The reason LCDs are so energy efficient is that they do not have to generate light. Instead, they control whether ambient light is transmitted or blocked. The liquid crystal material, which is located between two sheets of glass, rotates the polarization plane of the incident light depending on the applied voltage. A polarization film is always located on one of the glass sheets. (Incidentally, if you have a defective LCD module you should sometime pull off this film, because you can use it to perform interesting experiments with light [2][3].) The display element of an LCD module is driven by square-wave voltages with no DC component, which are applied across the individual segments. There are dedicated CMOS ICs for this purpose, but some microcontrollers can also drive LCDs directly. However, standard LCD modules have special built-in controllers that handle the actual drive tasks.

The great-grandfather of all LCD controllers was the HD44780. Even today, most controllers are HD44780 compatible. This involves several standardized control commands as well as the connections to the display module. There are usually 14 lines, and in some cases two more for the backlight. The display on the Elektor shield need only 14 lines because the backlight is hardwired.

About the individual lines: GND and Vcc are obviously ground and +5 V. Vee is connected to a contrast trimpot. With many LCD modules the optimal setting is approximately 4.5 V between Vee and Vcc. It is therefore often sufficient to connect a fixed resistor with a value of 470 Ω to 1 kΩ between Vee and ground, which saves the cost of a trimpot.

RS is an input that allows the display be set to receive either data commands (RS = 1) or control commands (RS = 0). Data is necessary to display text, while control commands are necessary for initialization and positioning the cursor. R/W switches between reading and writing. Reading is actually only necessary to determine whether the display is ready to receive the next data. This can also be ensured by a short wait time, so the read direction is often not used and R/W is tied to ground. E is the Enable input. It is used to indicate to the display controller that valid data is present on the data bus (lines D0 to D7) and should be read in by the controller. The controller always reads in the current data after a falling edge. In 8-bit mode all eight data bits are put on the bus and then a falling edge is generated on the E input. In 4-bit mode this process occurs twice, with only the upper four data lines (D4 to D7) being used for data transfer. The lower nibble (least significant bits) is sent first, followed by the upper nibble. The display must be initialized to operate in either 4-bit mode or 8-bit mode before it is used.

see that the cursor has vanished.
This is a good place to remind you how easy it is to display more information by using the Bascom help function. Simply move the cursor (here we mean the one on your PC monitor) to the keyword concerned and press the F1 key, and you're already a lot wiser. For instance, you can learn how to make the cursor blink.

A two-channel voltmeter

Happen to need a two-channel voltmeter? **Listing 2** shows one of many possible solutions. In the previous instalment we told you how to convert the data provided by the A/D converter into a voltage reading. Now let's see how you can show it on the display. First we write a string at the start of each line that identifies the input. The is followed by a space character to separate it from the reading. We also tack on a couple of spaces after the reading (in this case a real number) has been output. Without them, the following situation could occur: Your last reading was 4.859 V, and the new reading is 5.0 V. Bascom outputs "5.0" just as it should, but the rest of the previous output is still there on the display, so you see "5.059 V" and wonder what's going on. To prevent this from happening, we always delete anything that might be present after the current output. Here we accept the risk that this may involve characters intended for a larger display that are not visible on the actual display.

Incidentally, this program uses the ADC0 and ADC3 channels for its measurements. You can connect a signal source to ADC0, preferably with a protective series resistor as shown in **Figure 2**. There's already something connected to ADC3: the potentiometer on the Elektor shield. You can adjust it to provide a voltage from 0 to 5 V.
All of the Arduino pins are also fed out to socket headers on the shield. If you connect a hefty electrolytic capacitor (you surely have an electrolytic somewhere on your bench – but don't exceed 470 µF) between the GND and ADC3 pins, which puts it in parallel with the potentiometer, you have a low-pass filter and you can see on the display how the voltage gradually adjusts after you change the potentiometer setting. The reason for the 470 µF limit is that you might allow the capacitor to charge slowly to 5 V and then quickly turn the potentiometer towards zero. If the capacitor is too large, the resulting discharge current could fry the potentiometer.

Measuring brightness

Now let's edit the program so that the ADC2 channel is displayed instead of ADC0 (see **Listing 3**). On the shield board, LED1 is connected to this channel through a jumper and a 1 kΩ series resistor, since this port pin can also be configured as an output. However, in this case we leave it in the high-impedance state. Now you can see

Listing 2: Displaying voltages.

```
'-------------------------------------
'UNO_LCD2.BAS  Voltage A0, A3
'-------------------------------------
$regfile = "m328pdef.dat"
'ATmega328p
$crystal = 16000000            '16
MHz
$baud = 9600

Dim D As Word
Dim U As Single

Config Adc = Single , Prescaler = 64 ,
Reference = Avcc     '5 V

Config Lcdpin = Pin , Db4 = Portd.4 ,
Db5 = Portd.5 , Db6 = Portd.6 , Db7 =
Portd.7 , E = Portd.3 , Rs = Portd.2
Config Lcd = 16 * 2

Cls
Cursor Off
Do
  Locate 1 , 1
  Lcd "A0 "
  D = Getadc(1)         'A0
  U = D * 5.0
  U = U / 1023
  Lcd U
  Lcd "    "

  Locate 2 , 1          'Pot
  Lcd "A3 "
  D = Getadc(3)
  U = D * 5.0
  U = U / 1023
  Lcd U
  Lcd "    "
  Waitms 1000
Loop
```

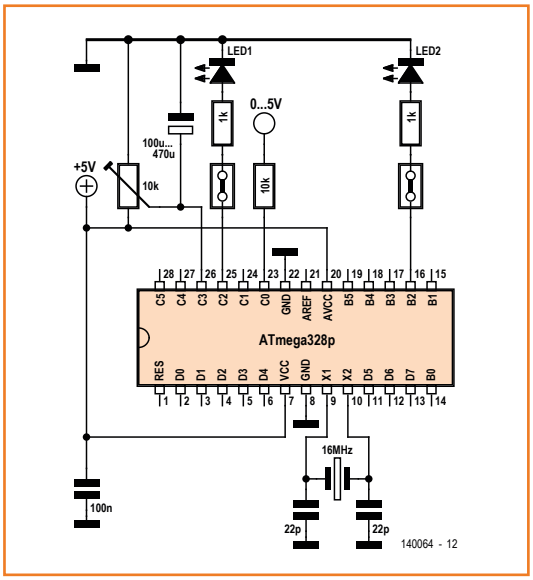


Figure 2.
Using the analog inputs.

that a perfectly ordinary LED can also act as a photodiode – in this case as a light sensor. If you shine a bright light on the LED, it can deliver a voltage as high as 1.5 V—three times as much as a silicon photodiode. That's because a larger band gap means a greater voltage, which applies equally well to the forward voltage of a diode and voltage it generates.
Another thing you might notice is that at low light levels the voltage measured on ADC2 is affected to a certain extent by the potentiometer voltage on ADC3. This is due to the sample-and-hold capacitor at the input of the A/D converter. This capacitor is first charged to the voltage present on the measurement input, and then its charge is measured. If the capacitor last saw 3 V from the potentiometer during the previous measurement, some of its charge will still be left because it cannot discharge quickly enough through the extremely high impedance of the LED. For this reason, the microcontroller data sheet recommends that the internal resistance of the signal source should not exceed 10 kΩ. Here we are dealing with many megohms instead, so it's better to set the potentiometer to zero. Then you will see that there is still a measurable voltage at low light levels. This circuit shows that the A/D converter has very high input impedance. A typical digital voltmeter with an input impedance of 10 MΩ would not be able to measure the LED voltage, but the ATmega can do so.

Listing 3: Measuring the LED voltage and the potentiometer voltage.

```
Do
  Locate 1 , 1
  Lcd "A2 "
  D = Getadc(2)          'LED1
  U = D * 5.0
  U = U / 1023
  Lcd U
  Lcd "    "

  Locate 2 , 1
  Lcd "A3 "
  D = Getadc(3)          'Pot
  U = D * 5.0
  U = U / 1023
  Lcd U
  Lcd "    "
  Waitms 1000
Loop
```

With very high impedance signal sources, you can connect a 10 nF bypass capacitor in parallel as shown in **Figure 3**. This makes the measurements much more reliable. Even with very dim light, you can now measure a voltage of roughly 1 V. For all PN junctions with their exponential characteristic curves, the rule of thumb is that the voltage rises by approximately 70 mV when the current is increased by a factor of 10. This means that the span between 1.0 V and 1.5 V corresponds to seven decades of current range, and thus seven decades of brightness. That should be more than enough for measuring from 10 lux to 100,000 lux, and we can see the makings of a light meter here. Maybe that's an attractive job for an enthusiastic reader? There's nothing wrong with a nice collection of Bascom programs with contributions from many people.

PWM outputs

When precise timing matters, timers come to the fore. Most microcontrollers have several timers. Timer1 of the ATmega328 has a resolution of 16 bits. You can regarded it as a sixteen-stage counter, similar to the well-known CD4040 (which has only twelve stages). A clock signal (or other pulse signal) entering at one end comes out at the other end with a lower frequency. You can use the CD4040 to build a clock generator or a frequency divider, or as the basis for a binary pulse counter. Timer1 can also handle all of these

functions; you just have to initialize it properly. There are also two 8-bit timers on board (Timer0 and Timer2).

One of the many operating modes of Timer1 is PWM mode. It can generate two pulse width modulated signals, which means rectangular pulse signals with adjustable mark/space (on/off) ratio. Among other things, they can be used to control the brightness of a lamp, much in the same way as a dimmer on the AC line which simply switches the voltage on and off at a high rate. Here we want to do the same thing by using the PWM output to control the brightness of an LED. The PWM output has a maximum adjustment range of 10 bits, but it can also be configured in 8-bit mode. Here 10 bits is an attractive choice because it matches the resolution of the A/D converter. In both cases the counter runs from 0 to

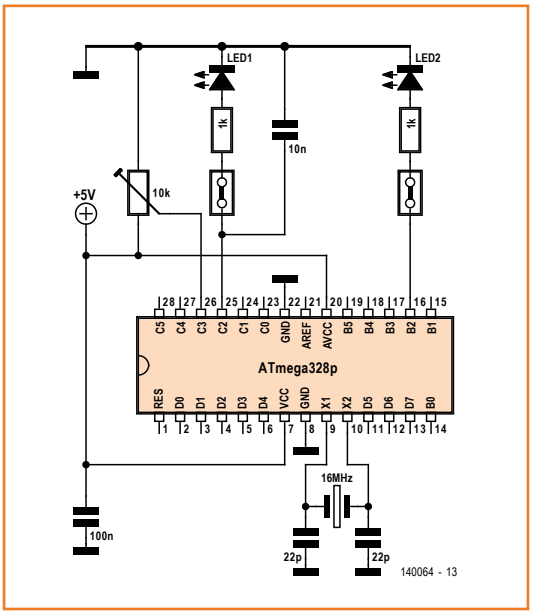


Figure 3.
Using an LED as a light sensor.

Listing 4: PWM control.

```
'--------------------------------------
'UNO_LCD3.BAS  PWM
'--------------------------------------
$regfile = "m328pdef.dat"
'ATmega328p
$crystal = 16000000            '16 MHz
$baud = 9600



S1 Alias Pinc.0
S2 Alias Pinc.1

Dim D As Word
Dim U As Single

Config Adc = Single , Prescaler = 64 ,
Reference = Avcc     '5 V

Config Timer1 = Pwm , Prescale = 1 ,
Pwm = 10 ,
Compare A Pwm = Clear Up , Compare B
Pwm = Clear Up

Config Lcdpin = Pin , Db4 = Portd.4 ,
Db5 = Portd.5 , Db6 = Portd.6 , Db7 =
Portd.7 , E = Portd.3 , Rs = Portd.2
Config Lcd = 16 * 2
Portc.0 = 1                    'Pullup
Portc.1 = 1                    'Pullup
Config Portb = Output
Cls
Cursor Off
Do
  Locate 1 , 1
  Lcd "PWMA="
  A = Getadc(3)
  Pwm1a = A
  Lcd A
  Lcd "    "

  If S1 = 0 Then
    If B > 0 Then B = B - 1
  End If
  If S2 = 0 Then
    If B < 1023 Then B = B + 1
  End If
  Pwm1b = B

  Locate 2 , 1
  Lcd "PWMB="
  Lcd B
  Lcd "    "
  Waitms 100
Loop
```

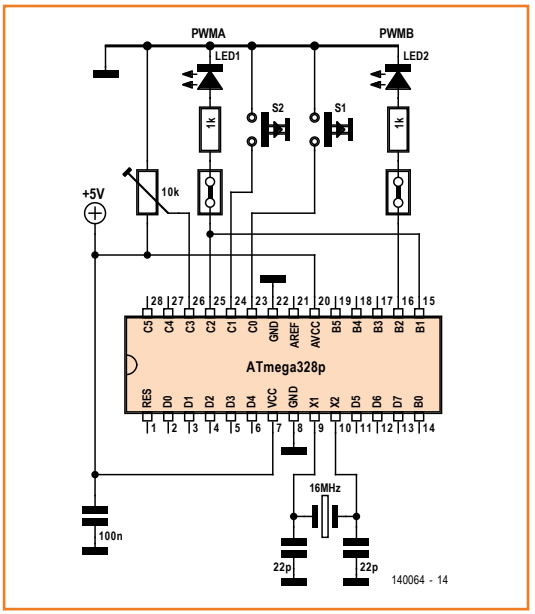


Figure 4.
A pair of LEDs connected to the PWM outputs.

1023. You could for example program the output to switch off when the counter reaches 511, which results in a PWM signal with a 50% duty factor. This is achieved by repeatedly comparing the counter value with the programmed value in a compare unit. There are two compare units in the microcontroller, so you can generate two independent PWM signals with the same timer. PWM1A is output on port pin PB1, while PWM1B is output on pin PB2. That reminds us of something. You guessed it: LED 2 is connected to PB2 through a 1 kΩ series resistor. Coincidence? Not at all, because now you can use the PWM signal to control the brightness of the LED.
The program in **Listing 4** uses the PWM1A output together with the A/D converter input ADC3, which is connected to the potentiometer. The measured value of the potentiometer setting is used as the comparison value for the PWM signal. If you set the potentiometer to mid-range, the PWM output is a symmetrical square wave. Everything from 0 to 1023 is possible. The actual output value is shown on the LCD. If you want to look at the signal, you can connect a scope probe to PB1 (Arduino pin 9).

What is the frequency of the PWM signal? That depends on the frequency of the clock input to the timer. It can be the clock signal from the Arduino board (16 MHz), but it can also be a lower-frequency signal derived from the clock signal by passing it through a prescaler. Here the initialization instruction `Prescale = 1` means that the full clock frequency is used. You might think that this yields a PWM frequency of 15.625 kHz (16 MHz divided by 1024), but in fact it is only half of this (approximately 7.8 kHz). This come from the fact that Bascom uses the "Phase Correct PWM" mode instead of the "Fast PWM" mode. You can learn more about this from the microcontroller data sheet, but that's more like a book than a data sheet. Here Bascom saves developers time and effort by always choosing a reasonable option for each setting. This means you can obtain working results with Bascom even if you don't fully understand all the details.

LED 2 is connected to the second PWM output (PB2). In our program this output is controlled by the two pushbuttons S1 and S2. To make the program easy to read, the `Alias` instruction is used to assign the button names to the port inputs. This means that when you write `If S1 = 0 Then` in your code, Bascom knows that what you actually mean is `If Pinc.1 = 0`, so the state of port pin PC1 has to be polled. Another important consideration is that the buttons are connected to ground and therefore require pullup resistors. As a result, the program normally sees the quiescent state S1 = 1. This only changes to S1 = 0 when someone presses the button. That makes things very simple. When you press S1 you reduce the PWM output level (assuming it is greater than 0), and when you press S2 you raise the PWM output level (assuming it is less than 1023). This proceeds in single steps if you press the button briefly, but if you press and hold the button it changes continuously at ten steps per second. The current output value is always shown on the LCD.

Button polling

The program shows a very simple example of how to use buttons and corresponding program branching. Of course, there are lots of other ways to obtain the desired result. Developing the ideal user interface for a given task is a fascinating job. You can configure a wide variety of things with two buttons and a potentiometer. A lot can be achieved with just a few buttons.
The program uses everything that the Elektor shield has to offer. The LCD shows the two current PWM values, the potentiometer controls the PWMA output, and the two buttons control the PWMB output and thereby LED 2. Is that really everything? Well, not quite—LED 1 isn't doing

anything. So let's plug a wire into PB1 (Arduino pin 9) and ADC2 (Arduino A2) to connect PWMA to LED 1 through a 1 kΩ resistor and a jumper (**Figure 4**). Now you can control the brightness of the LED with the potentiometer.
With two LEDs whose brightness can be set independently, you have the makings of a little skill-testing game. The first player sets the brightness of LED 2 to some arbitrary value using the buttons. Then this LED is covered, and the second player tries to set LED 1 to the same brightness using the potentiometer. The LCD reveals how well this went. Then the two players swap roles. All differences between the two values are recorded. The player with the lowest score at the end is the winner and is designated "Hawkeye".

(140064-I)

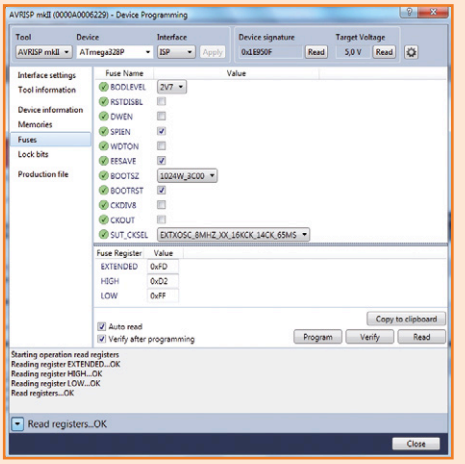
Web Links

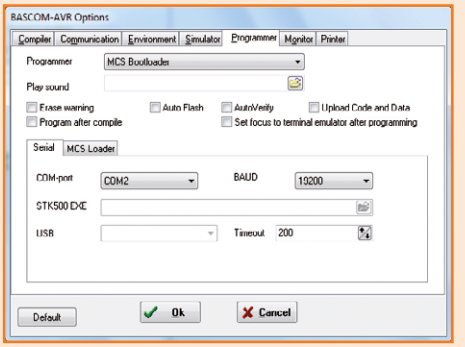
[1] www.elektor-magazine.com/140064
[2] http://b-kainka.de/bastel49.htm (in German)
[3] http://en.wikipedia.org/wiki/Polarizer

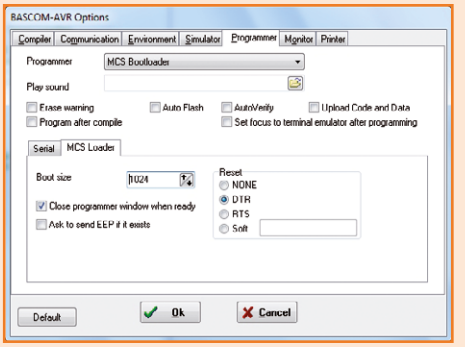
MCS Boot Loader

Up to now we have described two options for programming the Arduino: using the Arduino boot loader or using the ISP interface, which is also brought out on the Elektor shield. With an external programmer, you can also use the ISP interface to load a different boot loader.

Bascom has the MCS boot loader, which can be adapted to various microcontrollers. That's attractive because it allows you to equip different microcontroller boards with the same boot loader. The source code is included with the Bascom example programs. All that is necessary for adaptation is to configure the microcontroller type, the clock frequency and the desired baud rate. The configuration for the ATmega328P was not yet there, but the adaptation was easy. The fully adapted boot loader, including the source code and hex file, can be downloaded from the Elektor website [1]. The files are called BootLoaderUno_16.bas (16 stands for 16 MHz) and BootLoaderUno_16.hex. Note that the fuse settings must be slightly different (see the first AVR Studio screenshot) than for the Arduino boot loader because more memory space is needed. The boot area is 1024 words with the MSC boot loader and starts at &H3C00. If you intend to use the MSC boot loader, you must configure Bascom accordingly. Here again it is important to set the right COM port and the right baud rate, in this case 19,200 (see the second screenshot).



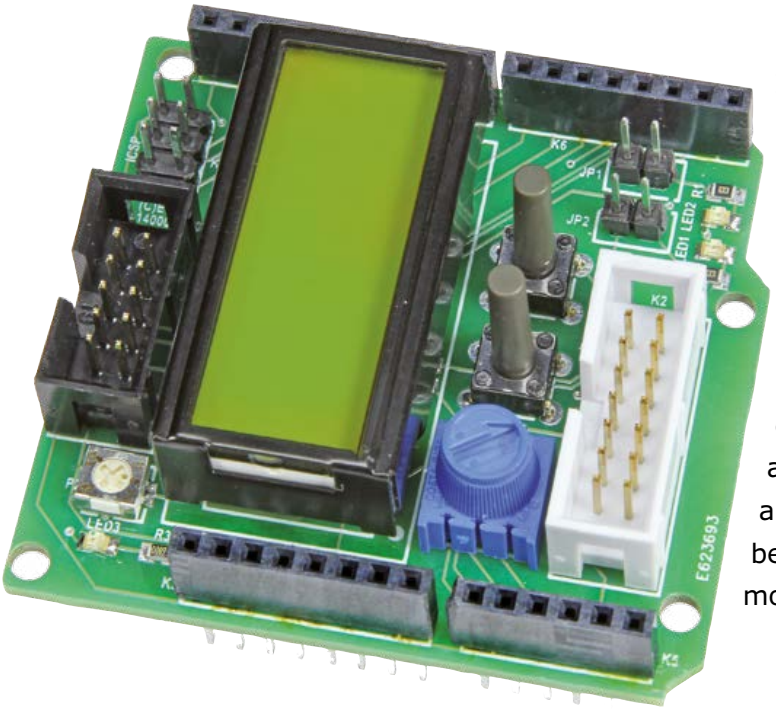




There is another important setting on the MCS Loader tab. As you may recall, data is transmitted to the Arduino board over the USB bus, but it arrives through a simulated RS232 port. All Arduino systems use the RS232 DTR line as a Reset line; a falling edge on this line triggers a reset. That's what starts the boot loader, because the microcontroller runs the program code stored at the boot address (&H3C00) when it starts up. This code waits for a specific action from the PC. Either new program comes down the pipe or it doesn't, and then execution branches to the start of program memory at address zero. This process occurs with every reset, which is what allows Bascom to force a jump to the boot loader by changing the level on the DTR line. Users have it easy because they don't have to press the reset button themselves.

My First Shield :-)

LEDs, buttons, display and more



In the microcontroller world as elsewhere, learning by doing is a good approach. The Arduino Uno, which features especially low cost and a low entry threshold, has a significant shortcoming: almost no on-board peripherals. We have therefore developed a compact shield that gives starters a text display, LEDs and pushbuttons to provide a good basis for their first projects. There are also two extension connectors for users who already have a bit of experience. They can be used to connect relay modules, wireless modules and many other devices.

By **Jens Nickel**
(Elektor Germany)

Like it or not, very low-cost, mass-produced microcontroller boards have changed the electronics landscape. If you want to or have to develop a working project or demo that does not need very much computing power, you can simply grab an Arduino Uno – and there's bound to be one lying around somewhere, just waiting to be used. Beginners in particular benefit from the free Arduino development environment, and all you need to download programs is a USB cable. If you prefer to program in Basic, you can also work with Bascom and the boot loader. Complex programs in C or C++ are also possible with the free AVR Studio environment and a small programmer, such as the AVR ISP mk2.

Intelligent shields

There aren't many peripheral devices on the Arduino Uno; a single LED is all you get from the board designers. You can also use the USB interface to access the microcontroller on the Uno board (an Atmel ATmega328P) over its UART port. However, most of the microcontroller pins are fed out to two rows of socket headers which provide the basis for the shield concept. If you equip another PCB with a matching set of pin headers, you can plug it directly into the microcontroller board. A shield can hold a wide variety of peripheral devices, including displays, sensors, or interfaces such as Bluetooth or WLAN. Thanks to the widespread use of the small Arduino Uno board, there are now hundreds of commercially available shields.

Getting started

Many of our readers are interested in learning how to work with microcontrollers and program in Bascom and/or C. For them, the Arduino Uno is a natural choice. To become truly familiar with programming a microcontroller, you need to come to grips with the key interfaces of the IC yourself at least once. The best way to do this, of course, is to work with peripheral devices that use these interfaces. Unfortunately, there is hardly any shield available that provides all the necessary peripherals in a small, low-cost package, so we

decided to develop our own shield. It contains a display module, two user LEDs, two pushbuttons and a potentiometer (**Figure 1**). That means you can get started right away after you plug in the shield. For larger projects the shield also has a pair of connectors which allow other extension boards to be connected over ribbon cables. A relay board and a wireless module are already available, for example.

The shield can be ordered from the Elektor Store [1] as item no. 140009-91. if you are a novice programmer and you want to try out or use this small board primarily for demo and example applications, you actually don't need to read any further in this article. Elsewhere in this issue we describe some initial small applications for our shield, written in Bascom Basic. They are also excellent for test purposes. For the C users (including would-be users) among our readers, we promise that C software will be coming soon, since we plan to use shield fairly often in future Elektor projects.

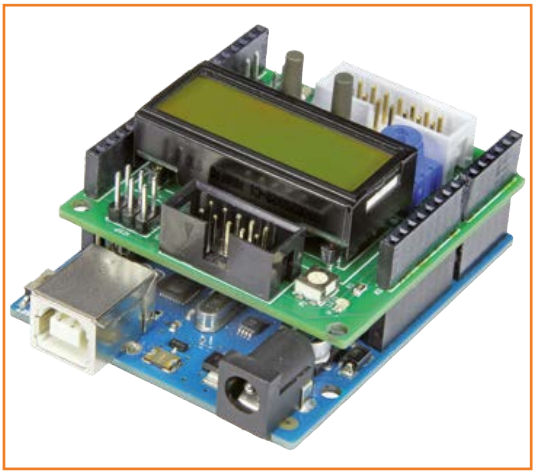


Figure 1.
The shield is shown here on an Arduino Uno board, but it can also be used with other 5-V Arduino boards.

Socket headers

The socket headers of the Arduino Uno are duplicated on the shield. There is an Arduino standard for the signals available on these headers, which must be adhered to regardless of the microcontroller fitted on the Arduino board concerned. The individual sockets are also called "Arduino pins". The Arduino pins and the corresponding pins of the ATmega328 microcontroller are listed in **Table 1**.

Due to the display and the buttons, it is not possible to plug another shield piggy-back onto the Elektor shield. However, all of the Arduino pins remain accessible when the Elektor shield is plugged in, and in most cases their operation is not affected by the connected shield, as you can see from the circuit diagram in **Figure 2**. Among other things, this applies to the analog inputs AD0 and AD1, the serial interface pins RX and TX, and the digital IO pins IO8 and IO9. Additional I/O pins can be freed up by pulling the jumpers (JP1 and JP2) and unplugging the display module.

LEDs, buttons and potentiometer

Two supply voltages are always generated on Arduino boards: 5 V and 3.3 V. They are available on a socket header and can be used by shields (see the "Voltages" inset). On our shield we connected a power indicator LED to the 3.3 V pin, so that you can see whether the Arduino board is powered up when the shield is plugged in. There are also two user LEDs on the shield, which can be driven by Arduino pins IO10 and AD2.

Table 1. Arduino pins, microcontroller pins and shield functions

Arduino Pin	ATmega328	Function
SCL	PC5	EEC-SCL (*)
SDA	PC4	EEC-SDA (*)
IO13/SCK	PB5	ISP-SCK
IO12/MISO	PB4	ISP-MISO
IO11/PWM/MOSI	PB3	ISP-MOSI
IO10/PWM	PB2	LED2
IO9/PWM	PB1	ECC-GPIOB
IO8	PB0	ECC-GPIOA
IO7	PD7	LCD-D7
IO6	PD6	LCD-D6
IO5	PD5	LCD-D5
IO4	PD4	LCD-D4
IO3	PD3	LCD-E
IO2	PD2	LCD-RS
IO1/TX	PD1	ECC-TX
IO0/RX	PD0	ECC-RX
AD0	PC0	S1
AD1	PC1	S2
AD2	PC2	LED1
AD3	PC3	P1
AD4	PC4	EEC-SDA
AD5	PC5	EEC-SCL

(*) Connected to AD4 and AD5 on the Arduino Uno

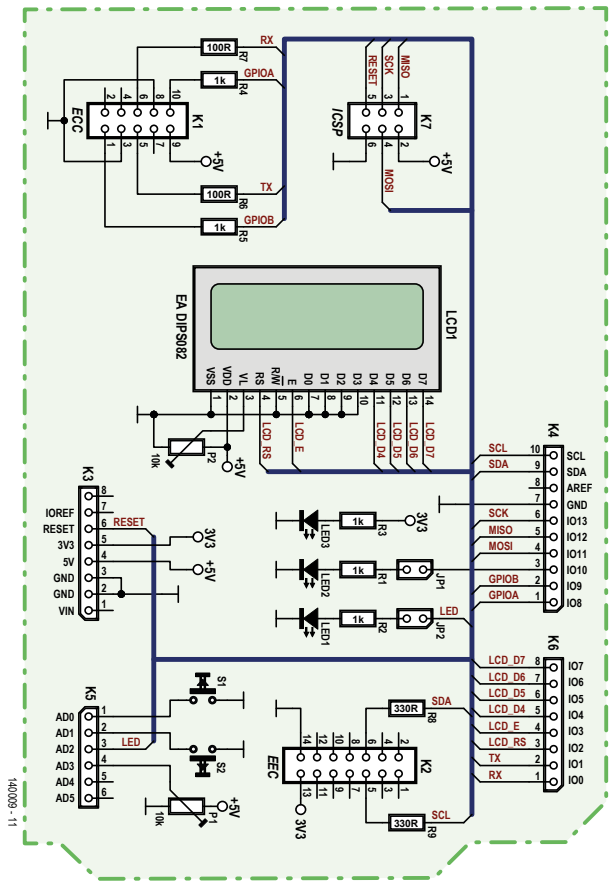


Figure 2. Circuit diagram of the shield. The Arduino signal lines are also available on socket headers on the shield.

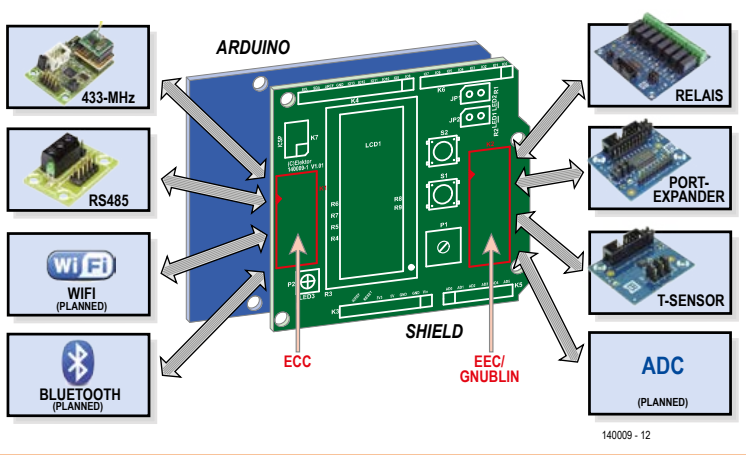


Figure 3. A wide variety of extension modules can be connected to the two extension connectors.

AD2 is actually intended to be used as an analog input, but on the Arduino Uno board it is connected to the microcontroller port pin PC2, which can easily be configured in software as a digital output. LED1 is connected to AD2 and can also be used as a photosensor, as described in the Microcontroller BootCamp article in this issue. According to the Arduino standard, pin IO10 must be able to output a PWM signal. This gives us the option of using LED2 to visualize the duty factor of that signal.

Two pushbuttons and a potentiometer are available for user input. They are connected to the AD0, AD1 and AD3 inputs. To allow AD0 and AD1 to still be used as analog inputs, we omitted pull-up resistors and debounce circuitry. However, it's very easy to connect internal pull-up resistors in the ATmega328, both in C and in Bascom. The potentiometer is connected to analog input AD3. If you configure the A/D converter of the ATmega328 with a 5 V reference voltage, which is common practice, the setting range is 1 to 1023 (10 bits).

Display

The alphanumeric display module from Electronic Assembly [2] is removable. It provides two rows of eight characters, which is sufficient for simple indications. Two buttons are located next to the display to enable simple menu control. We chose a display module with a 4-bit parallel interface, which means that four lines in addition to the E and RS signal lines must be routed to the display. This is a slight disadvantage compared to an SPI interface, but software support for the parallel interface is better. For instance, Bascom provides specific instructions for writing character strings to the display.
The display controller is of course HD44780 compatible, which allows popular C libraries to be used. The address offset for the second line is 40_{hex} [3].

UART connector

K1 is a fully wired Embedded Communication Connector (ECC), which has been described in detail in a previous article [4]. In accordance with the specification, the UART signals TX and RX as well as two digital I/O lines are available here, with a logic high level of 5 V. There is also 5 V constantly available on one pin for powering connected peripheral devices. In this case the

Component List

Resistors

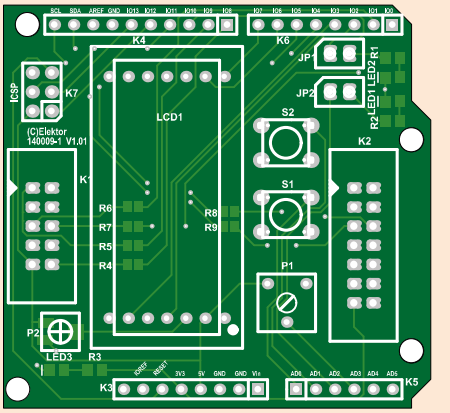
R1,R2,R3,R4,R5 = 1kΩ
R6,R7 = 100Ω
R8,R9 = 330Ω
P1 = 10kΩ trimpot with adjuster
P2 = 10kΩ trimpot, SMD, Vishay TS53YJ103MR10

Semiconductors

LED1,LED2 = LED, low-current, red (SMD 0805)
LED3 = LED, low-current, green (SMD 0805)

Miscellaneous

S1,S2 = pushbutton
K1 = 10-pin boxheader, 0.1" pitch
K2 = 14-pin boxheader, 0.1" pitch
K3,K4,K5,K6 = Arduino Shield stacking header, Adafruit ID 85
K7 = 6-pin (2x3) pinheader, 0.1" pitch
JP1,JP2 = 2-pin pinheader with jumper, 0.1" pitch
LCD1 = LCD 2x8 characters with background lighting, Electronic Assembly DIPS082-HNLED
2 pcs. 7-way precision socket strip for LCD1, TE Connectivity 1814655-7
PCB # 140009-1 [1]
or
Assembled board # 140009-91 [1]



peripheral device is a small module that gives user projects access to various interfaces to the outside world (see **Figure 3**). An RS485 module [4] and a wireless communication module have already been developed in the Elektor labs. The latter allows the Arduino board to transmit and receive data wirelessly at an ISM frequency like 433 MHz [5].

Voltages

Due to the countless shields now available, the layout and pin assignments of the Arduino Uno socket headers have become a sort of quasi-standard for extension connectors. More and more microcontroller boards, even with 32-bit processors, are being equipped with this interface. However, some of these microcontrollers require an operating voltage of 3.3 V instead of 5 V as on the Arduino Uno board, and their inputs and outputs are therefore compatible with 3.3 V signal levels.

Ideally, a shield can be plugged onto both 5-V and 3.3-V microcontroller boards and automatically adjust to the operating voltage of the microcontroller. That is why the Arduino socket headers always have not only 5 V and 3.3 V supply voltage pins, but also an IOREF pin that is connected to the supply rail for the microcontroller.

We originally planned to develop a shield suitable for all types of microcontroller boards, and furthermore equipped with ECC and EEC extension connectors compatible with both 5 V and 3.3 V peripheral devices. That would of course compel the use of several level converters, which would make the shield more complicated and more costly. However, we have not yet rejected this plan, and there are already several prototypes in the labs. We decided to design a less complex shield without level converters for the Microcontroller BootCamp series from Burkhard Kainka and relatively small projects. It is nevertheless possible to connect 3.3 V peripheral devices to the partially wired EEC connector, and the shield can even be used on a 3.3 V microcontroller board with some restrictions.

All signal pins of the ECC (TX, RX, GPIOA and GPIOB) are connected to the corresponding Arduino pins through resistors, based on a suggestion from Burkhard Kainka. This provides a bit of protection in the development environment, so that the Uno microcontroller will not die immediately if something a bit higher than 5 V is inadvertently connected to a pin.

EEC/Gnublin

Extension modules compliant with the Gnublin/EEC standard can be connected to K2, which is configured as an Embedded Extension Connector (EEC). This connector is not fully wired because the standard specifies a 3.3 V level and we omitted a level converter to keep the cost down (see the "Voltages" inset). Most of the Gnublin modules developed by the company Embedded Projects, which are also available in the Elektor Shop, are controlled using only the I²C lines SDA and SCL. One of the advantages of the I²C bus is that a master device operating at 5 V, such as the ATmega328 in this case, can easily be connected to a slave device operating at 3.3 V, such as a Gnublin module. Both lines are held at the high level by pull-up resistors, which in this case are located on the extension module, and pulled to the active low level by the master or the slave. Resistors R8 and R9 attenuate reflections on relatively long lines and provide some protection against noise pulses.
The Gnublin modules available in the Elektor Shop include a board with eight relays, a port extender with sixteen I/O lines, a temperature sensor and other devices [6].

Programming

The Elektor shield provides everything you need for experimenting with or learning about the essential function of the ATmega microcontroller or another 5 V microcontroller on some other Arduino board: digital outputs and inputs, analog inputs, a parallel interface (four bits), a serial UART interface and I²C. The only thing that's missing is SPI. The pins for this are fed out to connector K7, a 2x3 pin header. You can use this connector to program the ATmega328P on the Arduino Uno board, with the aid of a simple programmer such as the AVR ISP mk2. That will overwrite the Arduino boot loader, but it can be downloaded again using the Arduino IDE.

(140009-I)

Web Links

[1] www.elektor-magazine.com/140009
[2] www.lcd-module.de/eng/pdf/doma/dips082e.pdf
[3] www.datasheetcatalog.com/datasheets_pdf/S/T/7/0/ST7066.shtml
[4] Elektor March 2014, www.elektor-magazine.com/130155
[5] Elektor May 2014, www.elektor-magazine.com/130023
[6] Elektor Gnublin series http://www.elektor.com/development/gnublin

Simple Transistor Tester

With LED or Piezo Buzzer indicator

This handy transistor tester allows you to quickly test the function of an NPN/PNP transistor, JFET or (V)MOSFET and identify the correct leads.

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

A three-legged transistor or FET gives a total of six possible connection configurations—but only one will be correct. This small circuit provides a simple and unambiguous identification of the correct configuration and also makes a functional test of the device at the same time. The tester circuit itself contains a transistor which together with the transistor-under-test (TUT) forms a multivibrator circuit. The tester has five sockets close together identified with the labels:

E/S – B/G – C/D – E/S – B/G

This allows the following device to be tested using these configurations:

- Bipolar Transistors: EBC / BCE / CEB, and reversed: BEC / ECB / CBE.
- Unipolar Transistors (FETs): SGD / GDS / DSG, and reversed: GSD / SDG / DGS.

The multivibrator oscillates and flashes the high-efficiency LED (**Figure 1**) when the test transistor is correctly connected. The LED may also flash when the E and C leads are switched, but the flash rate will be faster. This reflects the fact that some types of transistor will operate with their emitter and collector leads switched although their performance characteristics will not be the same as a correctly configured device. When testing Junction-FETs (JFETs) with symmetrical source and drain construction it is only possible to identify the gate lead with any degree of confidence, the source and drain leads are interchangeable. The load resistance of the transistor-under-test is configured as a voltage divider network at half the supply voltage via resistors R3/R4. This allows a simple switch (S1) to change from N(PN) to P(NP).

A flashing LED indicates correct orientation; when the LED doesn't light or lights continuously the configuration is wrong or the device is FUBR or dead. Don't rule out the possibility that the device you are testing may not be a transistor, it could, for example, be a voltage regulator, thyristor or triac etc.

Version 2 of the tester shown in **Figure 2** uses a piezo buzzer in place of the LED indicator. The value of the frequency determining capacitor has been reduced compared to its value in **Figure 1** so that the oscillation frequency is audible. A low buzz indicates that the transistor is correctly configured and working, no sound indicates the test transistor is either wrongly configured or dead. The push button makes it easy to switch the unit on and test the transistor at the same time once it has been connected. The complete circuit will easily fit on a small piece of prototyping perfboard, power is supplied by a 9 V PP3 battery.

(130421)

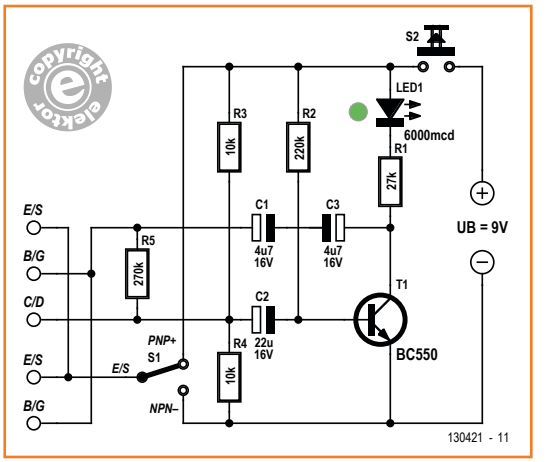


Figure 1.
Version using an LED.

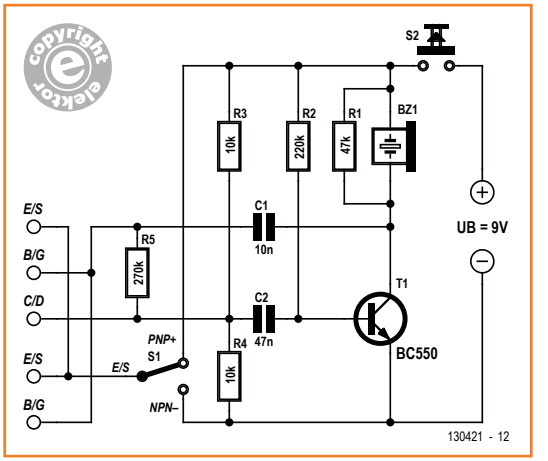
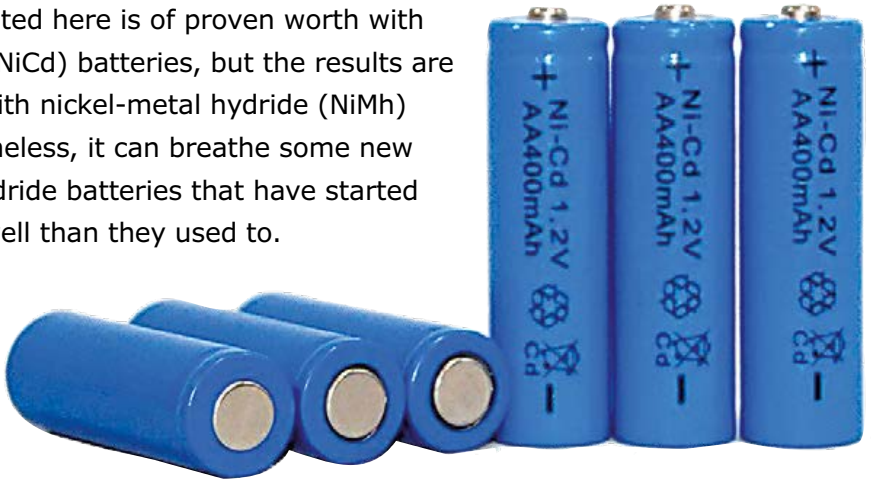


Figure 2.
Version using a Piezo buzzer.

Acupunctural NiCd Battery Conditioner

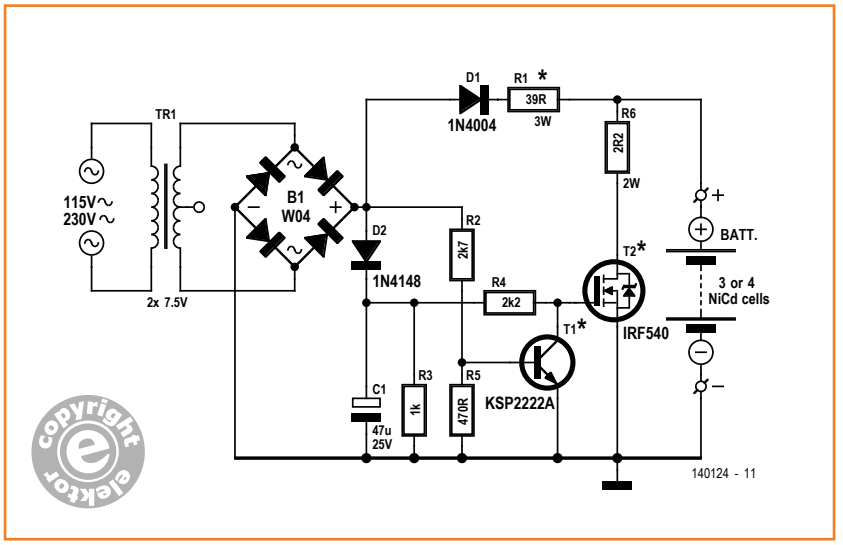
By **Ian Field** (UK)

The circuit presented here is of proven worth with nickel-cadmium (NiCd) batteries, but the results are less impressive with nickel-metal hydride (NiMh) batteries. Nevertheless, it can breathe some new life into metal-hydride batteries that have started performing less well than they used to.



Like the author, you may have a small and dwindling supply of AA NiCd cells that are needed for applications with intermittent high current draw. You too may have considered converting a salvaged SMPSU into a pulsed charger. This should work well with NiCd cells but not so with NiMh. In any case, modifying an SMPS is pretty tricky and possibly dangerous even if a suitable candidate can be acquired.

The basis for this project was the iron-cored power transformer salvaged from an old "dumb" NiCd charger. The transformer in question had a 7.5-0-7.5 center tapped secondary along with a two-diode full-wave rectifier to feed a row of current limiting resistors that charged up to four cells side by side.



Here a bridge rectifier is used to rectify the full 15 V available from the secondary and charge and condition the cells in series. The prototype was rigged for three AA cells in series but it should handle four without any component changes. Two cells could be accommodated but it would be advisable to increase current limiting resistor R1 to 56 Ω.

The clever bit of conditioning NiCd cells is to feed them charge current most of the time, but interspersed with needle-shaped pulses of discharge current. Apparently this shakes up the internal chemistry while the cell is charging—one description given online suggests that the electrode materials form smaller granules and thus exhibit higher energy density.

The circuit has been kept to the bare minimum—it would have been possible to diode-OR multiple MOSFETs from the collector of T1, but you will also need a battery holder that allows each cell to be wired independently in isolation from the others. Doing so will also require separate

charge current limiter and MOSFET current limiter resistors for each channel. Also, the resistors will have to be increased and/or a lower secondary voltage used.
The narrow discharge pulses are carried by the MOSFET (T2). In the prototype this was a surface mount type salvaged from a scrap board. Ratings of 70 amps and 20 or 30 volts are fairly usual. If this resource is not available an IRF540 can do the job—it may need a small heatsink though. The MOSFET is driven by T1, which serves as a zero crossing detector and inverter. Every time the rectified half-cycles dip to zero there is briefly no bias to keep the transistor saturated, at which point the MOSFET conducts hard forming a narrow discharge pulse to 'stimulate' the patients. An additional rectified supply—this time smoothed—feeds a voltage via R4 to pull the MOSFET gate up when the transistor is off. The 47-µF reservoir on the gate supply, C1, has a 1-kΩ bleeder resistor, R3, to quickly discharge it. Without C1, a power outage and no bias supply to saturate the transistor would keep the MOSFET in conduction and overheat R6 with current from the batteries.

Diode D1 prevents the battery voltage from interfering with the zero-crossing waveform or feeding the gate supply rail during power outages. The KSP2222A in position T1 is a manufacturer's designation for the PN2222A, which is the TO92 plastic version of the TO18 metal can 2N2222A. A BC337 would do also, or possibly the Japanese 2SC1815. The common 2SC945 and BC547 are other possibilities, but less powerful than the types previously mentioned and could run a bit warm.

(140124)

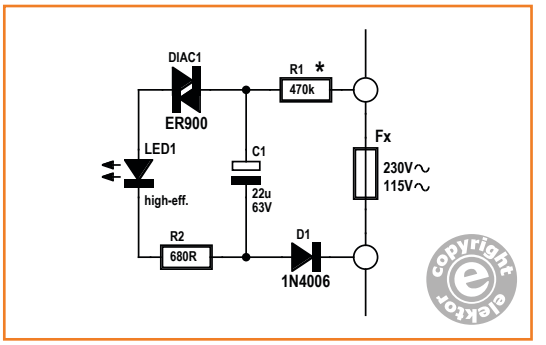
Blown Fuse Indicator

By
Hans-Norbert Gerbig
(Germany)

This simple six-component circuit flashes an LED to let you know that a fuse in some AC powered device has blown.

Its operating principle is quite straightforward: With the AC fuse intact the circuit will be short-circuited and inactive. When the fuse blows, power is supplied to the circuit and the 22-µF capacitor begins to charge up via the 470-kΩ resistor and the 1N4006 diode. All the time the capacitor voltage is below the DIAC trigger voltage the DIAC remains non-conducting. When the rising capacitor voltage reaches the DIAC's trigger point it conducts and discharges the capacitor through the LED and the 680-Ω series resistor which limits the LED current to a safe value. When the capacitor voltage drops below the DIAC trigger level it stops conducting, the LED extinguishes and the capacitor begins to charge up again and so the cycle repeats.

The complete circuit can be built on a small square of prototyping board. The finished module

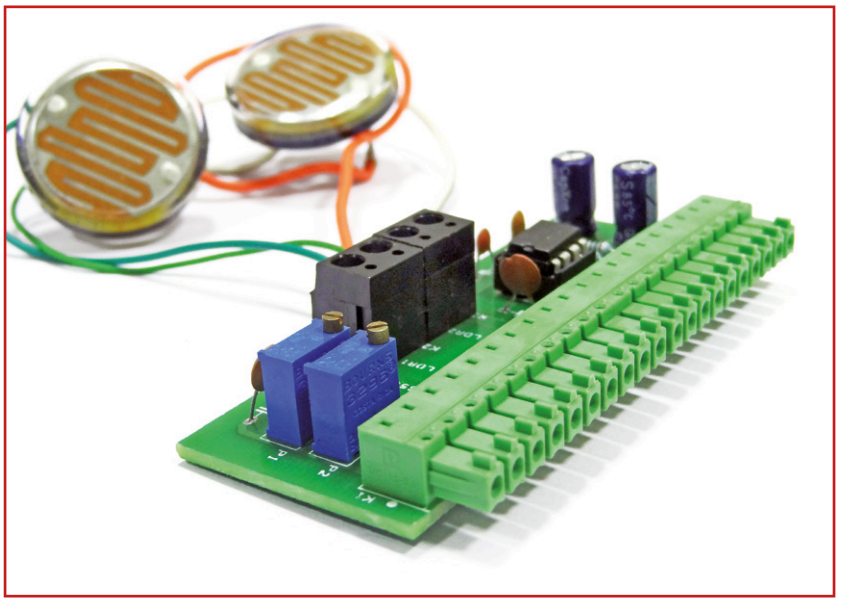


can then be wired in parallel to the AC fuse to be monitored. With the equipment switched on (assuming the fuse is intact) the circuit will be effectively short circuited and inactive. When the fuse blows the DIAC flash circuit becomes active, drawing current and the LED begins to flash.

The circuit is not galvanically isolated from the AC supply; ensure it's not possible to accidentally touch any part of the circuit when it's in use.

(130420)

Optical Theremin with myDAQ & LabView



By **Sunil Malekar** (Elektor Labs India)

National Instrument's myDAQ and LabVIEW are associated with austere applications in measurement and control by students and pros. Here the products get Elektorized meaning they get to work in a playful application.

The Theremin is one of the very first electronic musical instruments, dating back to the late 1920s [1]. It is played three-dimensionally by varying the position and speed of the player's hands between or above two antennae, one for pitch and one for volume (which interact to a degree). The Theremin is by no means an antiquated instrument from granddad's attic: *Muse*'s Matthew Bellamy mimics that eerie sound using the X-Y pad on his Manson M1D1 electric guitar and a ton of effects [2]; [5:02]. Rolling Stones' Brian Jones, Jean-Michel Jarre, and Led Zeppelin's Jimmy Page are known to have used the real instrument, not forgetting Léon Theremin himself who did a good job too with piano accompaniment [3]. The instrument you hear in Beach Boys' *Good Vibrations* is a Theremin derivate.
Since almost 100 years, the oscillator at the heart of the Theremin has taken many different implementations and technologies, all the way from from tube to DSP. In this project we connect such an oscillator to two products from National Instruments (NI): a MyDAQ unit for hardware [4] and LabVIEW for control, development and education.

Shopping list

In terms of hardware we require:

- MyDAQ with USB data cable;
- PC running LabVIEW software;
- Elektor Optical Theremin board;
- Active loudspeakers.

In terms of software, you need the driver software supporting myDAQ; it's called ELVISmx which uses LabVIEW-based software instruments to control the myDAQ device.

Meet myDAQ

National Instrument's 'myDAQ' product is a low-cost portable data acquisition (DAQ) device employing NI's famed LabVIEW-based software 'instruments', allowing measuring and analyzing of real-world signals. MyDAQ is good for exploring electronics in general and taking sensor measurements. Combined with NI's LabVIEW on the PC, we can analyze and process acquired signals as well as control simple processes anytime, anywhere. The myDAQ unit should be connected to a PC with a USB data cable which doubles as the power lead.

myDAQ provides a number of inputs and outputs as well as audio, power supplies, and digital multimeter (DMM) functionality in a compact USB device. The available I/O and supplies are our main concern here.

AI: two analog input channels that can be configured either as a general-purpose high-impedance differential voltage input or an audio input;

AO: as AI, but 2-mA max. differential voltage output or audio output;

DIO: eight digital inputs and outputs, each configurable as a general-purpose software-timed digital input or output, a special function input, or output for a digital counter.

Supply Voltages:–15 V, +15 V and +5 V.

The Elektor Optical Theremin add-on board

Our add-on board effectively interfaces LDR (light dependent resistor) sensors with the myDAQ device. One LDR on the board determines the frequency (*pitch*) and the other, the volume (*amplitude*) of the output signal with the help of LabVIEW and the 'Elektor Optical Theremin VI' software. VI stands for virtual instrument.

In the schematic of the add-on board pictured in **Figure 1**, connector K1 is used to link the board to the myDAQ unit. It carries the required signals and supply voltage for the board. With the help of dual opamp IC (an LM358) the change in resistance of the LDRs connected to K2 and K3 due to light falling on their surfaces can be used to generate an output signal that's variable in terms of frequency and volume. LDR1 on K2 is used to control the frequency and threshold value (adjustable with preset P1) whereas LDR2 defines the volume and associated threshold (adjustable with P2). The opamp outputs are connected to the analog inputs of the myDAQ device.

As the output is an audio signal, we need to connect active loudspeakers to the myDAQ Audio Out jack. Alternatively the "Elektor Optical Theremin Play Box" can be connected to the myDAQ Audio Out jack—that's assuming the Box contains a loudspeaker and LDRs, the latter arranged suitably to play the music.

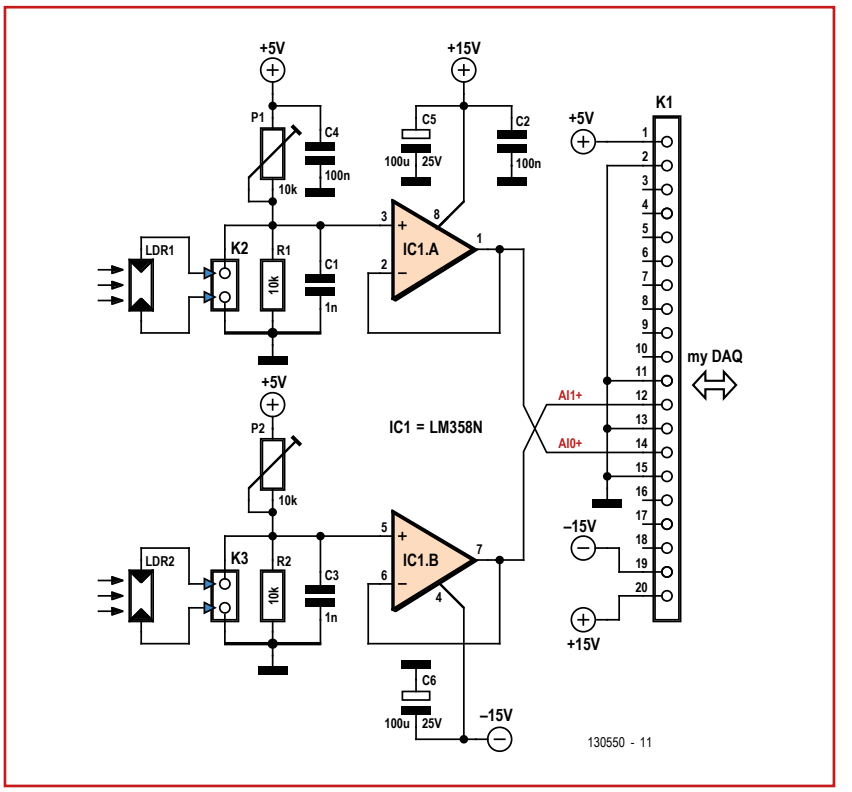


Figure 1.
Schematic of the human interface device effectively translating near-range hand movement into two analog signals controlling the pitch and amplitude of an audio oscillator. Not sure if Léon would have concurred.

A program for Optical Theremin

The application software called VI Front Panel provides two graphical representations to observe volume and frequency signal variations. The Frequency Meter block is used to observe the frequency variations, and the slider to display volume variations. The display window of the VI Front Panel is shown in **Figure 2**.
Analog inputs A1 and A2 of the MyDAQ device

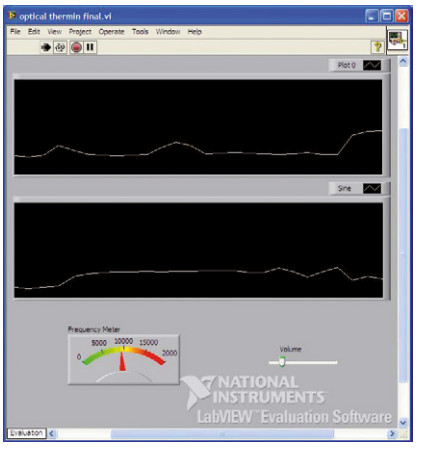


Figure 2.
VI Front Panel with frequency and amplitude readouts.

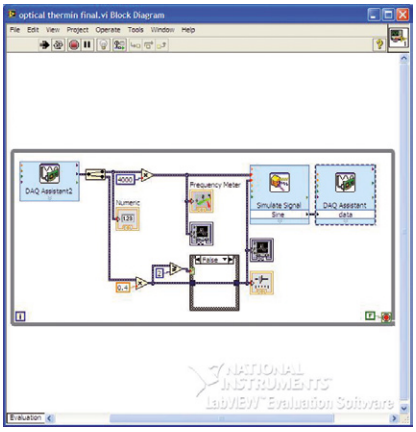


Figure 3.
VI Back Panel showing signal scaling block.

receive voltage signals varied by LDR1 and LDR2. In the *VI Back Panel*, DAQ Assistant 2 is used to introduce these two LDR-varied voltage signal into the VI. The analog signals vary between 0 to 5V. The tone is generated by the predefined MyDAQ library.

One of the analog inputs is used to manipulate the frequency of the tone to be generated. The maximum tone frequency that can be generated by myDAQ is 20 KHz and hence to scale the analog input value up to 20 kHz the signal value is multiplied by 4000. This signal is then fed to the Graph module as well as the Frequency Meter which is connected to the VI Front Panel.

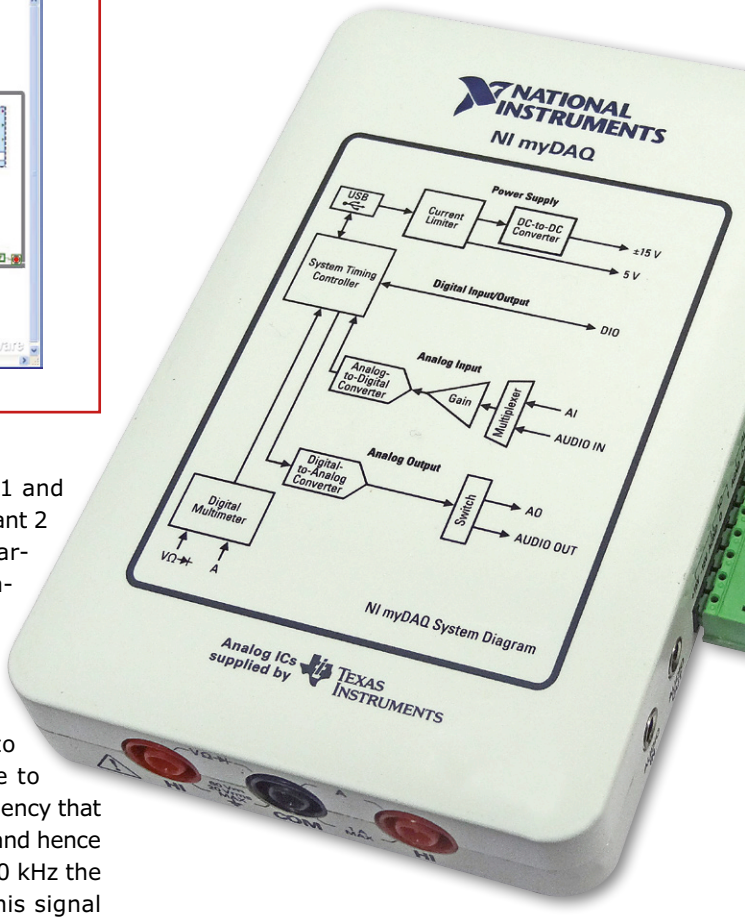


Figure 4.
Component layout of the extremely simple board to hook up to the myDAQ unit.

As the tone generator's maximum signal strength is 2 V_{pp} the analog input signal is matched and multiplied by 0.4. This signal is then fed to the Graph module and the bar meter which is connected to VI Front Panel. The VI signal scaling block is shown in **Figure 3**.

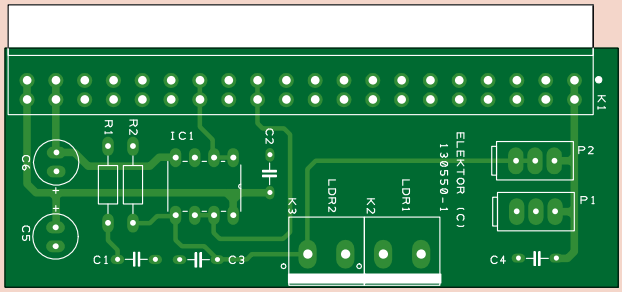
Component List

Resistors
R1,R2 = 10kΩ
P1,P2 = 10kΩ preset

Capacitors
C1,C3 = 1nF
C2,C4 = 100nF
C5,C6 = 100µF 25V radial

Semiconductors
IC1 = LM358N

Miscellaneous
K1 = 2-way terminal block header, right angled, 3.81mm pitch
K2,K3 = 2-way PCB terminal block
PCB # 130550



Start Theremin-in'

Assuming you have successfully built the board shown in **Figure 4**, and downloaded the project software from the Elektor Magazine website [5], you proceed as follows:

- connect the Elektor Optical Theremin add-on board to myDAQ;
- connect the speaker or Elektor Optical Theremin Play Box to myDAQ;
- connect the myDAQ unit to your PC via the USB data cable;
- run the VI file from the software folder;

- start the program by clicking on the PLAY button;
- tell your roommates there's free WiFi and pizza at *Omikron Kappa* dorms up the street;
- play with two hands over the LDRs to change the amount of light they 'see' to generate music or what you believe is music.

In case it helps with your Theremin practicing (headphones on, thank you) the software provides an instantaneous graphics rendering of your hand movements.

(130550)

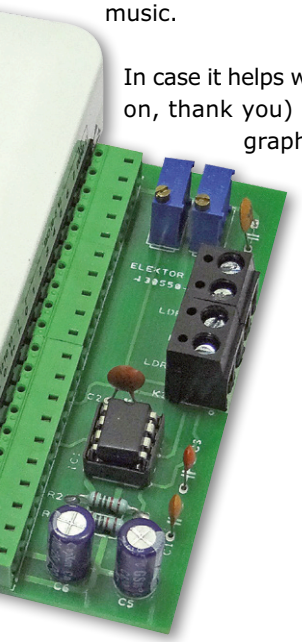


Figure 5. Top view of the Optical Theremin Box.

Web Links

[1] Theremin history: http://en.wikipedia.org/wiki/Theremin

[2] Muse: "Invincible" Live @ Wembley: http://youtu.be/t2KX0xBIEZI

[3] Léon Theremin playing: http://youtu.be/w5qf9O6c20o

[4] myDAQ and LabView products: www.elektor.com/ni-mydaq-dvd-labview-et-multisim

[5] Project software: www.elektor.com/130550

DesignSpark Tips & Tricks

Day #12: Using Buses

By **Neil Gruending**
(Canada)

Today let's look at how to use buses in your DesignSpark schematics. Last time we explored multiple page schematics and the cross probe tool using a more complex example in DesignSpark. We'll continue that theme today and learn how to use buses in our schematics.

Schematic Bus Types

Buses are very common in schematics because they are a great way to graphically group like signals together and make the design easier to understand. They are usually drawn as a wide line with named net connections branching to and from the bus like in **Figure 1**. The wide blue line is the bus and the lines that end on the bus are the connected nets named Data0 to Data5. Let's take a closer look at buses in DesignSpark and the two different types that it supports: open and closed.

An open bus doesn't keep track of connections—it's just a line in the schematic. Open bus net connectivity is specified using the net names that automatically appear when you connect it. **Figure 2** shows an example of an open bus. When I drew the connections from the resistors to the bus, DesignSpark automatically put the small 45-degree angle at the end of the connection and displayed the net name. If you delete the bus, all of the resistor connections will be deleted as well. You can change the net name by right clicking on the net and selecting "Change Net".

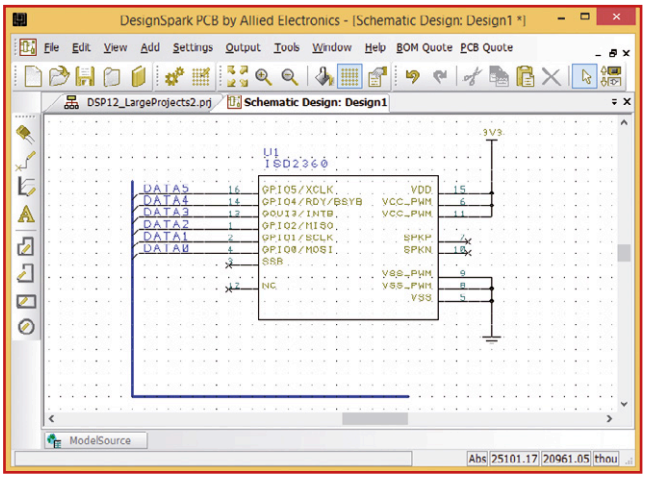


Figure 1.
Bus Example Design.

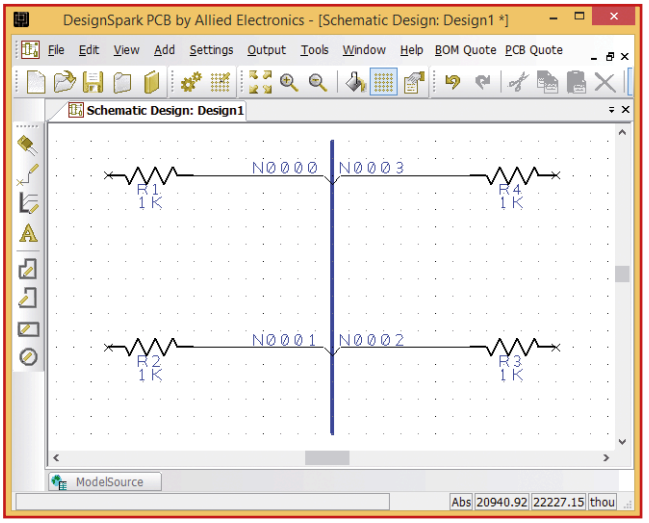


Figure 2.
Open bus.

A closed bus is more intelligent because it acts like a container for all of the bus nets and only those nets are allowed to connect to it. You change an open bus (the default) to a closed bus by right clicking on it and opening its properties window. There you can click on the Add Net button to open the Add Net to Bus window shown in **Figure 3**. A list of all of the nets in the design will be presented (N0001 to N0004 in this case) that you can then choose to add to the bus. If you want to add a new group of net names that end in a number like an address bus you can use the New Nets option. For example, if you wanted a 16-bit address bus you could set the name to A0 to A15 which are created when you click the "Add To List" button.

Open vs. Closed Buses

So which kind of bus should you use? It's a hard question to answer because

I think both types have their merits. I generally prefer to use closed buses for large buses like a data bus and open buses for short small buses like a SPI bus. But how DesignSpark treats them might also affect your decision.
Closed buses can be copied and they will keep all of the net associations which can be useful for large buses. But be careful when copying between different schematic pages because the nets in a bus follow the same rules as regular schematic nets so make sure you name them something different than the default Nxxxx. Also, when you paste the bus DesignSpark will ask you if you want to merge the nets because it will detect that the net names already exist in the design. Normally you would let DesignSpark merge the nets together.

Another advantage of closed buses in DesignSpark is that they can be probed using the new cross probe tool. When you probe a closed bus the cross probe tool will cycle through all of closed buses in the design that have the same nets associated with them. This is a great way to see where a bus is used in a design.

Open buses can also be copied but when you paste them all you get is their shape. You can also probe an open bus too, except that you're limited to probing each net individually instead of the entire bus.

Working with Buses

When you add a bus to a schematic it draws a solid line that you can then make connections to. DesignSpark automatically creates a 45-degree bus connection point to the bus when you add bus connections and you can control the size of the connection point line by using the Bus Defaults window in the Settings menu. It's also possible to change the direction of the connection line by clicking on it and using the F key (flip command).

One downside of buses is that it's very easy to miss a connection and make single node nets. DesignSpark doesn't have a schematic net check tool and the PCB design rule check (DRC) tool can't check for single node nets but it is possible to manually check your design. The easiest way to do this is to open the DesignSpark project file and run the "Generic Netlist" report from the Output → Reports menu. It's important to do this from the project file because if you run it from a schematic page the report will only include the nets from that page. **Figure 4** shows a report for a design with a closed bus with nets D0 to D3 and an open bus with A0 to A1 which are connected to several resistors R1 to R7. The report makes it easy to see that nets D2, D3 and A1 only have 1 connection. It's also easy to see that net D0 has 3 connections which may also indicate an error depending on the design.

Conclusion

DesignSpark's open and closed buses are easy to add to schematics and each as their strong points. The DesignSpark online help has even more details about how it implements buses and how to use them. Next time we'll look at some of the more advanced DesignSpark PCB features.

(140058)

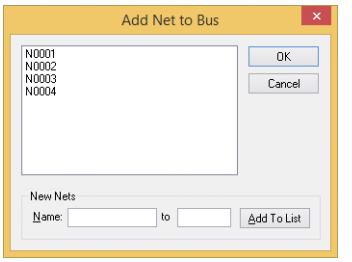


Figure 3.
Add net to bus window.

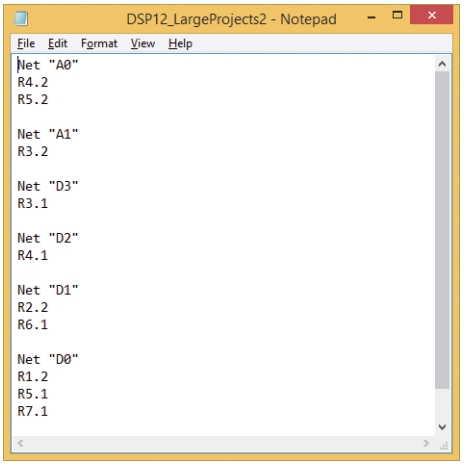


Figure 4.
Netlist report.

Tunnel Diodes
Weird Component #7

By **Neil Gruending**
(Canada)

I was looking for some inspiration around my lab this month and my Tektronix oscilloscopes caught my eye. I've always admired the older models and how they achieved such high performance with discrete components. One part that's always intrigued me was how Tektronix used tunnel diodes in some of their oscilloscope trigger circuits. They aren't common anymore but some new parts are still available if you look hard enough.

So why did Tektronix use tunnel diodes in oscilloscope trigger circuits? Probably because tunnel diodes switch quickly and can also act as a flip flop. For example a Tektronix 465 oscilloscope uses a couple of tunnel diodes to create a trigger pulse for the oscilloscope logic circuitry. One diode is used to arm the trigger pulse and the other is used to fire it [1]. The diodes are necessary to allow the circuit to have defined operating states for the trigger logic.

Figure 1 shows a tunnel diode in a simplified bias circuit. Tunnel diodes are made from a PN silicon junction with a very narrow depletion zone to maximize the tunnel effect. This means that the diode will start to conduct forward current with very little bias voltage. The forward current will increase and eventually peak (i_1) and then reach a minimum (i_2) as the bias voltage is increased like in **Figure 2**. This behavior gives the tunnel diode its negative resistance characteristic and lets it be used for an amplifier or oscillator. The peak current value is one the key characteristics when specifying them.

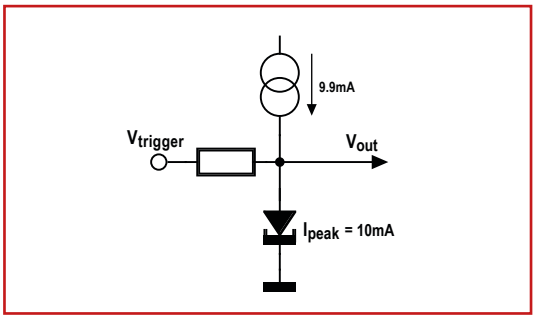


Figure 1.
Simplified tunnel diode bias circuit.

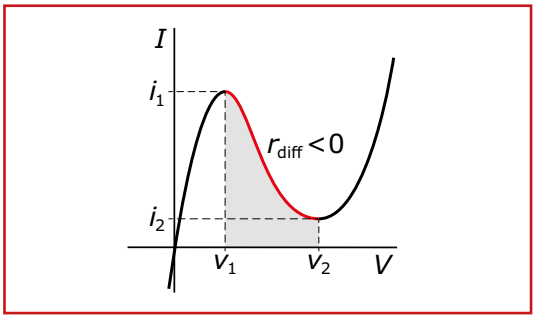


Figure 2.
Tunnel diode current characteristic.

Let's look **Figure 1** again. The 10-mA diode is biased with 9.9 mA of current which is right at the peak diode current when $V_{trigger}$ is high impendence. The diode is triggered into its minimum current state when $V_{trigger}$ injects enough current into it to push it past the peak current.

This triggering characteristic is also how you can test a tunnel diode. If you measure the resistance with a DMM and it reads a high value then the diode is defective. A low resistance reading means that it could be ok but to know for sure you need to verify that the diode will switch between its peak and minimum current values. In some critical circuits you may also have to verify that the values haven't drifted over time which is a common issue. If the values have changed, some circuit adjustment and calibration might be necessary.

Tunnel diodes are pretty interesting devices that met a real need in the 1960s and were heralded as the next big semiconductor discovery after the transistor. They're pretty hard to find now unless you are repairing vintage electronics but if you're interested in learning more about tunnel diodes and their uses, check out some of the Tektronix oscilloscope enthusiast websites [2].

(140057)

Web Links

[1] http://w140.com/mmm/tek-465-late.pdf

[2] http://w140.com/tekwiki/wiki/Tunnel_diodes

It's Tag Time!

Man has always wondered about the nature of time and how to measure it. Elektor.Labs too has its share of time-measuring projects. We tagged them all to save you time lookin' 'em up.

By **Clemens Valens**
(Elektor.Labs)

The problem of measuring time was solved in 1967 when the General Conference on Weights and Measures (CGPM; the acronym is French derived) redefined the second as being equal to 9,192,631,770 oscillations of a photon emitted by the transition of an electron between the two hyperfine levels of the ground state excitation of the cesium-133 atom—please excuse me if I made a mistake in the reproduction of this definition.
The CGPM could just as well have chosen 9 GHz or 10 GHz or another round value as the definition of the second, but it did not. The definition of the CGPM was chosen to be as close as possible to the historic value of the second. However, the last word on the second has not yet been said. Indeed, in 2012, physicists David Wineland and Serge Haroche were awarded the Nobel Prize for trapping individual ions, a feat that—experts say—is expected to change once again the definition of the second.

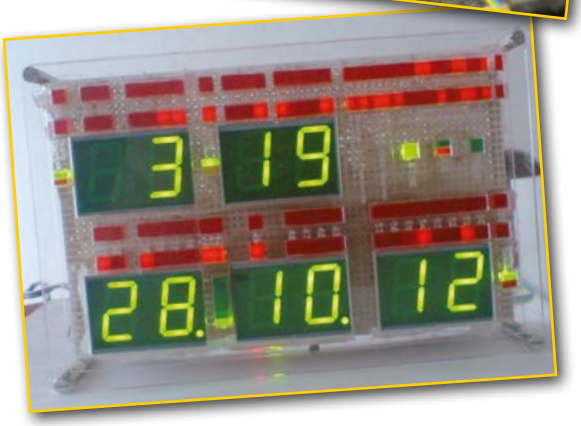
Clocks have always been a popular subject in Elektor, and at Elektor.Labs too several people have posted projects for measuring time. These projects can be subdivided into a few categories:

Pretty Objects: in this category we find clocks that were built to look good. Here we find Nixie tube designs, the Propeller Clock and other creative time display circuits.

Precision Instruments: some people want to have a clock with millisecond accuracy or with long-term accuracy. These are the GPS- and radio-based (DCF77, France Inter; WWF; MSF) clocks but also the designs using real-time clock (RTC) chips.

Programming Projects: in case you want to study how to program an FPGA or microcontroller, or even a PC, you need a subject to grind your teeth on. Time is such a subject because it ranges from easy (counting seconds) to complicated (perpetual calendars).

Add-on Modules: small projects that mostly show how to use an RTC chip allowing you to integrate it into another project that needs to know the time.



So now that you know that all these projects are out there waiting for you, how do you find them? This is where tags come in. Of course you can do a site search, but you can also use the tag cloud at Elektor.Labs. Click on the link "more tags" to view them all. Then click a tag that interests you and get rewarded with a list of projects related to the tag. As an example, find all the clock projects at [1].

(140060)

Web Link

[1] www.elektor-labs.com/tags/time

Arduino is a Tool

By **Clemens Valens**
(Elektor.Labs)

Barring you just woke up from a 10-year long coma you will have heard about Arduino. Everybody is talking about Arduino and all & sundry are developing programs (“sketches”) and extension boards (“shields”) for Arduino. We have filled scores of pages in Elektor about, or on using Arduino. Here is another one.

This article however is slightly different because it is not about an Arduino project but about using Arduino as a tool. I use Arduino quite often because it is so easy to program and use. I sometimes even use it as a calculator because it is quicker to write an Arduino program (called a ‘sketch’) than a PC application.

To give you an example, the other day at Elektor. Labs we had to test our prototype of the MIDI Channel Analyzer [1]. Hedwig our secretary had brought a nice keyboard with MIDI capabilities from home, but, for some reason, it wouldn’t allow us to test the analyzer properly. We had run into the chicken-and-egg paradox where we needed a MIDI analyzer to figure out the MIDI behavior of this keyboard to test the MIDI analyzer. So we called Arduino to the rescue. It took me less than five minutes to write a sketch that sent a Note-On and Note-Off message every second on a different, incrementing, MIDI channel. Elektor. Labs veteran Ton Giesberts added a transistor to the Arduino board to make the serial port MIDI compatible (the Data signal has to be inverted) and connected it to the device-under-test (DUT). We could have done without the transistor if we had used a software serial port on the Arduino, but it would have meant more testing.

With the tester ready all we had to do was power up everything, lean back and watch everything work just fine. This to me shows the real power of Arduino: an easy-to-use multi-purpose tool that can help you out in many prototyping and test situations.
So, if you are still wondering what all this Arduino fuzz is about, try to remember this article the next time you can’t figure out how to test your almost-completed-but-not-working yet super-duper ultra gadget.

(140059)

Web Link

[1] www.elektor-labs.com/node/3380

500-ppm LCR Meter: Feedback

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

The interest Elektor readers have shown in the 500-ppm LCR Meter published in March, April, and May 2013 [1], with a follow-up in November of that same year, was a pleasant surprise, taken as encouragement by the whole team—it's an unmistakable sign when such an advanced project generates so much enthusiasm. This publication won the confidence of hundreds of you who have successfully built the instrument. Let's just remind everyone that the LCR Meter is still available in the form of an assembled, tested module [3].

We'll never be able to praise highly enough the benefits of the direct interaction in the Elektor forum [2] between the author of this project and his readers, whether they have built the circuit themselves or are still thinking about doing so. These fruitful, multilingual exchanges have been the occasion for the author to complete the information given in the article and to correct it where necessary, and even to improve his software.

We can't recommend too highly reading these exchanges, especially in the *Test & Measurement* sub forum, all topics containing "LCR Meter" or similar in the title. The aim of this article is to bring together the key information from there.

Port selection

The port chosen in the Port/Select port… menu is saved in the preferences file (AU2011 X.Y.Z.ini). With the original version of the AU2011 software (version 2.1.2), once the port had been selected and was open, the Port/Select port… menu is disabled.

If you choose the wrong port (which can happen if several devices are connected to the same computer), the program opens the port OK but then won't be able to communicate with a device that is not the LCR meter—for a very good reason. But you are then not able to select another port. The only way to do so was to destroy this preferences file. A radical solution—but not ideal!

As from version 3.0.0, the addition of a Port/Close the port… menu lets you close the wrong port so as to select another.

No connection to port COMx

The rather unfriendly "No connection to port COMx" message can appear even though the connection is good and the port number is correct, if a second instance of the AU2011 program is run without closing the first one! It is impossible to connect to a port that is already busy.

Operation with USB supply

A few people have had a problem with the LCR meter in 'stand-alone' mode, powered using a USB supply. The majority of these supplies have the D+ and D− lines either biased at a voltage of around 2–3 V, or else simply connected together. For the LCR Meter's FT232R IC (U19) to be able to enable the regulator U15, its $\overline{\text{SLEEP}}$ output must be at logic 1. This is not the case if these two lines are floating. In this (rare) case, they need to be connected together (to the power rail)—or change the type of power supply!

Stand-alone and PC mode

To use the LCR Meter in stand-alone mode but powered via a USB cable connected to a PC, don't forget to press and hold one of the display module keys before powering up using SW1. Otherwise the LCR meter starts up in PC mode; it will briefly display the message "Bootloader v2.1" and then nothing!

ESR (Equivalent Series Resistance)

The LCR meter measures the in-phase and quadrature components of the DUT, i.e. the series resistance Rs and the series reactance X_s. The other parameters (Z, phi, R_p, X_p, Q, D, etc.) are then all deduced by calculation.
In the case of a high-value capacitor (from a few tens to thousands of µF), the series resistance Rs does indeed correspond to its ESR—just so long as the measurement is carried out at a frequency very much lower than its resonant frequency (because the series inductance of such a capacitor is not negligible). For these very high values, the measurement needs to be made at 1 kHz or even 100/120 Hz.

Inconsistency of the measurement in the extreme ranges

A very few readers have told us about a strange, non-reproducible phenomenon when measuring high value resistances (50 MΩ and above) with a value practically doubled @ 10 kHz, or low value capacitors (like 100 pF) with severe instability of this value @ 100 Hz.

This is caused by a slight instability in the U16 (LT1611) regulation loop, and depends solely on the particular specimen of LT1611 used.
The normal signal at output 1 of U16 (frequency around 1.8 MHz) appears in **Figure 1a**. **Figure 1b** shows the same signal with the instability present.

When this occurs, the oscilloscope will show a high level of noise at U6 output (TP6), which interferes with the measurement.
One solution is simply to replace the LT1611, in the hopes of coming across a less lively specimen! The real solution is to insert an RC network (39 kΩ / 1 µF) into the link between U16 pins 5 (V_{in}) and 4 ($\overline{\text{SHDN}}$) (**Figure 2a**) so that U16 performs a soft start.

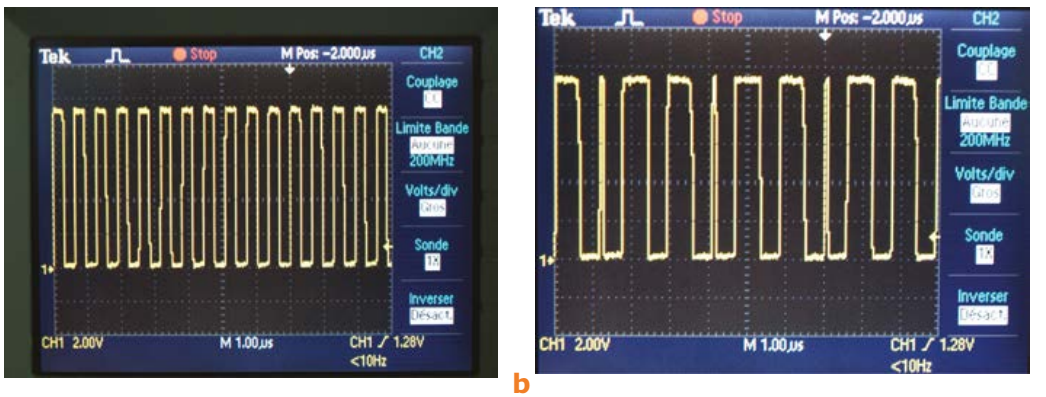


Figure 1.
The spread in certain parameters of the LT1611 (gain of built-in error amplifier, temperature, etc.) can cause instability (1b) that interferes with the measurements.

It's easy to modify the board (**Figure 2b**) as follows:

- cut the track between U16 pins 4 and 5, as well as the track between pin 4 and V_{in};
- reconnect pin 5 to V_{in}, at C67;
- solder a 39-kΩ resistor (0603 or 0805) between pin 4 and V_{in};
- solder a 1-µF capacitor (0805) to pin 4 and connect its other terminal to the ground side of C67.

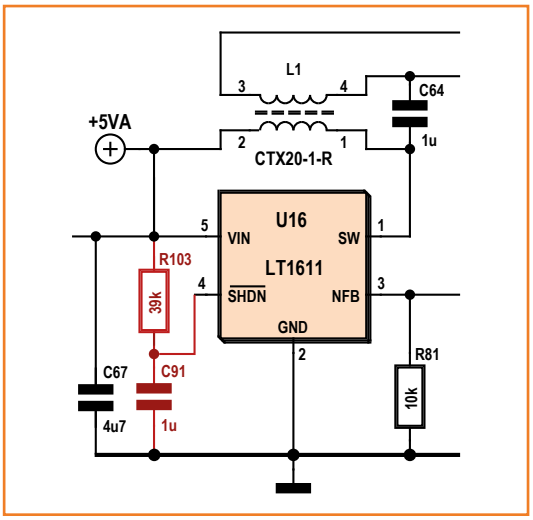


Figure 2a.
The most reliable solution is to delay the starting of LT1611 using an RC network (soft start).

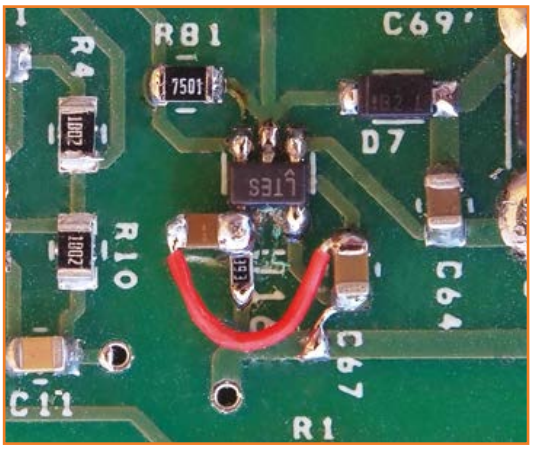


Figure 2b.
The modification to the board to add the RC network in Figure 2a is only needed if you observe the noise seen in Figure 1b.

Measuring RF inductors

A few readers have been disappointed by the results when measuring inductors with values below 200 nH (SMD RF inductors).

At the frequency of 10 kHz, an ideal 100 nH inductor has a reactance of 6.28 mΩ.
A real inductor also has a series resistance of several tens of milli-ohms. So the quality factor *Q* of this inductor is very much lower than 1 and the series resistance becomes the main parameter, the reactance being only the secondary parameter with a much greater inaccuracy!

What's more, using measuring cables, even short ones, does not allow you to perform an effective SHORT/TRIM; you only have to move one cable ever so slightly to see the measured value change by several (tens of) nanohenries.

(140069)

Web Links

[1] 500 ppm LCR Meter Part 1, Elektor March 2013: www.elektor-magazine.com/110758
Part 2: Elektor April 2013, www.elektor-magazine.com/130022
Part 3: Elektor, Elektor May 2013, www.elektor-magazine.com/130093
Important Update: Elektor November 2013, www.elektor-magazine.com/130307

[2] Elektor Forum: http://forum.elektor.com/viewforum.php?f=1543743

[3] Assembled LCR Meter module: www.elektor.com/500-ppm-lcr-meter-main-board
LCR Meter modules in stand-alone version (with display):
www.elektor.com/500-ppm-lcr-meter-kit-with-main-board-and-lcd-board

elektor PCB Service
powered by Eurocircuits

PCBs
Multilayer
Prototypes
Small series

Professional Quality
Trusted Service
Secure Ordering

Elektor PCB Service at a glance:

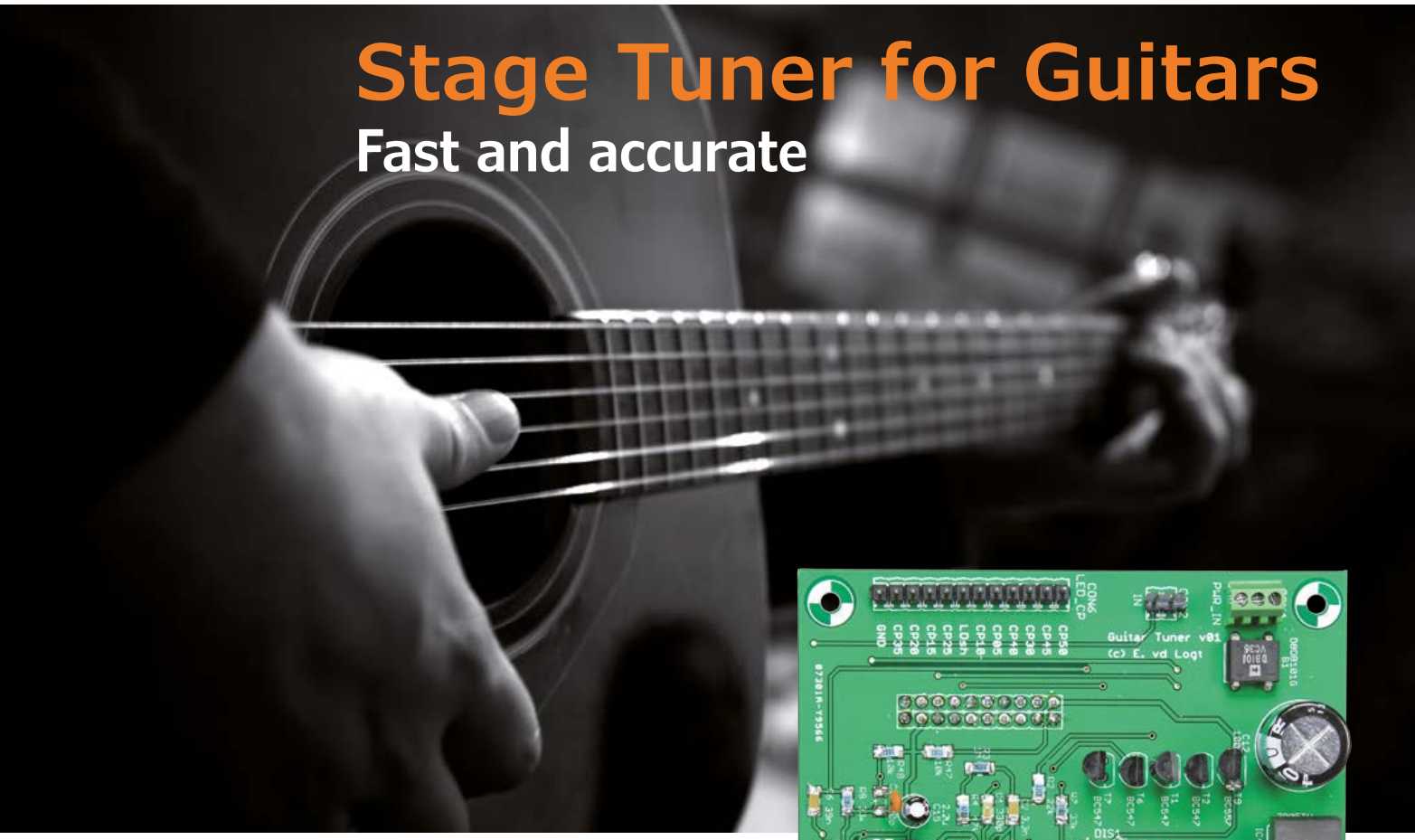
- 4 Targeted pooling services and 1 non-pooling service
- Free online PCB data verification service
- Online price calculator available
- No minimum order value
- No film charges or start-up charges

Delivery from 2 working days

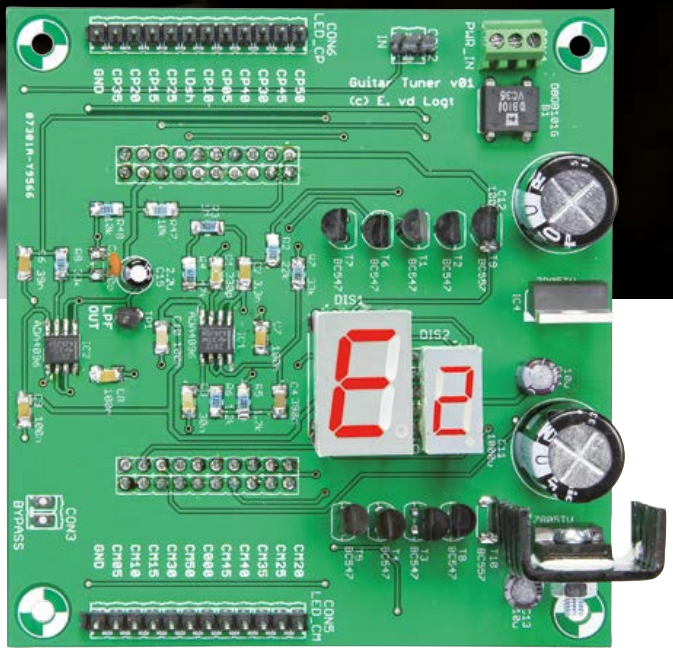
Further information and ordering at www.elektorpcbservice.com

Stage Tuner for Guitars

Fast and accurate



By **Emile van de Logt**
(Netherlands)



The final PCB differs from the prototype shown here in several places.

Although you can buy a professional stage tuner for your guitar, designing and building your own device is a lot more satisfying. At first I thought it would be fairly easy to determine the fundamental frequency of a guitar signal by using a fast Fourier transform (FFT), but in practice things turned to be a lot more difficult. This project uses a fast Cortex ARM M4 processor and a special frequency analysis algorithm based on autocorrelation.

For some time, the author of this article had been looking for a fast and accurate way to tune his guitars. Although there are 'stomp box' pedal devices for this purpose, the good ones are expensive and not always easy to read. After purchasing a professional guitar stage tuner, the author quickly started thinking about building his own. After all, he thought, determining the fundamental frequency of a guitar note shouldn't be all that difficult, and displaying the deviation from the desired frequency with a few LEDs should be pretty straightforward.

The ideal started to take on concrete form when the author received a TM4C123GXL Launchpad Evaluation Board (**Figure 1**) from a colleague. It contains an ARM Cortex M4 microcontroller, which has DSP capabilities and a lot of I/O pins on board. That makes it a powerful board at a very reasonable price—just $13 from TI, plus shipping charges. Ideal for reading in a guitar signal and processing it.

In this article we first have a look at the TM4C-123GXL board to give you a good idea of what it can do. As with every design, the design approaches and design choices are important—in particular the approaches that lead to something that doesn't work, because they provide insight into what you should do to arrive at something that does work. That's why we want to briefly examine the first design using an FFT, before moving on to the ultimate implementation.

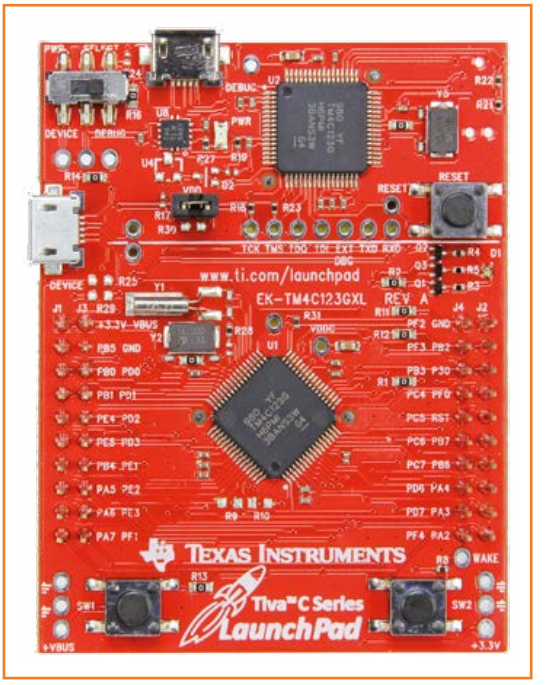


Figure 1.
The TM4C123GXL board from Texas Instruments, which is fitted with an ARM Cortex M4 processor. (photo: TI)

TM4C123G Launchpad Evaluation Kit

The manufacturer of this board, Texas Instruments, is known for their good documentation and support material that helps you get started quickly. The support resources for this board consist of a wiki with a workbook [2], presentations and even video clips. The integrated development environment (IDE) is Code Composer Studio (CCS), which truly includes everything you might want. All you have to do is follow the instructions in the workbook, connect the TM4C123GXL board to your PC through a USB port, and start designing. Following the presentations is a real pleasure. They give clear explanations and are supported by videos and example projects. The board even comes with a complete application programming interface (API) called Tiva, which you can use to control all of the I/O pins. For a student, engineer or hobbyist, this is the ideal way to start working with this advanced family of microcontrollers.

The evaluation board is populated with a TM4C-123GH6PM ARM Cortex M4 microcontroller with a maximum clock frequency of 80 MHz, a floating point unit (FPU) with DSP capabilities, 256 KB flash memory, 32 KB SRAM, 2 KB EEPROM, eight UARTs, four versatile serial interfaces, four I²C modules, CAN, USB, two 12-bit A/D-converters and a whole lot of I/O ports. In particular the 12-bit A/D converters, the generous I/O and the DSP capabilities make this board very suitable for the objective of implementing a professional guitar tuner.

First approach: fast Fourier transform (FFT)

Rendered overconfident by the impressive specs of the evaluation board, the author started working on an FFT implementation. A variety of ready-

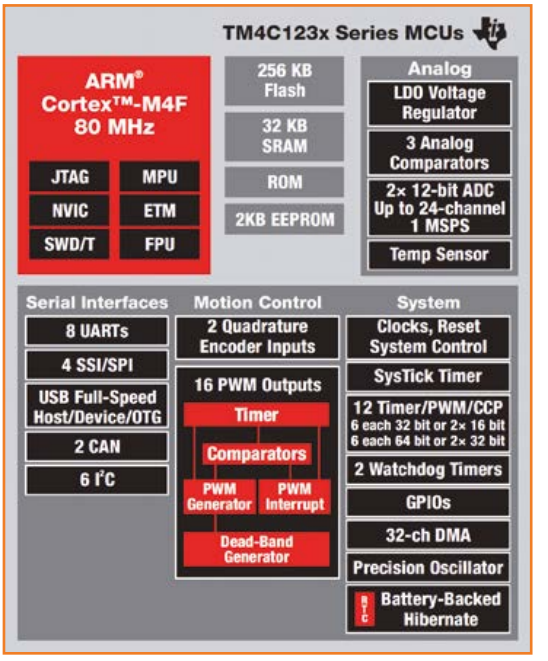


Figure 2.
High-level block diagram of all functions and connections on the TM4C123G board. (illustration: TI)

How Yin works: autocorrelation and parabolic interpretation

Instead of the fast Fourier transform, the Yin algorithm is based on another method that is often used in digital signal processing: autocorrelation (see http://en.wikipedia.org/wiki/Autocorrelation). An autocorrelation is the sum of the point-by-point multiplication of all samples of a signal with the samples of an offset version of the same signal.

Suppose the input signal is white noise with sample values in the range of −1 to +1. When these values are multiplied by other sample values offset in time, the products will also lie in the range of −1 to +1. If all these products are added up (and assuming the number of values is sufficiently large), the sum will be close to zero because the individual values tend to cancel each other out.
The situation is different with a non-random signal, such as a guitar signal. If the same operation is performed on a guitar signal, the resulting value will typically be greater than zero and will depend on the offset. The interesting point is that the maximum value is obtained precisely at the point where the offset signal is exactly aligned with the original signal, which means that it has been shifted by one period.

The Yin algorithm utilizes this effect. The signal is shifted in increments of the sampling period (here 1/6000 second) and correlated with itself. Then the algorithm looks for the offset value that yields the largest correlation result. This is a measure for the most significant frequency in the signal. The Yin algorithm also applies several refinements to increase the accuracy, such as parabolic interpolation. Suppose the Yin algorithm finds that the autocorrelation is greatest with a signal that has been shifted by 16 samples. This corresponds to a signal of 6000/16 = 375 Hz. The next frequency that also yields this result is 6000/15 = 400 Hz (these frequencies always correspond to multiples of the sample period). The desired frequency therefore lies somewhere between these two values. The optimal frequency is found by parabolic interpolation between these two points, which dramatically improves the accuracy.

made routines for this can be found. In the course of implementing these routines, the author soon ran into difficulties. Let's do some simple calculations: The spectrum of guitar signals is roughly 2 kHz. According to Shannon [3], this means that you have to use a sampling frequency of at least 4 kHz. If you then want to implement an FFT with a resolution of 0.5 Hz, you need an 8192-point FFT. Although the TM4C123G should be fast enough to handle this, the necessary amount of memory would quickly approach the limit of 32 MB available on the board. With a bit of cutting and trying, it ultimately proved possible to achieve a sampling rate of 1 kHz (for a usable spectrum 500 Hz) and fit a 2048-point FFT into the available memory.
However, the rest of the software still had to be written. It was clear that this approach would not lead to the desired result.

The second approach: autocorrelation and pitch detection

While searching the literature for a suitable way to determine the frequency components of a signal, the author quickly found a scientific article [4] describing how a autocorrelation function [5] could be used to accurately determine the fundamental frequency of a signal. The authors of this article underpinned their method for determining the fundamental frequency of a signal, which they called the Yin algorithm, with suitable mathematics. That was exactly what we needed. For more details about this algorithm, see the inset.

Based on this article, Joren Six wrote an C++ library [6], which Ashok Fernandez converted into an excellent C library [7]. This is a nice example of how a scientific article can lead to a specific practical application. The C library (Yin.h and Yin.c) was imported into this project virtually unchanged and worked immediately, which further testifies to the high quality of the library.

Notes and frequencies

In order to properly understand the design of the guitar tuner, you need some basic knowledge of musical notes, intervals and frequencies. The starting point for tuning musical instruments is often the A note with a frequency of 440 Hz. The other notes are called B, C, D, E, F and G. The intervals between these seven notes are either half tones or full tones. The interval between the B and C notes and between the E and F notes is a half tone. The interval between all other pairs of notes is a full tone.
An octave consists of a series of twelve half tones. They are called C, C#, D, D#, E, F, F#, G, G#, A, A# and B, where the sharp sign (#) indicates a note that is a half tone higher than the note without the sharp.
The B note of each octave is followed by the C note of the next octave. The octaves are numbered in sequence. The A note with a frequency of 440 Hz is in octave 4, which is indicated by the notation "A4". A note that is an octave higher than another note with the same name has twice the frequency of the latter note. For example,

the A5 note has a frequency of 880 Hz.
The six strings of a guitar are tuned to E2, A2, D3, G3, B3 and E4. The low open E string (E2) has a frequency of 82.407 Hz, while the high open E string (E4) has a frequency of 329.628 Hz. The twelfth fret of the high E string (E5) yields a frequency of 659.255 Hz. If you finger the high E string at the 24th fret (E6 note), the resulting frequency is 1318.51 Hz. This determines the upper limit of the spectrum for the guitar tuner. At the other end of the spectrum, you have the bass guitar (with four strings), which is tuned an octave lower than a regular guitar: E1, A1, D2, G2 (E1 is 41.203 Hz). A five-string bass guitar also has a B string (B0; frequency 30.868 Hz). This determines the lower limit of the spectrum for the guitar tuner.
The frequencies of the notes within an octave do not increase linearly, but instead exponentially by a factor of 2^x. For example, if the A4 note is 440 Hz then the A#4 note is $440 \times 2^{1/12} = 440 \times 1.059463 = 466.164$ Hz.
This forms the basis for the values in **Table 1**, which are represented directly in the C code.

Key requirements for the guitar tuner

Before we describe the actual design, let's summarize the key requirements for the guitar tuner:

- **Fast**: Updated ten times a second.
- **Accurate**: The error margin for determining the fundamental frequency must be less than 5%.
- **Display**: Indication of the note with the octave number and whether it is natural or sharp.
- **Deviation**: Indication of the relative deviation from –50% to +50% using 21 LEDs (5% per LED).
- **Light show**: Animated display moving toward the central lamp when the note is in tune.
- Flat tuning: It must be possible to set the tuner for flat tuning (all strings tuned one or more tones lower) from 0 to minus 6 so that the display can be read in the normal manner.
- **Range**: The lowest string of a bass guitar (B0 with a five-string bass) up to and including the highest note on a regular guitar (E6) must be detectable. This makes the tuner usable for a wide variety of musical instruments.

About the author

Emile van de Logt is employed at Rotterdam University of Applied Sciences as an Education Manager for the Electrical Engineering and Health Care Technology faculties. He studied Electrical Engineering at Eindhoven University of Applied Sciences, and Management Science at the Open University.
Emile is an electronics enthusiast. He conducted the Embedded C Programming and FPGA/VHDL workshops for Elektor, builds tube amplifiers and guitar effect equipment, and is an amateur beer brewer with a completely automated brewing system for which he personally designed all the hardware and software.

```
const float note_freq[MAX_NOTES] = {  20.602,  21.827,  23.125,  24.500,  25.957,  27.500,  29.135,  30.868,
  32.703,  34.648,  36.708,  38.891,  41.203,  43.654,  46.249,  48.999,  51.913,  55.000,  58.270,  61.735,
  65.406,  69.296,  73.416,  77.782,  82.407,  87.307,  92.499,  97.999, 103.826, 110.000, 116.541, 123.471,
 130.813, 138.591, 146.832, 155.563, 164.814, 174.614, 184.997, 195.998, 207.652, 220.000, 233.082, 246.942,
 261.626, 277.183, 293.665, 311.127, 329.628, 349.228, 369.994, 391.995, 415.305, 440.000, 466.164, 493.883,
 523.251, 554.365, 587.330, 622.254, 659.255, 698.456, 739.989, 783.991, 830.609, 880.000, 932.328, 987.767,
1046.502,1108.731,1174.659,1244.508,1318.510,1396.913 };

const char note_name[MAX_NOTES][4] = {"E0","F0","F#0","G0","G#0","A0","A#0","B0",
                "C1","C#1","D1","D#1","E1","F1","F#1","G1","G#1","A1","A#1","B1",
                "C2","C#2","D2","D#2","E2","F2","F#2","G2","G#2","A2","A#2","B2",
                "C3","C#3","D3","D#3","E3","F3","F#3","G3","G#3","A3","A#3","B3",
                "C4","C#4","D4","D#4","E4","F4","F#4","G4","G#4","A4","A#4","B4",
                "C5","C#5","D5","D#5","E5","F5","F#5","G5","G#5","A5","A#5","B5",
                "C6","C#6","D6","D#6","E6","F6"};
```

Table 1.

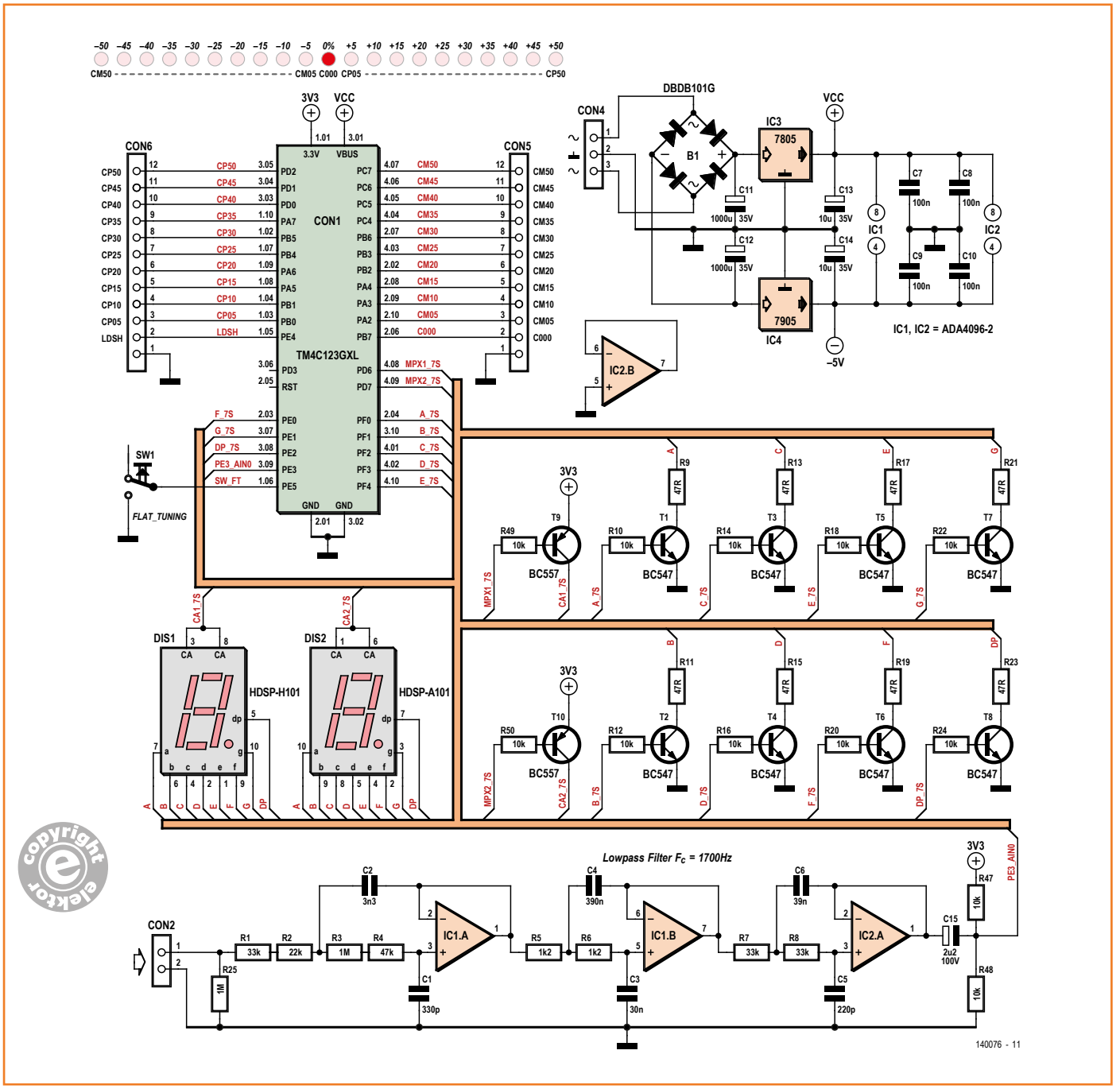
Figure 3.
The complete circuit diagram does not contain a lot of components, since most of the functionality is located on the TM4C123GXL board.

Design description

The hardware of the guitar tuner (see **Figure 3**) and the software work closely together, which is why they are described as a unit here. By the way, it's helpful to download the software for this article [8] to help you follow the description.

The design is basically structured as follows:

- The guitar signal is applied to the input of a sixth-order low-pass filter with a corner frequency of 1700 Hz. This filter was designed using TI's WebBench Filter Designer tool. The manufacturer specified their own opamp for this (ADA4096), but a standard opamp such as the LM833 is perfectly adequate here.
- The output of the low-pass filter is read in by



AIN0 (ADC input 0) of the microcontroller.

- The ADC clock frequency is 125 kHz, and there is 16-fold hardware averaging. This means that each sample value is determined by taking the average of 16 input samples. The time needed for this is 128 µs. The settings for this are configured in the routine `init_timer0_for_adc()`.
- The Yin algorithm needs at least two complete cycles of the guitar signal in order to determine the fundamental frequency. With the lowest note (B0 at 30.868 Hz), this corresponds to approximately 65 milliseconds. The update rate is 10 Hz, or in other words every 100 milliseconds. This means that the Yin algorithm should be able to detect frequencies as low as 20 Hz (E0).
- The sample rate FS is set to 6 kHz. This means that 600 samples are read in every 100 milliseconds. This takes place in the interrupt routine `ADC0_SS3_IntHandler()`. The input samples are written to a ring buffer, which is a special type of buffer that can be written by one process and read by another process.
- After 600 samples have been read in, a sign (variable `sampled_new`) is sent to the While loop in the main() routine. The While loop reads the 600 samples from the ring buffer and writes them to the input buffer for the Yin algorithm, when is then called. The Yin algorithm in turn determines the fundamental frequency (variable `pitch`) of the guitar signal.
- Next the routine `find_nearest_note()` is called. This routine determined which note in the previously mentioned table is closest to the measured value (variable `nnote`). It also determine the relative deviation in percent (variable `cents`).
- Both variables are passed on to another interrupt routine, `Timer1IntHandler()`, which runs at 200 Hz. It outputs the name of the note and the octave number to the 7-segment displays, which are driven in multiplex mode (this takes place in the routine `display_note()`). The relative deviation with respect to this note is also indicated by the 21 LEDs. This is handled by the routine `display_cents()`. The larger the deviation, the more LEDs are lit. This function is provided by a state machine that also generates the animated display when the string is in tune.

The only thing left to describe is how the various I/O signals of the TM4C123G board are used. Virtually all connector pins on the board are used, and thanks to the large number of pins there is no need for supplementary hardware. The pin assignments are shown in **Table 2**, and in the actual code.

The names shown in red in the table are the standard names of the connector pins, while the other names describe their functions in the guitar tuner.

- CP05 to CP50: connections for the 10 LEDs that indicate positive deviations.
- CM05 to CM50: Connections for the 10 LEDs that indicate negative deviations.
- C000: connection for the LED that lights up when the string is in tune.
- Flat tuning: input for the pushbutton switch used to set the degree of flat tuning. When it is first pressed the current flat tuning setting is shown; each subsequent press lowers the flat tuning by one step, down to a limit of minus 6. After this the flat tuning setting

	+3V3	**VBUS**	+5V
CP30	**PB5**	**GND**	GND
CP05	**PB0**	**PD0**	CP40
CP10	**PB1**	**PD1**	CP45
LED-sharp	**PE4**	**PD2**	CP50
Flat-tuning	**PE5**	**PD3**	
CP25	**PB4**	**PE1**	G_7seg
CP15	**PA5**	**PE2**	DP_7seg
CP20	**PA6**	**PE3**	AIN0
CP35	**PA7**	**PF0**	B_7seg

C_7seg	**PF2**	**GND**	
D_7seg	**PF3**	**PB2**	CM20
CM25	**PB3**	**PE0**	F_7seg
CM35	**PC4**	**PF0**	A_7seg
CM40	**PC5**	**RESET**	
CM45	**PC6**	**PB7**	C000
CM50	**PC7**	**PB6**	CM30
mpx1_7seg	**PD6**	**PA4**	CM15
mpx2_7seg	**PD7**	**PA3**	CM10
E_7seg	**PF4**	**PA2**	CM05

Table 2.

Figure 4.
The PCB for the guitar tuner. The TM4C123GXL board is plugged onto the corresponding connectors on the rear of the PCB.

returns to zero. Example: When flat tuning is set to –1, the E string is tuned to D#. However, when this string is plucked the display shows "E". If flat tuning is set to 0 in this situation, the display will show "D#".

- LED-sharp: Connection for the LED that lights up when a half tone (#) is detected.
- mpx1_7seg and mpx2_7 seg: Used to drive the 7-segment displays in multiplex mode.
- The xx_7seg signals supply the drive levels for the 7-segment displays.

Putting it all together

You have a choice of two different ways to install the software:

Extended: Download and install Code Composer Studio (CCS) version 6 and the Tivaware API [10]. This should not create any problems if you use the previously mentioned Getting Started workbook. Installation is described in detail in the first section. Then you can download the guitar tuner project from the Elektor website [8] and place it in the CCS workspace. This gives you the option of

Component List

Resistors

(SMD1206)
R1,R7,R8 = 33kΩ
R2 = 22kΩ
R3,R25 = 1MΩ
R4 = 47kΩ
R5,R6 = 1.2kΩ
R9,R11,R13,R15,R17,R19,R21,R23 = 47Ω
R10,R12,R14,R16,R18,R20,R22,R24,R47,R48 ,R49,R50 = 10kΩ

Capacitors

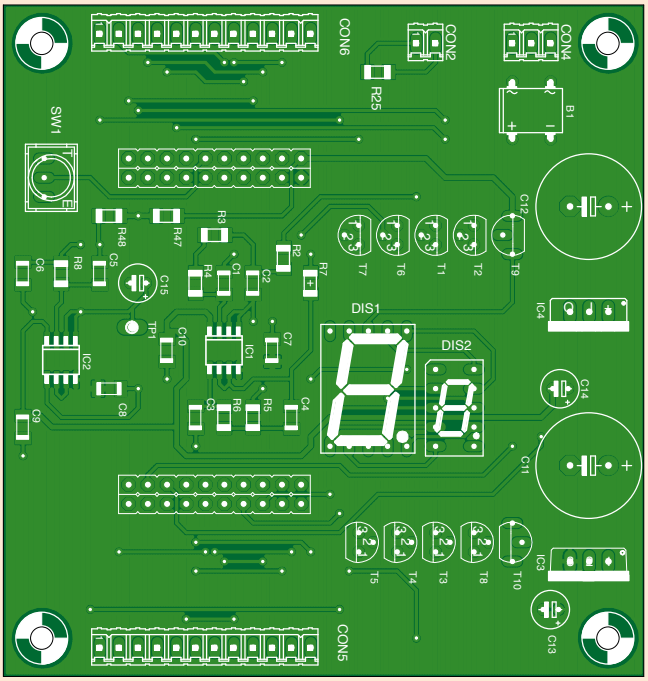
(C1–C10: SMD1206)
C1 = 330pF
C2 = 3.3nF
C3 = 30nF (e.g. Mouser 581-12061C303JAT2A)
C4 = 390nF 16V
C5 = 220pF 16V
C6 = 39nF 50V
C7,C8,C9,C10 = 100nF
C11,C12 = 1000µF 35V radial (0.2" pitch; 0.5" diam.)
C13,C14 = 10µF 35V radial (0.1" pitch, 0.2" diam.)
C15 = 2.2µF 100V radial

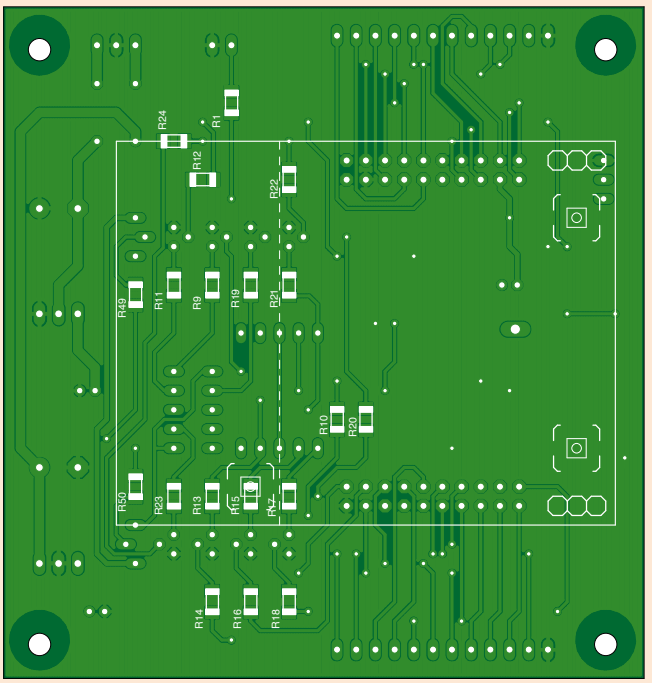
Semiconductors

B1 = DB101 bridge rectifier
T1–T8 = BC547
T9,T10 = BC557
IC1,IC2 = ADA4096-2 or LM833 dual opamp (SOIC-8)
IC3 = 7805
IC4 = 7905
21 pcs. LED. high-efficiency, red, 3mm (for CP05–CP50, C000, CM05–CM50)
1 pc. LED, high-efficiency, red, 5mm (for LDSH)

Miscellaneous

TM4C123GXL evaluation board
DIS1 = 7-segment-display, red, common anode, 0.56", 13 mcd@10 mA (e.g. Avago HDSP-H101)
DIS2 = 7-segment-display, red, common anode, 0.3", 10 mcd@10 mA (e.g. Avago HDSP-A101)
CON2 = 2-pin pinheader, 0.1" pitch
CON4 = 3- pin pinheader, 0.1" pitch
CON5,CON6 = 12-way PCB terminal block, 0.1"pitch
SW1 = pushbutton with make contact
2 pcs. 20-way (2x10) receptacles for evaluation board, 0.1"pitch
3 pcs. jack sockets, panel mount, 0.25" diam.
PCB artwork, # 140076, download at [1]





modifying the code if you so wish and generating a new .bin file. Next, connect the TM4C123GXL board to the PC with a USB cable and click the Debug button. The new firmware will be loaded directly to the TM4C123GXL board.
Minimal: This is intended for people who only want to program the TM4C123GXL board with the guitar tuner firmware and do not want to install the complete CCS development environment. In this case, only install the LM Flash Programmer software [11] and use it to upload the Guitar_Tuner_Yin.bin file from the Debug directory for the project to the TM4C123GXL board. The procedure for this is also described in the Workbook file.

You can order the TM4C123GXL board directly from the manufacturer, which is fast and the least expensive, or you can order it from a distributor such as Farnell. A small modification to be board is necessary before it can be used with the guitar tuner firmware. For compatibility with other TI boards, some port pins on the board are connected together. For instance, PB7 is connected to PD1 via resistor R10 (0 Ω) and PB6 is connected to PD0 by resistor R9 (0 Ω). These two resistors are located just below the microcontroller and above the manufacturer's name, and they are labelled R9 and R10. Using a soldering iron, carefully remove these two resistors. The PCB designed for the guitar tuner (**Figure 4**) is intended to be mounted on top of the TM4C-123GXL board as a sort of 'shield'. The shield holds the low-pass filter, the power supply, and the connections for all of the LEDs and the 7-segment displays. Several SMD components are used, but they can be soldered relatively easily with a fine-tip soldering iron. However, a hot-air soldering station is a better choice.
Start by assembling the power supply components on the shield board. The transformer (not shown on the schematic) can be a 2x6 V or 2x9 V type, and it does not have to supply much power. After assembling the power supply, first check whether it provides regulated +5 V and –5 V voltages. Disconnect power to the board and then solder the rest of the components on the PCB. The 7-segment displays may be soldered directly to the PCB or fitted in sockets to raise them off the board. Mount the 2x10-pin female headers on the rear of the shield board so that it can be plugged into the TM4C123GXL board. Wire the LEDs to connectors CON5 and CON6. There are 21 in total. The central LED connects to pin 2 of CON5 (C000), with the LEDs CP05 to CP50 (deviation +5% to +50%) to the right and the LEDs CM05 to CM50 (deviation -5% to -50%) to the left. There is also the Sharp LED for indicating half tones, which connects to pin 2 of CON6.

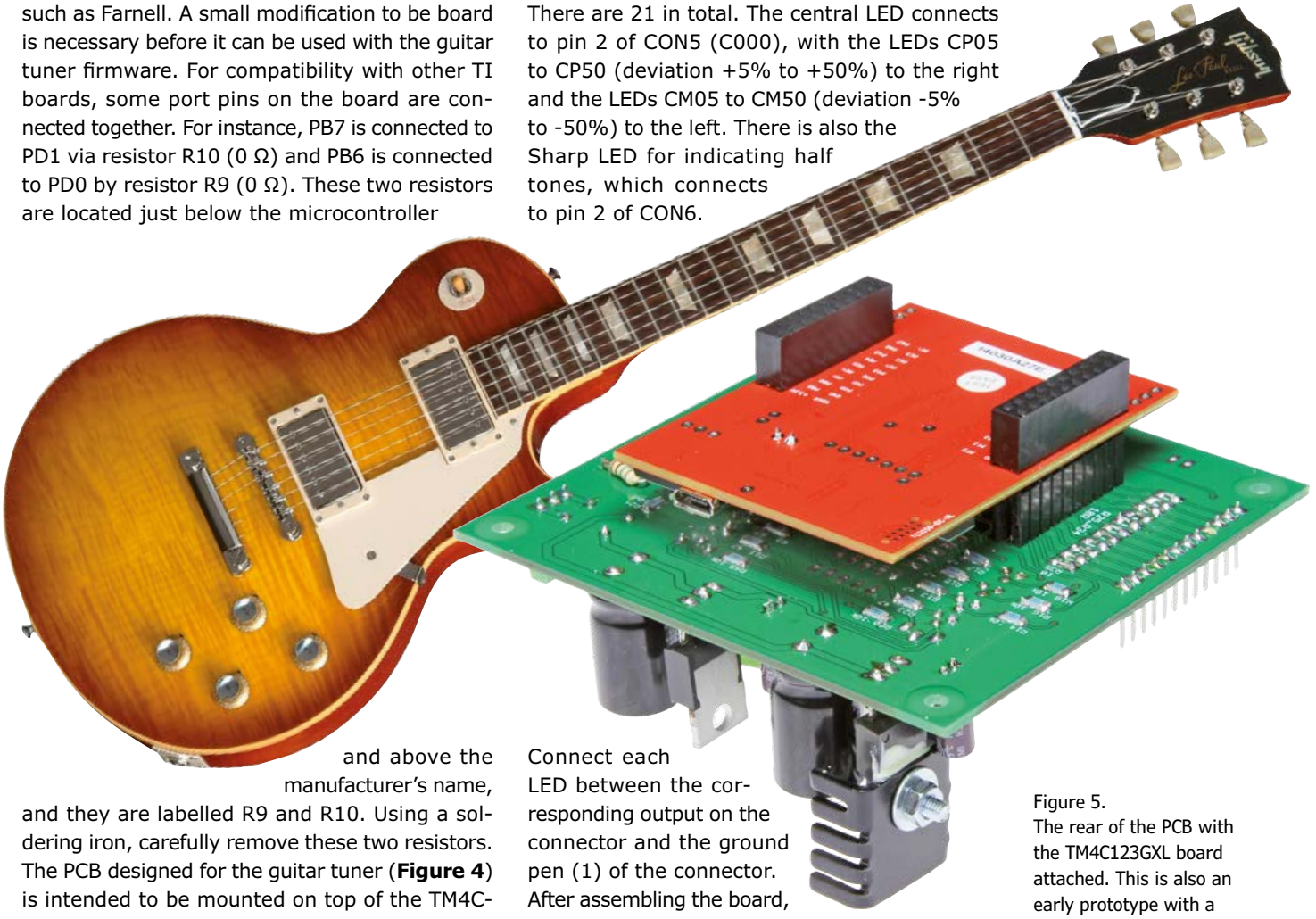


Figure 5.
The rear of the PCB with the TM4C123GXL board attached. This is also an early prototype with a number of changes made on the board.

Connect each LED between the corresponding output on the connector and the ground pen (1) of the connector. After assembling the board, check the low-pass filter for correct operation by connecting a sine-wave

Figure 6.
Photo of the author's prototype, which he regularly uses on stage.

signal (500 mV_{pp}) to CON2. The output signal from the filter can be seen at test point TP1 (LPF_OUT). The filter should allow signals up to 1700 Hz to pass through and block signals at higher frequencies.

If the filter works properly, you can connect the TM4C123GXL board (evaluation board on the bottom, shield on top, USB connector facing C11/C12/IC3/IC4). Now check that the decimal point of the small display blinks once per second. Then press the Flat Tuning button and check whether 0, -1, -2, ..., -6 appear on the display. Set Flat Tuning back to 0, connect a guitar, and pluck a string. The note and the octave number should appear on the display, and the LEDs should light up. Tune the guitar so that the central LED (C000) lights up. With the author's prototype, notes as high as D5 (22nd fret on the high E string) were detected properly. The low E string (E2) also works well. The tuner has not yet been tested with a bass guitar, but it was tested with a function generator. That showed that frequencies down to 20 Hz were detected without problems.

If you wish, you can fit the project in an enclosure. The author used a Hammond chassis (515-0950) for this, equipped with a front panel. That is also the reason why the LEDs are not mounted directly on the PCB, but instead fed out over CON5 and CON6.

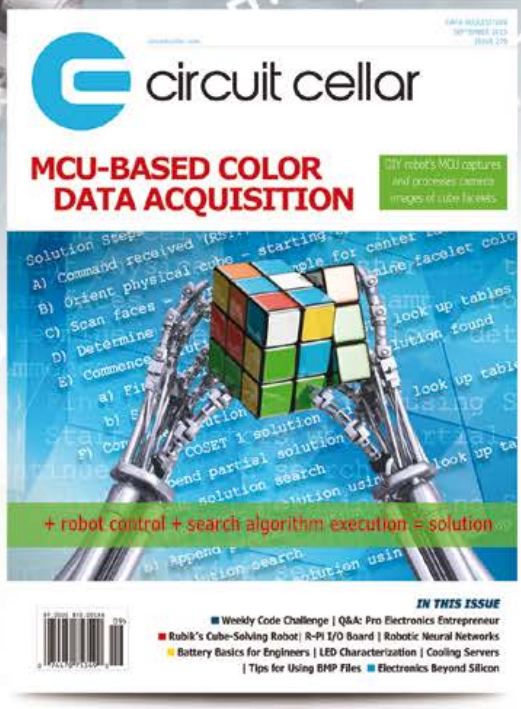
Happy Tuning!

(140076-I)

Web Links

[1] www.ti.com/tool/EK-TM4C123GXL
[2] http://processors.wiki.ti.com/index.php/Getting_Started_with_the_TIVA%E2%84%A2_C-Series_TM4C123G_LaunchPad
[3] http://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem
[4] YIN, a fundamental frequency estimator for speech and music, 2002, Alain de Cheveigne & Hideki Kawahara
[5] http://en.wikipedia.org/wiki/Autocorrelation
[6] https://github.com/JorenSix/Pidato
[7] https://github.com/ashokfernandez/Yin-Pitch-Tracking
[8] www.elektor-magazine.com/140076
[9] http://www.ti.com/lsds/ti/analog/webench/webench-filters.page
[10] www.ti.com/tool/ccstudio
[11] www.ti.com/tool/lmflashprogrammer

We're celebrating a new look — and the next 25 years.



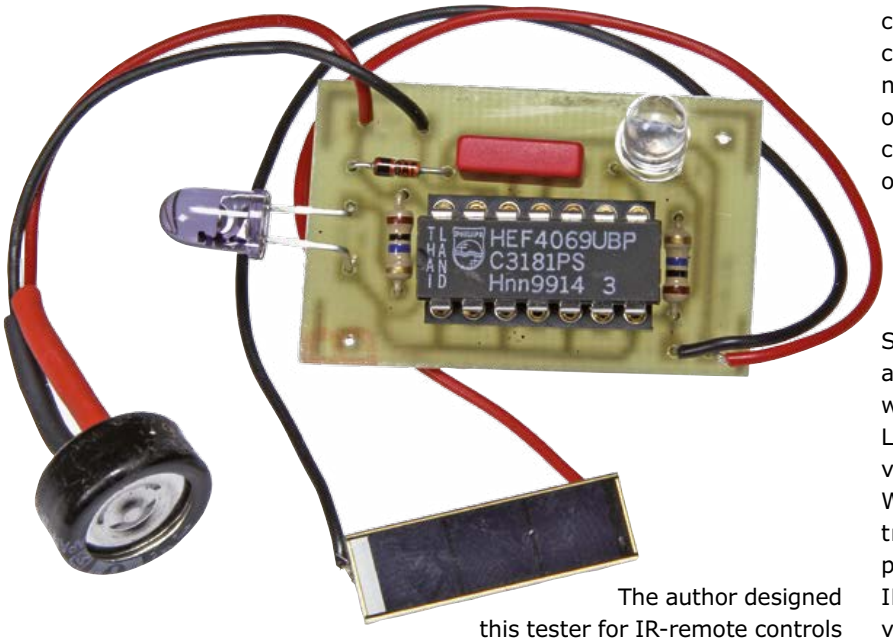
25% OFF CIRCUIT CELLAR

Whether it's **programming advice** or **design applications**, you can rely on *Circuit Cellar* for solutions to all your electronics challenges. **Raspberry Pi**, **embedded Linux**, **low-power design**, **memory footprint reduction** and more! Become a member, and see how the hottest new technologies are put to the test.

JOIN TODAY!

www.circuitcellar.com/sepN13

IR Tester with Solar Cell Power Supply



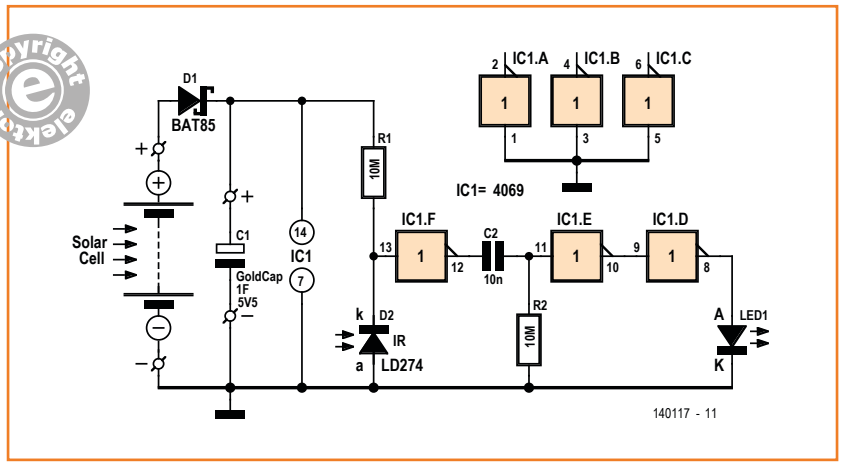
By **Wouter Eisema**
(Netherlands)

The author designed this tester for IR-remote controls as an excuse to experiment with a GoldCap. Of course there are simpler ways to test a remote control, but the version described is of singular design.

The energy consumption of this circuit is so small that it can easily be powered from the combination of a solar cell and a GoldCap. An on/off switch is not necessary.
A small solar cell charges a GoldCap (C1) via diode D1. This is a special type of electrolytic capacitor with small dimensions and a very large capacitance, in this case it is 1 farad! There is no need to limit the charging current into this type of super-capacitor. D1 prevents the electrolytic capacitor from discharging via the solar cell. In order to keep the voltage drop across the diode as small as possible, a Schottky type was used (BAT85) for this.

An IR LED is used to detect the IR signals. Such an LED does not only work as a generator of IR light, but also generates a voltage when exposed to IR light. For this purpose the LED needs to be connected the 'wrong way' via a resistor to the power supply voltage. When modulated IR-light from a remote control reaches the LS274, the voltage on IC1 pin 13 will change in sync with the detected IR pulses. The first inverter will convert these variations into digital pulses which then continue to a highpass filter, consisting of C2 and R2. This filter blocks the low frequency signals and ensures that the indicator LED is only driven by short-duration pulses. After the filter there are two more inverters connected in series. The final one drives the red LED directly, the inverter output will self-limit the LED current.

As you can see from the schematic, not all the gates in the IC have been used. As is customary, the unused inputs have been connected to ground.



You can test many different types of IR remote controls with this circuit. Press any arbitrary button on a remote control and the LED will light up as long as you keep the button pressed. The distance between the IR-source and the IR LED should not be too large, about 20 cm (8 inches). The circuit itself can also be tested in the absence of a remote control: Point the LED of the tester towards bright daylight, briefly cover the IR LED and the indicator LED will light up briefly.

A small circuit board has been designed for the tester, but you can also easily build it on a small piece of prototyping board. The solar cell and the

GoldCap are connected to the circuit board with a few short pieces of wire.
The circuit can be made very compact of you use SMD components. This would allow it to be implemented as a key chain, for example, with a transparent window in the enclosure for the solar panel.

(140117-I)

Web Link

www.elektor-magazine.com/140117

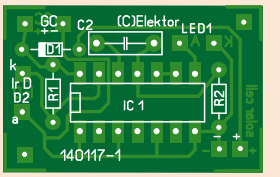
Component List

Resistors
R1,R2 = 10MΩ

Capacitors
C1 = GoldCap 1F 5.5V
C2 = 10nF

Semiconductors
LED1 = LED, high-efficiency, red, 5mm
D1 = BAT85
D2 = IR-LED LD274
IC1 = 4069

Miscellaneous
Solar cell, size approx. 22x7mm, e.g. Conrad Electronics # 160839
PCB artwork, # 140117-1, free download at [1]



LM317 Turns 78xx

By **Johnny Verhoeven**
(Netherlands)

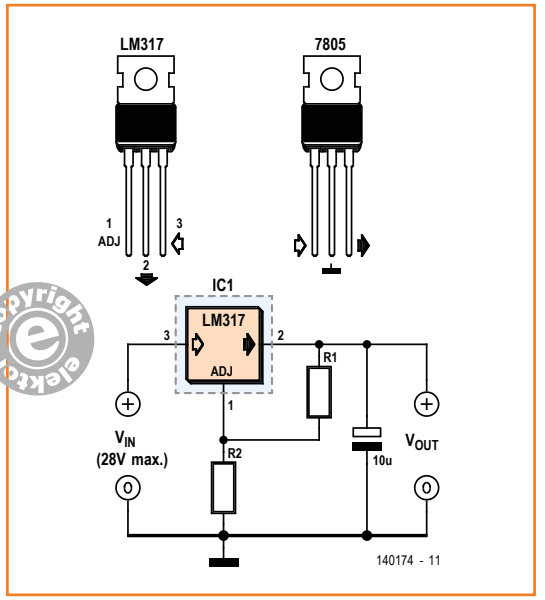
The author of this tip was recently able to get his hands on a bunch of LM317 voltage regulators. Now he has little need for an adjustable regulator in the circuits that he builds. Fixed voltage regulators of the type 78xx he uses frequently however, but he didn't want to modify his existing circuit board layouts to accommodate the LM317 regulator. To prevent the bunch of LM317s from being neglected in a corner he thought of the following. With the aid of a small piece of prototyping board, two (or three) resistors, an electrolytic capacitor and a PCB header he turned an LM317 into a 78xx (see schematic). The schematic is the standard application for the LM317. The output voltage amounts to:

$$V_{out} = 1.25 \times (1+R2/R1)$$

The table lists the resistor values for various output voltages. The theoretical value is shown for resistor R2. You can use the nearest E96 value for this or approximate this value by connecting two resistors in series.

The significant part is the construction of the entire assembly, as can be seen in the photo. It is important that the PCB header has the same pinout as a 78xx. If you would like to save some space then it is possible to omit the electrolytic capacitor.

(140174-I)



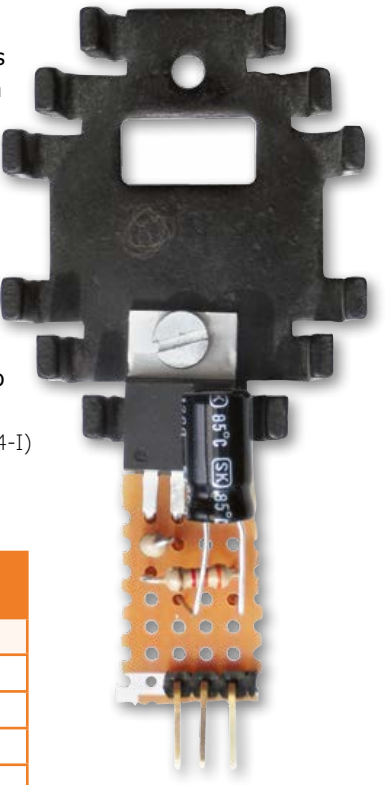


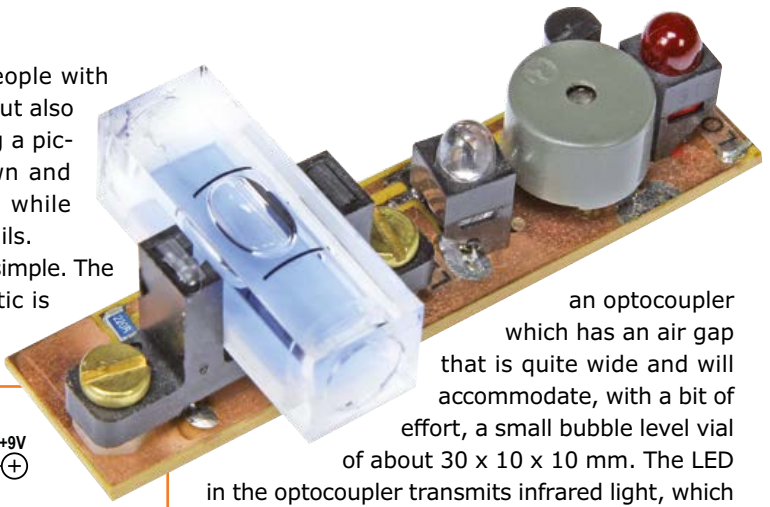
Table 1. Resistance values for various standard output voltages.

V_{out} (78xx model)	R1	R2
5 V (7805)	470 Ω	1410
6 V (7806)		1786
9 V (7809)		2914
12 V (7812)		4042
15 V (7815)		5170

Spirit-Level Acoustic Aid

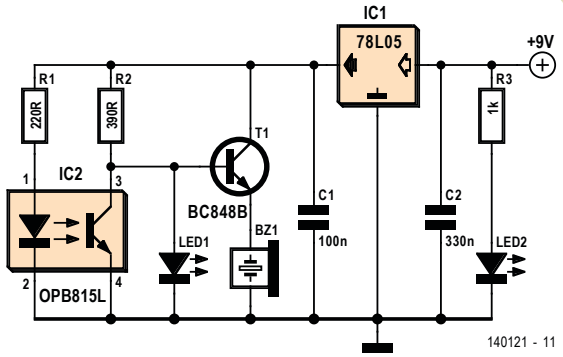
By **Wouter Eisema**
(Netherlands)

This aid for a spirit-level gives an acoustic and optical indication when the air bubble in the vial of the level is exactly in the middle. In comparison with existing optical and acoustical levels, this aid is much cheaper.



This circuit is very handy for people with hearing or vision impairments, but also for those who would like to hang a picture frame or shelf on their own and can't keep an eye on the level while they're occupied with other details.
The circuit design is exceedingly simple. The main component in the schematic is an optocoupler which has an air gap that is quite wide and will accommodate, with a bit of effort, a small bubble level vial of about 30 x 10 x 10 mm. The LED in the optocoupler transmits infrared light, which goes through the vial and then arrives at the opto-transistor. In parallel with this transistor are an LED and a transistor with a self-oscillation buzzer. When the IR light passes through the liquid, the transistor receives enough light so that it will conduct. The LED and buzzer are then not active. When the air bubble is in the middle, the IR-light becomes sufficiently scattered, with the result that the opto-transistor will turn off and as a consequence the LED turns on and the buzzer sounds. The power supply for the circuit is provided by a 5-V voltage regulator. Since the circuit will only be used intermittently and draws only a few tens of milliamps, it can easily be powered from a 9-V battery. LED2 serves as power supply indicator, so that you will not forget to turn the circuit off after use.
The author designed a small circuit board for the circuit, but it contains so few parts that it would also be very easy to build it on a small piece of prototyping board. Note that the circuit board shown uses some SMD parts. After mounting all the parts, the spirit-level vial can be clamped in the optocoupler and, if necessary, be held in place with a little glue once the correct position has been found for proper operation.

It is best to build the circuit into a small enclosure. The enclosure will house the circuit, an on/



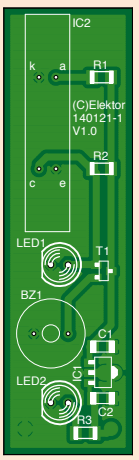
Component List

Resistors
R1 = 220Ω (SMD 1206)
R2 = 390Ω (SMD 1206)
R3 = 1kΩ (SMD 1206)

Capacitors
C1 = 100nF (SMD 1206)
C2 = 330nF (SMD 1206)

Semiconductors
LED1 = LED, white, 5mm
LED2 = LED, red, 5mm
T1 = BC848B (SOT-23)
IC1 = 78L05 (SOT-89)
IC2 = IR optocoupler with wide gap, e.g. Optek OPB815L

Miscellaneous
BZ1 = active piezo buzzer, small model
Small vial to fit in optocoupler
9V battery and clip-on connector
PCB layout # 140121-1, download at [1]



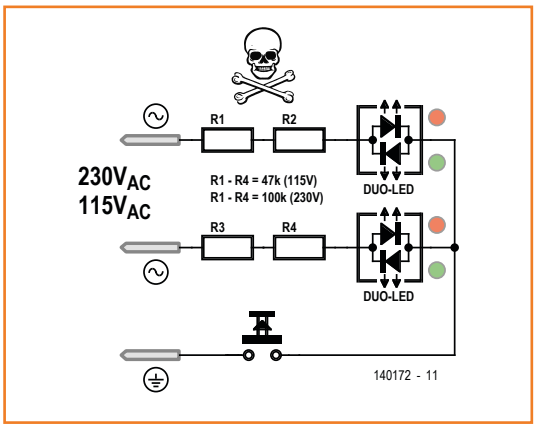
off switch and a 9-V battery. The circuit board is mounted in the enclosure in such a way that is can be moved slightly in order for the indication to match that of an external vial which is glued to the outside of the enclosure as an additional aid. The little enclosure can be used on its own to level objects, but can also be attached with a spring clamp to a normal spirit level and be used as an additional indicator.

(140121-I)

Web Link

www.elektor-magazine.nl/140121

Power Outlet Tester using Bicolor LEDs



Editorial note: this circuit may not work, or not as described, in all countries depending on the way the Neutral (N) and Earth/Ground lines are connected in accordance with the national electricity code and relevant guidelines for electrical safety.

Most simple phase testers indicate whether an AC powerline voltage is available; at the wall outlet they identifies the Live (L) connection.

This handy tester can do more. It uses just four resistors (two times two resistors in series to halve the voltage stress on each resistor), two bi-color LEDs and a pushbutton.
The complete circuit can be fitted inside a standard 3-pin plug-top with an Earth connector. Make sure that the circuit is completely enclosed in the plug housing and that the pushbutton is rated at AC line voltage.

With the socket tester plugged into an AC power outlet both LEDs should light to indicate a flow of current between the live (L) and neutral (N). The LED color is not important; both bicolor LEDs will be lit so the overall color will be orange. Bicolor LEDs have been specified to avoid the need for an additional diode which would be connected in anti-parallel if a single LED were used.

When the pushbutton is pressed only one of the LEDs will light, indicating that the power outlet Earth connection is correctly wired. The lit LED indicates the Live (phase) connection.

(140172)

By
Hans-Norbert Gerbig
(Germany)

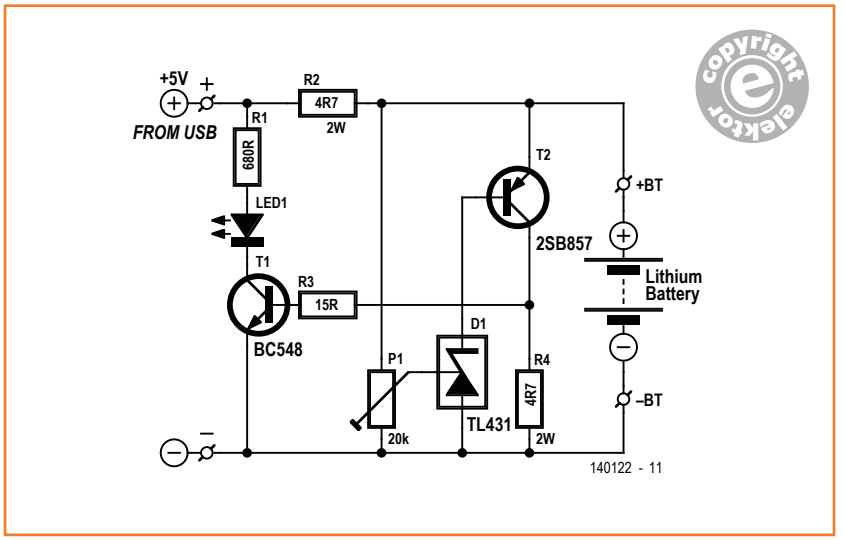
Current Boost for USB Lithium Charger

By **Ian Field** (UK)

A TL431 shunt regulator can be used to ensure that a lithium cell cannot be charged to more than 4.2 V, meaning: when the cell has reached its maximum voltage, the TL431 must shunt all of the current flowing through the limiting resistor. A resistor with a value of around 10 ohms then limits the current to within what the TL431 can safely shunt.

Another approach shown here is to add a PNP output transistor to the TL431 in much the same manner as a Sziklai pair. Now the maximum shunt current is the 100 mA that the TL431 can handle multiplied by the gain of the PNP transistor. As the TL431's maximum current rating is no longer the limiting factor, the only constraint is the maximum current that a correctly specified USB charger can supply (500 mA). When the battery is at its lowest voltage, the current limiting resistor works out at 4.5 Ω, with 4.7 Ω being the most suitable actual value.

The transistor used here is a 2SB857 (T2), this was the junkbox part that came readily to hand while rounding up the parts. There's nothing particularly special about this device, it is rated 50 V, 4 A and a $h_{fe(min)}$ of 60, meaning just about any TO220 PNP with similar ratings will do the job. The one in the prototype was not fitted with any heatsink. With no load (maximum shunt current) it did get moderately warm, but not in the least uncomfortable.



As the PNP transistor now does most of the heavy lifting and its collector is returned to the negative rail, a current sensing resistor, R4, can be inserted in the collector lead to switch a second transistor, T1, which operates the 'cell full' indicator, LED1. This would have been more complicated with only the TL431 as inserting the current sensing resistor in the anode lead would create a variable offset voltage working against its reference point. The current sensing resistor in the collector lead is included in the feedback loop and has very little effect on regulation. The 'battery full' voltage is accurately set with P1.

(140122)

Join the Elektor Community

Take Out a GOLD Membership Now!

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; our virtual, online laboratory
- 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 projects
- Exclusive GOLD Membership card

ALSO AVAILABLE:

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

elektor GREEN membership

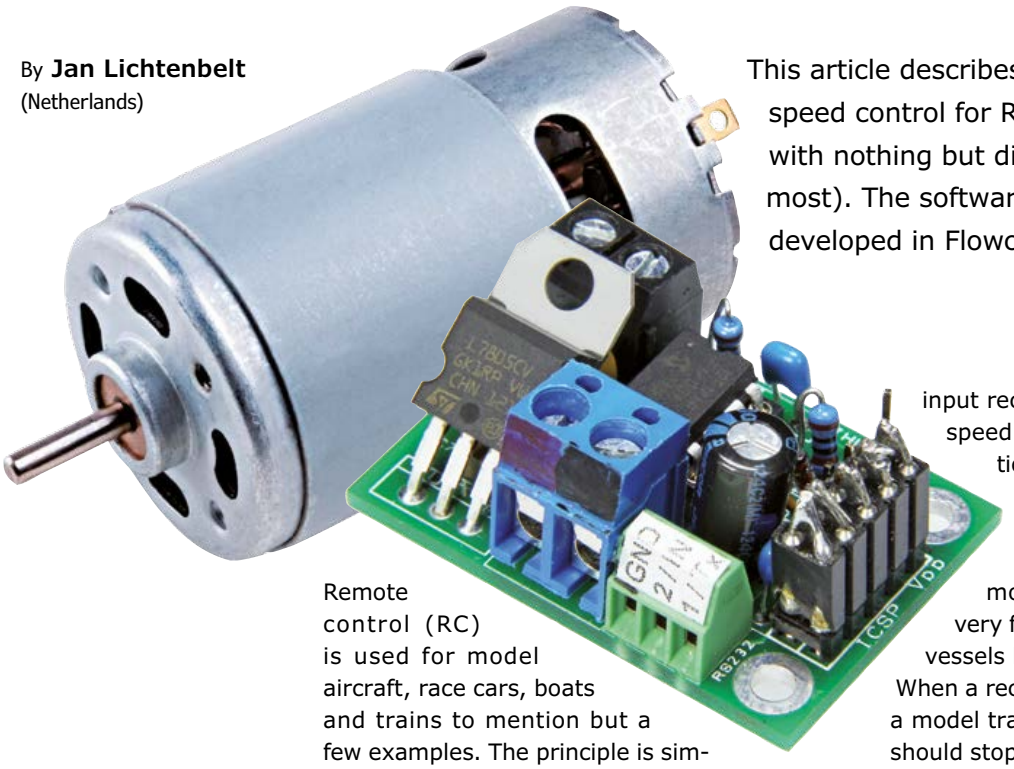
Take Out Your Membership Now at www.elektor.com/member

Connect with us! www.facebook.com/elektorim www.twitter.com/elektor

RC Speed Control for DC Motors

Small and versatile

By **Jan Lichtenbelt**
(Netherlands)



This article describes how to DYI a PWM-based speed control for RC models on a small board with nothing but discrete components (well almost). The software for the project was entirely developed in Flowcode.

Remote control (RC) is used for model aircraft, race cars, boats and trains to mention but a few examples. The principle is simple: an RC transmitter sends code representing the position of one or more control levers ("sticks") to the receiver in the model. Depending on the position of the levers a pulse of 1 to 2 ms with a repetition rate of 50 Hz will become available in the model. This pulse can be used directly for controlling servo motors, but must be 'translated' for DC motors into a forwards- (aircraft and race car models), or forwards/reverse- (model train, boat) speed control.

There are commercial RC receivers available with pulsewidth modulation (PWM) outputs to control the speed of a DC electromotor. This article describes how you can build a PWM-based speed control yourself on a remarkably small PCB still yet using discrete components for the most part. A microcontroller can translate the RC dutycycle (5-10%) into a 0-100% PWM signal both in a forward or reverse direction by means of two PWM signals and a MOSFET bridge, see **Figure 1**. When one bridge input is kept Low and the other input receives a PWM signal, then the motor speed will be regulated in 'forward' direction. Swapping the two inputs reverses the motor direction.

There are small differences in requirements for different types of models. Aircraft and race cars need a very fast response, while model trains and vessels have a more natural, slow response. When a receiver failure occurs it is desirable for a model train to keep running, while a race car should stop as soon as possible. Another failure situation to consider is what to do after an overload current or an overtemperature of the bridge driver or motor?

The original purpose of this design was an electronic bridge driver for the motor with through-hole parts only. The only exception in this circuit is a small bridge driver IC which fortunately can be soldered by hand. In experiments by the author a version of the circuit with four MOSFETs sometimes failed and therefore got rejected. The best choice was found in a MOSFET bridge ADP3624ARDZ, good for 4 amps peak current. This is enough for the model train motor used (gauge I) with typical currents of 0.3 A, and 1 A maximum. This bridge driver can be used with a supply voltage of 4.5-18 V; it has two non-inverting inputs and includes an over-temperature warning output. The IC is available in a SOIC package with 0.05" pin distance. This pin distance allows it to be soldered in a traditional

way with the aid of de-soldering wick to remove solder between the pins.

Hardware

The diagram in **Figure 2** shows the simple setup of the circuit. The Microchip PIC12F1840 receives the RC pulses of 1-2 ms and translates them into PWM signals. These signals are used to control the ADP3624 MOSFET bridge with the motor connected to the output. Resistor R6 (10 kΩ) in parallel with the bridge output is added to ensure that the design still works if no motor is connected. Capacitors C1 and C3 (100 nF) have to be as close as possible to the bridge driver and the motor. To use the MOSFETs as switches the gate voltages should be at least 4.5 V. With high current surges during switching, a supply voltage of 5 V can easily drop below this value. Therefore a ceramic capacitor (C7) with its high capacitance should be fitted as close as possible across the bridge driver. A 10-µF ceramic SMD capacitor is commercially available to hobbyists and can easily be soldered.

The bridge driver, IC2, gives a temperature warning by changing pin 8 from Low to High. This signal is used in the PIC microcontroller in an interrupt procedure. Resistor R1 (1 kΩ) and zener diode D1 (5.1 V) are included for safety reasons. The bridge shutdown pin (1; SD) of IC2 is not used, because shutdown is already implemented in the firmware executed by the microcontroller. When the RC transmitter is placed close to this

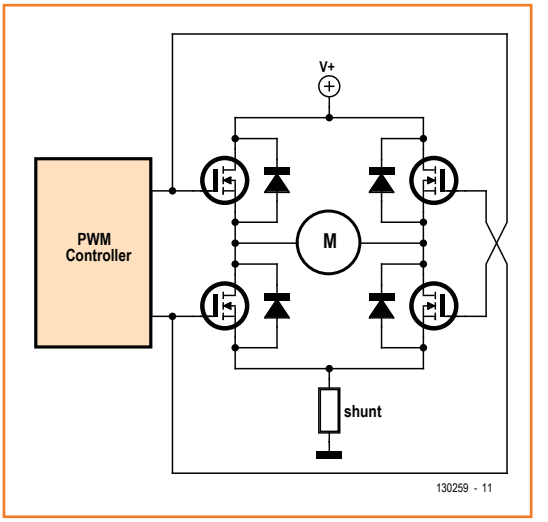


Figure 1.
Full bridge control of motor speed and direction. The shunt is optional.

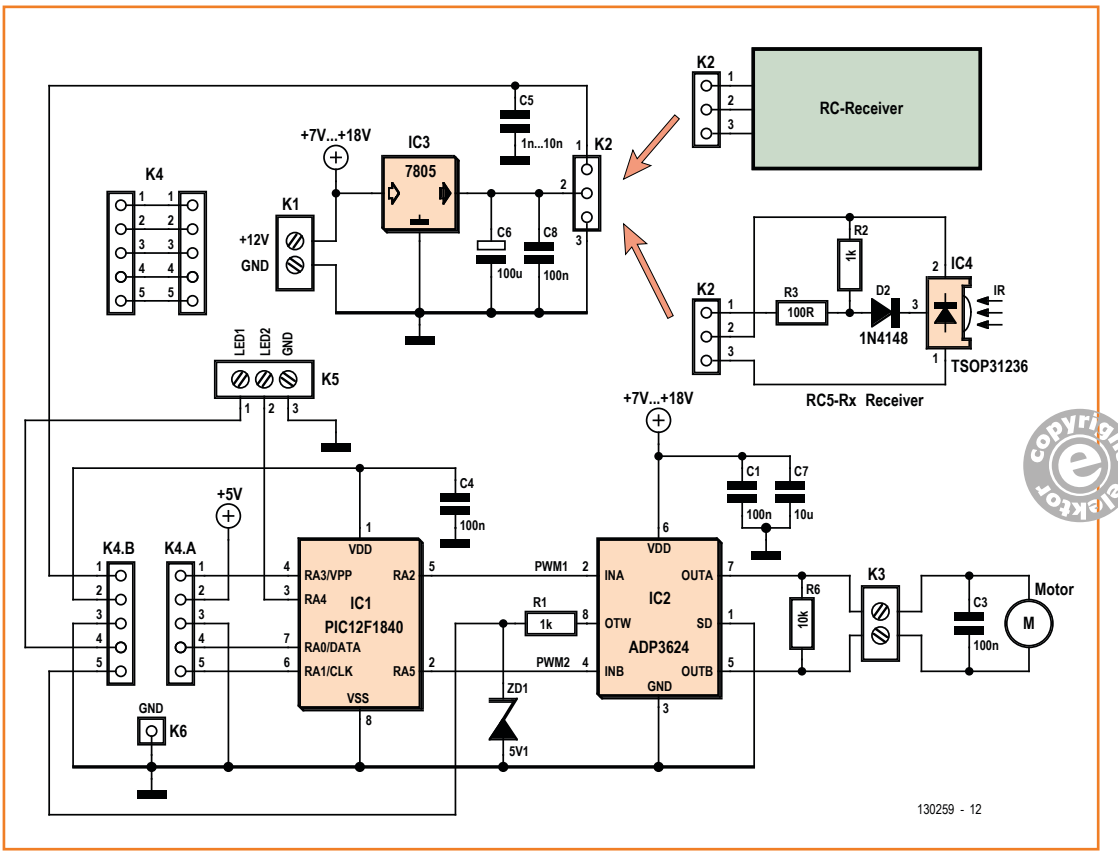


Figure 2.
Full schematic of the speed control circuit.

board some high frequency components (like 40 MHz in the author's country) can be present at the input of the microcontroller. A capacitor of 1-10nF (C5) at the input of the RC signal (K2a) will reduce these high frequency signals.

The power supply (7-18 Vdc) is straightforward comprising voltage regulator IC3 (L7805) and capacitors C6 (10 µF) and C8 (100 nF). C4 is an RF decoupling capacitor for the microcontroller. In principle this 1-amp voltage regulator can supply the RC receiver with 5 V and power additional servos connected to the same receiver. However, if the RC signal is lost, it is found that some RC receivers generate random pulses. Some servos will start to oscillate with likely high power peaks which can easily result loss of control by the microcontroller. If this happens, decouple the servos for a while and test if the speed control works correctly. Try to change to a different radio channel to improve the functioning. Connect the servos and test again. When the problems persist, it is strongly advised to use a separate voltage regulator for the servos or perhaps use additional capacitors. To avoid these surges when RC signals are lost, it is strongly advised to turn the RC transmitter on before the microcontroller is powered up, and to turn not only the RC transmitter off, but the microcontroller also. Use the best radio channel available.

The PIC 18F1840 microcontroller uses its internal oscillator of 16 MHz. The Master Clear is given by the microcontroller itself. A 2-µs software timer is used to sample the RC pulses of 1-2 ms, resulting in a higher accuracy than strictly needed for the byte (8 bits) to control the PWM output.

Essential data are permanently stored in EEPROM and will be available when needed in the program. The RC signal's pulses on pin 4 (port A3 of IC1) are recorded by means of interrupts on change. The RC receiver is powered at 5 V, but the RC pulses were found to be about 3 V. PIC input port A3 is TTL compliant and will just detect these 3-V RC pulses. The OverTemperature signal of the bridge driver is detected on pin 6 (port A1), which is part of a comparator with the reference voltage set internally to 2.5 V. The PWM frequency is produced by internal timer2 set here to about 40 kHz, which is within the requirements of 20-100 kHz for this type of bridge driver.

In-Circuit Serial Programming (ICSP™) via three pins of the microchip controller is possible using connector K4a. The data, clock and programming voltage of 13.8 V and the power supply voltage of 5V are connected to the microcontroller. During normal operation of the microcontroller a 5-pin header between the connectors K4a and K4b is used.

Instead of RC pulses on pin 3 of the microcontroller, the IR pulses of an IR remote control with RC5 protocol can also be connected to this port. This makes it possible to change input parameters of the microcontroller. A 36-kHz IR receiver TSOP31236 can be used for this. The output should be default high by means of resistor R2 (1 kΩ). Diode D2 (1N4148) will make this level Low by the IR receiver. Resistor R3 (100 Ω) protects the input of the microcontroller.

At connector K5 two LEDs can be connected to show the direction, 'Forward' and 'Reverse'. The maximum output current is 25 mA each. This allows using a double set of LEDs. The LEDs will used during the start-up procedure. If RC is lost (flickering) and during over temperature interrupts (both off). 'LED 1' connected to pin 7 of the microcontroller is also used to show data by means of RC5 communication with another micro. This RC5-Tx signal will be sent to another micro with an LCD screen to read start parameters used and to show various data during operation. Consider using the PIC16F628A or equivalent PIC 16F1847 as described in Elektor [1], without the RF transmitter. However, if you want to see the PIC data during operation, they can be sent and received with the RF modules described in that article. The software for this RC5_Rx Microchip PIC 16F1847 is part of this article.

To activate the RC5-Tx output on pin K5a-1, the other pin K5a-2 must be High at startup by means of a 1-kΩ resistor connected to +5 V. Attention: for normal use with two LEDs, K5a-2 must be Low at startup, for example, through a LED and current limiting resistor connected to GND.

Software

The software for this circuit is suitable for speed control suiting many applications.

The two possible start procedures take at least three seconds:

- Set the control stick to the maximum position. The 'Forward' LED will turn on. As soon as it turns off, put the control lever in the minimum position. The 'Reverse' LED will then turn on. As soon as it turns off return the control stick to its neutral position. This can be any position like the center position or the minimum position. It is important that

you can see both LEDs at the same time. If not, fit an additional pair of LEDs in a position where you can see them both. Minimum and maximum position will be stored in the EEPROM.

- Set the control stick to the neutral position. Turn the power for the microcontroller on and leave the handle in the neutral position for at least 3 seconds. The minimum and maximum handle position will be read from the EEPROM.

The center position can also be a little above the minimum, allowing braking with reversed motor direction in case of race cars. When using this for the first time, always test the correct rotation direction of the motor(s).

In general, when the RC receiver gets out of range of the RC transmitter the motor will continue with the last setting received. Using the RC5 communication as described below, this can be changed into halting the motor when RC signals are lost (with both LEDs blinking).

Only when the center position equals the minimum position, the motor will be controlled in the 'Forwards' direction only. The motor will only be able to go 'forward'. This can be used with race cars and airplanes. In that case the program stops the PWM if the car or plane is out of range of the transmitter for longer than 140 ms (or 7 RC pulses). It is strongly recommended to test this behavior regularly.

A PWM of 0% of 500 ms duration will prevent (too) high currents due to a sudden change of the motor direction. Each motor has to overcome its start friction and needs a minimum PWM percentage to start the motor. The program uses a typical PWM minimum value of 30%. Over-temperature warning of the bridge driver will stop the motor and it will start again after cooling down. Both LEDs will be off. The bridge driver does have an internal hysteresis for this control signal.

Some filtering of the RC signals was found to be necessary, especially if several transmitters are in the area. A moving average is used with a small filtering with an RC time (50%) of 60 ms (three RC pulses). Twelve RC pulses are used in the moving average, which takes about 12 x 20 = 240 ms. Theoretically for a step function, 93% will be achieved in this time. This all will result in a small degree of filtering in combination with a fast response, necessary for airplane applications in particular. For details, updates and Flowcode source code files, see [2].

Component List

Resistors

R1 = 1kΩ
R2 = 1kΩ (for IR receiver)
R3 = 100Ω (for IR receiver)
R6 = 10kΩ

Capacitors

C1,C4,C8 = 100nF
C3 = 100nF (on motor terminals)
C5 = 1-10nF
C6 = 100µF 25V
C7 = 10µF 25V, ceramic, SMD, e.g.
AVX-18123D106KAT2A (Farnell # 1327761)

Semiconductors

ZD1 = 5.1V zener diode
D2 = 1N4148 (for IR receiver)
IC1 = PIC12F1840
IC2 = ADP3624ARDZ or ADP3624ARDZ-RL (SOIC)
IC3 = L7805CV
TSOP31236 (36kHz version) connected to K2

Miscellaneous

K1,K3 = 2-way screw terminal block, 0.2"pitch
K2 = 3-pin pinheader, 0.1" pitch
K4 = 10-pin (2x5) pinheader, 0.1"pitch
K5 = 3-way screw terminal, 0.2" pitch
K6 = 1-pin pinheader
PCB # 130259-1

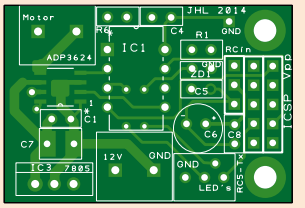


Figure 3.
PCB layout with ADP3624 bridge sitting between motor pins and C1. The dimensions of the board are 38.1 x 26.4 mm. RC-in doubles as the RC5-Rx input.

Communication

Changing the microcontroller parameters given above requires communicating with the microcontroller. An RS232 interface was rejected due to bad synchronization properties with only the start bit available if just the RS232-Rx is used. Instead, IR communication with RC5 protocol is used with much better synchronization half way each bit. RC5 IR-transmitters can be found in many households. The RC5 application addresses are not used. Consequently any RC5 transmitter, independent of its purpose, can be used. Only the button commands (6 bits) are read.

The RC5-Rx signal ends up at the same pin 4 (port A3 of IC1) as the RC pulses. Remove the RC input and connect the IR receiver IC to this pin. The RC5 input will be recognized due to the fact that its level is default high, while the RC pulse

are default Low. The software is very flexible and not only the official RC5 bit-time of 2 x 889 µs is detected, but also much faster pulses.
The RC5-Tx output will be available on the LED at pin 7 (port RA0 of IC1) if the other 'LED' at pin 3 (port A4 of IC1) is High by means of a 1-kΩ resistor to 5 V at startup (5 V is available on pin 2 of K4a/b for example).
RC5-Tx differs from the official protocol in two ways:

- The protocol is close to the RC6 protocol, which enables transmission of bytes (8 bits) or even integers (16 bits). The RC6' protocol used here, i.e. 2 start bits, 1 byte and 3 integers.
- We use a bit time period of 2 x 255 µs, which is the maximum frequency that can be used with the RF transmitter described at [1]. Each transmission will take (2+8+3x16 = 58 bits) 58 x 0.5 = 29 ms.

This requires a special RC5-Rx program for the PIC 16F1839 with LCD screen, which can be found in the free download offered with this article [4].

PCB

All components (through-hole and SMD) are mounted at the top side of the small PCB (**Figure 3**). Fix the bridge driver by soldering one pin, and then the other pins. Wick away any excess solder. Pin 1 of the ADP3624 bridge driver is indicated by the gray point on the package. At the bottom side of the IC there is an area intended to make thermal contact with the PCB. Remove the protection layer on the board at that particular place. Next, fit C7, the 10-µF ceramic capacitor. Pin 1 of the PIC microcontroller is at the upper left side. At the right side of the PIC the components are, top to bottom: R1, ZD1 (GND right side), C5 and C6. Right beside these are K3 (3-pin RC input) and C8. At the right side of the PCB you find 10-pin (2x5) ICSP™ connector K4. LED connector K5 has three additional fixing holes right at the edge of the PCB. Ground pin K6 at the upper right side is optional, this is convenient during testing the PCB.
See the video on YouTube [3] where the PCB gets tested—note that RS232 is still used for the communication, and the EB006 board for the ICSP™ programming.

(130259)

The firmware for this project was completely developed in Flowcode 6 for PIC (available from Elektor, www.elektor.com)

Web Links

[1] Modular RF Link using Manchester code, Elektor September 2013:
www.elektor-magazine.com/120049 and www.elektor-magazine.com/120187
Note: PIC16F628A replaced by the equivalent PIC16F1847. Programming of this microcontroller using Vpp on pin A5, Vdd, GND, Clock pin B6 and Data pin B7. Pin B0 is the RC5/RC6 input(Rx) or output(Tx) pin. LCD Data DB4…DB7 connected to IC pins A0–A3, RS to B4 and E to B5. Normally a resistor of 1kΩ is connected between VDD and MCRL.

[2] Jan Lichtenbelt: Remote Speed Control of DC Motors, Matrix Multimedia article MX040:
http://www.matrixmultimedia.com/article.php?a=677
and updates http://www.matrixmultimedia.com/mmforums/viewtopic.php?f=36&t=11613

[3] YouTube pre-testing the PCB (version with RS232, now replaced by RC5):
http://youtu.be/TSJCCADYKsE

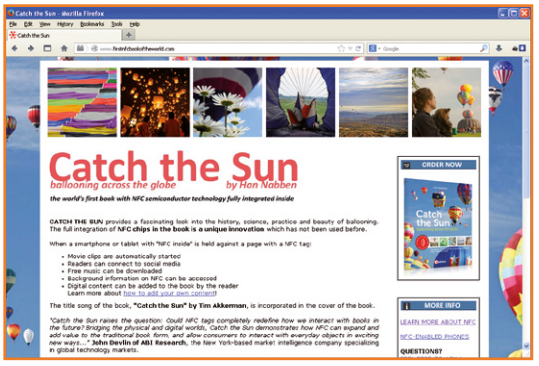
[4] Extra documentation and software: www.elektor-magazine.com/130259

Experimenting with NFC

Near field communication with the Elektor Member Card

By **Burkhard Kainka**
(Germany)

Like other members of the Elektor community, last year I received an Elektor Gold Member Card. It has an embedded Mifare chip for near-field communication (NFC). I tried to figure out how to use this capability, but I didn't get very far. My problem was that I didn't have any device that supported NFC. Things are different now, because I have become the proud owner of a Nexus 7 tablet. According to the data sheet, the Nexus 7 has a built-in NFC reader.



The first question was: does it actually work? To answer that question, I decided to build a simple resonant circuit for a quick test. NFC works at the allocated ISM frequency of 13.56 MHz, so I wound a coil with 15 turns and a diameter of 40 mm (about 1.6 inch) and soldered an LED to the leads. The resonant frequency of this arrangement is roughly correct. When I held the coil next to the rear of the Nexus, the LED blinked cheerfully, from which I concluded that the tablet was constantly transmitting signals to try to contact NFC tags.

Following this encouraging start, the next step was to hold my Elektor Member Card next to the Nexus to see what would happen. Guess what? The browser started up and opened the Elektor website at www.elektor.com. So it really works! Generally speaking, NFC technology has not made large inroads al least in Europe, so I had no idea at all of what to expect. Actually I should have known better, because I have a copy of the Elektor book *Catch the Sun*, which incorporates NFC technology. In particular, it contains several NFC tags, including two that are user-writable. So I decided to see what I could do with them.
It turns out that these NFC tags open or play quite a few things on the tablet, including websites, videos, and tunes. Among other things, I managed to find information about how to write my own data to NFC tags.

I got hold of a suitable app: TagWriter from NXP, the manufacturer of the Mifare tags [1]. The app also told me which kind of information I could write to the tags.

First I tried writing a link to my own website. Loading the data into the tag was easy – I only had to hold it next to the tablet. I used one of the tags on the last page of the book. In practically no time the link was stored in the tag, and it automatically launched the browser on the tablet. The next question was whether or not I could overwrite that data. I tried writing a message to the tag, and it worked. The tag opened a text window and the stored message was displayed. Next I decided to experiment with the Elektor Gold Member Card again. I wanted to see if I could reprogram it and write my website address to the tag. What I discovered is that the tag in the Elektor Gold card has less room for user data. However, it worked OK when I omitted the description of my website and only stored the Internet address (URL).

Now I have a good idea of what NFC is all about. Hopefully some other readers will find this short article useful.

(130216-I)

Web link

[1] https://play.google.com/store/apps/details?id=com.nxp.nfc.tagwriter

DVD'ed: The Full Range of 2000-2009 Volumes of Elektor Magazine!

A Must-Have for All Elektor Fans!

ISBN 978-1-907920-28-8

£77.95 • € 89.00 • US $121.00

Order Today at www.elektor.com/zeroes

Water Pump Regulator

for yachts, mobile homes, etc.

By **Paul Cordonnier**
(Belgium)

If you spend a lot of time at sea, or in and around harbors, surely you've been confronted at some point with the drawbacks of water distribution pumps on yachts, or pleasure craft in general.

In principle, a water system comprises a distribution network pressurized by a 12-V pump typically drawing 5–10 A, and usually operating in an on/off fashion, and an expansion vessel. The latter eliminates sudden changes in the flow and ram effects. It is a source of comfort, but also of problems: space taken up, maintenance, etc. New models of electric pumps make it possible to dispense with the expansion vessel. Instead, a little electronic circuit takes care of running the pump according to the flow. No expansion vessel, the pump makes less noise, its consumption drops, and the flow is modulated gently. Everything's going swimmingly! Except the price, that is. These new pumps with their encapsulated regulation circuit work out almost twice as expensive as the ones they replace.

The author is not an electronics engineer by training, but rather by conviction: so he started looking for—and found—a simple, cheaper solution, which is shared with the Elektor audience here.

Let's first focus on S1 on the schematic diagram in **Figure 1**. Although shown as an ordinary switch, it's actually a pressure switch, controlled—as its name indicates—by the pressure of a fluid when it exceeds a certain threshold. So its job here is to power the circuit when the pressure in the water distribution circuit is too low, and to cut it off again once it has come back up. The (existing) pump M1 is controlled by IC1,

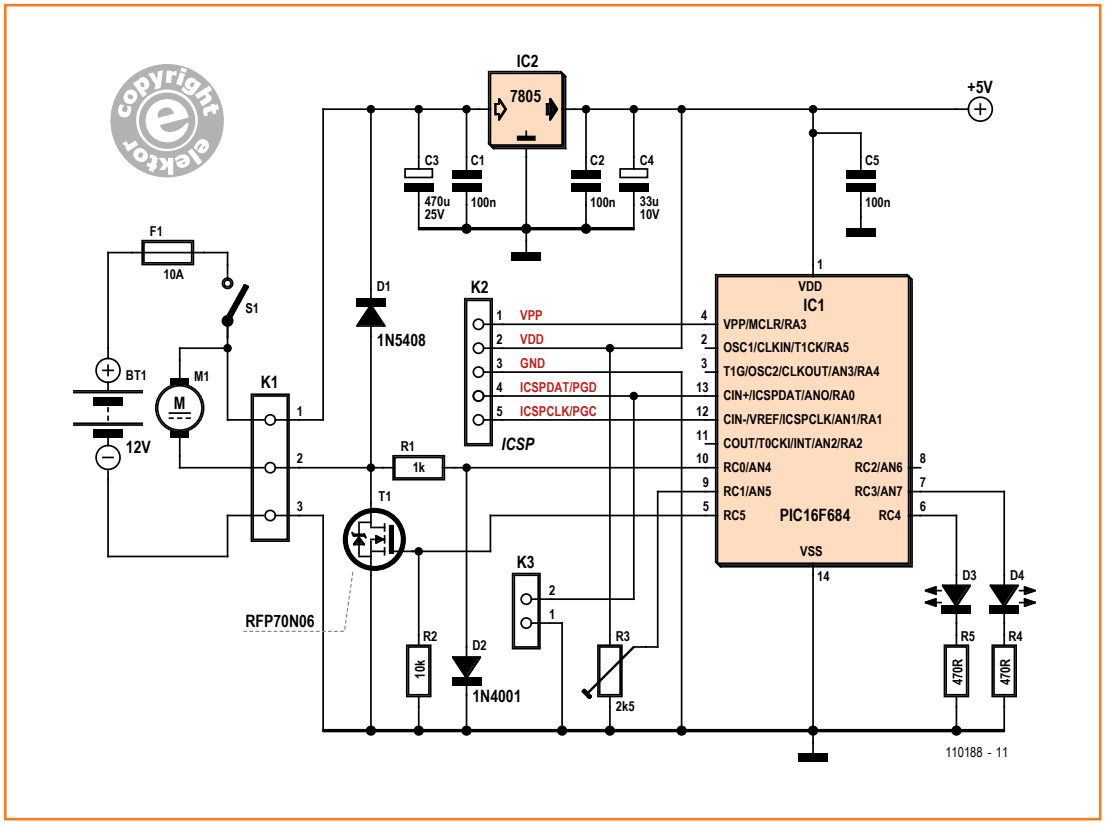


Figure 1.
The pulse width modulation control of the pump consists primarily of a PIC and a MOSFET.

which modulates the width of a 1-kHz signal (PWM) driving the gate of the power MOSFET T1. The latter, in series with the pump's negative power line, is designed to be driven directly by a microcontroller. The PIC16F684 compares the voltage across the FET's drain/source resistance and a reference voltage (set by the user with preset R3) on its pins 10 and 9 respectively, and aims to bring them together according to the difference measured. When the motor voltage is higher than the reference, the pulsewidth reduces; in the reverse situation, it increases. The PWM range is limited to between 55% (140/255) and 98% (250/255).

For debugging purposes, the PIC sends the instantaneous value of this ratio to the serial output (@ 2,400 baud).

If there is no flow in the water distribution circuit, the pressure rises to reach the actuatioon threshold of the pressure switch that opens S1 and the pump control circuit is disconnected.
Diode D2 clips the voltage peaks in order to (crudely) protect the comparator.
LED D3 is driven by the output of the analog comparator built in to the PIC, pins 9 and 10 of which are the inverting and non-inverting inputs, while LED D4 is controlled by the software.
The connector K3 is associated with debugging; it can be left out.

The regulation program in PICBasic Pro is as simple as can be, since the PIC has nothing else to do. The source code can be downloaded from the website [1]. The programmed PIC is avail-

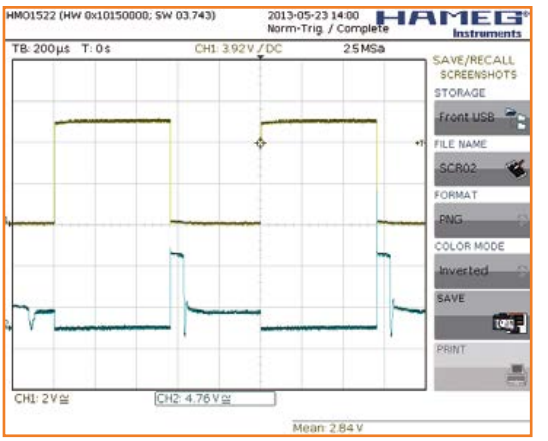


Figure 2.
The gate and drain signals of T1 (yellowish and green respectively).

COMPONENT LIST

Resistors
(0.25W)
R1 = 1kΩ
R2 = 10kΩ
R3 = 2.5kΩ preset
R4,R5 = 470Ω

Capacitors
C1, C2 = 100nF, 2.5mm
C5 = 100nF, 5mm
C4 = 33µF 10V, radial, 2mm
C3 = 470µF 25V, radial, 5mm

Semiconductors
D1 = 1N5408
D2 = 1N4001
D3, D4 = LED, red, 5mm
T1 = RFP70N06, N-MOSFET 70A 60V
IC1 = PIC16F684-I/P, programmed
IC2 = 7805

Miscellaneous
K1 = 3-pin PCB screw terminal block, 0.2'' pitch
K2 = 5-pin pinheader, 0.1'' pitch
K3 = not connected
14-way DIL IC socket
Pump used: Jabsco PAR-MAX 1.9 gal/min (7.2 l/min)
12 V 3 A

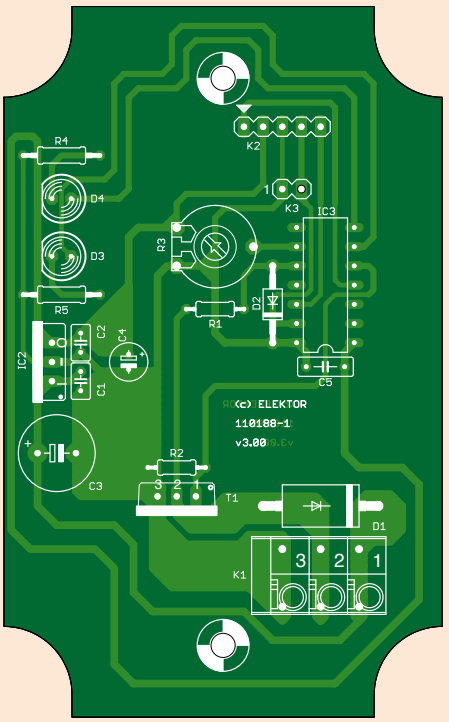
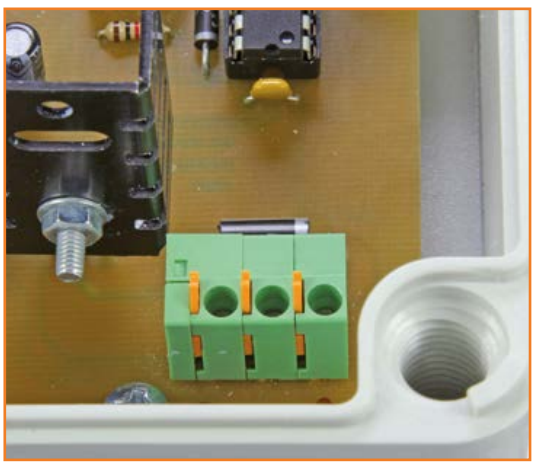


Figure 3.
The specific layout of the PCB is explained by the choice of a watertight case.

able from Elektor, as is a printed circuit board. Inserting the regulator in a pump's negative supply line in an existing installation ought not to cause any problems, but you do need to take special care with the installation, as it is going to be fitted in a potentially damp environment. This explains the specific design of the PCB, designed for one particular case (the PICCOLO ABS B 65 from Fibox), chosen for its ingress protection rating (IP 66/67); its contents are protected from powerful jets of water and even submersion down to about 3 ft. (1 m).

The pump used by the author is a 1.9 US gallon/min. 12-V, 3-A type. For K1, sailors will prefer a spring terminal block rather than a screw type, which risks coming loose in the presence of vibration.
Happy Sailing!

(110188)



Internet Link

[1] www.elektor.com/110188

CIRCUIT CELLAR / AUDIOXPRESS / ELEKTOR

New USB Counter / Timer DAQ Devices

Measurement Computing Corporation's two high-speed USB Counter / Timer DAQ devices are available with four or eight counter I/O. The USB-CTR Series feature high-speed pulse counting with a 48 MHz maximum input frequency. Resolution is programmable up to 64 bits, with an aggregate scan rate of 8 MB/s.

These new devices offer some of the most advanced counter / timer functionality available in USB DAQ today. Supported high-level input modes include totalize, period measurement, pulsewidth measurement, and timing measurement. Four independent PWM timer outputs and 8 digital I/O are also provided. Timer output channels can operate continuously or for a specified pulse count, with duty cycle and period changeable on the fly. Counter and digital channels can also be scanned synchronously. Flexible edge, level, direction, and debounce settings allow the counters to better adapt to user signals.

The new USB-CTR Series features 4 or 8 counter I/O; 48 MHz maximum input frequency; programmable resolution up to 64-bits per counter; aggregate scan rate of 8 MB/s. It supports the following counter input modes: Totalize; Period measurements; Pulsewidth measurements; Timing measurements.

The devices also feature debounce filter circuitry; 4 PWM timers; 8 digital I/O; Synchronous high-speed reads of digital and counter inputs. Software options include comprehensive support for Visual Studio® and Visual Studio® .NET, DASYLab®, and NI LabVIEW™.

The four channel USB-CTR04 is priced at US$359, and the eight channel USB-CTR08 is priced at only $429.

www.mccdaq.com (140323-II)

Microchip's GestIC® Technology Wins 11 Global Awards

Microchip Technology Inc., has been recognized by 11 global electronics industry publications for the product innovation and technology leadership of its patented GestIC® technology, which enables the next dimension in intuitive, gesture-based, non-contact user interfaces for a broad range of end products. The MGC3130 is the world's first electrical-field (E-field)-based 3D gesture controller, and it utilizes GestIC technology to provide low-power, precise, fast and robust hand position tracking with free-space gesture recognition.

In the Americas, the MGC3130 won five awards. It was handpicked by the editors of Electronic Design Magazine for their "Best of Electronic Design Awards," in the Digital category. EDN named the MGC3130 to their annual "Hot 100 List," in the Microcontrollers & Processors category. EE Times chose this product as a finalist in the ULTIMATE PRODUCTS: Sensors category of their "Annual Creativity in Electronics (ACE) Awards." Electronic Products Magazine bestowed a "Product of the Year Award" on Microchip's GestIC technology, while Design News awarded its coveted "Golden Mousetrap," in the Electronics & Test: Embedded Computing/Processing category.

In Asia, the MGC3130 received three prestigious honors. EE Times China Magazine selected it as a Product of the Year in the Microcontroller/Memory/Interface category of their "ACE Awards." EDN China bestowed an "Innovation Award" in the Embedded System: Microcontrollers category. Electronic Engineering & Product World China Magazine selected GestIC technology for an "Editors' Choice Award" in their Best Sensor Solutions category.

In Europe, the MGC3130 was recognized by three organizations. These included Europe's most prestigious electronics-industry honors, the annual "Elektra Awards," who named the MGC3130 as their Semiconductor Product of the Year in the Analog category. It also won France's "Electron d'Or Award," in the Sensors category. From Italy's "Innovation Awards" came a win for the MGC3130 in the Best Innovation Award category.

http://www.microchip.com/get/38XW (140232-III)

Shuntless Current Sensing from mA to kA

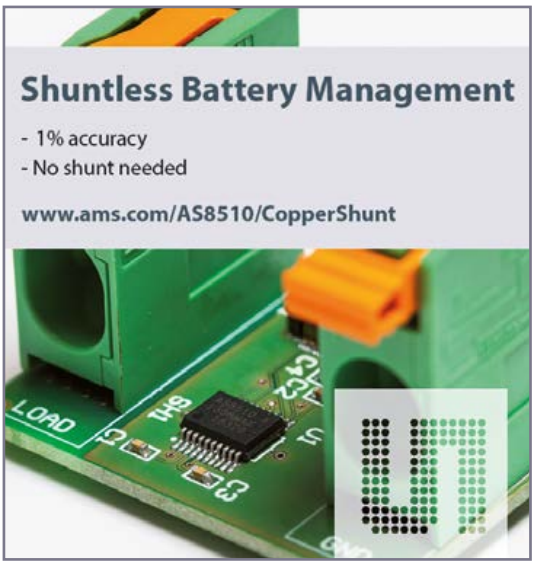
ams AG (has introduced a reference design board which measures current to an accuracy of ±1% by monitoring the voltage drop across a copper track on a PCB.

The development of the new technique, which uses the ams AS8510 data-acquisition front end, enables designers of Battery Management Systems (BMS) to reduce bill-of-materials cost by eliminating the precision shunt resistor normally used in current-sensing applications. A precision resistor with low temperature drift can typically cost as much as $1.50 in volume.

Accurate current measurement is an essential function in a BMS, which provides functions including monitoring of a battery's state of charge and state of health. The new reference design board from ams provides a blueprint for the current measurement function in a BMS, and can be applied in e-bikes, pedelecs and other applications drawing current of up to 40A. The same design can also readily be adapted to measure currents of up to 100A using only the resistance of a PCB's copper track.

The new ams reference design takes advantage of the very high sensitivity and precision of the AS8510, an integrated data acquisition front end which provides two measurement channels.

One channel is used to measure current by sensing the voltage drop over a 10-mm section of a PCB track with a known resistance value and temperature coefficient. The other, matched channel measures the temperature of the copper track. This temperature measurement can be performed either internally by the AS8510, or by an external temperature sensor.



By applying a compensation algorithm developed by ams, the AS8510 can eliminate the effect of the variation in the resistance of the copper track over temperature. This means that it can produce current measurements accurate to ±1% over its entire operating temperature range (–40°C to +125°C) without the normal requirement for a precision shunt resistor with a low temperature coefficient.

www.ams.com/AS8510/CopperShunt (140116-V)

Dual 20 A/single 40 A Step-down Power Module

Intersil Corporation's ISL8240M is a dual 20A/single 40A step-down power module that sets a new standard for efficiency and power density in infrastructure and embedded computing applications. Capable of delivering up to 100 W of output power in a 2.9 cm^2 footprint, greater than 90 percent power efficiency, and industry-leading thermal performance, the ISL8240M provides designers with a turnkey DC/DC converter solution for high current applications.

Today's power systems for communications and computing infrastructure support high current loads from increasingly power hungry FPGAs, ASICs, and microprocessors. To supply these high current circuits, equipment makers often rely on discrete power solutions that are complicated, take up valuable real-estate and may have significant power output limitations. Intersil's power module family reduces design complexity and creates headroom in the power system, increasing flexibility while improving efficiency and thermal performance.

The latest addition to Intersil's complete family of power modules, the ISL8240M is pin-to-pin compatible with the company's popular ISL8225M 30A power module, expanding the output current to 40 A and delivering up to 100 W output power. This offers designers a high level of flexibility, enabling a single PCB design across platforms. The ISL8240M is also capable of delivering up to 240 A output when six modules in parallel operation are current shared, enabling the industry's highest total output current.

Key Features and Specifications:

- Wide input voltage range: 4.5 V to 20 V;
- Adjustable output range: 0.6 V to 2.5 V;
- Capable of providing two separate outputs up to 20 A per channel or a single 40 A output with current sharing up to 6 modules in parallel operation for 240 A;
- Achieves >90% efficiency (12 V_{in}, 2.5 V_{out}) at 40 A;
- Full suite of protection features: over-current, over-/under-voltage, and over-temperature monitoring.

The ISL8240M leverages Intersil's patented technology and advanced packaging techniques to deliver greater than 90 percent power efficiency and the best thermal de-rating performance in the industry, enabling the ISL8240M to operate at full power over a wide temperature range. The module also provides over-temperature, over-current and over-voltage protection, further enhancing the robustness and reliability of the solution.

www.intersil.com/products/ISL8240m (140231-VII)

SMT Test Points and Ball and Socket Connectors

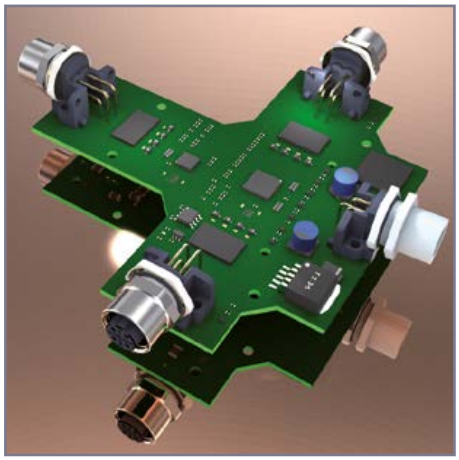
Vero Technologies has announced the introduction of a complementary range of SMT devices suitable for use with high-speed placement machines. Four industry-standard sized loop terminals, the 0603 (1.6 x 0.80 x 1.15 mm), the 0805 (2.0 x 1.25 x 1.45 mm) and the 1206 (3.2 x 1.60 x 2.0 mm) are initially available; all are compatible with industry-standard probes, clips and hooks. Additional designs and sizes will follow during the coming months. All sizes are available in a nickel-plated stainless steel construction; the largest size is also available in a copper alloy, allowing customers to choose the material that best fits their production process.

Also introduced is a 2 mm high, 1 mm diameter, ball and socket terminal. It requires a minimum pad size of only 0.080", 2.0 mm, enabling it to be placed by any pick and place machine in even the most congested board designs. The socket has less than 2 mΩ contact resistance, is rated at 3 A and offers gentle detachment. The matching 3.2-mm diameter socket has a ±30° angular movement before disconnection, giving good retention during use. All the new products are available ex-stock in units of 2000 or 2500 on standard 7" tape and reels for automated pick and place; they are also available in minimum quantities of 100 for smaller volume requirements.

Vero Technologies now offers a complete range of products suitable for all manufacturing processes, and all available from stock.

http://www.verotl.com/en/category/Surface+Mount+Technology (140231-VI)

PCB-POOL®: free 3D data for EAGLE PCB layout software



The PCB manufacturer Beta LAYOUT LTD is now introducing a virtual 3D EAGLE viewer in its PCB-POOL®. The unique tool is called "brd-to-3D" and works with a proprietary library. By uploading the EAGLE layout file *(*with more formats coming soon*)* the customer receives a 'free' comprehensive 3D package consisting of:

- Photo-realistic images of your printed circuit board and SMD stencil;
- a STEP file of the virtually assembled PCB;
- a PDF formatted 3D view that rotates freely allowing all angles of viewing

The brd-to-3D tool is a complement to CadSoft's EAGLE software that currently does not have a direct STEP-export.
Customers can also obtain a free laser-sintered 3D model of the assembled printed circuit board with the purchase of a PCB-POOL® prototype order.
For users not familiar with the 3D world the PCB-POOL® website provides prepared video tutorials that explain handling and processing of 3D files to the photo-realistic renderings.

www.pcb-pool.com/brd-to-3d www.pcb-pool.com/ppus/info_video_3d_freecad.html www.pcbpool.com/ppus/info_video_3d_simlab.html (140231-I)

Fairview Microwave Debuts New Lines of Tunable RF Filters

Fairview Microwave, Inc. announces their new lines of band pass and band reject tunable filters.
Fairview Microwave's tunable RF filters are commonly used in lab environments for testing many frequency bands including PCS, UHF, PMR, Tetra, LTE, and WiFi. The tunable band pass filters and band reject filters, also known as band-stop filters or band-rejection filters, are effective in band selection or frequency discrimination with high attenuation greater than 50 dB possible, allowing for excellent noise, harmonic and adjacent-band reduction. Fairview's tunable filters are designed with ruggedized aluminum casings with critical surfaces being silver plated, making them highly durable in a demanding environment. Fairview's latest release includes 6 bandpass filters capable of octave-band tuning from 125 MHz to 3 GHz depending on configuration, and a 5% pass band. They employ a 5-section tunable design and have a mechanical dial accurate within 1%. Also offered by Fairview, are 5 new band-reject models with octave-band tuning ranges from 100 MHz to 2 GHz depending on configuration, and a 1% reject band. These band reject filters use a 3-section design and have a mechanical dial tuner accurate to 0.5%. Each of the new bandpass filters and band reject filters from Fairview are rated to 50 Watts CW power handling and are designed with N female connectors.

www.fairviewmicrowave.com/rf-products/band-pass-and-band-reject-tunable-filters.html (140232-IV)

Hexadoku The Original Elektorized Sudoku

No *Project Generator Edition* is complete without a fresh Hexadoku puzzle, if only to take your mind off all that advanced hardware, software, soldering and programming on these pages. Find the solution in the gray boxes, submit it to us by email, and you automatically enter the prize draw for one of five Elektor book vouchers.

The Hexadoku puzzle employs numbers in the hexadecimal range 0 through F. In the diagram composed of 16 × 16 boxes, enter numbers such that **all** hexadecimal numbers 0 through F (that's 0-9 and A-F) occur once only in each row, once in each column and in each of the 4×4 boxes (marked by the thicker black lines). A number of clues are given in the puzzle and these determine the start situation.

Correct entries received enter a prize draw. All you need to do is send us **the numbers in the gray boxes**.

Solve Hexadoku and win!

Correct solutions received from the entire Elektor readership automatically enter a prize draw for five Elektor Book Vouchers worth **$70.00 (£40.00 / €50.00)** each, which should encourage all Elektor readers to participate.

Participate!

Before September 1, 2014,
supply your name, street address and the solution (the numbers in the gray boxes) by email to:

hexadoku@elektor.com

Prize winners

The solution of the May 2014 Hexadoku is: **18047**.
The €50 / £40 / $70 book vouchers have been awarded to: Rainer Klein (Germany), Alex Lo Furno (Italy), Luc Vandormael (Belgium), Thomas Eriksen (Norway), and Doug Brown (Australia).
Congratulations everyone!

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

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

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

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

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.

Hewlett Packard 71B Number Cruncher (1984)

A true tool for the trade

The HP-71B handheld computer caused something of a stir when Hewlett-Packard released it on February 1, 1984, some thirty years ago, with a rather hefty US$ 525 price tag. Inside a box measuring just 7.5 x 3.875 x 1 inch (19 x 9.7 x 2.5 cm) HP had squeezed not only a fully-featured and expandable BASIC computer, but also a professional and portable scientific calculator.

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

In today's terms the US$ 525 price would be around US$ 1190 [1]. What did you get for your money? The HP-71B was the first machine to use the then brand-new Saturn CPU, which proved very successful and formed the basis of other families of pocket calculators such as the HP-48. This four-bit(!) CMOS CPU was clocked at 640 kHz and run from a 5 V supply [2]. A single-line, 22-character LCD, a combined QWERTY and numeric keyboard, plenty of ROM (64 KB) and RAM (17.5 KB), and six expansion ports rounded out the system.

Software and some controversy

The device marked HP's second attempt, following the HP-75C/D, to enter the portable BASIC computer market which had hitherto been dominated by the Japanese companies Casio and Sharp: the Sharp PC-1500, in particular, is surely still remembered fondly by many older readers. It is interesting to note that the HP-75 design was developed by HP Labs based on the desktop HP-85 machine, whereas the HP-71B originated in the pocket calculator division based in Corvallis, Oregon.

Aside from the proprietary Saturn CPU, the distinctive features of the device lay mostly in its software and in the possibilities for expansion. The HP-71B had separate 'pocket calculator' and 'BASIC programming' modes, but the dynamic memory management system connected the two together. For example, variables computed in pocket calculator mode were also available for access from BASIC programs.

The calculator part of the device had a surprise (some might say shock!) in store for old HP hands who had come to know and love the reverse Polish notation (RPN) entry method of previous HP calculators: the HP-71B sported brackets and equals keys and used the same entry method as competing products from Texas Instruments. The entry method was called 'Algebraic Operating System' (AOS), and arguments raged among pocket calculator aficionados over the relative merits of the two methods. But HP would not be HP if the 71B's AOS did not offer something a little bit special. Over the course of the debates over the 'best' entry method it became apparent that each advantage of one of the methods was accompanied by a corresponding downside:

In AOS formulae are entered as they are written down or printed in a book, or as taught in school. However, this approach does not lend itself to displaying intermediate results and it is necessary to check the final answer carefully to make sure it is plausible. On the other hand, in RPN, an intermediate result can be shown at each step and mistakes can be spotted much more rapidly: for that advantage you pay the price of using a more machine-oriented rather than human-oriented input method. The HP-71B managed to bring together the best of both worlds in that as a formula was being entered using AOS it would try, as far as possible, to calculate and display intermediate results [3]. The example in **Table 1** illustrates how this works. One might have hoped that this elegant compromise would allow the two factions in the entry method debate to put aside their differences and live together in peace and harmony: that this did not happen was possibly down to the fact that programs written in RPN were generally a couple of instructions shorter than their AOS counterparts, and memory was scarce and costly.

There was another innovation, rather well hidden under the hood of the HP-71B: the device implemented the IEEE 754-1985 standard for displaying and performing mathematical operations on floating-point numbers, even though the standard had not yet been formally adopted. In contrast to the representation systems previously used, the standard allowed values such as 'infinity' and 'not a number' (NaN) to be displayed [4].

The built-in BASIC interpreter was very powerful, offering over 240 different commands. If those were not enough, the interpreter could be expanded using so-called 'Language Extension' or 'LEX' files. The dynamic memory management system meant that variables did not have to be declared before use, and arrays were allowed to grow and shrink as a program executed. Labels were used as the targets for jumps: other machines lacked this feature and used line numbers instead, which had great potential to cause frustration. A real-time clock and three timers meant that the BASIC interpreter could be used in laboratory applications involving measurement and control tasks: more on this in the section below on the expansion possibilities of the machine. Another handy little feature was the support for recursive subroutines, which considerably simplified the coding of such things as sorting algorithms.

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

Programs and data could be protected using a password. If set, the computer requested the password each time it is turned on and then turned itself off immediately if the correct one was not entered. The only way to remove the password protection was to take out the batteries, but this of course caused all the protected data to be lost [5].

Expansion possibilities

Six ports offered practically unlimited possibilities for expansion. Four of these ports were on the front edge of the device and were able to accept RAM and ROM cartridges (**Figure 1**). Two further ports, located under the device, were designed to accept a magnetic card reader and an HP interface loop module.

The optional magnetic card reader was certainly

1

Table 1. Calculate: 2 + (3 × 5)

Entry Method	Keystroke(s)	Display
AOS	2+(3*5)=	17
RPN	2 Enter	2
	3 Enter	3
	5 *	15
	+	17
HP 71B	2+(3*5)	2+(15)
	2+(3*5)=	17

the most important expansion option as it allowed the permanent storage of programs and data. With the reader installed, the user could pull a ten inch long, 3/8 inch wide magnetic strip, by hand(!), through the slot visible at the top right of the machine in order to write or read back data (see **Figure 2** and **Figure 3**). Each strip had a capacity of just 1.3 KB.

2

3

4

More spectacular in terms of the options opened up, however, was the HP interface loop (HP-IL) expansion module (**Figure 4**; module partly retracted). The HP interface loop was a network with a ring topology to which a huge range of different peripheral devices could be attached. You could connect almost any type of device you wanted, limited only by the size of your wallet: not for nothing was HP sometimes known at the time as standing for 'High Price'! Peripherals included printers, a plotter, a cassette drive, a floppy disk drive, a video interface, RS-232 and IEEE 488 interfaces, and a multimeter: all these could be connected over HP-IL and controlled from the computer. Furthermore, all these peripherals, with the exception of the video interface, could run on (rechargeable) batteries, and so the whole system could be used on the move. Whereas the HP-41 could only be fitted with one HP-IL module, the HP-71B could take two, operating simultaneously in different configurations: each module could be a master or a slave. The HP-71B could therefore run one interface loop as the master while participating in another loop as a slave. In combination with the HP 3468 multimeter and the IEEE 488 interface the HP-71B could be used as a data logger and processor.

These expansion possibilities, coupled with its portability and the excellent 'application packs' (ROM modules) that were available, meant that the HP-71B was the ideal computing environment for almost any task an engineer or scientist in the 1980s might be able to imagine.

(140062)

Web Links

[1] What's a dollar worth: www.minneapolisfed.org/index.cfm?&TC=1
[2] Features of the HP-71B: www.finseth.com/hpdata/hp71b.php
[3] http://h71028.www7.hp.com/enterprise/us/en/solutions/calculators-hp71b-math-machine.html
[4] IEEE 754: http://en.wikipedia.org/wiki/IEEE_floating_point
[5] Hewlett-Packard Journal July 1984: www.hpl.hp.com/hpjournal/pdfs/IssuePDFs/1984-07.pdf
[6] Wikipedia on the HP-71B: http://en.wikipedia.org/wiki/HP-71B
[7] The Museum of HP Calculators: www.hpmuseum.org/hp71.htm

Stacked!

By **Gerard Fonte** (USA)

There is a simple, but little known, software procedure that can improve the signal to noise ratio (or resolution) by any amount desired (theoretically). In practice, improvements by a factor of ten to one hundred can be accomplished by trading off time for resolution. In other words, you can change your microprocessor's 8-bit analog to digital converter to ten or twelve bits. I call the procedure data stacking and it's based on the concept of multiple measures.

Measure Twice

Suppose you have a fixed 10-volt DC signal with 10 mV of noise. The signal-to-noise (*S/N*) ratio is 1000. With a single measurement the best you can get is 10 volts ±0.01 V. No surprise there. Now take a second measurement and sum them. The DC voltage is doubled—obviously. But the noise is random and tends to partially cancel itself out. The increase in noise is only the square root of two.

This is the key concept behind this approach. The signal increases by the number of measures you take while the noise increases by the square root of the number of measures. Quite simply, the signal increases faster than the noise. Thus you can increase the *S/N* ratio by a factor equal to the square root of the number of measurements you take. If you take four measurements, your *S/N* ratio improves by two. If you take one hundred measurements, your S/N ratio increases by ten. If you take 10,000 measurements, your *S/N* ratio is 100 times better. That's a 40 dB gain in *S/N* or nearly 7 bits of improved resolution. So, with 10,000 measurements you can measure the 10 volts to ±100 µV rather than 10 mV. (Note that you may want to re-scale the result to bring the sum back to 10 volts; dividing by the number of measurements accomplishes this.)

Getting Past the Point

The real power and flexibility of this approach can be fully appreciated when dealing with multiple intervals rather than multiple points. However, this requires that the sample intervals be synchronized. For example, suppose you are using sonar to detect a small metallic object embedded in mud on the ocean floor. You can't find it because the return signal is weak and the mud randomly scatters the sonar return signal. But, by taking many signal returns, aligning them and then adding them, the signal can be brought out of the noise.

This approach has been used in many applications. Early brain studies measured "Evoked Potentials". This is where external electrodes are used to measure the neurological response to a sound or light stimulus. Normal brain activity swamped this very small signal. So they turned to multiple measures. Many sequential stimuli were presented to the subject and the results were added. The display showed the signal growing out of the noise floor. It was a remarkable sight.

This can be applied to communications as well. This requires that the transmitter and receiver be time-locked so that the receiver knows when the signal will be sent. That's not too hard to do nowadays. Let's say that the signal is deliberately sent at 20 dB below the atmospheric background noise and it's 0.1 seconds long. (You can pack a lot of information is 0.1 seconds.) It's undetectable. Let's also say that the signal is repeated at precisely the start of every second. After about three hours of stacking the signal it's now 20 dB above the background noise and it's very readable.

This is quite useful for deep space communications. It's also useful for secure data transmission. If the opposition can't find the signal they can't decode it. They would have to know the correct timing interval in order to detect it. And, of course, the interval doesn't have to be regular. Random intervals makes detection extremely difficult. (This has some similarities to spread-spectrum systems. Combining them with current encryption methods produces an incredibly secure data transmission system.)

Taking out the Trash

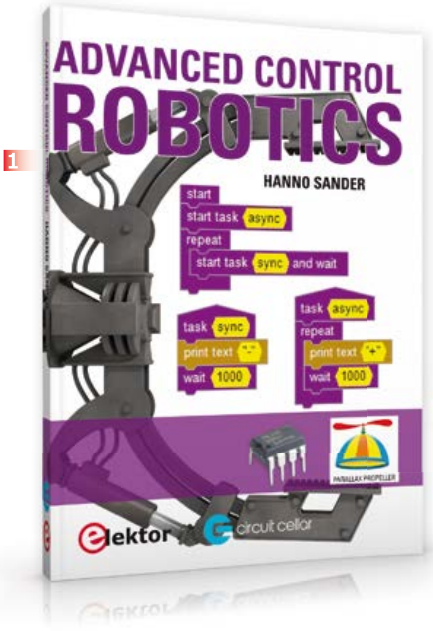
Working with intervals means that there are frequency considerations. Unlike noise, which is random, all frequency components not related to the sampling interval are reduced or eliminated. This is based on the fact that if you sum a full 360 degrees of any sine wave the result will be zero. These other frequencies are reduced faster than the noise. How much faster they are reduced depends on the frequency and the number of measures. So stacking is also a frequency selective filter.

But wait! There's more! If you sample and stack at precisely twice a particular frequency, that frequency is removed. For example, suppose there's a lot of 60 Hz noise in your signal (a common occurrence). If you sample and stack at 120 Hz, the 60 Hz noise goes away. The reason is very obvious when you stop and examine the situation. Sampling at 120 Hz means that the samples are 180 degrees out of phase for 60 Hz. Adding signals that are 180 degrees out of phase just wipes them out. (For best performance sum an even number of samples.) This is a powerful digital notch filter that is trivial to implement. It doesn't require difficult to define coefficients, or barrel shifters, or multiple registers or any of the usual complications associated with digital filters.

Summarizing, data stacking improves the *S/N* ratio, automatically reduces unwanted signals and can act as a notch filter simultaneously. Addition isn't just for Elementary School any more.

(140185)

NEW



1

2

3

4

20% DISCOUNT
for GREEN and GOLD Members!
www.elektor.com/july

Advanced Robot Technologies Made Easy

1 Advanced Control Robotics

It doesn't matter if you're building a line-following robot toy or tasked with designing a mobile system for an extraterrestrial exploratory mission: the more you know about advanced robotics technologies, the better you'll fare at your workbench. Hanno Sander's Advanced Control Robotics is intended to help roboticists of various skill levels take their designs to the next level with microcontrollers and the know-how to implement them effectively. Advanced Control Robotics simplifies the theory and best practices of advanced robot technologies. You're taught basic embedded design theory and presented handy code samples, essential schematics, and valuable design tips (from construction to debugging).

Approx. 170 pages • ISBN 978-0-96301-333-0
£34.95 • € 39.95 • US $54.00

The RPi in Control Applications

2 Raspberry Pi Hardware Projects

This book starts with an introduction to the Raspberry Pi computer and covers the topics of purchasing all the necessary equipment and installing/using the Linux operating system in command mode. Use of the user-friendly graphical desktop operating environment is explained using example applications. The RPi network interface is explained in simple steps and demonstrates how the computer can be accessed remotely from a desktop or a laptop computer. The remaining parts of the book cover the Python programming language, hardware development tools, hardware interface details, and RPi based hardware projects.

290 pages • ISBN 978-1-907920-29-5
£34.95 • € 39.95 • US $54.00

Lots of power with low distortion

3 Q-Watt Audio Power Amplifier

Elektor has a long history with audio power amplifiers. 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 (Q!) watts into 4 ohms with exceptionally low distortion thanks to the use of a special audio driver IC. Due to popular demand, this project is now supported by a semi-kit containing everything you need to build a stereo amplifier.

2 PCBs and all electronic parts
Art.#110656-72
£147.95 • € 169.95 • US $230.00

4 Android Breakout Board

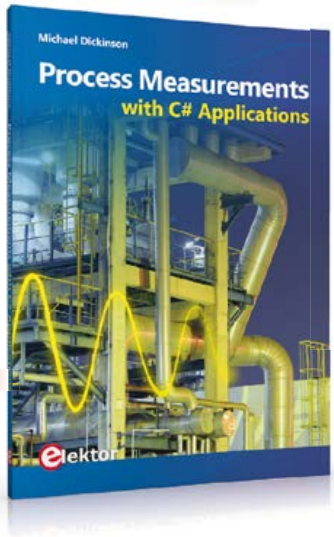
The FTDI FT311D is a flexible bridge that can interface your circuit to an Android smartphone or tablet. This Elektor Android Breakout Board offers options for seven digital outputs, four PWM outputs, asynchronous serial and I2C and SPI interfaces. The board is compatible with Android 3.1 (Honeycomb) or higher (Android Open Accessory Mode should be supported).

Ready-built module
Art.# 130516-91
£26.95 • € 29.95 • US $41.00

A Small Basic approach

5 PC Programming

There are many different PC programming languages available on the market. They all assume that you have, or want to have, a knack for technology and difficult to read commands. In this book we take a practical approach to programming. We assume that you simply want to write a PC program, and write it quickly. Not in a professional environment, not in order to start a new career, but for plain and simple fun... or just to get a task done. Therefore we use Small Basic. You will have an application up and running in a matter of minutes. You will understand exactly how it works and be able



to write text programs, graphical user interfaces, and advanced drivers.

194 pages • ISBN 978-1-1-907920-26-4
£30.95 • € 34.50 • US $47.00

All articles in Elektor Volume 2013

6 DVD Elektor 2013

This DVD-ROM contains all editorial articles published in Volume 2013 of the English, American, Spanish, Dutch, French and German editions of Elektor. Using the supplied Adobe Reader program, articles are presented in the same layout as originally found in the magazine. An extensive search machine is available to locate keywords in any article. With this DVD you can also produce hard copy of PCB layouts at printer resolution, adapt PCB layouts using your favorite graphics program, zoom in / out on selected PCB areas and export circuit diagrams and illustrations to other programs.

ISBN 978-90-5381-277-8
£23.95 • € 27.50 • US $38.00

The First Elektor Chip

7 E-Lock

The E-Lock chip allows you to connect your control system to the 'Network of Networks' and monitor and control it from anywhere on or around the globe using your computer, tablet or smartphone without having to worry about the security of the connection and in full assurance of protection against intruders.

Ready-built module
Art.# 130280-91
£96.95 • € 111.00 • US $150.00
See www.elektor.com/e-lock

An essential source of reference material

8 Process Measurements with C# Applications

Measurement is vital to the successful control of any process. This book introduces PC based measurement systems and software tools for those needing to understand the underlying principles or apply such techniques. Throughout the book, the C# programming language is used to give the reader immediate practical desktop involvement. C-Sharp has a wide support base and is a popular choice for engineering solutions. The basics of measurement and data capture systems are presented, followed by examples of software post-processing. Application examples are provided from a range of process industries, with reference to remote monitoring, distributed systems and current industrial practices.

144 pages • ISBN 978-1-907920-24-0
£23.95 • € 27.50 • US $38.00

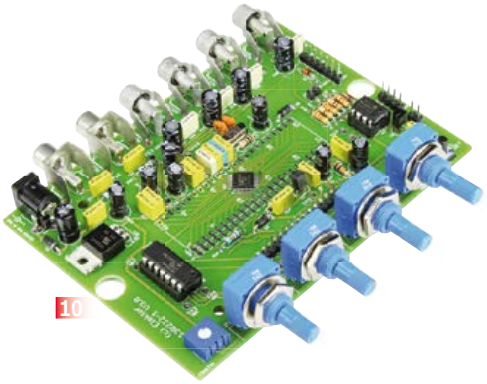
Circuits & Projects Guide

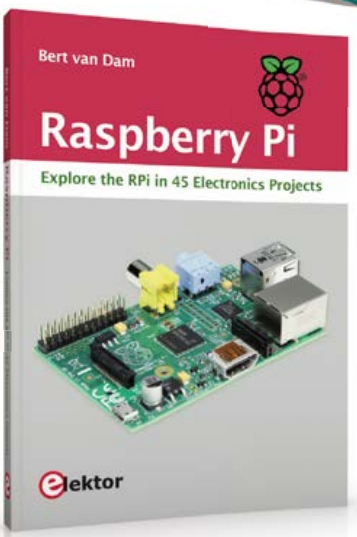
9 Arduino

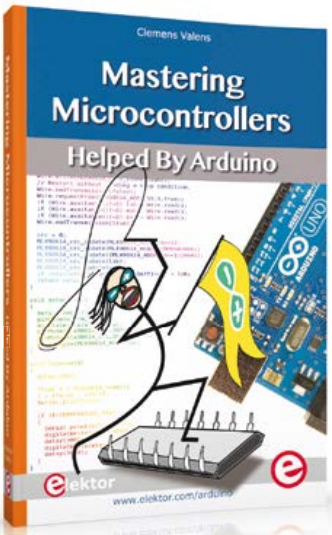
The Arduino user is supported by an array of software libraries. In many cases, detailed descriptions are missing, and poorly described projects tend to confuse rather than elucidate. This book represents a different approach. All projects are presented in a systematical manner, guiding into various theme areas.

In the coverage of must-know theory great attention is given to practical directions users can absorb, including essential programming techniques like A/D conversion, timers and interrupts—all contained in the hands-on projects. In this way readers of the book create running lights, a wakeup light, fully functional voltmeters, precision digital thermometers, clocks of many varieties, reaction speed meters, or mouse controlled robotic arms. While actively working on these projects the reader gets to truly comprehend and master the basics of the underlying controller technology.

260 pages • ISBN 978-1-907920-25-7
£34.95 • € 39.95 • US $54.00







Drag 'n Drop

10 ADAU1701 Universal Audio DSP Board

Always wanted to start out with DSPs, but afraid of the SMD? Here's the answer: Elektor's all-DIY Universal Audio DSP Board! Based on the Analog Devices ADAU1701 DSP chip and drag-and-drop software, this board will ease your way to becoming a sound craftsman, guaranteed maths-free.

Semi-kit:
All TH components, PCB with DSP preassembled
Art.# 130232-71
£63.95 • € 72.95 • US $99.00

110 Elektor Editions, Over 2500 Articles

11 DVD Elektor 2000 through 2009

This DVD-ROM contains all circuits and projects published in Elektor magazine's year volumes 2000 through 2009. The 2500+ articles are ordered chronologically by release date (month/year), and arranged in alphabetical order. A global index allows you to search specific content across the whole DVD. Every article is printable using a simple print function. This DVD is packed with ideas, circuits and projects that are ideal for any electronics enthusiast, student or professional, regardless of whether they are at home or elsewhere.

ISBN 978-1-907920-28-8
£77.95 • € 89.00 • US $121.00

Explore the RPi in 45 Electronics Projects

12 Raspberry Pi

This book addresses one of the strongest aspects of the Raspberry Pi: the ability to combine hands-on electronics and programming. No fewer than 45 exciting and compelling projects are discussed and elaborated in detail. From a flashing lights to driving an electromotor; from processing and generating analog signals to a lux meter and a temperature control. We also move to more complex projects like a motor speed controller, a webserver with CGI, client-server applications and Xwindows programs. Each project has details of the way it got designed that way. The process of reading, building and programming not only provides insight into the Raspberry Pi, Python, and the electronic parts used, but also enables you to modify or extend the projects any way you like.

288 pages • ISBN 978-1-907920-27-1
£34.95 • € 39.95 • US $56.40

Helped By Arduino

13 Mastering Microcontrollers

The aim of this book is not only to let you enter the World of Arduino, but also to help you emerge victorious and continue your microcontroller programming learning experience by yourself. In this book theory is put into practice on an Arduino board using the Arduino programming environment.

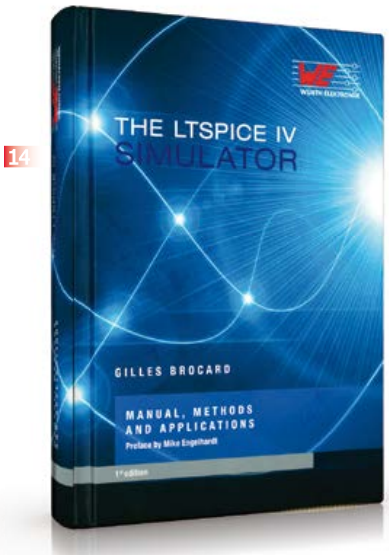
Having completed this fun and playful course, you will be able to program any microcontroller, tackling and mastering I/O, memory, interrupts, communication (serial, I²C, SPI, 1-wire, SMBus), A/D converter, and more. This book will be your first book about microcontrollers with a happy ending!

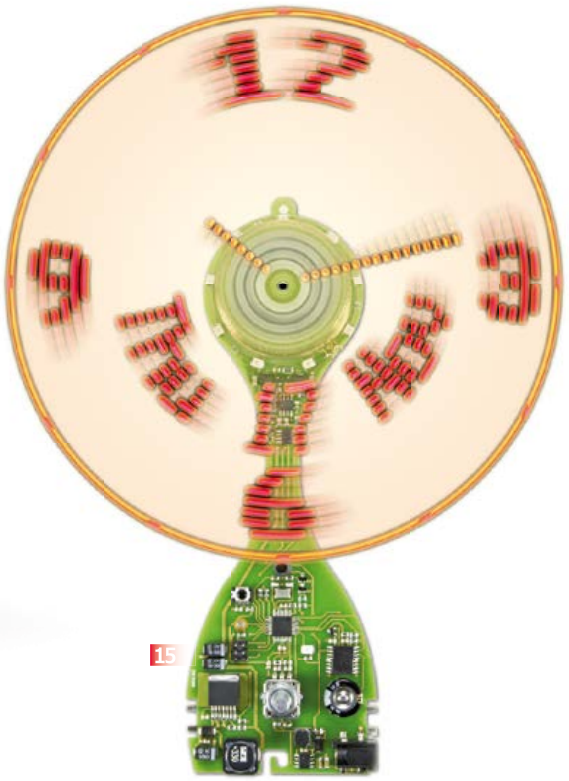
348 pages • ISBN 978-1-907920-23-3
£34.95 • € 39.95 • US $54.00

Manual, Methods and Applications

14 The LTspice IV Simulator

LTspice IV is one of the most extensive SPICE simulators currently available, with over 1.5 million users worldwide. It is completely free, allowing you to work with an unlimited number of nodes, components and simulations. This book is the ultimate guide to LTspice





IV. In a sturdy, hard-cover format, The LTspice IV Simulator describes the operation of the program, all available commands, the various editors, dealing with SPICE models, the use of non-linear components and more. The LTspice IV Simulator is more than just a manual. It also offers a variety of tips, methods and examples, all carefully illustrated using almost 500 drawings, diagrams and screenshots on high-quality paper. The book is designed so that it is suitable for both beginner and veteran SPICE users.

744 pages • ISBN 978-3-89929-258-9
£42.95 • € 49.00 • US $67.00

Time & Date Floating in the Air

15 UltiProp Clock

Electronics is never so fine as when it skillfully combines magic with physics, mechanics with software, imagination with thoroughness and precision, and a taste for beauty with good workmanship. This timepiece was designed to display the time and date in an original way—but we admit we did also design it to draw cries of amazement from the visitors who find it in our laboratory. We'll bet many of you will want to do the same in your own homes!

Module, ready assembled and tested
Art.# 120732-91
See www.elektor.com/ultiprop

Ideal reading for students and engineers

16 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. Undergraduate students should find the theory and the practical projects invaluable during their final year projects. Similarly, postgraduate students should be able to develop advanced DSP based projects with the aid of the book.

428 pages • ISBN 978-1-907920-21-9
£43.95 • € 49.90 • US $68.00

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

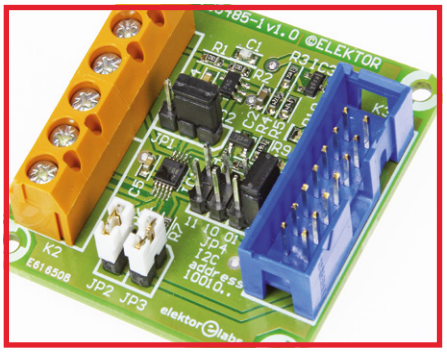
UK / ROW
Elektor International Media
78 York Street
London - W1H 1DP United Kingdom
Phone: +44 20 7692 8344
E-mail: service@elektor.com

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

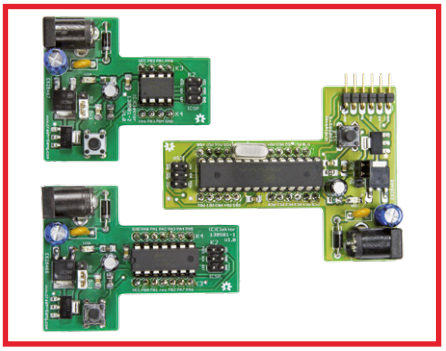
Switch-Mode Lab Power Supply

Announced in the June 2014 edition but held over till September due to lack of space: an ingenious circuit incorporating a custom-designed flat transformer with PCB track windings. The PSU has two LED displays for the current and voltage readout. The output is interruptible using a relay, and capable of sourcing up to 30 V, 1 A.



16-bit ADC module

Especially for low frequency data logging applications Elektor Labs designed a small circuit with an A/D converter type ADS1115. The board has an EEC connector for easy linking to the Elektor Linux Board, the Multifunction Xmega Board and the Elektor Arduino Shield. A special C library ensures that the converter can be easily included in any software for your own projects.



Prototyping with TeeBoards

When designing your own circuits incorporating a microcontroller you may suddenly find that breadboard awkward with its mass of connecting wires and jumpers. To solve the problem we designed TeeBoards—handy little PCBs accommodating a microcontroller and ready to plug onto a breadboard or your own circuit board.

Article titles and magazine contents subject to change, please check www.elektor-magazine.com for updates.
Elektor's September 2014 edition is processed for mailing to US, UK and ROW Members starting August 19, 2014.

PicoScope® PC OSCILLOSCOPES

For every application there's a PicoScope

Bandwidth from 5 MHz to 1 GHz • Sampling from 10 MS/s to 5 GS/s • Memory from 8 kS to 2 GS

2200A SERIES
The pocket-sized PicoScope

Channels: 2 + AWG
Bandwidth: 10 to 200 MHz
Sampling: 100 MS/s to 1 GS/s
Resolution: 8 bits
Buffer memory: 8 to 48 kS

MSOs
Mixed-signal analysis

Channels: 2 analog 16 digital + AWG
Bandwidth: 25 to 200 MHz
Sampling: 200 to 500 MS/s
Resolution: 8 bits
Buffer memory: 48 kS to 128 MS

3400 SERIES
High performance

Channels: 4
Bandwidth: 60 to 200 MHz
Sampling: 1 GS/s
Resolution: 8 bits
Buffer memory: 4 to 128 MS

PICOSCOPE 4824
8 channels, high resolution

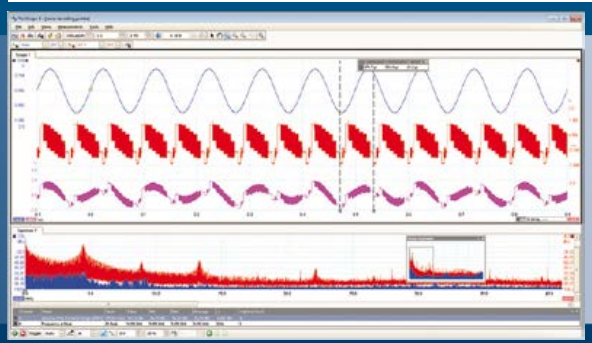
Channels: 8
Bandwidth: 20 MHz
Sampling: 80 MS/s
Resolution: 12 bits
Buffer memory: 256 MS

5000 SERIES
Flexible resolution

Channels: 2 or 4 + External trigger + Low distortion AWG
Bandwidth: 60 to 200 MHz
Sampling: 1 GS/s
Resolution: 8 to 16 bits
Buffer memory: 8 to 512 MS

6000 SERIES
Ultimate performance, USB 3.0

Channels: 4 + External trigger + AWG
Bandwidth: 250 to 500 MHz
Sampling: 5 GS/s
Resolution: 8 bits
Buffer memory: 256 MS to 2 GS

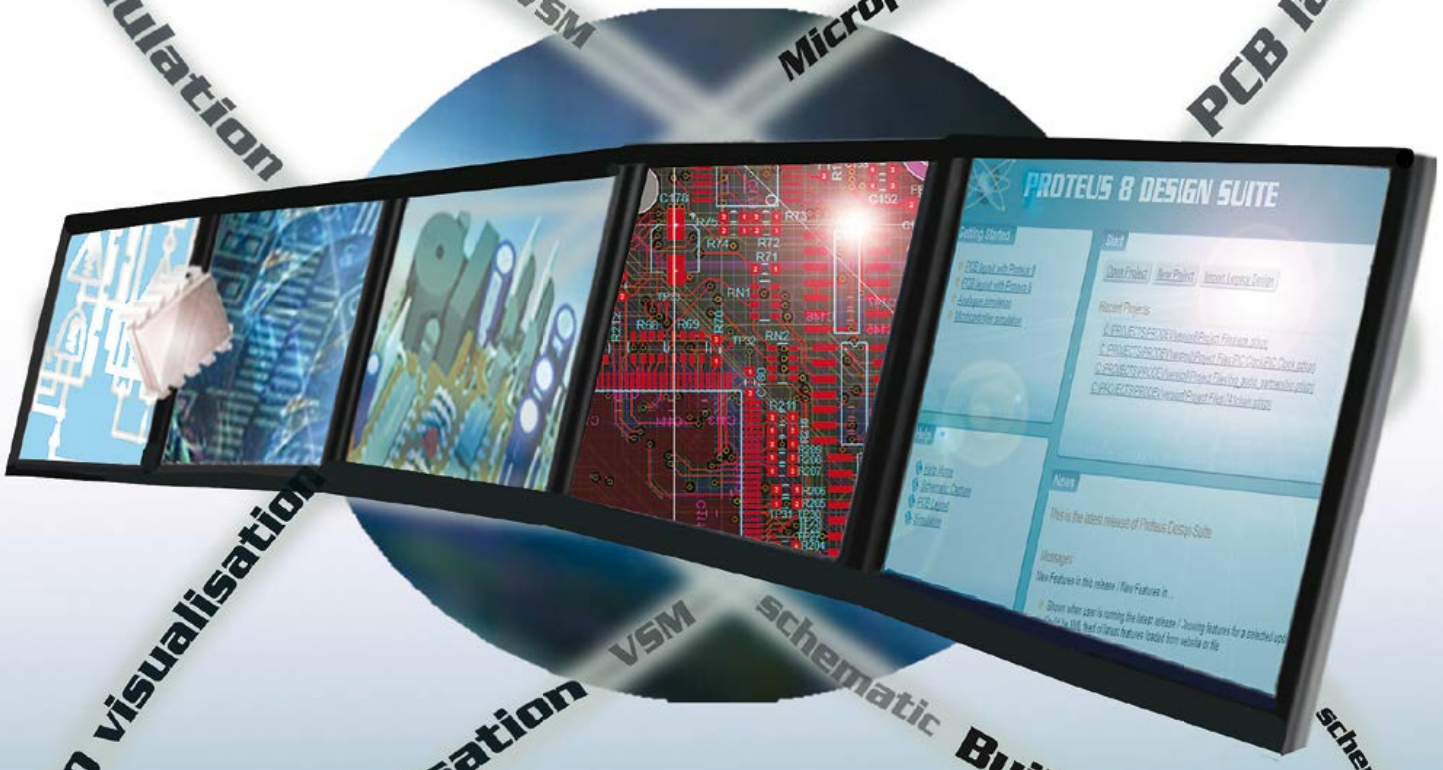


ALL MODELS INCLUDE FULL SOFTWARE AND 5 YEAR WARRANTY. SOFTWARE INCLUDES MEASUREMENTS, SPECTRUM ANALYZER, SDK, ADVANCED TRIGGERS, COLOR PERSISTENCE, SERIAL DECODING (CAN, LIN, RS232, I²C, I²S, FLEXRAY, SPI), MASKS, MATH CHANNELS, ALL AS STANDARD, WITH FREE UPDATES.

www.picotech.com/PS244

CAD CONNECTED

PCB Layout
Microprocessor Simulation
simulation
VSM
PCB layout
3D visualisation
VSM
schematic
3D visualisation
Built-in IDE
schematic



PROTEUS DESIGN SUITE VERSION 8

Featuring a brand new application framework, common parts database, live netlist and 3D visualisation, a built in debugging environment and a WYSIWYG Bill of Materials module, Proteus 8 is our most integrated and easy to use design system ever. Other features include:

. Hardware Accelerated Performance.
. Unique Thru-View™ Board Transparency.
. Over 35k Schematic & PCB library parts.
. Integrated Shape Based Auto-router.
. Flexible Design Rule Management.
. Polygonal and Split Power Plane Support.
. Board Autoplacement & Gateswap Optimiser.
. Direct CADCAM, ODB++, IDF & PDF Output.
. Integrated 3D Viewer with 3DS and DXF export.
. Mixed Mode SPICE Simulation Engine.
. Co-Simulation of PIC, AVR, 8051 and ARM MCUs.
. Direct Technical Support at no additional cost.

Version 8.1 has now been released with a host of additional exciting new features. For more information visit. www.labcenter.com

Labcenter Electronics Ltd. 21 Hardy Grange, Grassington, BD23 5AJ.
Tel: +44 (0)1756 753440, Email: info@labcenter.com
Registered in England 4692454

Labcenter Electronics www.labcenter.com