Edition 3/2015 | May & June 2015
www.elektor-magazine.com

elektor

LEARN • DESIGN • SHARE

AVR Programming

under Linux

Active DI Box

From unbalanced to balanced

PYE P87BQ Battery Radio

Found, restored, cherished

NFC Gateway

Smart Module with Serial Interface

In This Edition:
6 Labs Projects
4 Reader's Projects
7 New Modules & PCBs
1 Programming Course
2 Reviews
and lots more

ARM Microcontrollers for Beginners (3): Timers • Multilayer Boards • **DSO Nano V3 Pocket Oscilloscope** • **Codebender** • Input Protection • **Programming AVR micros under Linux** • 0.5 A/33 V LED Driver Module • **IoT with the WunderBar** • **TCA580 Integrated Gyrator** • Tips & Tricks • **HEF4000 Logic** • Model Aircraft Autopilot • **Musical Doorbell** • **Active DI Box** • **Basement Ventilation System** • Medication Alarm Clock • **BL600 e-BoB (2)** • USB-to-Multi-Protocol Serial Converter • **Arduino-Powered AM Transmitter** • **Nosing Around Hackaday** • **HP 400H VTVM Restoration** • Pye P87BQ Radio Find & Restore • **Hexadoku ... and more**

Edition 3/2015
Volume 41, No. 460 & 461
May & June 2015

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

Elektor Magazine, English edition
is published 6 times a year by

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

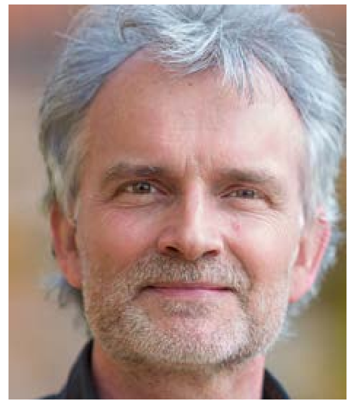
Head Office:
Elektor International Media b.v.
PO Box 11
NL-6114-ZG Susteren
The Netherlands
Phone: (+31) 46 4389444
Fax: (+31) 46 4370161

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

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

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

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



D-I-Y, the Extended Version

In this third edition into 2015 our new magazine formula dubbed Learn, Design, Share reaches Rev. B. Mod. 13b after a number of tweaks in the design department. That's the Graphics Design department for sure — no changes essentially in the way electronics gets fabricated by Elektor Labs and rolled out as products you can buy in our online Store. You can is meaning to say buying is an option; we continue to honor and support your freedom to go your own way in terms of parts, PCB, PCB artwork, and software if applicable Unsurprisingly our .com website is highly commercial and intended for those readers without the means and wherewithal to make everything themselves from individual parts. Our boards with SMD parts preassembled are the best of both worlds: if you do not want the hassle and anxiety of losing these tiny parts to the missus' vacuum cleaner, but do enjoy fitting half a dozen through-hole components with a 50-watt iron, the hybrid boards are good buy. I was delighted however to hear of a group of enthusiastic readers in Canada recently having etched their own double-sided boards, fitted the SMDs including the PLCC ICs and laughed at soldering leaded parts like .1" pinheaders and connectors, trimpots, and the odd radio transceiver module bought off EBay. Equally delighting I find the general rise in appreciation of the art of repairing and optimizing stuff, giving ex thrift shop and dumpster'ed electronic equipment a second lease of life, or a wholly different, unexpected use. Our magazine looks creative and educational overall, but 'reworking' or 'repurposing' stories have a presence too, although modest. For instance, this month we relive AM radio (page 97), we hack a certain Jobs product (page 110), confess our electronics design errors (page 113), and restore two relics, an HP VTVM and a Pye battery radio. All articles were produced by authors and editors with DIY in their blood and wanting to extend electronics DIY with four new activities minimum: Repair, Repurpose, Rework, Recreate, where the latter I hope retains it double meaning. Please send me instances of your DIY^PLUS activities, but limited to electronics please.

Enjoy reading this edition

Jan Buiting, Editor-in-Chief

THIS EDITION

Volume 41 – Edition 3/2015
No. 461 & 462 May & June 2015

LEARN DESIGN SHARE

LEARN DESIGN SHARE

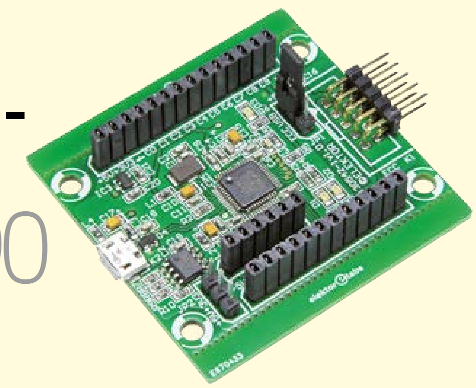
Active DI Box

48

This circuit adapts an unbalanced audio signal source (such as a guitar pickup) to an amplifier or mixing desk with a balanced input. It also has a rotary switch that allows the input signal attenuation or gain to be set to several different levels.

USB-to-Multi-Protocol Converter

90



DSO nano V3 Pocket Oscilloscope

20

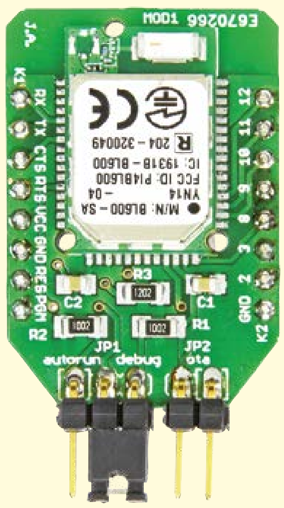
Basement Ventilation System

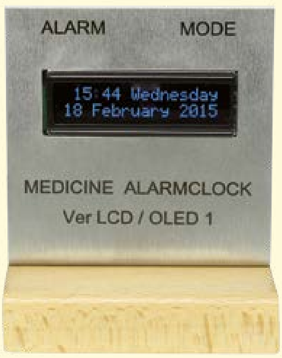
60

Using ChipCap2 sensors and a Platino microcontroller board we build an advanced ventilation system for damp cellars.

BL600 e-BoB Part 2

85



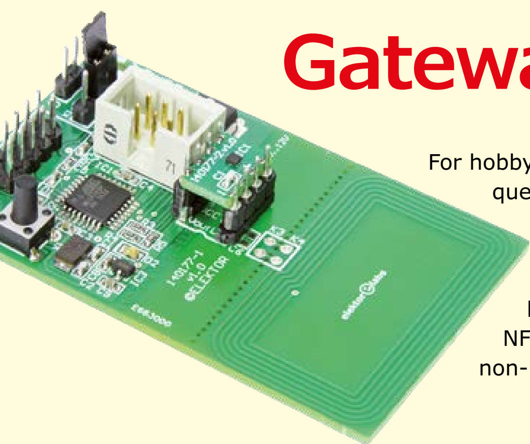


72

Medication Alarm Clock

NFC 'e-BoB' Gateway

78



For hobbyists the most interesting question is not what the smartphone looks like, but how this marvel of microelectronics can be hooked up to other systems. NFC provides a novel method of non-invasive connectivity.

Pye P87BQ Find & Restore

122

elektor magazine

LEARN | DESIGN | SHARE

NEXT EDITION

RS-232 Data Logger *
This nifty little circuit eavesdrops on an RS-232 link and sends data electrically isolated through a USB port to a terminal emulator running on your PC.

IR Remote Controlled Dimmer/Power Switch
This handy circuit with a PIC micro gives remote on/off and intensity control over a lamp or an electric heater, and is capable of learning IR codes.

Platino'd Pelco Control
Many security cameras on the market employ the Pelco protocol. An Elektor Platino and some dedicated software affords control over these cameras.

Edition 4 / 2015 covering July & August is published on June 15, 2015.
Delivery of printed copies to Gold members subject to transport.
Contents and article titles subject to change.
* regrettably this article could not be accommodated in the current edition as planned

Elektor Circuits &

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

56

Countries

246497

Enthusiastic Members

10

Experts &

Elektor Post

The e-inspiration weekly

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

www.elektor.com/newsletter

Elektor Community

Become a member, Green or Gold

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

www.elektor.com/memberships

Elektor TV

We're tubed too

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

www.youtube.com/user/ElektorIM

Elektor PCB Service

Boards at your service

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

www.elektorpcbservice.com

Elektor Labs

Learn, Design & Share

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

www.elektor-labs.com

Elektor Academy

Ride the learning curve

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

www.elektor-academy.com

Connections Guide

27
Authors

469
Publications

233461
Montly Visitors

11:03
APRIL 07 2015
Print Time

Elektor Magazine

Close to 1024 pages of surprising electronics a year

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

www.elektor-magazine.com

Elektor Web Store

Fill your shopping cart

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

www.elektor.com

Elektor Books & DVDs

E-Information Powerpacks

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

www.elektor.com

Become a member today!

GOLD	GREEN	FREE
€91.00 per year £79.95 / US $123.00	€68.00 per year £59.95 / US $92.00	
✔ Elektor Annual DVD	✖ Elektor Annual DVD	✖ Elektor Annual DVD
✔ 6x Elektor Magazine (Print)	✔ 6x Elektor Magazine (Print)	✖ 6x Elektor Magazine (Print)
✔ 6x Elektor Magazine (Digital)	✔ 6x Elektor Magazine (Digital)	✖ 6x Elektor Magazine (Digital)
✔ Access to Elektor Archive	✔ Access to Elektor Archive	✖ Access to Elektor Archive
✔ Access to Elektor.LABS	✔ Access to Elektor.LABS	✖ Access to Elektor.LABS
✔ 10% Discount in Elektor Store	✔ 10% Discount in Elektor Store	✖ 10% Discount in Elektor Store
✔ Elektor.POST Newsletter	✔ Elektor.POST Newsletter	✔ Elektor.POST Newsletter
✔ 25 Extra Elektor Projects	✔ 25 Extra Elektor Projects	✔ 25 Extra Elektor Projects
✔ Exclusive Offers	✔ Exclusive Offers	✔ Exclusive Offers
www.elektor.com/gold	www.elektor.com/green	www.elektor.com/newsletter

Welcome to LEARN

By Jens Nickel

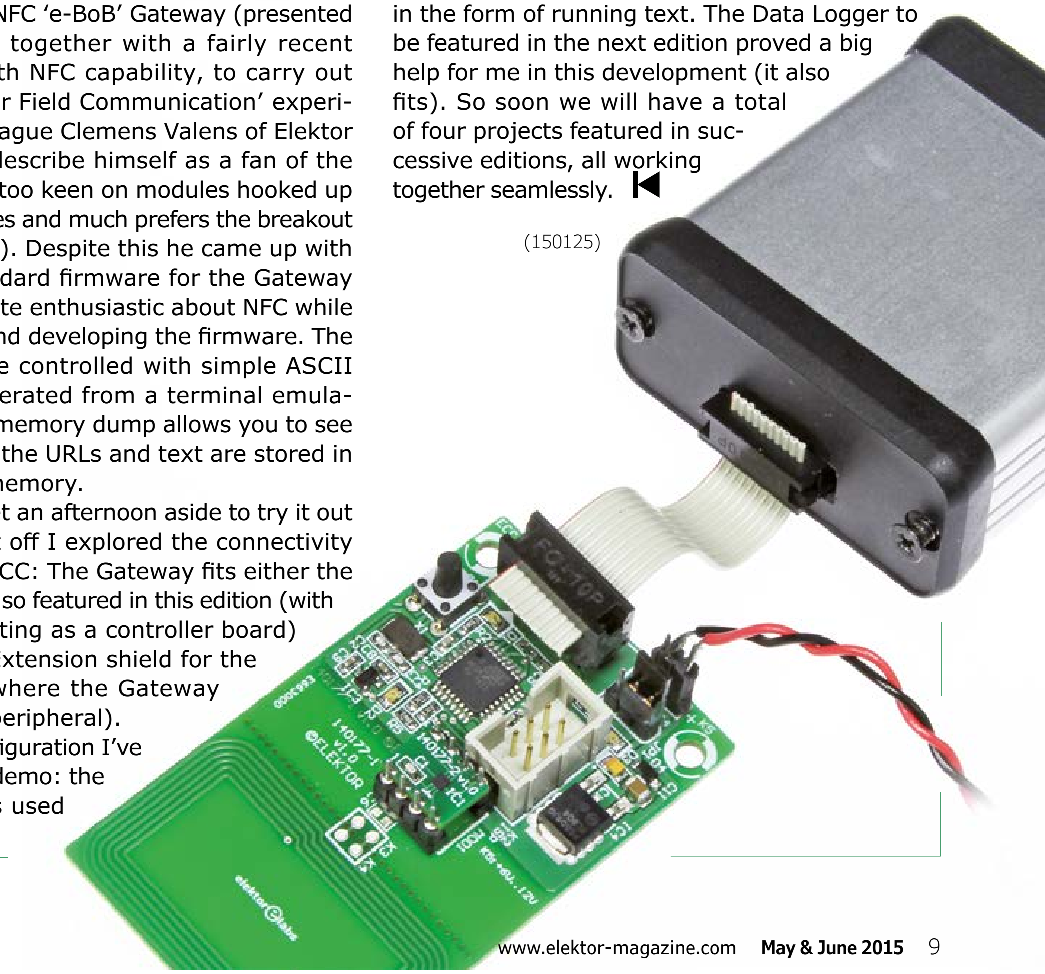
A (High)School Board

Trade fairs and exhibitions such as Embedded World are a great opportunity to chat with some of our readers and contributing authors and in general fish for good suggestions. Take this one for example: why don't we design a microcontroller board with an ARM Cortex controller and loads of 'neat stuff' on it: Ethernet, USB, an SD card slot, a good graphics display, pushbuttons, LEDs, DAC, PWM, CAN, expansion socket, etc…?
I must admit my first reaction to the suggestion was somewhat skeptical; a colleague was of the same opinion. We are living in a time where the Arduino and Raspberry Pi already have a huge following. In addition there are countless evaluation boards from semiconductor manufacturers who can afford to supply them at sub cost price. They almost always come bundled with a free software development environment and debugger. Is there really a market for yet another controller board that, at first guess, would need to be priced at around 100 dollars/euros?
Absolutely! Was the opinion of one of our authors and open-source gurus Benedikt Sauter. What it would need of course are loads of interesting example applications and a really comprehensive manual. What's missing today, particularly in the teaching of technology, is a well crafted, out-of-the-box system together with appropriate and well structured teaching resources. At the moment lecturers, teachers and support staff need to invest a lot of time researching and filtering appropriate resources from many different web sites to produce a teaching curriculum.
What's your opinion? Does the need exist? Let us know what you think, write to us at editor@elektor.com; subject [ARMsub100]!

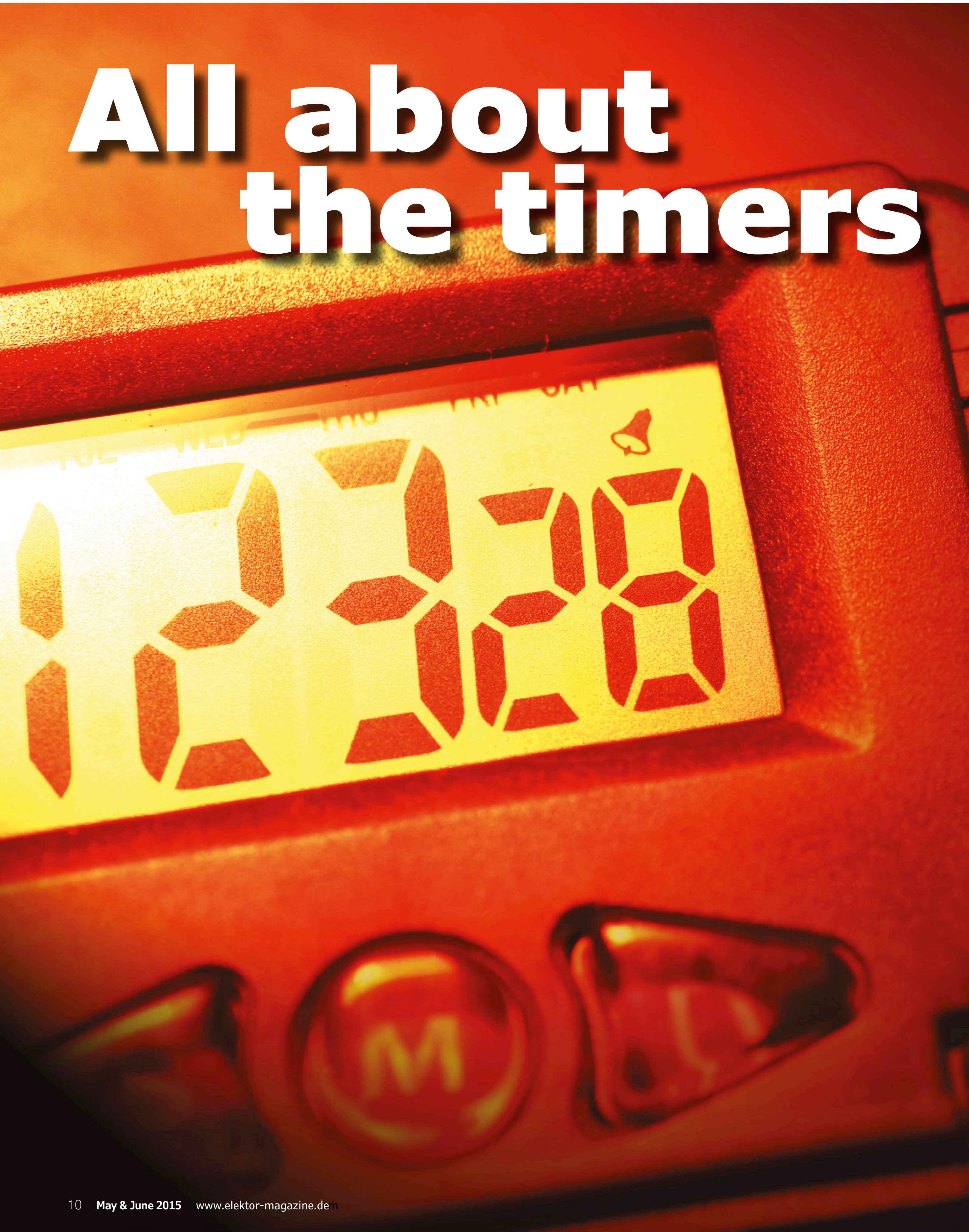
NFC Try-it-for-Yourself

The next ECC module is now finished. You will be able to use the NFC 'e-BoB' Gateway (presented in this edition) together with a fairly recent smartphone with NFC capability, to carry out some neat 'Near Field Communication' experiments. My colleague Clemens Valens of Elektor Labs wouldn't describe himself as a fan of the ECCs (he is not too keen on modules hooked up with ribbon cables and much prefers the breakout board approach). Despite this he came up with some neat standard firmware for the Gateway and became quite enthusiastic about NFC while programming and developing the firmware. The Gateway can be controlled with simple ASCII commands generated from a terminal emulator program. A memory dump allows you to see where and how the URLs and text are stored in the NFC tag's memory.
I managed to set an afternoon aside to try it out for myself. First off I explored the connectivity options of the ECC: The Gateway fits either the USB converter also featured in this edition (with the Gateway acting as a controller board) or our Elektor Extension shield for the Arduino Uno (where the Gateway functions as a peripheral).
For this last configuration I've written a small demo: the shield display is used to show the information stored in the NFC tag in the form of running text. The Data Logger to be featured in the next edition proved a big help for me in this development (it also fits). So soon we will have a total of four projects featured in successive editions, all working together seamlessly.

(150125)



All about the timers



From 8 to 32 bits: ARM Microcontrollers for Beginners (3)

By Viacheslav Gromov (Germany)

Accurate timing is very important in the digital world. For that reason we turn in this installment of our course to the SAM D20's timers, including its watchdog timer and real-time counter. In passing we shall also touch on other topics such as clock sources and external interrupts, as always accompanied by code excerpts illustrating the concepts in practice. Time (and tide) wait for no man, so let's get started!

In the previous installment of this series we looked (among other things) at the SERCOM communications module, although we did not explore all of the interfaces it has to offer. Before we can proceed to make proper use of them in the next installment we need to look at the various timers provided by the microcontroller.

For reasons of space we have not provided complete listings for all the experiments we will try below. The most important parts of the code will be found in the `Listings' text file in the download [1] accompanying this article, while complete Atmel Studio projects are also included in the ZIP archive. The hardware required is the same as that for the previous installment, although connection of the buttons and LEDs to the board is a bit different. So, first wire up the circuit shown in **Figure 1** to the SAM D20 Xplained Pro board, and then we can begin.

The WDT, clock sources and the EIC

The watchdog timer, or WDT, is one of the simplest timers in the SAM D20. Despite offering more functions than the watchdog timer present in many 8-bit microcontrollers (see **Figure 2**), it is relatively easy to get to grips with. The job of a watchdog timer is to cause a program to restart if it fails to complete its task within a set time period, for example if it gets stuck in a loop. As the block diagram shows, the WDT consists of a counter COUNT that can count up to 32768, and below that a number of registers with which the instantaneous counter value is compared. When the values match an early warning interrupt or a reset of the microcontroller is triggered. The program is therefore responsible for resetting the WDT regularly to avoid the reset being triggered.

The WDT in the SAM D20 has two modes:

- In normal mode a limit value from 8 to 16384 is chosen: if the counter reaches this value it is cleared and a reset is triggered. An early warning interrupt can also be set up in this mode, which triggers an interrupt shortly before the reset is due to occur.
- In window mode the program can reset the counter only within a predefined time window. If the program attempts to clear the counter outside this window, the microcontroller will be reset. The beginning of the window period is specified as a number from 8 to 16384; the duration of the window is set by another value, again in the range 8 to 16384.

The WDT is clocked by one of the eight outputs of the generic clock controller (GCLK) via a multiplexer. Each of the eight clock generators can be driven from one of a range of internal and external oscillator sources, or even from the output of another generator. Each generator includes a prescaler to divide down the input oscillator frequency.

The EIC, or external interrupt controller (see **Figure 4**) supports interrupts from 16 pins on the SAM D20. The signals on the selected pins are digitally filtered by the microcontroller and evaluated. Depending on the configuration, either a falling or a rising edge on each pin can trigger an interrupt or, for example, wake the microcontroller from sleep. Further options include the detection of pulses on input pins.

The EIC supports a further pin that can be configured to trigger a non-maskable interrupt. Depending on the priorities configured in the nested vectored interrupt controller (NVIC) this can interrupt the service of other interrupts.

First steps with the WDT

First we will try out the WDT in normal mode. Since its structure is relatively simple, it makes an ideal first timer project. And fortunately setting up the timer is made easy for us thanks to the convenient ASF libraries that we have discussed in previous installments.

The program 'First project with WDT' in the ZIP archive is very easy to understand. At the top, after the declaration of variables and the prototyping of functions, we find three callback functions. The first, called `watchdog_early_warning_callback()`, is called when the WDT's early warning interrupt is triggered and simply toggles LED0 on the board using a Port command.

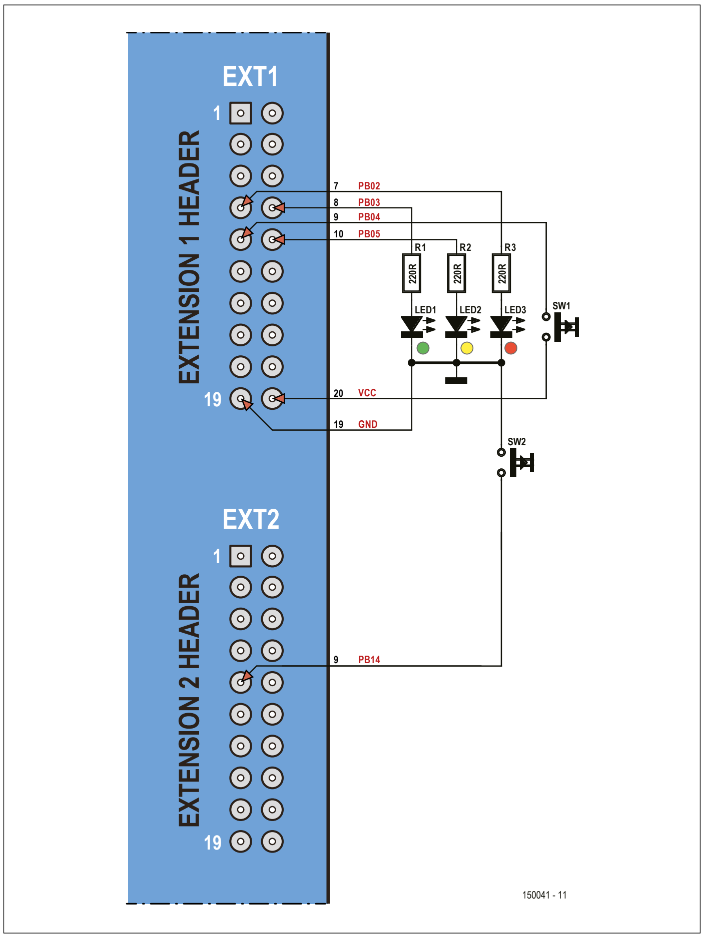


Figure 1. The circuit is a slight modification of the one we used in the previous installment.

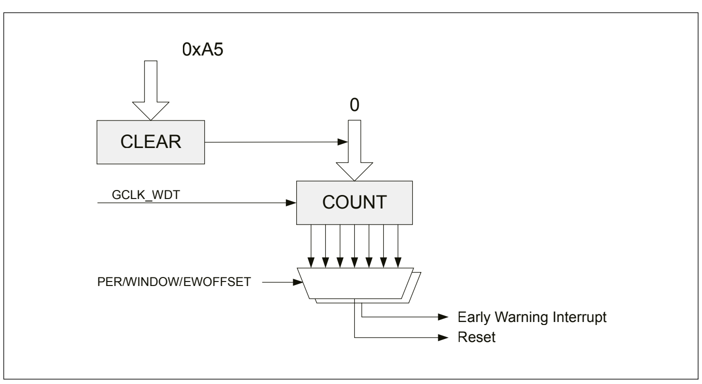


Figure 2. A block diagram of the watchdog timer (all screenshots and block diagrams courtesy Atmel).

The next function, called `extint4_detection_callback()`, is called when SW1, connected to interrupt pin 4, is pressed. The handler assigns the value 1 to boolean variable `i`. The final callback function, `extint14_detection_callback()`, is called when SW2, connected to interrupt pin 14, is pressed. It resets the variable `i` to zero. The next two functions are responsible for the important job of configuring the WDT and the EIC: these are shown in **Listing 1**.

The process of configuring the WDT follows the familiar pattern: first a configuration structure, in this case `config_wdt`, is created, and initialized with default values using `wdt_get_config_defaults(&config_wdt)`. Then any necessary elements of the configuration structure are modified. First the variable `always_on` is populated with the value `false` (if we set it to `true` then the WDT configuration is locked and it can only be reconfigured when the device is next switched on). The structure element `clock_source` is then set to select the desired clock source (GCLK generator 4 in this case) and then the timeout and early warning periods are set, here to 8192 and 4096 clock cycles respectively. Since the early-warning value is exactly half the timeout value, the LED will flash with a 50 % duty cycle. Finally, at the end of the function we transfer the values in the configuration structure into the watchdog timer with the command `wdt_set_config(&config_wdt)`.

The next function configures the EIC in a similar fashion. First the configuration structure `config_extint_chan` is created and initialized with default values. There follow two similar blocks of code. In the first the elements of the configuration structure specifying the pin, multiplexer, pull-up or pull-down resistors, and interrupt detection criterion for pin PB04 (which is connected to button SW1) are set. Because this button is active high a pull-down resistor is required to ensure a low level on the input when the button is not pressed. The detection criterion is a rising edge (from low to high) on the input. At the end of the block the new settings are transferred to the external interrupt controller with the command `extint_chan_set_config(4, &config_extint_chan)`. The arguments to the function are the number of the external interrupt (in this case 4) and a pointer to the configuration structure. The second block of code for configuring the EIC is built in the same way, but refers to pin PB14 (external interrupt 14) and SW2, the button connected to it. Since this button is wired to be active low a pull-up resistor is required in the configuration and the EIC must be told to detect falling rather than rising edges.

The functions that follow are called only once in the execution of the program and are responsible for registering and enabling the interrupt callback functions for the WDT and for the EIC. The first function, called `configure_wdt_callbacks()`, consists of just two lines of code:

```
wdt_register_callback(watchdog_early_warning_
callback,WDT_CALLBACK_EARLY_WARNING);
wdt_enable_callback(WDT_CALLBACK_EARLY_WARNING);
```

The first of these commands, which registers the callback function `watchdog_early_warning_callback()`, takes as its second parameter the event to which the callback function is to be attached. The second command, which enables the previ-

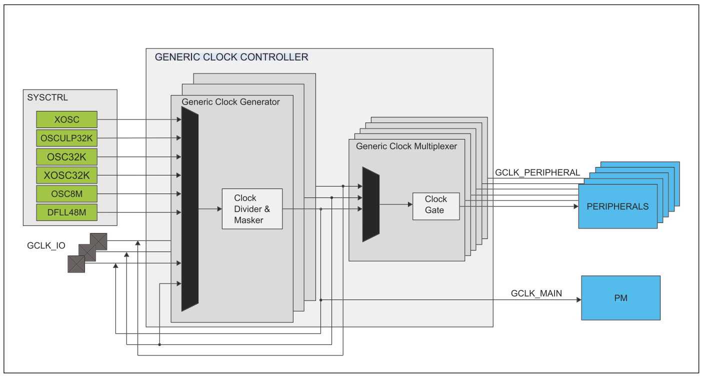


Figure 3. Block diagram of the generic clock generator.

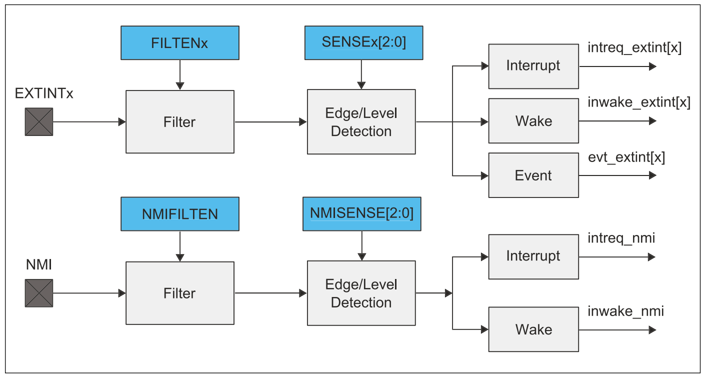


Figure 4. Block diagram of the external interrupt controller (EIC).

ously registered callback, takes the event as its only argument. The function for registering and enabling the EIC callback is similar in structure, but operates on two interrupts (EXTINT4 and EXTINT14). The commands used are similar to those above, but in this case we need to supply the number of the desired external interrupt. These configuration functions are called from the main program code, and then the output driving LED0 is set to a low level. In the infinite loop the variable i is continuously tested to see if it has been set to 1. When this is detected, the watchdog timer is reset using the function `wdt_reset_count()`.

Before compiling the program we need to take a look at the file conf_clocks.h (under src/config). The clock configuration set up in that file is easily changed: for this project we want to change the settings for GCLK generator 4, which we are using to clock the WDT (see **Listing 2**). The prescale value

Listing 1. Configuration functions for the WDT and the EIC.

```
void configure_wdt(void)
{
  struct wdt_conf config_wdt;
  wdt_get_config_defaults(&config_wdt);
  config_wdt.always_on = false;
  config_wdt.clock_source          = GCLK_GENERATOR_4;
  config_wdt.timeout_period        = WDT_PERIOD_8192CLK;
  config_wdt.early_warning_period = WDT_PERIOD_4096CLK;
  wdt_set_config(&config_wdt);
}

void configure_extint_channels(void)
{
  struct extint_chan_conf config_extint_chan;
  extint_chan_get_config_defaults(&config_extint_chan);

  config_extint_chan.gpio_pin = PIN_PB04A_EIC_EXTINT4;
  config_extint_chan.gpio_pin_mux = MUX_PA04A_EIC_EXTINT4;
  config_extint_chan.gpio_pin_pull = EXTINT_PULL_DOWN;
  config_extint_chan.detection_criteria = EXTINT_DETECT_RISING;
  extint_chan_set_config(4, &config_extint_chan);
  config_extint_chan.gpio_pin = PIN_PB14A_EIC_EXTINT14;
  config_extint_chan.gpio_pin_mux = MUX_PA14A_EIC_EXTINT14;
  config_extint_chan.gpio_pin_pull = EXTINT_PULL_UP;
  config_extint_chan.detection_criteria = EXTINT_DETECT_FALLING;
  extint_chan_set_config(14, &config_extint_chan);
}
```

Listing 2. Settings for GCLK generator 4.

```
#  define CONF_CLOCK_GCLK_4_ENABLE           true
#  define CONF_CLOCK_GCLK_4_RUN_IN_STANDBY   false
#  define CONF_CLOCK_GCLK_4_CLOCK_SOURCE     SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_4_PRESCALER        1024
#  define CONF_CLOCK_GCLK_4_OUTPUT_ENABLE    false
```

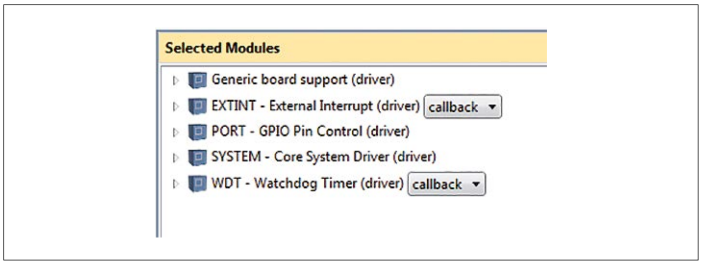


Figure 5. The set of libraries required for the watchdog project.

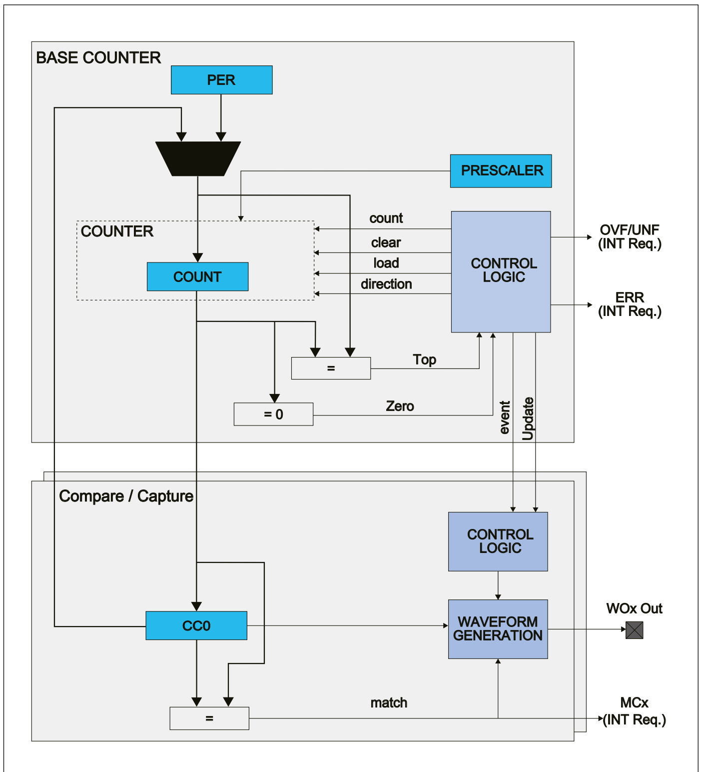


Figure 6. Structure of a timer/counter module.

is set to 1024, which means that the WDT counter is incremented at the slowest possible rate. `CONF_CLOCK_GCLK_4_ENABLE` is set to `true` in order to enable generator 4. Nearby in the listing you will also see that this generator is configured to use the 8 MHz main oscillator as its source: this oscillator is also used to clock the CPU. Make sure also that all the necessary libraries are linked in to the project using the ASF Wizard (see **Figure 5**).

We are now finally in a position to compile the program using Atmel Studio and download it to the board (using the command 'Start without debugging'). You should see that LED0 on the board blinks, as the early warning interrupt is first triggered and then, when the watchdog timer times out, the processor is reset causing the pin driving the LED to go low and hence the LED to be extinguished. However, when you press button SW1, the external interrupt is triggered which sets the variable `i` to 1. The WDT is then continuously reset in the infinite loop and hence generates no further interrupts. The LED therefore remains in its previous state. Pressing SW2 triggers an external interrupt that resets the variable `i` to zero. The WDT will no longer be reset and LED0 will start to blink again [2][3].

The timer/counter

The SAM D20 has eight standard timer/counters (TCs for short), which are very versatile. As we described in the first installment of this series a timer can be configured as an 8-bit timer or a 16-bit timer; furthermore, two timers can be combined to form a 32-bit timer. As you are probably already aware from experience with 8-bit microcontrollers, a general-purpose timer/counter can be configured to count clock pulses (and hence act as a period timer) and to generate an interrupt when the counter reaches its limit value or when a match is detected with the value in a compare register. Certain pins can also be configured to generate PWM outputs. The TCs in the SAM D20 can do all this and rather more besides. The structure of a TC in this microcontroller is shown diagrammatically in **Figure 6**. The counter block at the top is connected to a number of compare units below. When the counter value matches the value in a compare register the level at the output belonging to the timer and channel in question (for example, WO0 or WO1) changes, or an interrupt is triggered. Naturally the timer can be configured to count up or down, and to reset itself automatically on reaching its limit value. The architecture of the compare blocks makes the timer very flexible in use. For example, it can generate various types of PWM waveform, measure time by counting clock cycles, count events, generate various squarewave signals and much more. The timers can, if desired, run while the microcontroller is in sleep mode or during debugging. And, as can be seen from the block diagram, each timer includes a prescaler and can take its clock from any of the GCLK generators.

Our first timer project

We will now look at how the timer can be used in its counter mode using our existing circuit. The project will turn the LEDs on and off when the counter reaches its limit value or when it matches the value in one of the compare registers.

Open the project called 'The first project with the counter' in the ZIP archive. Taking a look a the main source file, you will see that the first lines of the program declare symbolic constants for the pins to which the three LEDs on the breadboard are connected. There follow the function prototypes, and then a structure of type `tc_module` called `tc_instance` is

Web Links

[1] www.elektor-magazine.com/150041

[2] www.atmel.com/Images/Atmel-42124-SAM-D20-Watchdog-Driver-WDT_Application-Note_AT03264.pdf

[3] www.atmel.com/Images/Atmel-42112-SAM-D20-External-Interrupt-Driver-EXTINT_Application-Note_AT03246.pdf

[4] www.atmel.com/Images/Atmel-42123-SAM-D20-D21-Timer-Counter-Driver-TC_Application-Note_AT03263.pdf

Listing 3. Configuration function for the TC.

```
void configure_tc(void)
{
  struct tc_config config_tc;
  tc_get_config_defaults(&config_tc);
  config_tc.counter_size = TC_COUNTER_SIZE_8BIT;
  config_tc.clock_source = GCLK_GENERATOR_1;
  config_tc.clock_prescaler = TC_CLOCK_PRESCALER_DIV1024;
  config_tc.counter_8_bit.period = 99;
  config_tc.counter_8_bit.compare_capture_channel[0] = 33;
  config_tc.counter_8_bit.compare_capture_channel[1] = 66;
  tc_init(&tc_instance, TC0, &config_tc);
   tc_enable(&tc_instance);
}
```

declared, which will subsequently be used to access the timer. Then come the familiar configuration functions, in which the three pins for the LEDs are configured as outputs. Three callback functions for the timer are then declared: `tc_callback_on_channel0()`, `tc_callback_on_channel1()` and `tc_callback_on_overflow()`. As their names indicate, the first function will be called when there is a match between the TC counter and the value in compare register 0, the second when there is a match between the TC counter and the value in compare register 1, and the third when the TC counter reaches its limit value. Each of the callback functions toggles one of the three LEDs using the command `port_pin_toggle_output_level(LEDx)`, where x can be G (for green), Y (for yellow) or R (for red).

Next is the configuration function for the TC, as shown in **Listing 3**. After the configuration structure has been declared and initialized with default values first the mode (8-bit), and then the clock source (GCLK generator 1) and prescaler ratio (1024) are set. Finally the values in the two compare registers and the counter limit are set. Notice that we have set these three values at equal intervals (33, 66 and 99) so that the LEDs will blink at a regular rate. The settings are then transferred to Timer/Counter 0 using the command `tc_init(&tc_instance, TC0, &config_tc)`, and finally the TC is enabled with `tc_enable(&tc_instance)`.

The function `configure_tc_callbacks()` simply registers the callback functions for the various events and then immediately enables them. The main routine just calls the configuration functions once each, and then the infinite loop is simply empty. In the file conf_clocks.h we need to enable GCLK generator 1 using `CONF_CLOCK_GCLK_1_ENABLE = true` and likewise, using `CONF_CLOCK_XOSC32K_ENABLE = true`, enable the external 32 kHz oscillator from which this generator receives its clock. Again the necessary libraries must of course be linked in using the ASF Wizard (see **Figure 7**).

We can now compile the project and observe the LEDs turning on in a regular sequence and then turning off in the same sequence.

Listing 4. TC configuration function to set up PWM generation.

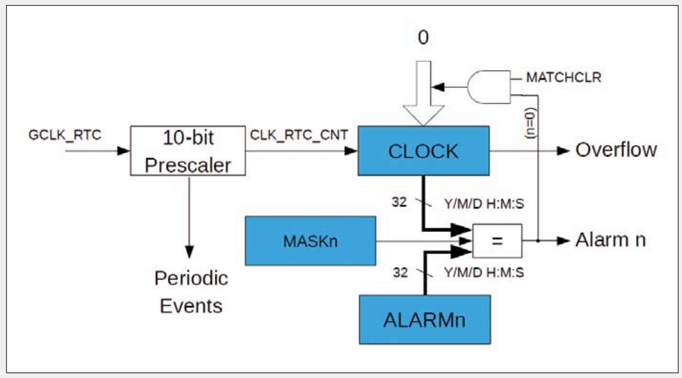
```
void configure_tc(void)
{
  struct tc_config config_tc;

  tc_get_config_defaults(&config_tc);

  config_tc.counter_size    = TC_COUNTER_SIZE_16BIT;
  config_tc.wave_generation = TC_WAVE_GENERATION_NORMAL_PWM;

  config_tc.counter_16_bit.compare_capture_channel[0] = 0xFFFF;
  config_tc.pwm_channel[0].enabled = true;
  config_tc.pwm_channel[0].pin_out = EXT1_PWM_0_PIN;
  config_tc.pwm_channel[0].pin_mux = EXT1_PWM_0_MUX;
  config_tc.counter_16_bit.compare_capture_channel[1] = 0xFFFF;
  config_tc.pwm_channel[1].enabled = true;
  config_tc.pwm_channel[1].pin_out = EXT1_PWM_1_PIN;
  config_tc.pwm_channel[1].pin_mux = EXT1_PWM_1_MUX;

  tc_init(&tc_instance, EXT1_PWM_MODULE, &config_tc);

  tc_enable(&tc_instance);
}
```

Generating PWM signals

We will now briefly look at how to generate PWM signals. Since in this project it is not appropriate to use callbacks, we will select the polled version of the TC library. We will also use the delay library (see **Figure 8**). The idea is to modulate the brightness of the green and the red LEDs on the outputs (WO0 and WO1) of TC6 simultaneously. The configuration function for the TC is as shown in **Listing 4**. After declaring and initializing the configuration structure we set first the size of the counter (16 bits) and then the mode of the TC (normal PWM in this case). Then the two PWM channels are set up. Both have a limit value of 0xFFFF (65535 decimal) and each is connected to its output pin. Finally we load the configuration into TC6 (under the name `EXT1_PWM_MODULE`) using the command `tc_init(&tc_instance, EXT1_PWM_MODULE, &config_tc)` and `enable it with tc_enable(&tc_instance)`. TC6 can subsequently be accessed by referencing `tc_instance`.

Within the infinite loop in the main routine the two compare registers for the PWM channels are continuously modified in a for-loop, using a command of the form `tc_set_compare_value(&tc_instance, TC_COMPARE_CAPTURE_CHANNEL_0, value)`. One of the compare registers is incremented, the other decremented. The higher the compare register value, the higher the mark-space ratio on the corresponding output and hence the brighter its LED. The greatest possible compare register

The real-time counter (RTC)



Like most 32-bit microcontrollers, and unlike most 8-bit microcontrollers, the main device on the SAM D20 Xplained Pro board has an on-board real-time counter. The RTC is a 32-bit counter with a 10-bit prescaler which can be supplied with a clock from a range of sources. It can be used in counter mode to count very long time periods or, as in a conventional RTC, in calendar mode to keep track of date and time. In counter mode it is also possible to compare the current count value with a compare register, just like in the TCs. In calendar mode the unit can be configured to generate an interrupt (the 'RTC alarm') at a preset date and time, expressed in 12-hour or 24-hour format. If the RTC is to be used in calendar mode it must be clocked at 1 Hz.

Two simple programs in the ZIP archive, 'The first project with RTC Counter' and 'The first project with the RTC calendar mode', demonstrate how the two modes of the RTC are used. Both work with the circuit described above.

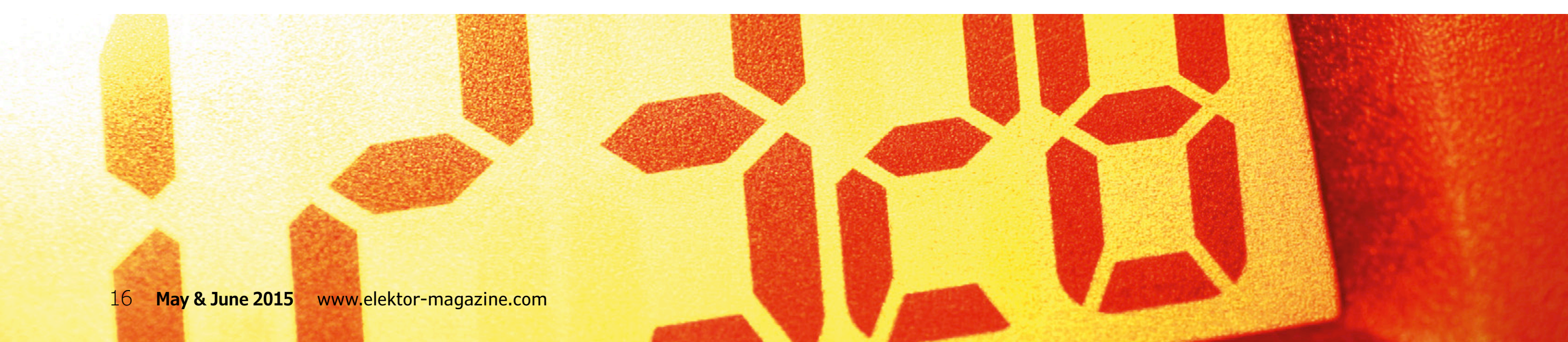
The first program is an LED flasher implemented using the RTC, arranged so that the red and green LEDs flash alternately with the yellow LED. First the pins which are connected to the LEDs on the breadboard are configured as outputs and then PB05 is set high. Then all we need to do is toggle all three pins at regular intervals: this is achieved in the RTC callback function which is called periodically, whenever the counter reaches its limit value. The configuration function follows the usual pattern (see the top of **Listing 5**). The two most important commands are in the middle, the first setting the prescaler division ratio to 1 and the second configuring the 32-bit RTC as a 16-bit counter.

After this function has been called the callback is registered and enabled, and then the main routine calls the configuration functions. The last command in the main routine before the empty infinite loop is rtc_count_set_period(&rtc_instance, 1000), which sets the limit value for the RTC to 1000. Since the RTC is programmed to receive a divided-down clock at exactly one kilohertz from GCLK generator 2, it will reach its limit value exactly once a second and so the LEDs change state at this rate.

The second project demonstrates the use of the RTC in calendar mode. This example uses only the green LED on the breadboard. An RTC alarm is used to cause the LED to change state once every second. The job of the RTC alarm callback is thus to toggle the state of the LED and then set up a new alarm exactly one second into the future. The configuration function for the RTC is shown in the lower part of **Listing 5**.

The main routine sets Alarm0, which triggers the callback function, to April 1 2015 at 00:00:01. Much further down, after the function to register and enable the Alarm0 callback and the calls to the numerous configuration functions, the RTC start time is initialized to 23:59:59 on March 31. The first alarm is therefore triggered two seconds after the program is started, and changes the state of the green LED. Subsequently the alarm is triggered once every second, and so the LED changes state at that rate.

For both projects GCLK generator 2 must be manually configured and enabled in the file conf_clocks.h. The callback-based version of the RTC is required, and as usual it must be linked in to the project using the ASF Wizard.



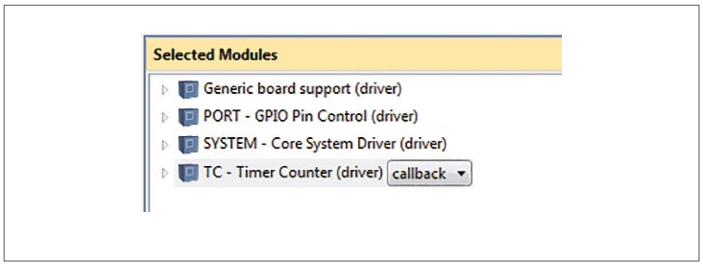


Figure 7. The libraries required for the timer project in ASF Wizard.

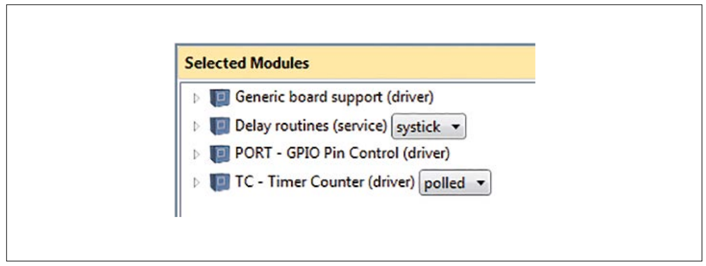


Figure 8. The libraries required for the PWM project.

value is 65535 and the lowest is zero. The second for-loop within the infinite loop does the same as the first but in the opposite direction, slowly dimming the green LED as the red LED becomes brighter. Since we make no other changes, TC6 is clocked from main clock generator 0, which does not need to be enabled specially as it also provides the clock to the CPU. We can now compile the program and observe the LEDs [4].

With that we come to the end of this relatively involved part of the course, which has covered one of the most complex peripheral elements in the microcontroller. Later, when we come to look at the event system, we will discover further functions of the timer; until then, we hope you have fun experimenting with and modifying the programs described here!

(150041)

Listing 5. The real-time counter (RTC).

```
void configure_rtc_count(void)
{
  struct rtc_count_config config_rtc_count;
  rtc_count_get_config_defaults(&config_rtc_count);

  config_rtc_count.prescaler           = RTC_COUNT_PRESCALER_DIV_1;
  config_rtc_count.mode                = RTC_COUNT_MODE_16BIT;
  rtc_count_init(&rtc_instance, RTC, &config_rtc_count);
  rtc_count_enable(&rtc_instance);
}
...
```

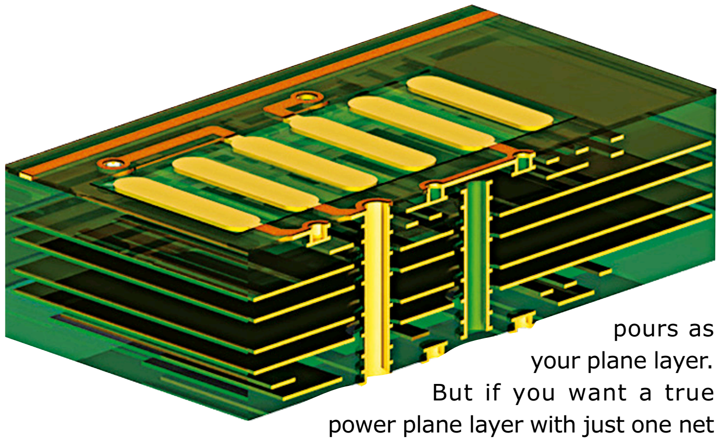
```
void configure_rtc_calendar(void)  //configure the RTC
{
  struct rtc_calendar_config config_rtc_calendar;
  rtc_calendar_get_config_defaults(&config_rtc_calendar);
  config_rtc_calendar.clock_24h = true;

  alarm.time.year      = 2015; //first alarm date
  alarm.time.month     = 4;
  alarm.time.day       = 1;
  alarm.time.hour      = 0;
  alarm.time.minute    = 0;
  alarm.time.second    = 1;

  config_rtc_calendar.alarm[0].time = alarm.time;
  config_rtc_calendar.alarm[0].mask = RTC_CALENDAR_ALARM_MASK_YEAR;

  rtc_calendar_init(&rtc_instance, RTC, &config_rtc_calendar);
  rtc_calendar_enable(&rtc_instance);
}
```

DesignSpark Tips & Tricks

Day #19 (final): Multilayer Boards

By **Neil Gruending** (Canada)

So far we've had a lot of 2-layer board examples, but DesignSpark can handle much more than that!



All of the examples I've used in this series have been 2-layer boards for clarity, but DesignSpark can be used for boards with four or more layers too. I've mentioned this briefly in other installments this series, so today let's take a closer look at the different ways to make multiple layer boards in DesignSpark.

Planning the Layer Stackup

The first step when designing a multilayer board is to decide what kind of layers you will need. Layers can typically be a signal layer, power plane, split power plane or a mixed signal plane layer. A *signal layer* is one where you can route anything but by convention is generally intended to be used for signal nets and not power nets. A power plane layer is one that where only one power net is routed using a large area of copper over the entire layer. A *split power plane* is when multiple power nets are routed using large areas of copper that are used to make the board power connections. A mixed signal plane layer is a power plane layer that also has some signal traces routed on it but usually the extra routing is kept to a minimum.

DesignSpark supports all of those layer types with its electrical layers and it's how you use them that defines their type. For example, if you only route signal traces on a layer then it's considered to be a signal layer. But there's nothing stopping you from adding some power pours to create a mixed signal plane layer. You make a split power plane by using multiple copper pours as your plane layer. But if you want a true power plane layer with just one net routed on it, then you can set the layer to be a power plane.

Setting up the board layers

The easiest way to set the board layer configuration is using DesignSpark's "New PCB Wizard" which is shown in **Figure 1**. You can choose the number of layers that you want on your PCB with the topmost layer being layer 1. Then you can configure your power plane layers if you choose more than four electrical layers and you're happy with the stackup options presented to you. For example, if you choose a 4-layer board, you can make layers 2 and 3 power planes.

Once the wizard if finished it's easy to modify the layer stackup by going into the design settings (Settings ➡ Design Technology). The Layers tab lists all of the PCB layers and lets you edit them or add new ones. **Figure 2** shows an example where I used the wizard to make an 8-layer board with layers 2 and 3 as power planes. A power plane in DesignSpark is an electrical layer with a "Power Plane" bias and don't forget to associate a net to it.

Now is also a good time give each layer a descriptive name like "Top Signal" or "Ground Plane" so that it's obvious which

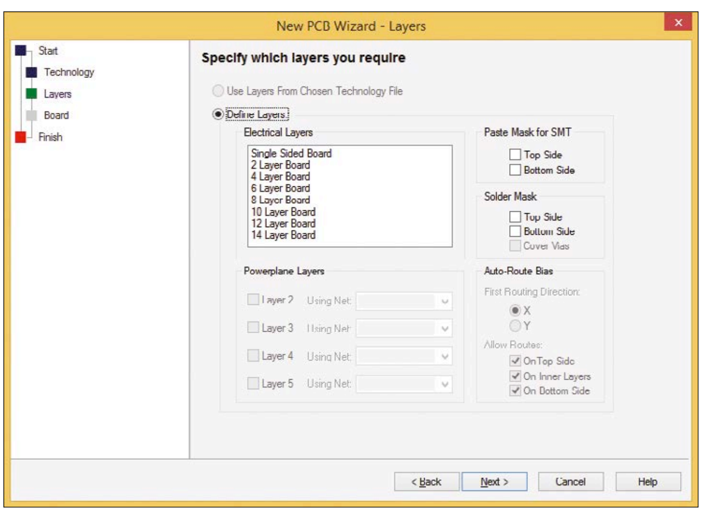


Figure 1. New PCB Wizard.

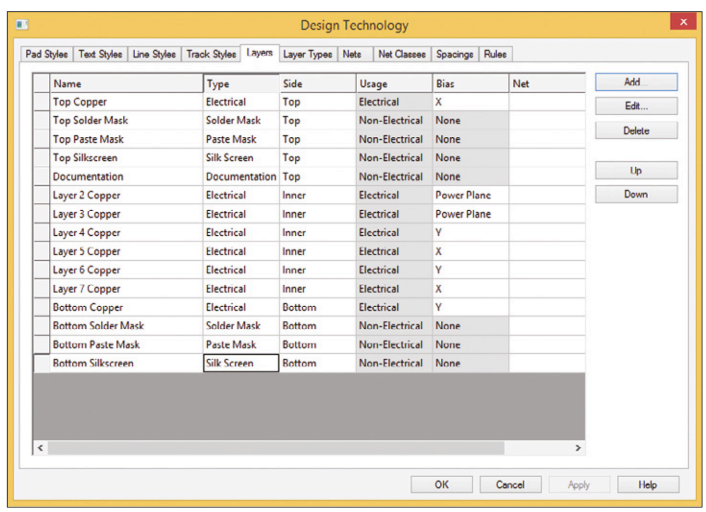


Figure 2. Design Technology Layers Window.

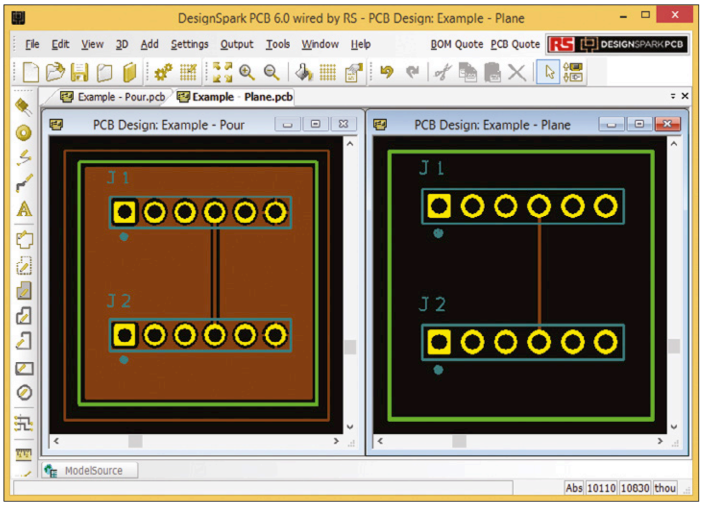


Figure 3. Pours *vs.* Planes.

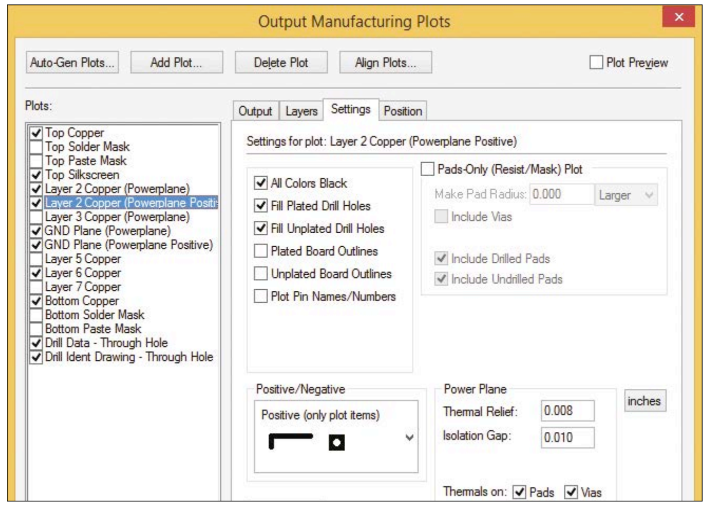


Figure 4. Gerber output options.

layers to use when routing the board. Normally you wouldn't route any traces on the plane layers but DesignSpark does allow traces on plane layers if you want them. The traces will be visible on the plane layer just like any other trace but when you generate the Gerbers DesignSpark will make sure the plane goes around all of the traces.

Compare pours to planes

Earlier I had mentioned that I prefer to use copper pours instead of planes even though pours can be a little more work. If you look at the example in **Figure 3**, the board on the left is using a pour for the ground plane and the board on the right is a power plane. Electrically they are both the same and in fact the Gerber output for the two boards is almost identical. The only difference is that you can see the final result in DesignSpark with a pour whereas you would need to generate and view the Gerbers with another application to see the final result for power planes.

Another nice thing about pours is that they are extremely flexible when it comes to power planes. For example, say you needed a small 1.8-V island in a 3.3-V power plane. While you could put a 1.8-V pour on the 3.3-V power plane layer, DesignSpark will give you a warning before letting you do that. The recommended method is to use a 1.8-V and 3.3-V pour and then set the pour order so that they pour correctly.

Generating Gerbers

Fortunately, generating Gerbers for multilayer boards is just like 2-layer boards. **Figure 4** shows the Output Manufacturing Plots window for our example 8-layer board where I have the Settings tab open so that you can see all of the settings that you can set for each layer. Normally you shouldn't have to change them but this is where you can change the plane isolation gap and thermal reliefs for power plane layers. You will also notice that not all of the copper layers are selected and that's because DesignSpark has only selected the layers that I used for this simple example. You also might notice that Layer 2 Copper and GND Plane have Powerplane and Powerplane Positive plots specified. Normally power planes are plotted as a negative image of the plane and that's what the Powerplane plots will do. However, I also put traces on those two layers which can't be represented by the negative image so DesignSpark needs to also output the positive image that contains just the trace and component pads. When you combine both of them you will get the total image for the board like in **Figure 5**. The black area inside the board is the plane except where the purple trace is routed. The dark blue color is the isolation gap between the plane and the trace. The isolation gap also goes around the board edge.

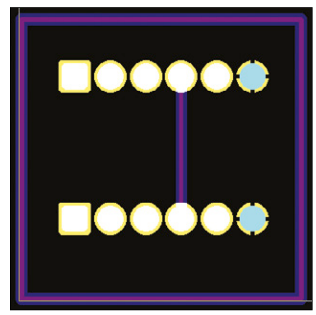


Figure 5. Ground Gerber plot.

But before you generate all of the Gerbers I would also recommend setting an output directory for all of the generated files. You do that by clicking on the Options button and then specifying the output directory for the Where Plot Files Are Written option. Then when you generate the Gerbers all of the output files will end up in that directory instead of in the main project directory.

Conclusion

'Today' we looked at how to set up a board layer stack up and then at different options for how to use planes. This is the closing installment in the series and I have enjoyed talking about DesignSpark PCB and what it can do. Hopefully this series has inspired you to give it a try on your next project!

(150023)

DSO Nano V3 Pocket Oscilloscope

Gadget or useful instrument?

By Harry Baggen (Elektor Netherlands Editorial)

When you see the DSO Nano V3 the first time, you right away think “I want it!” Measuring just 9 x 6 cm, this little device is a complete one-channel oscilloscope with buffer memory and a built-in signal generator. It’s also very affordable, with a price well under 100 dollars (or euros). We tested the DSO Nano V3 for a couple of days to see whether or not it is really useful for electronics professionals or enthusiasts as a portable instrument.

In addition to selling electronic components and equipment, the Chinese company Seeed Studio [1] develops their own products. The DSO Nano, a tiny one-channel oscilloscope with dimensions a good deal smaller than a typical smartphone, is probably their best-known product. The first version was launched in 2009 and became quite popular, in part due to its low price (about $90). The hardware and software are both open source. The current model is Version 3, which features the slightly modified hardware of the Version 2 model in a nice aluminum case.

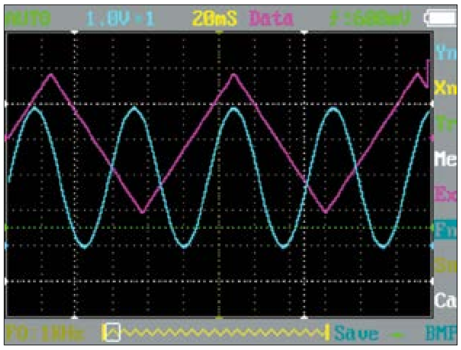


Figure 1. A screenshot showing the input signal and the signal in the buffer memory.

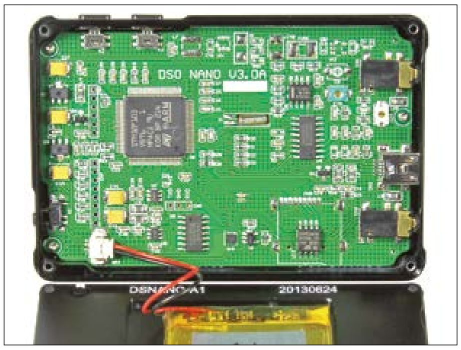


Figure 2. The hardware inside the DSO Nano. The ARM processor does virtually all the work.

Hardware and specs

The package you receive for less than 100 euros (tax included) is fairly complete, as you can see from the lead picture. Along with the DSO Nano V3, the box contains a storage pouch, two connecting cables (a probe cable fitted with mini-grabbers and a generator cable with two terminals at the far end), a small tool for opening the case and some self-adhesive feet.

> It takes a while to master the user interface.

As already mentioned, the oscilloscope is housed in a sturdy aluminum case. It is powered by a built-in rechargeable lithium battery that can keep the device running for several hours. A mini USB port allows the DSO Nano to be connected to a PC for transferring screenshots and measurement data, and it can be used to update the firmware. The mini scope can also be powered over this connection, including charging the battery. The connectors for the scope input and the generator output are 3.5-mm audio jacks, which mate with the connectors on the supplied cables. An on/off slide switch is located on the opposite side. The operator controls consist of five pushbuttons next to the screen and two at the top edge. The display is a 2.8” color TFT LCD with a resolution of 320 x 240 pixels. That may not sound like much, but it is sufficient for the size of the screen.

The circuit inside the Nano V3 is built around a fairly fast 32-bit ARM M3 microcontroller. The maximum sample rate is 1 Msample/s, and the analog bandwidth

is 200 kHz. The sample resolution is an impressive 12 bits, which is actually not necessary for the small screen dimensions but handy if you want to export the measurement data to a PC. The mini scope has an internal memory buffer with a capacity of 4,096 samples. There is also a built-in signal generator that can output signals up to 1 MHz with variable duty cycle.

Practical experience

My first impressions when unpacking and looking at the mini scope were every positive. The device looks neat and tidy — better than you would expect at the price. The included cables are not so great, but what can you expect from cables with audio jack connectors? After the device was switched on my general impression was still positive — the display is fairly bright and easy to read, and a demo waveform is shown right away.
After spending a while playing with the buttons and reading through the manual (available at [2]), my enthusiasm cooled down a lot and occasionally turned into frustration. The user interface is not particularly friendly. The scope itself works quite well and has a whole lot of features, but the software developers have tried to cram in way too many features, making the user interface very complicated and non-intuitive. You're constantly looking for buttons to press in order to make some sort of setting. It also doesn't help that the manual is very brief and some things are explained incorrectly or not at all. For example, there is a calibration screen that is shown in the manual without any explanation. However, some things have been implemented very intelligently. For example, installing new firmware is a piece of cake — I've rarely seen it so easy. Making screenshots, storing measurement data and copying data to a PC are also very fast and simple, and those are functions you would not immediately expect to see in a scope of this sort. In short, there are good points as well as bad points.
However, the user interface takes a while to get used to. After a day of messing about with this little marvel I had the feeling that I understood everything, including things not described in the manual (thanks to the Internet). Once you reach that point, the DSO Nano is a handy instrument — easy to take along and great for having a quick look at a signal. The accuracy is acceptable, and after you acquire some experience with the user interface you know where to find the main settings. Of course you can't compare it to a large oscilloscope, but it costs a lot less.

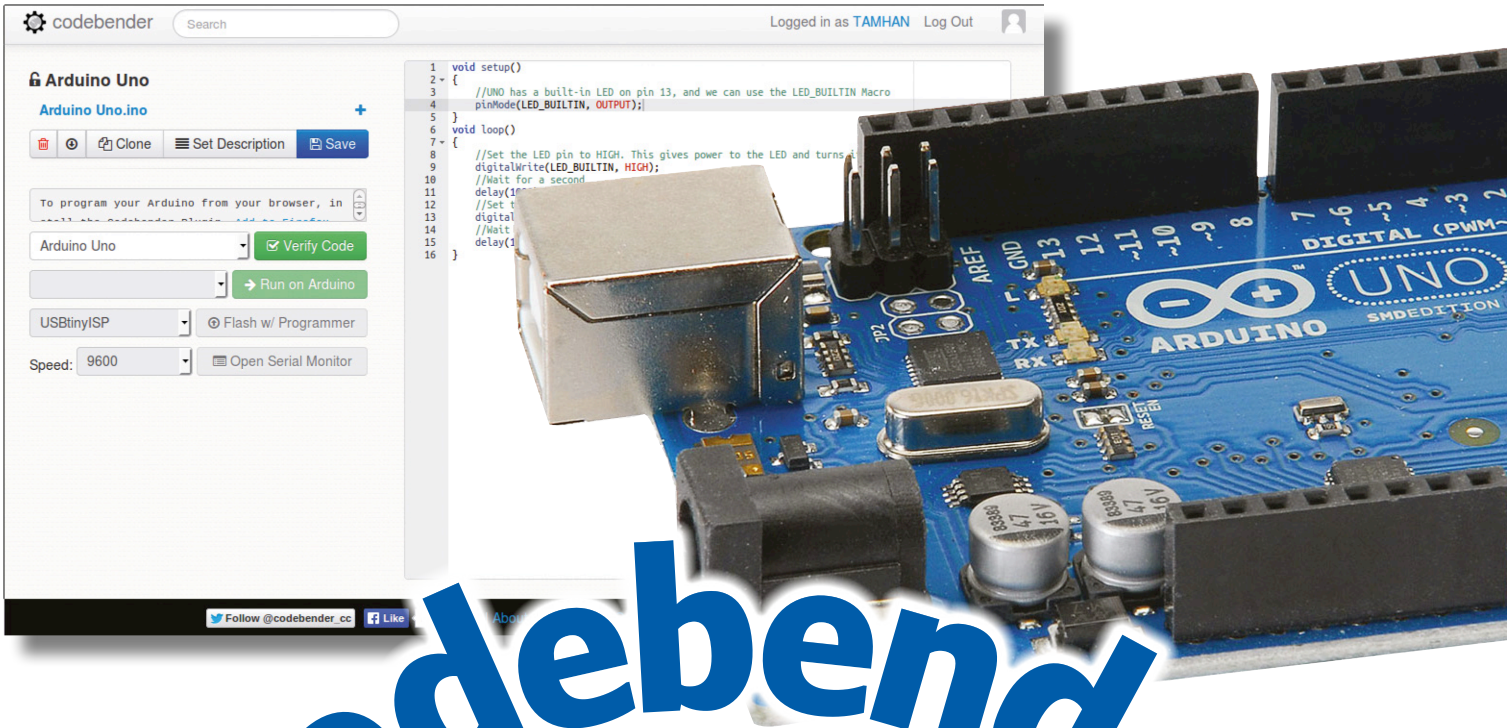
So is it good or is it bad? I am left with mixed feelings after this brief test. On the one hand I still think it's a really nice little instrument and very handy when you want to quickly view a signal somewhere outside your lab. On the other hand, the user interface sometimes makes me think: "grrrr – why is that function not there, or why didn't they chose a different way to implement that setting?" For example, the DSO Nano does not remember you settings, so you have to reconfigure everything after you switch it on each time. However, these are all software issues that could be resolved easily in future versions.
In the end, my conclusion is that the DSO Nano is more than just a gadget. The hardware is excellent for what you pay for it, and there's nothing wrong with the specs or the features. If the developers would put some more effort into sorting out certain aspects of the software, it would be a really handy little instrument that you would always take along for every electronics job away from your lab.
If you want to try it for yourself, the DSO Nano V3 is also available in the Elektor Shop now. However, be warned that you will have to put some time into learning how to use it. If that is not a problem for you, you can get a nice gadget and a reasonably usable little oscilloscope for under $/€100.

(150154)

Web Links

[1] www.seeedstudio.com

[2] www.seeedstudio.com/document/pdf/DSO Nano V3 User Manual 2.pdf



Codebender
Arduino online

By Tam Hanna (Germany)

Massimo Banzi's development environment is not everyone's cup of tea. In 2012 a group of Greek programmers built an Arduino IDE that runs in a browser window, more or less for fun. However, over the course of time it developed into a serious alternative to the standard Arduino IDE. Thanks to cloud storage, concerns about potential loss of data are now a thing of the past. Agreements with module makers allow most shield libraries to be included in a project with a click.

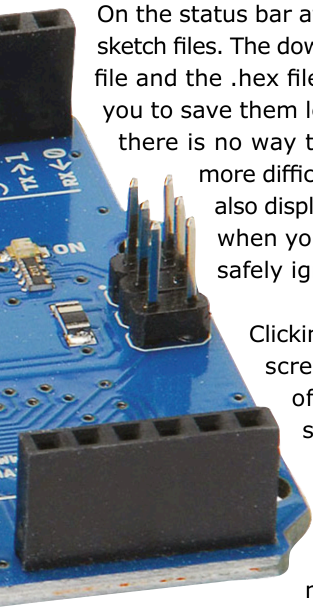
Microchip's PIC family of microcontrollers has the honor of having introduced masses of electronic enthusiasts around the world to MPU programming. The popularity of Massimo Banzi's Arduino platform is in part due to the fact that it totally eliminates the difficulties of working with ICs and programming devices for people without an electronics background. You simply connect the Arduino board to a USB port on your computer, and everything else is magic.
Downloading firmware is just as easy in the Codebender [cb] online development environment. Communication between the Arduino board and the computer is handled by a browser plugin that works with any version of Chrome or Firefox running under Windows or Linux (see **Figure 1**). However, only a few Arduino boards are supported (see **inset**).

If you want to use Codebender, you have to register. A setup wizard guides you through the procedure, including installation of the plugin. After this you can download your first sample program (Blink.ino) to the Arduino board.

EIt's all in the IDE

The Codebender user interface is shown in **Figure 2**. The window for the code editor, which is based on the ACE editor developed by Cloud9, occupies the right side of the screen. It is largely similar to editors such as Eclipse and kin, but the IntelliSense function offers significantly smarter code completion proposals than the Arduino IDE.

The compilation workflow is partially based on the Arduino IDE. To run a test compilation of the code in the editor window, you click the "Verify Code" button. If the compilation is successful, the program shows the size of the compiled code and you can copy it to the Arduino board by clicking "Run on Arduino".



On the status bar at the top you can switch between several sketch files. The down arrow lets you download the zipped .ino file and the .hex file written to the microcontroller, allowing you to save them locally on your own computer. At present there is no way to download the .elf file, which makes it more difficult to analyze the compiled code. Chrome also displays an error message about unsaved data when you click the download option, but you can safely ignore it.

Clicking the blue button on the right side of the screen activates the social network feature of the IDE. This allows you to share the source code over Twitter, Facebook and/or LinkedIn.

Smoking out the bugs

The familiar AVR GCC generates error messages that require a good understanding of C and/or C++ for proper interpretation. By contrast, the modular Clang compiler (see inset) used by Codebender is known for its friendly attitude to novices.
An example is shown in **Listing 1**, which contains three errors. After you click the “Verify Code” button, Codebender outputs a detailed error message that even marks the passage concerned with tildes. By comparison, the standard Arduino IDE is a lot less communicative.

Hobbyist magazines repeatedly claim that Codebender also uses Clang for the actual compilation. This is incorrect in so far as there is still no reliable backend that can translate code in the intermediate LLVM language into native code for the AVR family. The workflow in **Figure 3**, which describes the compilation process in some detail, is based on a blog posting [TSIP] published a good while ago.

Incidentally, the outputs of the two compilers are not identical. The compiled code generated by Version 1.5.7 of the Arduino IDE for the program shown above occupied 1030 bytes of code memory, while the code generated by Codebender occupied 1082 bytes. The difference is more significant with more complex programs. For example, a program that required 3052 bytes of code memory with the Arduino IDE needed 3256 bytes with Codebender. Complete “empty” projects are also a few bytes larger in Codebender, possibly because the boot loader is slightly larger.

Crowd coding

During the last few years the Arduino platform has become more or less standard in the maker scene and the semiprofessional sector. The broad popularity and large user community results in a large number of extensions that make it easier to access various components. These are usually called libraries, and Codebender offers several hundred of them.

If you want to use a ready-made library, start by searching through the list shown in Figure 4. In order to use the files, simply link them into the project with the displayed “include” statement.

What’s your processor?

Codebender supports Arduino Uno and Leonardo, but it doesn’t know anything about more complex systems such as Yun or the Due board, which is based on ARM technology.

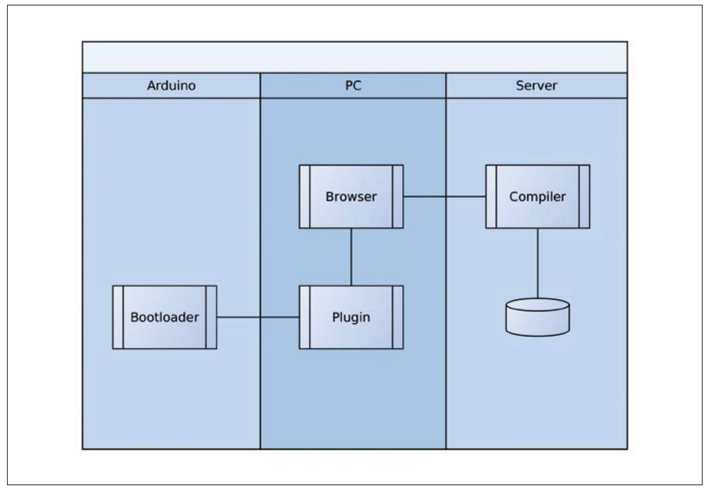


Figure 1. Codebender runs in a standard browser window. A plugin is necessary for transferring firmware to an Arduino board; versions are available for Firefox and Chrome.

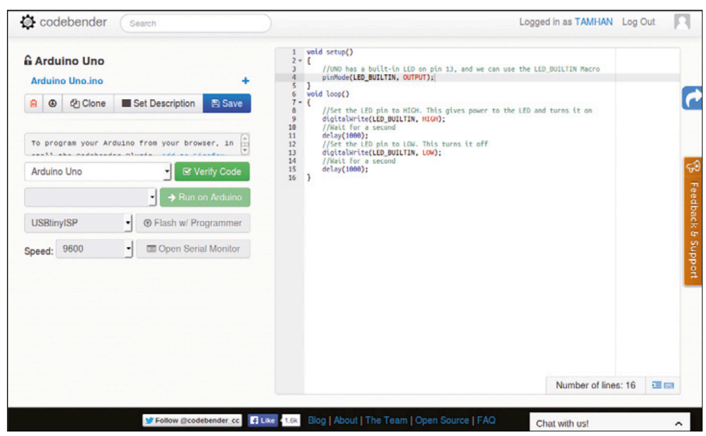


Figure 2. The code editor window is the core element of the online app.

Listing 1. A simple example with three errors.

```
void setup()
{
  pinMode(LED_BUILTIN, OUTPUT);
}
void loop()
{
  digitalWrite(LED_BUILTIN, HEIH);
  delaI(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000)
}
```

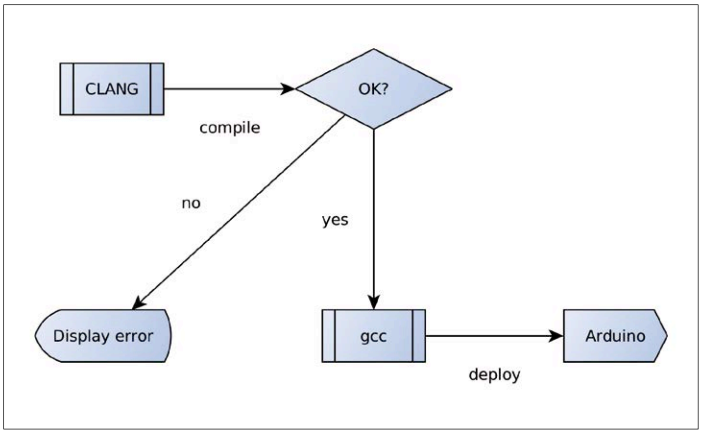


Figure 3. Errors in the source code are detected by Clang, a modular compiler frontend. However, the intermediate code generated by Clang is not used for anything else.

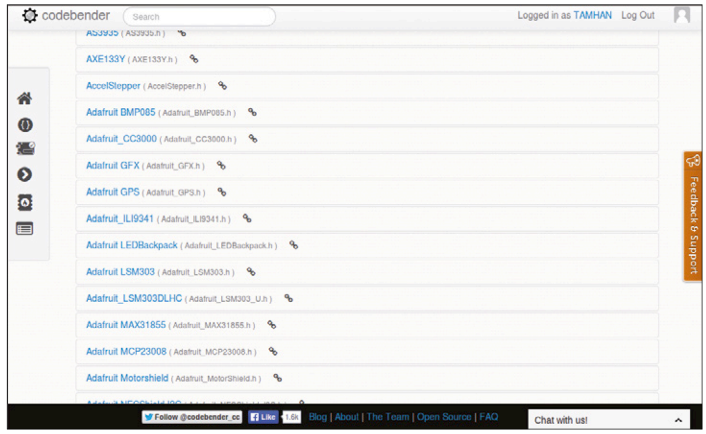


Figure 4. A list of libraries that you can easily use in your own Arduino sketches.

Clang and LLVM

Clang is a compiler project that has been around for a few decades, as an alternative to the well-known GCC compiler. It generates code for a virtual machine called LLVM, which can subsequently be converted into code for the target system either statically or dynamically.

The advantage of this architecture is that complier developers do not have to think about the target system. Their only job is to convert the input code into LLVM code that is optimized as much as possible (frontend function). Translation of LLVM code into real machine code (backend function) is then done by a specialized developer, who does not need to know anything about the high-level language.

In the case of Codebender, the problem is that at present no usable backend implementation is available for AVR microcontrollers. Codebender therefore uses Clang only for compilation to intermediate code, and if this process completes successfully the source code is regarded as error-free and can be handed over to GCC.

Error message from Codebender:

```
Arduino Uno.ino:7:28: error: use of undeclared identifier ‘HEIH’
        digitalWrite(LED_BUILTIN, HEIH);
                                                               ^
Arduino Uno.ino:8:2: error: use of undeclared identifier ‘delaI’; did you mean ‘delay’?
        delaI(1000);
        ^~~~~
        delay
hardware/arduino/cores/arduino/Arduino.h:107:6: note: ‘delay’ declared here
void delay(unsigned long);
     ^


Arduino Uno.ino:10:13: error: expected ‘;’ after expression
        delay(1000)
                   ^
                   ;
```

Error message from the standard Arduino IDE:

```
sketch_jan24a.ino: In function ‘void loop()’:
sketch_jan24a.ino:7:28: error: ‘HEIH’ was not declared in this scope
sketch_jan24a.ino:8:12: error: ‘delaI’ was not declared in this scope
sketch_jan24a.ino:11:1: error: expected ‘;’ before ‘}’ token
```

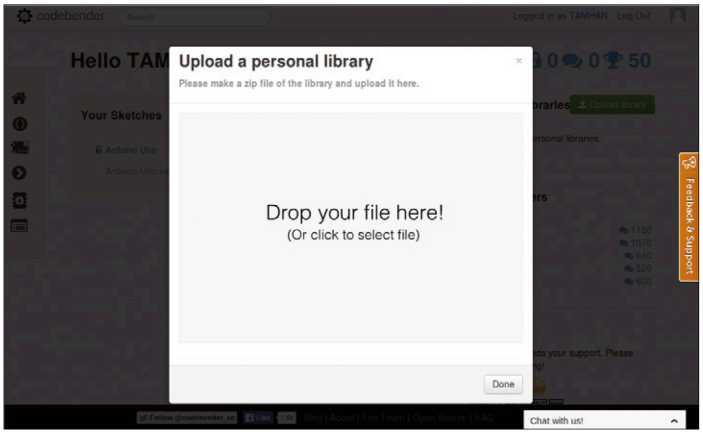


Figure 5. Codebender lets you upload your own libraries to the cloud.

Let the cloud do the work

Codebender lets you use your own libraries. Along with the conventional aspect of reuse, this enables you to modularize your sketches, which usually only consist of a single file.
To do this you have to log in to CodeBender.cc in order to get the page shown in Figure 5 on your monitor. You can use the "Personal Libraries" dialog displayed at the top right to upload .zip archives containing .c and .h files.
After you have uploaded your libraries, you can use them the same way as normal libraries. If there is a name collision, Codebender gives the version you uploaded preference over the version located in the framework.

Add more code

Last but not least, we should point out that the source code of Codebender is open. The developers use GitHub to distribute the code, so you can simply open [github] in your browser to download the various modules to your computer.
Change requests can be sent to the developers by using the issue tracker accessible at [IT]. There users can propose ideas or suggest new features and vote on their implementation. You can also use the chat window at the bottom of the website for direct contact with one of the developers on duty.
At present there are not yet any fixed agreements that allow the administration of private projects. If you would like to have this capability, you can make your wishes known by sending a message to support@codebender.cc. It's certainly conceivable that beta testers could already try out some of the functions that are intended to be limited to paying customers.

Summary

Codebender presupposes that your code is stored in the cloud, and as of now there is no way to mark projects as private. In addition, the compiled coded is 5% to 15% larger on average. Developers working on open-source products would do well to think about switching to Codebender because the improved IntelliSense function, the smarter compiler and automatic code hosting are valuable features.
In their own interest, providers of libraries and shields should consider a partnership arrangement, since the Codebender developers are now the proud owners of an enormous volume of application code thanks to the library system. A partnership arrangement also allows potential problems arising from API changes to be estimated in advance, since Codebender can determine for paying customers how much of the code (in percent) is no longer compilable due to the change.

(140519)

Web Links

[cb] www.codebender.cc
[TSIP] http://blog.tzikis.com/?p=454
[Lib] https://codebender.cc/libraries
[IT] http://feedback.codebender.cc/forums/165703-general
[github] https://github.com/codebendercc

Q&A

(Almost) Everything You Always Wanted to Know About...

Input Protection

Contributors: **Burkhard Kainka, Malte Fischer, Luc Lemmens, Clemens Valens**
Compiled by **Jaime González-Arintero**

It's safe to say that protecting the inputs of your MCU is a matter of preferences, with the application, the budget, and the available space all contending factors. Compare the one-off private project where cost is not a real issue to the mass-produced design where every half cent is CEO'd over. Or the case of wanting to implement a robust protection but PCB space extremely limited...

Q *Is it recommended to include additional resistors in line as overvoltage protection?*

A Inline resistors provide current limiting (due to high or low voltages/impedances), not over-voltage protection, although the first may be useful for the second. Consider however an output shorted to GND: The voltage is certainly within permissible limits, but the current may go excessive.
It is possible (but not recommended) to use a series resistor together with the pin's input impedance to create a voltage divider. When using an inline resistor for current limiting it must be chosen such that the typical working current flowing through it under normal circumstances does not create noticeable voltage drops (i.e. keep it small!).
Although we've mentioned the pin's input impedance, normally this has more to do with the intrinsic diodes of the input circuit (aka. "PIN diodes": P-type + undoped Intrinsic region + N-type). For our calculations, we should consider the maximum ratings of the input, or their sum in case of using several inputs, hoping that this will solve the problem. However, working with ICs in safety-related applications we shouldn't be using this approach, and opt for external terminals. And there's also something important to bear in mind: the current flowing through the input and the diode to V_{CC} in the IC "has to go somewhere," so to say. If it's smaller than the consumption of the IC and the remaining surrounding components, then it will be taken directly from V_{CC}, without further changes. But beware: If it's larger, it will raise the value of V_{CC} as well! As another solution, consider using depletion FETs in series with the input.

Q *Then, what should be used as overvoltage protection?*

A One of the most popular and easiest approaches to prevent voltage surges is to use a diode clamp, typically consisting of two reverse-biased diodes. This is a well-known circuit requiring no extensive explanation, but in short, the diodes will begin to conduct as soon as the voltage at the input exceeds the forward voltage of any of them. The most common application for a pushbutton is depicted in **Figure 1**. Several microcontrollers already integrate diode clamps in their inputs, thus it is recommended to check the data sheet first. Note the inline resistor, and see the previous question for references.

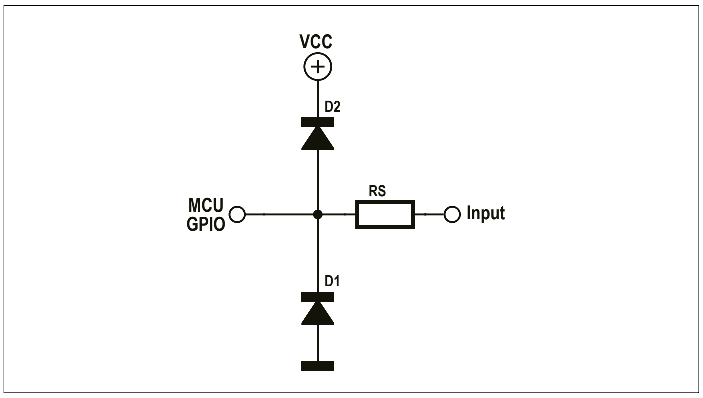


Figure 1. Push it ~~to the limit~~. A simple, cost effective diode clamp will prevent you from doing that!

Q *What about varistors?*

A Voltage-dependent resistors (VDR) provide over-voltage protection as well as transient voltage suppression. The most popular type is the metal-oxide varistor (MOV). Their capacity to withstand a relatively high current, with very high current surges, makes them especially useful in equipment connected to AC power lines. However, they can also be used in MCU digital inputs. Typically, varistors may be connected to ground "in parallel" with the circuit to be protected. For instance, in an application to protect a digital input, once the current rises undesirably as a result of an overvoltage, the varistor will shunt it to ground. Nevertheless, it's important to note that these components degrade with use, meaning that there's only a certain number of transient pulses they can withstand.

Q *Wait, if MOVs also provide transient suppression, why using TVS diodes then?*

A Transient voltage suppression (TVS) diodes are specifically designed to provide protection against surges and spikes that appear during a transient phase. These components react much faster than standard Zener diodes (i.e. picoseconds!), they are available in unidirectional and bi-directional types, and provide good ESD protection as well. The typical configuration is the same as with MOVs (connected to ground, "in parallel" with the input). However, it is to be considered that TVS diodes, although usually faster than anything else, might not withstand as much dissipated power as varistors, for instance. A brief application note from Semtech [1] explains their operation in depth.

Q *What are the pros and cons of using optoisolators?*

A If your design can afford it (in terms of the *budget*), optoisolators (aka. optocouplers) are certainly a great option. Most times, inside an opto-isolator there's an LED — the emitter — and a phototransistor — the receiver — which "communicate" optically instead of electrically. This feature also enables us to interface with signals rated in other voltage levels (for instance, a 12-V logic level signal with a 5-V tolerant input). At first blush this appears unbeatable. Still, there are many things to consider when using them, as they will inevitably add complexity to the original design. First of all, to be on the safe side, the optoisolator must not share the power supply with the circuit we want to protect (in this case, an MCU input). It wouldn't make sense if we intend to electrically isolate two circuits, and they have electrical contact via the power supply. Also, optoisolators are slow, at least compared to other input protection devices. Time critical applications may require other techniques. And last but not least, standard models are not suitable for analog signals, and we may have to fork some money out if we decide to add an analog optoisolator to our circuit.

Figure 2 shows a typical setup for a simplified optoisolator. Note the different, independent power supply (V+).

(150049)

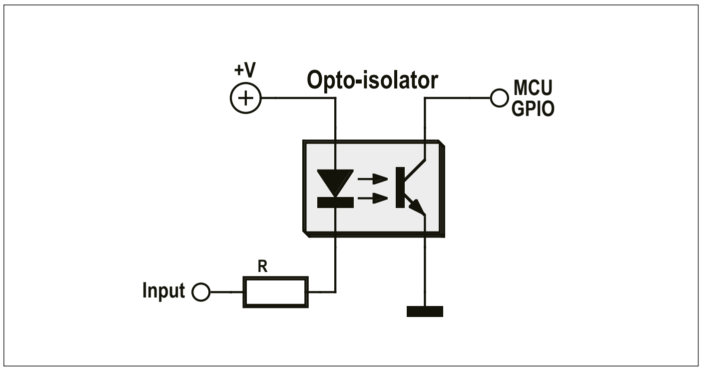


Figure 2. A typical, simplified optocoupler setup. Some ICs include a whole set of photocouplers to protect several inputs with one chip. For example, Lite-On's LTV-817 integrates four of them.

Web Link

[1] http://po.st/tvs

Still having unanswered Q's? Worry not!

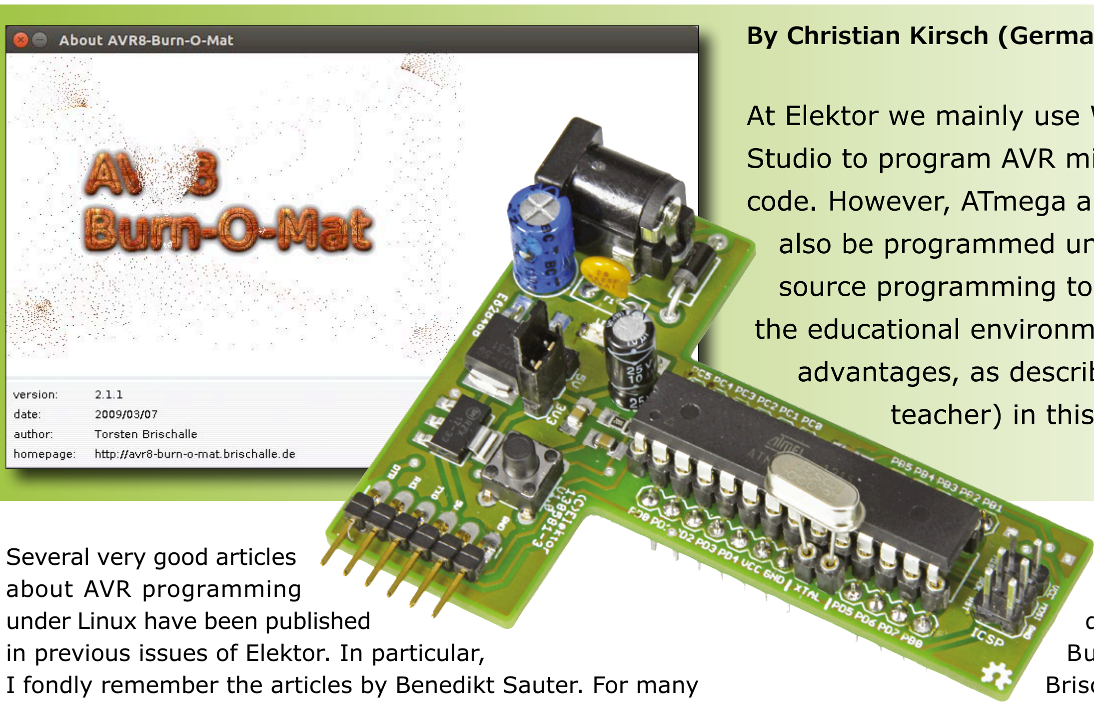
Not all the questions about **input protection** have been addressed! Two pages on such a far reaching subject is certainly not enough, so we will revert to the subject in upcoming editions, providing more A's. Stay tuned!

Stay tuned!

Q & A's next edition will cover...

Programming AVR Micros under Linux

Using Burn-O-Mat and Code::Blocks



By Christian Kirsch (Germany)

At Elektor we mainly use Windows and Atmel Studio to program AVR microcontrollers with C code. However, ATmega and ATtiny devices can also be programmed under Linux using open-source programming tools. Particularly in the educational environment, that has several advantages, as described by our author (a teacher) in this article.

Several very good articles about AVR programming under Linux have been published in previous issues of Elektor. In particular, I fondly remember the articles by Benedikt Sauter. For many years I have been using Linux (Ubuntu) exclusively in my teaching activities, with the learning focus on AVR (ATmega16 and other types), C and C++ in the console, and C++ with the wxWidgets GUI library.

Fewer driver problems with Linux.

First I would like to recommend separating programming (generating hex files) from downloading code to the microcontroller with the aid of a programmer and an ISP port. In our experience the well-known IDEs are not flexible enough to support the various programmers we have used over the course of time. At first we used the classic serial interface for programming, and after it disappeared we started using a number of alternatives, including the USBprog device from Elektor. Now we also use a low-cost programmer from Diamex and the programmer from Nand Eeckhout described in the July/August 2008 issue of Elektor. We build this programmer ourselves, so it is cheaper than any other option. For me, the sheer diversity of programmers was enough to justify using a separate program to download the compiled files. The ability to send code to the microcontrollers outside the IDE environment was another reason, since in many you only need to download an existing hex file.

For downloading we use the free command-line tool AVRdude [1] [2] together with AVR8 Burn-O-Mat [3] from Torsten Brischalle, which wraps a graphical user interface around AVRdude. This is very good combination, and it is also good for fuse management.

Development environment

My choice for the development environment is Code::Blocks [4]. This actively managed, high-performance open source project has a lot of attractive features. It is designed as a cross-platform IDE, so you can also use this IDE to develop many other projects, including projects for ARM microcontrollers, QT GUIs, Open GL and more. **Figure 1** shows a pair of screenshots that illustrate this diversity.

The advantage of using a single IDE for all platforms is especially important in the teaching environment. It saves a lot of time and frustration when teachers and students do not have to come to grips with many different tools.

Of course, the developers have not ignored the Windows community or even Mac users – the IDE is implemented with wxWidgets and runs under all common operating systems.

However, if you want to use Code::Blocks you will have to install some of the necessary components yourself. Among other things, it does not come with an AVR compiler or a C compiler for the PC. However, here as well the environment is very flexible and allows you to use a wide variety of compilers. For example, if you already have a Borland compiler installed then Code::Blocks will find it and you can use it.

For Linux the GNU GCC compiler is mandatory, and the version for AVR microcontrollers is called GCC AVR. For installation you

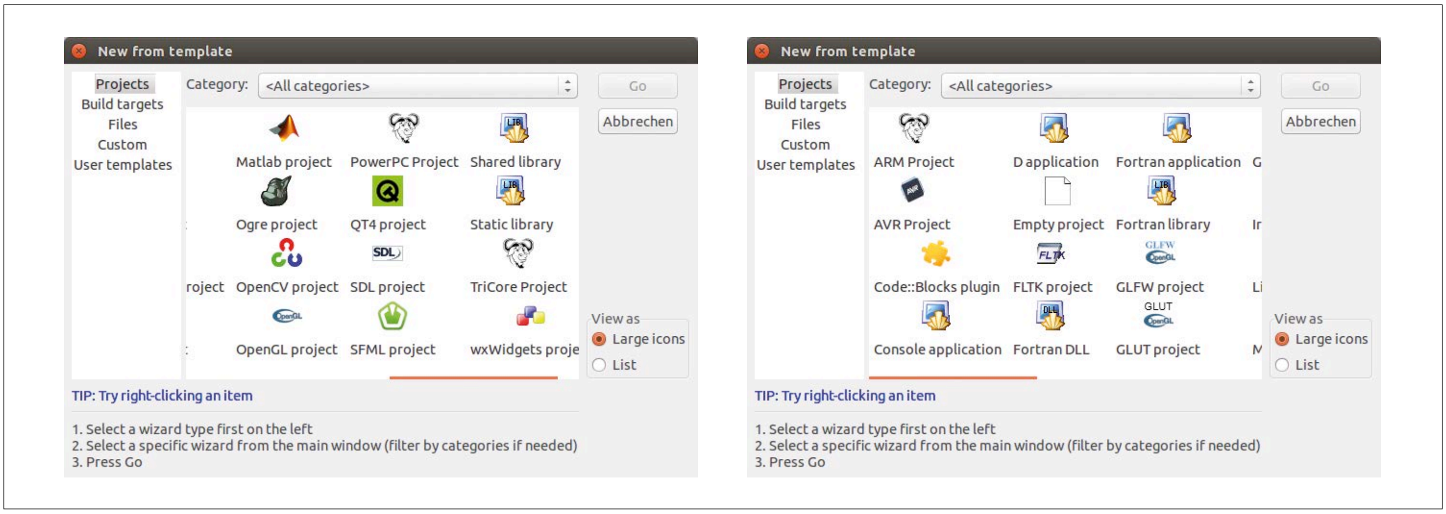


Figure 1. With Code::Blocks you can program a lot more than just AVR projects.

can use the Ubuntu Software Center, the Synaptic graphical packet management tool or the console. Here I have chosen the console route as an example. All of the procedures described below have been tested under Ubuntu 12.04LTS and 14.04LTS (at school I only use long term support (LTS) versions). In the console, enter the following command for GCC:

```
sudo apt-get install gcc g++ gdb
```

The following commands are necessary for AVRdude:

```
sudo apt-get install avrdude
sudo apt-get install avrp gcc-avr avr-libc gdb-avr avra
```

You also have to enter the following commands:

```
sudo apt-get install make udev
sudo usermod -aG tty student
sudo usermod -aG dialout student
```

User permissions

As with all open-release software, you have to study the commands listed above for a while before you understand them and know what they do. For example, consider the user permissions: the command line sudo usermod -aG tty student (where "student" is replaced by your own user name) is necessary to enable you to use the programmer over the USB port.

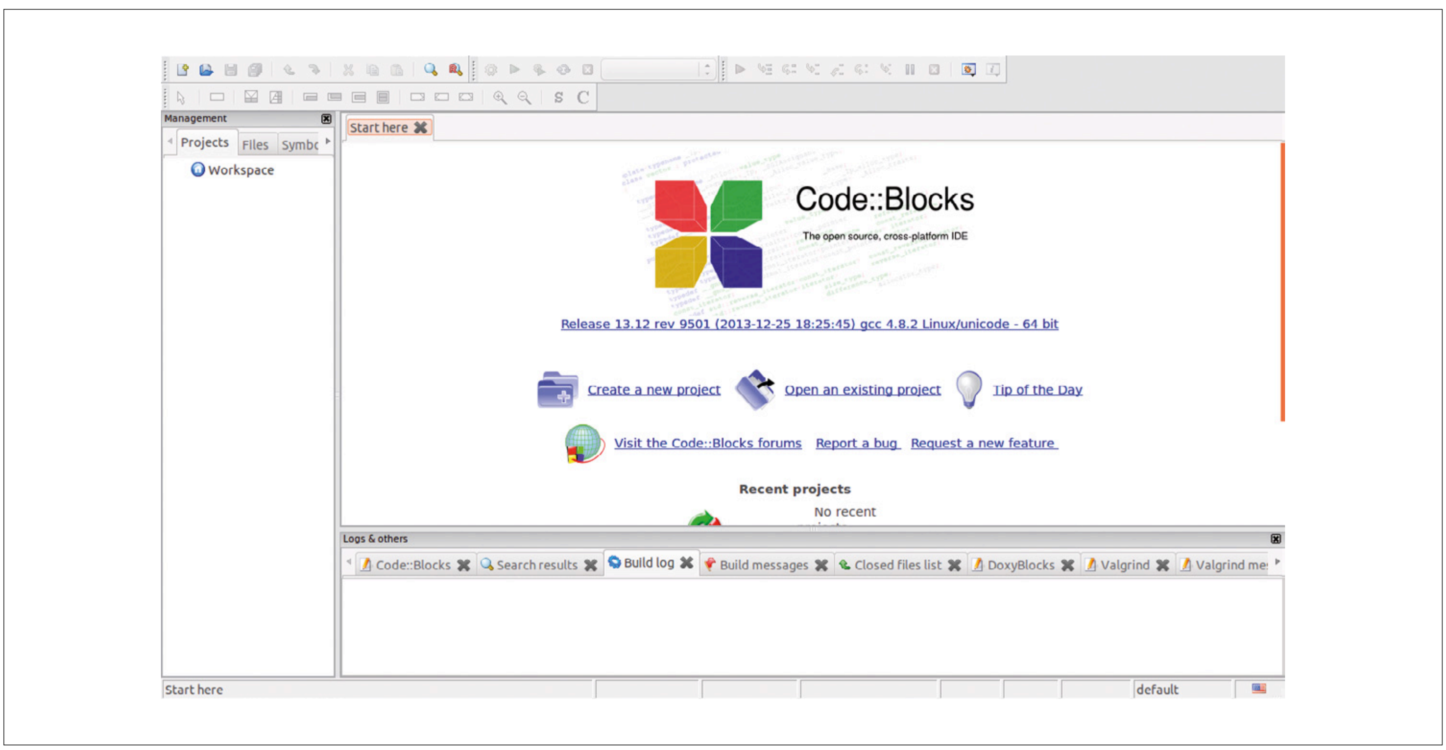


Figure 2. The start screen of the Code::Blocks development environment.

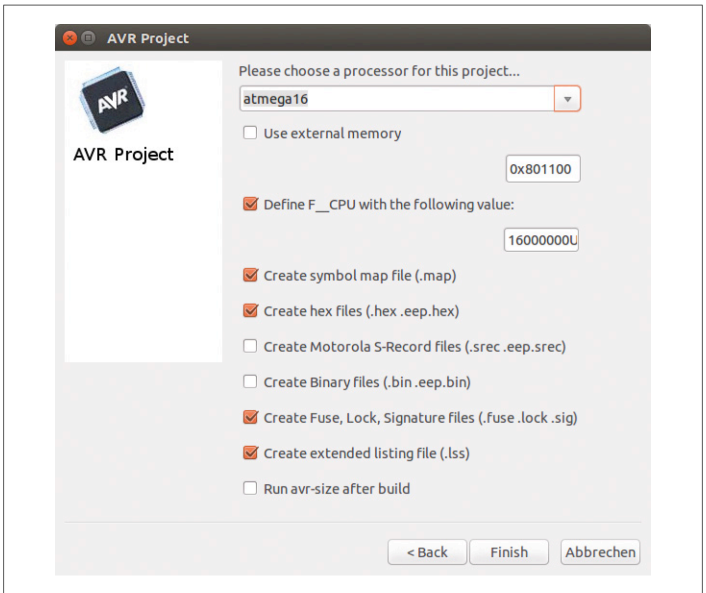


Figure 3. Selecting the microcontroller type and clock frequency.

Otherwise you will find yourself (like me at first) staring at your PC screen and wondering why the program keeps telling you “Permission denied”. Even under Linux (Ubuntu, Mint or whatever), things do not always run properly right off the bat. However, that’s nothing compared to the dismay of my students when they try to install the drivers for the Diamex programmer under Windows. Based on our experience so far, that doesn’t work properly most of the time (up to 90%) and often means that the students cannot continue working on their projects at home.

For Burn-O-Mat you need the Debian package from Torsten Brischalle. Unfortunately it cannot be found in the sources for Ubuntu, for some inexplicable reason. After downloading Burn-O-Mat, install the Debian package with:

```
sudo dpkg -i avr8-burn-o-mat-2.1.2-all.deb
```

The only thing you still need is Code::Blocks.

```
sudo apt-get install codeblocks-contrib codeblocks
```

Under Ubuntu 14.04 you should install version 13 of Code::Blocks. For this version you also need the following library:

```
sudo apt-get install libc6-dev-amd64:i386
```

If you want to program a GUI for the PC with wxWidgets, you should also install:

```
sudo apt-get install libwxgtk2.8-dev wx-common
```

Now you have everything you need to get started.

Settings

First launch Code::Blocks (**Figure 2**). After creating a project, select “AVR microcontroller” as the target processor. In the next window, give the project a meaningful name and put the project on the desktop. In the next window the AVR compiler should already be selected. The Debug and Release folders should both be created automatically.

In the next window (**Figure 3**) there are several things specific to AVR. For example, this is where you select a particular device type and set its clock frequency. You can leave the

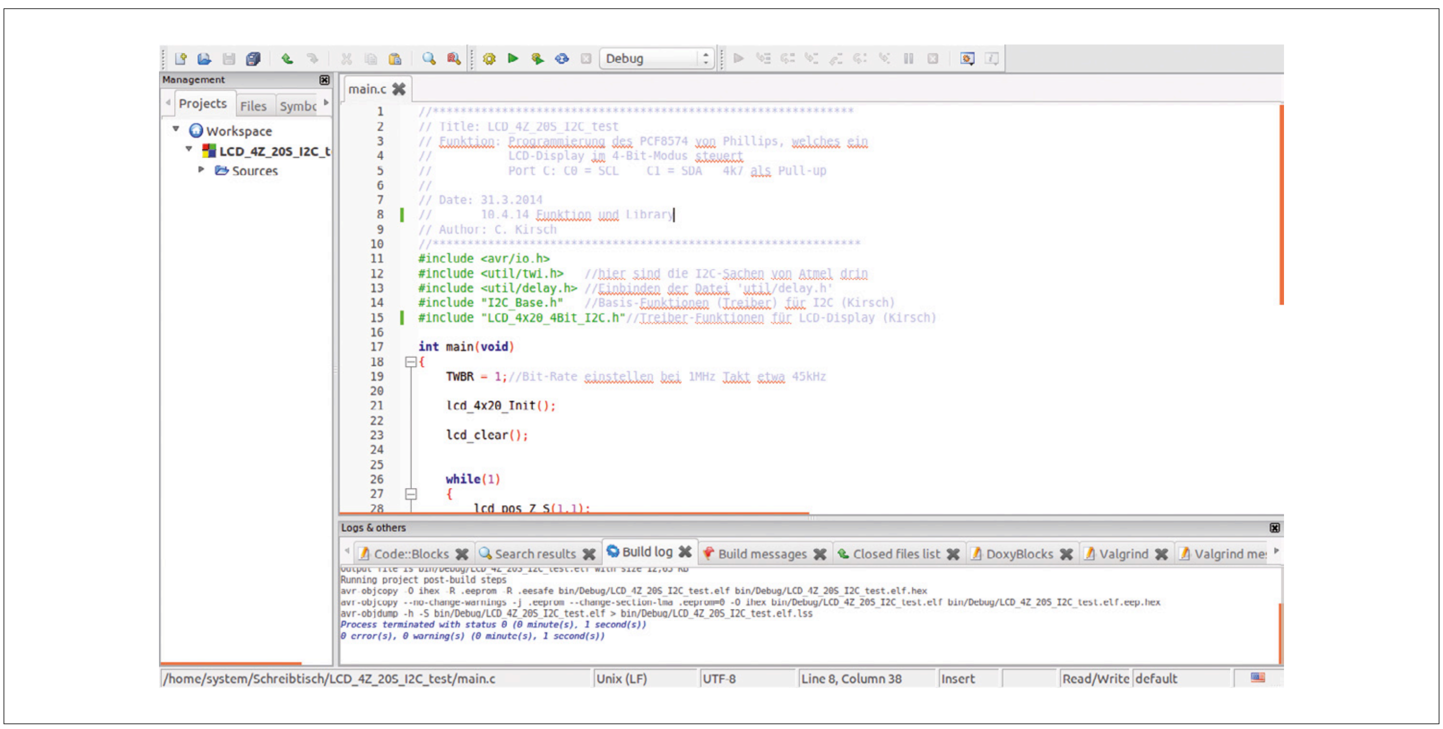


Figure 4. The code window with the author’s code for driving a display module.

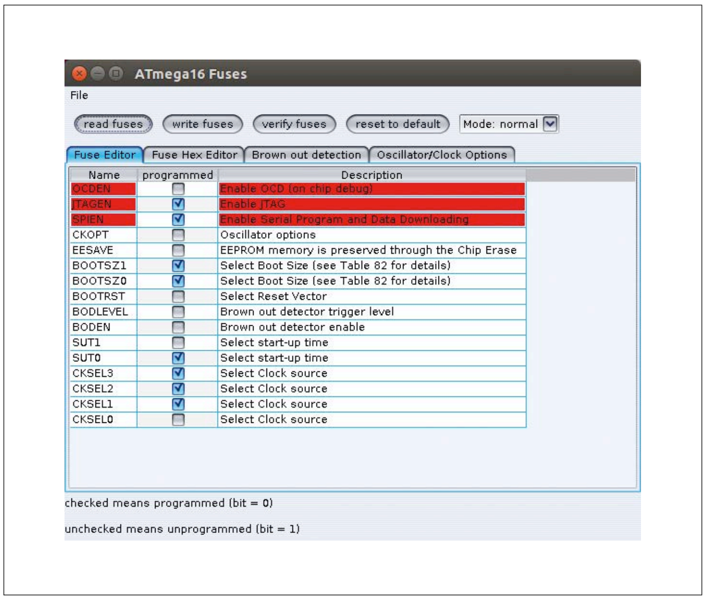


Figure 5. Fuse list in Burn-O-Mat.

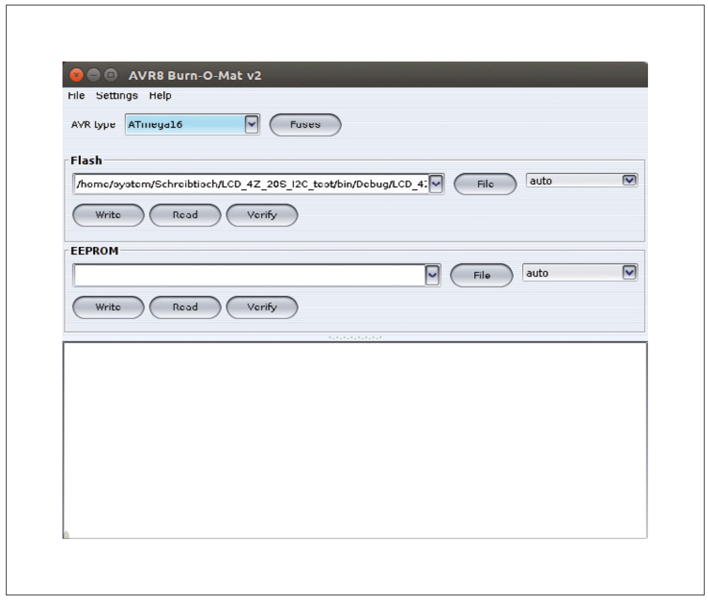


Figure 6. The “Write” button downloads the code to the microcontroller.

default ticks in the checkboxes unchanged. They are intended for more advanced users.
After you click the “Finish” button, a window with a bit of automatically generated source code opens. There you can edit the existing code as necessary and add your own code. When you have finished writing the source code (**Figure 4**), click the gear symbol to compile the code and create the hex file. After your program compiles without any errors, the hex file will be located in the Debug folder or the Release folder.

Now launch Burn-O-Mat if you have not already done so (that takes a while, probably due to the very elaborate “Help about” part). It’s fun to mess about with the opening screen while you’re waiting (see our lead photo).

The first time you launch Burn-O-Mat, you have to enter the correct path for AVRdude under “Settings”. Of course, this means you have to know where that program was stored. Under Ubuntu, the AVRdude program is located in /usr/bin/avrdude and the AVRdude configuration file is located in /etc/avrdude.conf.
After entering the path, close the program and then open it again. Now you will see Burn-O-Mat in the programmers list (which you can update if necessary), which is where you select your programmer and the associated interface. The Diamex programmer requires a virtual serial interface. For this you must manually enter: /dev/ttyACM0. No additional drivers are necessary.

You can check the connection in the “Fuses” window. When you press “read fuses” (**Figure 5**), the microcontroller returns the fuse settings. Incidentally, the error messages from AVRdude are displayed in the bottom part of the main window.
Now you can find the hex file under “File” and transfer it to the microcontroller by selecting “Write” (**Figure 6**).
Alternatively, you can use AVRdude directly from Code::Blocks. To do so you must enter the right settings under “Tools”. The IDE has may professional features, including version control, a debugger and a Doxygen plug-in (DoxyBlocks) for documentation. A Nassi-Schneidermann plug-in is also available starting with version 13 (important for me as a teacher).

Summary

We have been using all versions of this program since 2007 without any problems. By contrast, with each new version of Windows there are more and more problems, particularly in schools, in part due to various security issues (for example, at our school we use the HDGuard security software, which always resets all the settings). Another issue is that more and more free software is blocked by Windows. That’s why my students often have problems with installing drivers or free programs under windows.

Incidentally, you can also program Arduino devices with the software mentioned above. If you don’t enjoy the Arduino environment and would rather work with standard C+, you can program Arduino microcontrollers over the ISP port, which is usually present on the board. Low-power applications such as battery-operated data loggers running at just 1 MHz are also possible this way. You can even use the ISP port to reload a boot loader that has stopped responding.
So give it a try – for example, by programming some T-Boards under Linux.

(140427)

Web Links

[1] http://savannah.nongnu.org/projects/avrdude

[2] https://learn.adafruit.com/usbtinyisp/avrdude

[3] http://avr8-burn-o-mat.brischalle.de/

[4] www.codeblocks.org

Unwrapping the IoT

It's WunderBar

By **Jaime González-Arintero** (Elektor)

The Internet of Things (IoT) is one of the apples of discord within the electronics community. Mass tech media has been doing its part, as always, *hyping* the term and (in my opinion) generating even more confusion among the general public. But will the so-called ubiquitous computing really transform the world, or is it just, as some say, about finding solutions to *first world problems*?
To stop reading overhyped articles, and get our hands on what this really means, there are several IoT devkits available. The WunderBar, however, is a peculiar case: it doesn't come directly from a CA big 'gun (*hush*, Intel, *hush*), nor from Ten Pole Tudor, but a young Berlin-based startup called **relayr** [1]. Besides, it doesn't focus on the hardware, but on the software. So what's an electronics enthusiast to do here?

Figure 1. Unwrapping the box. The other "chocolate" case is a 3-D printed prototype we got some time ago, courtesy of relayr. It actually looked tastier! ;)

Chocolate-of-the-art

Okay, it looks fancy… but what's the WunderBar exactly? It's crystal clear, my dear: "An Internet of Things starter kit for app developers." You still don't get it? I confess, the first time I heard this I had no idea what was going on here. Last year, during the **electronica** show in Munich, many visitors dropping by at the Elektor booth enquired about the weird pieces of chocolate we had all over the place, so I made a vow to explain more in depth *la mystique* that surrounds this kit. And actually, it's quite straightforward!
Before delving into technical details, let's first grasp the basics of the WunderBars. This kit consists of a master module and a set of six sensor modules (more details later on). Just think of a kind of master/slave model. The sensor modules collect values and send them to the master module via Bluetooth Low Energy (BLE), which in turn, is connected to the Internet via WiFi, uploading all sensor data to a cloud platform. So in broad lines this *e-chocolate bar* is just the link between a network of sensors and the relayr cloud. All data collected by the sensor modules is sent and stored there, and can be used and retrieved later by other devices with the right permission, unless you make a certain sensor 'public' (as we'll see later). My assumption is, obviously, that every time you check the value of a certain sensor, you are not really 'polling' the sensor itself, but reading the values stored in the cloud — which were previously delivered by the sensor, and uploaded by the master module.

A secret recipe?

Not secret at all, but as we'll discover, entirely open source. We could call the WunderBar a *softwarecentric* devkit, underpinning that aside from the *chocolate-y* product concept itself, the uniqueness of it is in the code samples, the available SDKs and libraries, not forgetting the impressive amount of documentation. As said, all hardware and software is open source, so it's there for everybody to see and re-implement. Silicon and substrates is what we love, so don't miss to check this hub [2].

The WunderBar comes in a truly attractive packaging, and unboxing it makes you feel pretty much like a kid unwrapping a toy in Xmas. Inside the box the first thing you'll find is a 'chocolate bar' plastic casing, a USB cable, a 3.7-V / 130-mAh Lithium battery for the master module, 5 pcs 3-V CR2032 coin cells for the sensor modules, some foam tape pieces to stick on the module, and of course the circuit board (**Figure 1**). Everything is in one single circuit board, and you may break off the sensor modules you intend to use. Before this, powering the master module via USB (with the lithium battery attached!) will also power all modules, even if you haven't attached the coin cells to every module yet. In fact, although breakable,

the sensors modules are connected to the same supply line by means of some sort of castellations (see "Escaped from the Labs," Elektor 2/2015). Later, these notches also prevent the modules from coming out of the plastic casings.
The master module is based on a Freescale 32-bit ARM Cortex-M4 MK24 from the Kinetis series, with 1-MB program flash memory and multiple peripherals and interfaces (incl. SPI, I²C, UART), which comes in an extremely thin XFBGA package that we would never be capable of *reballing*, not even in our wildest dreams. A GainSpan GS1500 WiFi module takes care of the Internet connection, while a Nordic Semiconductor nRF51822 48-pin QFN SoC with embedded transceiver for BLE connectivity establishes the communication with the sensor modules, relieving the Freescale MCU from this task.

The cornerstone of the sensor modules is the same Nordic SoC, but in this case it is the star actor. As said, this kit includes 6 'sensor' modules: light/color & proximity, accelerometer/gyroscope, temperature/humidity, sound/noise level, IR transmitter (beware appliances!), and a 4-pin Grove connector (to interface with other dev tools). Indeed, these last two are not sensors, hence the quotation marks, but they'll surely bring the most

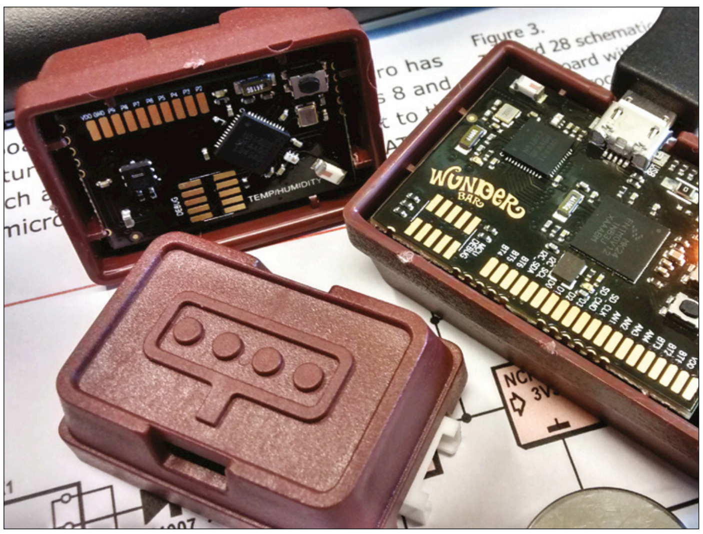


Figure 2. Everyone, and I mean EVERYONE, will ask you WTF are those things.

interesting applications. All modules integrate an 8-pin debug port, and a series of exposed pins for interfacing purposes.

Let's "onboard" the bar

The first-time configuration of the WunderBar takes place in a process dubbed "onboarding." A step-by-step guide is available at relayr's website [3] — it's as easy as pie. For that you would need to install an app in our smartphone named "relayr Manager App" [4] (available for Android and iOS), and of course power up the master module. Once provided with the credentials of our WLAN network, the app will synchronize with the master module, and after this it will scan for the rest of the sensor modules. Before starting this routine, you will be asked to activate the Bluetooth connection in the smartphone. Considering

> Everyone, and I mean EVERYONE, will ask you WTF are those things.

that in normal operation, by default, the WunderBar **does not** communicate with our device in this manner, my guess is that during this process Bluetooth is only used as a bridge while the connection with the cloud is being established. Later on, if needed, you could even access the modules individually, via direct BLE (this feature may be enabled with the app). If not, you can deactivate the Bluetooth connection again, and the kit will keep working perfectly.

Once the "onboarding" is done, you can snap off the sensor modules from the PCB, fit the batteries accordingly, and insert the modules in the chocolate cases for a fancy look (**Figure 2**). You can already play around with the sensors, and place them wherever we want. For instance, **Figure 3** shows one of these *e-chocolate* pieces in a window at the Elektor Castle, in this case the temperature/humidity sensor. A screen-

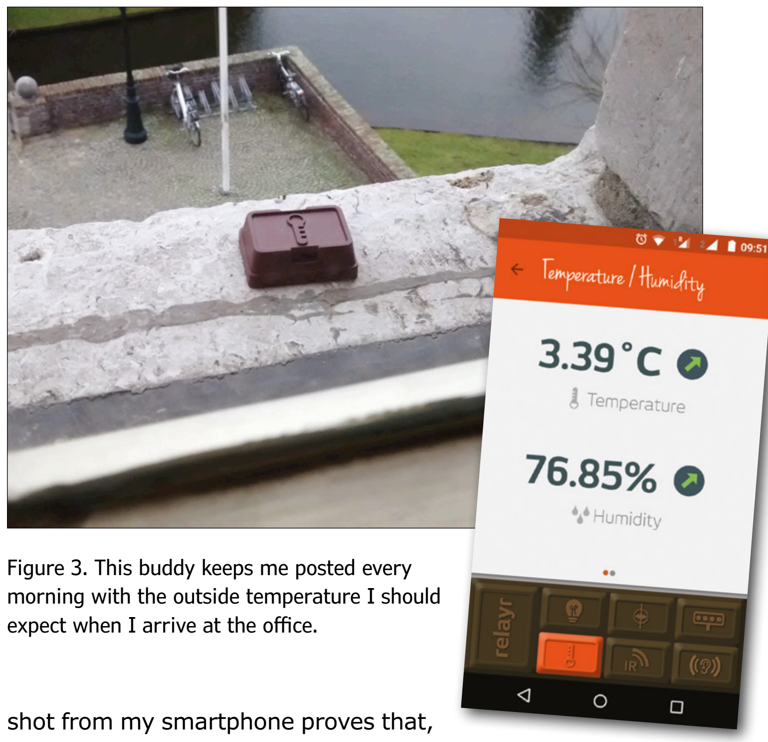


Figure 3. This buddy keeps me posted every morning with the outside temperature I should expect when I arrive at the office.

shot from my smartphone proves that, yeah, we're actually in the Netherlands! (as a Spaniard, don't you think I have the *right* to complain?)

Charlie & the Apps Factory

As we mentioned before, what really differentiates the WunderBar is the software. First of all, currently there are five SDKs available, all of them open source: Android, iOS/OSX, browser (JavaScript based), Python, and the recently added C# (.NET). To get started with coding, you would need an API key, which can be obtained right away in the same dashboard where everything else is available [5]. Here you can also check the status of our WunderBar(s), and retrieve the values of every sensor. For remote testing purposes, I set one at the

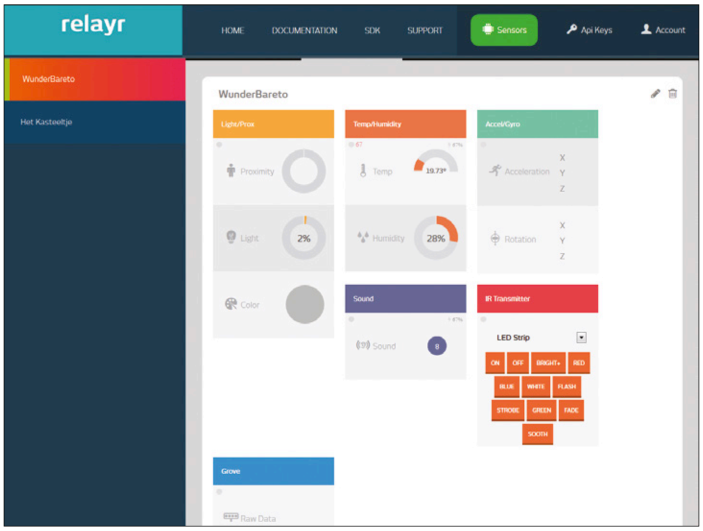
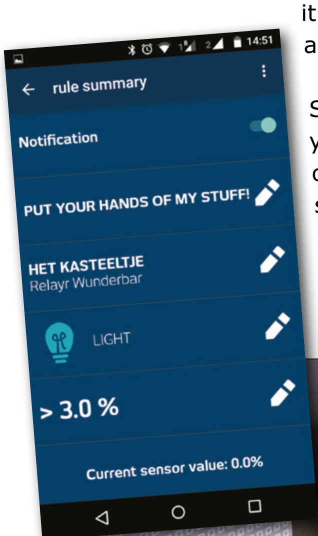


Figure 4. In the dashboard, under Sensors -> My Devices you can view the status of all the WunderBars linked to our account.

office, and another at home, as shown in the left side of the dashboard (**Figure 4**). In this board you can make the sensor values publicly available, and embed them by means of a script. Otherwise they will remain private, and visible only to the owner via the dashboard or an app. Besides, you can also define the refresh rate of every single sensor. For example, it doesn't make much sense to check an ambient temperature once per second…

So you're still there? Well, that could mean you're *up to scratch* with your app development skills. Worry not! Aside from the software devkits, there's also a couple of apps you may want to try (apart from the manager app itself, which already provides all the data from the sensors).



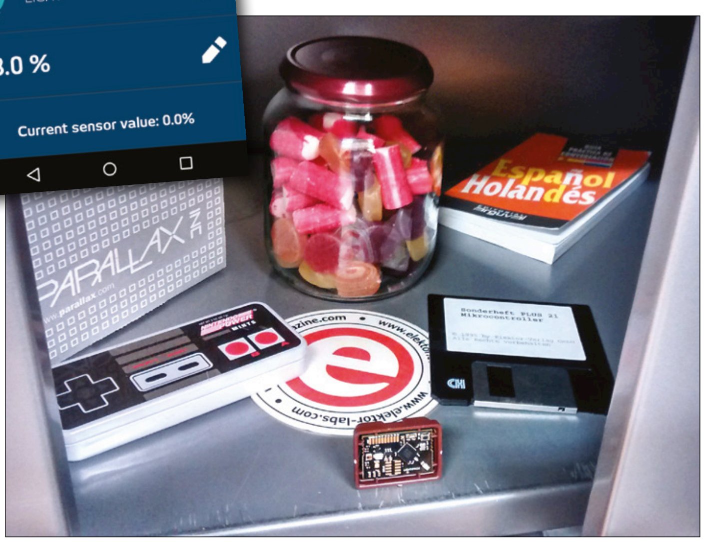


Figure 5. My candies, my Spanish/Dutch survival guide, and my own portable cloud storage system (a.k.a. floppy disk) are safe now. Sure, I could close the cabinet and take the key with me, but that would ruin the fun of searching for culprits.

The app "TellMeWhen" enables you to set rules that will be triggered according to user configurable values collected by the sensors. Picture it: I'm tired of colleagues opening my cabinet at the office in search of some of my most precious goodies, before I arrive there (that's easy to do, since I hate mornings). Thus, I decided to place the light/proximity sensor inside (**Figure 5**), which will let me know instantly every time someone opens the cabinet, no matter where I am. Caught!

Yummy?

Only time will tell. I must admit, I was quite skeptic at first with all the IoT fever, when everybody was basically hyping the term and mentioning futuristic *useless applications*. Now I'm sure that the Internet of Things will happen, but as with every other new technology, it will be subject to a slow, gradual adaptation process. The industrial and real life applications are simply countless: just imagine having instant real-time feedback of thousands of wireless sensors, and the possibility of retrieving the (accumulative) historical data anytime, from anyplace in the world. Of course that's a bigger deal than the trivial example applications we discussed here… By the way… Wait… Whaaat? Darn! Someone opened my cabinet again!

(140474)

La chocolaterie de Elektor.TV

Some time ago we featured the WunderBar at Elektor.TV, so if you want to see a live unboxing of this devkit, simply navigate to our channel [6]. Interested in weekly updates from the Labs? Don't forget to subscribe!

Internet Links

[1] http://relayr.io
[2] http://po.st/wunderbarhw
[3] https://relayr.io/wunderbar
[4] http://po.st/wunderbarapp
[5] https://developer.relayr.io
[6] http://po.st/wunderbarunboxing

TCA580 Integrated Gyrator

Peculiar Parts, the series

By **David Ashton** (Australia)

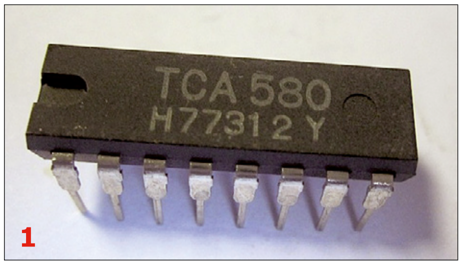
Some time ago when disassembling an old two-way radio console I came across an IC I didn't know: the TCA580. Looking up the datasheet, I found it was a gyrator IC.

My introduction to Gyrators was in Elektor issue 2 (February 1975, I still have it!) which had an article entitled "How to Gyrate" along with some drum simulators which made practical use of them. A Gyrator simulates an inductor. Recalling basic AC circuit theory, the currents in an inductor and a capacitor are in antiphase. The Gyrator looks at the current in a capacitor on an AC supply, and feeds back a larger current in antiphase to the supply. Hey presto, it looks like an inductor. The original article goes into more detail and a bit of the mathematics involved — it's given to you as a free download at [1].

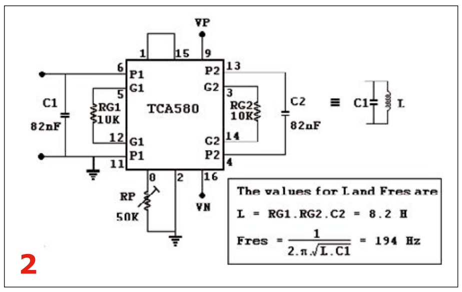
The TCA580

The TCA580 (**Figure 1**) is an integrated circuit gyrator IC produced by Philips/Signetics in the mid-1970s as a telecommunications building block. With the IC, one capacitor and two resistors, you can create a virtual inductor with a high and stable value, up to 1 megahenries (MH), and very high quality factor (Q >1000). Add one more capacitor and you have a tuned circuit which you can use as a notch filter or frequency detector up to 10 kHz, without the size, expense and poorer performance of physical inductors. These days you would be more likely to use Active Filters, Phase Locked Loops or Digital Signal Processing (DSP) to accomplish these tasks, but in the '70s this was an economic, achievable and effective way to do this.

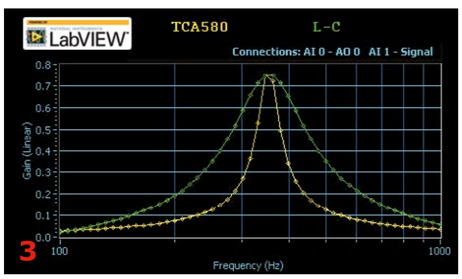
Data on the TCA580 is difficult to come by. There is a three-page datasheet widely available on the 'net [2], but it is short on detail and applications. It shows a 50-kΩ preset in the circuit but there is no mention of what it is for (it turned out to be for balance and to minimize distortion). I found reference to a 1977 Philips technical library volume titled *TCA580 Gyrator IC. A Replacement for Wound Inductors in Low Frequency Circuits*, but I cannot download or even buy this book anywhere. Nevertheless, the datasheet offers enough information to breadboard a tuned circuit with the IC, so I did this.



1



2



3

The circuit from the datasheet is shown in **Figure 2**. After initially getting the circuit to work with 47-nF capacitors for C1 and C2, I did a comparison with a discrete L-C tuned circuit (using a pot core inductor measuring around 14.4 H, and a 15-nF capacitor) that resonated around the same frequency (350 Hz). Wanting to compare apples with apples, however, I then used a 160-nF capacitor for C2 (the gyrator capacitor) and the same 15-nF capacitor that I used for the L-C tuned circuit for C1 (the resonance capacitor). Happily the resonant frequencies were almost identical. I used a 100-kΩ resistor between the source and the tuned circuit.

TCA580 *vs.* real L

Once I had my TCA580 circuit and my L-C circuit resonating at the same frequency, with the same 15-nF capacitor, I made a frequency response plot for both. These are shown in **Figure 3** — the TCA580 circuit response is in yellow and the L-C in green. You can see the steeper and narrower peak of the TCA580 response. Even using reasonable sized 1970's type components, the TCA580 circuit would take up less space and weight, and have a better performance than an L-C circuit. They would have been used for subtone (CTCSS) filters, detection and generation, etc.

TCA580's are still available on EBay for around $10 each — quite a lot these days for a DIL IC. I have a few of them removed from the old consoles, and while I don't know if I can use them for much these days, it was fun playing with what was — in the '70s — quite a high-tech IC.

(150024)

Web Links

[1] How to Gyrate article: www.elektor-magazine.com/150024

[2] Datasheet: www.datasheetarchive.com/dlmain/Databooks-4/Book618-6582.pdf

Tips and Tricks

From readers for readers

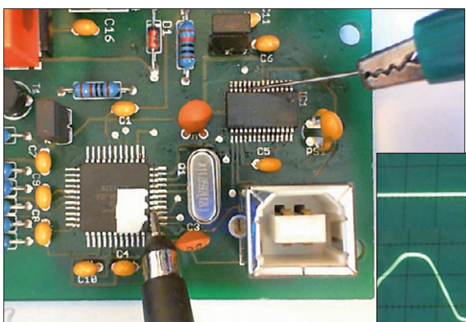
A selection by Burkhard Kainka

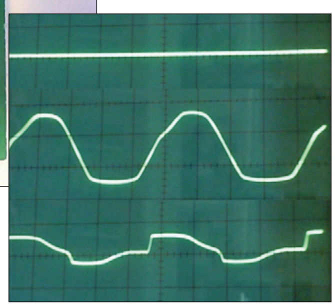
Here are some more neat solutions from our readers, sure to make life a little easier for engineers and electronics tinkerers.

Continuity Testing with a 'Scope

By Burkhard Kainka

Using a standard ohmmeter to troubleshoot a board stuffed with SMD components can be a bit tricky and time consuming. Try this new method; it uses an oscilloscope and a signal generator. Pressing the pointed tip of the scope probe onto the pad of interest, touch the tip of the signal generator output onto each pad which should have a connection to the first pad. Now it is fairly easy to see out of the corner of your eye whether there is a good connection, no connection or intermittent/faulty connection. The trace in the center indicates a good connection. Clipping on the lower half of the sine wave is likely produced by a chip's CMOS input protection diodes and is a good sign. Make sure that the leads are correctly earthed to the circuit. The more distorted waveform (below) indicates a bad connection.





Simple Battery Tester

From an idea by Eckhard Koch

The latest green LEDs are so efficient they produce a glow from a current of less than one microamp. This opens up new possibilities. As an example, to test a battery you won't need a resistor, just your skin resistance! Touch the battery's positive pole with a finger while holding the LED's anode and touch the LED's cathode on the negative pole of the battery. The photo shows it working, with a typical skin resistance of around 1 MΩ a current of a few μA flows through the LED.

Simple AVR port toggle

A tip from Andreas Riedenauer (Ineltek Mitte GmbH)

Writing a '1' to a bit in the PIN register, toggles (switches the state of) the corresponding port pin. An advantage of using this method is speed. The screen shot shows a test carried out using Bascom on the Arduino Uno's ATmega328 (it works!). Using a Blink program on the ATtiny13 shows that it works here too.

```
'------------------------------------------------------------------------------
'LEDblink.BAS
'------------------------------------------------------------------------------
$regfile = "m324pdef.dat"                                  'ATmega328p
$crystal = 16000000                                        '16 MHz

'------------------------------------------------------------------------------

Config PORTB = Output

'13.2.2 Toggling the Pin
'Writing a logic one to PINxn toggles the value of PORTxn,
'independent on the value of DDRxn.
'Note That The Sbi Instruction Can Be Used To Toggle One Single Bit In A Port.

Do
  PINB.5 = 1                                              'B5 toggeln
  Waitms 500
Loop

Do
  PORTB.5 = 1                                             'LED  on
  Waitms 500                                               '500 ms
  PORTB.5 = 0                                             'LED  off
  Waitms 500                                               '500 ms
Loop
```

A search of the data sheet finally revealed justification of the behavior (e.g. for the ATtiny13): The Port Input Pins I/O location is read only, while the Data Register and the Data Direction Register are read/write. However, writing a logic one to a bit in the PINx Register, will result in a toggle in the corresponding bit in the Data Register.

(150027)

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

HEF4000 Logic

Peculiar Parts, the series

By **Neil Gruending** (Canada)

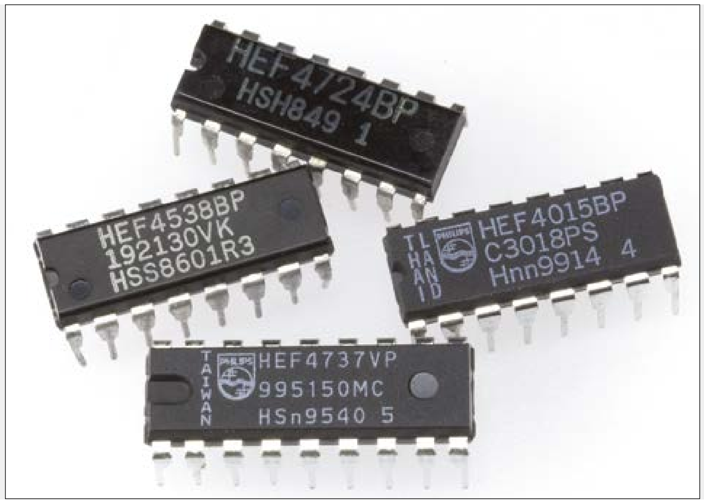


Figure 1. A few HEF logic devices, dusted off @ Elektor Labs.

Discrete logic gates are some of the most versatile circuit components available and are used in all kinds of circuits. You are probably familiar with the various 74xxx logic families but what about 4000 series logic chips? Let's take a look at the Philips HEF4000 series of logic ICs like in **Figure 1**. Amazingly, while LOCMOS (Local Oxidization Complementary MOS) technology was developed and refined in the early 1970's, ICs using it were not mass produced until the late 80's, and they are still available today.

The first mass produced generation of logic chips were the 7400 series that used TTL (transistor-transistor logic) and then the 4000 series logic using CMOS (complementary metal oxide semiconductor) technology appeared some 20 years later. TTL logic uses bipolar transistors which made them the fastest logic technology at the time. But the downside of bipolar transistors is that they require bias current, so TTL ICs always consume current no matter what the logic state of a gate may be. This isn't a problem with mains powered devices but is a real drawback when using battery power. CMOS logic addresses this by using FETs which only require current when switching logic states. But using FETs also meant that CMOS technology from that era was inherently slower than TTL.

Different manufacturers have used their own numbering systems for their versions of RCA's original CD4000 series logic, like Motorola's 'MC1' prefix. Philips as late as 1988 also produced 4000-series compatible logic originally dubbed HE4000. A wildly uninspiring family specification was released as late as 1995 for "filing under" Philips' renowned *Integrated Circuits Handbook IC04*. The plastic HEF devices were a success, their ceramic HEC, rarely seen. The HEF4000 series was special for its ability to operate at much higher frequencies. For example an RCA CD40174 hex flip flop was rated for 6 MHz input clock whereas the HEF40174 was capable of 15 MHz which was much closer to TTL speeds but at a fraction of the current (and dissipation). The increased speed really made a difference when powering them at lower voltages because 4000 series logic slows down with lower supply voltages.

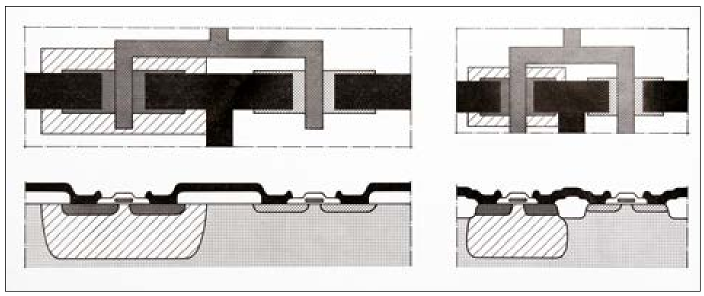


Figure 2. Spot the difference! Layer structure of traditional CMOS (left) and "all the latest" LOCMOS (right). Source: *Philips Technical Review, edition 1/1974*.

Another advantage of HEF4000 devices is that they integrate output buffers that improve their output symmetry and transfer characteristics with different output loads. The tentative LOCMOS layer structure shown in **Figure 2b** (1974!), enhanced later for use in the HEF4000 devices also reduces stray capacitances and allows for a smaller package size which also helps with their performance.

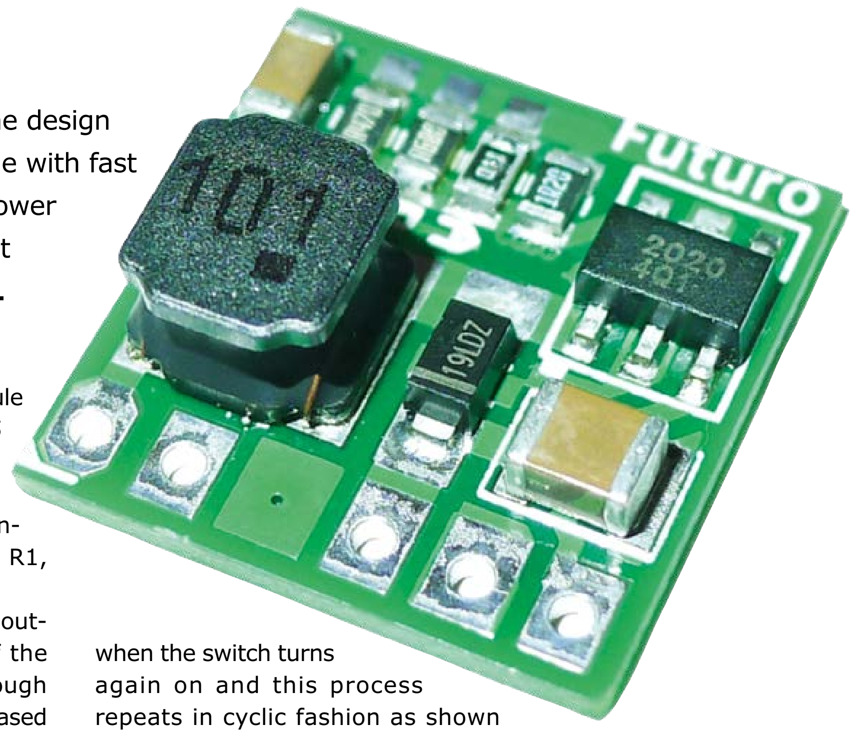
People will often choose 74HCxxx CMOS logic for new designs but HEF4000 logic is still useful too although now it's been replaced with HEF4000B logic. One obvious use is in higher voltage systems because HEF4000 logic is designed to be used in 5, 10 and 15 volt logic systems. HEF4000 logic can also offer substantial noise immunity for interfacing delicate circuitry to noisy environments. Electrostatic protection is also very good compared to run of the mill CMOS. Compared to 74xxx TTL, HEF4000 logic is *cool*. HEF4000 logic is a great choice to use when interfacing to A/D and D/A converters. Hopefully this inspires you to give them a try in your next design!

(150139)

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

High Efficiency Low-cost 0.5 A/33 V LED Driver Module

By **Valentin Kulikov** (FuturoLighting)



Here we outline the component selection and some design criteria for a simple constant-current driver module with fast PWM input, suitable for driving middle and high power LEDs, and operating from 8 up to 33 V with output current configurable in steps from 0.1 up to 0.5 A.

The schematic in **Figure 1** shows that the LED driver module is built around buck driver IC type TS19376 (in SOT89-5 package) produced by Taiwan Semiconductor. The device features buck driver hysteretic regulation enabling it to reach efficiencies exceeding 90% without need for compensations. The output current is set by paralleled resistors R1, R2, R3 to a rate of 0.13 Ω/A.

The principles of hysteretic (hysteresis-based) regulation outlined in [1] can be summarized: the internal switch of the TS19376 driver connects input voltage to the load through inductor L1. Current through the inductor gets linearly increased and monitored as a voltage drop on R1||R2||R3. Assuming a hysteresis of 15% (19.5 mV), once the current causes this voltage to be reached:

$$130 + V_{csn\_hys} = 149.5 \text{ [mV]}$$

the integrated switch turns off and current flowing through inductor and D1 linearly decreases till it drops down to:

$$130 - V_{csn\_hys} = 110.5 \text{ [mV]}$$

when the switch turns again on and this process repeats in cyclic fashion as shown in **Figure 2**.

The switching frequency is given by output current (I_{LED}), supply voltage (V_{cc}), the output voltage, and the value of L1.

PWM dimming

The average LED current can be controlled by a PWM signal. This method is popular and easily implemented helped by an MCU or by other techniques like a 555 timer (**Figure 3**). The PWM signal is connected to the module's PWM input and should be <0.3 V for Low, and >2 V for High (CMOS). The TS19376

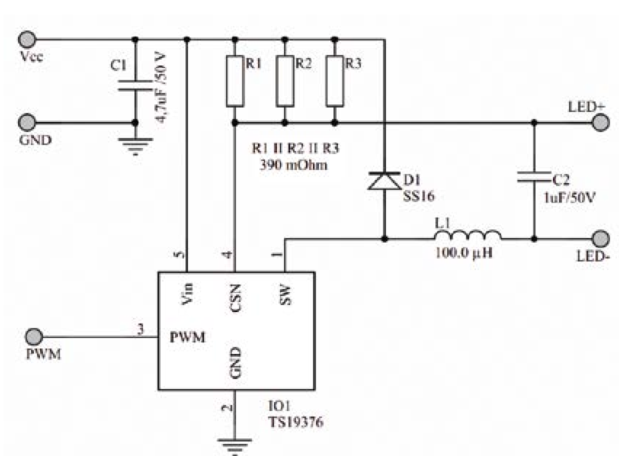


Figure 1. LED driver schematic.

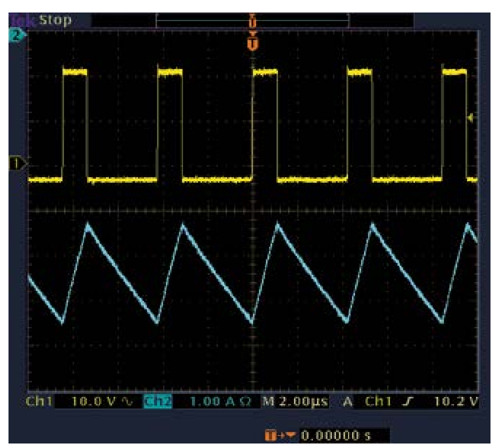


Fig. 2. Current and voltage waveform at the switching node (oscilloscope GND connected to V_{cc})

accepts relatively high PWM frequencies, so no problems realizing fast PWM dimming with more than 8-bit resolution.
Since the PWM input has a pullup resistor, with no PWM applied to the module, I_{LED} reaches the maximum current value. The recommended PWM frequency is >100 Hz to prevent visible flickering.

Practical realization

The TS19376 requires the usual copper layer cooling area at the back side of the PCB and thermally connected with the top side through vias. A low ESR input capacitor is required to suppress current spikes during driver switching. The recommended value for C1 is 4.7 up to 100 µF and the dielectric material should be X7R, X5R or better. C1 should be mounted as close as possible to the IO1 supply pads.
Optimal range of the L1 inductance is 47–120 µH, where lower inductance is more appropriate for higher currents and conversely, higher inductances are more appropriate for lower currents, where switching delay is eliminated. The placement of the other components should follow design rules aiming to obtain the 'smallest' switching loop and consequently minimum EMI. The start of the inductor winding should be connected to the switching node as well (SW pad of IO1).
D1 was selected to ensure low saturation currents at maximum operational temperature and low t_{rr}. D1's forward voltage affects efficiency, and lower V_f results in higher efficiency and lower heat dissipation.
It is recommended to use 30% margin for maximum forward diode current comparing to I_{LED}. In this case, an SS16 (1 A / 60 V), from Taiwan Semiconductor, was selected.
Capacitor C2 reduces output current ripple, where higher capacitance results in lower ripple and lower PWM frequency.
The TS19376 includes thermal shutdown. Once the die temperature reaches 150°C the driver is disabled until temperature drops below 115 °C. This protection is useful to prevent burning of the module PCB. The driver module can be attached to the heatsink by 2-sided thermoconductive tape (e.g. Bergquist Bond Ply). It is possible to extend the driver module with an EMI filter, reverse polarity protection (e.g. P-MOS switch), but this depends on specific application. The driver module is populated on a double-sided FR4 PCB with a thickness of 1 mm and dimensions 16x16 mm.

Conclusion

The LED driver described here has numerous applications ranging from driving middle and high power LEDs, through battery

Specification:

- Topology: Buck
- Regulation: Hysteretic
- Input voltage: 8–33 VDC
- Output current: 100–500 mA
- Switching frequency: 1 MHz max
- Current ratio: 0.13 Ω A⁻¹
- Dimensions: 16 x 16 x 5.5 mm (0.63 x 0.63 x0.22 in)
- Weight: 1.6 g
- Thermal shutdown
- Current protection
- Dimming: PWM up to 20 kHz

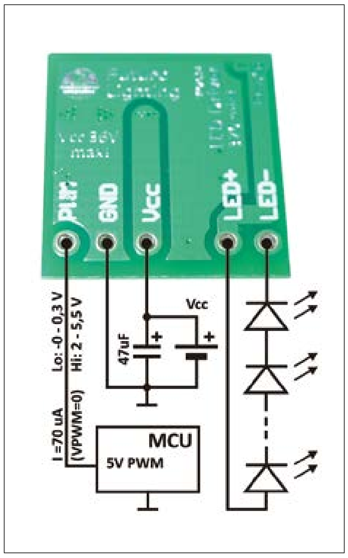


Figure. 3. LED Driver module connection diagram.

charging and much more where
a constant current source is required. The number of LEDs in the string is determined from the lowest allowed supply voltage (V_{cc}). As can be seen from the charts in **Figure 4**, keeping the string's voltage (V_{LED}) close to V_{cc} yields higher efficiency. For example for V_{cc} = 12 V, three LEDs in series is a good choice (V_{LEDF} ~ 3 V). All measurements were recorded in an automated measurement setup at room temperature.

The LED driver module with selectable output current from 0.1 up to 0.5 A is available for purchasing in FuturoLighting's store [2]. The TS19376 and diode SS16 can be purchased in MOQ from Microdis Electronics [3], authorized distributor of Taiwan Semiconductor.

Here I would like to thank Mr. Bilik from Würth Elektronik and Mr. Reguli from Microdis Electronics for their great support on this project.

(150048)

Literature

[1] www.taiwansemi.com

[2] www.fulit.eu/store

[3] www.microdis.net/

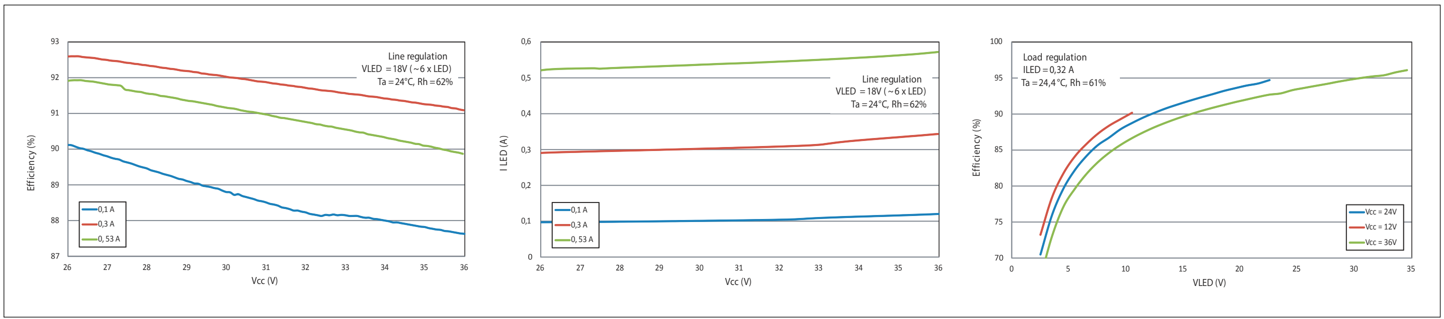


Figure 4. LED driver characteristics measured at different conditions.

Have U Heard of Automotive Varistors

AVX Corporation, has introduced a new series of high temperature, low leakage automotive varistors. Qualified to AEC-Q200, the new CANATL Series automotive controlled area network varistors exhibit extremely low leakage (≤1 µA at 32 VDC, 25 °C) and are tested, qualified, and specified at 150 °C, making them ideal for use in high temperature underhood automotive applications, such as transmission control units (TCUs), brake control modules (BCMs), data lines, bus interface units, and other capacitance sensitive applications.
Rated for use in temperatures spanning –55°C to 150°C, CANATL Series automotive varistors are currently available in a 0603 case rated for 32 VDC operating voltage. RoHS compliant, CANATL Series varistors are supplied with Ni barrier/100% Sn terminations and packaged on 7" or 13" reels. Pd/Ag terminations for hybrid assembly are available upon request.

(140525-IX)

http://avx.com/docs/Catalogs/lowleakagemlv.pdf

World's First RF / EMC Camera System

Aaronia presents the world's first RF / EMC "camera system", the "SPECTRAN RF VIEW". The system consists of a complex measurement unit at each "pixel". Each unit consists of a Spectrum Analyzer from the SPECTRAN RSA series and connected isotropic broadband antenna which are all connected via network to a central server.
The antennas are arranged in equal distance in an X / Y grid. The measurement data (amplitude and/or frequency) are visualized as a "chess-board-display". Each field of the chessboard represents a measurement unit.
An RF camera with 8x8 points (64 pixels resolution) requires 64 measurement units that way. This resolution seems to be rather small but already allows amazingly detailed measurements of antenna spreading characteristics or the graphical classification of radiated emissions during an EMC test.
The SPECTRAN RF VIEW saves considerable measurement time and allows very detailed information on the spreading patterns of emissions. The system can be extended to any size in order to reach much higher resolutions and is already successfully in use for research purpose and product evaluation of a leading telecommunications provider.

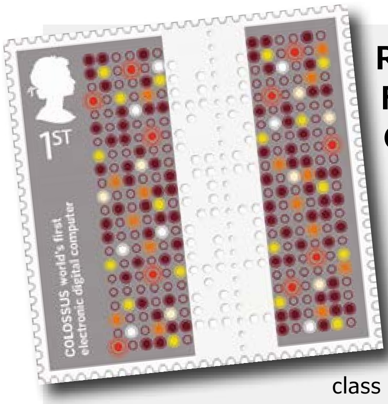
(150145-3) www.aaronia.com

MM7150 + Sensors = Easy Motion Monitoring

Microchip Technology Inc., announced from the Embedded World Conference in Germany the MM7150 Motion Module — which combines Microchip's SSC7150 motion co-processor combined with 9-axis sensors, including accelerometer, magnetometer and gyroscope in a small, easy to use form factor. With a simple I²C™ connection to most MCUs/MPUs, embedded/Internet of Things (IoT) applications can easily tap into the module's advanced motion and position data.
The motion module contains Microchip's SSC7150 motion co-processor which is pre-programmed with sophisticated sensor fusion algorithms which intelligently filter, compensate, and combine the raw sensor data to provide highly accurate position and orientation information. The small form factor module is self-calibrating during operation utilizing data from the pre-populated sensors; Bosch BMC150 (6-axis digital compass) and the BMG160 (3-axis Gyroscope). The MM7150 motion module is single sided to be easily soldered down during the manufacturing process. Microchip makes it easy to develop motion applications for a variety of products using their MM7150 PICtail™ Plus Daughter Board.

(150145-2) www.microchip.com/MM7150-Press-Presentation-022415a

... RS Components: new affordable inspection cameras ... SmartFusion2 SoC FPGA dual-axis motor control from Microsemi ... GlacialLight LED modular ceiling lightings ... TEA offers up to 800µ wafers ... Firmware update from embedded guru Michael Barr ... Just how good is the QorIQ Linux software development kit? ... Dialog Semiconductor's Bluetooth Smart Development Kit ... TDK Corporation: 300-watt 1/8th bricks ... Exablaze select Cypress'



Royal Mail Releases First Class Colossus Stamp

Royal Mail's Inventive Britain stamp issue celebrates eight key inventions of the past century in disciplines and applications ranging from materials to medicine. One of the two first class stamps in the series celebrates Tommy Flower's creation of Colossus. The series was released on 19 February 2015.

Colossus, the world's first electronic computer, was designed by Tommy Flowers to speed up the code-breaking of Lorenz-encrypted messages between Hitler and his generals. The Lorenz cipher was much more complex than Enigma and could take weeks to decipher by hand. By reducing code-breaking times to a matter of hours, Colossus enabled the Allies to learn of German war plans almost in real time. The knowledge obtained is widely recognized to have shortened the war and saved countless lives.

A working reconstruction of Colossus and the story of the breaking of Lorenz can be seen daily at The National Museum of Computing on Bletchley Park. The Rebuild of Colossus is now used in TNMOC's Learning Programme to inspire students to become the next generation of computer scientists and engineers.

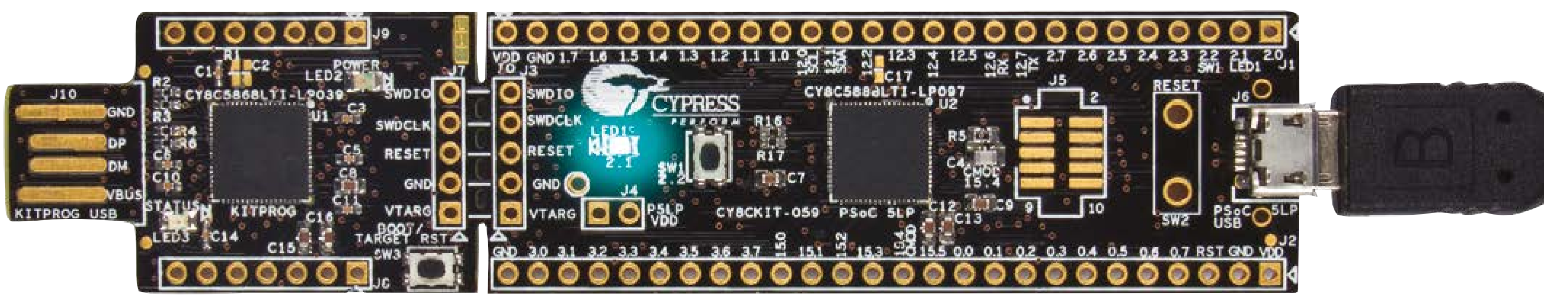
(150145-4) www.tnmoc.org

Pre-compliance EMC Probe Kits

The new TBPS01-TBWA2 EMC Probe Kit from Saelig Company, Inc. includes investigative near-field probes and a wideband amplifier to increase the versatility of economical spectrum analyzers to identify EMC issues. The economical TBPS01-TBWA2 EMC Probe Kit consists of four rubber-handled near-field probes (three H-field and one E-field), a 20-dB or a 40-dB wideband amplifier, and associated cables, supplied in an attractive wooden case. The shielded probes have built-in ferrites and insulated rubber handles to insure that measurements are insensitive to the human hand.

The included USB-5V-powered TBWA2 Wideband Amplifier, housed in a compact metal box (1.9″ x 2.6″ x 1″) provides either 20 dB or 40 dB amplification (depending on model) with a flat response from 10 MHz to 3 GHz, increasing the sensitivity of near-field probe measurements when attached to a spectrum analyzer.

(150145-8) www.saelig.com/category/MFR00154.htm



$10 Buys ARM Cortex-M3 PSoC

Cypress' new, low-cost prototyping kit gives engineers unprecedented access to design with the powerful 32-bit ARM® Cortex®-M3 core in Cypress's PSoC 5LP programmable system-on-chip architecture. The $10 CY8CKIT-059 Prototyping Kit delivers PSoC 5LP's ability to reduce systems costs by integrating analog front-ends, digital logic and user interface ICs with the architecture's programmable analog and programmable digital blocks. PSoC 5LP also simplifies power architecture design by supporting the industry's widest operating voltage range of 0.5 V to 5.5 V. The kit, combined with Cypress's free, easy-to-use PSoC Creator™ Integrated Design Environment (IDE) and hundreds of available example projects, enables rapid prototyping of designs and accelerates time-to-market.

The PSoC 5LP architecture provides unmatched processing performance with a 24-bit digital filter coprocessor and 24 Universal Digital Blocks — Cypress's programmable digital blocks containing two programmable logic devices and a programmable data path with status and control registers. A high-performance direct memory access (DMA) controller enables smart peripherals to operate in parallel to the ARM Cortex-M3 core, saving cycles and increasing throughput. PSoC 5LP one-chip solutions are ideal for a wide range of applications, including motor control, digital power management, switch mode power supplies, solar inverters and metering.

(150145-1) www.cypress.com/go/PSoC5LP

QDR-IV SRAM ... Non-magnetic RF contacts and modules from Molex ... It's here! The SEMI-THERM Program Guide ... Mouser signs global distribution deal with Digilent ... Celeb engineer Grant Imahara tours Mouser's distribution center ... 77 GHz radar transceiver for ADAS ... Green Hills software support for TI TDA2x ADAS ... TI MSP430i202x /3x/4x MCU has 24-bit ADCs ... Intersil: industry's smallest dual 3A/single 6A step-down power module ...

Miniature Snap-Acting Switches

C&K Components' new TF2 Series miniature snap-acting switches with a range of ordering options enable design engineers to select the ideal operating force, actuator style, termination type, and electrical rating for their specific applications.

The TF2 Series switches are available with the following selectable specifications:

- Seven operating forces (18, 45, 75, 110, 170, 230, and 330 g)
- Four electrical ratings (0.1, 6, 10, and 16 A)
- Two mounting styles (standard and metric)
- 15 actuator options (including: pin plungers, levers, lever rollers, simulated rollers, and simulated levers)
- Six termination options (two quick connect, two offset quick connect, screw, and solder)
- Three circuitry options: single pull double throw (SPDT); single pull single throw, normally closed (SPST NC); and single pull single throw, normally open (SPST NO).

All versions of the TF2 Series switches feature a 10,000 life cycle (minimum) at 16 A/250 VAC, and an operating temperature range from –40 °C to 125°C. They are RoHS-compliant and conform to IEC 61058-1 standards.

(150145-5) www.ck-components.com

13 Megapixel CMOS Image Sensor

Toshiba Electronics Europe has launched the T4KB3, a 13-megapixel backside illumination (BSI) CMOS image sensor with an optical format of 1/3.07 inch. Offering video capture at up to 120fps with full 1080p HD, the image sensor is ideally suited for use in high end smartphones and tablets.

With demand growing for small form factor chips, destined for ever slimmer mobile devices, the T4KB3 makes use of newly developed Toshiba design techniques to deliver the world's smallest 13 megapixel sensor. Low power circuitry has reduced power consumption to 53% of that used by the preceding T4K82 image sensor, and the new chip achieves 200 mW, or less at 30 fps, with full 13 megapixel output.

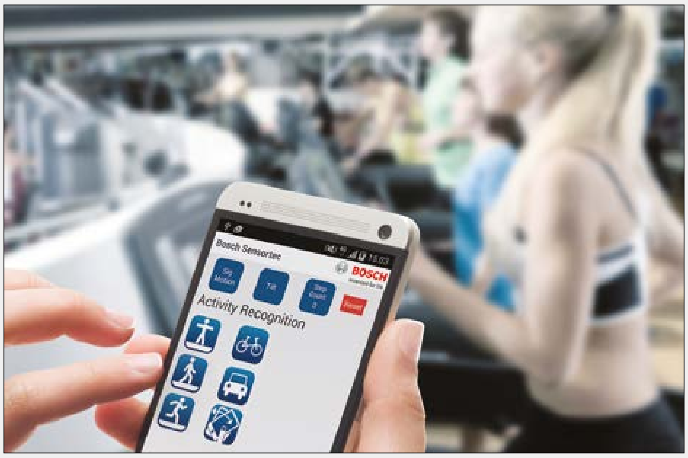
Image brightness is boosted four fold by utilizing Bright Mode technology and the T4K82 offers full 1080p HD video capture at 120 fps. The sensor is fitted with 8 Kbit of OTP memory and lens shading correction data for two conditions can be stored. This permits different correction data for indoor or outdoor use.

The T4K82 measures 8.5 mm x 8.5 mm, with auto focus, and 6.7 mm x 6.7 mm with a static focus. Samples are available now.

(150145-6) www.toshiba.semicon-storage.com

MEMS Sensors with Integrated Micro for Android Smartphones

Today's smart phones rely on always-on sensors for applications such as fitness tracking, step counting, indoor navigation and gesture recognition. By offloading sensor processing to the new Bosch Sensortec BHI160 or BHA250, as well as buffering sensor data locally on the devices, system designers can ensure the main application processor is never woken up just to process sensor data. This significantly reduces system power consumption, and thus extends battery life — a major competitive advantage for phone manufacturers.

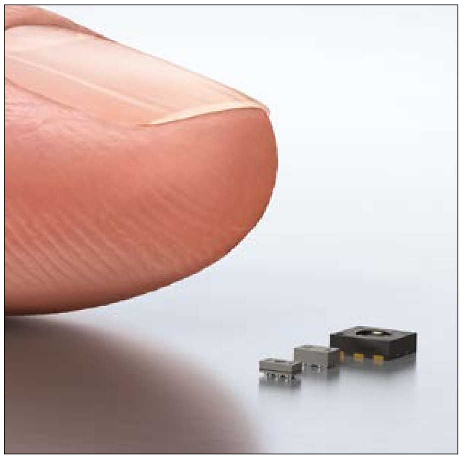


The new devices integrate a best-in-class 3- or 6-axis MEMS sensor with the new Bosch Sensortec DSP "Fuser Core". The BHI160 and BHA250 are specifically designed for applications in Android smart phones — implementing the full Android Lollipop sensor stack inside the devices where they provide a flexible, low power solution for motion sensing and sensor data processing.

The new devices implement the full Android Lollipop sensor stack and can be updated with new software features to support future releases. The BHI160 is the industry's lowest-power solution that is fully compatible with Lollipop, using less than 1.55 mA for a complete 9-axis solution including the Fuser Core, the integrated accelerometer and gyroscope and an external magnetometer.

(150145-7) www.bosch-sensortec.com

Smallest, Most Accurate Gas and Pressure Sensors

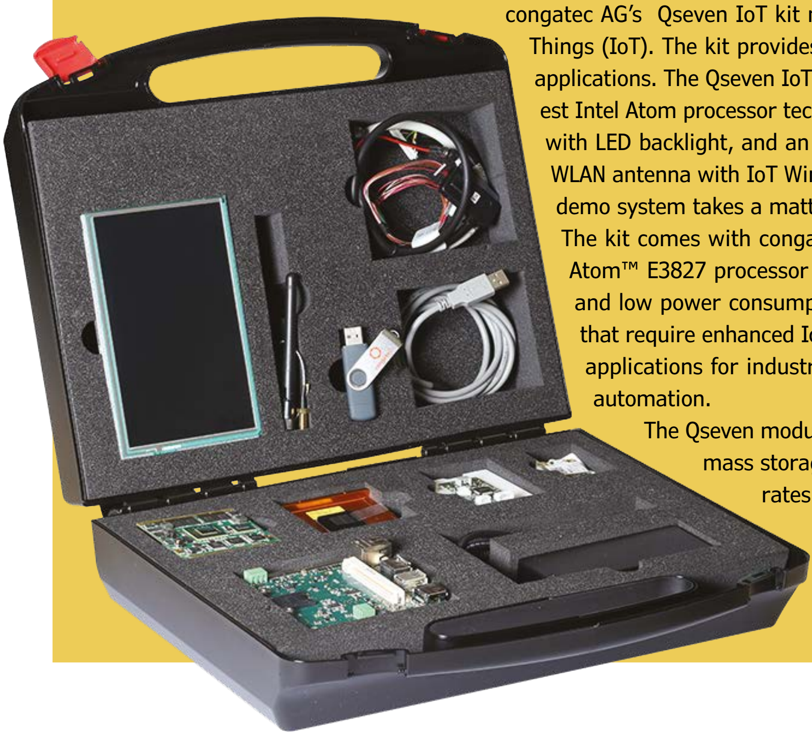


Sensirion's new gas sensor is the first in the world to be based on multi-pixel technology. This allows the sensor to perceive its surroundings using various receptors that, with the help of intelligent algorithms and state-of-the-art pattern recognition, are able to detect the type and concentration of gases. The flexibility that this provides means that now, for the first time, a single sensor is capable of detecting and distinguishing between different gases. Thanks to its very small dimensions of just 2.45 × 2.45 × 0.75 mm, the Sensirion multi-pixel gas sensor can be integrated anywhere. This will enable mobile devices to sense their surroundings in a way that was never possible before, for example in order to measure indoor air quality, determine the alcohol content of a person's breath, or recognize smells.Together with the gas sensor, Sensirion is also unveiling a new barometric pressure sensor offering unrivaled relative accuracy: it is capable of detecting altitude differences of as little as ±1 Pa, equivalent to the height of a single step on a stairway.

(150145-10)

www.sensirion.com/environmental-sensing

IoT Application Dev Kit



congatec AG's Qseven IoT kit makes it quick and easy to develop applications for the Internet of Things (IoT). The kit provides a complete starter set for the rapid prototyping of embedded IoT applications. The Qseven IoT kit contains a Qseven Computer-on-Module (COM) based on the latest Intel Atom processor technology, a compact IoT carrier board, a 7" LVDS single touch display with LED backlight, and an extensive set of accessories including AC power supply and 802.11 WLAN antenna with IoT Wind River Linux image on a USB stick. With this kit, developing an IoT demo system takes a matter of minutes.

The kit comes with congatec's successful conga-QA3 Qseven COM based on the new Intel® Atom™ E3827 processor (XM cache, 1.6GHz, XW TDP). A space-saving single-chip processor and low power consumption make this an ideal solution for fanless designs in applications that require enhanced IoT connectivity. These include, for example, M2M and motion control applications for industry 4.0, gateways, or system and control monitoring in smart home automation.

The Qseven module comes with 2 GB of DDR3L memory and up to 16 GB eMMC 4.5 for mass storage. Thanks to native USB 3.0 support, the module achieves fast data rates with low power consumption. A total of six USB 2.0 ports are provided, one of which supports USB 3.0 SuperSpeed. Three PCI Express 2.0 lanes and two SATA interfaces operating at up to 6 Gb/s enable fast and flexible system extensions.

(150145-11) www.congatec.com

Welcome to Elektor Labs

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

Our offer: Become Famous

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

Our Experts and Designers

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

Our Facilities

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

Our Standards

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

Our Products

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

Our Webinars

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

Our History

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

elektor labs

Sharing Electronics Projects

Search

Home | Proposals | In Progress | Finished

Log in

Upload your own projects!

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

Who's this for?

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

Welcome to the DESIGN section

Rethinking electronics design

Do you know Boldport? Someone I value recommended it to me, so I took my virtual plane and discovered an amazing and inspiring universe. I landed on a strangely shaped printed circuit board sporting traces that were neither straight nor snapped to a grid as we do where I come from, but instead were smooth curves winding like roads. Later, I learned that it is the map of Boldport with its sinuous mountain tracks in the West and its straight connector blocks in the East. Traveling further I passed small boards and colorful assemblies I had never seen before; I met the tiny engineer superhero emergency kit where parts are buried in the board. Then slowly the landscape started to change and became more familiar. A sign stated "we also know about signal integrity", I noticed a link to a Bluetooth IoT module, and I even discovered references to FPGA design. The language spoken in Boldport is unfamiliar to most of us rectangular PCB folks. Their language is called PCBmodE and it is a mix of JSON, Python, SVG and Gerber. You are invited to learn and practice it, and everything you need to do so can be taken home for free.

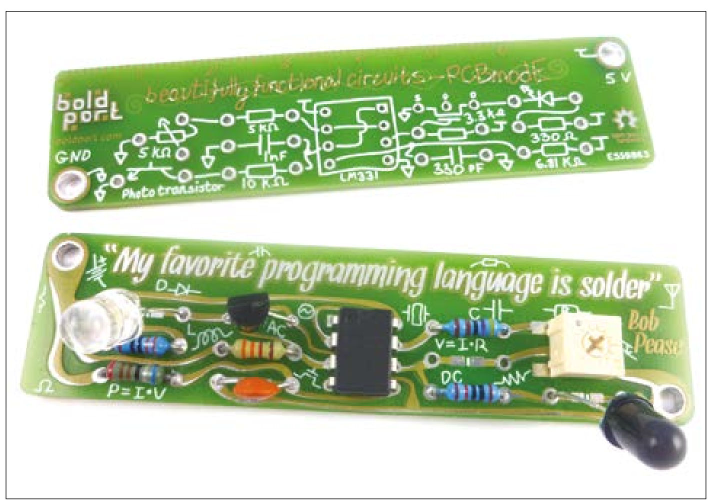
Saar Drimer is the Ruler of Boldport. He agreed to leave his country to meet me in person to explain Boldport's customs and habits, their way of life and how they see the future. www.boldport.com

Saar Drimer rethinks the way electronics design is done today. According to Saar the traditional design paradigm of schematic — net list — PCB with all its back annotation problems is obsolete, a relic from the past. We work like that because we have always done so. However, it is the end product that contains all the information and therefore it should be the starting point from which the design documents can be generated. It sounds like a paradox, but it is not. It boils down to top-down thinking instead of bottom-up.

Combining his passions for painting and electronics Saar Drimer developed his own way of creating printed circuit boards. Since PCB design is essentially drawing he prefers to use the drawing tool InkScape instead of a clumsy old-school CAD program with a steep learning curve. Using the EMACS text editor Saar creates JSON files to describe component footprints. Python scripts translate these into a scaled vector graphics (SVG) file representing the PCB. The drawing is finalized in InkScape before more Python scripts transform it into PCB production standard Gerber files. His suite of scripts is called PCBmodE, it is open source and available for anyone to download and use.

http://pcbmode.com

(150144-I)



Model Aircraft Autopilot

Experimental project for a motorized glider

See it at elektor-labs.com

By Camille Gay-Jourdan (Japan)
ccamyy@gmail.com

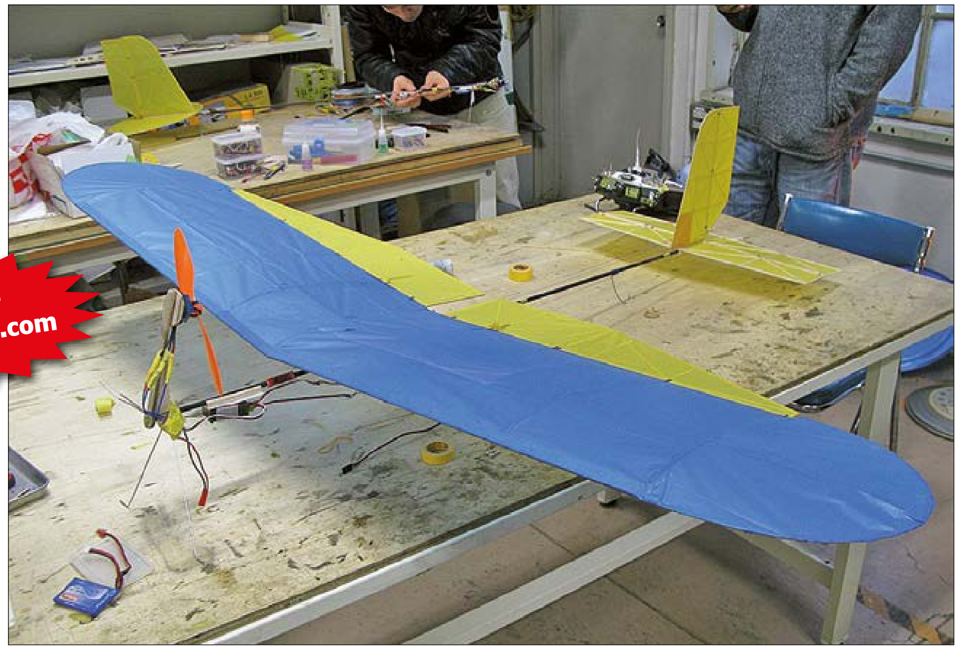
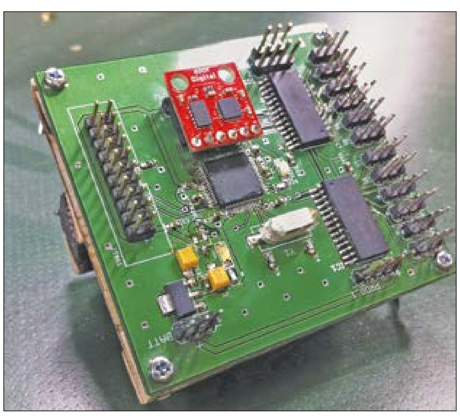


Photo of the indoor glider used for developing the Autopilot described in this article.

To fly a model aircraft is a satisfying experience, but to fly one with your hands in your pockets is… not as difficult as you would imagine. That's the experience I had in making this autopilot, and juggling the numerous constraints in doing so. I offer here a simple approach in the form of an autopilot circuit board which is customizable to suit your own (and your plane's) requirements.

The board is based on an ARM LPC2148 microcontroller supported by an accelerometer/gyroscope module, an altimeter and two PSoCs. Because of the great flexibility offered by the PSOCs, this basic configuration can easily be adapted to many projects for different models of plane. The development of the model itself is beyond the constraints of this article, but the curious reader will find lots of information in the works of model experimenters [1]. A basic knowledge of aeronautics is required to tackle this project, so let's first refresh our memory on that.



Assembled prototype board with the accelerometer/gyroscope module (red p/o board)

Basics of aeronautical theory

An aircraft is subjected to four principal forces:

- the thrust of the motor, in the direction of movement;
- drag, or air resistance, against the direction of movement;
- lift, another effect of the air, perpendicular to the direction of movement;
- gravity.

The aircraft can make rotations about three axes (**Figure 1**):

- the transverse axis, called pitch. A rotation about this axis will cause the plane's nose to pitch up or down;
- the vertical axis, called yaw. A rotation about this axis will cause the plane to turn to left or right. The wings remain horizontal;
- the longitudinal axis, called roll. A rotation about this axis will cause the plane to turn about itself, like rolling a barrel.

To make these rotations, an aircraft is equipped with 3 control surfaces which, by disturbing the forces that are present, give rise to a *moment*:

- Rotation about the pitch axis is done by the elevator. This is a flat surface

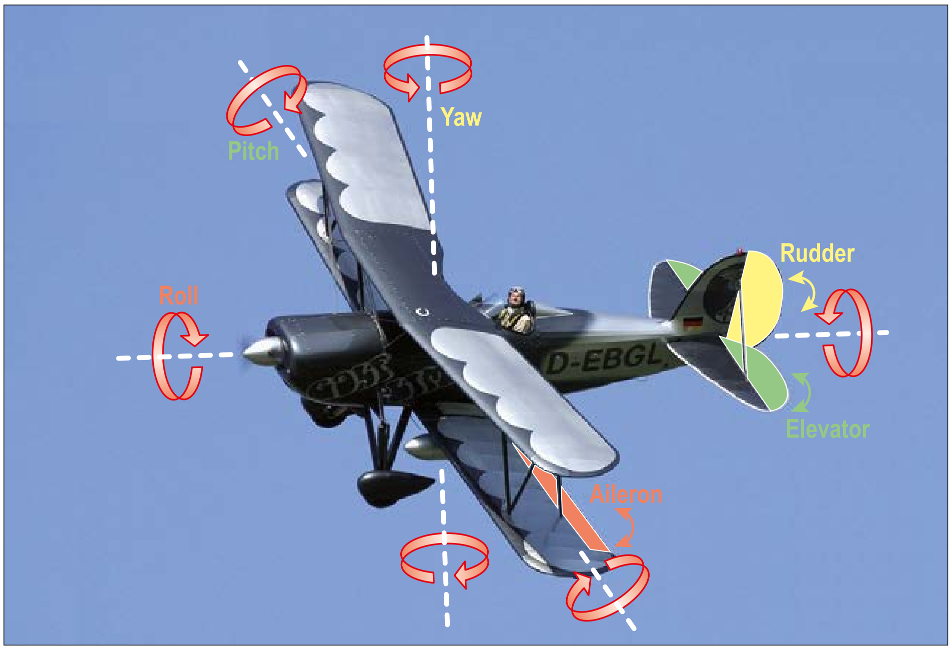


Figure 1. The three axes of movement of an aircraft.

at the rear of the aircraft, parallel to the horizon, which the pilot can turn about the transverse axis.

- Rotation about the yaw axis is done by the rudder. This is a flat surface at the rear of the aircraft which the pilot can turn about the vertical axis.
- Rotation about the roll axis is done by the ailerons. The ailerons are two flat surfaces, at the rear of the wings, parallel to the horizon, which the pilot can make a turn about the transverse axis, in opposite directions.

These axes and the corresponding control surfaces are shown here for a classical aircraft form. It's important to note that the rudder theoretically allows the aircraft to turn left or right, but in practice a turn is done by inclining the aircraft about the roll axis. The rudder is used to stabilize the aircraft and prevent a 'skidding' motion.

Mathematical modeling

An aircraft is a non-linear system, governed by a large number of parameters. Because of this, it is necessary to make certain assumptions for simplification. Firstly, we will consider the motor speed to be constant and we'll only fly inside, where there is no wind. So the only parameters to worry about are the three rotations of the aircraft.

We are thus dealing with a system with three inputs (elevator, rudder and ailerons) and three outputs (pitch, yaw and roll). So let's consider how these parameters are related.

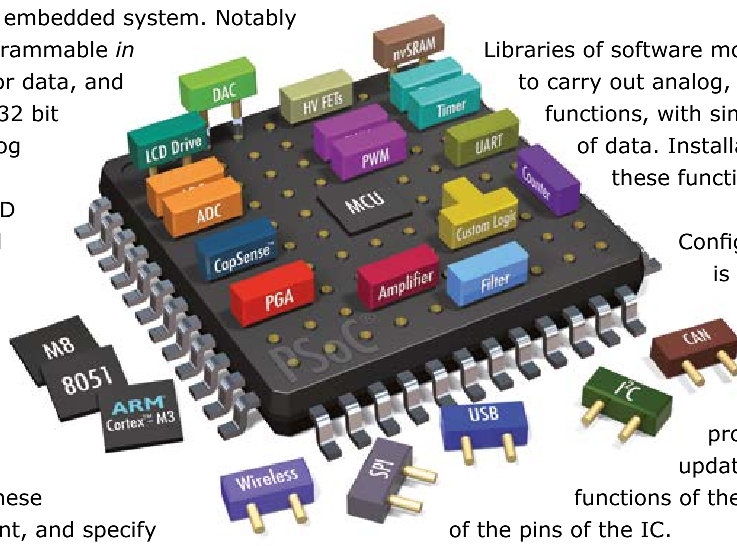
We will for the moment consider pitch completely independent of yaw and roll [2]. We are then left with an independent 'elevator → pitch' control loop (**Figure 2a**).

Yaw and roll are m
ore or less related as a function of a parameter of the aircraft structure: the dihedral, which is the angle between each wing and the horizontal plane. If this angle is zero (wings parallel to the horizontal, which is the case with aerobatic planes) the relationship between yaw and roll is very slight. If the angle is non-zero (wings in the form of a V, which is the case with gliders) the relationship is appreciable.

If the model aircraft has a zero dihedral, one can use two independent control loops 'rudder → yaw' and 'ailerons → roll', similar to that shown in Figure 2a.

PSoC (*Programmable System on Chip*)

These circuits are programmable and configurable by the user ; they were developed to replace a microcontroller and for peripheral circuits in an embedded system. Notably they offer a Flash memory programmable *in situ*, a static memory (SRAM) for data, and a hardware 8x8 multiplier with 32 bit accumulator. They contain analog and digital configurable blocks which are user-configurable (A/D and D/A converters, operational and instrumentation amplifiers, programmable filters and comparators, counters and timers, UARTs, I²C and SPI bus controllers, an EEPROM, etc.)



The user can choose which of these functions he wishes to implement, and specify the type of signal for each pin of the IC (input, output, analog, digital…).

Libraries of software modules allow the PSoCs to carry out analog, digital or mixed signal functions, with simple or complex treatment of data. Installation and programming of these functions can be done *in situ*.

Configuration of then PSoCs is dynamic: on power on, stored configuration information is written to the SRAM registers directly by the application program, which can also update parameters to change the functions of the blocks, or the assignment of the pins of the IC.

However, it is more probable that a model aircraft will have a strongly positive dihedral, and thus a strong coupling between roll and yaw. We can attempt to model these couplings and take account of them in the controller. We can also opt for a mechanical trick, in the form for example of opposed ailerons, which remove the need for yaw. The simplest solution is to adopt two-axis control. This consists of ignoring the ailerons, and using the rudder to control roll indirectly, due to **induced roll** by the rudder 'rudder → yaw → roll' (**Figure 2b**). If we do this, the yaw is not independently controllable, but as we shall see, this isn't a problem, because yaw is not so useful when piloting. It is this method which we will use for our first autopilot flight.

Lastly, for each control loop we will use simple and proven control method: Digital PID control (for proportional, integral, differential). If you want to model your aircraft and come up with a precise control circuit, you can refer to the examples on the net [3].

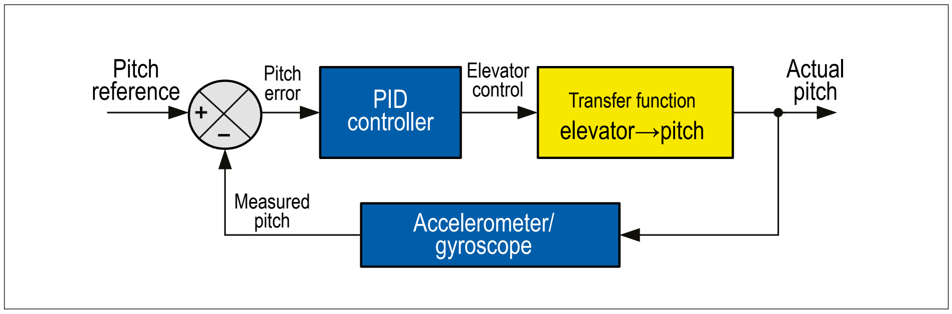


Figure 2a. PID control loop for control of pitch.

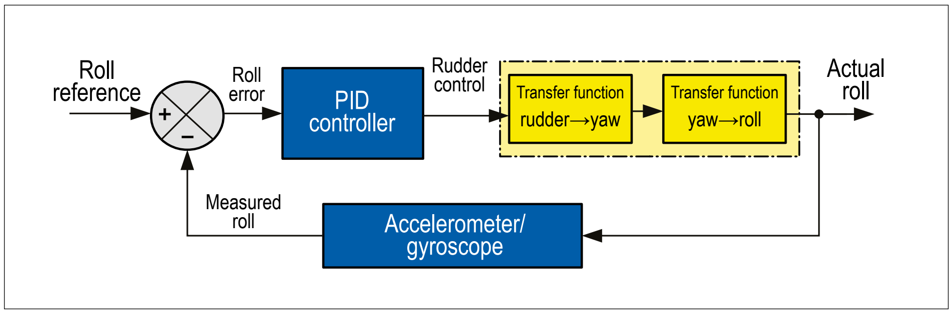


Figure 2b. PID control loop for control of roll

Architecture of an aircraft

The body of the aircraft, as designed by Kazuya Yaginuma, is based on a carbon fibre rod. The wings have a framework of light wood and a covering of paper. The electrical components of the model are:

- 1 battery (Lithium-Polymer);
- 1 8-channel RF receiver;
- 1 propeller (brushless motor and its control circuitry);
- 4 servos to activate the control surfaces (1 for the elevators, 1 for the rudder, 2 for the ailerons);
- 1 autopilot board.

The aircraft is piloted with a remote control with levers and switches. The RF receiver outputs a pulse width modulation (PWM) signal where the duty cycle indicates the angle of a lever or the position of a switch. Under manual flight, these signals are sent direct to the servos. Under autopilot, the autopilot board is interposed between the RF receiver and the servos, and replaces the receiver's PWM signals with its own signals. The piloting mode (manual or automatic) is selected by two switches on the remote control: when both are set to 0, the aircraft is in manual mode.

Architecture of the autopilot

The autopilot board comprises:

- 1 ARM LPC2148 microcontroller, supplied from 3.3 V, but compatible with 5 V, clocked at 12 MHz, its 3.3 V regulator (LM1117) and its power supervisory circuit (MCP130T) [4];
- 2 PSoCs CY8C29466, supplied from 5 V, but compatible with 3.3 V;
- 1 ITG3200/ADXL34 I²C accelerometer/gyroscope module from Sparkfun;
- 1 28015 PING))) altimeter module from Parallax.

The modules are mounted horizontally and face-up. The accelerometer/gyroscope module is inserted directly onto dedicated pins on the autopilot PCB. The altimeter module is mounted independently and connected by a cable. The board is attached to the aircraft using a gel to minimize vibration.

My autopilot board gets its 5-V power from the motor control circuit, via a 5-V regulator. A Lithium-Polymer battery has some dangers. We should use some standard security precautions (protection against reverse polarity, static electricity, etc.). We should choose reliable components (avoiding for example tantalum capacitors which are prone to short-circuits). Before connecting the Lithium-Polymer battery, one should test with another protected source of power.

The software of the LPC2148 runs a loop as follows:

- determine pitch and roll values (as a function of the chosen flight mode);
- read the date from the accelerometer/gyroscope module over I²C;
- calculate the actual values of roll and pitch;
- use the PID algorithm to calculate the control values;
- send corresponding PWM signals to control the servos.

The role of the PSoCs is to:

- change the PWM signal levels (3.3 V from the LPC2148 to 5 V for the servos);
- act as a multiplexer for the PWM signals. Under manual piloting, the servos are controlled by the PWM signals output by the RF receiver. In automatic mode, the servos receive PWM signals from the LPC2148. The PSoCs are used for routing the correct signals.

As the PSoCs have an integral microcontroller, they look after several functions:

- PSoC 1 translates the PWM signals showing the state of the switches from the remote control in the TOR signal from the Autopilot board.
- PSoC 2 communicates with the altimeter and sends the data to the LPC2148 over I²C. The altimeter uses ultrasound: it sends a pulse and measures the time for the signal to return. As the altitude of the aircraft is not needed to determine its attitude in the air, I have not used the altimeter in this version of the software.

The essential role of determining the status of the board (under either automatic or manual piloting) is carried out by PSoC1. It is sensible to give this function to a component external to the LPC2148. Thus, even in the case of a software crash, the user can change back to manual mode.

The block diagram of the electronics is shown in **Figure 3**. The configuration here shows the case where the controls for elevators and rudder are under the control of the LPC2148, and the motor and the ailerons are controlled manually. The printed circuit board is probably more cluttered than is necessary, because the routing on the prototype has not been optimized. If my project interests you, I would strongly recommend you to optimize the routing, not only to de-clutter the board, but also to allow the program-

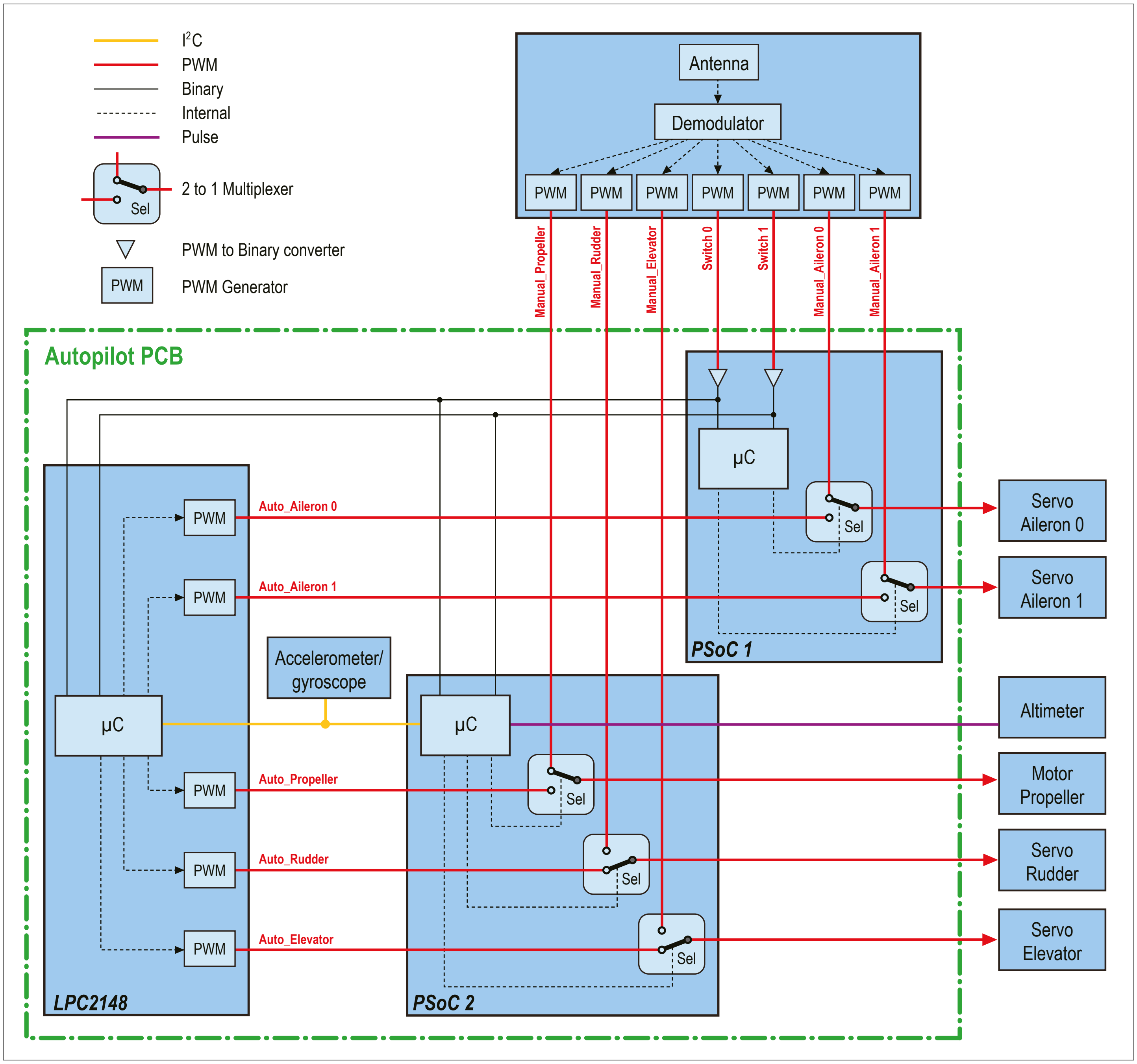


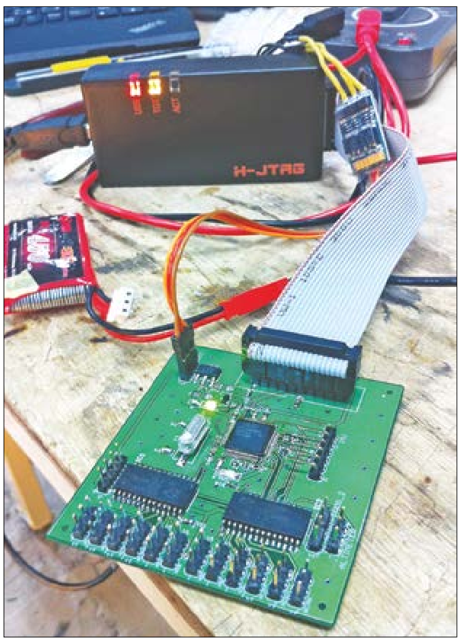
Figure 3. Detailed schematics of the project are found on the Elektor Labs site [5], but a block diagram is shown here to give a rough idea of how it works. However this diagram shows very well how the main blocks of the autopilot interact with each other.

If you are flying your powered model plane on autopilot in the open air, ensure you conform to the prevailing legislation, as well as the frequency bands that are allowed for this.

ming of the PSOCs without using ports P0_i and P2_i (where i = 0–7) simultaneously as PWM inputs, without which it is impossible to use the digital blocks of the PSoCs.

Flight testing

The initial tests were done inside only, and the coefficients of the PID controller were fixed empirically. It is wise to undertake the first tests close to the ground and at low speed, because too high a gain may tend to make the aircraft oscillate, risking a stall. PD control has been shown to be more effective than PID control,



LPC2148 programming

which is prone to oscillation, probably due to the limits of digital integration. To avoid making things too complicated, the altimeter has not been used for the moment.

This model, although very simple, has been shown to be stable. With it I have implemented three types of automatic flight:

- straight line (pitch = 20°, roll = 0°)
- circling (pitch = 20°, roll = 30°)
- figure 8 (pitch = 20°, roll = +30°/–30°)

The Author

After my Electrical Engineering studies at INSA in Lyon (France) and my last year at the University of Tokyo, I am an international volunteer in Japan at Ichikoh Industries, a maker of automobile lighting. I have worked in image processing and software development in Tokyo and Paris.

Japan is a country often subject to natural catastrophes. The purpose of this project, in collaboration with other amateurs at the University of Tokyo, was to show the suitability of drones for reconnaissance operations and for airdrops of medical supplies in disaster areas.

It is easy to modify the software to add many other functions, for example:

- Take measures in flight to determine the coefficients of the mathematical model of the aircraft. The PSoC1 can serve as an EEPROM to save the data.
- Implement automatic learning to determine the PID coefficients in open flight.
- Add control of the ailerons, motor and use the altimeter.
- Add other flight modes: automatic takeoff and landing, flying at low altitude, etc.
- Use the PSoCs as external watchdogs for the LPC2148 and for the RF Receiver. In case of loss of signal, PSoC 1 could initiate an urgent landing procedure (making use of its I^2C connection to the accelerometer/gyroscope module).

The project source code files and the printed circuit board files are available at www.elektor-labs.com. Please don't hesitate to contribute to this project.

(140204)

Web Links

[1] General information about radio controlled aircraft: http://en.wikipedia.org/wiki/Radio-controlled_aircraft

[2] Example of a transfer function [elevator à pitch]: http://goo.gl/EPkm6H

[3] Course on aerodynamics: http://goo.gl/3jShU5

[4] Reference design for the LPC2148: http://goo.gl/u4e0hW

[5] Project page on Elektor Labs: http://www.elektor-labs.com/project/model-aircraft-autopilot.13945.html

P-W-M Doorbell

(Programmable, Wireless, Musical)

Bit-bang your personal melody

By **Victor Hugo Pachon** (HUGO Tecnologia) (Colombia)

The e-fun you get from any wireless doorbell bought at Wallmart & Co. is as short-lived as the first melody or chime that sounds when the battery is installed. We electronicz people want the contraption to be personal, meaning programmable not just at the PIC level but also with crude though at least, original melodies.

El-cheapo wireless doorbells across the globe are so unoriginal it makes you wonder how much more Far Asian rubbish we can tolerate in and around our homes. The ding-dong, two-tone chime, angry dog *ersatz* and other “sounds” are corny and badly reproduced; they sound cheap and are so widespread in some neighborhoods people do not recognize their own doorbell sound which defeats the purpose. But to personalize these things — impossible.
In this article we put forward a doorbell that is as open as electronics engineers can want it to be: the system is programmable in terms of firmware (PIC), channel encoding (hello Holtek HT12), and music (*doh-reh-meh* switches, tables … love it).

Features

- User-created melody, memory resident, max. 127 notes
- 8-bit note/octave/length programming
- 433-MHz LPR wireless control
- Off-the-shelf radio modules
- PIC16F873A controller
- Holtek HT12 data encoding
- All through-hole project
- No USB, No MP3

Transmitter

As opposed to the classic doorbell with its transformer, buzzer or solenoid, and the long wire to the illuminated pushbutton at the door, we’re dealing here with a wireless system comprising a battery-operated transmitter mounted at the door and an associated receiver mounted indoors, usually in the hallway or the kitchen for all to jump up at the annunciating sound.
The schematic in **Figure 1** shows the transmitter. When the doorbell touch button connected to K5 is pressed, high-gain amplifier T5-T6 causes LED2 to light and serial data encoder IC7, a Holtek HT12E, gets enabled via a Low level at its $\overline{\text{TE}}$ pin (14). At the same time, transmitter module IC8 gets its supply voltage through pnp transistor T7. A 433-MHz license-free, type-approved LPR transmitter type QAM-TX1-433 is used here, which should provide a range of about 50 m (150 ft) indoors. QAM stands for quadrature amplitude modulation.
The personal code used on the radio channel is set on configuration block K7, which determines which of the word encoding pins A0 – A7 on the HT12E are High (default) or Low (jumper fitted). The code set on the TX must match that on the receiver you want to respond, and is secret for obvious reasons. The Holtek HT12E/D wireless encoder/decoder chip set is an industry standard device pair, widely available at low cost and easy to implement [1],[2]. The device type coding is disarmingly simple: *HT* for HolTek, *12* for 2^{12} encoding possibilities (hard pushed), *E* for encoder, *D* for decoder.
The transmitter is powered off a 3-volt CR2032 coin cell (BAT1). The TX antenna is a 17-cm (quarter-wavelength) piece of stiff isolated wire connected to ANT2. It is recommended to have the antenna wire stick out freely from the case, and to keep it straight and vertical. The same applies to the receiver.
In countries where 433.92 MHz LPR is not legal, the radio modules suggested here should be replaced with functionally compatible devices (868 MHz, 315 MHz, 915 MHz depending on region).

Receiver

The receiver whose schematic is pictured in **Figure 2** is powered by a wall adapter with 9 volts DC out at 100 mA or so. A trusted 7805 steps the 9 VDC down to +5 V regulated for the logic circuitry and the receiver module. The unregulated 9 volts is used to power the small audio amp around T2 and T3.
Once picked up by ANT1 the radio signal from the doorbell transmitter gets mixed

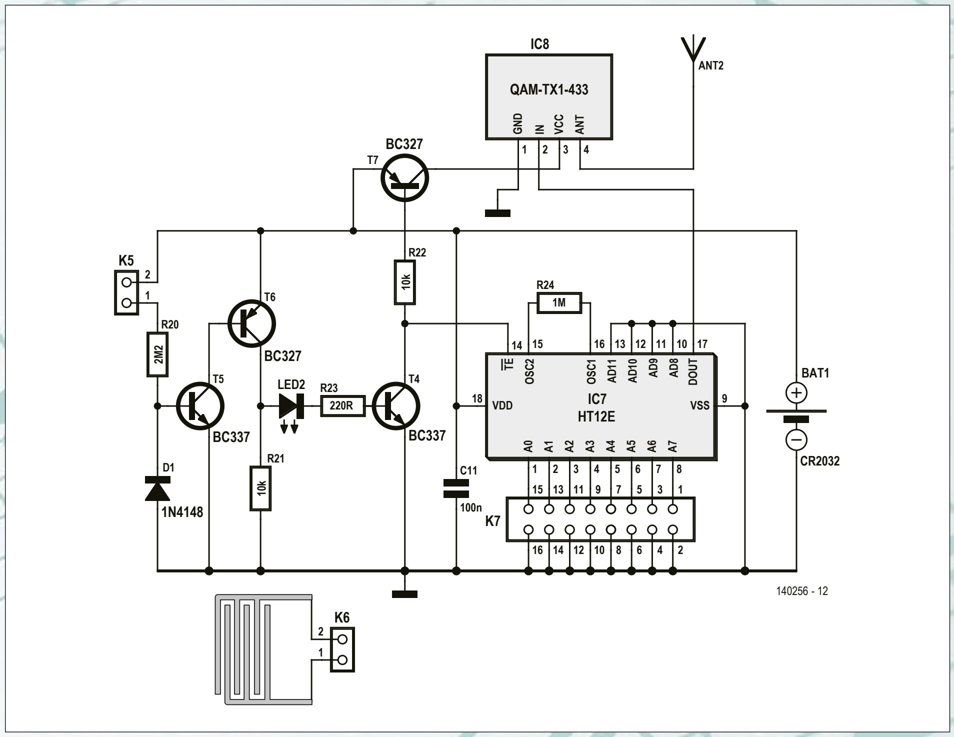
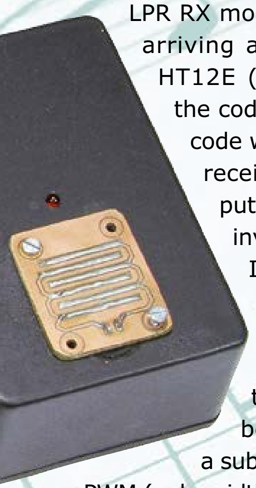


Figure 1. Transmitter schematic.

down and demodulated to recover the original datastream by IC2, a QAM-RX2-433 LPR RX module. In the data bursts arriving at the DATA IN pin the HT12E (IC3) constantly seeks the code word set on K8. If the code word matches that in the received signal, the VT output is swung High. Through inverter IC4.A, NAND gate IC4.C, and the RB7 input, the PIC micro in (IC5) is flagged that someone has pushed (or touched) the 'one and only' doorbell switch. Also, it starts a subprogram that produces a PWM (pulsewidth modulated) datastream on port line RA2 that hopefully sounds like a coherent bit of music. The melody is homebrew as far as the notes sequence goes, and you have yourself to blame if it sounds ugly (try *Anarchy in the UK*, *Eastenders SignOff*, *My Doorbell*, *Für Elise*). The melody is programmed note-for-note using S3, LED1, and 8-way DIP switch bank S1 connected on K2 and shown separately in **Figure 3**. The melody composer on K2 is enabled under software control via port line RA4. The melody remains stored in the PIC.



The PIC16F873A is clocked at 4 MHz using quartz crystal X1 and load capacitors C9/C10. It is automatically reset at power on by network C6/R4.

Pushbutton S3 allows the melody to be programmed as well as to sound under local control i.e. without pressing the doorbell button. The switch should not be readily accessible.

Slide switch S2 offers two basic volume levels to be set on the loudspeaker LS connected to K4.

Construction

The receiver, transmitter, composer, and touchswitch boards are cut out from a single panel pictured in **Figure 4** along with the **component lists**. Note that the separate brownish boards shown in photographs here are prototypes produced in-cellar by Elektor Labs using their Colinbus PCB milling machine. The

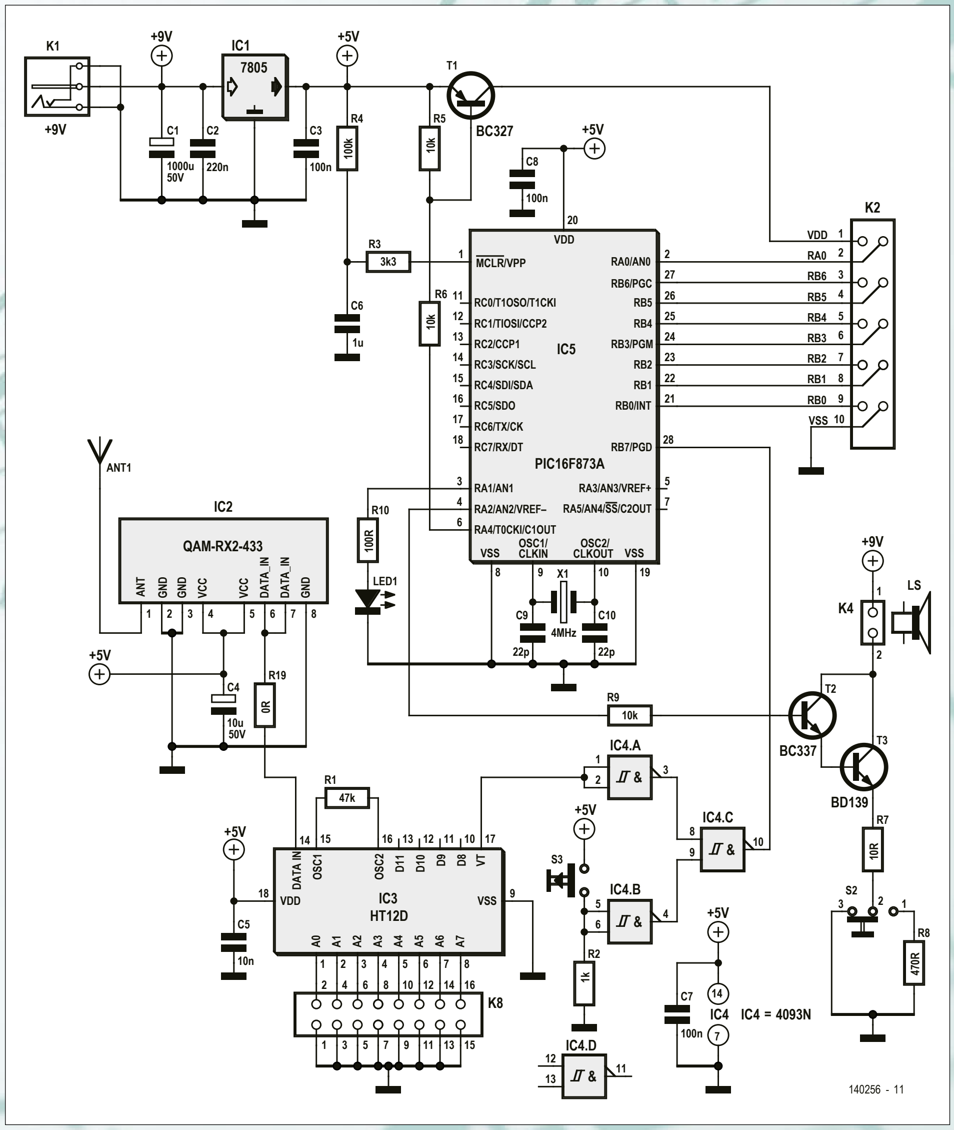


Figure 2. Receiver schematic.

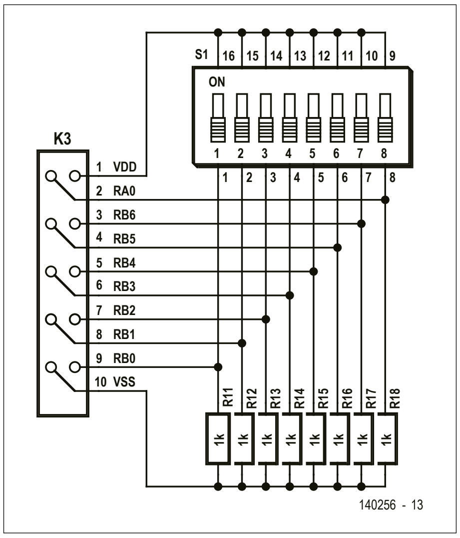


Figure 3. Admittedly the notes composer for the P-W-M Doorbell is not the most advanced of programming tools. It does the job though.

boards you obtain through the Elektor Store are green, high quality, and have a silkscreen printed component overlay. We recommend using sockets for all DIL integrated circuits.
All three sub boards are spaciously laid out and besides the TX and RX radio modules, use through-hole parts only, hence our confidence in everyone with reasonable soldering skills but no special tools being able to build the project successfully.

Table 1. P-W-M Doorbell, notes programming

			Bits on S1							
			1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Note	Silence		0	0	0	0	×	×	×	×
	doh	*C*	1	0	0	0	×	×	×	×
	doh#	*C#*	0	1	0	0	×	×	×	×
	reh	*D*	1	1	0	0	×	×	×	×
	reh#	*D#*	0	0	1	0	×	×	×	×
	mih	*E*	1	0	1	0	×	×	×	×
	fah	*F*	0	1	1	0	×	×	×	×
	fah#	*F#*	1	1	1	0	×	×	×	×
	soh	*G*	0	0	0	1	×	×	×	×
	soh#	*G#*	1	0	0	1	×	×	×	×
	lah	*A*	0	1	0	1	×	×	×	×
	lah#	*A#*	1	1	0	1	×	×	×	×
	tih	*B*	0	0	1	1	×	×	×	×
Octave	Low / Down		×	×	×	×	0	×	×	×
	High / Up		×	×	×	×	1	×	×	×
Note Length	1/64th		×	×	×	×	×	0	0	0
	1/32nd		×	×	×	×	×	1	0	0
	1/16th		×	×	×	×	×	0	1	0
	1/8th		×	×	×	×	×	1	1	0
	1/4th		×	×	×	×	×	0	0	1
	1/2		×	×	×	×	×	1	0	1
	Full		×	×	×	×	×	0	1	1

Web Links

[1] Holtek HT12E encoder: www.holtek.com.tw/english/docum/consumer/2_12e.htm

[2] Holtek HT12E decoder: www.holtek.com.tw/english/docum/consumer/2_12d.htm

[3] Project software: www.elektor-magazine.com/140256

[4] Elektor Labs spot: www.elektor-labs.com/project/musical-programmable-bell.13959.html

The size of the transmitter board allows it to be installed in a Hammond 1591 XXL case, with the touchswitch little board mounted on the outside, with LED2 protruding from a hole just above the touchswitch. If you want to use a regular pushbutton switch at the door, go ahead. The receiver board got designed to fit in the much larger Hammond 1591 XXG ABS case. The case also acts as a small but effective loudspeaker cabinet.
The melody composer board is built separately and connected to the receiver by a length of 10-way flatcable and matching IDC connectors at the ends.

Programming and use

And now, for the programming of "your" doorbell sound, riff, melody, sound bite or soundscape, whatever. The lazy among you will have their musically gifted daughters grab the score of a *Für Elise*-ish tune from the Internet and bang the byte transcription into the PIC. Even lazier Elektor readers can enjoy *Jingle Bells* if they purchase the ready-programmed PIC from the Elektor Store (#140256-41), courtesy Niek & Jan @ Elektor Labs.
The relation between the notes, octaves and note lengths on the one hand and the bits of each 8-bit word to be pushed into the PIC memory, is given in **Table 1**. This the crux of the P-W-M Doorbell.
Here's the programing sequence, assuming you have the 8-bit words written out and handy for "your song" of 127 notes maximum, see Table 1. Here goes:

1. After power-on LED1 lights for a moment. During this time, press S3 to enter programming mode;
2. Have the 8-bit word for the first note available on DIP switch S1;
3. Press S3 to program the 8-bit word into the PIC;
4. Flip, toggle or slide the DIP switches to configure the next note word;
5. Press S3;
6. GOTO 4.

Program the pseudo word *FF* (All ON; 1111 1111) and push S3 to mark the end of your melody. The maximum melody length is 127 notes. Our Retronics editor said: " 'Tis like programming an Altair, ELF or Junior computerette back in 1979!" The system is now ready to play the melody. Try it by touching the switch on the TX, or pressing S3 for RX local control.

The source code for the project is waiting for all Elektor members at [3]. It's written in assembly language!
Advanced users will sniff at the primitive DIP switch block and have an ARM Cortex 44+ up and running to compose a melody, copyright it, stream it on the web, Twitter around, and program the lot into the humble PIC.
Please post your best 127-note melodies on the project page at Elektor labs [4]. The author is also available there to techtalk with you.

(140256)

Component Lists

Receiver Board

Resistors
Carbon film, 5%, 0.25W
R1 = 47kΩ
R2 = 1kΩ
R3 = 3.3kΩ
R4 = 100kΩ
R5,R6,R9 = 10kΩ
R7 = 10Ω
R8 = 470Ω
R10 = 100Ω
R19 = 0Ω

Capacitors
C9,C10 = 22pF 50V, C0G/NP0, 0.1" pitch
C6 = 1µF 50V, X7R, 0.2" pitch
C3,C7,C8 = 100nF 50 V, X7R, 0.2" pitch
C5 = 10nF 50V, X7R, 0.1" pitch
C2 = 220nF, 50V, X7R, 0.2" pitch
C4 = 10µF, 50V, 2mm pitch, 5x11mm
C1 = 1000µF, 50V, 7.5mm pitch, 16x26mm

Semiconductors
LED1 = LED, red, 3mm
X1 = 4MHz quartz crystal, C_L=18pF
T1 = BC327
T2 = BC337
T3 = BD139
IC1 = MC7805
IC2 = QAM-RX2-433
IC3 = HT12D
IC4 = 4093
IC5 = PIC16F873A, programmed, Elektor Store # 140256-41

Miscellaneous
K1 = DC barrel jack, 1.95mm pin, 12V, 3A (center pin GND!)
K4 = 2-way PCB screw terminal block, 0.2"
K2 = 10-way (2x5) boxheader
K8 = 16-pin pinheader, 2 rows, straight
S2 = SPDT slide switch, PCB edge mount
S3 = pushbutton, tactile feedback, 6x6 mm
Hammond Type 1591 ABS enclosure, 122.45 x 95.9 x 35.25 mm
ANT1 = wire, 17cm
Miniature loudspeaker, 36mm diam., 1 watt
DIP-14 IC socket,
DIP-18 IC socket
Narrow-DIP-28 IC socket

Transmitter Board

Resistors
Carbon film, 5%, 0.25W
R20 = 2.2MΩ
R21,R22 = 10kΩ
R23 = 220Ω
R24 = 1MΩ

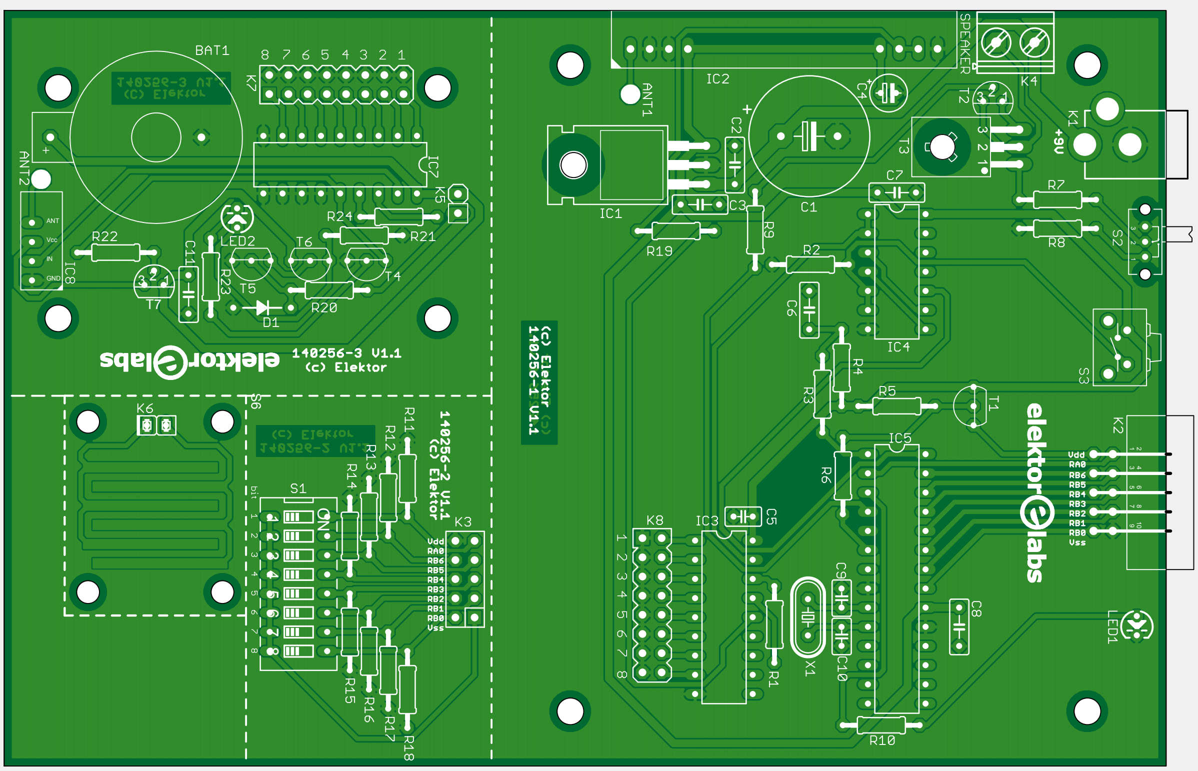
Capacitors
C11 = 100nF, 50V, X7R, 0.2" pitch

Semiconductors
LED2 = LED, red, 3mm
T4,T5 = BC337
T6, T7 = BC327
IC7 = HT12E
IC8 = QAM-TX1-433
D1 = 1N4148

Miscellaneous
K5 = 2-pin pinheader
K6 = 2-pin socket strip
K8 = 16-pin pinheader, dual row, straight
G1 = PCB holder for CR2032 coin cell
Enclosure: Hammond Type 1591XXLBK
BAT1 = CR2032 battery
S6 = discrete switch or touchswitch board
ANT2 = straight wire, 17cm
DIP-18 IC socket

Programmer Board
K3 = 10-way (2x5) IDC receptacle
R11-R18 = 1kΩ
S1 = 8-way DIP switch

Figure 4. The composite printed circuit board for the P-W-M Doorbell project has four sections: receiver, transmitter, melody programming tool, and touchswitch. Each board should be cut out individually with a Junior or hacksaw.



Active DI Box

With floating balanced output

By Harry Zuijderduijn (Netherlands)

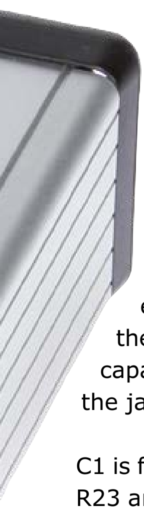
This circuit adapts an unbalanced audio signal source (such as a guitar pickup) to an amplifier or mixing desk with a balanced input. It also has a rotary switch that allows the input signal attenuation or gain to be set to several different levels.

Musicians use DI boxes to connect an instrument with a high-impedance or unbalanced signal output, such as an acoustic guitar, to a mixing desk with balanced inputs. The abbreviation 'DI' probably stands for 'direct injection', although some sources claim that it stands for 'direct input" or 'direct insert'.

There are two types of DI box: passive and active. A passive DI box uses an audio transformer to convert the unbalanced signal into a balanced signal. However, good audio transformers are rather expensive. With low-cost audio transformers, core saturation can occur at low frequencies and high frequencies are often attenuated. In an active DI box the conversion is performed electronically. This also makes it fairly easy to add gain adjustment.

You may be wondering why conversion from unbalanced to balanced is necessary. Mixer boards are generally located a good distance away from the stage, so the signal cables are long and tend to pick up interference. For that reason, mixer desks usually have balanced inputs with XLR connectors or stereo jacks. Of course, balanced signal cables also pick up interference, but the interference signal has the same phase on both signal leads. At the input of the mixing board, one of the signals is inverted and then added to the other signal, which yields a gain of 6 dB. The interference signal on the one lead is also inverted and therefore has the opposite phase as the interference signal on the other lead, so the interference is cancelled out when the signals are added. This results in very high interference suppression.

De acronym DI probably stands for Direct Injection, although Some Say the original meaning is Direct Input or Direct Insert



There are also other applications where a DI box can come in handy. For example, if you want to connect a low-cost DJ mixer with Cinch outputs to a professional PA amplifier, you can use a DI box to make the connection (note that you will need two of them for stereo).

The circuit

The input connectors consist of two 6.3-mm (0.25") jacks, which are wired together (see **Figure 1**). This makes it easy to daisy-chain the input signal to other equipment. K2 is the actual input connector, and K1 is the daisy-chain connector. Resistor R22 ensures that input capacitor C1 is tied to ground (through a spring contact in the jack) when the input is open.

C1 is followed by an attenuator network consisting of R1, R12, R23 and R24. Attenuation levels of 0, –20 and –40 dB can be selected with section B of switch S1.
Three gain levels can also be selected with section A of the switch: +6, +12 and +18 dB. This is done by switching the resistors that control the gain of U2a.
The highest attenuation can be used for purposes such as tapping off a signal from the speaker output of a guitar amplifier, so that you can get the effects from the amplifier that you cannot get with a direct connection to the guitar. If you do not need this switch, it can be replaced by two wire links at the locations marked in the schematic diagram.

There is also a low-pass filter at the input, consisting of R1 and C4. It is dimensioned to attenuate undesirable high frequencies above approximately 30 kHz, taking into account the high source impedance (50 kΩ) of an acoustic guitar pickup. The buffer/amplifier stage built around U2a is followed by a second opamp that inverts the signal, so that a pair of opposite-phase signals are available to drive the balanced output stage built around U1a and U1b. The output stage has dual negative feedback with crossed feedback paths. U1a has negative feedback from its own output via R7 and R6 as well as feedback from the output of U1b, which is out of phase with the positive input of U1a, via R8 and R25. U1b has the same arrangement.

The balanced signals pass through capacitors C3 and C5 to header J1, which is connected to a three-pin XLR socket or

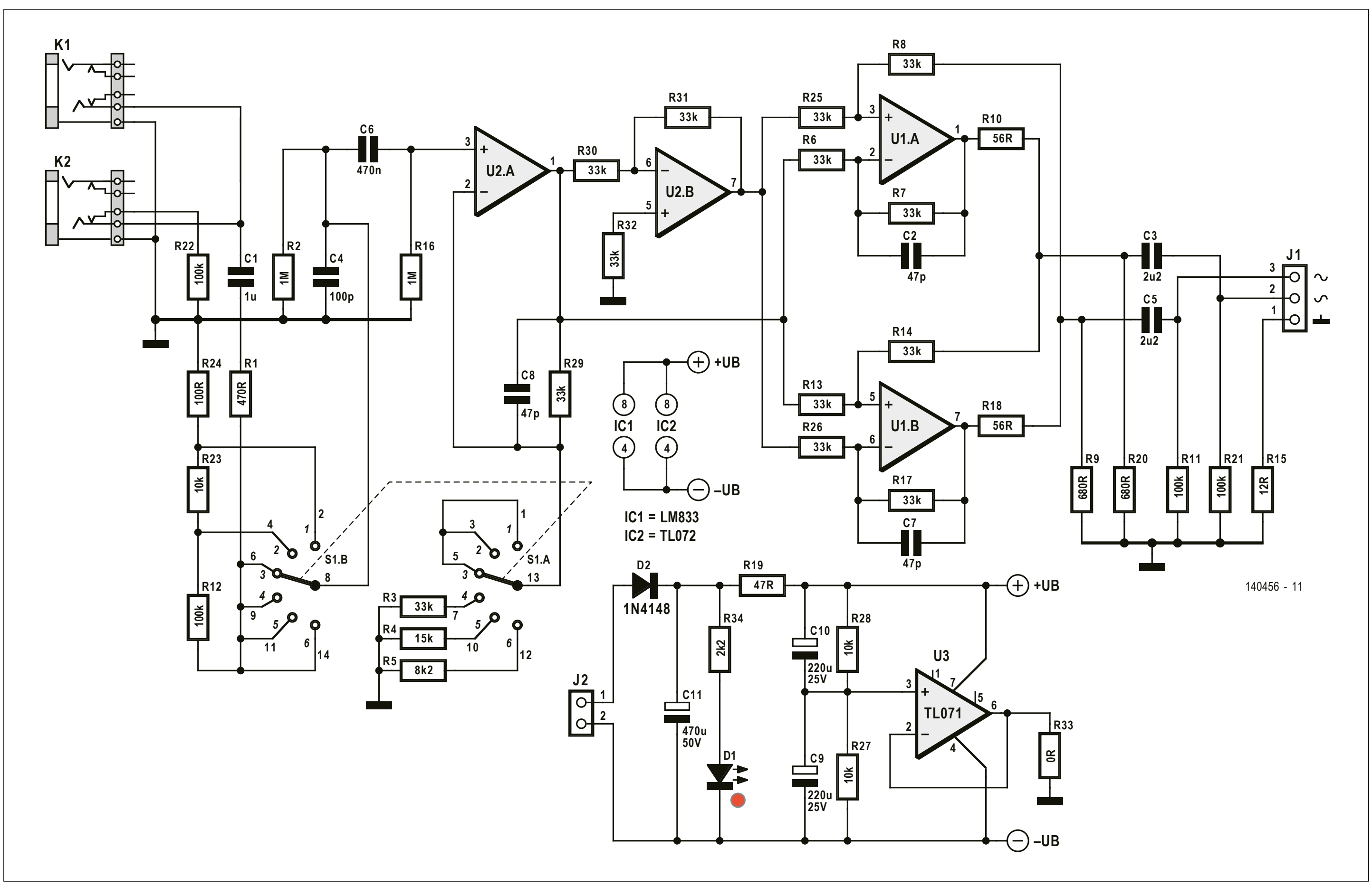


Figure 1. The unbalanced audio input signal is converted to a balanced output signal by a few opamps.

a 6.3-mm stereo jack. If one of the outputs is connected to ground while the unit is in use (accidentally or intentionally), the signal is still available from the other output. This is the same as what would happen with an output transformer.

A 12 Ω resistor (R15) is located between the ground terminal of the connector and circuit ground. This ‘soft’ ground connection is primarily intended to avoid interference problems with laptop power adapters. Instead of R15, you could use a switch to provide the option of grounded or floating (ground lift) operation. A ground connection is not always necessary for proper operation with a balanced input. For example, the author connected a dual version of the DI box to his laptop with the ground disconnected and did not have any interference on the signal connection to the mixer desk. With the ground connected, there was interference from the laptop power adapter.

Power supply

The circuit is powered by an AC adapter with a regulated output voltage. The design can tolerate a fairly wide range of supply voltages and works from 9 V_{DC} to as much as 30 V_{DC}. The advantage of using a higher supply voltage is that the maximum signal amplitude (headroom) is larger. The author uses an adapter with an output voltage of 24 V, which is enough to avoid compression even at high peak signal levels. Diode D2 provides polarity protection to avoid problems if the power adapter is connected the wrong way. Any interference that may come from the power adapter is filtered out by R19, C9 and C10. The circuitry around U3 generates a virtual ground at half the supply voltage. Resistor R33 is included in the circuit because the schematic drawing program used by the author does not allow the output of an opamp to be tied to ground. This means that R33 is actually a wire link on the PCB, which can be used as a ground reference point when you make measurements on the circuit.

Onderdelenlijst

Resistors

R1 = 470Ω
R2,R16 = 1MΩ
R3,R6,R7,R8,R13,R14,R17,R25,R26,R30,R31,R32 = 33kΩ
R4 = 15kΩ
R5 = 8.2kΩ
R9,R20 = 680Ω
R10,R18 = 56Ω
R11,R12,R21,R22 = 100kΩ
R15 = 12Ω
R19 = 47Ω
R23,R27,R28 = 10kΩ
R24 = 100Ω
R29 = 33kΩ
R33 = 0Ω (wire link)
R34 = 2.2kΩ

Capacitors

C1 = 1µF MKT or MKS
C2,C8 = 47pF
C3,C5 = 2.2µF MKT or MKS
C4 = 100pF
C6 = 470nF
C7 = 47pF
C9,C10 = 220µF 25V
C11 = 470µF 50V

Semiconductors

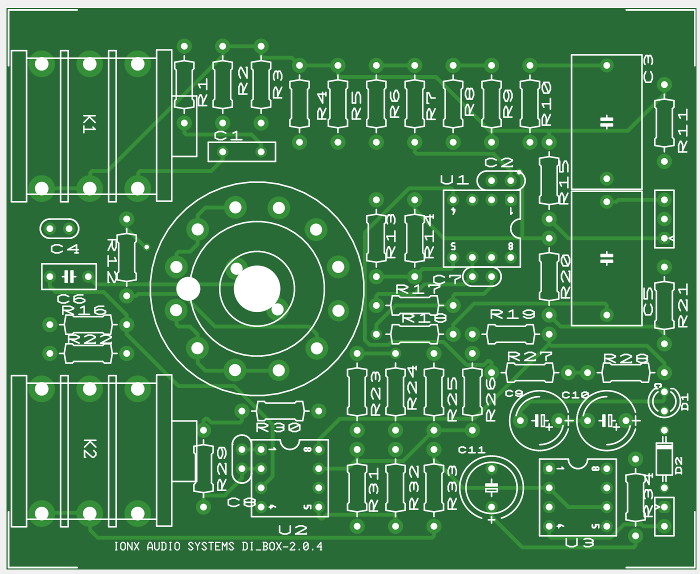
D1 = 1N4148
D2 = LED, red, 3mm
U1 = LM833
U2 = TL072
U3 = TL071

Miscellaneous

S1 = 6-position 2-pole rotary switch, PCB mount
K1,K2 = 6.3mm stereo jack connector, PCB mount
J1 = 3-pin pinheader
J2 = 2-pin pinheader
Supply connector, panel mount, 2.1mm center pin
3-pin XLR socket, panel mount
Case, Hammond type 1455K1202

Switch S1 settings

Position	Effect
1	-40 dB
2	-20 dB
3	0 dB
4	+6 dB
5	+12 dB
6	+18 dB



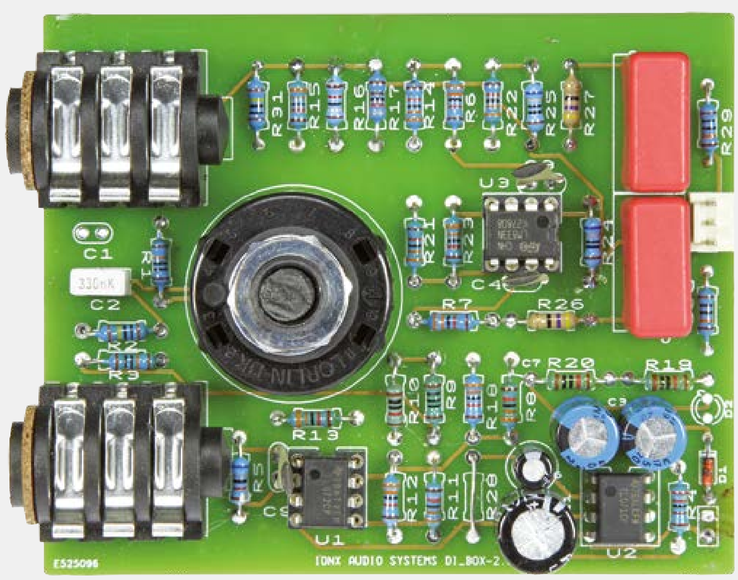


Figure 2. The PCB layout designed by the author for the DI box. (note: small differences w.r.t. prototype)

Construction

There aren't any unusual or tiny (SMD) components in this design, so construction should not present any problems even if you do not have advanced soldering skills. The PCB designed for this circuit by the author (**Figure 2**) is dimensioned to fit in a Hammond 1455K1202 box, which is readily available. The layout is available in PDF format at [1].

The chassis-mount shells of the jacks are fitted on the board and secured to the front or rear lid of the box by screws inserted in holes drilled in the lid. Three openings have to be made in the opposite lid for the XLR socket, the power connector and the power indicator LED. Use short lengths of cable to connect the XLR socket and the power connector to pin headers J1 (shielded) and J2. The lids are plastic, so you have to make a separate connection between pin 1 of the XLR chassis-mount connector and the metal box. In the prototype this connection is made by a piece of stranded wire that is clamped between the lid and the metal box when the lid is closed.

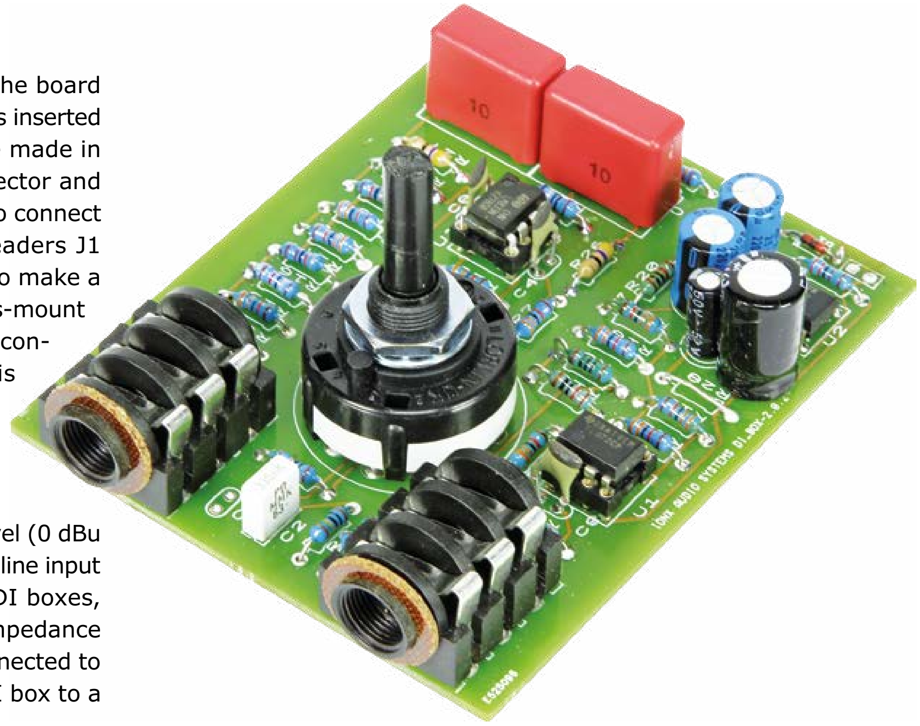
Use

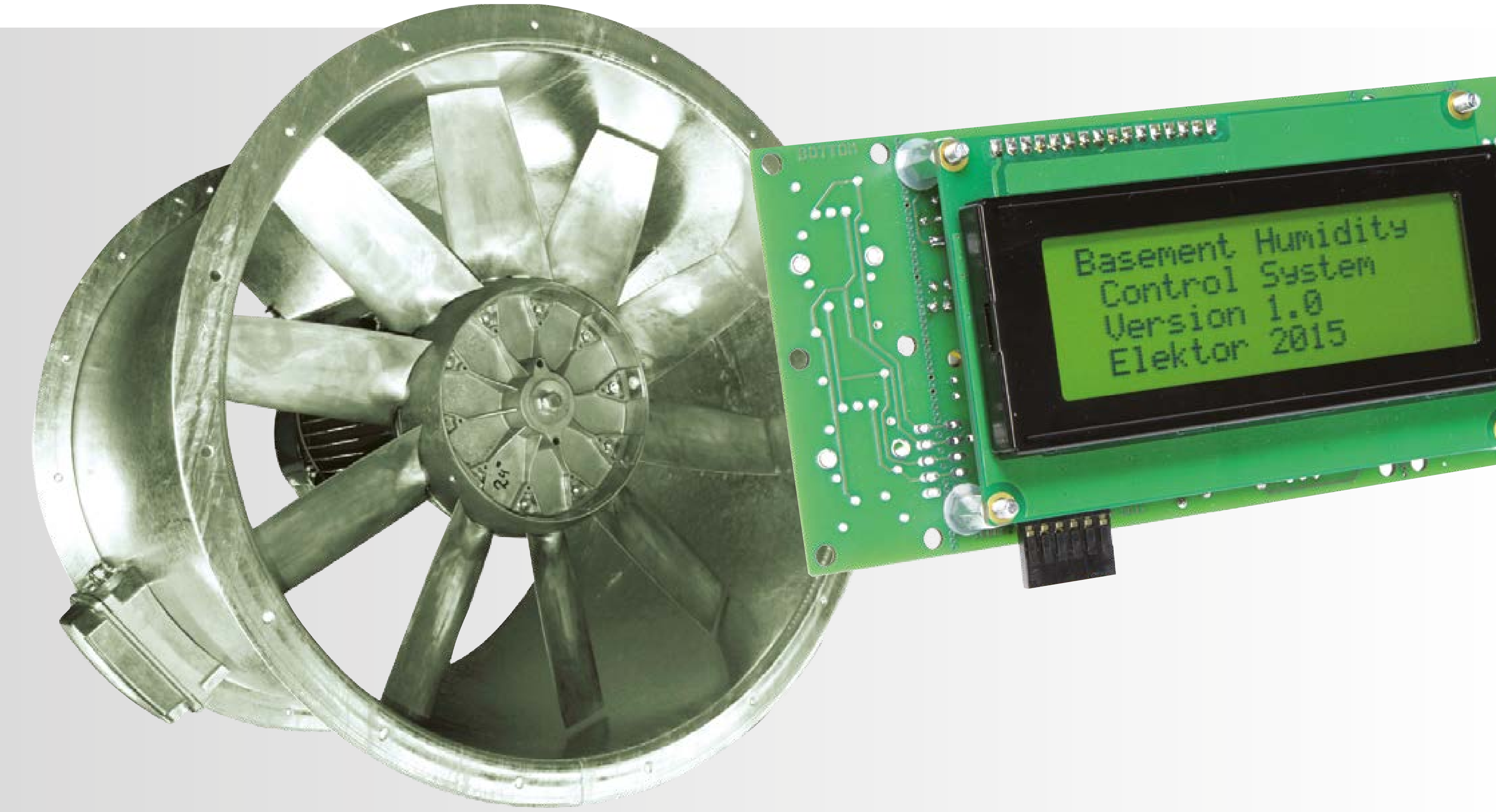
This DI box is designed to work with signals at line level (0 dBu or 0.775 V). It is therefore important to connect it to a line input on the mixing desk. This is different from passive DI boxes, which reduce the signal level by 20 dB due to the impedance conversion transformer and therefore have to be connected to a mic input. Although it is possible to connect this DI box to a mic input by selecting 20 dB attenuation with the attenuation/gain knob, that degrades the signal to noise ratio by 20 dB.

(140456)

Web Link

[1] www.elektor-magazine.com/140456





Basement Ventilation System (1)

Humidity control with Platino

By **Luc Lemmens** (Elektor Labs)   Based on an idea from **Danny Winkler**

Moisture is often a problem in basements. This is frequently a consequence of condensation due to the low temperatures in basements, which are usually unheated. It frequently leads to unpleasant odors from fungal growth and damage to stored items from corrosion and wood rot, all of which are undesirable. Good ventilation can go a long way toward eliminating or reducing these problems, but good ventilation involves more than simply opening a window or a door. The system described here takes a more intelligent approach and can even help you convert a basement into a pleasant living space.

Like attics, basements are traditionally not designed for comfort. They were originally intended to be used as storage spaces, where you only go when you need to fetch something or have to store something and then leave right away. These spaces were rarely if ever heated, and they were cleaned and tidied up once a year (at most), as part of spring cleaning. In short, they were not places where you would sit around and relax. Wine cellars and beer cellars may be a happy exception, but that's a completely different story.

Attics are warmed a bit by heat from the rooms beneath them, but basements are usually cold. This property of basements was (and still is) good for keeping foods and beverages somewhat cool and therefore prolonging their shelf life, but there's less need for this nowadays when most households have a refrigerator and a freezer. In any case, high humidity is certainly something you don't want to have in a modern house. That's because the cold air in the basement causes condensation when warm, humid air flows into the basement and gets chilled. The cooler air cannot hold as much moisture, so water condenses on cold surfaces such as walls and floors. Particularly in the summer, you can often see water droplets hanging from water pipes in a basement, which does not mean that the pipes or fittings are leaky.

Of course, moisture management in

an underground space is a fairly complex subject and other factors, such as groundwater seepage through cracks or moisture penetration through basement walls, can also play a role. That's not what we're interested in here, and problems of that sort should be left to building experts and other specialists. The ventilation system described here is intended to be used in basements that have condensation problems but are otherwise okay. This situation is frequently found in relatively old basements, but in new buildings complying with modern building regulations it actually shouldn't occur. Fungal growth as a result of high humidity can be very bad for your health, and rusting metal or rotting wood, as well as damage to other things you would rather keep intact, are equally undesirable. Stale basement air is in any case unpleasant. Ventilation can prevent a lot of these problems, but you have to do it properly.

Features

- Two ChipCap2 sensors for measuring relative humidity (2% accuracy) and temperature (within ±0.6°C)
- 4-line by 20-character LCD
- Supply voltage: 8–12 V DC
- Software runs on ATmega328 microcontroller (Arduino compatible)
- Rotary encoder with integrated pushbutton for user interface
- Controls electric window opener
- Controls fan or dehumidifier

Electronics to the rescue

On our .Labs project website [1], an Elektor community member in Germany presented a microcontroller-based system designed to maintain the humidity in his basement at an acceptable level. It has a pair of sensors that measure the relative humidity and temperature indoors and outdoors. Based on the measured values, a basement window is opened or closed automatically and a fan is switched on and off. Using this system, he found that he could do a very good job of solving his humidity problem.

We found the design interesting enough to pick up on it and elaborate it for publication in the magazine. Unfortunately there was no PCB layout, and instead of designing yet another microcontroller board with an LCD module, buttons and a power supply, we decided to use our stalwart Platino board as the basis for this project. We rewrote the software in the Arduino environment, so most members of the Elektor community should not find it difficult to modify the control characteristics or other functions of the system if necessary or desirable. More circuitry for controlling the fan and operating the window, as well as proper connections for the sensors, will be published in a future issue.

The original design used two rather expensive humidity and temperature sensors, which we replaced by lower-cost ChipCap2 sensors. In this year's January & February edition we described our CC2-eBOB module, a small plug-in board that makes these SMD sensors a lot easier to use for people without advanced soldering skills (see **Figure 1**).

The instrumentation

This is about as simple as it gets: one sensor measures the humidity and temperature in the basement, and the other one measures the humidity and temperature outdoors. However, these sensors measure relative humidity, and that isn't especially useful for this application. The temperature is also important because warm air can hold more water vapor than cold air. This means that ventilation on a hot summer day has an adverse effect, because warm, moist air flows into the basement and the moisture condenses in this cool space. Fortunately, the absolute humidity can be determined from the air temperature and the relative humidity. The absolute humidity is a measure of the moisture concentration in the air (in grams per cubic meter), which is exactly what we need. As with other material flow processes, moisture will move from a space with high concentration to a space with low concentration. If you know the absolute air humidity indoors and outdoors, you know when it makes sense to ventilate the basement.

The control algorithm

This is also fairly simple in theory: you should ventilate if the moisture concentration is higher indoors than outdoors, and otherwise you shouldn't. However, things aren't that simple in practice. For example, it isn't a good idea to leave a basement window open for hours on end when it's well below freezing outside. Aside from the heat loss from the house, there is a risk of freezing things such as water pipes running through the basement. It's also possible that one of the sensors may stop working for some reason, and in that case you don't want to have the system run wild. Another consideration is that you don't want to have the relative humidity in the basement get too low, even if that's fairly unlikely. Putting all these considerations together, we came up with the following list of conditions for opening and closing the basement window and/or starting or stopping the fan:

1. If one of the sensors fails, close the window and switch off the fan.
2. If the indoor temperature is below freezing, close the window and switch off the fan.
3. If the relative humidity indoors is too low, open the window and switch off the fan.
4. If the absolute humidity indoors is higher than outdoors, open the window and switch on the fan.
5. If the absolute humidity indoors is lower than outdoors, close the window and switch off the fan.

The above conditions are listed in order of priority (1 = highest priority). The window position and the fan state are adjusted every ten minutes based on

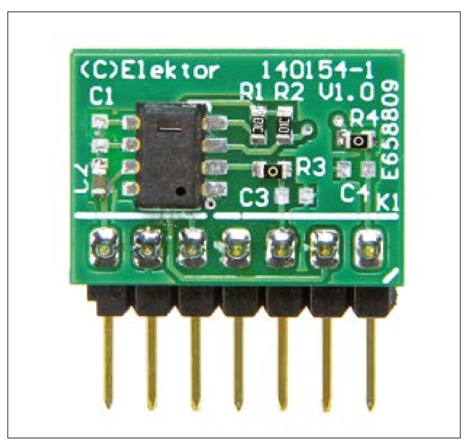


Figure 1. This adaptor board makes it a lot easier to work with the ChipCap2 sensor.

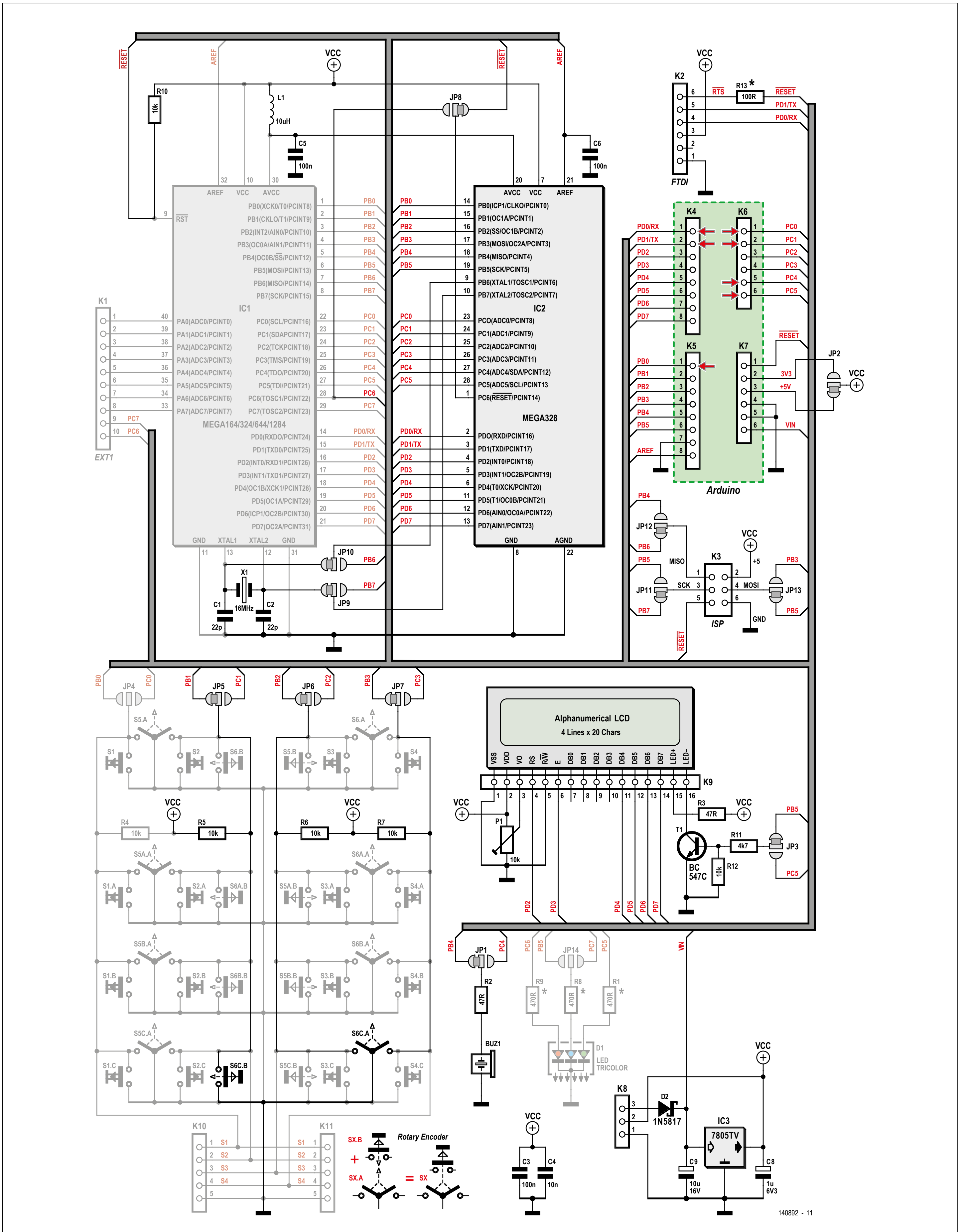


Figure 2. Schematic diagram of the Platino board. The components and jumpers not required for this project are greyed out.

the most recent readings, so the system does not need any hysteresis. This approach avoids the noise nuisance and unnecessary energy consumption that would result from constant cycling of the system.

Required hardware

The system is built around the Elektor Platino board describe in the October 2011 issue of the magazine. This board is fitted with an ATmega328 containing an Arduino boot loader, so it can be used in the Arduino development environment the same way as the Arduino Uno. The complete schematic of the board is shown in **Figure 2**, with the components not required for this project greyed out. The schematic also shows which solder bridges (jumpers) on the board have to be open or closed. The components list for this article indicates which components you need for the project.

Of course, you shouldn't forget the humidity and temperature sensors. As previously mentioned, we use a pair of our ready-made CC2-eBoB modules for this purpose (item number 140154-91). The ChipCap2 sensors communicate over the I²C bus and have a factory-programmed address of 0x28. One of them therefore has to be assigned a different address so that they can both be connected to the same bus. This is handled by a bit of software, but you have to assemble the Platino board before you can use it.

Construction

Assembling the Platino board is fairly easy if you follow the schematic diagram and the components list. It's a good idea to start by configuring solder bridges JP1 to JP14. To make things easier, the correct settings of the solder bridges are listed in **Table 1**. JP11 to JP13 are only necessary if you want to use an AVR ISP adapter to program the ATmega328. If you want to use the Arduino boot loader (as we did), connect a USB to TTL serial interface converter to K2 on the Platino board. A 5-V FTDI cable is perfectly adequate, and it can provide the supply voltage to the entire control board during testing or further software development.

The only function of inductor L1 is to decouple the analog supply voltage for the ATmega328. Since this application does not use the A/D converter, you can simply replace L1 by a wire bridges if you wish.

That brings us to the two small sensor boards. As previously mentioned, the I²C address of one of the two sensors must be changed, and that is done by a sketch written specifically for this purpose.

Start by connecting one CC2-eBoB board to the Platino board. The I²C bus leads must be connected to pins 5 and 6 of K6, which provide the SDA and SCL signals. *For the time being, connect the V_{DD} lead to the digital output pin PB0 (pin 1 of K5) on the Platino board.* That's because the I²C address can only be changed during a short time window after power is applied to the ChipCap2 sensor, and this arrangement allows the sketch to switch the supply voltage on and off by itself. Connect the GND lead to pin 4 of connector K7.

Open the Arduino development environment on your computer and load the sketch *CC2A_set_I2C_address.ino*, which is available at [2] as a free download.

Two constants are defined in this sketch:

1. *CURRENT_I2C_ADDRESS = 0x28* (the default value, which is used for the indoor sensor in this application)
2. *NEW_I2C_ADDRESS = 0x22* (for the outdoor sensor)

Other addresses in the range from 0x00 to 0x7F are also possible, but then the addresses in the source code of the ventilation controller, where they are defined as the constants *Inside_sensor* and *Outside_sensor*, must be changed to match.

Connect the Platino board to the PC via an FTDI cable (if you have not already done so), and then compile *CC2A_set_I2C_address.ino* and download it to the board. It will then look after changing the I²C address.

After that you can connect both sensor boards to the same I²C bus. Provisionally connect their supply leads to pin 3 (+5 V)

Table 1: Solder bridges and jumpers on the Platino board

Jumpers:			Jumpers:	
JP1	PB4 (buzzer)		JP8	Reset
JP2	5V		JP9	XTAL2
JP3	PB5 (backlight)		JP10	XTAL1
JP4	-		JP11	PB7
JP5	PB1		JP12	PB6
JP6	PB2		JP13	PB5
JP7	PB3		JP14	-

Relative and absolute humidity

This system uses absolute air humidity as the control variable. However, most humidity sensors measure relative humidity and temperature and are therefore not suitable for the direct indication of absolute humidity. Although several factors are involved in determining the absolute humidity, the following formulas give a reasonably good approximation:

$$\text{SVP [hPa]} = F/100 \times 6.1078 \times 10^{((a \times T)/(b + T))}$$

$$\text{RH [\%]} = 100 \times VP/SVP$$

$$\text{AH [g/m}^3\text{]} = 216.7 \times VP/(273.15 + T) = (2.167 \times RH \times SVP)/(273.15 + T)$$

where:

SVP =saturated vapor pressure
VP = actual vapor pressure
RH = relative humidity
AH = absolute humidity
a = 7.5 (constant)
b = 237.5 (constant)
T = measured temperature in °C

Adapted from: www.wettermail.de/wetter/feuchte.html#f1

Table 2: Connections and pin designations on the control board

Arduino	ATmega328	Function
Window operation:		
Digital 0	PD0	High during opening
Digital 1	PD1	High during closing
Analog 0	PC0	Limit switch; Low at end position (window open)
Analog1	PC1	Limit switch; Low at end position (window closed)
LCD:		
Digital 2	PD2	RS
Digital 3	PD3	Enable
Digital 4	PD4	Data D4
Digital 5	PD5	Data D5
Digital 6	PD6	Data D6
Digital 7	PD7	Data D7
Analog 5	PB5	Backlight on/off
Rotary encoder:		
Digital 9	PB1	Pushbutton
Digital 10	PB2	Phase A
Digital 11	PB3	Phase B
Fan:		
Digital 8	PB0	On/Off
Buzzer:		
Analog 4	PB4	On/Off

and pin 4 (GND) of K7, and connect the I²C lines SDA and SCL to pins 5 and 6 (respectively) of K6. A separate connector for these four lines will be provided on the extension board for this project, which will be described in an upcoming issue. Of course, the fan and the window opener are still missing, and their control and construction will also be described in a later article. If you're impatient, have a look at **Figure 3** for the wiring and **Table 2** to see which pins of the Platino board are assigned to which functions. As you can see, one output of the Platino is reserved for controlling the fan (on or off). For the window opener there are two control lines for the motor (opening and closing) and two pins for limit switches (optional) that detect when the window is at either end of its travel.

Platino Component List

Resistors
Default: 5% .25W
R2,R3 = 47Ω
R5,R6,R7,R10,R12 = 10kΩ
R11 = 4.7kΩ
R5 = 3.3kΩ
P1 = 10kΩ trimpot

Capacitors
C1,C2 = 22pF ceramic
C3,C5,C6 = 100nF
C4 = 10nF
C8 = 1µF 50V radial, 2.5mm pitch
C9 = 10µF 50V radial, 2.5mm pitch

Inductor
L1 = 10µH or wire jumper

Semiconductors:
D2 = 1N5817 Schottky diode
T1 = BC547
IC2 = ATmega328, programmed with Arduino-bootloader, Elektor Store # 100892-41)
IC3 = 7805

Miscellaneous
LCD1 = 4x20 character alphanumerical LCD with backlight (e.g. Midas MC4200B6W-SPR
X1 = 16MHz quartz crystal
S6 = rotary encoder with pushbutton, Alps EC12E2424407
BUZ1 = 12mm AC buzzer ABT-410-RC
Platino board, Elektor Store # 100892-1
2 pcs CC2-eBoB, Elektor Store # 140154-91

Firmware

Now you're completely ready for the initial test of your ventilation system. Download the sketch *_140432_Control_Unit.ino* from the Elektor website [3], load it into the Arduino environment, and then download the sketch to the Platino board. After you power up the system, it first closes the window and switches off the fan. The states of the window and fan are adjusted as soon as the first measurements have been made.

You may be wondering why the rotary encoder is included in the circuit. To make the system as general-purpose as possible and allow it to be adapted to specific situations or user wishes, several configuration settings are available. They are stored in the internal EEPROM of the ATmega328 so that they are retained when the supply voltage is switched off. When the unit is started up the first time, these memory locations are empty and default values for the various settings are configured automatically. This involves the following parameters:

- The threshold temperature for frost protection. The default value is 3°C, adjustable from 3°C to 10°C.
- The lower limit for relative humidity in the basement. When the relative humidity drops below this limit, ventilation is activated to raise the humidity level in the basement. The default value is 30%, adjustable over the range of 20% to 80%.

These parameters can be modified during normal operation by pressing the rotary encoder knob. When the parameters menu appears, the window is closed and the fan is switched off. At this point automatic control stops so that the system can be switched off with the window closed. This can also be handy in other situations, such as when you want to clean the window. The ventilation system becomes active again after the parameters have been adjusted.

There are also settings for the electric window opener. Quite a few different types are available, and there are bound to be users who want to design the electromechanical components themselves. However, that's the subject of a follow-up project. After installation, the window opener settings can only be modified by pressing and holding the encoder knob while switching on the system.

Once all the settings have been configured, operation of the system is entirely automatic. The relative humidity and temperature indoors and outdoors are continuously measured and shown on the display, along with the calculated moisture concentrations.
In the unlikely event that one of the two sensors stops responding, which means that a valid status cannot be retrieved over the I²C bus, the system reacts immediately and indicates the faulty sensor on the LCD. The buzzer also sounds to draw attention to this fault condition.

When the sensors are working properly, the window position and the fan state are adjusted as necessary every ten minutes. The display also shows which condition has been fulfilled and the corresponding status of the window or the fan.
Several LCD screenshots are shown in **Figure 4**.

Final remarks

In this article we have only discussed the control system; in an upcoming article we will devote attention to controlling the fan and in particular the automatic window mechanism. The latter certainly deserves extra attention due to the associated mechanical design and construction. We will also present a PCB that allows everything to be neatly connected to the Platino board.

Although this ventilation system turns out to work very well in practice, we certainly do not claim that it is the perfect solution for transforming every moist and stuffy basement into a dry and fresh-smelling little paradise. The original description on our .Labs website [1] clearly shows that the developer is still working hard to optimize and extend this project. You should regard it as a handy way to improve the humidity situation in a basically well-built and properly maintained basement. If you have leaks or other relatively serious problems, always call on expert assistance — not only for the sake of your health and the preservation of the things you have stored in your basement, but also to avoid structural damage to your house.

(140432)

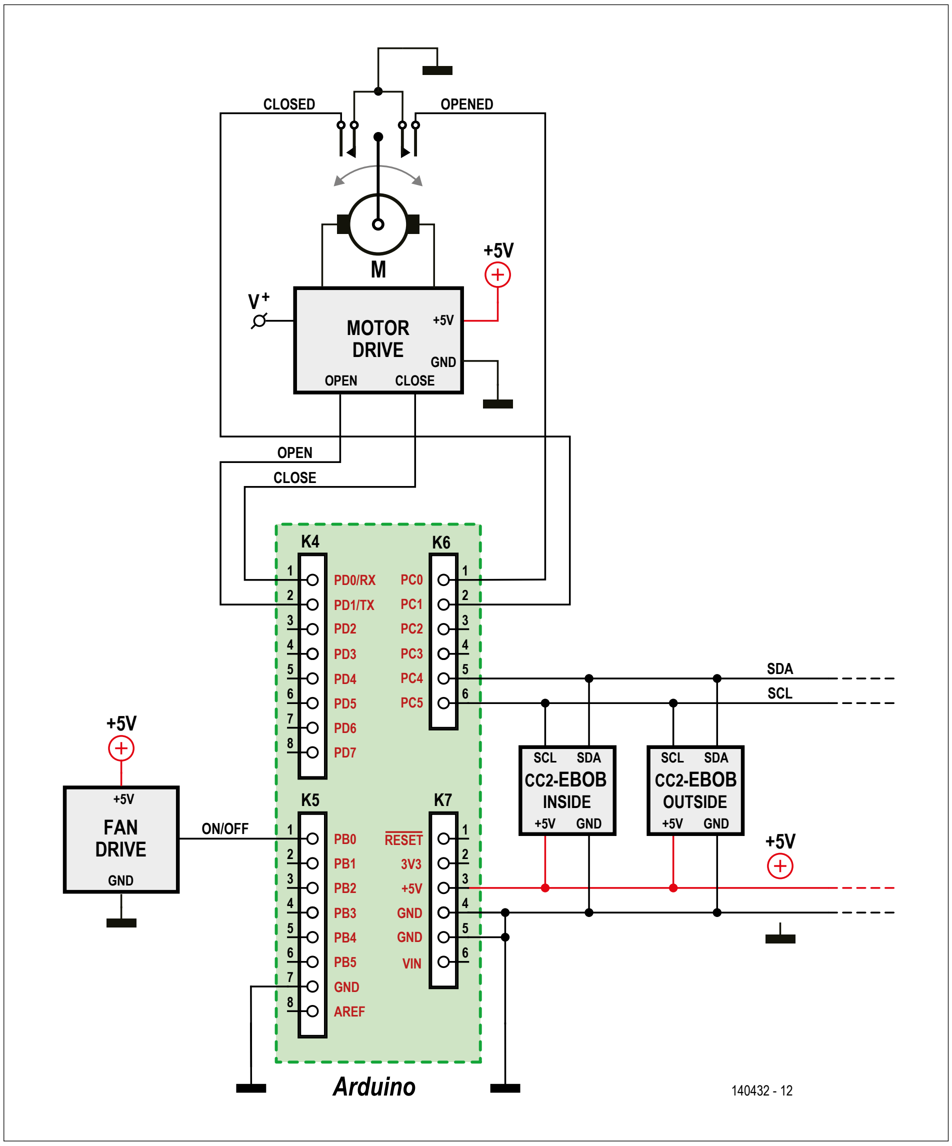


Figure 3. Peripheral wiring of the ventilation system.

Web Links

[1] Original design: www.elektor-labs.com/project/feuchteges-teuerte-kellerl-ftung-humidity-basement-ventilation-140154.13770.html

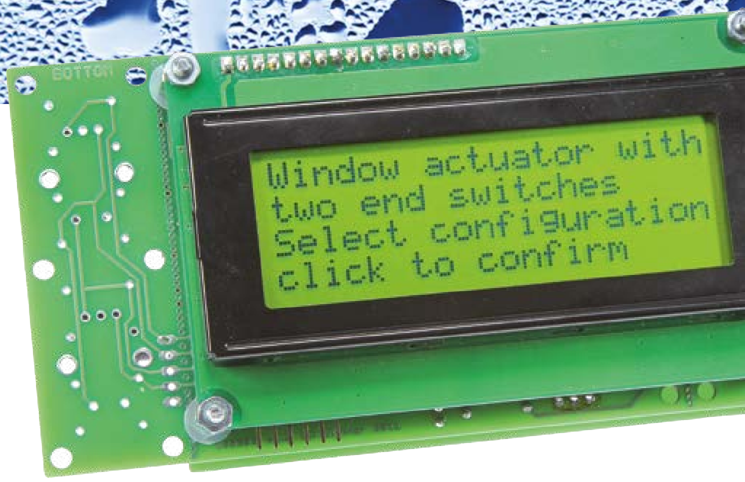
[2] CC2-eBoB project page: www.elektor-magazine.com/140154

[3] Project page for this article: www.elektor-magazine.com/140432

Figure 4. Some of the reports shown on the LCD module of the ventilation system.

Basement Ventilation System (2)

Installing and connecting the window actuator and fan



By **Luc Lemmens** (Elektor Labs)
Based on an idea from **Danny Winkler**

In a companion installment we describe a control unit for managing the humidity level in a basement or other damp space with a ventilation system. The relative humidity and temperature indoors and outdoors are measured using ChipCap2 sensors. Based on the measured values, a microcontroller on a Platino board decides when the fan should be switched on and off and when the window should be opened or closed. The Platino board also handles overall ventilation control. In this article we describe how to put all of the parts of the system together.

In the first article of this pair we explained how the humidity level in a basement can be kept within acceptable limits with the help of a bit of electronics. Moisture problems are fairly common in unheated spaces such as basements due to condensation, particularly in the summer when warm, moist air flows in from outside. That's because warm air can hold more moisture than the relatively cold air in the basement.
The layout and initial use of the Platino board and the software are described in detail in the first article.

The relative humidity and temperature indoors and outdoors are measured using a pair of sensors. From this data the absolute air humidity in both areas is calculated to determine which has the higher moisture concentration. This is necessary in order to decide whether ventilation will help remove moisture from the basement or have exactly the opposite effect. The system also takes into account that the basement window should not kept be open too long when it's freezing cold outside. For this reason, it keeps an eye on the temperature in the basement to avoid freezing the water pipes.
The microcontroller on the Platino board then determines when a window should be opened or closed and when a fan or ventilation unit should be switched on to blow even more damp basement air outside. A number of control signals are necessary for this: the fan must be switched on and off and the window must be operated by a suitable mechanism. The latter requires a good deal of extra attention, as you will see later own. In short, we need to switch some motors (among other things). Although our AVR board has a lot of built-in functions and is very versatile, it does not have the hardware or the space on the board necessary for switching large currents. We therefore designed a general-purpose extension board for this popular microcontroller board, which is also handy for other projects or developing your own applications.

Platino extension board

Our original intention was to design a PCB specifically tailored to the basement ventilation system. However, once we got to the hardware for controlling the window actuator we realized that — as

is so often the case — there are so many applications that need the same basic hardware that it would be better to create a general-purpose design. Furthermore, most versions of the ventilation system only need a handful of components and a couple of connectors for the cables, so it's hardly worth the effort of designing a separate PCB for each version.
A simple piece of prototyping board would have also been a reasonable option, but we decided to design a specific 'prototyping shield' for the Platino as shown in **Figure 1**. It provides room for a standard DC power jack on the board, which is connected directly to the supply voltage lines for the Platino on connector K8. We even remembered to drill a hole in the board so you can access the LCD contrast trimpot on the Platino board without removing the extension board. It goes without saying that the mounting holes are exactly aligned to the mounting holes of the microcontroller board, making it easy to join the boards together in a tidy manner.

The supply voltage VIN (from an AC adapter or other power source), as well as VCC and GND, are available on a set of clearly marked solder pads. There is also space reserved for three PCB-mount terminal blocks with 0.2 inch pitch, which usually have pins that are too thick for mounting on standard prototyping board. The connections between the two boards are provided by wire-wrap sockets with 13-mm pins, which are fitted on the Platino board. The extension board is fitted with pin headers that mate with these sockets. For best results, you should mechanically join the two boards together with standoffs before soldering the sockets in place on the Platino board, to make sure the wire-wrap socket are at exactly the right height.
Incidentally, this is an excellent way to connect extra hardware to the Platino board in a neat and tidy manner. Now let's see what we have to do to make all this suitable for the ventilation system.

Sensor enclosures and placement

As previously mentioned, we use a pair of CC2-eBOB sensor modules to measure air humidity and temperature. It's a good idea to fit these modules in small enclosures, both for the sake of appearance and for protection. Of course, the enclosures must have adequate ventilation openings to enable the sensors to maintain good contact with the ambient air. Particularly with the outdoor sensor,

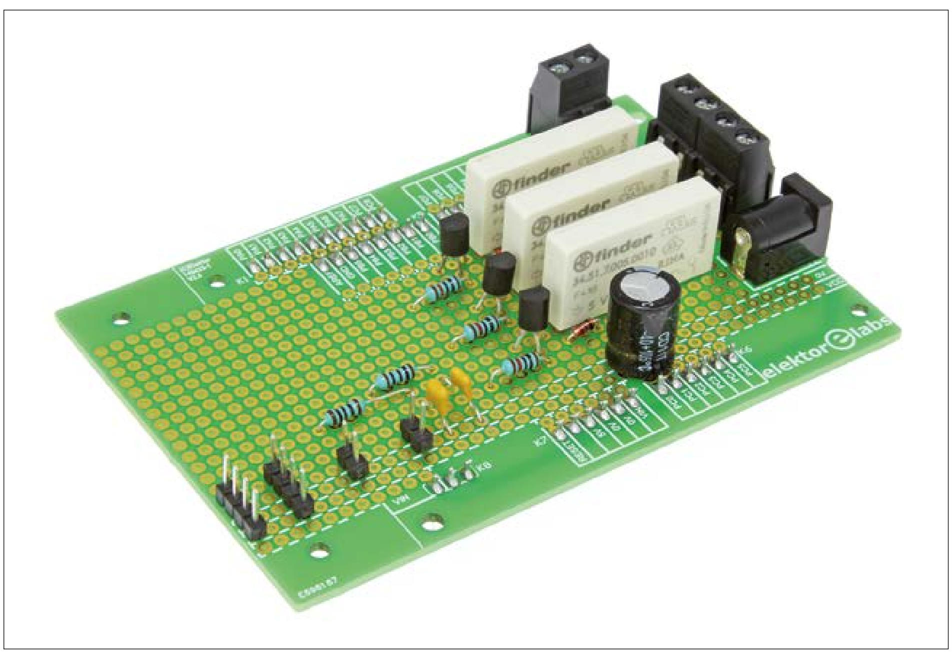


Figure 1. This extension board makes it easy to add circuitry to the Platino board.

which is exposed to wind and weather, you should make sure that everything is sound and sturdy.

Proper operation of the ventilation system depends on correct placement of the ChipCap2 sensors. Putting the indoor sensor close to the basement window is obviously not a good idea. Instead, you should put it in a location where the air in the basement is naturally still. It is difficult to give specific advice about this because it depends on the building and how the basement is partitioned, so you will probably need to try several locations to find the best arrangement. If under reasonable weather conditions the absolute air humidity in the basement does not drop within a few weeks (or even a few days) after the ventilation control system is put into service, or actually rises, relocating the indoor sensor may improve the situation.

The outdoor sensor should likewise not be located in the air stream from the fan or the basement window, or where it is exposed to direct sunlight. For lots of tips and suggestions on how to make reliable temperature and humidity measurements, you can consult professional or amateur meteorology websites for information about building weather stations.

As previously mentioned, the sensors are connected over an I²C bus. You should keep the connecting cables reasonably short to avoid problems with communication errors. LAN cable (Cat5, for example) or shielded cable is a good choice for connecting the sensors, and cable lengths up to several meters should not be a problem considering that the firmware works with a bus speed of 100 kHz (the standard setting in the Arduino Wire library). If there are any communication problems with either of the sensors, the firmware will indicate this immediately. If that happens, you should consider shortening the cables if possible. In the worst case you may need to use an I²C repeater.

Connectors K13 and K14 in **Figure 2** are the terminal points for the cables. If both sensors are connected to the same cable, only one of these connectors has to be used.

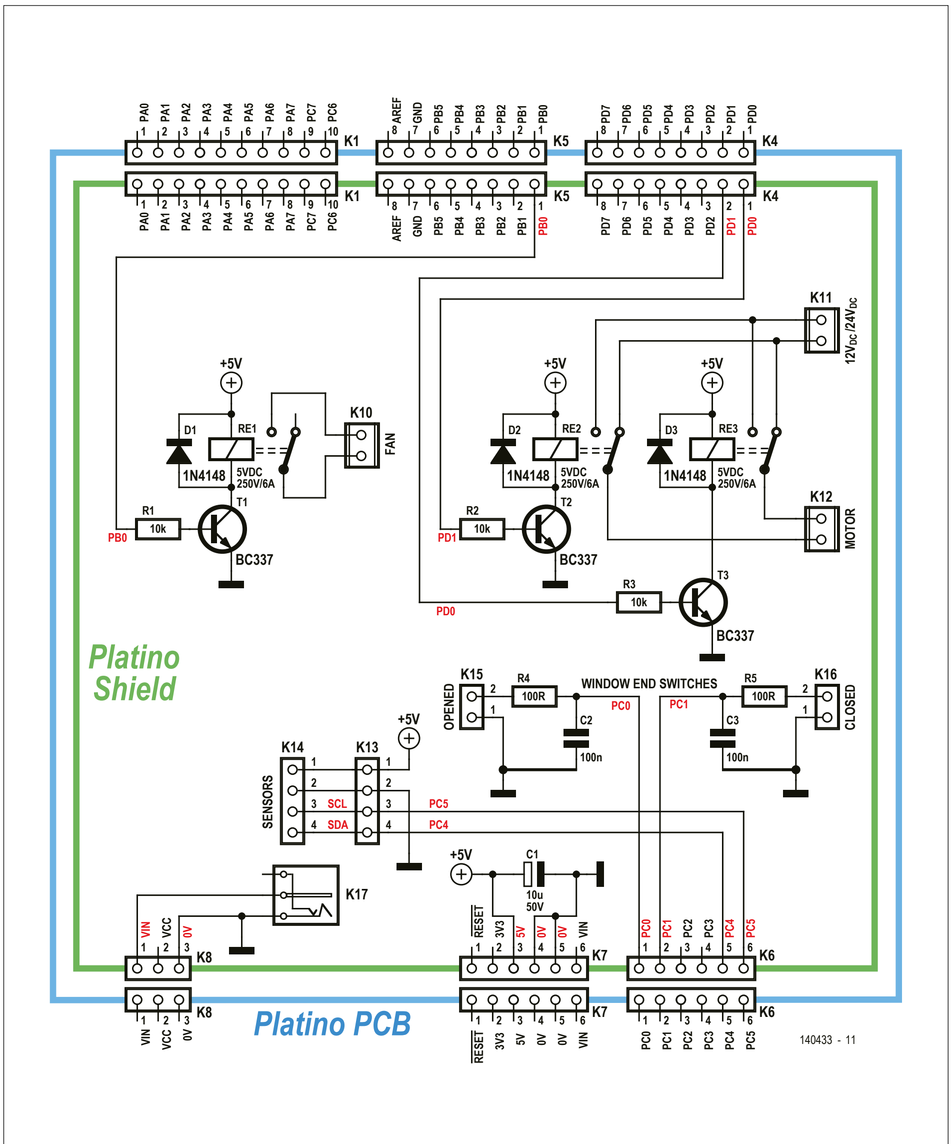


Figure 2. Schematic of the circuit on the expansion board.

Fan

The fan or ventilation unit is basically the simplest part of the ventilation system in terms of control. It is either on or off, so the logic level on pin PB0 of the microcontroller can simply be fed through a transistor to operate relay Re1, which switches the supply voltage for the fan. However, there's something that has to be considered in the layout of the shield: the fan is powered from the AC line (mains) voltage. That means you have to pay careful attention to safety and ensure that the insulation distance (creepage distance) is at least 3 mm. You must therefore remove the solder pads next to the relay switch terminals on the PCB (see **Figure 5**). To do this, first drill out the through-hole plating and then heat the pads with a soldering iron. After this it is usually fairly easy to detach the pads. Another option is to use a small router.

The relay pins are relatively thick, so you will have to enlarge the mounting holes slightly with a drill. The footprint also differs slightly from the standard 100-mil lead pitch, so you will have to enlarge one of the holes for the relay coil to make it fit.

Use well insulated wire of suitable diameter for the leads between the relay and the terminal block. It's also a good idea to secure the wires with a bit of hot-melt glue.

An alternative is to fit a suitable relay in the fan itself, but you can only do that if there is enough room in the housing. Freewheel diode D1 should be soldered directly across the relay coil, but the transistor and its base resistor can remain on the PCB. The main advantage of this approach is that you only have a low voltage from the ventilation controller to the fan, so a thin cable is sufficient for the connection.

Window actuator

This is the most difficult part of the entire system. That's not because electronic control is complicated, but instead because this involves some mechanical construction and it's not possible to provide a ready-made solution for every basement window. Fortunately, basement windows are usually not especially large or heavy (actually we're talking about cellar windows here), so the mechanical construction does not have to be heavy duty. Of course, there are a few things you need to consider, including the fact that a window that opens outward can be a great temptation for "uninvited guests" in your home. You should therefore pay careful attention to making your construction burglar-proof. For good ventilation, the window needs to open far enough and close properly. That means you have to figure out some way to detect the end positions of the window. We opted for two microswitches that close when the window is open or closed, respectively. You could also do this by sensing the rise in motor current when the motor stalls, but that is only possible if your mechanical structure can handle the stress (you don't want to pull the sash out of the frame, after all).
Ready-made window actuators are also available, mostly with a motor that runs on 12 V or 24 VDC. That's not a particularly low-cost solution, but at least you get a nicely finished package. A small satellite dish actuator (used to aim dish antennas) is also a good choice, and they are often equipped with limit switches or stop automatically when they reach the end of travel. Another possibility is an electric window mechanism from a car. If you prefer to roll your own, you probably have plenty of ideas about how to do it or you can surf the web to look for inspiration from other designers.
As you can see in Figure 2, an external power source for the motor is connected to K11. This can be an AC adapter with a sufficient power rating – a laptop power supply module is more than enough for most motors. The motor leads are connected to K12. Pay careful attention to the polarity of these leads, since the controller always closes the window when it is switched on. Check that the motor runs in the right direction before mounting the actuator on the window, and swap the motor leads if necessary. The limit switches for the window are connected to K15 and K16. You should use shielded cable for these connections. The resistor-capacitor networks (R4/C2

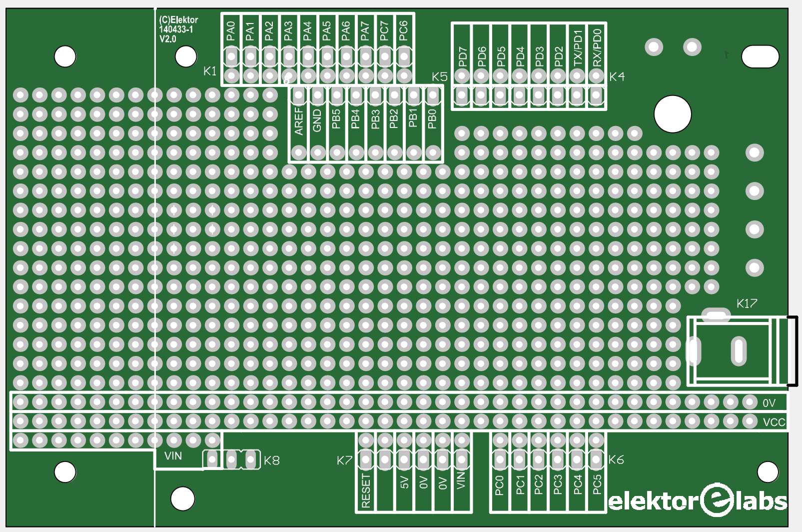
Component List Platino Extension Board

Resistors
5%, 0.25W
R1,R2,R3 = 10kΩ
R4,R5 = 100Ω

Capacitors
C1 = 100µF 50V radial
C2,C3 = 100nF

Semiconductors
D1,D2,D3 = 1N4148
T1,T2,T3 = BC337

Miscellaneous
RE1,RE2,RE3 = relay, 5V SPDT, 6A (e.g. Finder 34.51.7.005.0010)
K1 (on Platino) = 10-pin wirewrap socket, 0.1" pitch, pin length 12.9mm
K4,K5 (on Platino) = 8-pin wirewrap socket, 0.1" pitch, pin length 12.9mm
K6,K7 (on Platino) = 6-pin wirewrap socket, 0.1" pitch, pin length 12.9mm
K8 (on Platino) = 3-pin wirewrap socket, 0.1" pitch, pin length 12.9mm
(wirewrap sockets on Platino: Farnell/Newark #1023031, 20-pin version)
K1 (on shield) = 10-pin pinheader
K4,K5 (on shield) = 8-pin pinheader
K6,K7 (on shield) = 6-pin pinheader
K8 (on shield) = 3-pin pinheader
(pinheaders on shield: Farnell no. 1022218, 32-pin version)
K10,K11,K12 = 2-way PCB screw terminal block, 0.2" pitch
K13,K14 = 4-pin pinheader, 0.1" pitch
K15,K16 = 2-pin pinheader, 0.1" pitch
K17 = DC adapter socket, PCB mount
PCB no. 140433-1



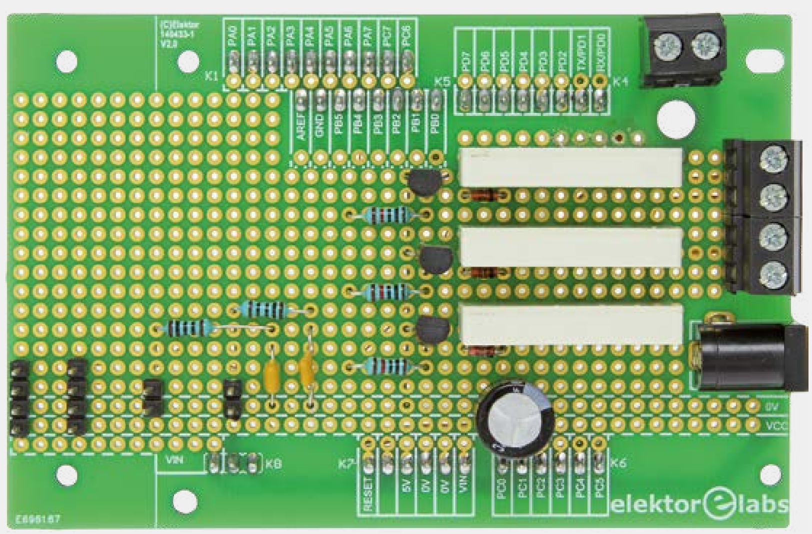


Figure 3. This PCB is handy for building all sorts of peripheral circuitry for Platino projects.

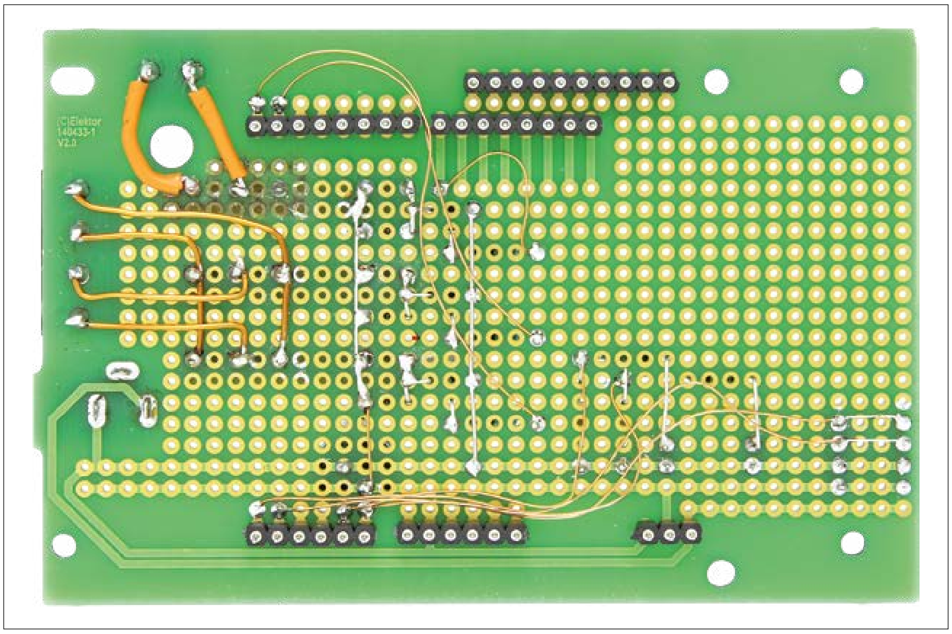


Figure 4. The wiring on the rear of the board.

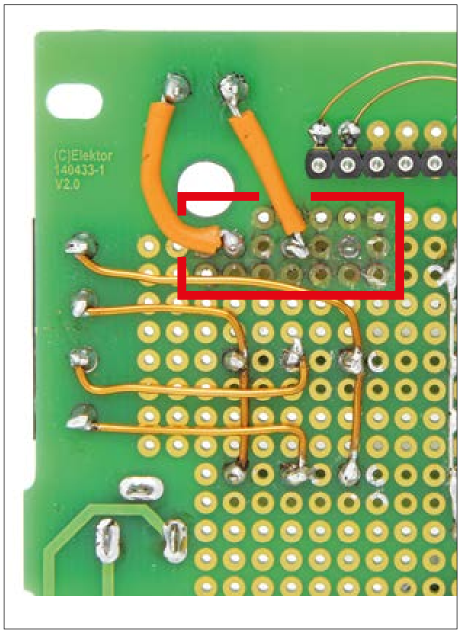


Figure 5. Several solder pads in the AC line voltage area of the PCB must be removed in the interest of safety.

and R5/C3) suppress noise picked up by the switch cables.

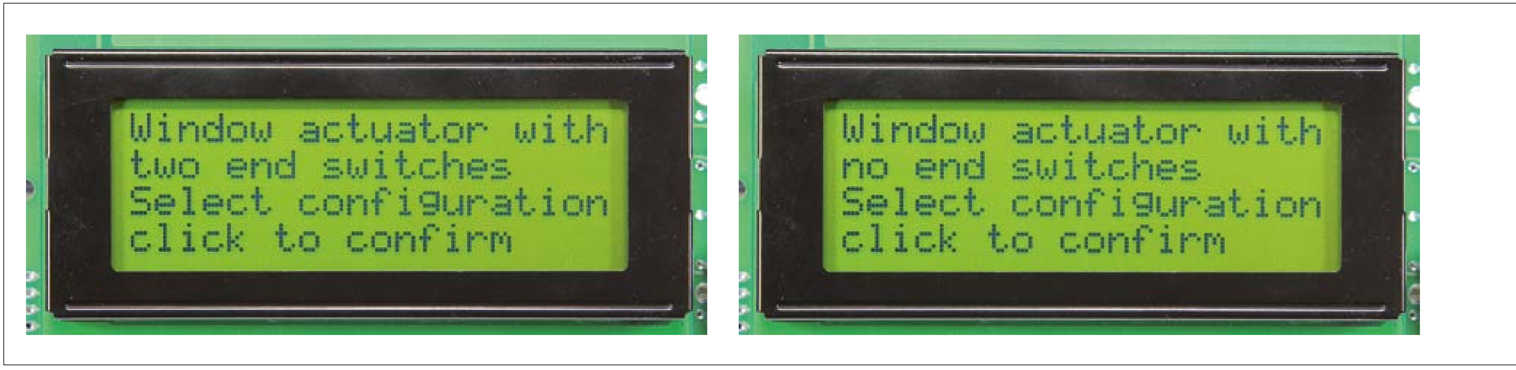
The software looks after everything

We have done our best to make the software suitable for every possible type of window actuator. What that actually means is the various options for detecting the end positions of the window. If this is done with limit switches or current sensing, there is always feedback to the microcontroller to tell it when the motor should be switched off. However, some actuators have built-in travel limits, and in that case the Platino has to reset the control signal for the motor after a suitable timeout, without any form of feedback. This timeout is also used for window actuators with limit switches to provide extra protection if the switches somehow fail. In both cases the time interval must be a bit longer than the time the actuator actually needs to operate the window.

The first time you power up the system, you automatically land in a menu for configuring the window actuator parameters (see **Figures 6a** and **6b**). These settings are stored in the internal EEPROM of the microcontroller, so that they are retained when the system is powered down. Later you can also access this menu by holding the knob of the rotary encoder pressed while switching on the supply voltage for the Platino board.

First you can enter which end positions

Figure 6. Some configuration options for the window actuator.



of the window are fed back: both, one of the two or none. You can then enter the timeout values for motor control (range 1 to 40 seconds), with separate settings for the opening time and closing time (**Figures 6c** and **6d**). These times may differ depending on the mechanical design, and some actuators run at different speeds in the two directions. After this the program goes directly into ventilation control mode, and you can see how effective it is at bringing the humidity in the basement to an acceptable level and keeping it there.

Concluding remarks

This article fills in the gaps left at the end of the first article last month. The window actuator is probably the most difficult part of this project, but it is not an insurmountable obstacle. A certain amount of fine tuning will probably be necessary after installation. You may have to move the sensors to more favorable locations, or you might want to adapt the software to your specific wishes. The source code for the program is available for free download [3], and it can easily be edited or enhanced in the Arduino development environment. We cannot guarantee that this ventilation system will make every basement nice and dry, but it will certainly improve the situation.

(140433)

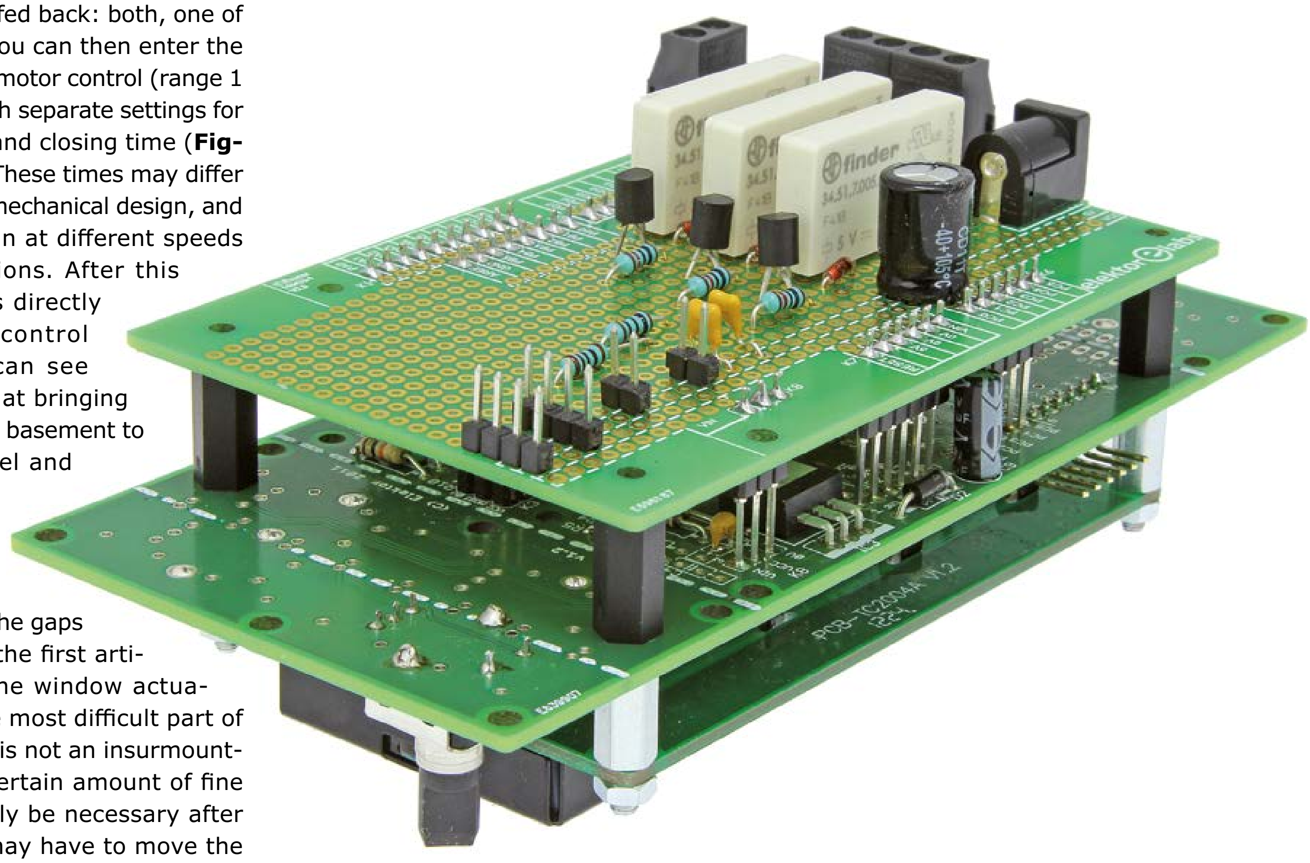


Figure 7. Mounting the extension board on the rear of the Platino board.

Web Links

[1] Original design: www.elektor-labs.com/project/feuchteges-teuerte-kellerl-ftung-humidity-basement-ventilation-140154.13770.html

[2] CC2-eBoB project page: www.elektor-magazine.com/140154

[3] Project page for this article: www.elektor-magazine.com/140432

Medication Alarm Clock

With 9 different alarm times

By Martien Schot (Netherlands)

Many people have to take medication at regular times. To be continually reminded that it is time again to take the medication, you really need an alarm clock which goes off at different times. This alarm clock was developed specifically for this purpose, and in contrast with ordinary alarms clocks this one has a large number of adjustable alarm times.

An ordinary alarm clock has usually only one, or at most two, adjustable alarm times. But when you have to take different types of medication then such an alarm clock is of little use. In addition, an ordinary alarm clock does not allow for alarms only certain days of the week. All of this is solved by this compact clock that can be easily built by anyone who has a little soldering experience. It consists of a base circuit board fitted with an LCD or OLED display. It is operated using a few push buttons. The correct time is supplied by a DCF receiver.

Circuit

The heart of the circuit comprises a microcontroller of the type ATmega328P (IC2 in **Figure 1**). This drives an LCD or an OLED display with two lines of 16 characters each. This display is a standard type with the connections along the top edge and measures 80 x 36 mm. The contrast adjustment is achieved with P1 (only when using an LCD). Only a few additional components are required around the microcontroller, a 16 MHz crystal provides a stable clock. In order to generate an easily audible alarm sound a separate audio section is present, which is built around an LM386

(IC3). This too requires only a few external components. The sound volume level is set with P2. A small 8-Ω loudspeaker is connected to X2.
As has been mentioned already, a DCF receiver module takes care of the time information. This is connected to connector X3.
The alarm clock is operated using two pushbuttons, S1 and S2. S1 (Mode) serves to set the alarm times and S2 (AL/Nav) has three functions: setting (navigating) of the alarm times, the silencing of an active alarm and on/off-control of the impulse monitoring (refer to the software description later on).
With the Mode button (S1) the alarm setting menu appears on the bottom line. The active field flashes (initially that is the alarm number). The following positions are the alarm day, the time and the direction indicator, see **Figure 2** and the user manual.
The circuit is powered from a line power adapter, which needs to supply a regulated voltage of 5 V. These switching power supplies are dirt cheap these days and very energy efficient. You do have to take notice of the correct polarity. When the polarity is the wrong way around, D1 will protect the circuit (for a short while).

Software

The software is developed with the aid of an Arduino UNO and the accompanying Arduino-IDE. **Table 1** shows the relationship between the Arduino I/O pins and the ATmega I/O pins.
The outline of the software includes the following parts, which are called from within the '*void loop*':

- *InsideClock* This contains the internal clock and the comparison with the alarm times.
- *DCF_receiver* The DCF receiver part with decoding.
- *ModeButton* The function *Alarmedit* is invoked via a contact-bounce suppressor and shows alarm line #1 on the bottom line of the LCD/OLED.
- *NavigateButton* Is called when *Alarmedit* is active. This routine also contains a contact-bounce suppressor.
- *AlarmHandler* Is only called when an alarm is active.
- In *InsideClock* worden de seconden gegenereerd door de functie *millis()* en worden aan de hand daarvan de minuten, uren en weekdagen uitgerekend.

Technical Characteristics

- Up to 9 different alarm times
- The alarm day can be a specific weekday are all days
- The resolution of the alarm time is 15 minutes
- Alarm times are stored in EEPROM
- When an alarm signal is not acted upon the alarm signal repeats every 3 minutes for the following 15 minutes
- Internal clock with synchronization using a DCF receiver
- Monitor function of the received DCF signal
- Double-checking of the correctness of the received time
- Easy to build, no surface mount components

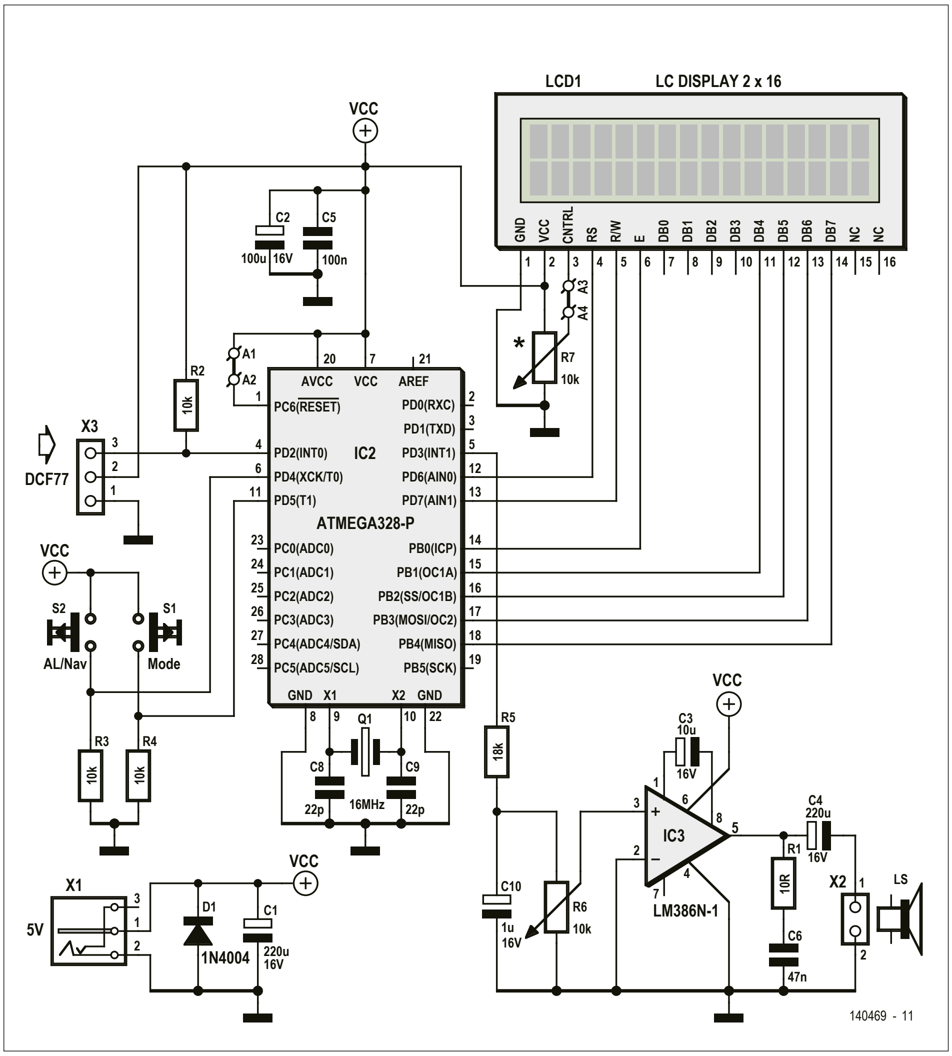


Figure 1. The circuit consists mainly of an ATmega microcontroller, a two-line display and a small audio amplifier.

Listing 1.

```
int TotalMinutes = (WkDay*1440)+(Hour*60)+Minute;
if (TotalMinutes==ExpectTotal)CorrectReceive=true;
else CorrectReceive=false;
if (TotalMinutes==11519) ExpectTotal=1440;
else ExpectTotal= ++TotalMinutes ;
```

Figure 2. After pressing S1 the alarm-setting menu appears on the bottom line of the display.

User manual

After switching the clock on wait until it has received the correct time.
Setting the alarm times:

- Push Mode button S1. The following text appears on the bottom line:
  `#1 XXXXXX 00:00|`

- `#1` Alarm number
  Select from alarm 1 through 9 (1 flashes now).

  `XXXXXX` Alarm day (now switched OFF).
  Here you can choose between AllDay, Monday, Tueday, Wenday, Thuday, Friday, Satday, Sunday, XXXXXX

  `00:00` Alarm time
  Setting in 3 steps: tens of hours, units of hours and minutes in units of quarter of an hour.

  | Vertical line: indicates that the flashing alarm number can be changed with alarm button S2.

- Push Mode button S1 again. The vertical line now changes to a horizontal line. Alarm-button S2 can now be used to jump the the next field `XXXXXX`. This will now flash. By pressing S1 again the line will change to a vertical line and you can select for example AllDay. This means that the alarm will go off every day.
- In this way you can also set the time `00:00`.
- At the end of the line, the line _ is reached, this will now flash. If the entered line is correct you can store it in EEPROM by pressing mode-button S1. This will now exit the alarm-edit mode.
- If another line has to be entered or if the entered line is incorrect then instead of mode-button S1, press alarm-button S2, and the same line can be entered again or a subsequent alarm line can be set.
- The procedure for changing an existing alarm is the same as described above.

Functions of the buttons

The Mode button S1 is only used when changing the alarms.
The Alarm/Navigation button S2 has three functions

- Navigation as described above.
- Silencing of an active alarm.
- Switching-on of the monitor function

The seconds are generated in *InsideClock* by the function *millis()* and based on these the minutes, hours, and days of the week are calculated.
In *DCF_receiver* the DCF77 signal is first transformed to zeros and ones depending on the impulse lengths. Subsequently, these are converted to useful decimal values including checking of the parity. When a complete time cycle has been received correctly, the expected time of the next cycle is calculated.
This calculated time comprises the sum of the minutes that have already passed since the beginning of the week (Monday 00:00 hour), refer to calculation in **Listing 1**. Normally only one is added to this sum for the following minute, but from Sunday 23:59 to Monday 00:00 the ‘minutes sum’ goes from 11519 to 1440 (that is one entire day). If the previously calculated sum corresponds to the received minutes sum, then the received time is correct and the internal clock will be synchronized. Now also the bottom line will be displayed. This also shows that the reception was successful. **Figure 3** shows the two lines with the normal time and date.

In *ModeButton*:

- *Alarmedit* is switched **on**
- During *Alarmedit*, position 15 alternates between a horizontal and vertical line.
- *Alarmedit* is switched **off** at position 15.

In *NavigateButton, Right* (horizontal line) or *UP* (vertical line) is called. The *Right* routine is shown in **Listing 2**.
The alarm data is stored in a two-dimensional array ‘Alarms[9][4]’. This contains 9 records with 4 fields each. In the program these are record 0 through 8 and fields 0 through 3.
Each record contains one complete set of alarm data:

- Field 0 = Alarm day; 0 = Each day, 1 through 7 = Monday through Sunday, 8 = XXXXXX.
- Field 1 = Tens of hours; 0, 1, 2.
- Field 2 = Units of hours; 0 through 9.
- Field 3 = Quarters; 0 = 00, 1 = 15, 2 = 30, 3 = 45.

Because the fields are 1, 2 or 6 characters long the standard blink function which flashes only one character at a time was not used, but a routine was written that flashes the correct number of characters simultaneously. See the section *void CharBlink* in **Listing 3**.

Listing 2.

```
void RIGHT()
{++CurPosH;
 lcd.setCursor(CurPosH,1);
 switch (CurPosH)
     {case 0: CurPosH= 1; break;//goto 1=Alarmnumber
      case 2: lcd.setCursor(1,1);lcd.print(Ala_Num+1);
              CharacStr=AlarmDaySt[Alarms[Ala_Num][0]];CurPosH=3;break; // DOW
      case 4: lcd.setCursor(3,1);lcd.print(AlarmDaySt[Alarms[Ala_Num][0]]);
              CharacStr=String(Alarms[Ala_Num][1]);CurPosH=10; break; // ten minutes
      case 11:lcd.setCursor(10,1);lcd.print(String(Alarms[Ala_Num][1]));
              CharacStr=String(Alarms[Ala_Num][2]);CurPosH=11; break; //minutes
      case 12:lcd.setCursor(11,1);lcd.print(String(Alarms[Ala_Num][2]));
              CharacStr=QuarterStr[Alarms[Ala_Num][3]];CurPosH=13; break; //Quarters
      case 14:lcd.setCursor(13,1);lcd.print(QuarterStr[Alarms[Ala_Num][3]]);
              CharacStr="-";CurPosH=15; break; //write data
      case 16:CurPosH= 1;CharacStr=String(Ala_Num+1); break;//goto Alarmnumber again
     }
 lcd.setCursor(CurPosH,1);
}
```

To compare the alarm times with the actual time a special format was invented, which makes this comparison easier to carry out (**Listing 4**). This is a different format than what was used when checking the received time.

The real time (*RealTime*) is equal to the sum of the minutes plus 100 times the hours plus 10000 times the weekdays. The format of the sum of an alarm time (*AlarmSum[i])* is equal to the format of the real time.

The weekdays of the real time can have a value of 1 through 7. The weekdays of the alarm time can have a value of 0 through 8. The real time value will therefore never reach 80000 or more. So when ‘XXXXXX’ (8) is selected in Field(0) that alarm will never become active.

When ‘Each Day’ (0) is selected, the alarm sum will always be smaller than 2360 (24.00 hours). When this is the case, the value of the real weekday is added to the alarm sum so that it can be compared to the real time.

AlarmHandler. This is called when in **Listing 4** *Alarm=true* on the bottom line is executed. At the same time the corresponding alarm is shown on the bottom line. A short melody is played and during the next 15 minutes this little melody is repeated every 3 minutes, until the alarm-stop button S2 is pressed.

The software uses the EEPROM-library from Arduino. This is present in the standard libraries directory of Arduino. When

Figure 3. The normal time and date indication on the display.

Listing 3.

```
void CharBlink()
{if (millis()-LastOne > 400)//  0.4 sec
    {LastOne=millis();
    switch (CharacStr.length())
         {case 6:if(Charac=="      ")Charac=CharacStr;
                 else Charac="      ";// 6 characters days
                 break;
          case 2:if(Charac=="  ")Charac=CharacStr;
                 else Charac="  ";// 2 characters quarters
                 break;
          case 1:if(Charac==" ")Charac=CharacStr;
                 else Charac=" ";// 1 character number, minutes
                 break;          // ten minutes
         }
    lcd.print(Charac);lcd.setCursor(CurPosH,1);
    }
```

Listing 4.

```
if (IntSec == 0 && AlarmEdit == false)
  {RealTime=IntMin+(IntHour*100)+(IntWkDay*10000L);
    for (byte i = 0; i<9; i++)// 9 alarms
      { if(AlarmSum[i]< 2460) CompareSum=AlarmSum[i]+(IntWkDay*10000L);
        else CompareSum=AlarmSum[i];
        if (CompareSum==RealTime){Alarm=true;ParticuAla=i;TriggerTime=RealTime;}
      }
  }
```

the software is started up for the very first time, the EEPROM part will still be empty (FF) and the first 36 bytes of the EEPROM have to be initialized with the 9 alarms. Fields 0 are all initialized with the value ‘8’ so that all alarms are switched off. The remaining fields are all initialized to 0 (see **Listing 5**). All alarms therefore look like alarm 1. (**# 1 X X X X X X 0 0 : 0 0)**

There is also a DCF receiver monitor function. This is active when S2 (AL/Nav) is operated. This fills the display with information about the received impulses. For example: ‘**P13 0=100 mS**’ or ‘**P23**

Component List

Resistors
Default: 0.25W
R1 = 10Ω
R2,R3,R4 = 10kΩ
R5 = 18kΩ
R6,R7 = 10kΩ trimpot, horizontal
A1-A2 = wire jumper
A3-A4 = wire jumper

Capacitors
C1,C4 = 220µF 16V radial
C2 = 100µF 16V radial
C3 = 10µF 16V radial
C5 = 100nF
C6 = 47nF
C7 = 10nF
C8,C9 = 22pF
C10 = 1µF 16V tantalum

Semiconductors
IC2 = ATMEGA328P, programmed
IC3 = LM386
D1 = 1N4004

Miscellaneous
Display = LCD, 2 x16 characters, or OLED display
(e.g. Winstar WEH01602 or EA W162-X3LG)
Q1 = 16MHz quartz crystal
X1 = power supply connector, 2mm center pin
X2 = 2-way PCB terminal block, 0.2” lead pitch
X3 = 3-way PCB terminal block, 0.2” lead pitch
S1,S2 = pushbutton (e.g. Conrad # 700046-89)
DCF77 receiver module (e.g. Conrad # 641138-89)

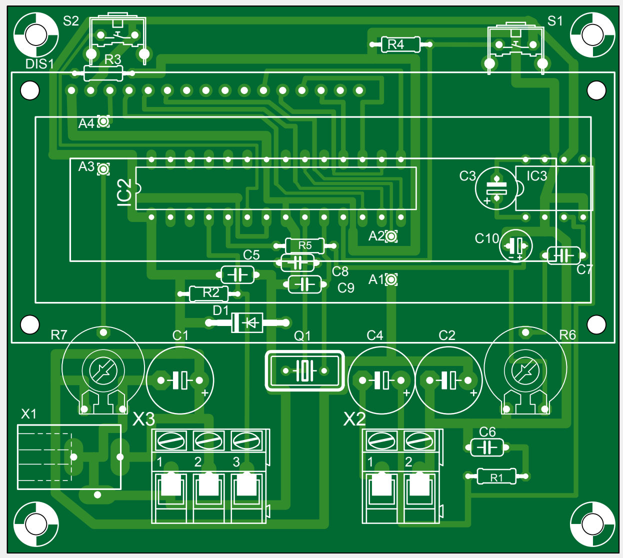


Figure 4. The circuit board for the alarm clock that has been designed by the author. There are two wire links (A1-A2 and A3-A4) on the single-sided circuit board.

1=200 mS'. Here Pxx is the impulse number. This is followed by a 0 or 1 with the received impulse duration in milliseconds. This impulse length can vary and deviate up to 10%. Operating S2 again returns the display to the normal indication.

For driving the display, use is made of the 'LCD_OLED_FourBit' library which can control both an LCD as well as an OLED display. Before you program the ATmega328P using an Arduino, you first have to create a sub-directory *LCD_OLED_FourBit*. In there put:

- LCD_OLED_FourBit.h
- LCD_OLED_FourBit.cpp
- keywords.txt

These files can all be downloaded from the Magazine website [3].
Make a directory 'HelloLcdOledChar' and put 'HelloLcdOledChar.ino' in this directory.

Listing 5.

```
byte Address=0;//EEPROM init 36 bytes
if (EEPROM.read(Address)== 0xFF)//(“empty=FF”)
  {for (byte i = 0; i < 9; i++)
    {for (byte j = 0; j < 4; j++)
      {if (j == 0) EEPROM.write(Address, 8);
       else EEPROM.write(Address, 0);
       ++Address;
      }
    }
  }
```

Table 1. Relationship between the I/O-pins of Arduino and ATmega

ARDUINO	6	7	8	9	10	11	12
ATmega pin	12	13	14	15	16	17	18
ARDUINO DCF77PIN = 2	ATmega pin 4						
ARDUINO SPEAKPIN = 3	ATmega pin 5						
ARDUINO NAVIGBUT = 4	ATmega pin 6						
ARDUINO MODEBUTT = 5	ATmega pin 11						

Construction

Figure 4 shows the circuit board that has been designed by the author. This layout can also be downloaded from the Elektor Magazine website [3]. The circuit board is single-sided and measures 7.2 x 8.1 cm. Fitting the parts is very straightforward. Many components end up underneath the display, so pay attention to the height of these components when mounting them on the PCB. The push buttons are types with a long operating stem. When the circuit board is mounted in an enclosure then the operating push buttons can extend through the top. The Mode button is positioned slightly lower, to help prevent inadvertent silencing of the alarm.
The display is connected to the circuit board using short lengths of wire or using a male/female pin header. For the display you can choose either a standard two-line LCD or an OLED display (for example Winstar WEH01602), whichever you prefer. The software is suitable for either type. The connections to these displays are identical.

The little loudspeaker is mounted in a suitable location in the enclosure and connected to X2.
The DCF77 receiver is fitted in a separate plastic enclosure and connected via a three-wire cable with a length of about 50 cm to X3. Note: The numbering of X3 on the circuit board runs from left to right, on the Conrad circuit board it runs from right to left. This means the connecting wires have to be crossed over.
A few remarks regarding the position of the DCF77 receiver. Keep the cable from the line adapter away from the receiver. TVs, computers and similar appliances can also cause interference, even some types of fluorescent lights.

(140469)

Web Links

[1] DCF77 signal structure: http://en.wikipedia.org/wiki/DCF77
[2] ATmega pinout: www.hobbytronics.co.uk/arduino-atmega328-pinout
[3] www.elektor-magazine.com/140469

NFC 'e-BoB' Gateway

By Ton Giesberts, Clemens Valens, Dr Thomas Scherer and Jens Nickel

Practically every modern smartphone has an NFC interface which allows it, for example, to read RFID tags or exchange data with another NFC device. Our NFC gateway in e-BoB style lets you explore the capabilities of your own smartphone, but that is not all: the built-in tag IC lets you read and write data using simple commands over a serial interface, with the help of an ATmega328 acting as the bridge. This project makes it easy to create a contactless connection between your own circuits and your smartphone.

Modern smartphones are not only beautiful on the outside, but also well equipped on the inside. For hobbyists the most interesting question is how this marvel of microelectronics can be hooked up to other systems. Transferring information over Wi-Fi or a mobile data connection requires rather complex protocols and accompanying network technology; while Bluetooth is somewhat easier, for some applications the range is too great. That's right, the connection is sometimes not secure enough. Near Field Communication, or NFC (see text box), in contrast, is not only a mature technology, but also, because of its relatively short range of just a few centimeters, is sufficiently secure that it can be used for contactless payments. It is also consumes little power, and communications can be encrypted. Before going further, take a quick look at [1] to see whether your smartphone is equipped with an NFC interface. One piece of good news for hobbyists attached to their Apple devices: the NFC interface is included on the iPhone 6. Thus you can even make use of the notoriously isolationist Apple hardware for these experiments...

The list in the box gives an idea of the range of possible (intended) applications for NFC technology, but the only real limit is your imagination. For example it is easy to create simple projects such as door locks, message displays and so on: see below for more.

NFC tag module with serial interface

The gateway

Many manufacturers offer NFC tag ICs that can be written to and read from by an NFC smartphone. Some of these, for example, can be controlled by a microcontroller over SPI, thus providing a possible link to your own hardware. However, controlling these devices is not a trivial task for beginners, and likewise the design of the required antenna is tricky. We at Elektor Labs have therefore developed an NFC tag module which includes a tag IC from NXP, an ATmega328P microcontroller (which will be familiar from the Arduino Uno) and an antenna. Loaded with its standard firmware this unit can be controlled using simple ASCII-based commands over its serial interface, either from a PC or from another microcontroller. The ATmega then in turn controls the tag IC, saving you as developer from having to know about the details of the I^{2}C communication between the two. We have brought out the UART RX and TX signals to the familiar 'Embedded Communication Connector', or ECC, which allows direct connection using a ribbon cable to the Elektor Extension Shield [5] for the Arduino Uno, or to the USB to-Multi-Protocol Converter elsewhere in this edition. An adapter board for connection to the SAM D20 board that features in our ARM course is also in the works. Finally, for testing and debugging, the Elektor FTDI BoB can be connected directly. The unit is somewhat reminiscent of the 433 MHz wireless gateway published in the April 2014 issue of Elektor [6], and consequently we have dubbed this project 'NFC Gateway'.

Technical Characteristics

- NFC module with UART connection
- Direct reading and writing of data using a PC or microcontroller
- Up to 1904 byte EEPROM and 64 byte SRAM available
- Tag IC on separate breakout board
- Printed antenna on main circuit board
- Serial interface at TTL voltage levels
- Power supply 5 V or 6 V to 12 V
- ECC header, compatible with a range of Elektor projects

NFC e-BoB

When designing projects of this type the question always arises of whether we should offer our readers a ready-built module or if they would prefer a blank printed circuit board so that they can solder the components themselves. For the NFC gateway we have chosen a middle path. The NFC chip itself is an absolutely tiny device, just 1.6 mm by 1.6 mm, and its eight pins are on the bottom surface of the IC. We have therefore mounted this device on its own breakout board, or BoB, which can then be treated like a conventional eight-pin DIL IC. This board is available ready-soldered from Elektor (order code 140177-92). For constructors undaunted by the IC's package (and those that happen to possess a reflow oven) we have also made the bare board available (# 140177-2). **Figure 1** shows the ready-assembled e-BoB 'circuit'. The

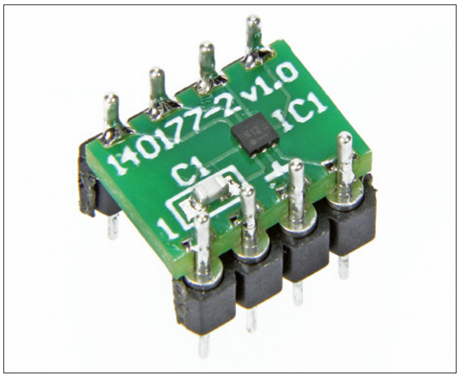


Figure 1. Prototype of the breakout board with NFC tag IC soldered on.

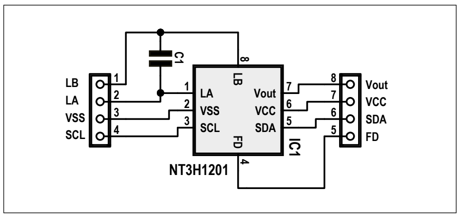


Figure 2. Circuit of the breakout board.

Component List NFC e-BoB Board

Capacitors
C1 = 18pF 5%, 50V, SMD 0603, C0G/NP0 *

Semiconductors
IC1 = NT3H1201W0FHK, SMD XQFN8

Miscellaneous
K1,K2 = 4-pin pinheader, round pins, 0.1” pitch
PCB # 140177-2 v1.0 or ready-assembled unit # 140177-92* [4]
* see text

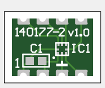


Figure 3. Component mounting plan for the breakout board.

inverted commas are definitely warranted: as can be seen from **Figure 2**, besides the IC and the pins of the two headers there is just a single capacitor, C1, which forms a resonant circuit with the antenna coil. The pins FD, SDA and SCL, forming the serial I²C interface, can easily be seen. The component mounting plan in **Figure 3** also shows the locations of the pads hidden under the IC.
Those interested in finding out more about the IC can take a look at NXP’s comprehensive datasheet [3]. It is in any case worth briefly noting that the device allows a transfer rate of 106 Kb/s, allowing a ‘page’ of 4 bytes to be transferred from the SRAM inside the IC in 0.8 ms. The 2 K device that we are using has a total of 64 bytes of SRAM, a seven-byte UUID (unique serial number) and a total of 1904 bytes of EEPROM available to the user. There is also an even more inexpensive 1 K version of the device.

Piggyback

The NFC e-BoB is in turn mounted on the NFC gateway board, which includes the antenna, the ATmega328P, the power supply circuit and the ECC header.
At the heart of the circuit (see **Figure 4**) is the AVR microcontroller IC1 that manages everything. Above and to its left is the conventional six-pin ISP connector K2 that allows the microcontroller to be programmed. K1 is the ECC header. It is possible to power the gateway over this header using a ribbon cable to connect it to another microcontroller board such as the extension shield. Jumper JP1 should then be fitted in the position marked ‘K1’ on the circuit board’s silk screen, so that VCCin is connected to the gateway power supply line VCCout. The polarization key on the plug on the ribbon cable should always face inwards: the gateway thus functions as a peripheral device.
The gateway can also be used as a standalone controller board to which ECC peripherals can be connected. In this configuration the peripheral can be provided with power. An example peripheral is the USB Multi-Protocol Converter [7]. Here the polarization key of the plug on the ribbon cable should face outwards on the NFC module: for more on this see [6].
If the jumper is fitted in the position marked ‘IC3’ the gateway can be supplied with DC power at any voltage from 6 V to 12 V using an ordinary mains adapter. The current consumption is low at just a couple of tens of milliamps. IC3 regulates this voltage to 5 V at VCCout to power IC1. A second regulator, IC2, is needed to generate the 3.3 V supply required by the tag IC.
K3 and K4 are designed to allow the printed antenna on the circuit board (see the prototype shown in **Figure 5**) to be separated from the rest of the circuit, sawing through its two connections. The antenna can then be mounted at a distance from the main board, connected via a cable, or a different antenna can be connected instead at K3. The circuit board (whose layout is shown in **Figure 6**) is perforated along a line to make separation easier. Initially C6 is not fitted: an additional capacitor with a value of a few picofarads can be added here to trim the resonant frequency of the antenna.
There are three LEDs on the board: D1 is available to the programmer to use as desired, and has no function in the standard firmware. D2, on the other hand, is connected to the Vout connection of the tag IC. The IC is able to power itself by

NFC

NFC is a communication standard for the contactless interchange of data by radio over short distances, perhaps up to a few centimeters. So far it has been used, for example, in micropayment applications: it is already in use by several banks in Europe for payments of amounts up to about 20 Euro and, for example, by German railways in their ‘Touch&Travel’ system. Some universities use NFC chips in their student identity cards to allow small payments. The standard is set by the NFC forum, which besides its founders NXP, Nokia and Sony, includes practically all firms operating in the mobile communications arena.
NFC allows the exchange of data such as telephone numbers, images, audio files and certificates between two devices by just bringing them briefly near to one another. No further action is required. Communication is relatively secure against eavesdropping as a result of the short transfer duration (a consequence of the relatively high data rate of 424 kbit/s at 13.56 MHz) and the short range; there is also an encryption option.
Typical intended applications include:

- Cashless funds transfer (such as Google Wallet and Apple Pay)
- Paperless ticketing
- Transport services
- Online streaming
- Content download
- Access control
- Marketing via the use of NFC tags in shops
- Online banking

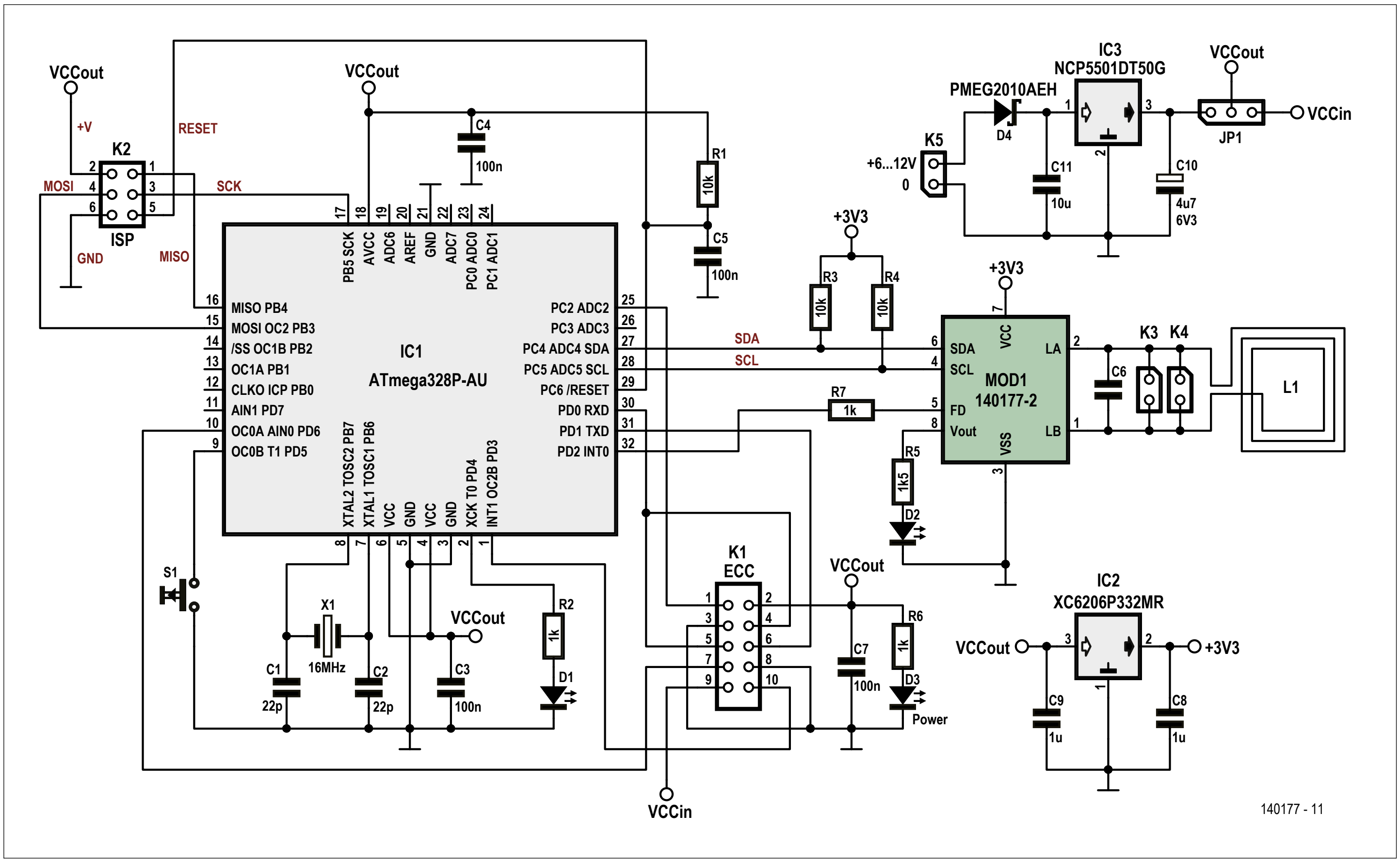


Figure 4. The circuit of the module includes a microcontroller, power supply, the e-BoB and the antenna.

harvesting HF energy from the other party to the communication using the antenna, and can also power external circuitry using this pin. D2 therefore gives a good indication as to whether the antenna has accurately tuned (using C1 on the e-BoB and possibly also C6 on the main board) to resonate at the frequency of the transmitter. If desired the total capacitance can be adjusted while experimenting with varying the distance between transmitter and receiver to achieve optimal reception. If the antenna is separated from the main board and connected via a cable the resonant frequency will be significantly affected and the total capacitance of C1 and C6 will need to be changed.

Finally, green LED D3 indicates the presence of power at VCCout.

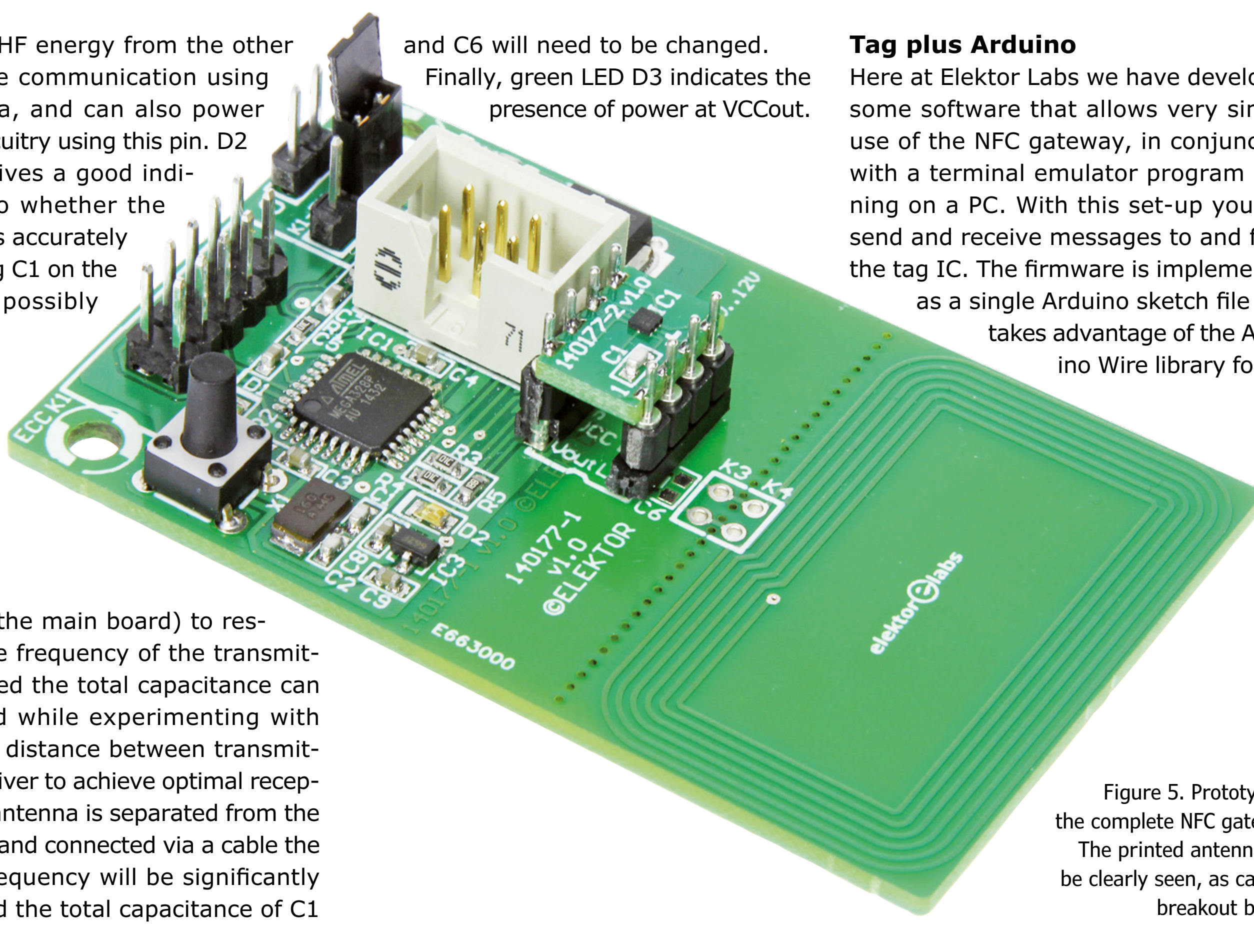


Figure 5. Prototype of the complete NFC gateway. The printed antenna can be clearly seen, as can the breakout board.

Tag plus Arduino

Here at Elektor Labs we have developed some software that allows very simple use of the NFC gateway, in conjunction with a terminal emulator program running on a PC. With this set-up you can send and receive messages to and from the tag IC. The firmware is implemented as a single Arduino sketch file that takes advantage of the Arduino Wire library for I²C

Commands

The NFC gateway is shipped loaded with its standard firmware, which is of course free and available in source code form. A total of four different commands can be processed: it sounds straightforward and indeed it is. There is an erase command, a command to write data and two to read data: see the table. If the module receives a command it does not recognize it will output a 'help' message to the terminal. Reading is simple, in that you simply specify a number of bytes to be read from a memory block (see Figure 7). Writing is more complex, since it is possible to write not just the various forms of standardized message but also free-format 'NTF' messages, which the developer can design himself.

There is not sufficient space here to go into the details, but a manual called '140166_Firmware.pdf' is available as a part of the download accompanying this article [4]. The manual explains clearly the structure of the memory and the range of possible messages.

Each command must be terminated with a <CR>, an <LF> or a <CR><LF> sequence. Most terminal emulator programs can be configured so that they for example output a <CR> character when the Enter key is pressed. Note that in the table the notation [C|c] means that either a 'C' or a 'c' is equally acceptable, with the same meaning.

Command	Command line	Description
Clear	$[C|c]	Erases the tag IC. The NFC device (the smartphone) will now see an 'empty' tag.
Read	$[R|r],<mema>,<count>	Reads <count> bytes from the tag memory, starting with block <mema> (from 0 to 255). If <mema> and <count> are both zero, the whole memory will be read. Memory contents are dumped as hex values and as plain text (ASCII 32 to 127).
GetChars	$[G|g],<mema>,<count>	As 'Read', but only plain text is output, framed by ASCII 02 and 03 characters. Designed for use when interfacing to another microcontroller.
Write	$[W|w],[T|U|<mema>],<data>	Writes one of: a free-form NDEF message (T) a URI NDEF message (U) data to memory block <mema> (from 0 to 255). <data> is the message payload.

interfacing. Of course the source code and the ready-compiled hex file are available for download, via [4].

The most important tasks of the code are carried out in the main loop which takes the form of a simple state machine. The state machine decodes the incoming commands and carries out the corresponding necessary low-level actions. It also checks the status of button S1: when it is pressed, the gateway sends a complete memory dump to the terminal.

Component List NFC Gateway Board

Resistors
Default: 5%, 0.1W, SMD 0603
R1,R3,R4 = 10kΩ
R2,R6,R7 = 1kΩ
R5 = 1.5kΩ

Capacitors
Default: 10%, SMD 0603
C1,C2 = 22pF, 50V, C0G/NP0
C3,C4,C5,C7 = 100nF, 16V, XR7
C6 = not fitted *
C8,C9 = 1µF 20%, 6.3V, X5R
C10 = 4.7µF, 6.3V, tantalum, SMD 0805, 7Ω ESR
C11 = 10µF +80/−20%, 20V, SMD 1206, Y5V

Semiconductors
D1,D2,D3 = LED, green, SMD 0805
D4 = PMEG2010AEH
IC1 = ATmega328P-AU, TQFP 32A, programmed
IC2 = XC6206P332MR, SOT23-3
IC3 = NCP5501DT50G, DPAK 3

Miscellaneous
K1 = 10-pin (2x5) pinheader, 0.1" pitch
K2 = 6-pin (2x3) boxheader, 0.1" pitch
K3,K4 = not fitted
K5 = 2-pin pinheader, 0.1" pitch
MOD1 = 8-way (2x4) socket strip
JP1 = 3-pin pinheader, 0.1" pitch, with jumper
S1 = pushbutton, 6x6 mm, 12V, 50mA
X1 = 16MHz quartz crystal, CL = 18pF, SMD, 5x3.2mm
PCB no. 140177-1 v1.1 [4]
Ready assembled unit # 140177-92* [4]
* see text

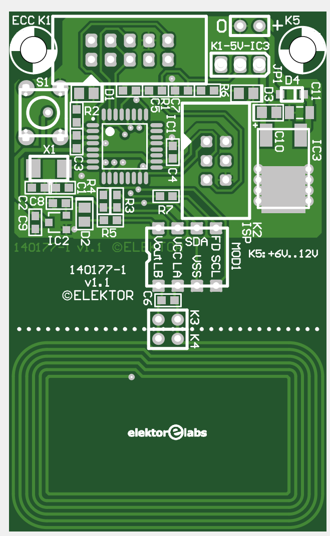


Figure 6. Component mounting plan for the double-sided gateway printed circuit board.

The commands available for controlling the NFC module from the terminal emulator program or from another microcontroller are described in the 'Commands' text box. To understand in more detail exactly what is going on, it is necessary to look at the memory architecture of the NFC tag IC (see **Figure 7**). As mentioned above the larger version of the tag IC makes 1904 bytes of its 2 K EEPROM available. The memory is divided into 'pages' of four bytes each which in turn are organized, rather like a hard disk, into 'files'. These files have a predefined structure, called 'NDEF' (NFC Data Exchange Format), and can include plain text, or so-called URI records in the form of URLs or URNs. More on these possibilities can be found in the high-quality documentation available for download via [4].
One further point of interest for more advanced users: the software prevents write operations to page 0, since it is possible to use this to render the tag IC non-functional. If you do indeed wish to write to this page the code must be modified to allow it. The 64 bytes of SRAM are mapped to overlay pages 248 to 251 of the EEPROM.

Sector adr.	Page address		Byte number within a page				Access
Hex.	Dec.	Hex.	0	1	2	3	conditions
0h	0	0h	Serial number				READ
	1	1h	Serial number			Internal data	READ
	2	2h	Internal data		Lock bytes		READ/R&W
	3	3h	Capability Container (CC)				READ&WRITE
	4	4h	User memory				READ&WRITE
		...					
	15	0Fh					
	...	...					
	225	E1h					
	226	E2h	Dynamic lock bytes			00h	R&W/R
	227	E3h	Invalid access - returns NAK				n.a.
	228	E4h					
	229	E5h					
	230	E6h					
	231	E7h					
	232	E8h	Configuration				See section 8.5.9
	233	E9h					
	234	EAh	Invalid access - returns NAK				n.a.
	...	...					
	255	FFh					
1h	...	...	Invalid access - returns NAK				n.a.
2h	...	...	Invalid access - returns NAK				n.a.
3h	0	0h	Invalid access - returns NAK				n.a.
	...	...					
	248	F8H	Session registers				See section 8.5.9
	249	F9H					
	...	...	Invalid access - returns NAK				n.a.
	255	FFH					

Figure 7. Memory map of the NFC tag IC (see the device datasheet [3]).

Try it out

Using the NFC e-BoB makes life rather easier when assembling the module, but even the 0603-footprint SMD components require a little digital dexterity. For an initial test connect an FTDI BoB [4] to the gateway as shown in **Figure 8**, fit jumper JP1 in the 'IC3' position and power the board over K5.
Configure the terminal emulator program to use the correct COM port and set the baud rate to 38400 baud. When you press the button on the gateway as described above the memory dump should appear in the terminal emulator program's window (see **Figure 9**). To check that communication works in the reverse direction, type the following string into the emulator:

$R,0,0<CR>

Again, the entire contents of the unit's memory should be dumped to the screen. Now we can try out the other commands supported by the standard firmware (see the text box). You can try writing a short text string to the tag and then see how it appears in the hex dump.
All that is required for the next test is a suitable NFC application for your smart-

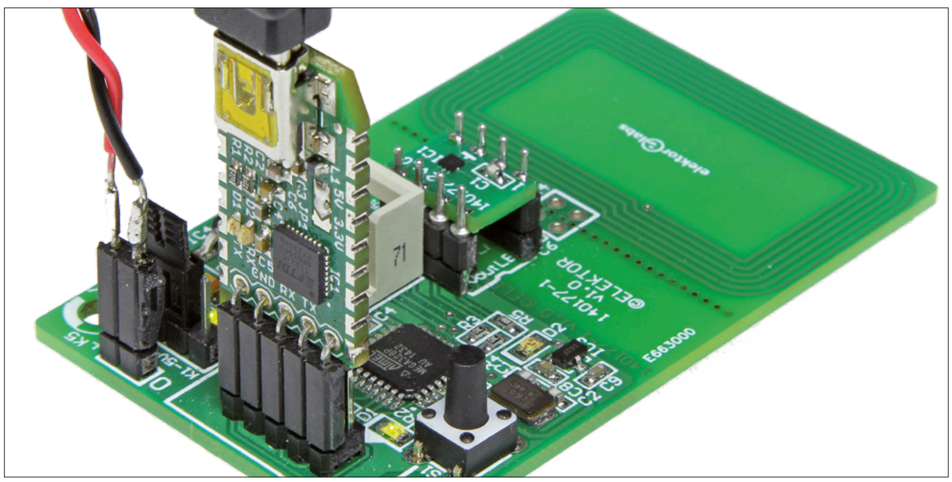


Figure 8. The FTDI BoB can be connected for testing.

Web Links

[1] https://en.wikipedia.org/wiki/List_of_NFC-enabled_mobile_devices
[2] NFC Forum website: http://nfc-forum.org/
[3] NT3H1201W0FHK datasheet: www.nxp.com/products/identification_and_security/smart_label_and_tag_ics/ntag/NT3H1201W0FHK.html
[4] www.elektor-magazine.com/140177
[5] www.elektor-magazine.com/140009
[6] www.elektor-magazine.com/130023
[7] www.elektor-magazine.com/130542

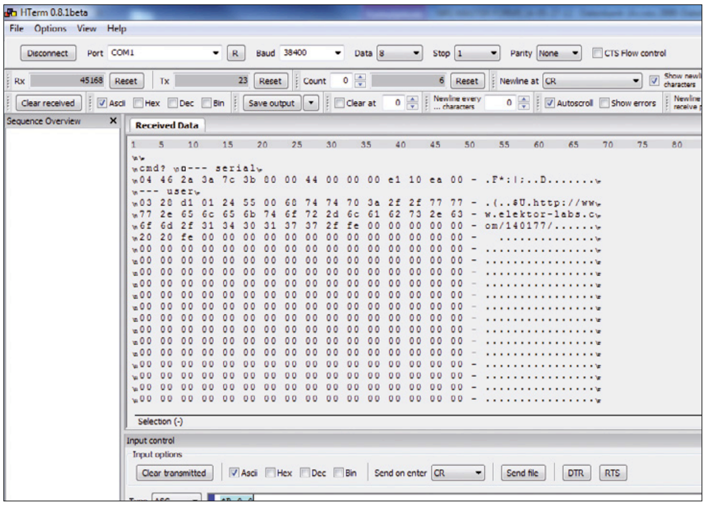


Figure 9. Memory dump as it appears in the HTerm terminal emulator: a URL has been stored in the tag.

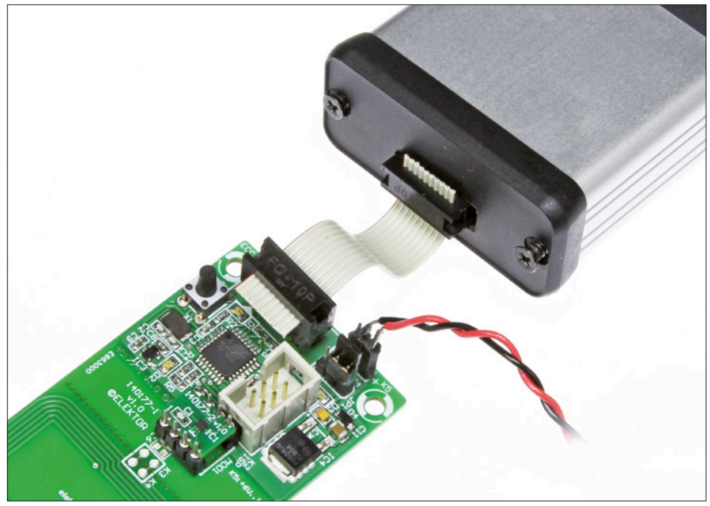


Figure 10. The NFC gateway can be connected directly to the USB converter described in the previous issue of Elektor in order to interface the unit to a PC.

phone. We have had a good experience with 'TagWriter' from NXP, which can, for example, be downloaded for free from Google Play. This application not only lets you read short text strings and URLs from the tag, but also write them to the tag.

Demonstrations

For our first application we can replace the rather flimsily attached FTDI BoB with our USB Multi-Protocol Converter [7], printed elsewhere in this edition (see **Figure 10**). The converter is connected to the gateway using a 10-way ribbon cable, making sure that the polarization key on the plug faces outwards. Now, with the command

$W,U,www.elektor.com<CR>

the given web address will be written to the tag. The Android applications 'NFC Tasks' and 'NFC Tools' by 'wakdev' offer a wide range of functions and are free in their simplest versions. Launch 'NFC Tools' and hold the smartphone next to the gateway, and it should automatically open the home page of our website (and show you all our latest special offers!).

We have also developed a small demonstration for the Arduino Uno in conjunction with the extension shield to demonstrate how to use the module with a microcontroller: see the text box 'NFC Display'.

(140177)

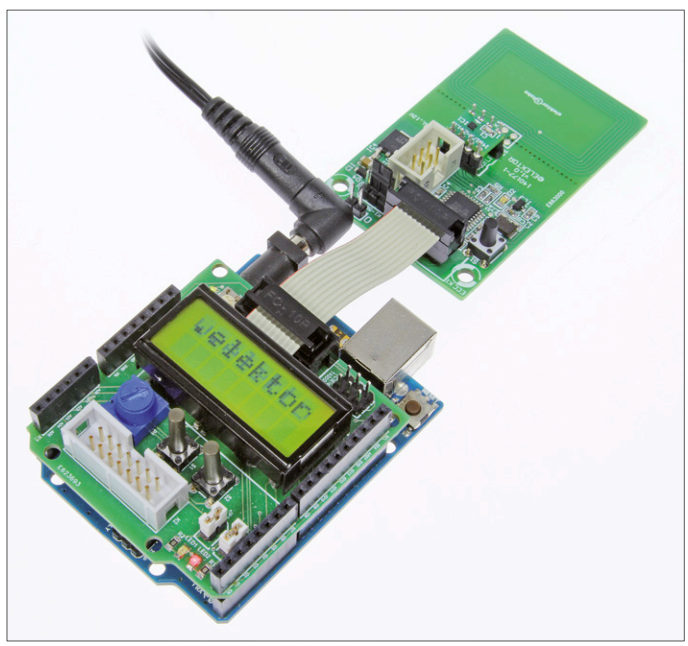
NFC display

The gateway can easily be used as a peripheral for another microcontroller. As a demonstration of this we have developed a small application for the unit in combination with and Arduino Uno and an Elektor Extension Shield [5]. The Elektor Extension Shield is connected to the gateway using its ECC connector (making sure that the polarization key on the ribbon cable connector always points inwards). Jumper JP1 should be set in position 'K1' so that the gateway is powered from the Arduino.

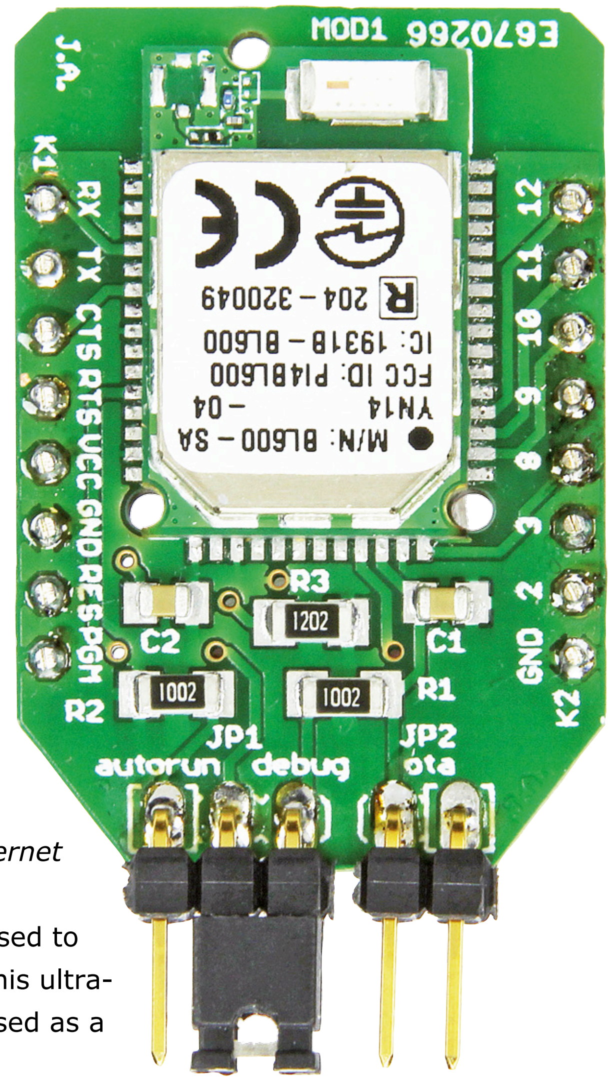
The demonstration firmware for the Arduino Uno was developed as a project in Atmel Studio 6 ('ElektorNFCDisplay' at [4]). The hex file must be downloaded into the microcontroller over the ISP interface on the shield using a suitable programmer.

If you press the left button on the shield, a message stored in the NFC tag will be read via the gateway and will be scrolled across the display. The right button resets the scrolling to the beginning of the message, and the potentiometer can be used to adjust the scroll speed.

The demonstration software always shows the first 48 characters stored in the tag. A rather more elegant version would filter out the NDEF prefix 'T' and cope with messages of different lengths; but the version here suffices to let you use your smartphone to leave simple messages for loved ones (or even colleagues).



BL600-eBoB (2)



Editing, compiling, and transferring a program

By Jennifer Aubinais (France) elektor@aubinais.net

If we're to believe certain predictions, we'll soon no longer be talking about the *Internet of Things*, but rather about the *Internet of All Things*. This is going to imply a proliferation of wireless communication **and** low power consumption for the circuits used to connect up all these things. Which is exactly the purpose of this ultra-low consumption radio communication board, which can be used as a miniature computer, easily programmable in *smartBASIC*.

In the first of this series of articles devoted to the BL600-eBoB [1], we introduced the hardware: on the one hand the UART port, for communicating with a connected object — in our case a PC via the FT232 eBoB; and on the other, seven input/outputs we can use as we like: two ADC inputs, the I²C port, the SPI port. Then, using the OTA (over-the-air) function, we have seen how to transfer via our BL600-eBoB's radio link a first UART program, downloaded using an Android phone.

Here we're proposing a first application program, after familiarizing ourselves with the editing, compiling, and then transfer tools. This is going to be a simple light-chaser – but a Bluetooth Low Energy light-chaser! We'll be going through it step by step, and even one toe at a time, as even though the possibilities of this module are remarkable, you do have to master it. In the third article, we'll be discovering the principles of programming events in smartBASIC, which will enable you to create your own projects.

Editing

We're going to start by downloading the latest version of the BL600 firmware (Firmware Files version 1.5.70.0 – Revision 5 [2]) along with an editor; Laird Technologies recommends Notepad++ (Figure 1) [3] which recognizes smartBASIC syntax.
To obtain a login in the SOFTWARE DOWNLOADS section of the Laird Technologies site, click "If you need credentials, please click here" and wait a few days. Once you're logged in, download Firmware Files version 1.5.70.0 – Revision 5 which you'll need to unzip. This contains (**Figure 2**):

- program examples in the *smartBASIC_Sample_Apps* directory
- the UwTerminal.exe program in the *smartBASIC_Sample_Apps* directory
- the library in the *smartBASIC_Sample_Apps/lib* directory
- a number of examples (*UserManualExampleCode*)
- the special Notepad++ configuration for smartBASIC (*smartBASIC(notepad++).xml*)

Once the Notepad++ editor is installed, run it. In the menu, select *language* (**Figure 4**) then click *Define your language.* Click on *import*, select the *smartBASIC(notepad++).xml* file from the ZIP

Figure 2. The options on the Laird homepage, to which you're soon going to be a frequent visitor.

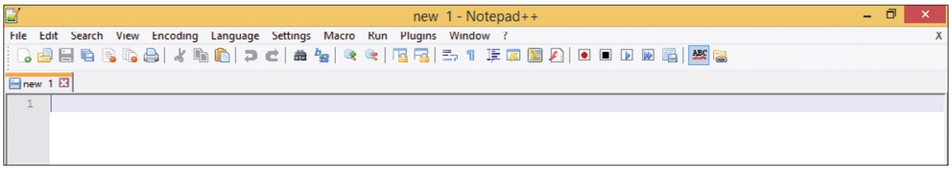


Figure 3. The Notepad++ editor you'll be using to write your applications.

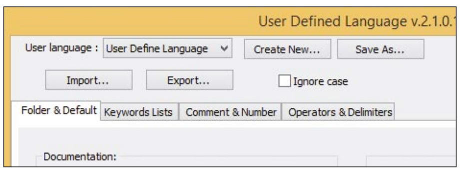


Figure 4. For Notepad++ to learn the smartBASIC syntax...

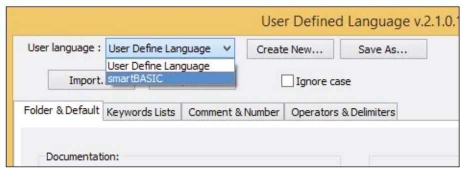


Figure 5. ... you just have to load the appropriate configuration file provided by Laird.

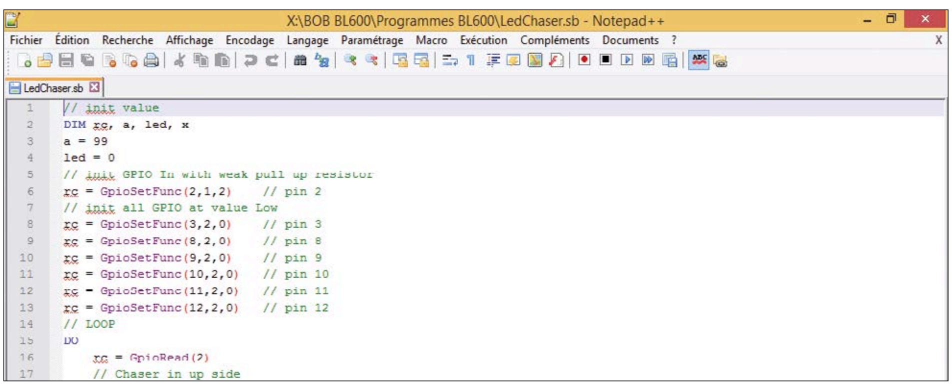


Figure 6. The code editing window (full listing at the end of the article).

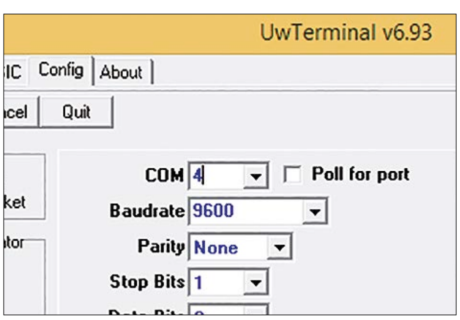


Figure 7. Configuring the *UwTerminal* communication program

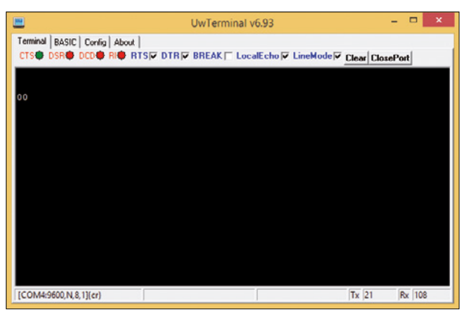


Figure 8. The two 00 on the *UwTerminal*'s black screen – that's a good sign!

file and restart Notepad++. Lastly select *language* and click on *User language* (**Figure 5**), select *smartBasic*, and close. Notepad++ is now configured to display the syntax of the smartBASIC code in color. Try for example the LedChaser.sb program (**Figure 6**), which we'll be coming back to later in this article. But we're not going to be using this now for learning how to compile in the next paragraph; for this, we're going to use the upass program we've already used in the BL600 e-BOB article.

Compiling

Before we go any further, we ought to point out that the BL600 e-BoB must be connected to the PC via a UART/USB interface. For this, I recommend using the FT232 eBoB. It's the first in Elektor's eBoB series, a simple and very effective serial/USB bridge covered in the September 2011 issue [4]. The duo formed by our two eBoB's appears in **Figure 14** and on the adjacent photo. We'll come back to this arrangement later when we talk about our light-chaser application example. But first, the time has come to familiarize ourselves with compilation. To do so, we're also going to use the UwTerminal.exe program that's included in the software bundle downloaded from Laird. At the end of the previous article, our BL600 e-BoB was left in *AutoRun* mode, in which it runs the UART program previously loaded into the module. In order to compile (and transfer), jumper JP1 must be in the *cmd* position (called *debug* in the previous version of the BL600 e-BoB), while JP2 (*ota*) is not fitted. Reset the module either using the button, or by momentarily interrupting the power. Run *UwTerminal.exe* then select the correct COM port on the FT232 eBoB and a data rate of 9600 baud (**Figure 7**).

Press Enter when the black screen appears (**Figure 8**). Although it is possible to do everything at once, we're going to go through it step by step, starting with compiling. Right click on the black background. A window will open. Select *XCompile* (**Figure 9**) and then the file $autorun$.upass.vsp.sb. A compilation window opens, then the result of the compilation is displayed (**Figure 10**). The compilation creates the file $autorun$.upass.vsp.uwc supplied in the download with the previous installment [1]. Now here you are in a position to re-create it yourself. And soon, once you are familiar with the procedure described, you'll be able to modify it as you see fit.

Transferring

We're not going to use the BLE Over The Air service for downloading the program we've just compiled on our BL600 e-BoB. As it's connected to the PC, this is the quickest way to transfer the compiled program. What's more, it lets me show you in detail the steps that are going to help you write your own programs and… make your own mistakes — as that's the best way to learn! The configuration of the jumpers is JP1 in cmd mode and no jumper for JP2 (OTA). As the UwTerminal.exe software is already running on your PC, all you have to do is press ENTER to verify that the BL600 eBoB responds correctly with 00 on the UwTerminal's black screen. If necessary, you may have to reset the BL600 eBOB.
If you've already downloaded a program into the BL600, e.g. with the first article (Over The Air), you must now delete the program memory by entering the following command AT&F 1 which will also restart the module. Select Download (**Figure 9**). Then BASIC, then Load Precompiled BASIC and lastly the compiled program upass.vsp.uwc. The transfer of the compiled program ends with the message +++ DONE +++ (**Figure 11**).

To get a list of the programs downloaded, enter the command *AT+DIR* and to run your program, the command *AT+RUN "upass"* (**Figure 12**).

We've used UwTerminal.exe to compile, transfer, then run the program on your BL600 e-Bob. These three steps can be combined into a single one using the command *XCompile + Load + Run* (**Figure 9**).
Reminder of common commands:

- AT I 0: BL600 revision number
- AT I 3: BL600 firmware version
- AT+DIR: list of the programs in the BL600
- ATZ: reset BL600
- AT&F 1: clear memory and restart BL600
- AT+RUN "xxxx": run program xxxx

Wireless light-chaser command

This little 6-LED light-chaser controlled by the BL600 changes the chaser direction when you press the button. It makes simple use of the BL600's logic inputs/

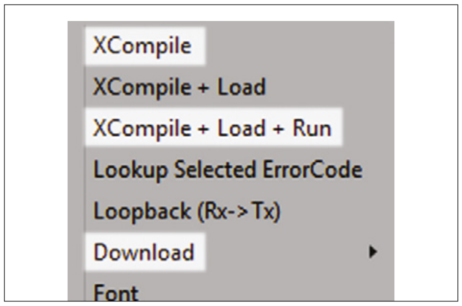


Figure 9. Xcompile menu | Download | *XCompile + Load + Run*

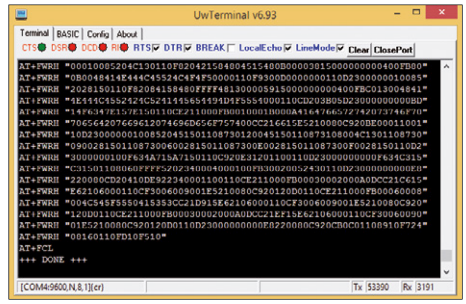


Figure 11. The transfer of the compiled program ends with the message +++ DONE +++.

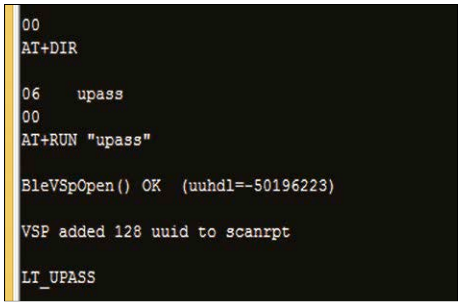


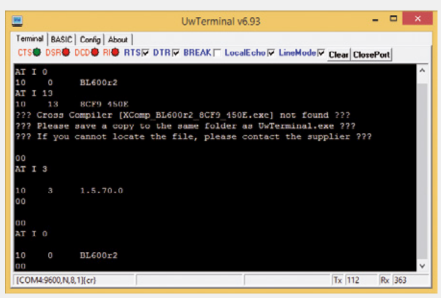
Figure 10. The result of the compilation is displayed in the UwTerminal window.

Figure 12. *AT+DIR* gives a list of the programs and *AT+RUN* runs your program.

> This module offers amazing possibilities – you just need to master it.

Error: Cross Compiler [XComp_BL600…….exe] not found ???

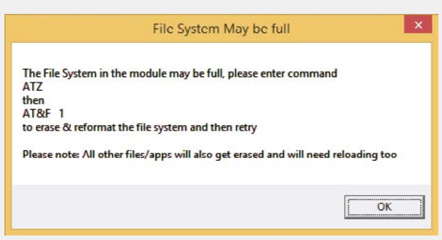
This error occurs when the firmware in your BL600 doesn't match the BL600 bundle downloaded from the Laird Technologies website. The commands for identifying the firmware in your BL600:



AT I 3 for the firmware
AT I 0 for the revision

Once you have identified the firmware and revision of your BL600, download the corresponding bundle from the Laird Technologies site [2]. Perform the compilation described again.

Error: The File System in the module may be full



This error is due to a lack of memory for the new program.
Clear the BL600 memory using the commands ATZ then AT&F 1 and download your smartBASIC program again.

Listing LedChaser.sb

```
// init value
DIM rc, a, led, x
a = 99
led = 0
// init GPIO In with weak pull up resistor
rc = GpioSetFunc(2,1,2)     // pin 2
// init all GPIO at value Low
rc = GpioSetFunc(3,2,0)     // pin 3
rc = GpioSetFunc(8,2,0)     // pin 8
rc = GpioSetFunc(9,2,0)     // pin 9
rc = GpioSetFunc(10,2,0)    // pin 10
rc = GpioSetFunc(11,2,0)    // pin 11
rc = GpioSetFunc(12,2,0)    // pin 12
// LOOP
DO
  rc = GpioRead(2)
  // Chaser in up side
    IF (rc == 1) THEN
   IF (led == 0) THEN : GpioWrite(12,0) : GpioWrite(3,1) : ENDIF
   IF (led == 1) THEN : GpioWrite(3,0) : GpioWrite(8,1) : ENDIF
   IF (led == 2) THEN : GpioWrite(8,0) : GpioWrite(9,1) : ENDIF
   IF (led == 3) THEN : GpioWrite(9,0) : GpioWrite(10,1) : ENDIF
   IF (led == 4) THEN : GpioWrite(10,0) : GpioWrite(11,1) : ENDIF
   IF (led == 5) THEN : GpioWrite(11,0) : GpioWrite(12,1) : ENDIF
   led = led + 1
   IF ( led == 6) THEN : led = 0 : ENDIF
  ELSE
  // Chaser in down side
   IF (led == 6) THEN : GpioWrite(3,0) : GpioWrite(12,1) : ENDIF
   IF (led == 5) THEN : GpioWrite(12,0) : GpioWrite(11,1) : ENDIF
   IF (led == 4) THEN : GpioWrite(11,0) : GpioWrite(10,1) : ENDIF
   IF (led == 3) THEN : GpioWrite(10,0) : GpioWrite(9,1) : ENDIF
   IF (led == 2) THEN : GpioWrite(9,0) : GpioWrite(8,1) : ENDIF
   IF (led == 1) THEN : GpioWrite(8,0) : GpioWrite(3,1) : ENDIF
   led = led - 1
   IF ( led == 0) THEN : led = 6 : ENDIF
  ENDIF
  // tempo : speed
  for x = 0 to 2000
  next
DOWHILE (a != 0)
```

Web Links

[1] BL600-eBoB, Elektor 2/2015, p. 42
www.elektor-magazine.com/140270

[2] https://laird-ews-support.desk.com/?b_id=1945

[3] http://notepad-plus-plus.org/

[4] www.elektor-magazine.com/110553

[5] https://www.youtube.com/watch?v=SxwaVI0Kkk8

[6] Bluetooth Low Energy Wireless Thermometer, Elektor 1/2015, p. 64
www.elektor-magazine.com/140190

outputs. This time, we're not going to look at either analog/digital conversion or pulse width modulation — both of which are possible with the BL600.

The serial link between the FT232 eBoB and our new BL600-eBoB was described in last edition's article. Remember that the operating voltage must be 3.3 V. There's nothing special about the BL600-eBoB's inputs/outputs: 6 LEDs, each with its 470-Ω dropping resistor (approximate value).

As for the BASIC programming, we're going to concentrate here on the general-purpose input/output (GPIO) functions by taking a closer look at the program for this light-chaser (**listing**).

GPIOSETFUNC(nSigNum, nFunction, nSubFunc)

The second argument for this function defines the role of the inputs/outputs on the GPIO pin identified by the first argument.

a. Configuration as a logic input

The push-button is connected to GPIO pin 2, pulled up to high via an internal resistor:

`rc = GpioSetFunc(2,1,2)`

- **nSigNum** = 2: GPIO pin 2
- **nFunction** = 1: port as input
- **nSubFunc** = 2: internal resistor

b. Configuration as a logic output

Each LED is connected via a dropping resistor to one of the outputs (3, 8–12). E.g. configuring output 12:

`rc = GpioSetFunc(12,2,0)`

- **nSigNum** = 12: GPIO pin 12
- **nFunction** = 2: port as output
- **nSubFunc** = 0: output low

rc is the return code (= 0x0000 when the function has been successful).

GPIOREAD(nSigNum)

Read (logic input) on the GPIO pin identified by the argument.

The push-button is connected to GPIO pin 2 configured as an input with an internal pull-up resistor. This input is read as follows:

`rc = GpioRead(2)`

- **nSigNum** = 2: GPIO pin 2
- **rc** returns the value of the input signal on pin 2: 0 or 1

Note: if the GPIO input is configured as an analog input, **rc** returns the voltage at the terminals of this input. We'll be using the A/D converter in a future article.

`GPIOWRITE(nSigNum, nNewValue)`
Write (logic output) to the GPIO pin identified by the first argument (if the pin number is invalid, nothing happens). Pins 3 and 8–12 are configured as outputs. In the program, we'll need to set the various outputs to 0 or 1:
`GpioWrite(12,0)`

- **`nSigNum`** = 12: GPIO pin 12
- `nNewValue` = 0: sets to low
  Note: GPIO pins can be configured for pulse width modulation. For the BL600, this function is limited to a choice of up to two pins.
  The value of nNewValue is in the range 0 to N, where N is the max. modulation index corresponding to a duty cycle of 100 %. In a future article, we'll be making use of this possibility of the BL600-SA module by way of an application example in a little vehicle remote controlled via Bluetooth [5].

At the end of this second article devoted to such a small module — in fact the third, if we count the Bluetooth wireless thermometer [6] — there's still a lot more I want to say about it. I've been having fun with it now for over 18 months, and I'm delighted to share my experience with you. In passing, I'd like to thank the whole team at Laird Technologies for their help, and I hope to see you again in the next edition for programming the BL600.

(150014)

Selection of topics

to be covered in the future episodes of this series on the BL600-eBoB:

- handler, events
- the Red Green Blue program
- Low Energy, 5 µA
- the I²C & SPI ports
- Bluetooth communication
- explanation of the wireless remote thermometer program
- writing a program for Android
- writing a program for iOS (hmm... the Apple license isn't free)

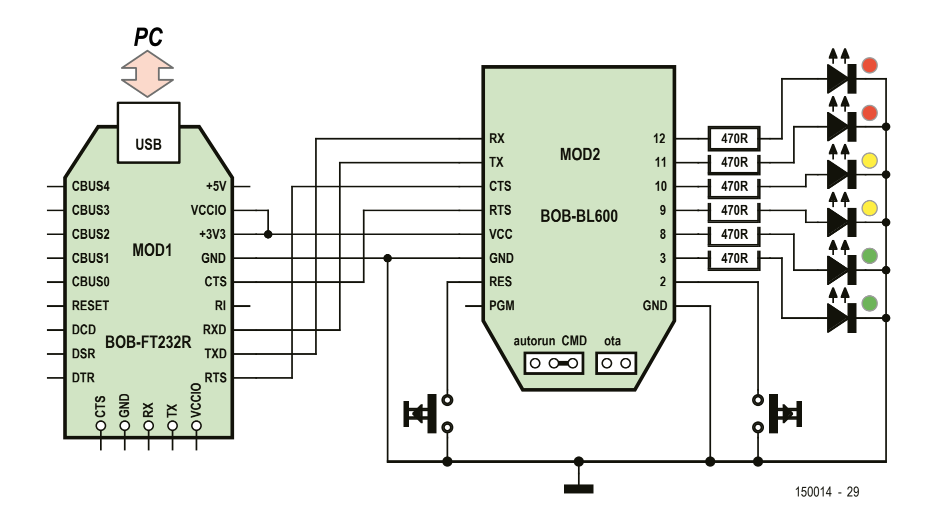


Figure 14. The BL600-eBoB is self-contained, but in order to program it from the PC, I use it in conjunction with the FT232 USB/serial eBoB bridge.

Component List

Resistors
R1–R6 = 470Ω

Semiconductors
D1–D6 = LED, 3mm (select color)

Miscellaneous
K1,K2 = pushbutton
MOD1 = FT232 eBoB, assembled, Elektor Store # 10553-91 (elektor.com)
MOD2 = BL600-eBoB, assembled, Elektor Store # 140270-91 (elektor.com)

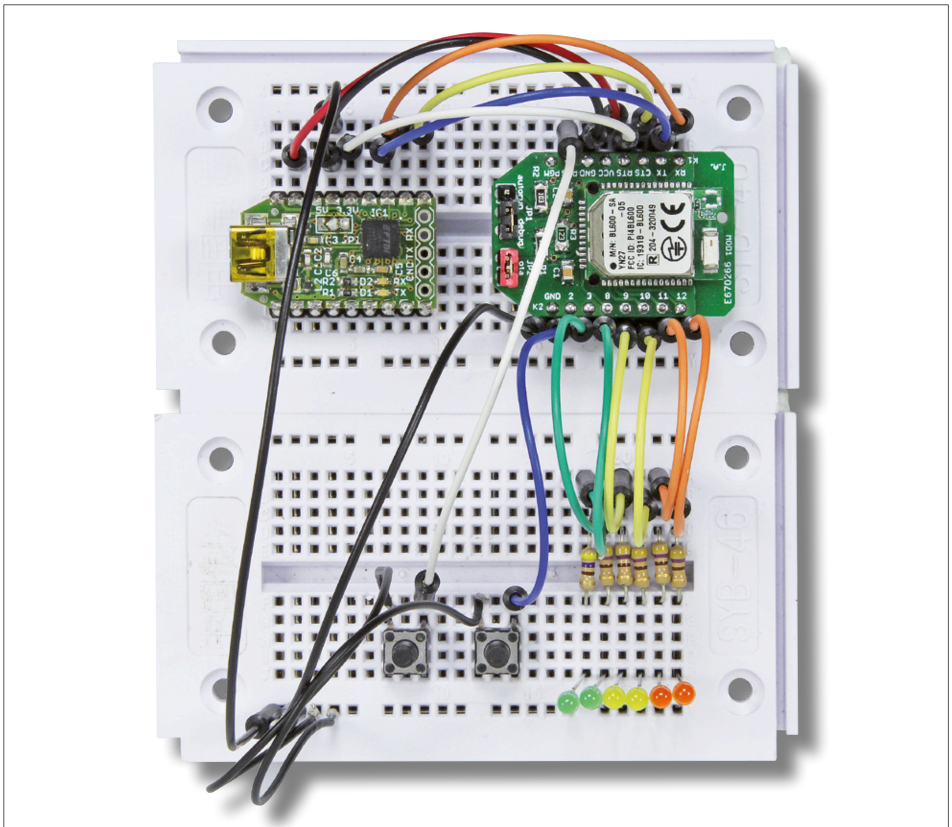


Figure 15. The light-chaser built on perforated prototyping board.

USB-to-Multi-Protocol Serial Converter

head-banging... bit-banging, no more!

By **Ton Giesberts** and **Clemens Valens** (Elektor Labs)

A good many situations exists where a computer has to communicate with a device sporting some kind a serial port, be it RS-232, RS-485, I²C, SPI, or a microcontroller serial port. How do you manage when all you have at your disposal is a bunch of USB connectors?

In terms of coverage of the traditional serial port (see **inset** "What's a serial port anyway?") Elektor has a long history of solutions, projects and workarounds (see **inset** "A long history"), all based on dedicated USB-to-Serial converter chips. But what about the other types of serial port? Microcontroller and other electronics enthusiasts encounter ever more components having I²C and SPI interfaces. These are serial ports too and it seems logical to provide USB converters for these communication standards as well. Actually, our Serial Bob [6] lets you do this thanks to the FT232R's bit-banging mode. However it requires executing the protocol in software rather than in hardware, which sadly slows things down. But today there are better or easier solutions to achieve our goal.

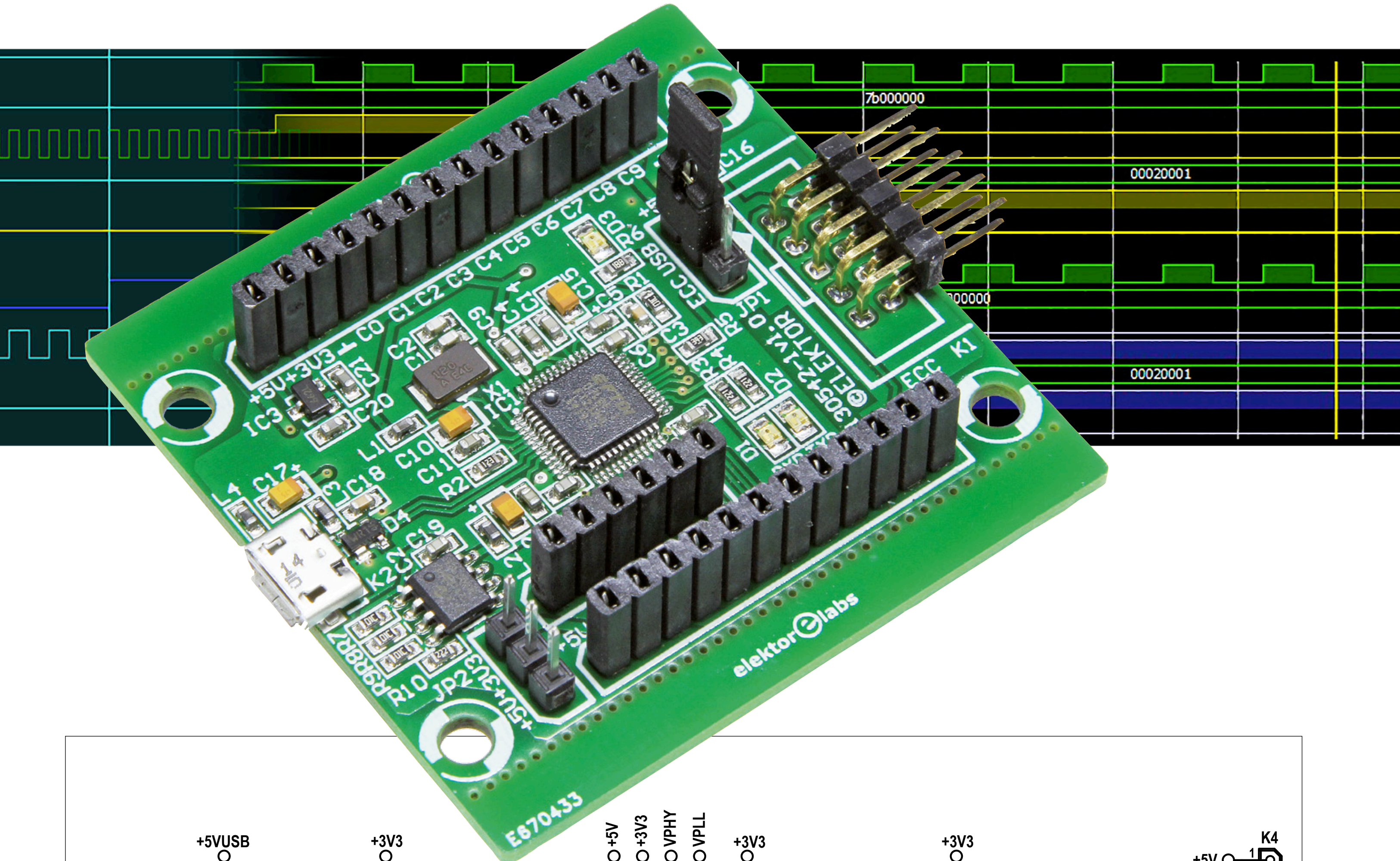
FT232H

One solution is using the FT232H. Although its type code closely resembles FT232R, inside the 'H' chip is very different. Firstly, although both are USB 2.0 compliant, the 'R' chip is a Full Speed device (12 Mbit/s) whereas the 'H' is a High Speed device (480 Mbit/s). Secondly, in terms of the converter capabilities, the 'R' is a USB UART (with some additional bit-bang options), while the 'H' is described as a *Hi-Speed USB to Multipurpose UART/FIFO IC*. Consequently the 'H' is not just a UART, it also does I²C or SPI and on top of that it has parallel port functionality. All this is made possible by its so-called Multi-Protocol Synchronous Serial Engine (MPSSE).

MPSSE

According to the manufacturer's documentation the MPSSE provides a flexible means of interfacing synchronous serial devices to a USB port. Note the word synchronous. Synchronous serial communication protocols use at least two connections — a bus — one for data and one for the clock signal. In such systems there is one device — usually the master —providing the clock signal, with the other devices connected to such a bus busy synchronizing the transmission and reception of data bits to this clock. This approach obviates the need for super precise timing because everyone is watching the same clock. By contrast, in asynchronous communication systems the clock signal is combined with the data, allowing the use of only one wire. On the down side, extracting the clock from the data in every receiving device requires precise timing. To use an analogy, in a synchronous system all participants use the village's church clock; in an asynchronous system everybody relies on their own wristwatch. Protocols like SPI, I²C and JTAG are examples of synchronous communication schemes. The MCU's serial port and its UART are examples of asynchronous communication interfaces.

Because the MPSSE is multi-protocol, it allows communication with many different types of synchronous devices, the most popular being SPI, I²C and JTAG. Data formatting and clock synchronization can be configured in a variety of ways to satisfy almost any requirement with speeds up to 30 Mbit/s. Although it is not explicitly mentioned in the name, the MPSSE also supports asynchronous serial communication, albeit limited to one protocol.
The MPSSE module of the FT232H features not only serial communication protocols, it can also do parallel synchronous communications in a variety of ways. One is the so-called FT1248 mode, an interface using a 1-, 2-, 4-, or 8-bit wide bidirectional data bus. On the face of it the MPSSE can also emulate the famous (some say, $\overline{\text{famous}}$) 8048/8051 multiplexed address and data bus, but it is not easy to find information on using this mode. FT245-style synchronous and asynchronous FIFO modes are available too (the FT245 is a USB to parallel FIFO interface chip from the same manufacturer).
In addition to the serial and parallel protocols, additional GPIO signals are available and free-form bit-banging is supported too.



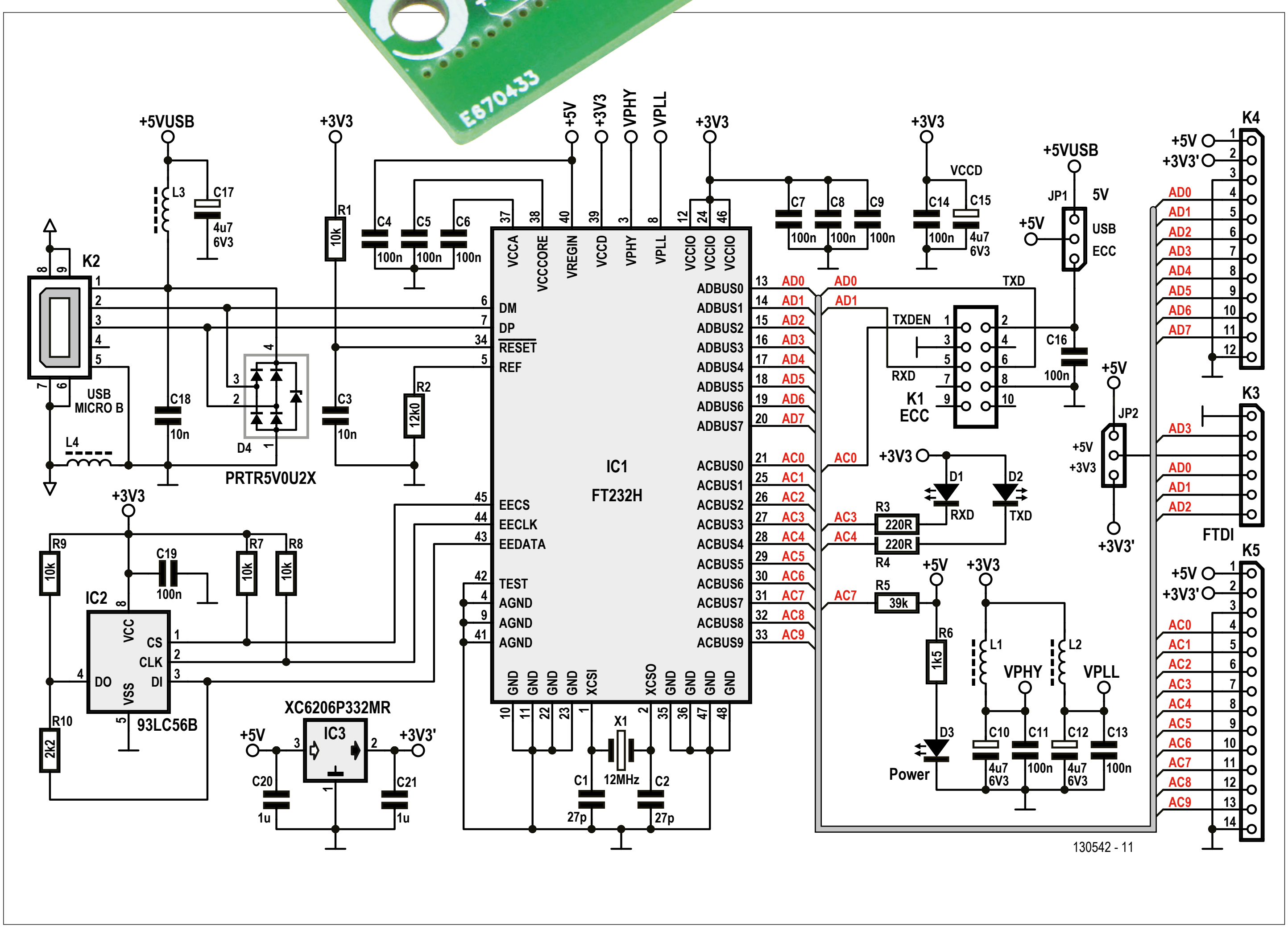


Figure 1. Circuit diagram of the FT232H board.

FTDI only?

In this article we mention only FTDI products, and in Elektor we often use FTDI USB-to-Serial converter chips. Is it because only FTDI makes these kinds of ICs? By no means. Several other semiconductor manufacturers produce USB interface chips too. Here are a few you may have heard of:

- Prolific and their famous PL2303;
- Cypress has always been very active in the USB arena. A well-known chip is the FX2LP, their CY7C65211 is similar to the FT232H except that it is a Full Speed USB device;
- Texas Instruments, being world's largest semiconductor manufacturer of course also has some products in this market. A well-known device is the TUSB3410 that has an 8052 inside;
- Located in China the Nanjing QinHeng Electronics Co. produces several USB interface chips. One of their products, the CH340, can be found on the internet in extremely cheap EEPROM programmers;
- Microchip sells the MCP2200, a USB-to-Serial converter. It appears that it is actually a PIC18F14K50 running a USB stack.

Currently the MPSSE is available in four FTDI devices: FT2232D, FT2232H, FT4232H (the first digit indicates the number of UARTs) and the FT232H (one UART). The FT232H offers the MPSSE module with the largest number of protocols.

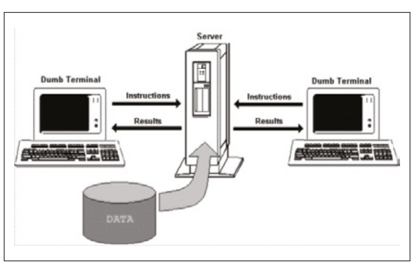
Into the circuit

We decided to design a multi-purpose board for the FT232H because it offers the most options compared to its family members. The resulting schematic pictured in **Figure 1** is downright simple, although it appears cluttered due to all the filtering and decoupling capacitors and ferrites. Because it is also a kind of break-out board (BoB), all pins potentially of interest to you are brought out to connectors K4 and K5. Besides these connectors we put two other connectors on the board: K1 and K3.

K1 is an *Elektor Embedded Communication Connector* (ECC) that's also present on some of our other boards intended for microcontroller experiments. By shorting the pins 1 and 2 of jumper JP1, connector K1 can power the board in case you don't want to load the USB port too much. K3 provides an FTDI-cable compatible connection, so the FT232H board can replace such a cable. There is a subtlety to be aware of here concerning the signal levels on this connector (and on all other connectors for that matter): all the signal levels are 3.3 V, not 5 V. Luckily the chip's pins are 5-V tolerant so it can safely be used in a 5-V system. With jumper JP2 you can select the voltage on the VCC pin of the FTDI 'cable'. On a real FTDI cable this is invariably 5 V, but for 3.3 V systems it is often more practical (and safer) to have 3.3 V on this pin. IC3 is the voltage regulator providing 3.3 V to K3, which avoids unnecessary loading of

What's a serial port anyway?

In the early days of computing, when computers still were expensive and large, a common way to save a bit of money was to buy one computer (the mainframe), put it in the basement and connect several terminals to it. Users would employ a terminal to interact with the computer; the terminal itself was nothing more than a keyboard and a display (hence 'dumb terminal'). Connecting the terminals to the computer in an economical way and allowing a certain distance finally resulted in the well-known RS-232 standard. This standard specifies an interface with up to 25 pins including a wire to send data (TXD) and a wire to receive data (RXD). Some fifty years later this kind of interface has virtually disappeared; all that's left of it today is a 3-pin interface with the signals RXD, TXD and GND. Also in most cases the RS-232 voltage levels have been replaced by either 5 V (TTL) levels or even lower. That's why you cannot connect a microcontroller directly to a real RS-232 port, the voltage levels are incompatible.



Why is it called a serial interface? This has to do with the way computers handle information. Inside a computer information is represented by bits. A bit can have two levels: one or zero. Because information according to often concerns groups of more than two objects (like the alphabet with its 26 objects), bits sequences are needed to uniquely identify each object. Somewhere along the line it was felt that grouping sequences of 8 bits into so-called bytes was a good idea. Information can be transported eight times as fast if it is transported in bytes. To send bytes from one device to another a port with a 'width' of eight wires is needed (ignoring the common). Such a port conveys bits in parallel, and so it was called a parallel port. If you want to send a byte over one wire you must send the bits that make up the byte one by one, i.e. they have to be put in series, which gives you the serial port.

A lot of water passed under the bridge since the birth of the first serial port and as it happens with everything touched by engineers, many different versions have seen the light. Today, the words 'serial port' can indicate many kinds of protocols and yet it always refers to the serial port as in the old days. SPI, I²C, USB, JTAG, I²S, Ethernet, SATI, SWD, etc. are serial ports too that transport information bit per bit and there are many more.

the 3.3-V rail generated inside IC1.
Three LEDs on the board provide visual feedback in certain situations. LED D3 shows if the board is powered. JP1 plays an important role here.

When the board is used as a 'normal' USB-to-Serial converter, LEDs D1 and D2 light up when something is happening on the RX and TX lines. In other modes they do not have a predefined function. To be able to deploy the full power of the FT232H an external EEPROM is necessary. In the EEPROM configuration data is stored like the vendor's USB VID (vendor ID) and PID (product ID), the power source (bus-powered or self-powered) and the function of the ACBUS pins. Other parameters control the hardware interface mode; feel free to delve into the datasheet for all the possible options. One thing that should be mentioned here concerns the USB Suspend Mode (i.e. a low power state in which the chip will not draw current from the USB port) because it has a visible artifact in the schematic in the shape of R5. If this mode is enabled a logic Low on ACBUS7 will make the chip enter USB Suspend Mode. The default setting in the EEPROM for the ACBUS7 pin is input with pull-down resistor, so an unconnected ACBUS7 pin will put the chip to sleep. R5 connects ACBUS7 to +5 V to prevent this from happening automatically. The value of R5 is given in the datasheet.

Though the FT232H operates just fine without an EEPROM, it will have limited capabilities. Connecting an EEPROM to the chip results in the default values being loaded. To change these values you can use FTDI's utility FT Prog. More on that later.
Resistor R2 (1%) and xtal X1 (±30 ppm) are 'precision' parts that are obligatory for IC1 to function. Xtal loading capacitors C1 and C2 ensure the on-chip oscillator works properly. The value of these two capacitors depends on the crystal, although the given values will work in most cases.

K2 is the USB connector. It is a micro-USB so you can use your phone charger cable (if it is detachable). Diode D4 protects the chip's USB data inputs against potentially dangerous voltage surges and noise. Components L3, L4 and C18 provide some protection against high-frequency noise on the power and shield wires.

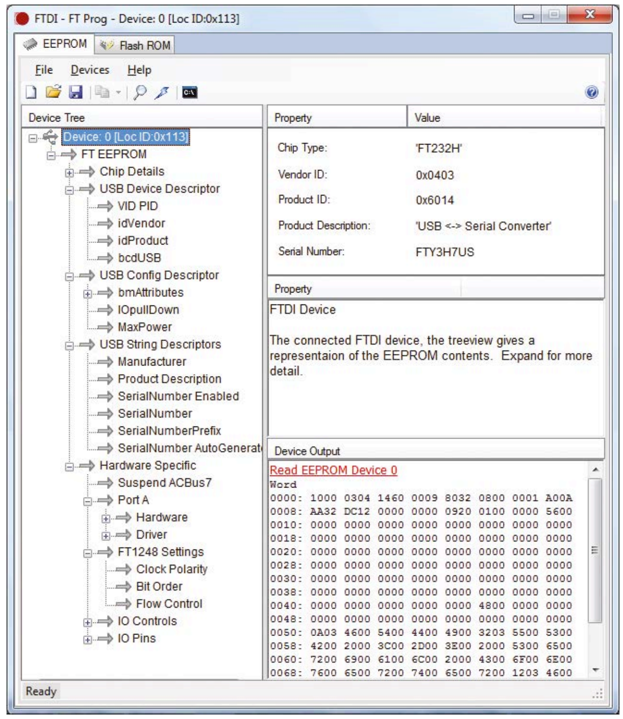


Figure 2. Drinks all round, FT Prog found our FT232H device.

FT PROG

As mentioned, a special tool named FT Prog is available to configure the FT232H (IC1) and to program and/or read its accompanying EEPROM (IC2). You can download it from the FTDI website. Using it is simple: connect the board to your PC, launch FT Prog and click the "Scan and Parse" button (the magnifying glass). If all is well you should see something like **Figure 2** appearing. By expanding the different options you can activate or deactivate certain functions and you can configure pins and interfaces related to the asynchronous part of the chip and the USB driver.
The MPSSE part — like activating and using the I²C or SPI port — is controlled

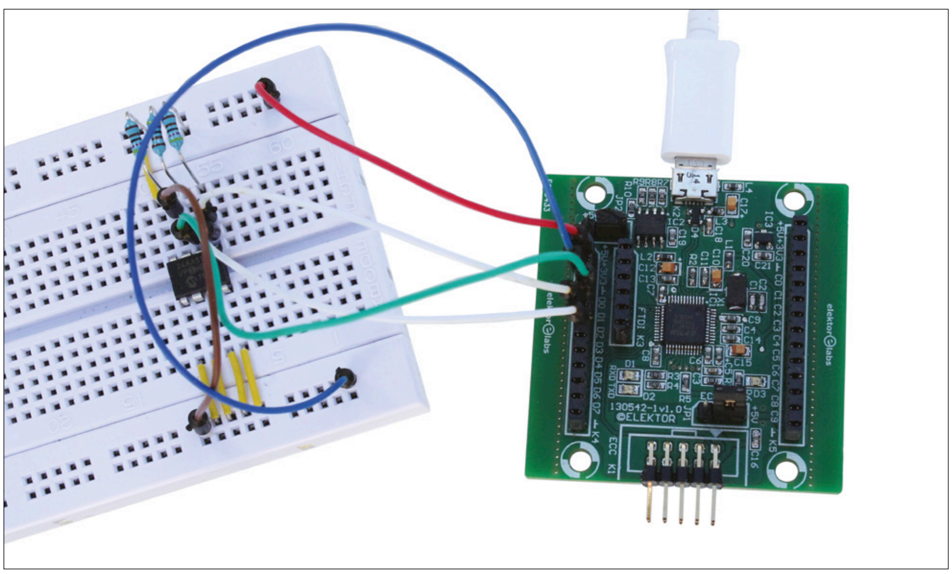


Figure 3. The FT232H board wired up to an I²C EEPROM slapped on a breadboard. Note the two white wires linking both pins AD1 and AD2 to the SDA pin.

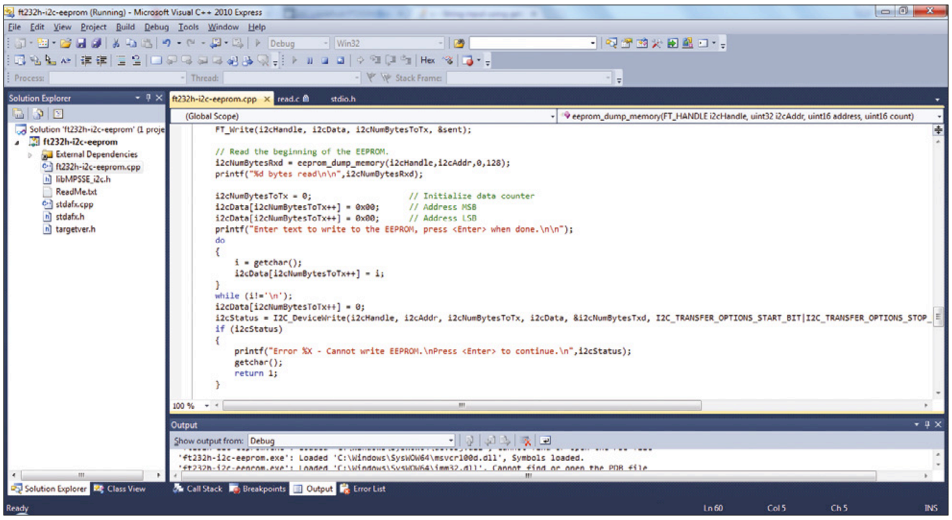


Figure 4. A piece of code showing how to write data to the I²C EEPROM on the breadboard.

Table 1. Jumper JP1	
Short	**Function**
1-2	Powered from K1 (ECC)
2-3	Powered from USB

Table 2 – Jumper JP2	
Short	**Function**
1-2	K3 pin 3 is 5 V
2-3	K3 pin 3 is 3.3 V

Component List

Resistors
Default: SMD 0603, 0.1W
R1,R7,R8,R9 = 10kΩ
R2 = 12kΩ 1%
R3, R4 = 220Ω
R5 = 39kΩ
R6 = 1.5kΩ
R10 = 2.2kΩ

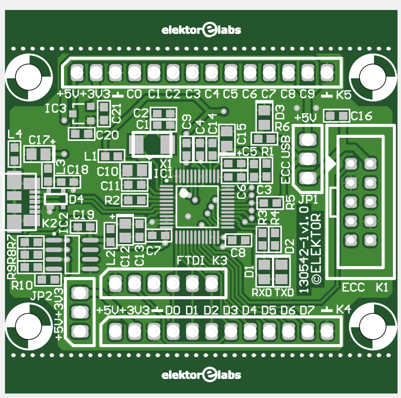
Capacitors
Default: SMD 0603
C1,C2 = 27pF 50V, C0G/NP0
C3,C18 = 10nF 50 V, X7R
C4-C9,C11,C13,C14,C16,C19 = 100nF 50V X7R
C10,C12,C15,C17 = 4.7µF 6.3V, tantalum, SMD Case R (0805)
C20,C21 = 1µF 6.3V, X5R

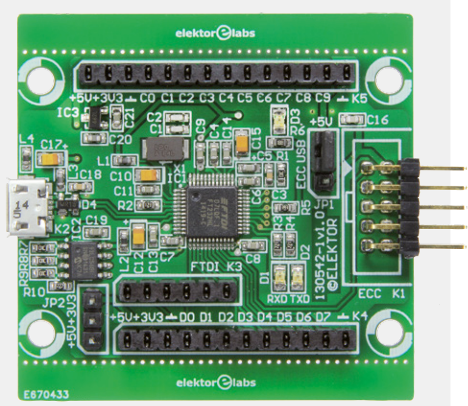
Inductors
L1-L4 = ferrite bead, 600Ω @100MHz, 25%, 0.15Ω @DC, 1.3A (SMD 0603)

Semiconductors
D1,D2,D3 = LED, green (SMD 0805)
D4 = PRTR5V0U2X (SOT-143B)
IC1 = FT232HL (LQFP-48)
IC2 = 93LC56B-I/SN (SOIC-8)
IC3 = XC6206P332MR (SOT-23)

Miscellaneous
K1 = 10-pin DIL (2x5) pinheader, right angle
K2 = Micro USB type-B receptacle, SMD
K3 = 6-way SIL socket, 0.1” pitch
K4 = 12-way SIL socket, 0.1” pitch
K5 = 14-way SIL socket, 0.1” pitch
JP1, JP2 = 3-pin pinheader, 0.1” pitch
JP1, JP2 = jumper, 0.1” pitch
X1 = 12MHz quartz crystal, 18pF C_L, SMD 5 x 3.2 mm

PCB, bare, # 130542-1 v1.0 from Elektor Store
Enclosure, Hammond Manufacturing type 1455C802
Partly assembled board, # 130542-91 from Elektor Store





through software and does not require the use of FT Prog.

The interesting things for the UART can be found in the “Hardware Specific” section. Port A refers to the labels ADBUS*x* and AD*x* in our schematic. You can select the interface you want on this port (defaults to UART and best left at that, though you may have other needs) and the driver that will be loaded by the OS. The VCP driver (default) is the one you need for traditional or legacy serial port applications. Under ‘IO Controls’ you find the port labeled ACBUS*x* and AC*x* in our schematic (Port C). This port is used for control signals like TX enable (for RS-485 and RS-422 interfaces) and LED indicators. You have to be a bit careful here, because the ten pins are not all identical, some have more possibilities than others.

Then, to confuse matters slightly, under ‘IO Pins’ you find both ports A and C but now named as groups AD and AC. Here you can select slew rate, input type and drive current for these pins.

Once you are satisfied with your configuration, you can program the settings in the EEPROM by clicking the ‘Program Devices’ button (the lightning icon). Note that for our serial port experiments, synchronous or asynchronous, you can leave everything in its default position.

Programming examples

As you will have gathered from the previous sections, the FT232H is a relatively complicated IC with lots of possibilities. To make all this power available to the user FTDI have published several application notes [1] together with programming examples (including source code). The most interesting ones are application notes 180, 188 and 190. Note that even though the documents refer to specific FTDI hardware, they will work without modification with the board described in this article. All examples rely on the FTDI D2XX Device Driver.

Application Note 180 describes a rather interesting example in Microsoft Visual Basic 2008 that shows how to take voltage and current measurements by adding two Microchip MCP3201 analog-to-digital (ADC) converters with an SPI interface. It also uses a few general purpose I/O (GPIO) pins for controlling things.

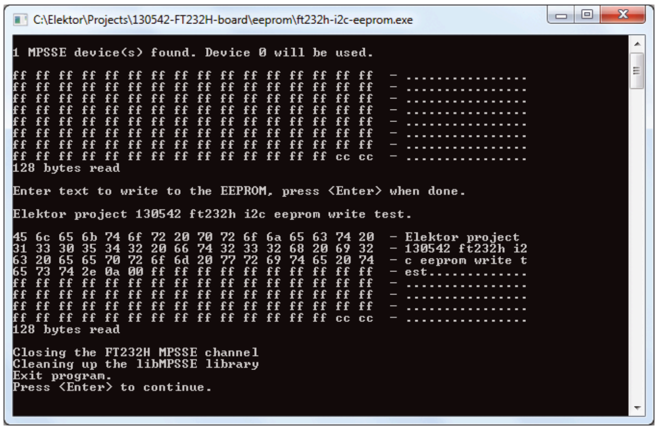


Figure 5. It worked! Data entered by the user successfully written into I²C EEPROM.

Application Note 188 is another SPI example showing how to control a port extender IC MCP23S08 from Microchip, but this time the program is a Microsoft Visual C++ 2010 project. It is a simple console application, written in C and using the libMPSSE-SPI library.
Application Note 190, also written in Microsoft Visual C++ 2010, is an I²C example using a Maxim MAX17061 LED controller. This example relies on the libMPSSE-I2C library. We have modified this example to make it read and write (part of) a 24xx512 I²C EEPROM similar to the one used for instance in the Elektor ADAU1701 Universal Audio DSP board [2]. This example shows that with our FT232H board you can easily make a fast I²C EEPROM programmer (**Figures 3, 4, 5**).

As you will have noticed, the UART, SPI and I²C ports, but also the JTAG port, use the lower pins of the AD port. Except for I²C all these ports have separate TX and RX signals, but I²C needs a bidirectional data line. To achieve this you must tie pins AD1 and AD2 together. AD0 will carry the SCL signal.
These four programming examples show most of what you need to know to make good use of the FT232H board. In case you want to go deeper you may want to read these as well:

- Application Note 135: MPSSE Basics;
- Application Note 108: Command Processor for MPSSE;
- Application Note 129: Hi Speed USB to JTAG Example;
- Application Note 167: FT1248 Dynamic Parallel/Serial Interface Basics;
- D2XX Programmer's Guide.

Enter Python

While researching the stuff we wanted to cover in this article we came across the Adafruit Python GPIO library that includes support for the FT232H. An excellent tutorial on how to use it is available at [3]. There you will also find extensive details on how to install drivers for the FT232H. (Ain't Internet great? Saved us a lot of work ☺). Do note however that you don't want the libusb based drivers presented in this tutorial if you want the Visual Studio examples above to work. In case you did install something other than the official FTDI drivers, you can revert to "normal" operation by uninstalling the non-FTDI driver by ticking the "Delete the driver software for this device" box. Now disconnect and reconnect the FT232H board to make the OS install the FTDI drivers again.

Based on the Python GPIO library we created an example showing GPIO programming. The result is a running-lights effect using 16 LEDs from AD0 to AC7. At this point yet another remark is needed: AC8 and AC9 are not accessible this way. Actually, when you look closely at the datasheet, there is no mode where these pins have a predefined purpose. The only way to make use of them is to give them some function using FT Prog.
One final thing on Python: avoid frustration, use Python 2.7.x (instead of 3.4.x).

Table 3. Connector K1 Pinout (ECC)

Pin	Function
1	TX enable
2	VCC
3	GND
4	Not connected
5	RX
6	TX
7	Not connected
8	GND
9	Not connected
10	Not connected

Table 4. Connector K3 pinout (FTDI cable)

Pin	Function	Color
1	GND	Black
2	CTS (AD3)	Brown
3	VCC	Red
4	TX (AD0)	Orange
5	RX (AD1)	Yellow
6	RTS (AD2)	Green

A long history

Elektor has a long history in terms of USB-to-Serial port converters. In 2008 we started selling the now ubiquitous FTDI USB/TTL Serial cables [4] that expose a 6-pin socket with the standard serial port signals RTS, RX, TX, 5 V, CTS and 0 V. Depending on the cable these signals have 5-V or 3.3-V voltage levels (but beware, the 5-V supply is always 5 V) that can be connected directly to a microcontroller. With a small adapter board [5] such a cable can be used with RS-232 ports. In 2011 we presented our FT232R USB/Serial Bridge/BoB [6] ("Serial Bob" for friends & one secretary). This is a small PCB based on the same chip as used in the FTDI cables, but giving access to all of the chip's pins making it more versatile than a cable. Only a few months ago we published the mighty 4-way RS-232/RS-485 USB-to-Serial converter [7] and we now have USB converter solutions for everything from single 3.3 V up to multiple RS-485. All these products are based on the FTDI FT232R chip or a derivative.

Figure 6. The FT232H board slides into a Hammond aluminum enclosure with K1 sticking out the other end.

Finished tool

Because the FT232H board is such a useful tool, we decided to make the PCB a bit larger allowing it to fit in a standard enclosure (**Figure 6**). We chose an aluminum enclosure from Hammond that is rather long, but you can easily cut it to the right length. The FT232H board simply slides in this enclosure, giving you a sturdy USB-to-Serial converter. If you think the PCB is too large, you can cut it off at the dotted lines.

(130542)

Web Links and Literature

[1] FT232H: www.ftdichip.com/Products/ICs/FT232H.htm

[2] ADAU1701 Universal Audio DSP Board, Elektor Magazine January & February 2014, www.elektor-magazine.com/130232

[3] Adafruit FT232H Python GPIO library: https://learn.adafruit.com/adafruit-ft232h-breakout/

[4] These cables r still on sale: refs. 080213-71 (5 V version); 080213-72 (3.3 V version) on www.elektor.com

[5] USB/TTL serial cable RS-232 extension, ref. 080470-1 on www.elektor.com

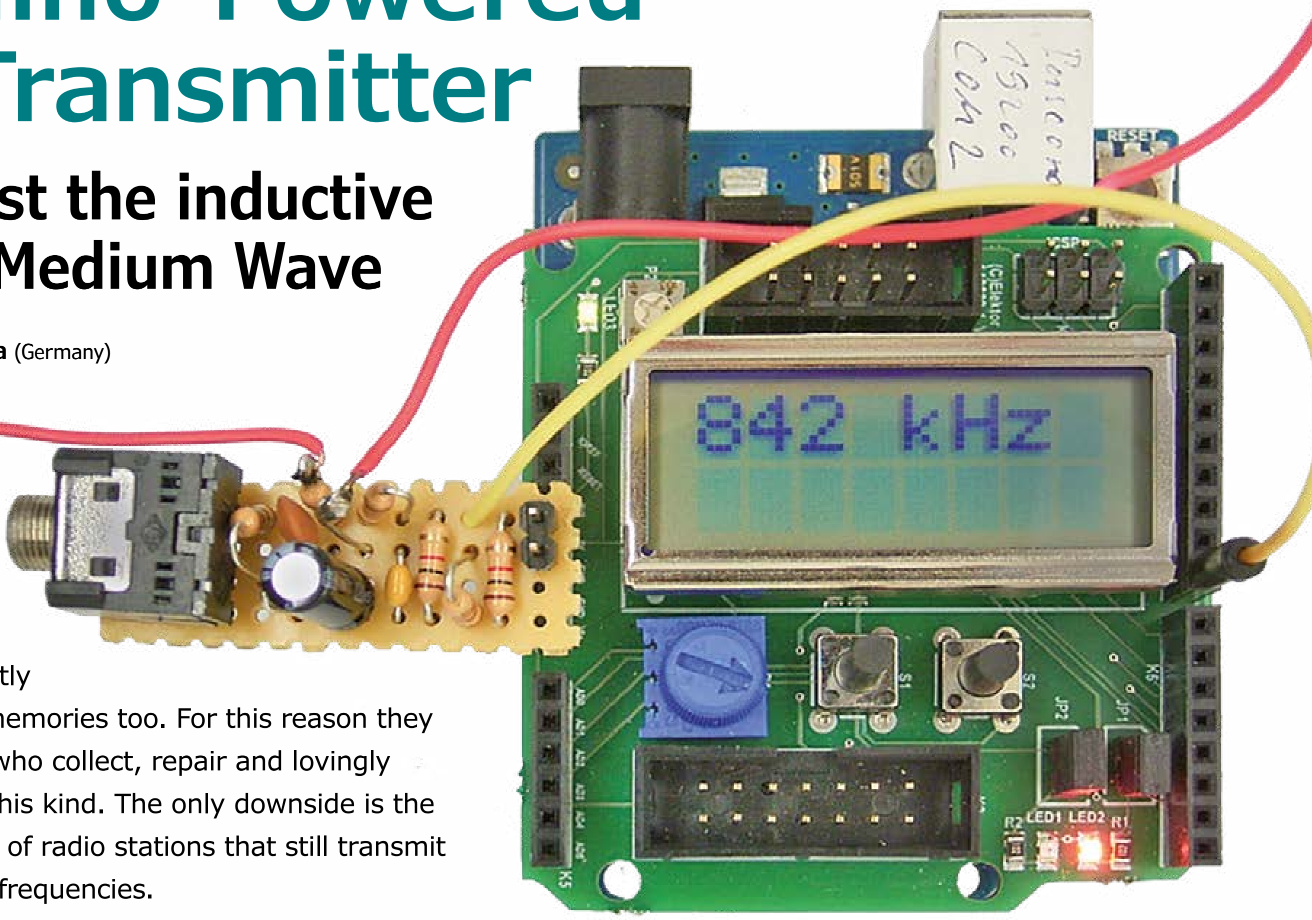
[6] FT232R USB/Serial Bridge/BoB, Elektor Magazine September 2011, www.elektor-magazine.com/110553. Sales item 110553-91 on www.elektor.com

[7] USB Hub feat. RS-232/422/485, Elektor Magazine November 2014, www.elektor-magazine.com/140033. Sales item 140033-91 on www.elektor.com

Arduino-Powered AM Transmitter

Broadcast the inductive way on Medium Wave

By **Burkhard Kainka** (Germany)



Old tube radios have a very special charm. They look handsome, sound great and frequently arouse pleasant memories too. For this reason they have many fans, who collect, repair and lovingly restore radios of this kind. The only downside is the dwindling number of radio stations that still transmit on the classic AM frequencies.

Beside me on my workbench is an old tube radio dating back to 1957, which I still switch on regularly. And yes, on the medium wave bands. It's true that amplitude modulation on the medium wave makes do with a restricted bandwidth, but this gives reception a uniquely warm sound. And in the evening, one can pull in countless stations from across the borders. The only snag is that there are fewer and fewer of them these days. Many local stations have already closed down altogether and even the BBC has abandoned medium wave transmissions. This calls for a new use of these now-vacant frequencies. Our own private AM station is what we need! And that's just what this article describes, enabling me to receive good old BBC by Internet radio and rebroadcast it on medium wave, just as I listened in the olden days.

> Regrettably there are fewer and fewer broadcast stations in the classic AM bands

But is this lawful? Pirate transmitters are not favored generally but there is a perfectly legal solution. You need to broadcast the signal inductively (just as some college and hospital broadcasting systems do) over a restricted range. In this way there is no risk of upsetting your neighbors. This is permitted in my country under the 'General allocation of frequencies in the range 9 kHz to 30,000 kHz for inductive radio applications' (Vfg. 4/2010 amended by Vfg. 4/2014); other countries have comparable allocations. For the frequency range 148.50 kHz to 5,000 kHz you are confined to a limit of 15 dBµA/m at a distance of 10 meters (30 feet).

Magnetic field strength is admittedly not easy to measure but you can calculate it from the antenna current flowing in an inductive loop. If you use a wire loop of radius r = 1 m (i.e. 2 m diameter) with just one turn and restrict the RF current to 0.3 mA, then the magnetic field strength at a distance of 10 meters will amount to only −15 dBµA/m. This will keep everything legal. In practice your 'transmit antenna' will be fixed to the underside of the table or hung on the wall close to the radio. Inside the wire loop reception will be good but outside this the field strength drops off rapidly. You could also create two coils from the same length of wire, concentrating the field strength inside a smaller area, with even faster fall-off at distance.

So we're going to construct an MW AM transmitter, but how? A transmitting tube is driven hard by the carrier signal (Class C final stage), with its anode voltage modulated by the audio signal. For flea power we can do the same thing with a transistor, either bipolar or FET. All we need then is a crystal-controlled RF signal on the appropriate frequency.

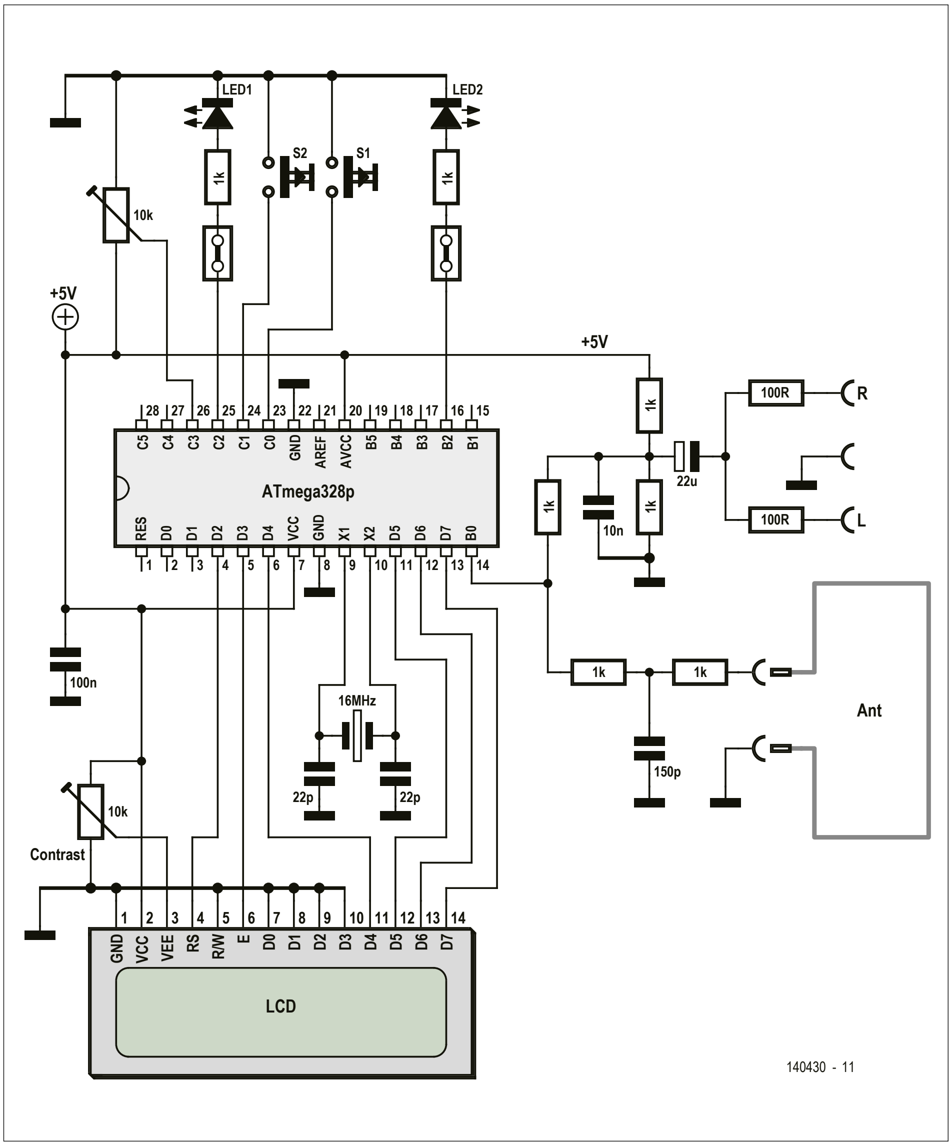


Figure 1. Circuit of this AM transmitter. All components on the left-hand side are located on the Arduino Uno and the Elektor Extension Shield.

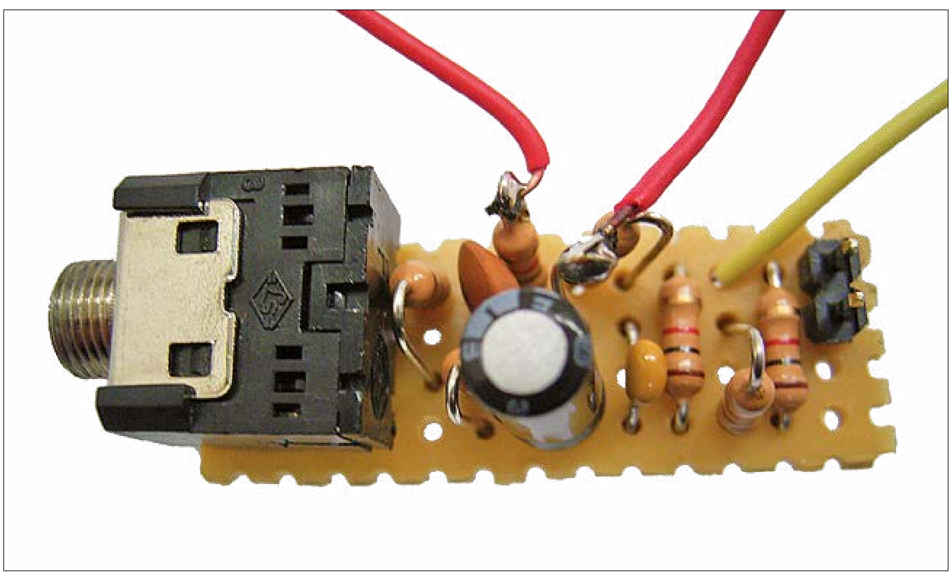


Figure 2. These additional components are located on a small piece of perf board.

Microcontroller RF source

Bring on the microcontroller. The Arduino Uno has a built-in crystal that we can use to derive frequencies in the medium wave band by programmed division factors. This elegant and straightforward solution has the minor disadvantage that we cannot transmit in the 10 kHz spacing normal in the USA (9 kHz in other territories). Despite this you'll have no difficulty in finding a suitable unoccupied frequency. A small control module with a frequency display is easy to produce with the Elektor Extension Shield [1].

Even better, the Uno already has a transmitter final stage on board! By using a little trickery, any Port pin of your choice can be used for this. Instead of addressing the Port register we use the Data Direction register. Take, for example, the PB0 connector. With PORTB.0 = 0 and DDRB.0 = 1 (output) this puts in low-ohmic (Low) state. If we now switch DDRB.0 = 0 (normally you do this when you wish to read in digital signals), the Port is set High. These two states are equivalent to a FET with an open drain. We can now apply any 'drain' voltage of our choice to PB0 (so long as the voltage is within the range 0 V to 5 V). When DDRB.0 = 1, it is grounded, in sync with the phase of the RF signal. Naturally this drain voltage can be modulated, for example with an audio signal.

Now we have everything necessary! We just need to sort out a couple of little details and do a bit of programming.

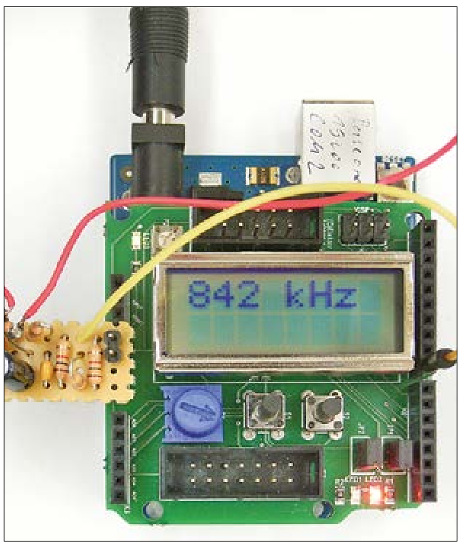
Circuit

The headphone output of a typical signal source (CD player, PC sound card, etc.) normally produces up to 1 Vrms, which is almost 3 Vpp. It definitely does not want to be much more than this, because the final stage must be driven within the range 0 to 5 V. This makes a modulation amplifier unnecessary. A couple of resistors and capacitors suffice to 'condition' the audio signal (Figure 1, right-hand side). The left-hand side of the schematic shows the ATmega328 of the Arduino Uno and the components located on the Extension Shield. You could also construct the whole affair on an experimenter's breadboard of course.

At the antenna output we have a simple low-pass filter for suppressing the harmonics. Theoretical considerations and actual measurements both indicate somewhat less than 0.3 mA of current should

**Listing 1.
Producing the RF signal.**

```
D11:
Do                 ,/11, 1454 kHz
  Ddrb.0 = 1
  nop              ,+4 nop
  nop
  nop
  Ddrb.0 = 0
  nop
Loop

D10:
Do                 ,/10, 1,6 MHz
  Ddrb.0 = 1
  nop              ,+3 nop
  nop
  nop
  Ddrb.0 = 0
Loop
```



flow in the loop antenna. You will be well within the guidelines so long as the total length of wire in the loop antenna is not greater than 3 meters. For your first tests a far smaller wire loop will be fine (using just 20 cm of wire for example).
The output from a regular sound card (about 3 Vpp) will give us a modulation factor of around 50%, which conveniently rules out the risk of overmodulation. As AM signals are transmitted only in mono, we combine the left and right-hand channels. This will produce absolutely clean modulation without any kind of distortion. When heard on a tube radio you will enjoy the agreeable, warm sound of the old days.

Software

To generate the carrier frequency we simply toggle the corresponding DDRB bit on and off fairly rapidly. To perform the simple Do Loop (Listing 1) programmed in Bascom, the ATmega328 takes seven clock cycles. Add three NOPs and we have then divided the clock frequency of 16 MHz by 10. This means we can transmit on the upper end of the medium wave at 1.6 MHz.
If you add in another NOP command, the loop will divide by 11, producing 1454 kHz. And so on. In our program, which as always can be downloaded from the Elektor Magazine website [2], every frequency gets its own loop, each with a label stating the relevant division factor. The lowest frequency of 500 kHz is produced using D32 (16 MHz divided by 32). Incidentally, the NOPs are arranged so that the pulse/pause ratio is as close as possible to equal. By its very nature a symmetrical square wave produces the lowest number of harmonics.
A total of 22 different medium wave frequencies can be selected. At the same time the frequencies at the lower end (500 kHz, 516 kHz, 533 kHz, etc.) are closer together than at the upper end.

Operation

Once in the (RF) loop there is no going back. Of course you could come up with some solution involving an Interrupt but the following operating procedure is simpler (Listing 2). First you use the pot to select a suitable frequency, confirmed by the display. Then you press button S1 to switch on the transmitter. This also lights up the LED2, indicating 'on the air'. At all times the display shows the last frequency selected. Any operation of the pot will have no action now and the frequency remains unaltered. This is the same as with the high-power tube transmitters of the olden days, on which you also could not just change frequency once up and running. If you nevertheless wish to alter the frequency, you need to press Reset first, which stops the transmitter. Now you can set a new frequency and press S1 to activate this.

(140430)

Web Links

[1] www.elektor-magazine.com/140009

[2] www.elektor-magazine.com/140430

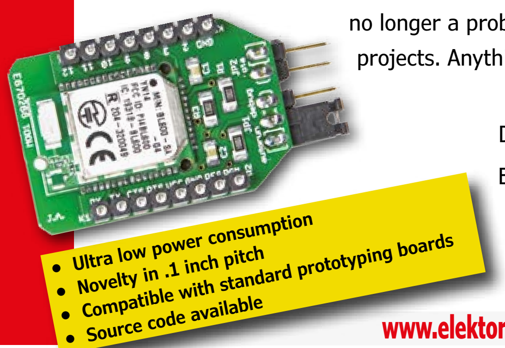
Listing 2. Selecting the transmit frequency.

```
Do
  U = Getadc(3)
  D = U / 46
  D = D + 10
  F = 16000 / D
  Locate 1 , 1
  Lcd F
  Lcd „ kHz „
  Waitms 500
  If S1 = 0 Then
    Led2 = 1
    If D = 10 Then Goto D10
    If D = 11 Then Goto D11
    If D = 12 Then Goto D12
    If D = 13 Then Goto D13
    If D = 14 Then Goto D14
    If D = 15 Then Goto D15
    If D = 16 Then Goto D16
    If D = 17 Then Goto D17
    If D = 18 Then Goto D18
    If D = 19 Then Goto D19
    If D = 20 Then Goto D20
    If D = 21 Then Goto D21
    If D = 22 Then Goto D22
    If D = 23 Then Goto D23
    If D = 24 Then Goto D24
    If D = 25 Then Goto D25
    If D = 26 Then Goto D26
    If D = 27 Then Goto D27
    If D = 28 Then Goto D28
    If D = 29 Then Goto D29
    If D = 30 Then Goto D30
    If D = 31 Then Goto D31
    If D = 32 Then Goto D32
  End If
Loop
```

welcome in your ONLINE STORE

EDITOR'S CHOICE

In our 2/2015 edition, Jennifer Aubinais presented her outdoor thermometer, communicating wirelessly with iOS and Android smartphones. Her design made use of Laird Technologies' BL600 Bluetooth module. This tiny part is a veritable miniature computer, with a very economic radio transceiver incorporated. The design took up a lot of Jennifer's free time — having gone through the abundant documentation from the manufacturer, she contacted them directly to clear up some points. Jennifer is passing her hard-earned knowledge on to you in an article series on programming the BL600. Yes, this module offers easy programming with its smartBASIC language, specially adapted to the needs of the IoT. Thanks to the Elektor BL600 e-BOB, this tiny module is no longer a problem to use on your breadboard or personal projects. Anything to please!



Denis Meyer

Editor-in-Chief, Elektor French Edition

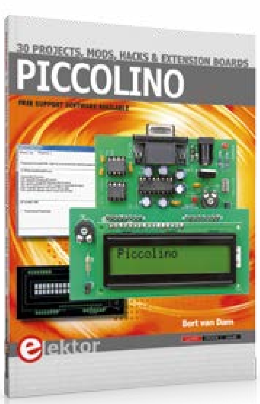
www.elektor.com/ebob-bl600

Elektor Bestsellers

1. DVD Elektor 2014
   www.elektor.com/dvd-2014

2. DVD Elektor 1980 through 1989
   www.elektor.com/eighties
3. Mastering Microcontrollers
   www.elektor.com/mm-revised
4. GREEN Membership
   www.elektor.com/green
5. BL600 e-BoB
   www.elektor.com/e-bob-bl600
6. Raspberry Pi 2 Model B
   www.elektor.com/rpi-2
7. RPi Advanced Programming
   www.elektor.com/rpi-book
8. EveryCircuit App
   www.elektor.com/everycircuit

Piccolino

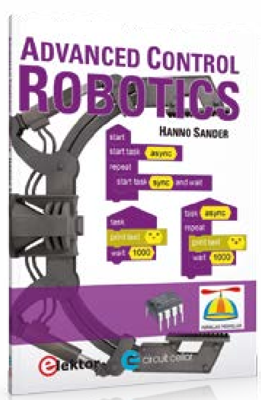


The Piccolino rapid development board can be used to quickly design microcontroller circuits. The Piccolino has a fast 16f887 PIC microcontroller, voltage regulator, and communications module, and can be easily extended using its four headers. This book contains 30 projects based on the Piccolino. The clear descriptions along with circuit diagrams and photos will make the building of all the projects an enjoyable experience!

member price: £31.95 • €35.96 • US $49.00

www.elektor.com/piccolino-book

Advanced Control Robotics



Hanno Sander's Advanced Control Robotics simplifies the theory and best practices of advanced robot technologies. You're taught basic embedded design theory and presented handy code samples, essential schematics, and valuable design tips (from construction to debugging). The book is intended to help roboticists of various skill levels take their designs to the next level with microcontrollers and the know-how to implement them effectively.

member price: £31.95 • €35.96 • US $49.00

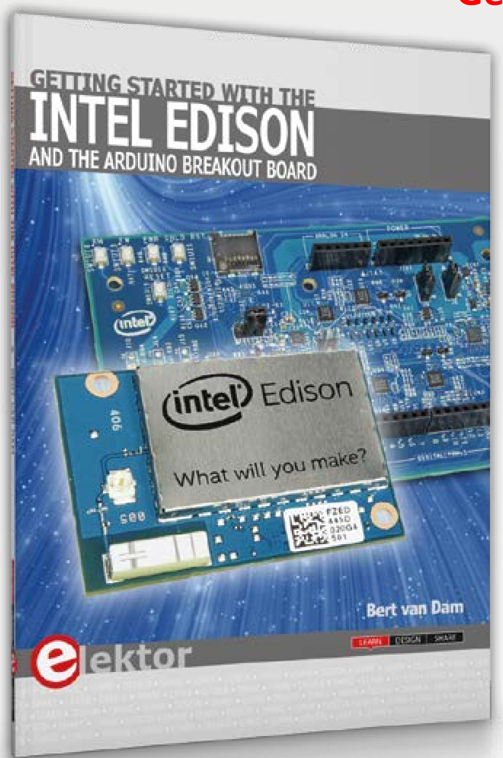
www.elektor.com/robotics-book

Raspberry Pi 2 Model B

Raspberry Pi 2 (Model B) is a low cost credit-card sized computer, based on the Broadcom BCM2836 ARM Cortex-A7 quad-core processor running at 900 MHz. This means it runs approximately 6x faster than the original Raspberry Pi model B and B+. The RAM has also been upgraded from originally 512 MB to 1 GB LPDDR2 SDRAM. The 'RPi 2' is still backwards compatible with its former models.

member price: £33.95 • €38.66 • US $53.00

www.elektor.com/rpi-2



Getting Started with the Intel Edison and the Arduino Breakout Board

This book aims to help you get up to speed with the Edison, a tiny computer with a lot of power and built-in wireless communication capabilities. The Edison Arduino break-out board will be used because it is easy to work with.

We will discuss Linux, Arduino C++ and Python, and show examples of how the Edison can interface with other hardware. Wi-Fi and Bluetooth will be utilized to set up wireless connections, and show you a trick to program sketches over Wi-Fi. Once you have completed this book, your Edison will be up and running with the latest software version, and you will have sufficient knowledge of both hardware and software to start making your own applications.

MEMBER PRICE: £23.95 • €26.96 • US $37.00

www.elektor.com/edison-book

New Book on Intel Edison

Limited time offer for GREEN and GOLD members:
15% Discount plus free shipping!

NOSTALGIA

All articles published in Elektor magazine's year volumes 1980 through 1989!

USB-to-Multi-Protocol Serial Converter

For UART, SPI and I²C

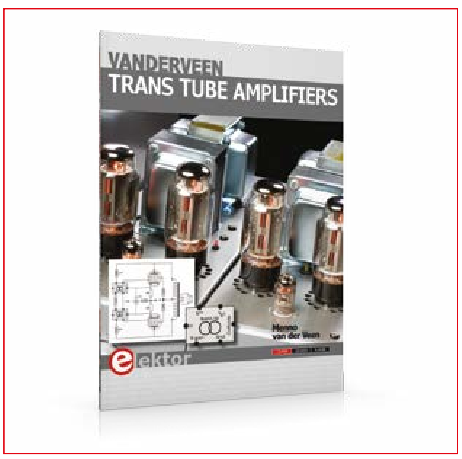
iFixit 54 Bit Driver Kit

The key to getting inside modern hardware! This is the set that spawned the repair revolution. This handy toolkit contains all the screwdriver you need to repair electronics, smartphones, tablets, computers and game consoles. The comprehensive standard in precision screwdriver bit sets, features 54 carefully selected bits. When it comes to precision bit sets, there is no better kit!

member price: £15.95 • €17.96 • US $25.00

www.elektor.com/54-bit-driver-kit

Vanderveen Trans Tube Amplifiers

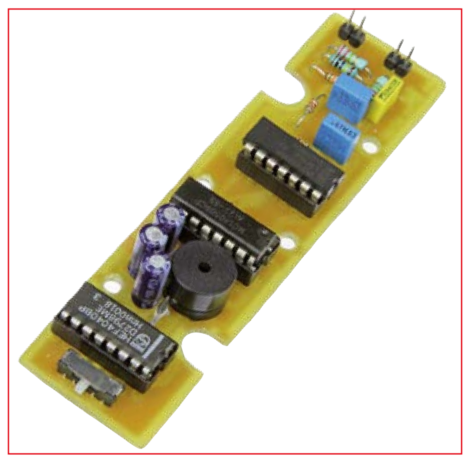


In this book Menno van der Veen describes one of his research projects focusing on the question of whether full compensation for distortion in tubes and output transformers is possible. A variety of methods have been developed with the aim of attaining this goal. One of them has largely been forgotten: transconductance, which means converting current into voltage or voltage into current. Menno van der Veen has breathed new life into this method.

member price: £23.95 • €26.96 • US $37.00

www.elektor.com/transie

Beep Logic Probe



Even if you have an upmarket logic analyzer and are completely at ease with its operation, you are unlikely to use it to debug simple circuits like Arduino and Raspberry Pi add-on boards. That's where 'Beep' comes in. 'Beep', as we nicknamed our logic probe, is a simple tool that employs sounds to indicate signal levels so you can use it without having to look up from your work. Constructing Beep should not be overly complicated. A kit of parts is now available.

member price: £11.95 • €13.46 • US $19.00

www.elektor.com/beep

REVIEW OF THE MONTH

BY NIGEL SMITH

Elektor's Arduino Extension Shield is arguably an essential accessory for any Arduino owner. It addresses the only major shortcoming of having an Arduino on its own: being limited a single peripheral LED. It provides excellent functionality in a small tidy convenient package, instantly providing a display allowing anything to be viewed, such as measured parameters, menu options, etc. The two input switches and the potentiometer allow you to easily communicate back with your program that is running on the Arduino. In addition, the shield features two communication headers allowing other peripherals to be interfaced, which I look forward to adding soon. The shield is well designed; there are no components which protrude and interfere with the Arduino. The PCB is well laid out, the soldering is excellent — with no dry joints. The product was well packaged and arrived with no bent pins. It took only minutes to mount and get information displayed using the Arduino IDE, only having to change the output pins in the LCD examples to the ones used on the shield to communicate with the display. This is a brilliant device for anyone wishing to get started in Arduino microcontroller programming, which I highly recommend.

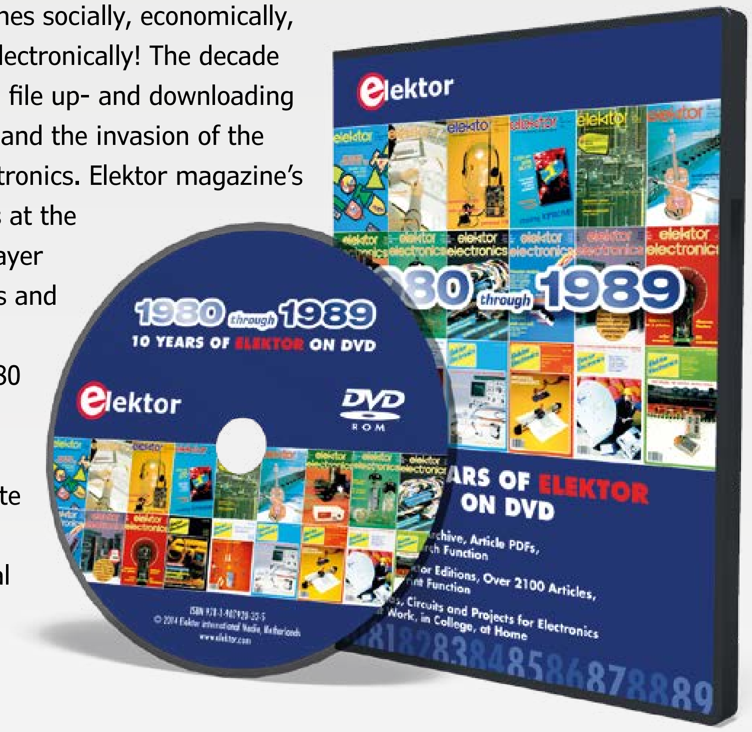
Read this review and more about this product at
www.elektor.com/arduinoshield

Submit a review of your favorite Elektor product and qualify for winning a €100 voucher for redeeming in the Elektor Store.

For further information, please visit www.elektor.com/rotm

DVD Elektor 1980 through 1989

The 1980s were turbulent times socially, economically, politically, musically, and… electronically! The decade saw the introduction of SMD, file up- and downloading over private telephone lines, and the invasion of the microcontroller into DIY electronics. Elektor magazine's English language edition was at the forefront of it all! This dual-layer DVD-ROM contains all circuits and projects published in Elektor magazine's year volumes 1980 through 1989. The 2100+ Elektor 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.



MEMBER PRICE: £52.95 • €62.10 • US $82.00

www.elektor.com/eighties

New Book on Intel Edison

Limited time offer for GREEN and GOLD members:
15% Discount plus free shipping!

NOSTALGIA

All articles published in Elektor magazine's year volumes 1980 through 1989!

USB-to-Multi-Protocol Serial Converter

For UART, SPI and I²C

The Bottle Builder

This massive compendium of tube amplifier designs presents an impressive gallery of tube amplifier projects described not just with a personal stance, but also humor, an open mind, good anecdotes, and an equally good emphasis on all the technical aspects of design and implementation. With every project, the focus is on what makes a particular amplifier "special" in terms of design or performance, which as we all know are not necessarily in agreement.

member price: £62.95 • €71.96 • US $98.00

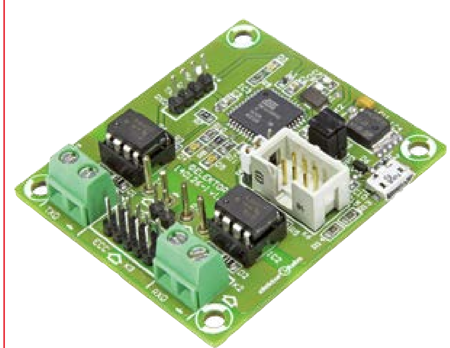
www.elektor.com/bottle-builder

Mystery Product

We dare you ...

www.elektor.com/mystery

RS232 Data Logger



An ATXmega128A4U-based RS-232 data logger and Spy tool with two optocoupler-isolated inputs supporting up to 115.2 kbaud. Input current as low as 1mA, JFET input withstands 30V max. Three of five Xmega internal UARTs are used (2 x data acquisition, 1 x PC data transfer). ASCII- and HEX-format data saving and analysis using TeraTerm with FT232 linking to the PC. 23-bit timestamping interval range is 1 ms to 2.3 h, with SRAM capacity for up to 900 samples.

member price: £23.95 • €26.96 • US $37.00

www.elektor.com/rs232-data-logger

THIS MONTH'S SHOPPING LIST

P-W-M Doorbell
PCB: 140256-1
Progr. Microcontroller: 140256-41

NFC 'e-BoB' Gateway
PCB: 140177-1
Progr. Microcontroller: 140177-41
Module: 140177-92

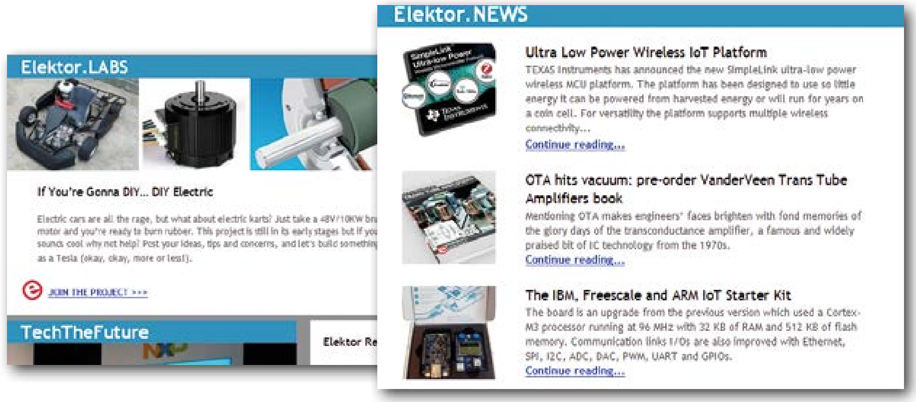
Basement Ventilation System (1)
PCB: 100892-1
Progr. Microcontroller: 130320-41
CC2-eBoB : 140154-91

Basement Ventilation System (2)
PCB: 140433-1

BL600-eBoB (2)
PCB: 140270-1
Module: 140270-91

Get it all @ www.elektor.com/out-now

elektor•post : DON'T MISS OUT!

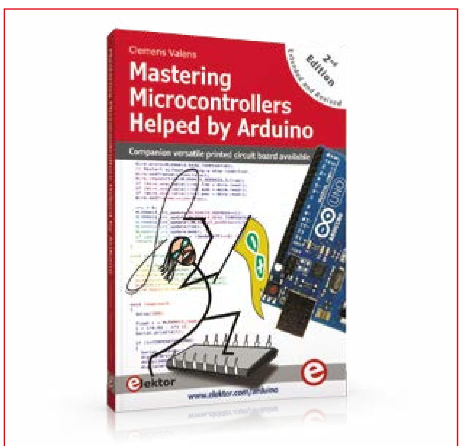


Over 100,000 people worldwide already receive our free Elektor.POST newsletter in their inbox every Friday. It is packed full of the latest news items, tips and trends from the world of electronics. If you do not already have a .POST membership, sign up now and receive these extras:

- **An exclusive free Elektor project worth €2.50 (£2.95 / US $4.00) every second week**
- **Special offers in the Elektor Store**
- **A free Ebooks Inspiration Bundle valued at €32.95 (£28.95 / US $45.00)**

REGISTER TODAY, IT'S FREE
www.elektor.com/newsletter

Mastering Microcontrollers Helped by Arduino



This second, extended and revised edition of Mastering Microcontrollers Helped by Arduino will not only familiarize you with the world of Arduino but it will also teach you how to program microcontrollers in general. In this book theory is put into practice on an Arduino board using the Arduino programming environment. The implementation of the presented concepts is simple and fun. This book will be your first book about microcontrollers with a happy ending!

member price: £33.95 • €38.66 • US $53.00

www.elektor.com/mm-revised

Neopixel Digital RGB LED Strip

With this impressive NeoPixel Digital RGB LED Strip with a massive 60 LEDs per meter you're able to control each LED individually. Yes, that's right, this is the digitally-addressable type of LED strip! To get high density, the controller chip is inside the LED. This means that the chip only uses a single pin for input and a single pin for output. The protocol used is very timing-specific and can only be controlled by microcontrollers with highly repeatable 100nS timing precision.

member price: £19.95 • €22.46 • US $31.00

www.elektor.com/led-strip

Microcontroller Based Radio Telemetry Projects

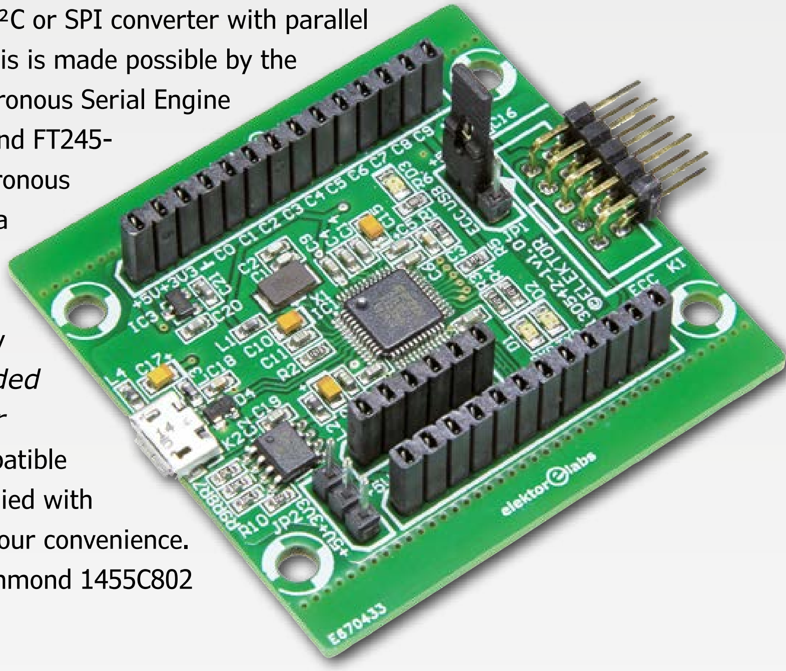
This book is written for people who want to learn more about radio telemetry applications and microcontroller programming using the PIC18F series of microcontrollers. The design of a radio telemetry based mini weather station is considered as an example system in the book where the developed system can measure the temperature, humidity, atmospheric pressure, carbon monoxide level, nitrogen dioxide concentration, air quality level, wind direction wind speed and more.

member price: £31.95 • €35.96 • US $49.00

www.elektor.com/radio-telemetry

USB-to-Multi-Protocol Serial Converter

Compared to the familiar FT232R, the FT232H chip used in this project results in a High Speed device (480 Mbit/s) with a High-Speed USB to Multipurpose UART/FIFO. Consequently this project is not just a UART, but also a I²C or SPI converter with parallel port functionality added. All this is made possible by the FT232H's Multi-Protocol Synchronous Serial Engine (MPSSE), supporting FT1248 and FT245-style synchronous and asynchronous FIFO modes. The converter is a break-out board (BoB), with all pins of interest brought out to connectors. Connectivity includes the *Elektor Embedded Communication Connector* (ECC), and an FTDI cable compatible connection. The board is supplied with SMD parts preassembled for your convenience. It was designed to fit in a Hammond 1455C802 aluminum enclosure.



MEMBER PRICE: £25.95 • €29.25 • US $40.00

www.elektor.com/130542-91

New Book on Intel Edison

Limited time offer for GREEN and GOLD members:
15% Discount plus free shipping!

NOSTALGIA

All articles published in Elektor magazine's year volumes 1980 through 1989!

USB-to-Multi-Protocol Serial Converter

For UART, SPI and I²C

Welcome to the SHARE section

By **Jaime González-Arintero**
jaime.glez.arintero@eimworld.com

The End Was Nigh

I was 14 when that so-called *digital apocalypse* was imminent: the Year 2000 problem. To cut a long story short, as we all know it was the consequence of using only two digits to represent four-digit year numbers, which was obviously simpler in several systems. In the 1960s, those digits were expensive!

We saw some funny effects, like the possibility to book train tickets for backward time travels to the Victorian Era (i.e. 1900!). Others less funny, namely a general mess in some databases that caused more than a headache. But hey! Not just that. Apparently, all the viruses of the world were supposed to go active at the same time, on January 1, 2000, plunging us back into the Stone Age. That's what most ordinary people thought back then. Believe it or not, I remember hearing people asking in those small local electronics shops in Madrid if "that watch" was ready for the millennium bug, or if "their 1994 electronic typewriter" would eventually self-detonate during New Year's Eve. I'm serious: This happened.

After the first official communications explaining what the whole thing was about, the subject sky rocketed, and suddenly anything electronic was becoming *ready for the Y2K problem*. Eye-catching stickers appeared on almost every software package (back then, boxes you could touch), together with TV spots, of course, you could also purchase all sorts of antivirus' suites that would magically prevent Armageddon. I also remember spotting ads of local IT service companies, with "consultants" addressing small businesses to get their computers, printers (?), phones (??) and alarm systems (???) *ready for the Y2K problem*. Several of these "IT consultants" squeezed money out of small local businesses managed by technology-uneducated people afraid of losing their customer databases and profiles, or simply to prevent any accounting mess. Once again, believe it or not, that happened.

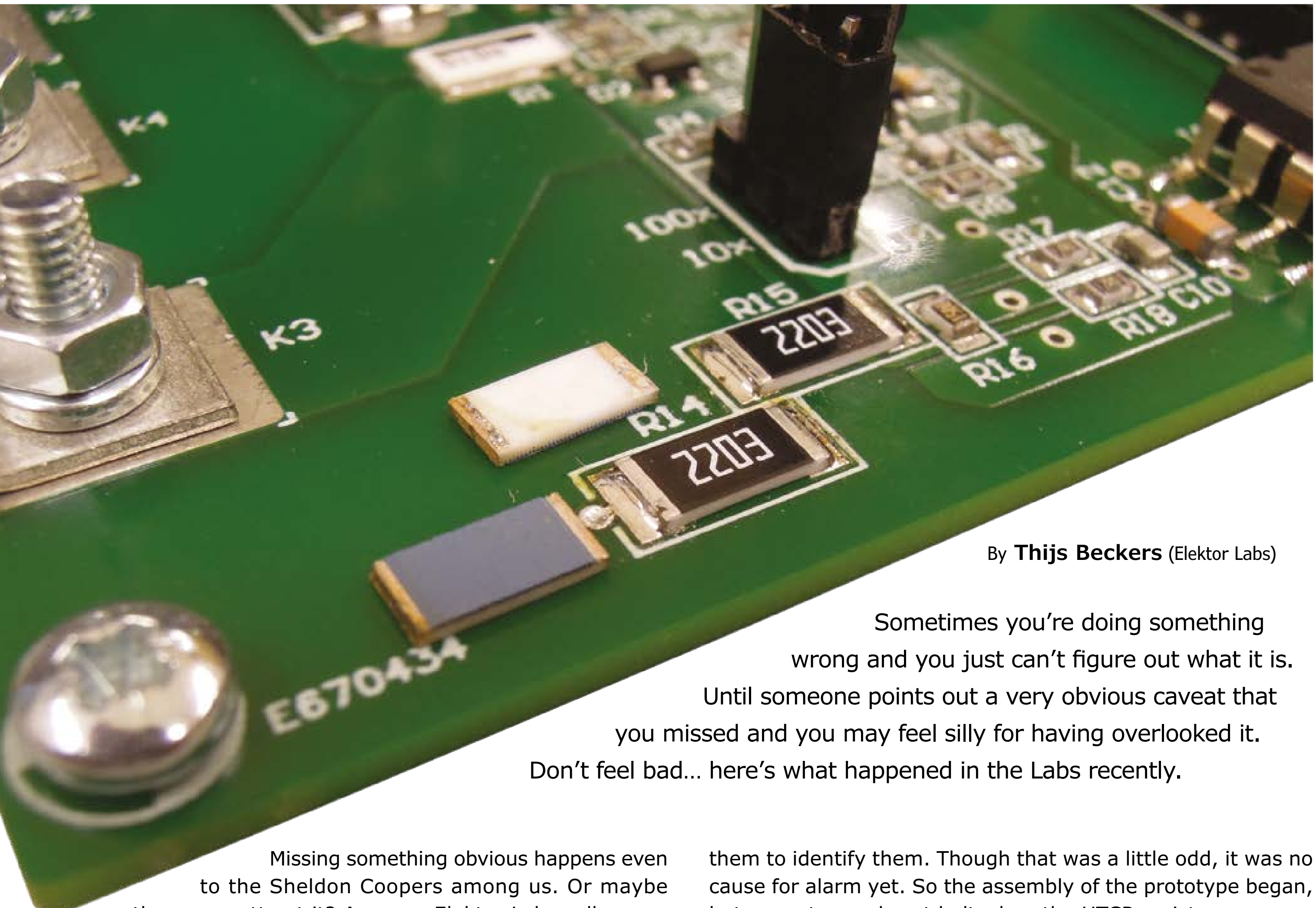
Okay, I hope these were just odd examples. To throw some light onto the subject, we can have a look at the figures. It is calculated that the total worldwide investment to get ready for such a *catastrophic event* came to more than 210 billion euros, while the total potential loss in the worst case scenario would have been around 160 billion euros. Well, exaggerated or not, it seems that digital misunderstandings are potentially lucrative for some people. And it's also true that engineers don't usually solve problems until seeing the writing on the wall. Have you ever been in a similar *digital apocalypse* that went out "not with a bang but a whimper"? Dare to SHARE!

> "Never come till you have been called three or four times; for none but dogs will come at the first whistle." - Jonathan Swift, *Directions to Servants*, 1731.

Ticket to the Past

So you lost your chance of purchasing a train ticket back to the 20th century? Worry not! You can still wait until 2038 to get your time travel pass back to 1901. The widely reported Year 2038 problem (Y2K38) affects Unix systems in which time is stored as a signed 32-bit integer. Consequently, as of 19 January 2038, an integer overflow will cause these systems to ~~catastrophically explode~~ display wrong dates, as a result of incrementing the sign bit as well. It is worth noting that iPhones as well as Android devices do not allow the user to set a date past 2038, while Windows Phone does. That's a declaration of intentions. Just sayin'. (150138)

Unsolderable



By **Thijs Beckers** (Elektor Labs)

Sometimes you're doing something wrong and you just can't figure out what it is. Until someone points out a very obvious caveat that you missed and you may feel silly for having overlooked it. Don't feel bad... here's what happened in the Labs recently.

Missing something obvious happens even to the Sheldon Coopers among us. Or maybe they even attract it? Anyway, Elektor Labs colleagues were working on this prototype which incorporated a couple of high-temperature chip resistors in SMT case. These were essential for their ability to carry a considerable current and dissipate a fair amount of heat. A couple of TT electronics resistors from the HTCR Series were chosen for the job. At 70 °C (158 °F) the 2512 model is stated to be able to dissipate 1 watt, at 155 °C (311 °F) this figure drops to 0.75 watts and at 230 °C it can still shake off 0.5 watts.

Wait! Come again, 230 °C? Errm, doesn't leaded solder tin melt at 188 °C already, and lead-free solder at around 220 °C? Sure, and that's why this resistor comes with different termination options. In the ordering procedure for these resistors you have to specify HTCR2512**G** for gold planar termination suitable for wire bond and adhesive, ...**P** for PtAg planar (adhesive), ...**EW** for Polymer AG wraparound (adhesive) or ...**F** for Ni barrier & Sn plated wraparound, the last option being the one you want when you intend to use your soldering iron or hot-air rework station. Can you guesstimate what went wrong?

So the components for this project had arrived and among them were the HTCR resistors, which didn't have any printing on them to identify them. Though that was a little odd, it was no cause for alarm yet. So the assembly of the prototype began, but came to an abrupt halt when the HTCR resistors were up for soldering. The solder just didn't stick to the terminals. We first blamed it on the solder, being lead-free, RoHS compliant and generally a pain to work with as it doesn't flow half as well as the trusty old 'prohibited' Sn60Pb40. Its adherence was so poor eyebrows were raised. And of course eventually someone had a close look at the datasheet of the HTCR resistors and noticed the terminal options for these series. Then it became clear very quickly what happened. We had ordered resistors with the wrong terminal version.

We have never used adhesive instead of solder in our prototypes and we aren't planning on doing that soon, so we reordered the resistors and made sure so select the correct termination option this time. In the photo you can see the original resistors next to their replacements. One resistor is turned upside down, so you can see there's no marking on them whatsoever. You can also see there's almost no solder on the termination tabs. That's how reluctant these terminations are to adhering to solder.

(150143)

Lesson learned

Special components need special attention!

What’s hot at Dot Labs

…aside from the soldering iron…

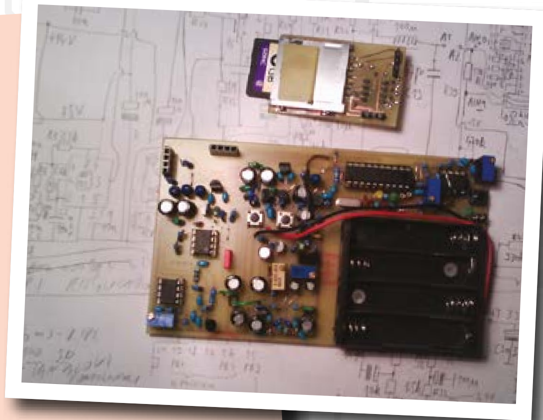
Here is a selection of cool, great, fun and interesting projects that we like. They’re all at www.elektor-labs.com.

Knockin’ on Mozart’s door

Or that’s what your guests will think when they come to your door! Based on a classic PIC16F873 microcontroller, this programmable musical doorbell will entertain your visitors with your custom made tunes while they wait. They may even decline to enter before the melody is finished. To make things even better, the doorbell’s pushbutton (actually a touch sensor) is wireless to allow for frustration-free installation. [http://po.st/doorbell]

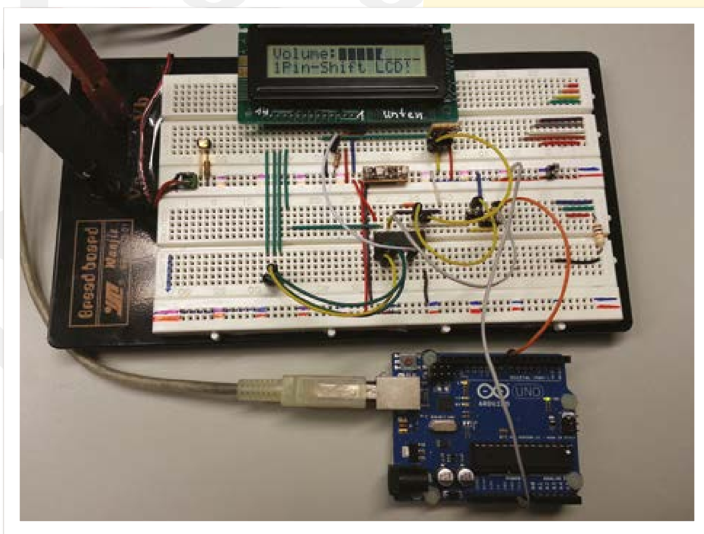
I bet you didn’t see it coming

In electronics, even when things don’t work properly, things usually follow certain (predictable) patterns. And in computer algorithms, there’s no such thing as true randomness (not counting the weird behavior of some programs experiencing memory problems), only pseudo-randomness. This random number generator makes use of the avalanche noise generated by zener diodes. The noise is cleaned and filtered to degrees and then gets stored as a binary sequence on an SD card. Don’t believe it’s truly random? Ask MATLAB! [http://po.st/random]



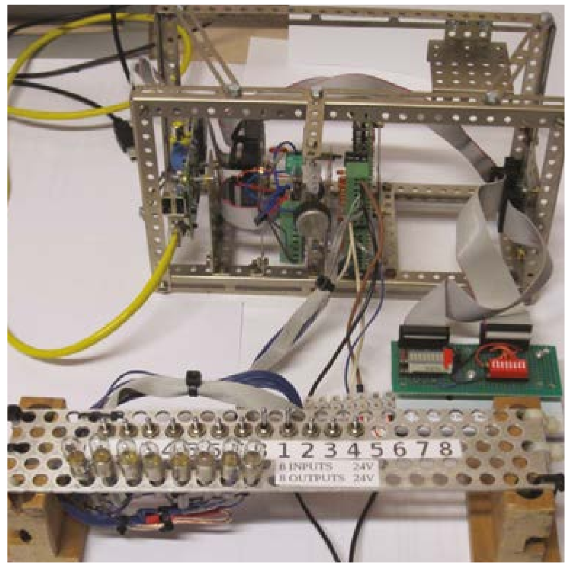
The one

“Darn, looks like I’m running out of I/O pins…” You probably know the feeling, when you suddenly realize you don’t have enough GPIO ports available on the microcontroller to implement that new feature the CTO-oh just requested. The only option is to migrate to a different MCU with more ports or add some kind of serial-to-parallel port expander. This project takes the latter concept a step further by reducing the serial interface to one single wire. Control any standard alphanumerical LCD with just one port pin, a 74HC595 shift register and a few passive components. It might not be as quick as an 8-bit parallel interface, but it is good enough for a good many low-speed applications. [http://po.st/onewire]



PLC PLμX

The PLμX project is a Linux-based PLC with a resident graphics editor. The author has been working on it for quite a while and the project has grown steadily over time. PLμX allows the user to program the system by simply dragging and dropping blocks like you do when using ready-made commercial solutions. Built around GNUBLIN modules, it provides 64 digital inputs, 64 digital outputs and 32 analog inputs: More than enough, we hope!
[http://po.st/PLuX]



Seven segments

For some reason I still find 7-segment LED displays attractive. I can watch the — dimmable — of my alarm clock for hours. Their warm, uniform glow just captivates me (sometimes I even forget to sleep). So when someone decides to revisit the concept in an original manner, that has my attention. The Leditron, inspired by Numitron tubes, is such a project. It combines laser cutting and LED filaments to create large 7-segment digits. An LED filament is what you often get when you buy a modern "incandescent" light bulb, they can also be bought on the internet. (Of course. Why did I say that?)
[http://po.st/leditron]

Sir, the network is down!

This is a bit of a mysterious project. Looking at the project avatar makes you think of a game or some kind of computer from a sixties sci-fi movie, you know, the one that you can talk to and that talks back Muppet Hospital style. The accompanying photograph (reproduced here) is out of focus as if it was taken with a spy camera. According to the description the project is actually intended to monitor the status of your computer network. And it is pretty serious about it too as it can even call you on your mobile when a problem comes up and tell you what is wrong. This device would be really cool if you could answer it with: "So? Why haven't you fixed it yet?"
[http://po.st/PINGcube]

It all started with a big bang

Don't you wonder sometimes about Sound and Vision? Poster Stevie did after a friend asked him if he could add some visual effects to his carnival drum. Stevie went through his junk box and came up with a simple piezo triggered LED flash light. With the piezo stuck on the skin of the drum the LEDs will flash when the drum is hit while being immune to ambient noise. The piezo was scavenged from an alarm clock, the LEDs were collected from one of those ever-lasting low-energy LED light bulbs that, as happens way too often, had died prematurely.
[http://po.st/partydrum]

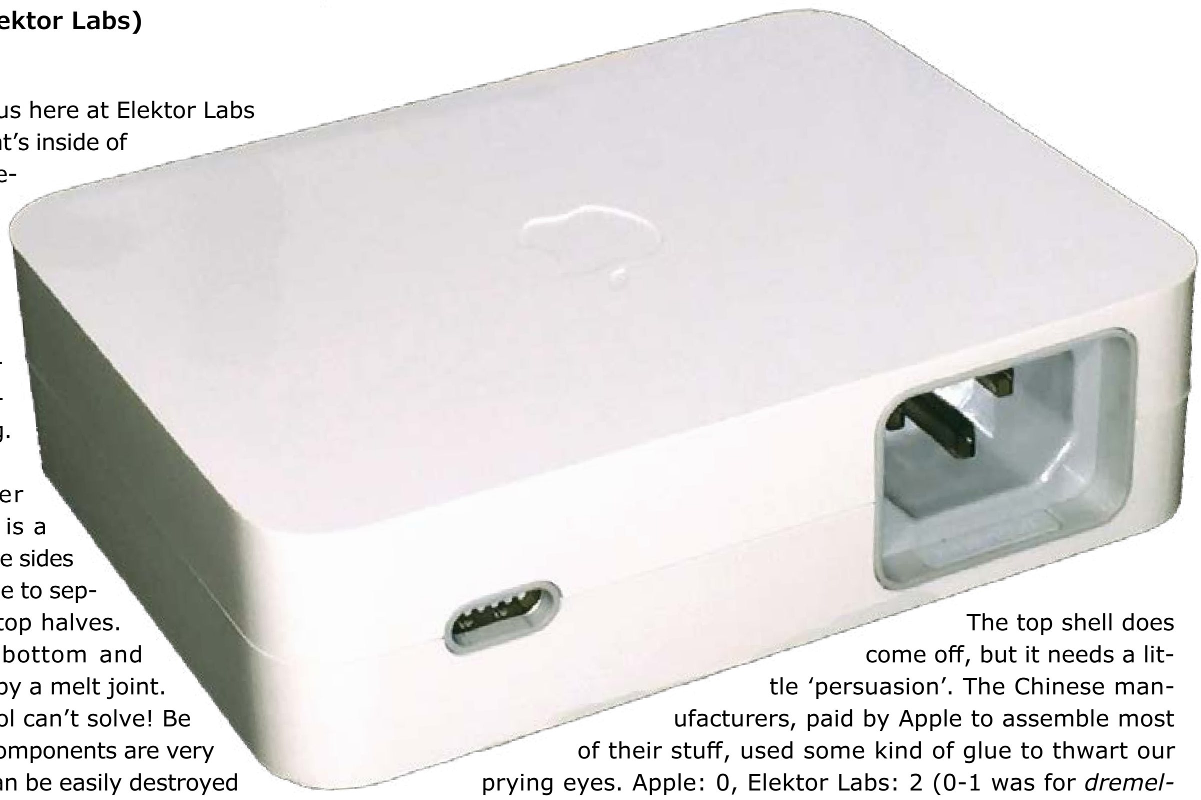
I don't know what you did last summer…

…but I know what you are doing right now. Thanks to a Platino board and a bunch of security cameras I can follow your every movement, like a real Big Brother. Easy, simply teach Platino how to speak Pelco, the security system's communication language of choice, and away you go. And since I also stuck a GPS tracker on your car, I know where you are going. Ha, Ha, Haaa! [Psychopathic laughter] Platino is your friend, or, [psycho whisper] is it…? You better behave, because I will know what you did *next* summer…
[http://po.st/platinopelco]

Inside Jobs

Most of today's Apple products are not meant to be opened easily by the general public and consequently require special tools for the job. A few products stand out by being complete black boxes, which .Labs hates to see. Like you, we have a strong urge to know what's inside the gizmos we buy. Most .Labs staffers started opening radios and TVs out of curiosity at an early age, not held back by any 20 kilovolts or more lurking inside. So why should a sealed plastic power adapter case be a show stopper?

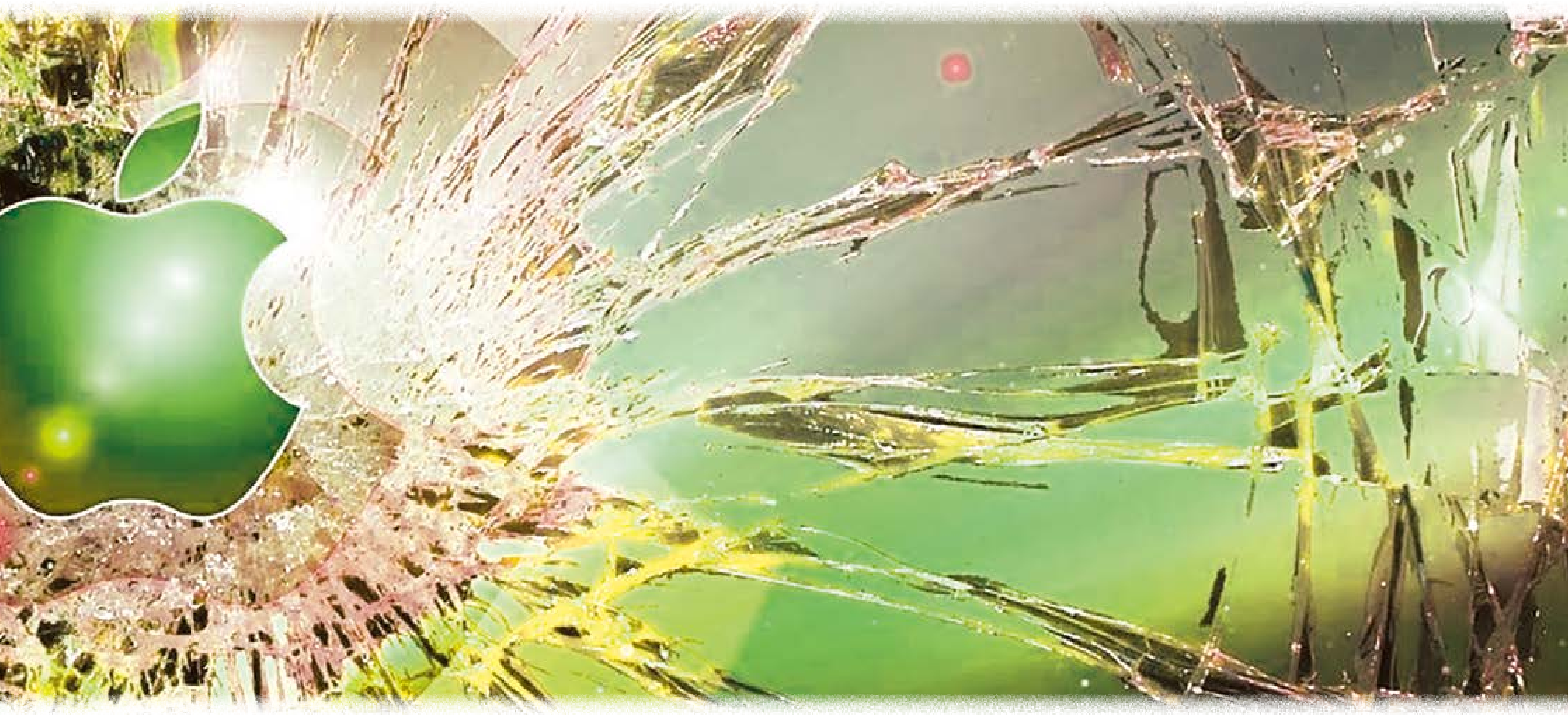
By Thijs Beckers (Elektor Labs)



If you're anything like us here at Elektor Labs you're interested in what's inside of anything electrical, especially when it's kaput and you feel the urge to fix it. Well, here's a little peek inside a 90-watt Apple Cinema HD Display Power Adapter that was suspected of malfunctioning.

Looking at the Power Brick it appears there is a seam running around the sides where you would be able to separate the bottom and top halves. *Appears*. In fact the bottom and top shells are secured by a melt joint. Nothing our Dremel tool can't solve! Be careful though: some components are very close to the case and can be easily destroyed if you plunge your abrasive disc a little too far into the plastic.

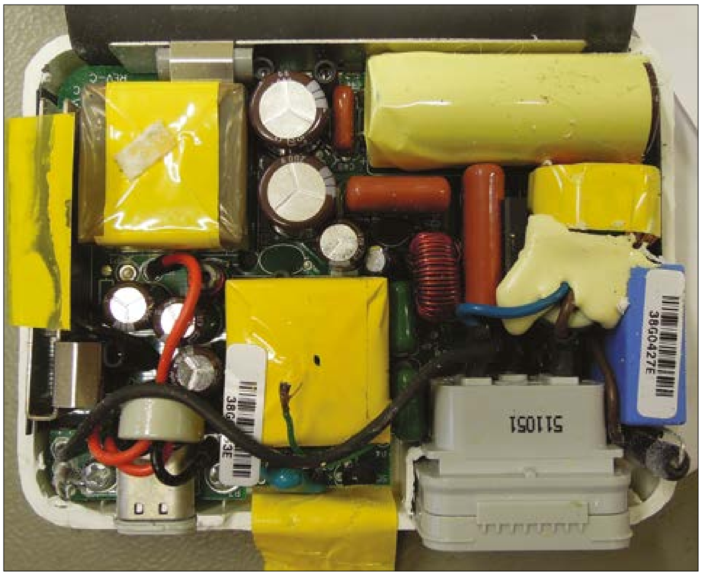
The top shell does come off, but it needs a little 'persuasion'. The Chinese manufacturers, paid by Apple to assemble most of their stuff, used some kind of glue to thwart our prying eyes. Apple: 0, Elektor Labs: 2 (0-1 was for *dremeling* the sealed case). Okay, so now we see a big copper slab that is folded around the whole circuit, acting as a shield. To

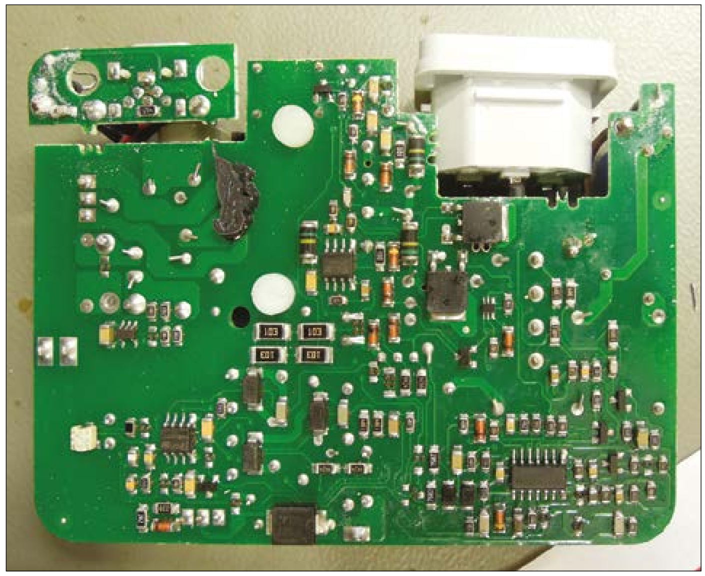


it a wire is soldered that connects the shielding to the PCB. This wire has to go to be able to open the shielding and get access to the circuit. Flipping over the copper shielding reveals a switch-mode power supply.

Now that it's out in the open we can freely have a look around. We notice that the PCB isn't screwed to the case, it's just a really snug fit. We also see that the connector for the Apple display power connector is mounted on a small separate board. That's pretty clever actually. This connector and the AC one, which is also not directly soldered to the main PCB, are the only parts subject to mechanical strain, so not mounting them onto the PCB, or mounting them on a separate PCB that is not firmly connected to the main PCB, is a neat way of making sure the connector doesn't break away from the board. The display power connector obviously isn't moving around freely, the small PCB holding the connector is screwed to the case.

The circuit is configured is like your dead standard switching power supply. There are some filter inductors and capacitors at the AC input, a bridge rectifier and a fat capacitor to generate the high DC voltage the transformer taps its power from. Then come two power MOSFETs for the switching part, followed by the transformer. The MOSFETs are driven by an ST Microelectronics L6571B high voltage half bridge driver with oscillator that seems to be a logical solution in this application.
On the secondary side of the transformer the circuit continues with a (heatsinked) rectifier and finally the output filter capacitors. The whole circuit is controlled by an STMicro integrated circuit, its type print undecipherable. And of course there's the obvious chicken feed of passives (resistors and capacitors) needed to complete the circuit.
There's also quite an extensive power factor correction (PFC) circuit. The transformer not completely coved in yellow tape is part of that circuit.





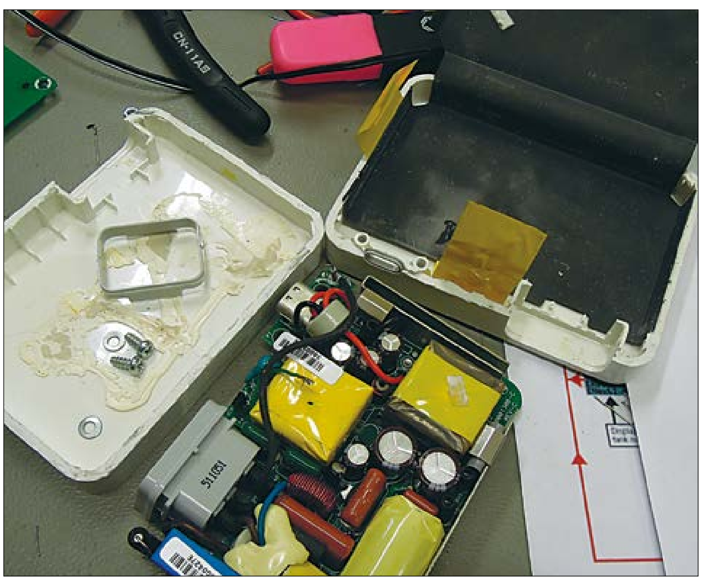
Looking at the cooling measures we see only two metal strips on two edges of the board. One strip holds the rectifier diode for the output circuit and the other one holds what we surmise is a switching FET for the PFC circuitry, which is based on an ST Microelectronics L6561D. For the type print on the FET to be readable a transformer has to be removed, which we really didn't feel the need to do.

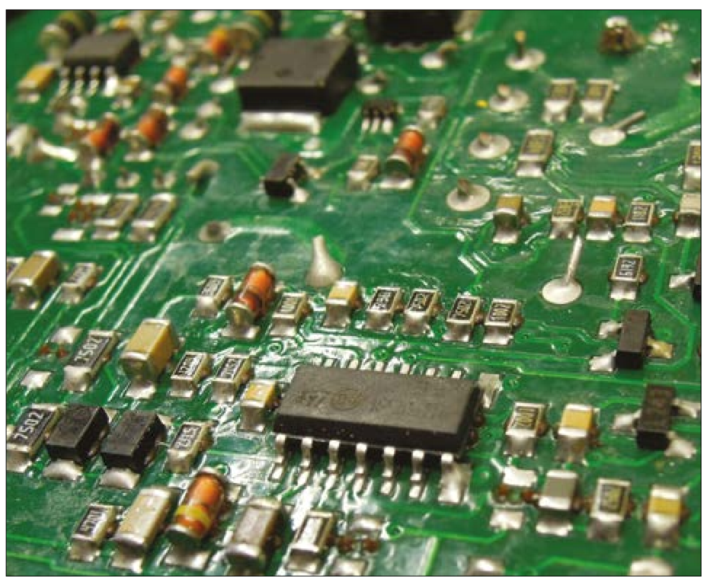
The main switching MOSFETs, type STD5NM50 500V-0.7Ω-7.5A in DPAK cases from (again) STMicro, are not mounted on a heat sink, they're SMDs soldered with their tabs straight onto the board. No copper plane for cooling underneath them. This must be an extremely efficient circuit, as it is able to convert 90 watts of power from 100-240 V_{AC} into ±24 VDC, which is the output voltage of this Power Brick, and dissipate the excess heat even when mounted in a *closed* plastic case. Not exactly an ideal situation...

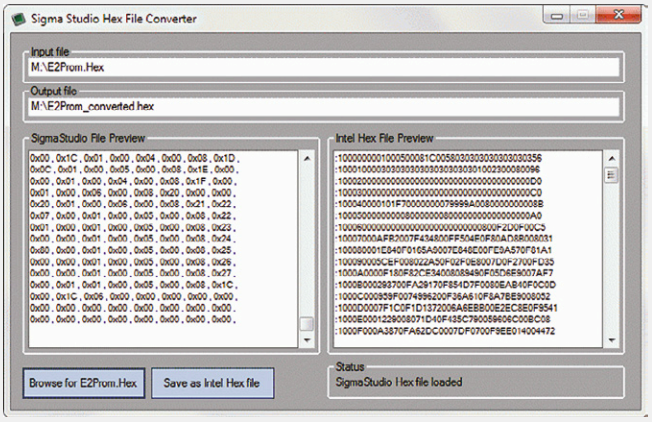
The Power Adapter turned out to work fine, correctly supplying the required ±24 V. Googling around we found that the Apple Cinema Display is prone to drawing too much current from the power adapter because of the backlight CFLs getting old, causing its overcurrent protection to trigger and shut off the power. In most cases you seem to be able to buy some extra time for your Cinema Display using a heavier 130-watt Power Brick, but it just postpones the sad outcome.
Since a new Cinema Display was already on order for replacing the old one (which had seen 7 years of day in, day out use) we didn't see any benefit in looking into it any further and accepted the adapter as a write-off.

Still, it was an interesting exercise and we now know what's inside an Apple Cinema HD Display Power Adapter, which is nice and useful to know.

(150022)







Convert or…

As seen in "ADAU1701 Universal Audio DSP Board," Elektor 1/2014, page 12 (130232)

Reader Christian Weidner adapted the Elektor programming software for the Universal Audio DSP Board into a file converter (Sigma Studio HEX File Converter) and sent us the code. We updated the project at Elektor.Labs, but you may also download the source code and the compiled executable from his website (link below!). Christian also suggests using a CH341A module to program the I²C EEPROM much faster, which can be found cheap (prices ranging from €4 – €12) on sites such as eBay. As and aside, the "USB to Multi-protocol Serial Converter" elsewhere in this edition can also do the job, since it also *speaks* I²C! Don't miss to give it a look. Christian also publishes about many other DIY projects, including DSPs, amps, free electronics CAD, and more. It's definitely worth a visit. Thanks Christian! [http://po.st/SigmaDSP]

Err-lectronics

Corrections & Updates to published articles

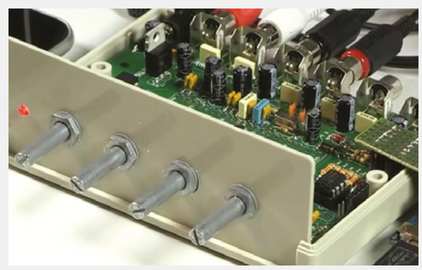
Compiled by **Jaime González-Arintero**

This time we aim to show not only that things can always be improved, but even re-worked entirely! Naturally, the *e-typos* in the current edition are totally intended: the flaws are there just to check if you are paying attention … (That's what bad teachers always say, right?)

You know you could be faster

As seen in "ADAU1701 Universal Audio DSP Board," Elektor 1/2014, page 12 (130232)

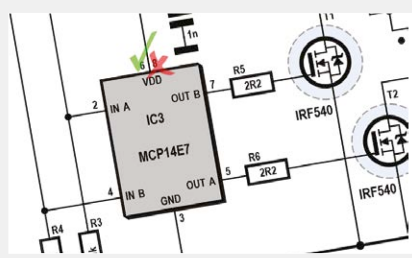
This board is famous! Here's another update. Alfred Rosenkränzer proposed a faster programming interface for the Universal Audio DSP Board, since he noticed that using the serial port for that purpose may be unreliable at times. Thus, he developed a small additional board with a real interface module, and he shared it with us. Indeed, trainee Niek at Elektor.Labs is giving it a try now, so we'll keep you posted. A big thank you, Alfred!



Gr8 catch!

As seen in "350-V Step-Up Converter," Elektor 2/2015, page 107 (130496)

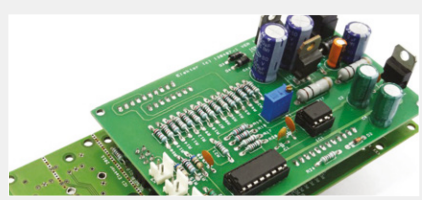
Reader Henrik Andersson recently spotted a typo that could drive you nuts if you build this project. In the published schematic of the DC booster (in Figure 6), there's an error with IC3 (MCP14E7). In the published article, V_{DD} is incorrectly shown as connected to pin 8. The datasheet of this MOSFET driver however shows that V_{DD} should be connected to pin 6. The correct schematic can be downloaded at the url below. Thanks for your eagle eye, Henrik! [http://po.st/130496]



Half is enough

As seen in "Experimenter's Function Generator," Elektor 1/2015, page 22 (130407)

The component list on page 27 names the reference diode IC6 as LM336BZ-5.0V, which is not correct. Instead, this component should be an LM336BZ-2.5V. Although this doesn't affect the schematic and/or the article since the suffix is not specified, those 2.5 V more might result in some head-scratching…



Nosing Around

Terrific source of inspiration for electronics enthusiasts

By **Harry Baggen** (Elektor Netherlands Editor)

The website www.hackaday.com will already be very familiar to many electronics hobbyists. But for those readers who do not know about this remarkable website, we will give an impression of all the things that can be seen and experienced there.

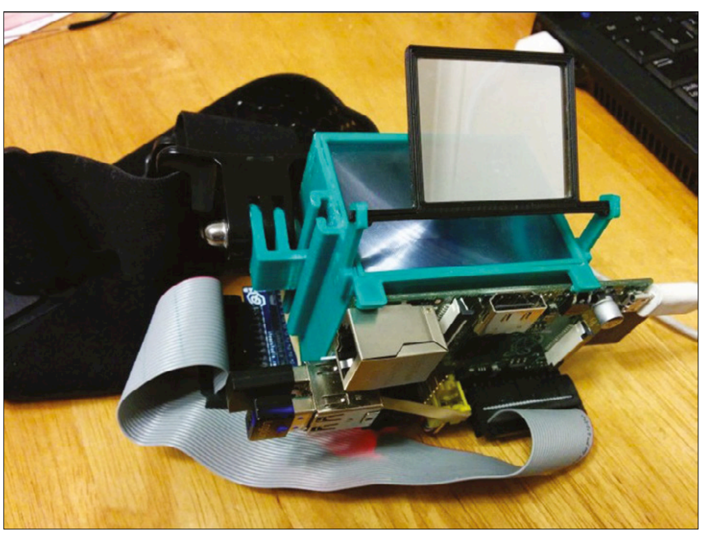
Actually, the name Hackaday is somewhat of a misnomer and does not accurately describe what you can expect to find on this website. For most people the name 'hacker' has negative connotations and this word is mainly associated with people who occupy themselves with various illegal things in the computer world (mainly on the Internet). This website therefore has absolutely nothing to do with that!

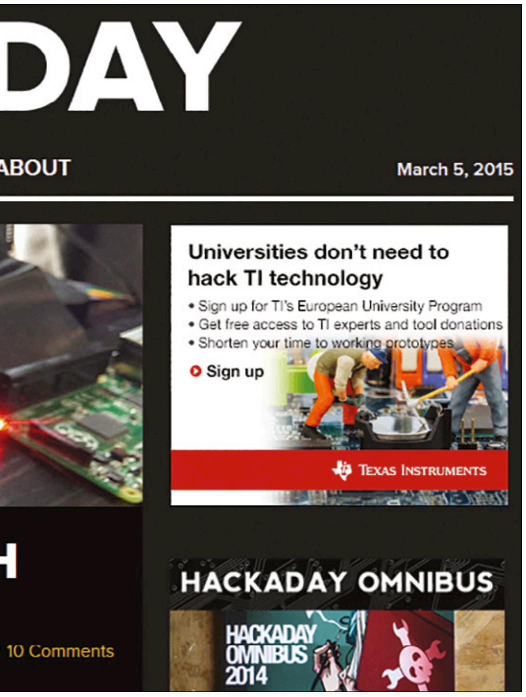
According to the people behind Hackaday, 'hacking' is the art of using things in a different way than for which they were originally intended or designed. These are foremost technical hacks, usually electronics related. You can come across all sorts of things, from a complicated DSP project to making an enclosure for your circuit.

Hackaday collects hacks that can be found on the Internet and presents them with a brief description and a link to the original website. In addition, Hackaday also offers a platform where hackers, modders, tweakers, designers and hobbyists can show off their projects, complete with descriptions and schematics, if desired. There is now also a large Hackaday-community where anyone can exchange information about projects. Just register and you can join in!

The number of projects or 'hacks' is enormous, you could browse around this website for an entire day and you will still have only seen a small fraction. In order to give you an impression, I looked around for a few nice and interesting projects. I do have to add that it was very difficult to make my selections, because of the large number of interesting projects!

A project that immediately appealed to me was 'Using the Red Pitaya as an SDR' [1]. Red Pitaya is an experimenting board with a powerful FPGA, which can, among other things, be used as a two-channel oscilloscope with high resolution and a considerable bandwidth. Brian Benchoff made a software-defined radio using this board. He wrote the software, starting with an application note from Xilinx. The only piece of hardware that you need to add is a simple loop antenna connected to one of the inputs.





Web Links

[1] http://hackaday.com/2015/03/02/using-the-red-pitaya-as-an-sdr/

[2] http://hackaday.com/2014/04/20/the-raspberry-eye-sees-all/

[3] http://hackaday.io/project/1477-the-worlds-simplest-hack-ipod-party-amplifier

[4] http://hackaday.com/2015/03/02/the-hard-drive-midi-controller/

[5] http://hackaday.com/2015/02/27/hacklet-36-oscilloscope-projects/

[6] http://hackaday.com/2014/12/12/hacklet-26-arduino-projects/

[7] http://hackaday.com/2014/11/21/hacklet-23-the-groove-tube/

Devotees of Google Glass should cast a glance towards the 'Raspberry Eye' [2], which was made by Roman Rolinsky. Roman made a head-up display using a RPi, a 2.4" display and a half-transparent mirror, which projects all kinds of information in front of one of his eyes. The entire construction is not very compact, but it does work and he mounts it on his head with the aid of a GoPro headband.

The levels of the various projects that are described here are very diverse. One of the simplest hacks I came across was a little trick to connect an iPod to an audio system [3]. Place one of the ear pieces against the head of the cassette player and switch it to Play. And voila, the sound from the iPod comes out of the speakers. Not optimal, but it works – very simple, you just have to think of it.

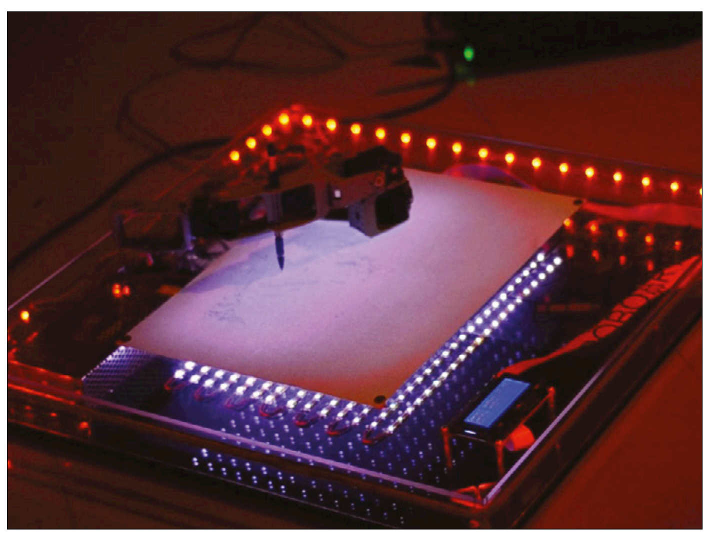
The 'hard drive MIDI controller' [4] is a great example of the re-use of discarded things. The builder of this beautiful controller used a disc from an old hard disk as rotary encoder on the front panel. Very original and probably very practical in use.

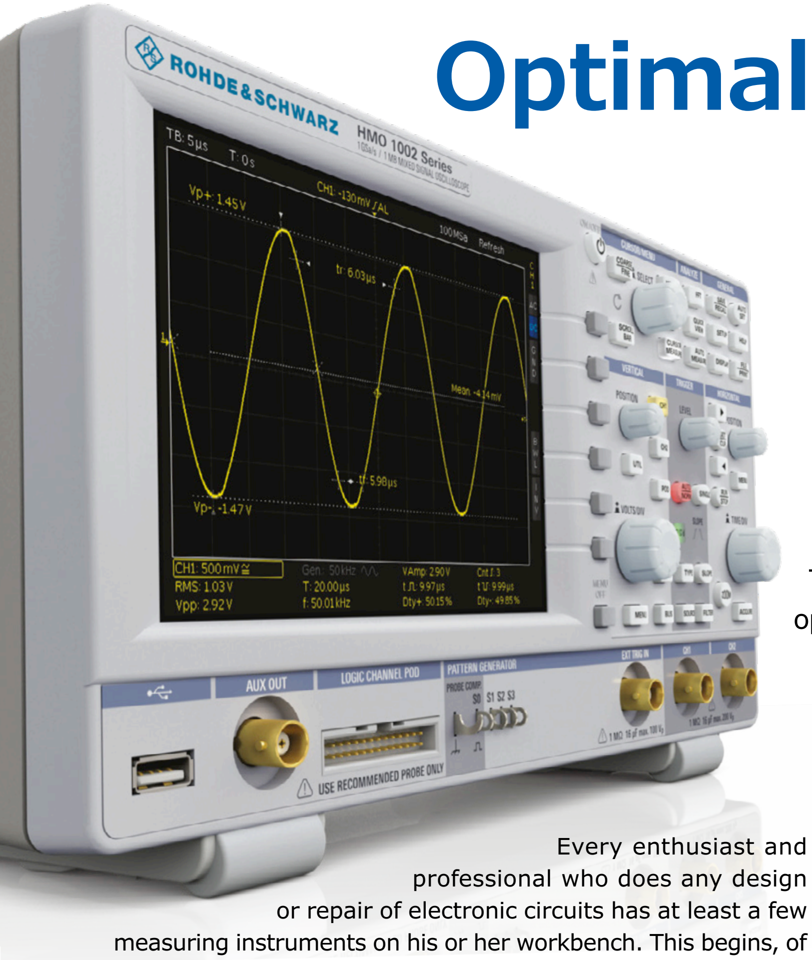
In addition to all the individual projects, every week there is also a so-called 'hacklet', a selection of interesting projects which all cover the same subject. So there are, for example, hacklets about oscilloscope projects [5], Arduino projects [6] and tube projects [7].

There are, by the way, more than 100 categories from which you can choose, including clocks, FPGA, drones, GPS, laser, but also some less obvious subjects such as the brewing of beer and the opening of locks! We must also mention the video section, a YouTube channel with hundreds is videos, which explain or demonstrate numerous hacks.

We can easily make the list of web links much longer, but it would be best if you were to take a look at this website yourself. Make sure you have plenty of time, because before you know it you will be wandering from one interesting subject to the next.

(150151)





Optimal Measurements with your 'Scope

Fundamental information and measurement tips

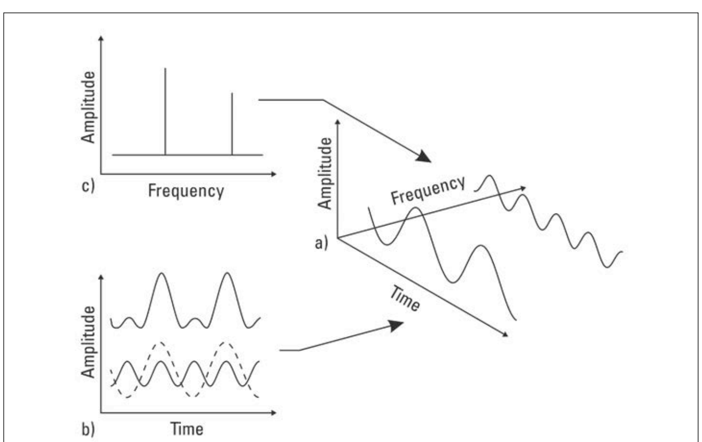
By **Harry Baggen** (Elektor Netherlands Editor)

An oscilloscope is a fantastic measuring instrument, particularly if you know how to use it properly. There are countless descriptions on the Internet about the operation and usage of a 'scope. But really, it should be the large oscilloscope manufacturers that should be providing this information. We went searching for what these manufacturers have to offer regarding basic information.

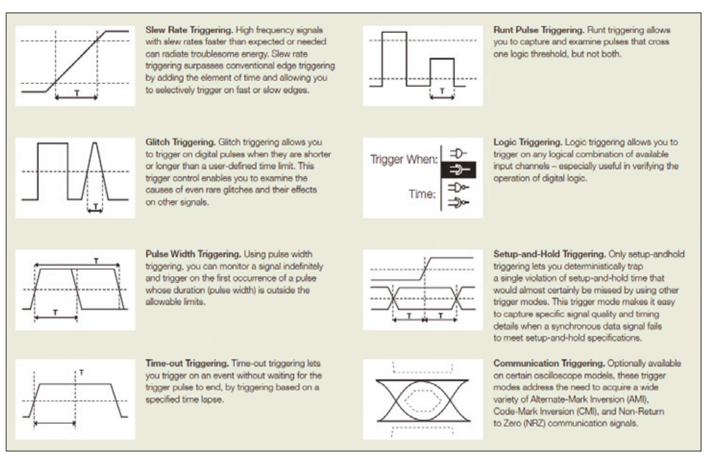
Every enthusiast and professional who does any design or repair of electronic circuits has at least a few measuring instruments on his or her workbench. This begins, of course, with a multimeter, but in addition you also really need a universal bench power supply and an oscilloscope. Real oscilloscopes were quite expensive in the past, but these days you can buy a nice USB 'scope for around 200 pounds. A usable, stand-alone, two-channel instrument can be purchased for about 500 pounds, thanks to various Chinese manufacturers. Once you are in the possession of a 'scope it is a very good idea to familiarize yourself with all the features and functions that the instrument offers. And you have to know exactly what the 'scope does to the measured signal before it appears on the screen, so that you can interpret the result correctly. This was already true for analog 'scopes in the past, but is even more important now with the digital versions.
Recently, I was standing in the lab next to a designer and we were staring at a strange looking signal on the 'scope, while we were expecting a completely different signal waveform at this particular point in the circuit. It turns out that this 'incorrect picture' was caused by the interference between the measured signal (which contained different fast-changing frequencies) and the sampling frequency of the 'scope. In a case like this you have to be really aware what operations the 'scope performs before the signal is made visible on the screen.
For those who would like more fundamental and background information about the internal operation of a 'scope and its correct usage will find plenty of explanations, tips and tricks on the Internet. The large manufacturers in particular have a lot of information on their websites, which is free for anyone to look at and download. And this does not only include manuals and tips for their own products, but after a little searching you will also find plenty of general information and tips.

Keysight

The American manufacturer Keysight (formerly Agilent) has an extensive database with articles. Under the tab 'Technical sup-



1



2

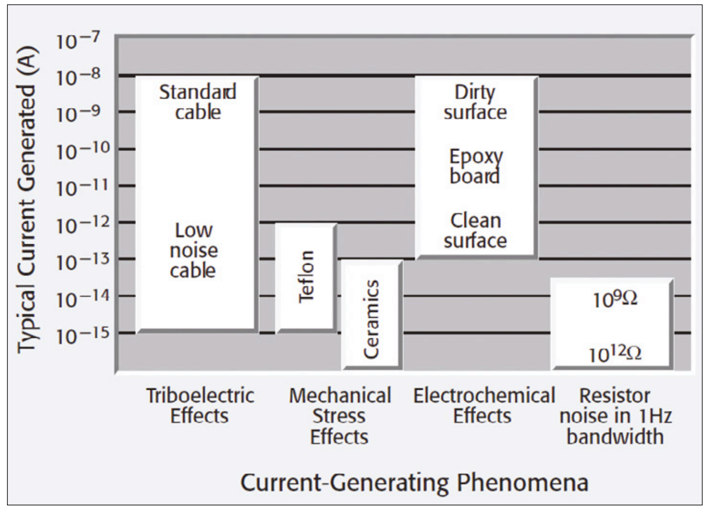
port' there are thousands [1], which of course also includes the brochures and manuals for the instruments produced by this manufacturer. There is however also plenty of documentation that contains general information. We selected a few interesting ones. The two-part application note 'Fundamentals of Signal Analysis' gives a general introduction to the measurement of signals in electronic circuits. The first part [2] is very suitable as an introduction to measuring in the time and frequency domains and often explains the operation of instruments and circuits based on familiar mechanical objects. The second part [3] takes this a step further and deals with topics such as FFT, sampling and aliasing (**Figure 1**). These are important subjects to the understanding of how a digital 'scope operates.
There are many more interesting subjects to be found in Keysight's document library. We can recommend the so-called '8 Hints' application notes, each of which give tips on a particular topic. We specifically mention here '8 Hints For Successful Impedance Measurement' [4] and '8 Hints for Making Better Measurements Using Analog RF Signal Generators' [5].

Tektronix

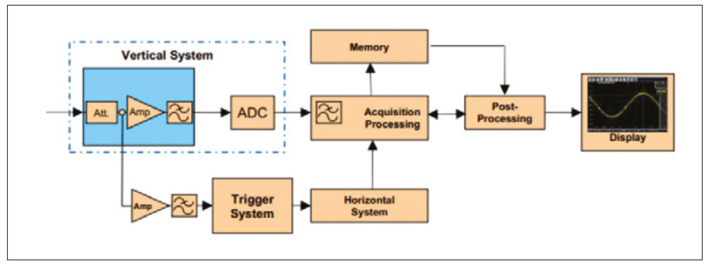
Tektronix even has a special place reserved for its informative brochures and application notes [6]. The first on this list is immediately a direct hit, the 'Oscilloscope Primer' [7] (which is actually called 'XYZs of Oscilloscopes'). You do have to enter your email address or register first before you can download certain PDFs, but this is certainly worth the effort! This informative 59-page e-book provides a lot of explanations about the use and operation of oscilloscopes. It also starts very easy with the description of the most common waveforms, but continues with the features that modern 'scopes offer (such as trigger options, **Figure 2**).
Another interesting PDF is about a subject that many electronics enthusiasts often underestimate, the Probes Primer (ABCs of Probes) [8]. The 60 pages in this explain in detail how a probe is built, what influence it has on the measurement, which types there are and how to use them in different situations.
Via the Tektronix website we also arrived at Keithley, which these days is part of Tektronix. Here we found an interesting e-book that does not deal with 'scopes directly. For measurements of small signals the 'Low Level Measurements Handbook', with a size of 240 pages, gives very detailed tips and explanations (see **Figure 3**) that can also come in handy when using an oscilloscope. Here too you have to leave your email address behind.

And the rest...

The other large oscilloscope manufacturers also have informative articles available for anyone who is interested in this subject. For example, at Teledyne LeCroy we found an interesting article about the use of probes (Probing Tutorial) [10]. This mostly deals with the effect that the probe has on the measured node of the circuit, with various examples. Another subject that is more suitable for 'advanced' 'scope users is the PDF 'Why Differential' [11], which explains the advantages of differential measurements.
We cannot finish, of course, without also mentioning Rohde & Schwarz in this overview. There too we found interesting reading material with basic information about 'scope s [12]. A nice feature of this document is that the entire 'scope is



3



4

explained from block diagram through to special functions (**Figure 4**), including a bit of history and a digression about the use of probes.

As you can see, with a little searching a lot of excellent technical information can be found that anyone can download. In addition you can be assured that the contents of the articles described here are technically correct, something that is certainly not generally the case on the Internet.

(150032)

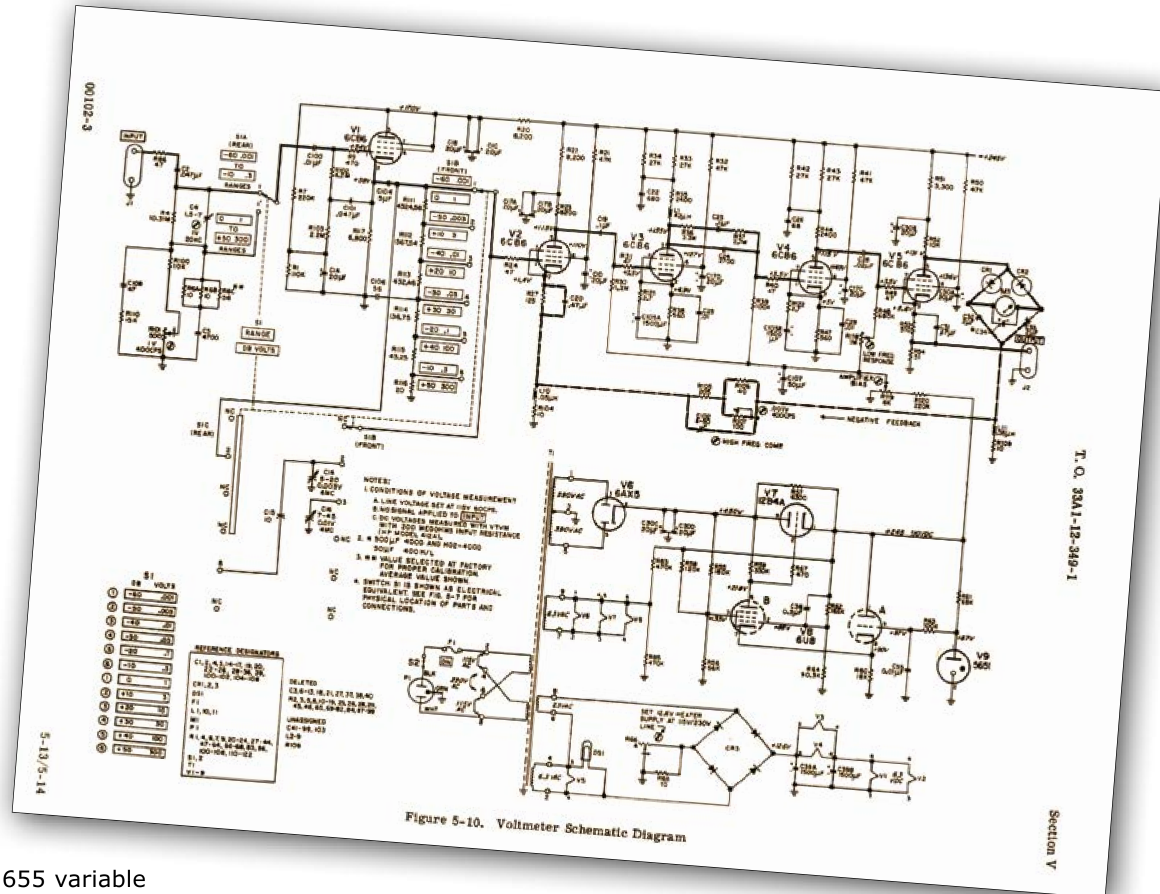
Web Links

[1] www.keysight.com/main/facet.jspx
[2] http://literature.cdn.keysight.com/litweb/pdf/5988-6765EN.pdf
[3] http://literature.cdn.keysight.com/litweb/pdf/5988-6774EN.pdf
[4] http://literature.cdn.keysight.com/litweb/pdf/5968-1947E.pdf
[5] http://literature.cdn.keysight.com/litweb/pdf/5967-5661E.pdf
[6] www.tek.com/learning/oscilloscope-tutorial
[7] http://info.tek.com/www-xyzs-of-oscilloscopes-primer.html
[8] http://info.tek.com/www-abcs-of-probes-primer.html
[9]http://info.tek.com/KI-Low-Level-Measurements-Handbook-LP.html
[10] http://cdn.teledynelecroy.com/files/appnotes/lecroy_probing_tutorial_appnote016.pdf
[11] http://cdn.teledynelecroy.com/files/whitepapers/wp_differential_measurements.pdf
[12] www.rohde-schwarz-usa.com/rs/rohdeschwarz/images/Oscilloscope-Fundamentals_v1.1.pdf

HP 400H VTVM Restoration
Part (2)

By **Chuck Hansen** (USA)

Here we continue from Part 1 published in Elektor 2/2015 (March & April). In the text below, reference is made to the 400H schematic, which is reprinted here for your convenience.



Testing begins

I installed an NOS 5651A V9 regulator tube, and powered up the HP 400H, gradually increasing the line voltage with my B-K 1655 variable AC power supply over a 15-minute period. The regulated B+ bus was now at the proper +245 VDC, but the screen and plate voltages for all four amplifier tubes V2-V5 were between 15% and 37% high.

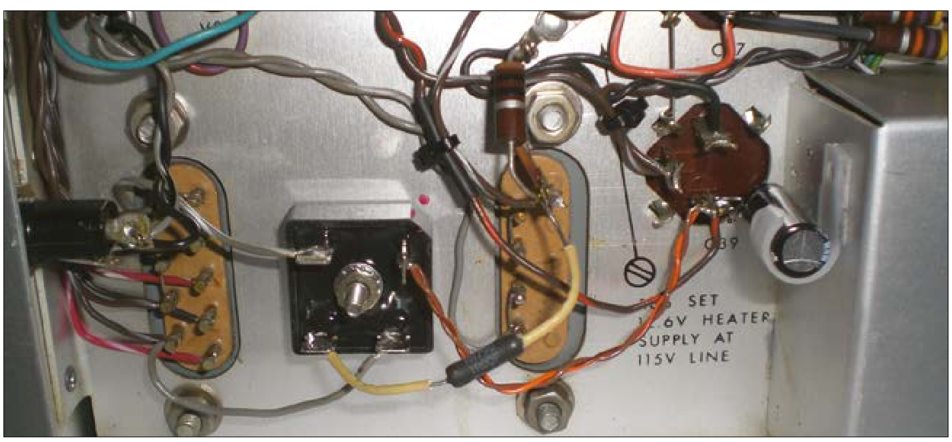


Figure 7. CR3 replaced with a silicon bridge; the added 1-Ω resistor; and C39C paralleled with a discrete 105-°C, 1500-μF capacitor (on V1-V2 shield at right).

Next, I found the rectified filament voltage for tubes V1-V4 was low at 10.5 VDC. R66 is used to adjust the filament voltage to the specified 12.6 VDC at 115 VAC line voltage, but it would only reach 11.5 VDC with R66 fully clockwise. The series-connected filament windings have the proper 13.6 VAC output.

Before CR3 fails in its odiferous (and toxic) mode, I will replace it with a full-wave silicon rectifier bridge. I am sure I will have to add a resistor in series with the silicon bridge since the silicon forward diode drops are usually lower than selenium, especially as the selenium rectifier ages. Then I can adjust the amplifier bias pot R119 to try to obtain the correct voltages on those amplifier tube elements.

I will go ahead and, as a precaution, replace the C30 quad 20-μF 450-VDC can cap with a quad 20-μF 475-VDC cap I ordered from AES. This is the cap that sees the highest B+ voltages, and may have suffered the most when the regulator tube failed.

ESTD 2004

www.elektor.tv

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

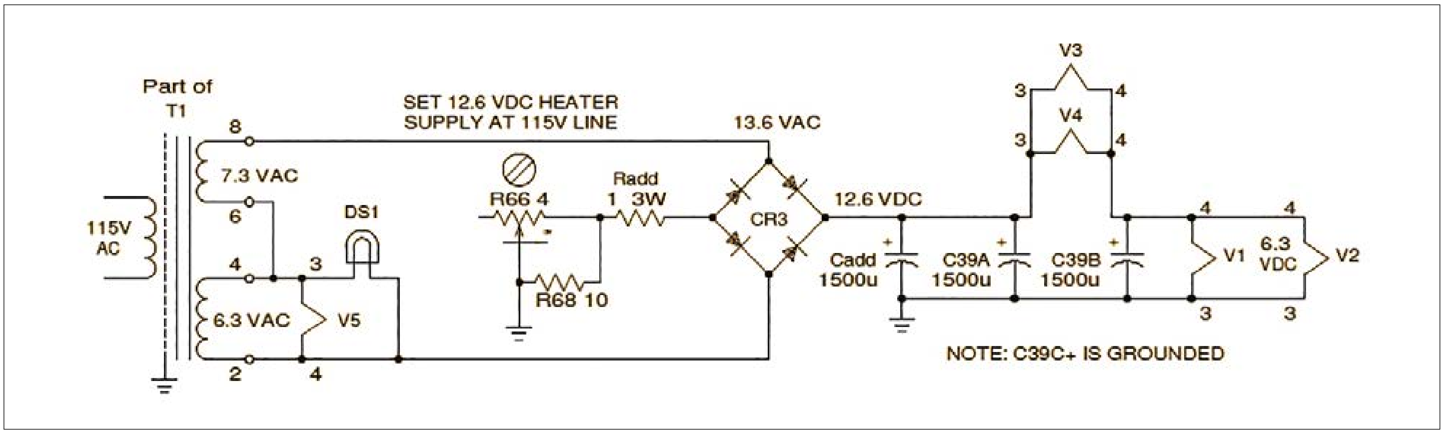


Figure 8. Schematic for the modified filament DC power supply.

HP-400H repairs and further testing

After replacing CR3 with a 50-V 15-A silicon rectifier bridge, I found the forward drop on two legs of the selenium rectifier was almost 2.5 V. I added a power resistor decade box in series with the negative end of the bridge. When I energized the HP-400H again my 'scope showed very high ripple on the DC filament voltage. C39 appears to be degraded but, unfortunately, AES does not list a replacement in their online catalog (tubesandmore.com).

I paralleled a 1500-µF discrete capacitor across filament filter capacitor C39A. I supported it with a modified Burndy wire clamp on the left side of the V1-V2 shield in **Figure 7**. I also had to adjust the series decade box resistance to 1 Ω to reduce the filament voltage back to 12.6 VDC with filament adjust pot R66 centered. I installed a Dale RS2B, 1-Ω 3-W wire-wound resistor in place of the decade box.

I had to extend the gray and gray/white wires from the transformer filament winding a bit to reach the AC connections on the bridge rectifier. Then I had to splice the brown/orange wire that went to the selenium bridge + terminal, to the filament wire, in order to complete that circuit. I didn't have any brown/orange wire to make the connection from the silicon bridge + connection to C39A, so I twisted brown AWG-22 and orange AWG-24 Teflon® wires together to retain the color identification.

The schematic for my modified filament DC power supply is shown in **Figure 8**. With the HP 400H input shorted, the meter needle pins on the two lowest ranges, 0.001 V and 0.003 V. It should be about one minor division off zero on the 0.001 V scale. There are two discrete Sprague 50-µF, 6-V aluminum caps; C34 across the meter, and C107 for the amplifier bias supply. I ordered two 47-µF, 16-V Vishay caps rated for 105 °C from Mouser.

First, I addressed the high residual voltage readings, which pinned the meter on the 0.003 V scale. I replaced C34, the cap across the meter terminals, with the Vishay 105-°C 47-µF, 16-V aluminum cap (**Figure 9**). This brought the meter reading down to 0.18 mV on the 0.001 V range with the input shorted and case removed.

I checked the old cap with a Sencore LC102 LC meter and it only read 25 µF at 5.4 VDC. The ESR was a high 15.7 Ω. The dielectric absorption reads 31%, with over 700 µA leakage, both flagged as BAD by the LC meter. It wasn't really even a capacitor anymore.

The bias voltage was too high on all the 6CB6 amplifier tubes. I decided to replace C107 with the Vishay 47-µF, 16-VDC aluminum cap. Unfortunately, C107 is buried under a lot of discrete components and a black jumper (**Figure 10**), so I had to find a way to minimize the number of parts I had to remove in order to replace it.

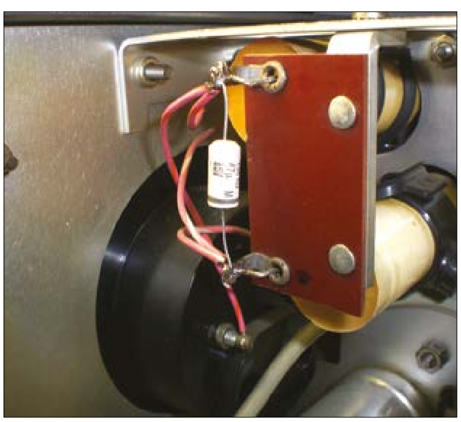


Figure 9. Replacement C34, the cap across the meter terminals.

Figure 10. C107 (silver case) is located under discrete components and the black jumper.

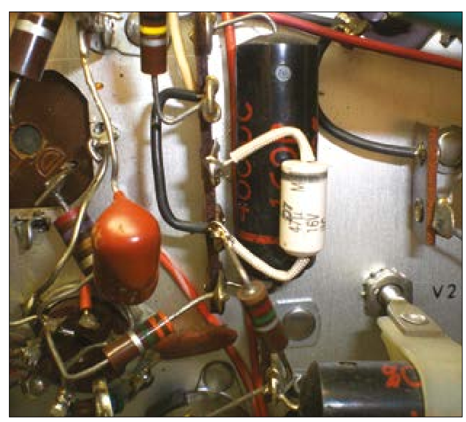


Figure 11. C107 replacement capacitor mounted above C23.

Figure 12. New C30 can capacitor and NOS 6CB6A tubes.

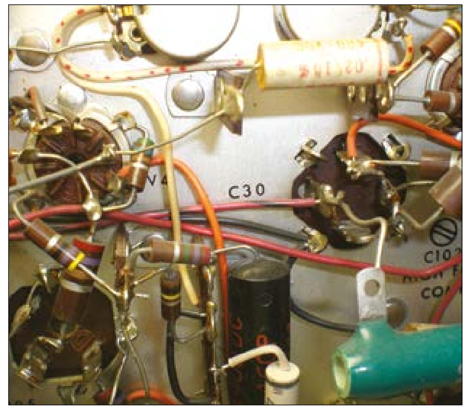


Figure 13. Original C30 and the surrounding components.

Table 1.
HP 400H Voltage Error Test

VTVM Scale	Freq.	$V_{applied}$	V_{error}
300	60 Hz	300	0%
		270	0%
		30	–3.3%
100	60 Hz	100	0%
		90	0%
		10	5.0%
30	60 Hz	30	0%
		27	0%
		3	0%
10	60 Hz	10	0.8%
		9	0.22%
		1	4.0%
3	1 kHz	3.0	0%
		2.7	0%
		0.3	1.3%
1	1 kHz	1.0	0.5%
		0.9	0.6%
		0.1	1.6%
0.3	1 kHz	0.30	2.3%
		0.27	2.5%
		0.03	4.3%
0.1	1 kHz	0.10	2.0%
		0.09	2.2%
		0.01	8.0%
0.03	1 kHz	0.030	1.7%
		0.027	2.2%
		0.003	23%
0.01	1 kHz	0.010	3.0%
		0.009	5.9%
		0.001	28%
0.003	1 kHz	0.0030	2.3%
		0.0027	4.1%
		(see text)	
0.001	1 kHz	(see text)	

The easiest way was to snip out the old C107 and install the new one above the black Sprague 160P film cap C23 (**Figure 11**).
I checked the old capacitor and it measured 51.6 µF, with an ESR of 3.9 Ω. The 17-µA leakage and 12% dielectric absorption is still quite respectable. Both were rated GOOD by the Sencore LC102. The new Vishay capacitor reads 45.1 µF and ESR was 3.7 Ω, well below the 5.4-Ω max limit.

With a little tweaking of R119 I corrected the grid bias on all the tubes. But V3 plate voltage was 154 VDC instead of 121 VDC. I swapped V3 and V4 and now V4 had the high plate voltage (152 instead of 119). Since at least one tube was weak, I ordered 6CB6 pentodes for V1-V5 from Antique Electronic Supply.
While checking the parts in the amplifier circuits I found someone else had previously worked on this VTVM. There were a number of cold solder joints, indicating whoever worked on it was not all that experienced as a tech. I also found the following changes:

1. C24 originally called for a Sangamo 1800-µµF 500-VDC mica cap, or 2700-µµF mica on my later schematic. It had been replaced by a 0.0025-µF 600-VDC Sprague 220P Orange Drop polyester film cap.
2. R27 was supposed to be a 125-Ω ½-W Dale wire-wound. It had been replaced with a Dale CS-2 125-Ω 3-W wire-wound. I can't find a reference to the Dale CS series in my 1991 Dale Electronics catalog. It looks exactly like the Dale RS series I am familiar with.
3. C105, the capacitor that leaked onto the chassis, had been replaced with another identical Sprague can capacitor with a newer 1957 date code. All the other can caps are dated 1952. When C105 was replaced, the leads for R122 were clipped too short. It was then reinstalled with tinned bare jumper wires.
4. The plastic sleeve over C30 can has some tan/brown areas, suggesting that it got hot after the V9 regulator tube failed. I installed the set of five NOS 6CB5A tubes, then tackled the replacement of the C30 20 µF x4, 450-VDC

Replacement parts used in the HP 400H

From Antique Electronic Supply (tubesandmore.com):

- C30 — 20 µF x 4, 475 V aluminum can cap — C-EC20X4-475
- V1-V5 — pentodes — 6CB6-A/6CF6
- V9 — cold gas regulator — 5651

From Mouser (mouser.com):

- CR3 — 12 A, 100 V silicon bridge rectifier — 512-GBPC1201
- C34, C107 — 47 µF, 16 V, 105 °C alum cap — 594-2222-138-25479
- C added — 1500 µF, 16 V, 105 °C alum cap — 667-EEU-FR1C152LB
- R added — 1 Ω, 3 W, 1% wire-wound resistor — 71-RS2B-1.0

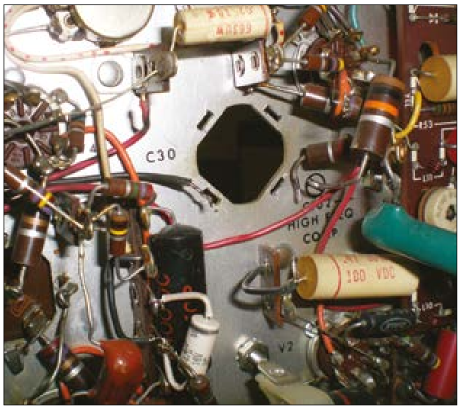


Figure 14. C30 removed from the chassis.

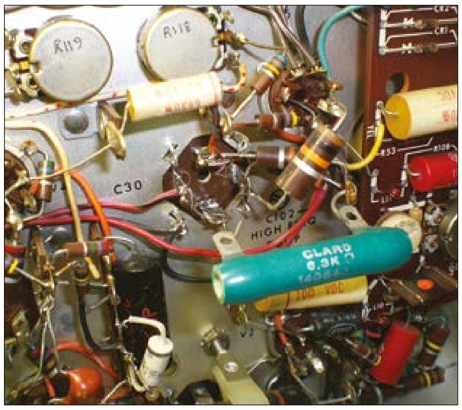


Figure 15. Replacement C30 installed and surrounding components reconnected.

Figure 16. Replacement C30 installed, chassis side.

can capacitor with the AES 475 VDC version. The process is illustrated in **Figures 12 through 16**.

The plate voltages are now within spec except for V4 and V5, which are a bit high. Swapping tubes around did not appreciably change any of the tube voltages. The 47-kΩ series screen grid resistor for V5 measures 52 kΩ.
The meter now reads 0.017 mV on the 0.001 V range with the input shorted and the cover installed.

Accuracy check

With all the repairs made, I calibrated the HP 400H in accordance with the HP manual, using my Fluke 8050A DMM (calibrated with a Fluke 5200 AC voltage calibrator in June 2014) in parallel with the HP 400H. I used my HP-339A distortion test set oscillator as a signal source.

First, I ran a frequency accuracy test with a 1.000 V_{rms} sine wave from 10 Hz to 50 kHz. I set the HP 400H on the 1 V range. It measured 0.95 V at 10 Hz and 0.96 V at 20 Hz. Then it was flat from 50 Hz to 20 kHz, and read 1.005 V at 50 kHz, which is the maximum flat frequency response limit for my 8050A DMM. I don't have the equipment to test the HP 400H up to its maximum frequency limit of 4 MHz.

Next, I ran a voltage accuracy test at 10%, 90% and 100% of full scale on all twelve voltage scales. I used my HP-339A oscillator set to 1 kHz, increasing the output voltage from 2.7 mV_{rms} to 3 V_{rms}. The lowest level I can generate from the HP-339A is 2.5 mV, so I could not accurately test the 0.001 V range of the HP 400H. The 10% of full scale voltage error increased as the signal level decreased. However, the meter is well within the ±1% error limit at 90% and 100% of full scale down to the 1-V range. Below that level noise seems to be affecting the meter accuracy. See **Table 1** for a test data summary.
The HP 400H manual suggests installing an AC line "cheater" plug to break any ground loops in order to improve the accuracy on the lowest voltage ranges. I am not a big fan of this practice, especially with a piece of equipment over 50 years old whose chassis insulation withstanding voltage is unknown.
The highest HP-339A oscillator output voltage is 6.75 V_{rms}. For the HP 400H 10 V and higher ranges, I used a 0 to 325 V variable 60 Hz AC power supply. The 10% of full scale reading errors were less than ±5%. The HP 400H fared much better at the 90% and 100% of full scale readings, usually being dead on. See Table 1 again.

Unfinished work

I need a new handle to replace the tattered leather handle. AES used to sell nice replacement leather handles, but now they only offer molded rubber handles for guitar amplifiers, complete with logos. There are two more quad 20-µF, 450-V can caps I probably should replace; C1 and C17.

I would very happy if I could find a replacement for the two low voltage 1500 µF x 3, 15 VDC can capacitors, C34 and C107. I ran into the same problem when I renovated my Scott 222C integrated amplifier (*audioXpress* May 2006 through Sept 2006). While AES carried the high voltage can caps, there was no replacement available for the quad 75-µF, 75-V output tube grid bias can cap (C207-C210). I had to bridge the internal caps with discrete external 100-µF, 100-VDC 105-°C aluminum caps.

All in all, I am quite satisfied with the performance of this HP 400H. It was an interesting unit to troubleshoot and repair, and to see how problems had been addressed by those who worked on it in the past.

(150034)

The Author

Chuck Hansen is an Electrical Engineer and holds five patents in his field of engineering, and works as a consultant in the aerospace industry. He has written two books for Audio Amateur Publications, and has over 260 magazine articles to his credit. Chuck began building vacuum-tube audio equipment in college. He enjoys sailing and playing jazz guitar. He likes to modify guitar amplifiers and effects to reduce noise and distortion, as well as building and restoring audio test equipment.

Pye P87BQ Radio Find & Restore

Brasso & The Battle of (Old) Hastings

By Ronald Dekker (Netherlands)

When my wife and I had just met, now nearly 18 years ago, we spent a few fantastic holidays camping and travelling "in Britain by motorcar". It has been a longstanding wish to return to the beautiful countryside of Britain, but for a number of personal reasons that has been difficult these past years. But this year it finally happened, the children were old enough to appreciate the trip, dogs were allowed to travel the channel (be it after a lot of paperwork), and we had something to celebrate! After a lot of searching my wife found a beautiful cottage in the charming village center of Old Hastings, Sussex. Old Hastings is ancient with many houses dating from the 15th and 16th century situated between two cliffs at walking distance from the beach with many (seafood) restaurants, nice shops and pubs. I think I could spend a whole page describing all the attractions and sights in and around Hastings, but I will limit myself to one specific treat I enjoyed in particular. Every morning, while my wife was still sleeping, I first walked the dog on the greens on top of the cliffs. It is amazing how green and free of litter the grass is on those hills! Coming from Holland, which is extraordinarily flat (except Southern Limburg, *Ed.*), the view of the beach from the cliffs is amazing. After my walk with the dog, I would go to the beach and watch in the mild May sun the fishing boats being towed on and off the beach. At around 9 when the coffee bars opened I would go to the corner of George Street and High street and enjoy a cappuccino in The Corner Coffee Shop, absorbed in Mark Frankland's *Radio Man* [1]. What a delight!

Before we left, I had agreed with myself to keep my eyes open for a nice vintage tube radio, preferably from Pye. At the time of the trip I was researching the history of Pye for my page on the history of the EF50 tube [2]. One way or the other Pye Radio, with its very out of the ordinary, outspoken and extravagant director C.O. Stanley had caught my fascination, and I thought it would be great to take a Pye product back to Holland as a souvenir, although I realized that the actual chance of finding a treasure like that during such a short holiday would be minimal.

Now for some reason there are literally dozens of second-hand and antique shops in Old-Hastings. Most of them have really nice stuff. I think it was the second or third day of our holiday when my son Geert and I returned from a nice walk on the East Hill greens when we passed one of the these antique shops in Courthouse Street. It was a strange shop, more of a courtyard filled with tiny shops or even sheds from a handful of antique dealers (**Figure 1**). Geert, with little interest in antiques, went ahead with the dog, and I went in to have a look. And there it was! A beautiful Pye battery portable! I immediately recognized it from a pic-

Figure 1. The antiques shop in Courthouse Street, Old Hastings, where I bought the P87BQ, and its proud owner in front of the cottage rented in Church Passage.

Figure 2. Inside the radio.

ture on the cover of *Radio Man*. What a coincidence! Apart from the fact that unfortunately the back cover was missing it was otherwise in pretty fair condition. There was very little rust but most importantly, all the tubes were there. The best thing of all: it was only £15, an unbelievable bargain! I immediately grabbed it and triumphantly walked to the shop owner, if you can call his old garage filled with antiques and vintage bric-a-brac a shop. He was a flamboyant character with a straw hat, and I am sure that when he noticed the twinkle in my eyes, he realized that he had charged me too little.

The P87BQ at a glance

Although I am not a vintage radio specialist at all, I happen to see from time to time some old radio sets. Being a Dutchman, most of these sets are made by the Philips

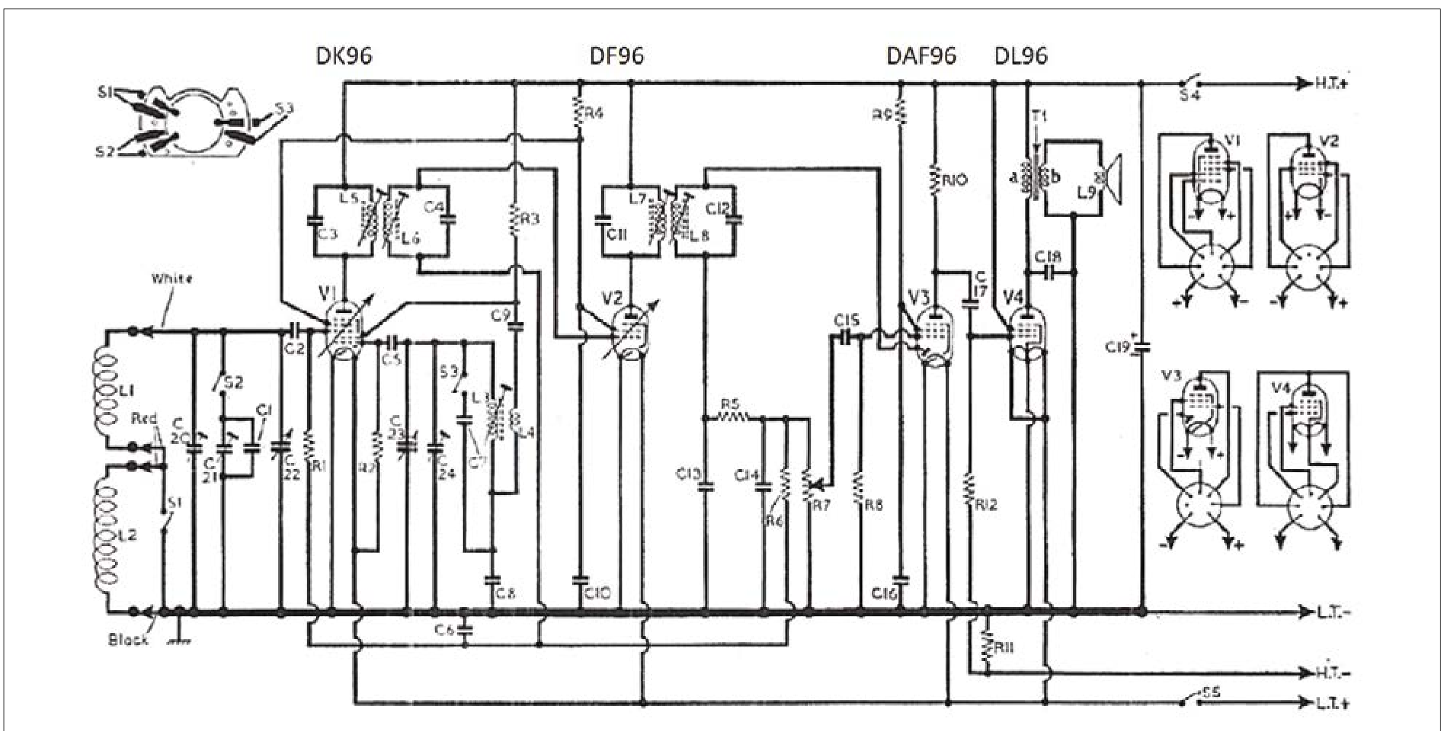


Figure 3. Circuit diagram taken from the *Trader Service Sheet* for the P87BQ radio [3].

corporation, so I am used to what may be called "the Philips style" of making radios. To me it came as a surprise that this British Pye set had a completely different look. I was pleasantly surprised by the mechanical robustness and the neatness of the whole assembly, which was in the well-known bird's nest style. Color coded flexible wire was used for all the longer wire connections. Underneath the chassis, color coded wires were used for the interconnects also, while the longer component leads were neatly isolated with colored plastic isolation tubing. The tuning capacitor and the volume potentiometer looked robust and of high quality.

What struck me was the different look of the resistors and capacitors compared to what I am used to seeing in Philips sets. We live in a time where everything is global, distance these days seems trivial and the number of electronic component manufacturers seems to have dwindled to a handful, mainly located in Asia. Components have been standardized, and look the same all over the world. This was certainly not the case some fifty years ago, when every industrialized country had their own component and radio makers, and in many cases even more than one! Although most of the components at first sight appeared pretty much ok, there were some wax coated tubular capacitors looking grimy, greasy and some even felt sticky.

Into the case

Figure 2 shows the radio opened up. Strikingly there was no ferrite rod antenna. Philips pioneered the development of ferrites and held some very important patents in this area. In Philips radios of that epoch ferrite rod antennas were common, providing a higher sensitivity and being smaller than a frame antenna such as used in this Pye receiver. The frame antennas are located in two slots at opposite sides of the carrying case underneath the two corrugated plastic strips. One of the frame antennas is for the medium-wave (MW) band and when both antennas are switched in series, the long-wave (LW) band is covered. **Figure 2b** was taken with the camera inside the carrying box, showing the point where the frame antenna wire enters. Apparently the antenna is wound on a paper or cardboard strip first and then glued in the wooden case. The slot in the wooden case is used to tighten the antenna wire around the case. It must have been quite a labor intensive procedure altogether.

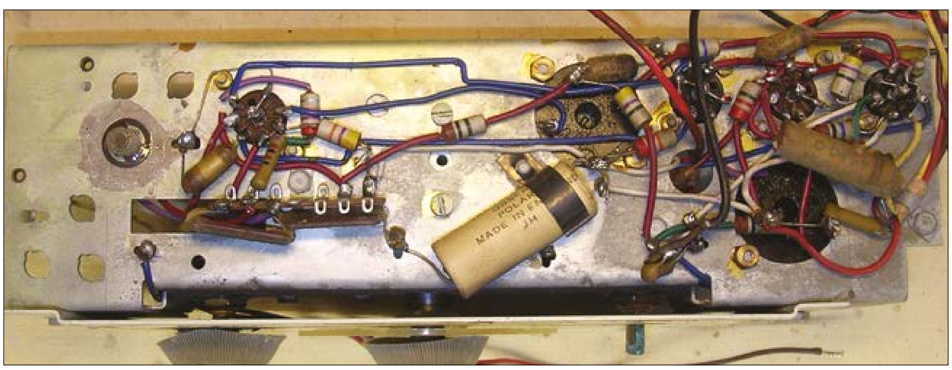


Figure 4. Chassis underside after removal from the carrying case.

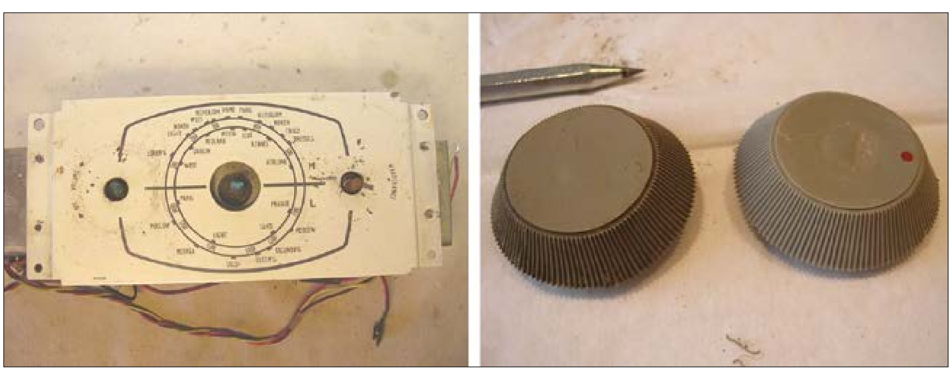


Figure 5. The dial plate before cleaning (left), and one of the knobs before and after cleaning (right).

I was lucky to be able to download *Trader Service Sheet 1184* for free from [3]. I also downloaded the official Pye service datasheet from www.service-data.com for a small fee, but this didn't contain any new information. The *Trader Service Sheet* contains a lot of useful information. Besides the schematic, it gives a description of the circuit, the procedure for circuit alignment, and — very useful — the anode and screen grid voltages and currents.

Referring to **Figure 3**, the circuit itself is pretty straightforward. A tube lineup like the Dx96 series was designed by Philips/Mullard for a certain basic circuit configuration, allowing set makers to add features like tone controls or multiple band coverage. The DK96 heptode (V1) is used as a classic RF amplifier and mixer. For the MW band switch S1 is closed (low inductance), while S2 and S3 are open (low capacitance). For the LW band S1 is opened (both frame antennas in series) while S2 and S3 are closed. The intermediate frequency (IF) is 470 kHz and the IF signal is amplified by a DF96 (V2). The diode section of V3 (DAF96) rectifies the IF amplified signal by short circuiting its positive portion to ground via the diode in V3. This will result in a negative DC voltage across C14 onto which the AC audio signal is superimposed. This signal is further smoothed by R6 and C6 and used for the automatic gain loop (AGC). A large signal results in a more negative feedback voltage, which in turn reduces the bias on the control grids of V1 and V2, hence reduces the gain (V2 is a variable-mu pentode). The audio signal is finally amplified by V3 (DAF96) and V4 (DL96). The negative control grid bias for V4 is obtained from the voltage drop of the total current drawn from the anode battery via resistor R11. With a total current of slightly less than 10 mA and a resistance value of 470 ohms for R11 the grid bias of V4 amounts to –4.5 V. Since the trick involves feeding the rest of the circuit through R11, the high voltage supply for the circuit proper has to be adequately decoupled, which is done with battery reservoir capacitor C19.

Cleaning and degreasing

As mentioned before, the set was in a reasonable condition but in bad need of cleaning and degreasing. Since I wanted to inspect the circuit anyway, I first removed the chassis from the carrying case, which was a simple matter of disconnecting the wires to the frame antenna and the output transformer, and removing two screws securing the chassis to the wooden case.

Though dirty and greasy, the chassis fortunately didn't show any signs of rust. Especially the components and wires at the bottom side of the chassis were dirty (**Figure 4**). The underside of the IF coils were greasy as if the wax used on the ferrite core inside had found its way to the bottom under intense heat. The whole set by the way looked as if it had been stored in a very hot place. The plastic knobs had warped and to some extent lost their original shape (**Figure 5, right**). The grease and dirt were easily removed with some methylated spirit and cotton tips.

Next, the knobs were removed from their spindles. This was a bit difficult since everything was caked and a slightly rusty. Considerable pulling force had to be used, and it is a small miracle that nothing broke. After removal of the knobs the front, or if you like, top plate could be removed. Unfortunately, intensive use of the set over the years had left its traces. After removal of dirt and grease the front plate showed quite a few scratches, but fortunately not so many as to draw immediate attention. I have considered touching up the spots with a dab of paint, but I think I will leave it this way because it also gives it a bit of an authentic look (**Figure 5, left**).

The knobs were grimy, especially the knurled edges. With a sharp needle every groove was cleaned individually and the whole thing was given a good rubdown with an old toothbrush and some mild detergent. The result was that they looked like new. The Plexiglass (Perspex) tuning dial was heavily scratched. With the help of a soft cloth and some good old "Brasso" metal polish and a bit of patience, it was possible to remove most of the scratches. The same treatment was given to the metal hinges that hold the plastic carrying strap. After a good vacuum clean of the interior, the P87BQ was in a "presentable"

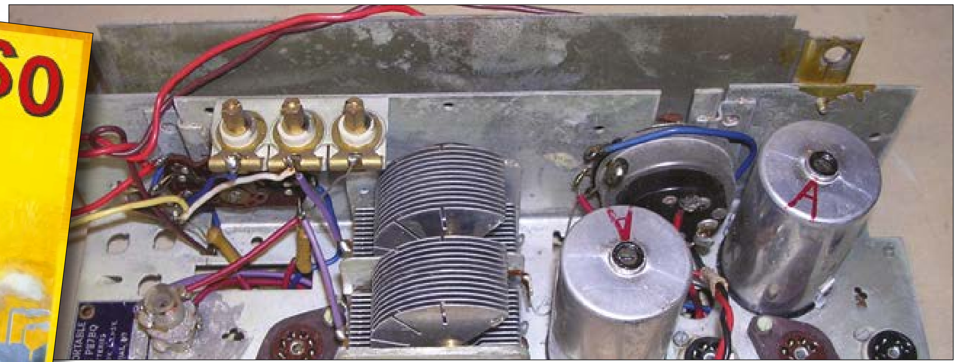


Figure 6. Top side of chassis after cleaning.

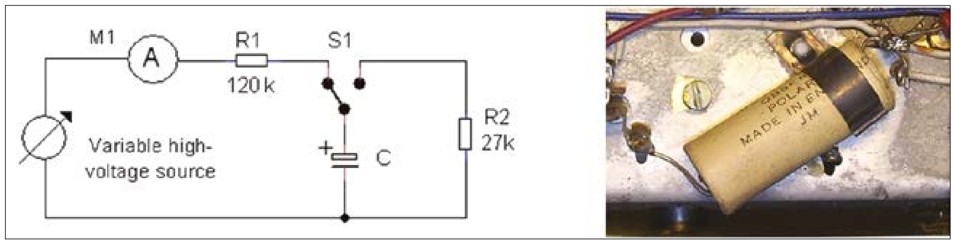


Figure 7. Simple circuit for forming electrolytic high-voltage capacitors (left), and the battery reservoir capacitor C19 (right).

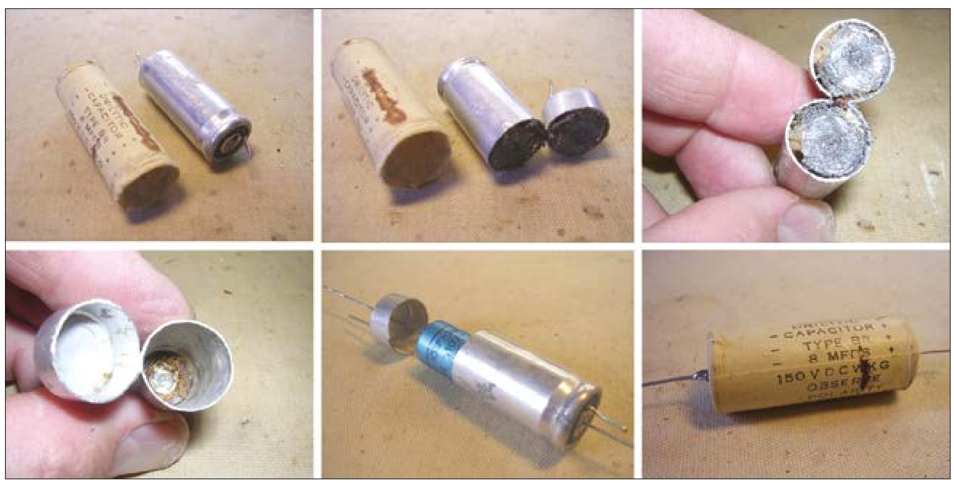


Figure 8. Replacing the old electrolytic capacitor with a new one, yet retaining that vintage look and feel.

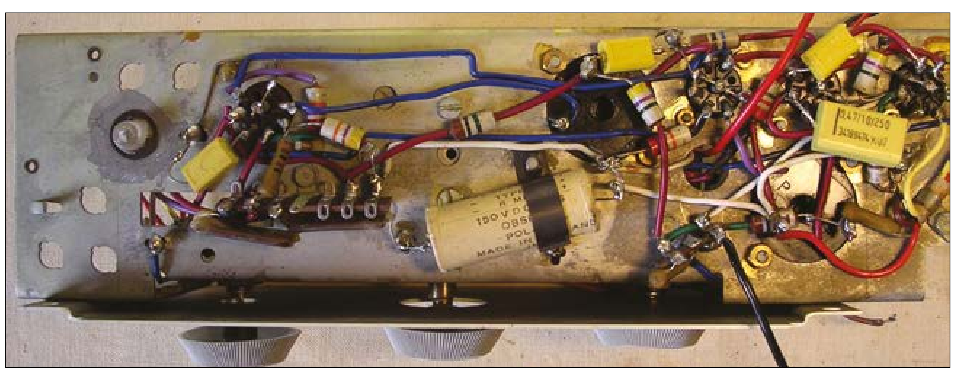


Figure 9. Underside of chassis cleaned and with new capacitors fitted.

Figure 10. The valves pulled from the P87BQ are stamped Mullard, B.V.A. (British Valve Association) **and** Made in Holland. A strange combination!

I inherited my love and enthusiasm for electronics from my father. My father was a war child. He had a pretty miserable childhood, and during the war there was a shortage of almost everything. The only radio my father's family had during the war was a simple homebuilt crystal set receiver which had to be stashed because radios were forbidden by the Germans. This explains my father's fascination for small and portable radio and later, television sets. A few years after the war, Philips introduced several "line-ups" of battery tubes in the small miniature envelope the Americans had developed during the war. Fascinated by the possibility of building a small portable radio with these tubes, my father saved more than a month's wages to buy a set of battery tubes. Hot from enthusiasm and anticipation, he built the small radio, but then he made a terrible mistake: eager to try the radio he accidentally swapped the 1.5-V filament battery with the 45-V anode battery. It was instant death for the tubes and a loss of a month's wages. There was no money for new tubes, and the radio never worked. I still remember how he told me this story when I was a child, and I remember how, as a child, I already felt deeply sorry for him. My father never was very lucky in his life, and this was just one of those sad moments. As a consequence of the affair, I got stuck with a fascination for these nice small tubes and whenever I find one, it feels as if I have won a prize. I have a nice small collection of these tubes, and I sometimes wish that my father was still around to see them.

state again, in principle. The top side of the cleaned chassis is shown in **Figure 6**.

Bringing the P87BQ back to life

After cleaning the chassis, it was time to make the old beauty produce music. A first quick check showed that the filaments of the four tubes were still intact and were drawing their nominal current. Although this says nothing about the emission of the filaments and/or other defects, it is a good start.

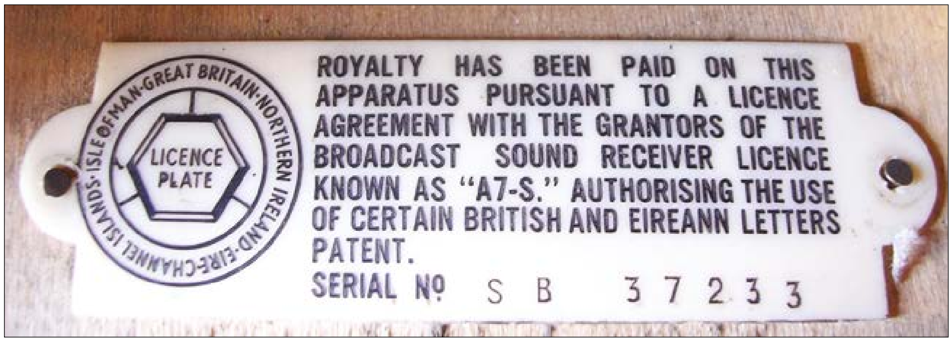
When dealing with vintage high voltage equipment, it is always a good idea to be cautious about switching on the high voltage (a.k.a. plate voltage) for the first time. The main causes for problems here are leaky or even short-circuited capacitors, especially the electrolytic ones. For the dielectric, these rely on a very thin layer of aluminum oxide. When an electrolytic capacitor has not seen use for a long time, the Al_2O_3 layer is reduced in thickness. When the capacitor is suddenly charged to its maximum voltage, the resulting current peak may be destructive. Fortunately in most cases it is possible to restore the thin dielectric layer by a process called "reforming." It requires an adjustable high voltage source, a resistor, and some patience.

The circuit I use to reform electrolytic capacitors is depicted in **Figure 7** (left). With the switch in the position as drawn in the figure, the high voltage source is connected to the capacitor via R1. I usually start by setting the high voltage to a low value, say 30 V. If the capacitor is still ok, the current will first surge, but as the capacitor charges, decrease to almost zero. I always wait until the current drops to a value below a few microamps. It is interesting to note how much time this takes. Inside the capacitor the thin oxide layer is now starting to be restored. Next the capacitor is discharged through R2. The process is repeated using a higher voltage with every step, until the maximum rated voltage of the capacitor is reached. The reforming process can be checked by charging the capacitor to the first voltage used in the recovery process. If everything is ok, the time needed for the current to decrease to a few microamps is a fraction of the original time.

Unfortunately, the original electrolytic capacitor in the P87BQ was already damaged beyond repair. Already at the lowest voltage the current just wouldn't decrease below a few hundred microamps. To keep the look of the radio as authentic as possible, I stowed a new buffer capacitor in the old capacitor housing (**Figure 8**). A neat little trick that I copied from the many websites covering the restoring of vintages radio sets. Beware, although the replacement capacitor is relatively new, it still needed forming!

With the reservoir capacitor replaced it was time to insert the tubes again and try the set! A 1.5 amp-hour type battery was used for the filaments, while the high voltage terminals were connected to my variable high voltage supply. With both voltages applied sadly the set remained silent while the total plate current was way too high: 22 mA instead of the 9.5 mA specified in the service sheet. The one probable cause is one of the tubes, in particular output valve V4, drawing too much current, and that in turns boils down to a defective tube, or a too high control grid voltage. A quick check revealed that the control grid voltage was about 5 V positive (!), while it should have been 4.5 V negative. It was not difficult to deduce that culprit is DC blocking capacitor C17 leaking current, which proved to be the case. It was one of those sticky, greasy looking capacitors. Paul Stenning on his *UK Vintage Radio Repair and Restoration Site* [4] states that this particular type of capacitor is a notorious troublemaker. For this reason I decided to replace all capacitors of this type with modern ones (**Figure 9**). Lo and behold, the radio now produced some sound, and it was even possible to listen to several radio stations, although the sound was spoilt by a nasty rattle.

Finding the origin of the rattle proved to be a bit more difficult. Before delving deep into the circuit of the radio itself, I wanted to eliminate the possibility that it was some external cause which caused the rattle. My "knutselkamer" (radio shack) is filled with equipment potentially causing interference one way or the other. However, with everything shut off, the rattle was there just as bad. Next I wanted to eliminate some fault in the tubes themselves. One by one all four tubes got replaced (**Figure 10**) and I was fortunate, to have a set of spare tubes in my collection. As it turned out though, the cause was different. The next question was, does the rattle emanate from the audio section or from the RF section? After removal of V1 and V2 the rattle was gone. So the audio part was ok. Also after IF amplifier tube V2 was replaced, the receiver remained quiet. However, as soon as V1 was in place again, the noise returned. To test the radio I had connected the frame antennas and the speaker with some 50 cm long wires. Could it just be that these long wires picked up the noise or caused some oscillations? To eliminate this possibility the chassis was returned to the carrying case, and the original wires were attached. Again, no result. And then I thought I noticed something strange. I had the impression that when I switched my external high voltage supply off, the rattle disappeared immediately, while the radio continued playing for another fraction of a second. Could it as simple as my (Dutch designed) Delta E0300-01 high voltage supply introducing the noise? I had always assumed that the Delta supply was just a simple linear supply, but possibly it had a switching preregulator? I quickly switched a number of normal power supplies in series and connected them to the radio's HT input — lo and behold, the rattle was gone and the set produced a nice and warm sound, be it with a hint of typical AM crackle!



Of course I had to find a method of powering the radio off AC but the original anode batteries of the 1950s obviously are not available today. To remain all portable I designed an electronic HV battery "ersatz" which employs 5 NiMH batteries for the HV inverter. The circuit is controlled by a PIC processor and may be found at [5] along with its design story.

(150149)

ESTᴰ 2004

www.elektor.tv

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

Web Links

[1] Mark Franklan, "Radio Man, The remarkable rise and fall of C.O. Stanley, IEE History of Technology series 30, ISBN 0-85296-203-7

[2] EF50 — The Story. Retronics, *Elektor* magazine December 2010, www.elektor-magazine.com/100657

[3] "Trader" service sheet no. 1184, supplement to Wireless & Electrical Trader, 5th February 1955. Download from www.savoy-hill.co.uk

[4] UK Vintage Radio, Repair and Restoration: www.vintage-radio.com

[5] http://dos4ever.com/battery/battery.html

For Further Reading

The History of the UK Radio Licence: www.radiolicence.org.uk/

Adrian's British "Battery Portable" Tube Radio Pages: www.portabletubes.co.uk

The National Valve Museum: www.r-type.org

DL96 on The Radio Museum: www.radiomuseum.org/tubes/tube_dl96.html

John's Radio Web: www.hupse.eu/radio/tubes/IndexValves.htm

Compiled by **Beatriz Sousa**

Jaycar partners up with Elektor

Elektor's Carlo van Nistelrooy (right) during his visit of beautiful Australia was privileged to get not just a warm welcome but also VIP treatment from Jaycar owner Gary Johnston. Carlo was first shown a Jaycar store and the company's offices and then enjoyed a tour of the impressive distribution center in Rydalmere, NSW in the coolest tuned up golf cart! Gary started Jaycar back in 1981 and developed the company into Australia and New Zealand's premier electronics and component retailer. He is a long time, enthusiastic Elektor reader and in his offices signed a partnership deal with Elektor initially aiming to work together in a number of areas.

Microcontroller Design Contest 2015

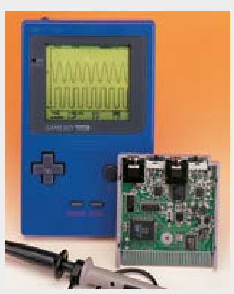
ARM and Elektor launched a Design Contest with $10,000 in cash prizes in cooperation with controller manufacturers ST, NXP, Freescale & Infineon. We had more than 1000 participants from over 80 countries and the STMicroelectronics STM32F429 DiscoveryKit turned out their favorite board. The 400 best design proposals got selected and received their chosen board for free. Now it's time to start working on the winning projects, there will be $10,000 in cash prizes — the winning article will be published in Elektor magazine edition 6/2015 (November & December).

READ ONLY MEMORY

Elektor magazine and its parent publishing company boast a long and rich history. In this space we picture a gem from the past.

Elektor GameBoy DSO Blockbuster

At the end of the previous century, portable games consoles like the GameBoy were hugely popular with kids. Elektor caused a stir back in October 2000 with the publication of a GameBoy cartridge that transformed the "toy" into a simple two-channel digital oscilloscope. Initially only presented as an all-DIY project, a primitive Dutch auction on the web proved the huge demand for ready-assembled units, making the Gameboy DSO the first completely assembled Elektor project. **Wildly successful, GBDSO sales reached 4K7 ±20%.**



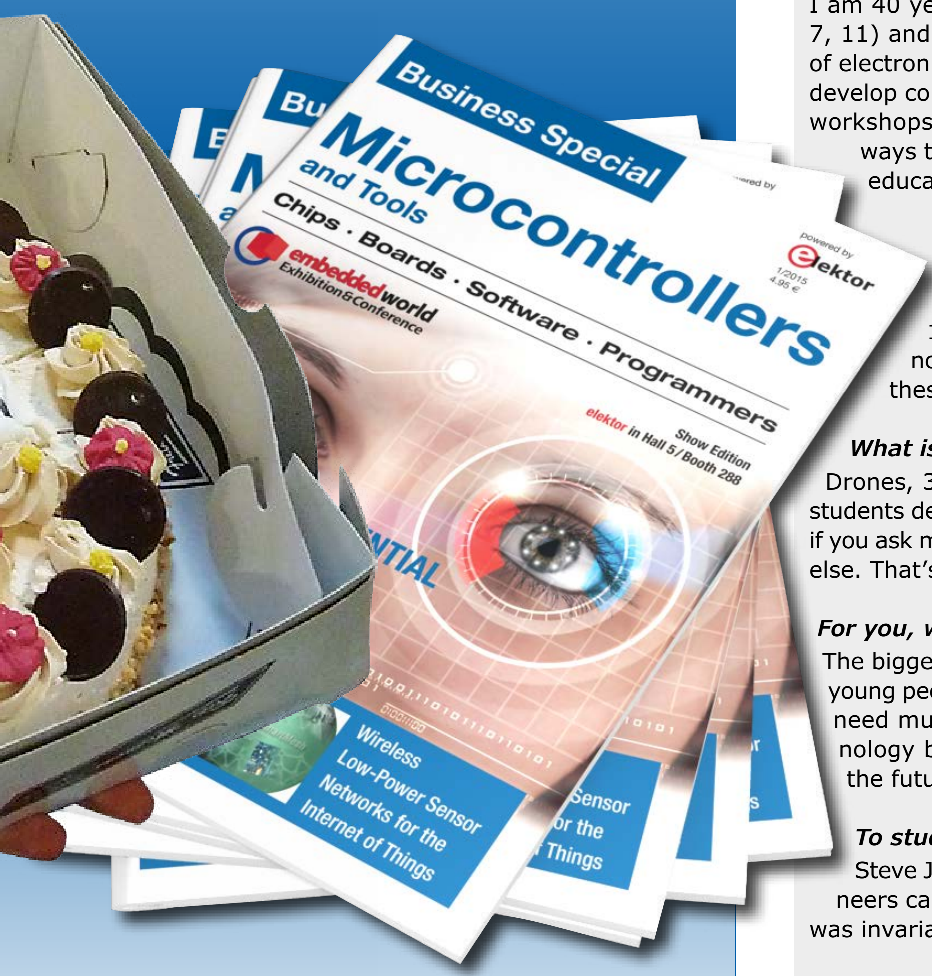
... Piece of for embedded

The world's largest conference for embedded engineers took place in Nuremberg, Germany, last February. Roughly 25,500 engineers, scientists and electronics businessmen crowded the conference rooms and exhibition halls. Thousands of dinners took place in and around the old city. Considering that Elektor magazine is published in more than 80 countries, a *Business Special* was produced for the occasion, a booth was built and several visitors were treated to a piece of Elektor cake. Professor Matthias Sturm, leader of the conference program was mighty pleased with his cake and — more importantly — the results:

PEOPLE NEWS • Sarah Quilter joined the English team to help the growing number of companies that productions • Julia Grotenrath and Margriet Debeij are working on 74 proposals on request of electronic Romagnoli joined us to help French companies that want to bring their knowledge to more than 25,000 Chief, Tessel Renzenbrink represented Elektor at Mobile World Congress, one of the largest conferences in the

Cake engineers ...

***"I think it is great how Elektor retains its vitality after 60 years on the market. The teams that are your lab and editorial departments manage to keep up with all innovation"*. Thanks to this, the Elektor community is expanding not only in Europe and the Americas, but now also in Asia. One of our sponsors said: "This success is of course very welcome in our campaigns to mobilize more engineers from these regions to visit our booth. Next year we will contribute to your new trade magazine Elektor Business for sure. Great that your are the first and only publisher to deliver a business magazine to home addresses."**



want proposals from Elektor for sponsorships and co- companies that participated in Embedded World • Fabio French Elektor members • Tech the Future Editor in world about mobile technology, in Barcelona, 2-5 March.

EXPERT PROFILE

Elektor works closely together with more than 1,000 experts and authors for the publication of books, articles, DVDs, webinars en live events. In each installment of Elektor Word News we put one of them in the limelight.

Name:
Bart Huyskens

Age: **40**

Education: **Electronics**

Publications:
Several books, courses and video lessons on embedded electronics. See www.e2cre8.be for more info.

Hardware: **Brainbox Pro, Brainbox Fun, Brainbox Junior, Elektor Proton Robot, Formula Flowcode Buggy...**

Who is Bart Huyskens?

I am 40 years old, happily married with 3 children (ages 4, 7, 11) and the biggest part of my job is to teach the basics of electronics to 16-18 year old students. In my 'free' time I develop courseware, video lessons and hardware. I organize workshops for teachers and professionals and I try to find ways to motivate young children to opt for STEM related education.

How many hours of education have you produced so far?

Quick count: over 10,000 hours in school and over 100 8-hour workshops for professionals and that's not taking into account the 5 sets of video lessons — these are used in all Flemish technical schools.

What is the easiest part to keep students' attention?

Drones, 3D printers & controlling hardware with apps that students develop themselves is hot currently but I'm sure that if you ask me the same question in 3 years, it will be something else. That's what makes teaching electronics so much fun.

For you, what triggers the future of teaching?

The biggest challenge will be to find ways to motivate more young people to opt for STEM related education. We urgently need much more technological experts to drive our technology based economy and to solve the big questions of the future.

To students, who is THE 'role model' engineer?

Steve Jobs without any doubt. He was one of the few engineers capable of combining technology with design and he was invariably one step ahead of the competition.

Should the EU develop European based education?

Technological education programs need to evolve really quickly to keep up with the fast changing world of technology. Different regions in Europe also have different needs and expectations of education. I think that the technological programs had better remain a regional matter and that the role of the EU is more in the organization of proper interaction between all these technology teachers and students.

Hexadoku The Original Elektorized Sudoku

To electronics minded people it's plain evident that hexadecimal numbers enable lots of combinations to be forged, and this month's freshly brewed puzzle just goes to show. In every edition we present a new challenge for you to crack. 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 June 1, 2015, supply your name, street address and the solution (the numbers in the gray boxes) by email to:
hexadoku@elektor.com

Prize winners

The solution of the March & April 2015 Hexadoku is: **16A8E**.

The €50 / £40 / $70 book vouchers have been awarded to: Peter Budts (Belgium), Gerald Schönecker (Germany), Patrick Ferrari (France), Seairth Jacobs (USA), and Olli Hakala (Finland).

Congratulations everyone!

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

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