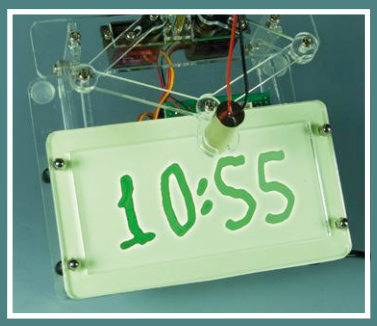
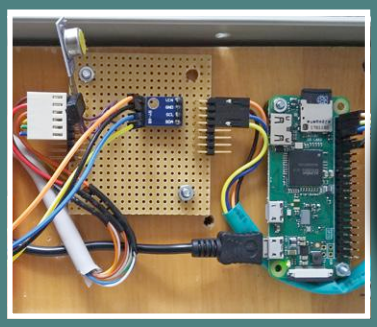
Edition 1/2018 | January & February 2018
www.elektormagazine.com

elektor

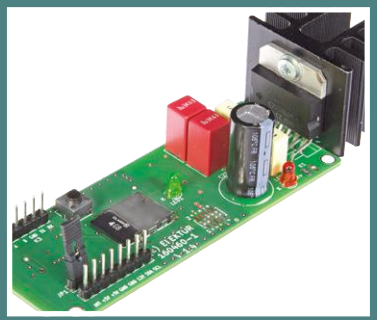
LEARN DESIGN SHARE



Laser Time Writer
Writing with Light



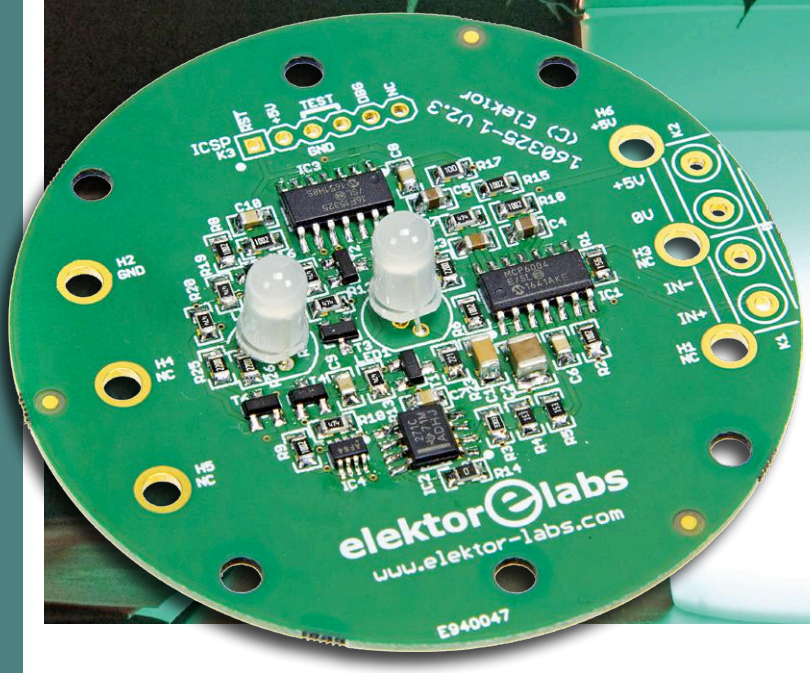
Weather Station
With no moving parts



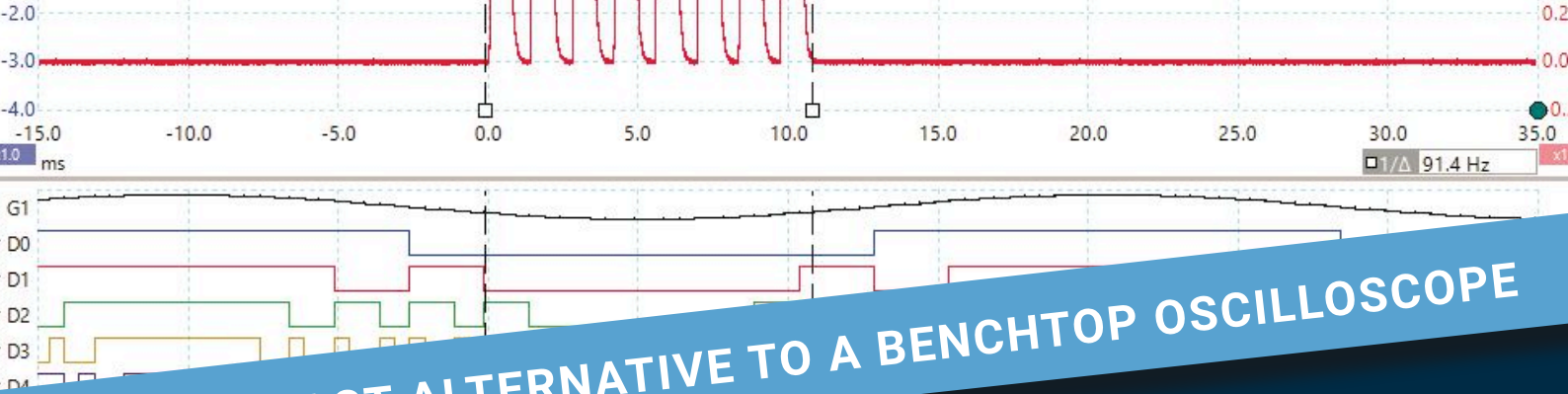
Card Sound
Plenteous audio
from a tiny micro SD Card

Bio-Light

Plant expresses 'feelings' through colours



Arduino Experimenting Shield 2.0 • Bi-colour Transistor Tester • **Bio-Light** • Card Sound • **Communicating Current Loops** • Connecting an LCD to an Rpi • **DoubleSPIder** • Elektor Labs Pipeline • **Elektor Store** • EPS — Easy Parking System • **Err-lectronics** • ESP8266 on the Android I/O Board • **Everything (almost) you need to know about small displays** • Get Busy with the CAN Bus • **Hexadoku** • Laser Time Writer • **MicroTesla Music Synthesizer** • New Life for a Failing Monitor • **OLED Displays** • Open the Radio Spectrum! • **Peculiar Parts: the Intersil ICM7216** • PlatformIO, the Universal Programming Tool • **Retronics: Uher Report 4000L (1964)** • Skip! • **SmartPi Reviewed** • Solar Power for Wi-Fi Repeater • **The EEBUS is on its Way** • USB Programming Adaptor for ESP8266 • **We have Moved!** • Weather Station • **Wi-Fi Desktop Thermostat** • ... and more



THE COMPACT ALTERNATIVE TO A BENCHTOP OSCILLOSCOPE

pico® Technology

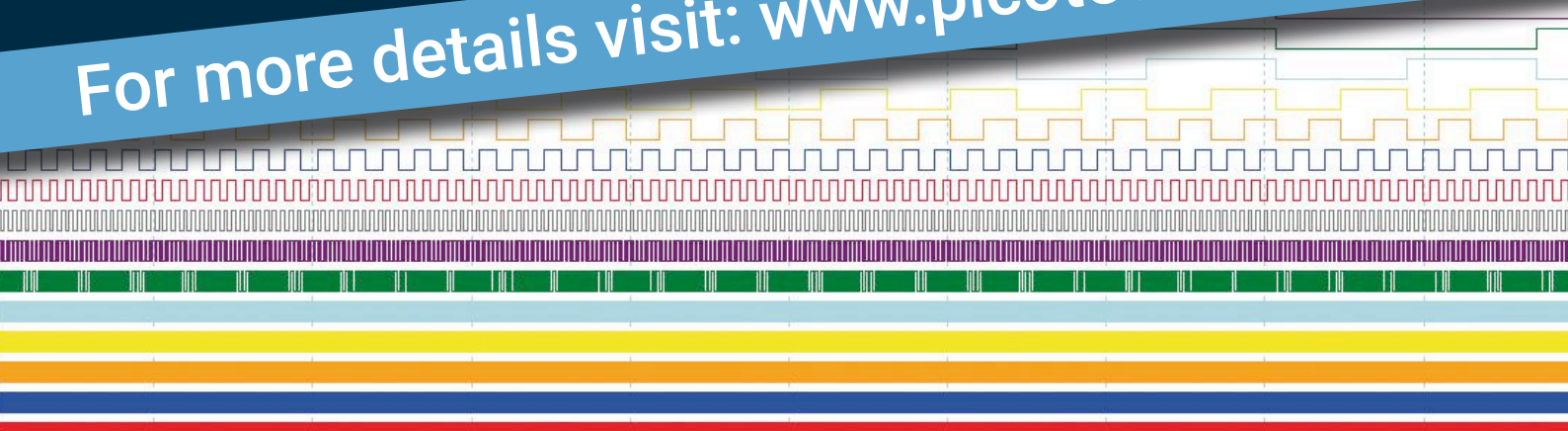
PicoScope®

2000 Series Oscilloscope

From £79

- 2 channel, 4 channel and MSO models
- Decode 16 serial protocols as standard
- Up to 100 MHz bandwidth
- Up to 128 MS buffer memory
- USB connected and powered

For more details visit: www.picotech.com/A10



Elektor Magazine
Edition 1/2018
Volume 44, No. 487
January & February 2018

ISSN 1757-0875 (UK / US / ROW distribution)

www.elektor.com
www.elektormagazine.com

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

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

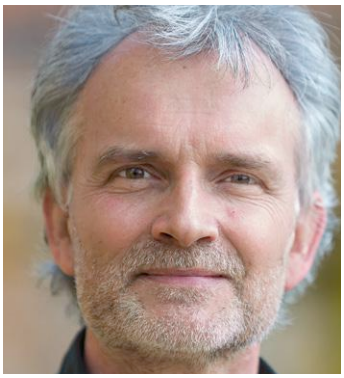
Head Office:
Elektor International Media b.v.
PO Box 11
NL-6114-ZG Susteren
The Netherlands
Phone: (+31) 46 4389444
Fax: (+31) 46 4370161

Memberships:
Please use London address
E-mail: service@elektor.com
www.elektor.com/memberships

Advertising & Sponsoring:
Margriet Debeij
Phone: +49 170 5505 396
E-mail: margriet.debeij@eimworld.com

Benoit Simoneau
Phone: +44 7891 920 370
E-mail: benoit.simoneau@eimworld.com

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



Logic low on the $\overline{\text{BUSY}}$ line

Halcyon days when an Elektor editor or lab worker could hand in copy on paper to the typesetters 'downstairs' and then patiently wait for the results of the publication in terms of sales realized, or cries from the Complaints Desk. A lot had to be done before the first board or micro actually reached the customer, and a lot of time it took! After all, there was the page layout to wait for, followed by two or more correction rounds, then approval for printing, pagination, printing and magazine distribution — all in due sequence with lots of Red Tape in between. Then a period of ominous silence during which readers apparently made up their mind whether or not to jump the bandwagon and order the relevant item. By post, of course. Until about 20 years ago, the 'time-to-market' lag was such that we had no useable feedback on how specific projects were faring until about six months after publication, to which another two to three months should be added ahead on account of the production by Elektor's designers and writers.

These days we have the web to enable rapid communication with our readers and suppliers, as well as flash dashboard systems that will tell us what's a hit or a flop. However, none can securely forecast the response of Elektor readers to a published project. It's a sure cause not just for a lot of excitement and guesstimates on our part but also frustration occasionally.

For example, no one on the small teams normally formed by Elektor staff and supplier representatives could have foreseen the overwhelming interest in the Avnet A6 3-D printer, the SandClock or the Bio-Light projects, just to mention three hits from our magazines and weekly newsletter. Of these, the Avnet A6 is causing the most concern as the supplier is challenged by order volumes beyond his most optimistic forecasts. The process of ordering, stocking, retailing and shipping these goods to our customers is fairly smooth in most cases but occasionally subject to delays in the megaseconds range, drift, noise and tolerances you can't believe existed as an electronicist. On such occasions, faced with lead times, we realise the truth and reality in Elektor's byline "more than just a magazine".

Jan Buiting, Editor-in-Chief

This Edition

Volume 44 – Edition 1/2018
No. 487 January & February 2018

Regulars

Card Sound

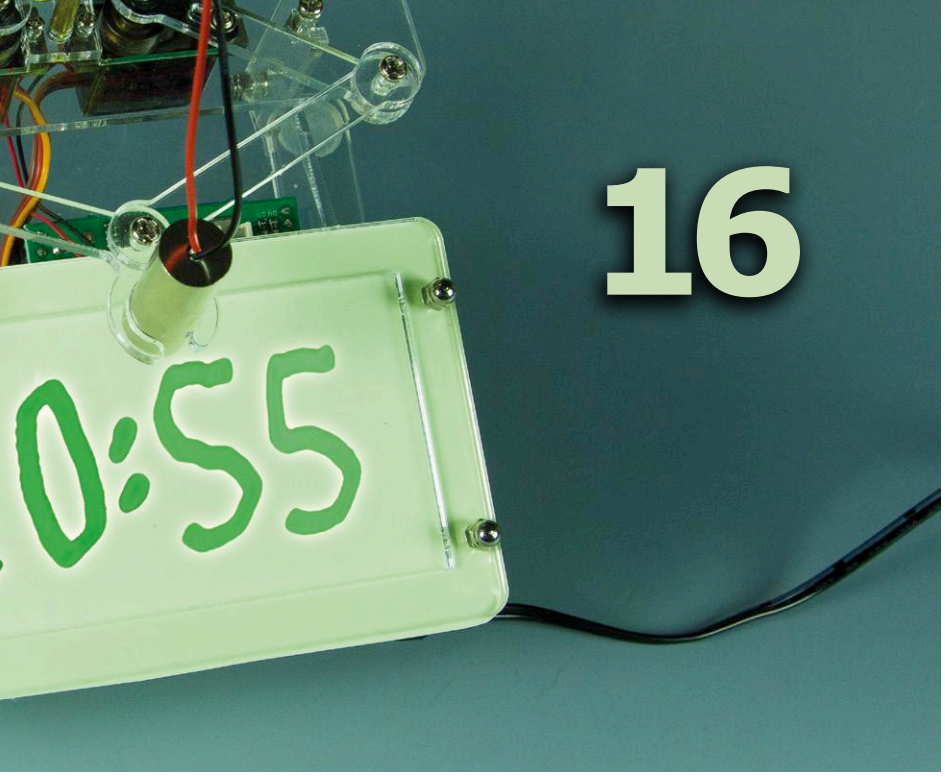
Plenteous audio from a tiny micro SD Card

With a smartphone or an MP3 player it's easy to play audio files, but generally not at room strength or if you can, only in wretched quality. The Card Sound project makes it possible to fill the room with music and have excellent sound reproduction! This lightweight and compact player also stands out for its modest quiescent current consumption and desirable extra features such as I²C and RS-485 control.

Features

Projects

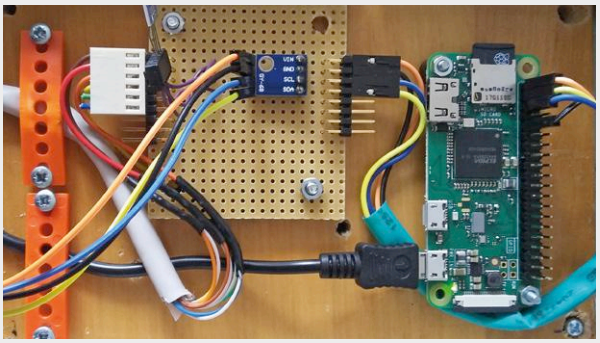
The EEBUS is on its Way



Weather Station

With no moving parts

54



Many aspects of our daily lives are determined by the weather: whether or not you take an umbrella with you, whether you walk to work or take the car, whether you close the window or leave it open, whether you stay in bed or get up, and so on. Even though a glance out the window can tell you a lot about the weather at the moment, you need data to help you estimate what it will be later – and that's where the weather station described here comes in handy.

A voice of sanity amid the babble of IoT

elektor magazine

Next Editions

Elektor Magazine 2/2018

Multi-Timer • EMC Issues with LED Lamps• Q & A: Nixie Tubes • FPGA Starter Kits • KiCAD • CAN Bus Debugging Tool • Distortion Measurement at 10 MHz • Close-to-Universal IR Remote Control • DIY Solder Station • Video Olympics Results • 1 MHz – 10 GHz RF Power meter • Universal I²C bus isolator and level adapter • and more.

Elektor Magazine edition 2 / 2018 covering March & April is published around 20 February 2018. Delivery of printed copies to Elektor Gold Members is subject to transport.
Contents and article titles subject to change.

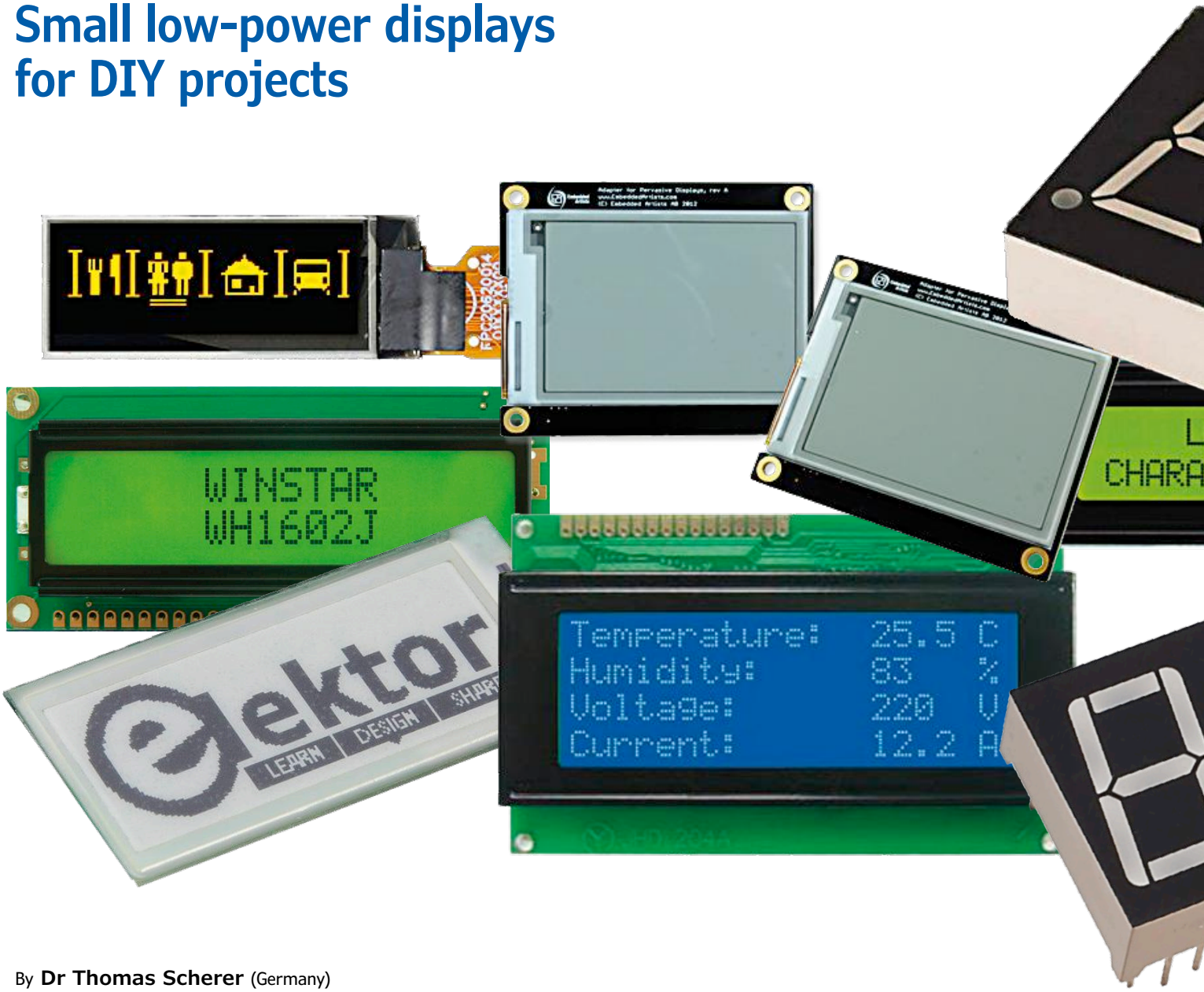
Elektor Business Magazine 1/2018

Elektor Business Magazine edition 1/2018 has a focus on Embedded, Programming & Tools and is a special release for the embedded world trade show and conference in Nuremberg on 27 February – 1 March 2018.
Among the contributors are companies and research institutions including Microchip, congatec, STMicroelectronics, PEAK System, Factorylab, Intel FPGA, Bytesnap, Cypress, Infineon, Keil und ARM. Plus you'll find fresh instalments of all the EBM regulars like Infographics, Operation Marketing, Our Business, and Business Store.

Elektor Business Magazine edition 1 /2018 is published on 15 February 2018 to Elektor Magazine Gold members in print, and Elektor Green members as a pdf download. The edition is also available for purchase at www.elektormagazine.com.

OLED Displays

Small low-power displays for DIY projects



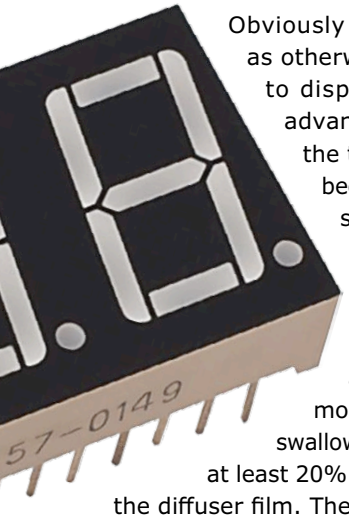
By **Dr Thomas Scherer** (Germany)

Television sets with OLED screens are gradually becoming affordable, and some high-end smartphones already come with OLED displays as standard – even Apple has now joined the crowd with the new iPhone X. Although computer monitors are still equipped with conventional LCD technology, there are lots of small low-cost OLED displays available. They can be connected to a microcontroller through a serial interface and easily used in your own electronics projects.

For a long time, television sets and monitors were all equipped with cathode-ray tubes, but now that is ancient history. At the turn of the millennium there was a shift to flat-screen liquid-crystal displays, which consume less energy but still do not qualify as true low-power devices. That's because LCDs do not generate their own light, but only change the polarisation of light shining through them. This means that an LCD pixel, in combination with a polarisation film, can only change the brightness or intensity of the light passing through the pixel. In the early days this light was usually generated by fluorescent lamps. Around a decade ago, white LEDs (or even RGB LEDs in luxury models) took over this backlight task. LCD technology was initially hampered by a variety of shortcomings, but it

quickly evolved to the point that modern monitors and television sets are very good and difficult to beat. But as the saying goes, better beats good, and the OLED revolution is now being ushered in by a variety of displays.

The first low-cost, mass-produced OLED displays are simple devices with small dimensions, and in most cases monochrome. New technology is usually more expensive than proven technology, and with small screen dimensions the display cost is a relatively minor factor. It's thus not surprising that OLED displays first appeared on mobile phones, and only later on television sets with their larger screens. For several years, even smaller OLED displays have been produced in Asia for fitness trainers and similar devices. Now electronics professionals and hobbyists who want to equip small product runs or prototypes with modern displays are benefiting from the expansion of production capacity for these mini displays. Now you no longer have to wrestle with LCDs or (heaven forbid) 7-segment displays. On eBay you can find hundreds of modern OLED displays at attractive prices; a typical example is available in the Elektor Store [1].



OLED advantages

Obviously OLEDs have to do something better, as otherwise they would not have the potential to displace LCDs in the medium term. The advantages of OLEDs are directly related to the technological weaknesses of LCDs. This becomes clear when you compare the basic structures of LCDs (**Figure 1**) and OLEDs (**Figure 2**).

Efficiency: The luminous efficacy of modern semiconductor LEDs is very good and still distinctly better than OLEDs, but most of the light from the LCD backlight is swallowed up in the LCD structure. To start with, at least 20% of the light is lost in light distribution by the diffuser film. Then the polarisation filter basically blocks at least 50% of the remaining light. Furthermore, the light transmission of the LC material in fully on (bright) pixels is never 100%, even if they block virtually all of the light in the off state. On top of that there are the electrode layers, which even though they are very thin are not completely transparent. Last but not least, the colour filters in front of the individual pixels (not shown in the figure) only transmit light in narrow spectral bands. The result is a net transmission percentage in the single-digit range. OLEDs do not have a polarisation filter or a diffuser, and the light only has to pass through one electrode layer. Another significant factor is that OLED pixels only consume power when they emit light. The overall efficiency of an OLED display is therefore significantly higher, so the power consumption is only a fraction of the power consumption of an LCD with the same brightness.

Contrast: In an LCD the active layer rotates the polarisation of the light depending on the voltage applied to the electrodes.

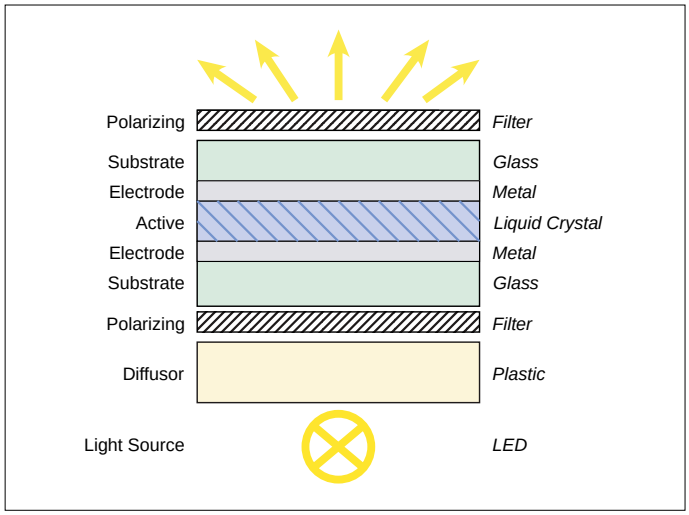


Figure 1. Basic structure of an LCD.

It is difficult to do this so precisely that all the light is blocked. In addition, the various layers of the LCD reflect part of the ambient light. It is therefore nearly impossible to achieve extremely high contrast (the ratio of bright to dark) between adjoining pixels. Although large areas can be made darker in "direct lit" displays by dimming the corresponding backlight LEDs, this technique does not completely solve the problem. An OLED pixel, on the other hand, is completely dark when no current flows through it. The rear surface of an OLED display can also be blackened to reduce the reflection of ambient light.

Viewing angle: Although the latest LCDs have advertised viewing angles of up to 178°, you can easily see for yourself that the brightness decreases at large viewing angles, and in some cases not uniformly for all colours, leading to angle-dependent hue changes. If you look at the structure shown in Figure 1, it's easy to understand that these effects are inherent in the design and cannot be eliminated completely, even with very thin layers. Things are a lot easier with OLEDs. Light emission in the top layer is basically the same in all directions,

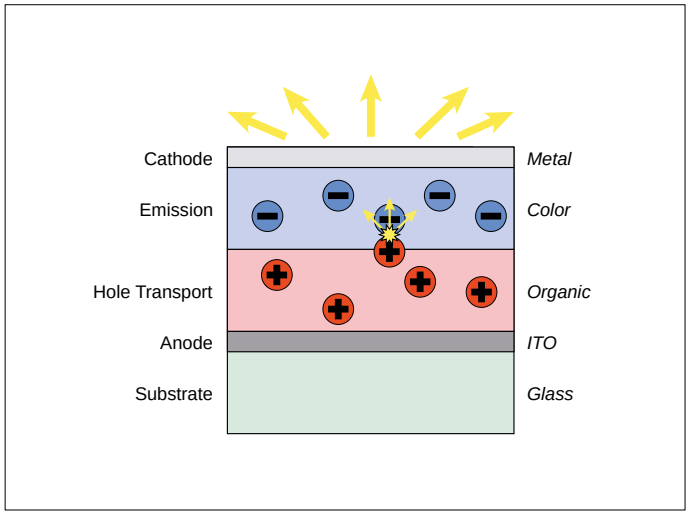


Figure 2. Basic structure of an OLED display.

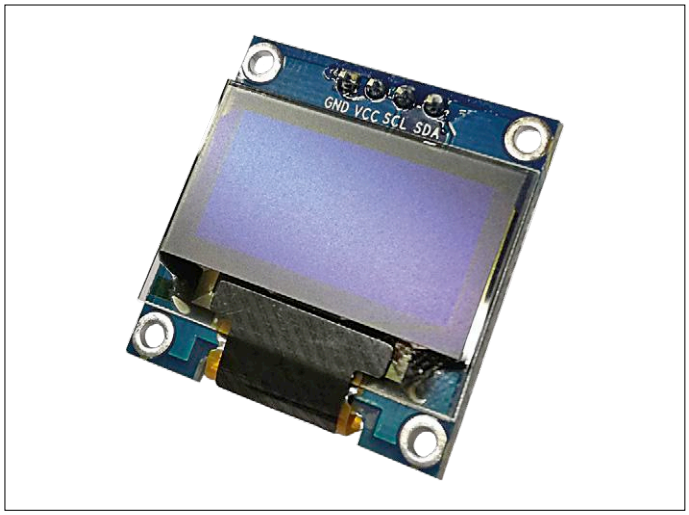


Figure 3. The OLED display from the Elektor Store, with a diagonal of 0.96", 128 × 64 pixels and an I²C interface.

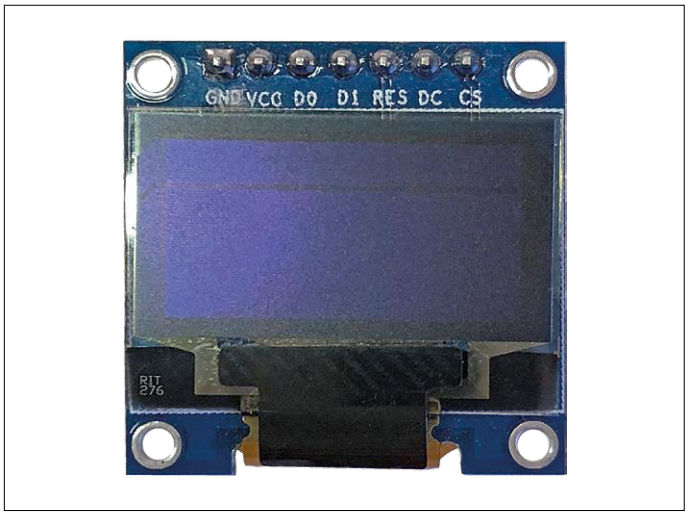


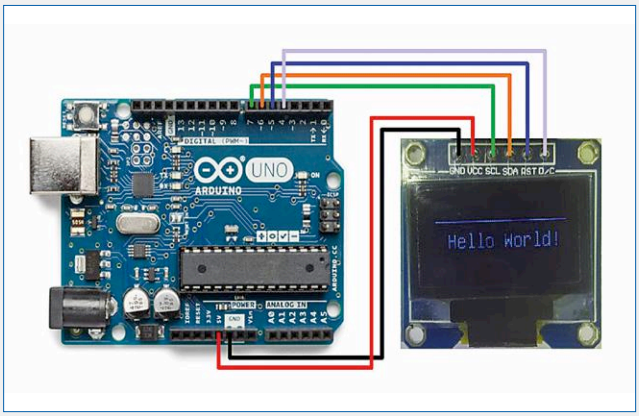
Figure 4. Direct import from China: an OLED display with I²C and SPI interfaces.

OLED Display and Arduino

It's easy to connect a standard OLED module to any desired microcontroller board, because an I2C or SPI serial interface is always available.

The picture hows how to connect the 0.96" OLED display from the Elektor Store to a standard Arduino Uno via the I2C bus interface. Simply make the connections listed in the following table:

No.	Display	Arduino
1	GND	GND
2	VCC	5V
3	SCL	7
4	SDA	6
5	RST	5
6	D/C	4



Connecting the standard OLED display from the Elektor Store to an Arduino Uno.

Next, download the library *U8glib.h* from [3] and add it to the IDE. You can read the descriptions in the User Reference Manual [4] to learn how to use the individual functions, initialise the display, and conveniently render lines, curves, text or images on the display. You can also download the "Hello World" demo sketch from the web page for this article [5].

enabling a nearly 180° viewing angle without special effort, and with minimal brightness reduction or hue changes when viewed from the side.

Response time: In a liquid crystal device, movement of a physical substance is necessary to change the polarisation direction. That takes time, and despite many design tricks in the drive circuitry, it cannot be made arbitrarily fast with the available power. An LCD with a true frame rate of several hundred hertz is virtually impossible in practice. The response times of OLEDs are at least an order of magnitude less, which means that in theory it would even be possible to make an OLED display with a frame rate of 1 kHz – which nobody actually needs.

Disadvantages: Every upside has a downside, but with OLEDs it is not as severe. The main difficulty is the stability of the organic material. Semiconductors age more slowly and are more robust. However, OLEDs have significantly improved in recent years, particularly with regard to long-term stability. The electronics industry simply has much more experience with semiconductor technology than with organic chemistry. With continuing progress in fabrication, it can be assumed that in the medium term OLED displays will actually cost less than LCDs because cleanroom facilities are not necessary for OLEDs. At that point LCDs will be a dying breed. Of course, this forecast could turn out to be wrong if advanced LED matrix displays using tiny semiconductor LEDs make rapid progress and start displacing not only LCDs but also OLEDs in the coming years.

Standard display

Now let's focus our attention on simple OLED displays with serial interfaces for connection to microcontroller boards or development boards. As previously mentioned, you can find hundreds of monochrome OLED displays with diagonal dimensions between 0.6 and 3 inches at prices from less than €5 to €50 (£4.50–£45.00 / $6.00–$60.00) by entering the string "OLED display" in the eBay search box. Nearly all of them are graphic LED matrix displays with typical resolutions of 64 × 48, 128 × 32, or about 256 × 64 pixels. They are driven via

conventional serial interfaces, such as SPI or I²C. Some display modules even support both interfaces. Electrical connection is very simple, requiring only four to six lines including power and ground.

Figure 3 shows a "standard" display. It has a diagonal of 0.96", a resolution of 128 × 64 pixels, and an I²C interface. Displays of this sort are readily available. With direct import from China via eBay, Alibaba or the like, you can easily get them for less than €10 or the equivalent in other currencies. However, for that you have to wait a bit longer and you run the risk that the package is inspected by a customs officer in a bad mood who decides that the value declared by the Chinese sender is unrealistic. Then you receive an invitation in the mail to visit the nearest customs office in person to complete the formalities and provide proof of the declared value. At least in Germany, VAT (sales tax) is due on all goods with a value of €20 or more, as well as customs duties in some cases. That might be acceptable, but if the nearest customs office is a good 120 km away from home (as it is in my case), then the display is not worth the cost of the trip, so it remains at customs and I sit at home with a sad face.

If you want to be on the safe side, you should order from an inland (domestic) supplier or at least a supplier in a neighbouring (EU) country. That is faster and easier, although it costs a bit more. For example, a display of this sort costs €14.95 (or €13.46 for members) in the Elektor Store [1]. Despite all this, I decided to try the "Chinese Way of Shop-Ping" and ordered a module with combined I²C and SPI interface (**Figure 4**) for €7, including shipping. I was lucky — about three weeks later a grey plastic bag with intact content was in my mailbox. As shown in **Figure 5**, this 0.96" module has a bi-colour split display, with one part blue-green and the other part white. Virtually all of these displays are fitted with Solomon SSD1306 controller chips [2]. The modules can operate with a supply voltage between 3.3 and 5 V and draw about 10 mA. If you google SSD1306, you get many pages of hits with links to pinouts and application data, as well as libraries for Arduino or even Bascom. These modules are therefore ideal for status and message display in DIY projects, prototypes or small product runs.

See the **OLED Display and Arduino** inset for information on how to connect the module to an Arduino Uno and what you need on the software side. Thanks to the wealth of open-source software, you don't have to reinvent the wheel and control every pixel yourself — instead you can access all sorts of tested drawing functions.

Other OLED displays

Aside from the quasi-standard module with 0.96" display, there is a half-size version with a resolution of 128 × 32 pixels and a diagonal size of 0.91". An example is shown in **Figure 6**. The price is the same as for the full-size version, and it is also based on the SSD1306 IC. Due to the different horizontal pixel count, you have to be careful if you use libraries for the standard module. The standard module is also available in somewhat larger sizes, such as 1.3" or 1.54". That makes the razor-sharp

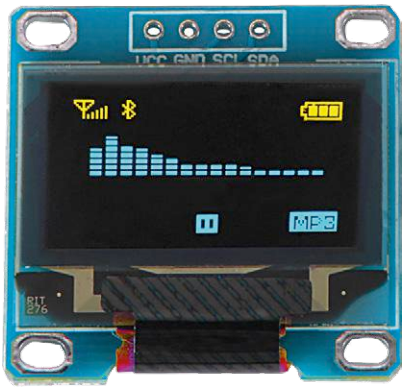


Figure 5. Active bi-colour OLED display with demo content.

Figure 6. An OLED display with a diagonal of 0.91", 128 × 32 pixels and an I²C interface.

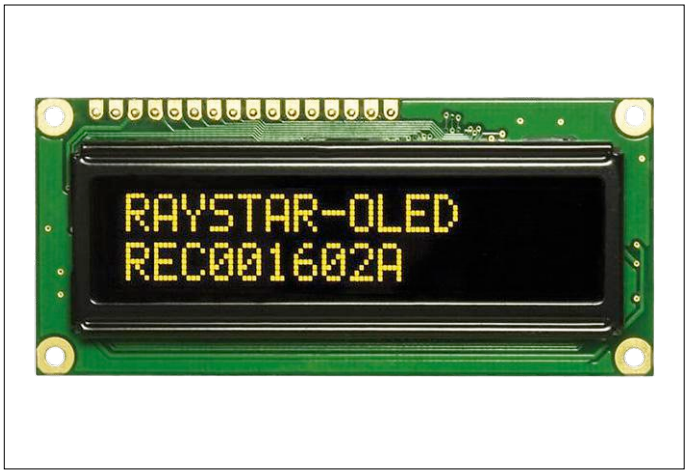


Figure 8. An alphanumeric OLED display with 2 rows of 16 characters, suitable as a replacement for conventional alphanumeric LCDs.

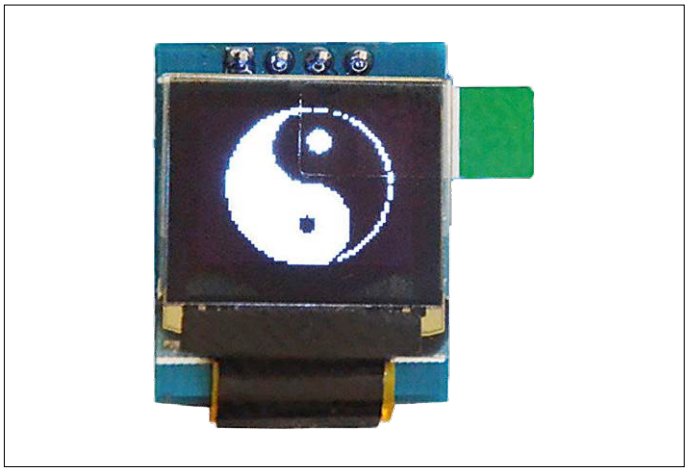


Figure 7. An OLED display with a diagonal of 0.66", 64 × 48 pixels and an I^2C interface.

but rather small characters a bit easier to read, at the cost of slightly higher current consumption. **Figure 7** shows a tiny version with a diagonal of 0.66" and 64 × 48 pixels. As you might have guessed, it also has an SSD1306 controller chip. Naturally, in China you can also find other display modules for just about any imaginable electronic device. However, with these special types it is often difficult to figure out what sort of controller they have and to get your hands on sufficient technical data. There are also a lot fewer ready-made projects in the community for them than for the standard SSD1306 controller. Sometimes these displays come with a ribbon cable instead of pins or solder pads, which means you first have to find a matching connector. With SSD1306 modules, by contrast, you are on the safe side and there's little chance of anything going wrong.

OLEDs: razor-sharp display and low power consumption

Of course, there is an exception to the SSD1306 rule: there are alphanumeric OLED displays (**Figure 8**) with two or four rows of 12, 16 or 20 characters in white, blue, yellow or red on a black background, with display diagonals from 2.5" to 6". They can be used as replacements for conventional alphanumeric LCDs, with the advantage of much better readability and significantly lower current consumption (LCD backlights draw a fair amount of current). However, this luxury comes at a price — starting at more than €20 / £17 / $25 and ending well above €50 etc. for the larger types. That puts you above the duty-free limit, so you have to live with the associated inconvenience. These displays are also based on a Salomon controller – the SSD1311 chip with integrated character generator.

Summary

It's never been easier to equip a microcontroller project with a monochrome, full-graphic display at low cost. OLED displays are razor-sharp and well suited to battery operation thanks to their low current consumption. However, these projects are only easy if you have a display based on the quasi-standard SSD1306 controller chip. There's more information about this chip available on the Internet than you would ever need. The display modules are especially low-cost of you import them directly from China. However, if you do not want to wait for several weeks or risk paying customs duties, there are many options available closer to home at reasonable prices. If you have an Arduino or other development board, you should certainly treat yourself to one of these modules, because you can always use a nice display.

(160550-I)

Web Links

[1] **OLED display in the Elektor Store**: www.elektor.com/3d-printer-head-and-mat-temperature-controllerboard

[2] **SSD1306**: www.solomon-systech.com/en/product/display-ic/oled-driver-controller/ssd1306

[3] **Arduino library**: https://github.com/olikraus/u8glib

[4] **Reference manual for library functions**: https://github.com/olikraus/u8glib/wiki/userreference

[5] **Demo sketch**: www.elektormagazine.com/160550

The Elektor Community

LEARN DESIGN SHARE

82	248153	1040	489	235332
Countries	Enthusiastic Members	Experts & Authors	Publications	Monthly Visitors

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.

Elektor Web Store: 24/7 candy store for every electronics engineer! Permanent 10% discount for GREEN and GOLD Members.
www.elektor.com

Elektor Magazine: Six times per year a thick publication packed with electronics projects, news, reviews, tips and tricks.
www.elektormagazine.com

Elektor PCB Service: Order your own PCBs, both one-offs and larger runs.
www.elektorpcbservice.com

Elektor Weekly & Paperless: Your digital weekly news update. Free.
www.elektor.com/newsletter

Elektor Academy: Webinars, Seminars, Presentations, Workshops and DVDs ... Practice-oriented learning.
www.elektor-academy.com

Elektor Books: Arduino, Raspberry Pi, microcontrollers, Linux and more. Available in our online store with a 10% Member discount! **www.elektor.com/books**

Elektor TV: Reviews, timelapse, unboxing and personal journals. Watching is learning.
www.youtube.com/user/ElektorIM

Elektor Labs: Showcasing your own projects and learning from others. We develop and test your ideas!
www.elektor-labs.com

Become a member today!

GREEN
€5.67 per month
£4.08 / US $6.25

- ✘ 6x Elektor Magazine (Print)
- ✔ 6x Elektor Magazine (PDF)
- ✔ Access to Elektor Archive (Thousands of Articles)
- ✔ Access to over 1,000 Gerber files
- ✘ Elektor Annual DVD
- ✔ 10% Discount in Elektor Store
- ✔ Exclusive Offers

www.elektor.com/green

GOLD
€7.58 per month
£5.50 / US $8.42

- ✔ 6x Elektor Magazine (Print)
- ✔ 6x Elektor Magazine (PDF)
- ✔ Access to Elektor Archive (Thousands of Articles)
- ✔ Access to over 1,000 Gerber files
- ✔ Elektor Annual DVD
- ✔ 10% Discount in Elektor Store
- ✔ Exclusive Offers

www.elektor.com/gold

FREE

- ✘ 6x Elektor Magazine (Print)
- ✘ 6x Elektor Magazine (PDF)
- ✘ Access to Elektor Archive
- ✘ Access to over 1,000 Gerber files
- ✘ Elektor Annual DVD
- ✘ 10% Discount in Elektor Store
- ✔ Elektor weekly e-zine
- ✔ Exclusive Offers

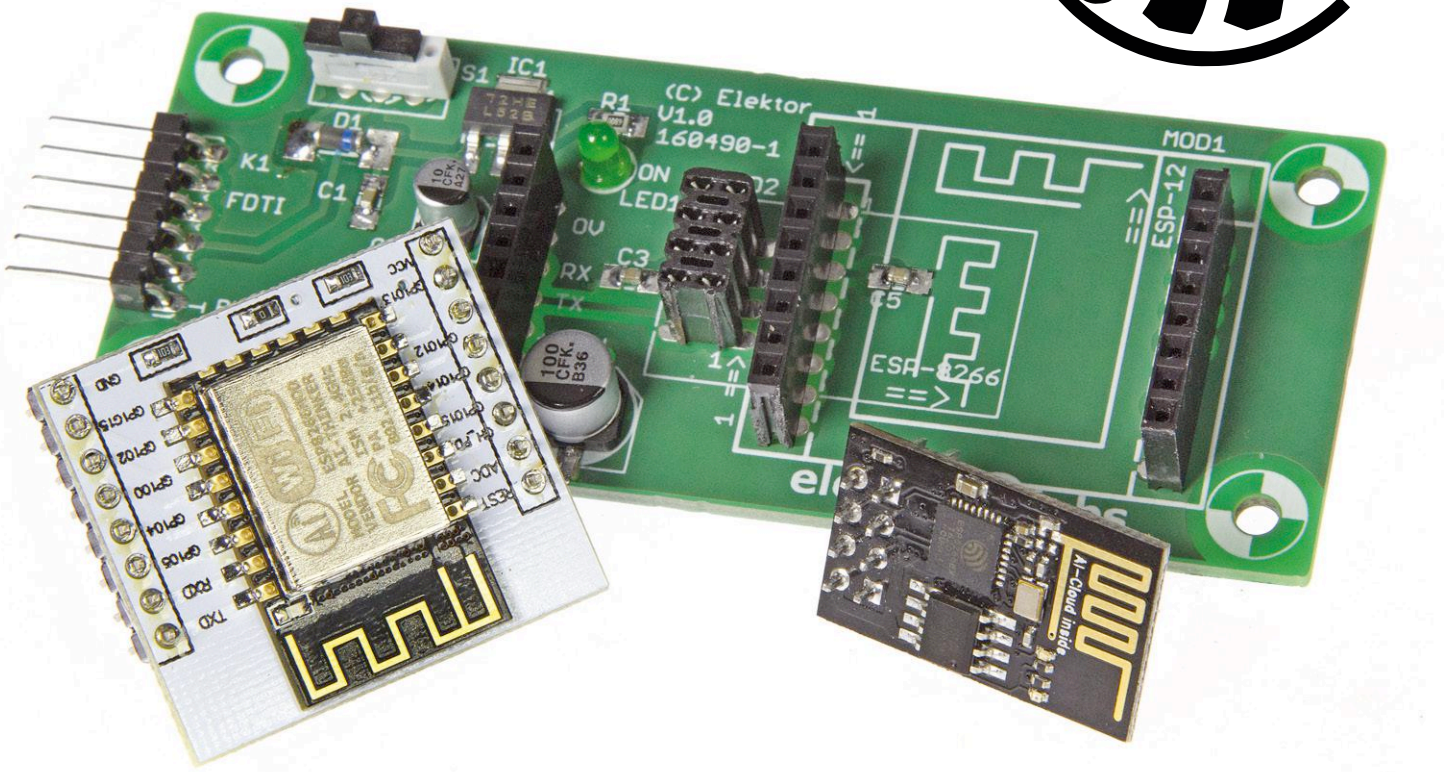
www.elektor.com/newsletter

facebook.com/ElektorIM

twitter.com/Elektor

USB Programming Adaptor for ESP8266

from the Espressif family, I'd like the ESP-01 and ESP-012



By **Pascal Rondane** (France)

This project lets you program ESP-01 and ESP-012 type ESP8266 modules via an FTDI USB interface. An additional connector makes it possible to talk to any other module in the ESP8266 family, using flying leads.

This project is very handy, particularly when developing on breadboard or for quick prototyping, as it lets you dispense with switches or jumpers for enabling the programming mode. A simple cable with an FTDI USB–TTL converter lets you power the circuit and transmit the code.

Electronics

The circuit (**Figure 1**) shows that the incoming 5 V supply on the 6-way connector K1 is dropped to 3.3 V by regulator IC1. This reduced voltage is required for powering the ESP8266 modules. A switch allows the power to be disconnected without having to unplug the FTDI cable; the green LED indicates if the circuit is powered or not. Capacitors C3 and C4 provide effective power supply decoupling of IC1 on the one hand and the ESP modules on the other.

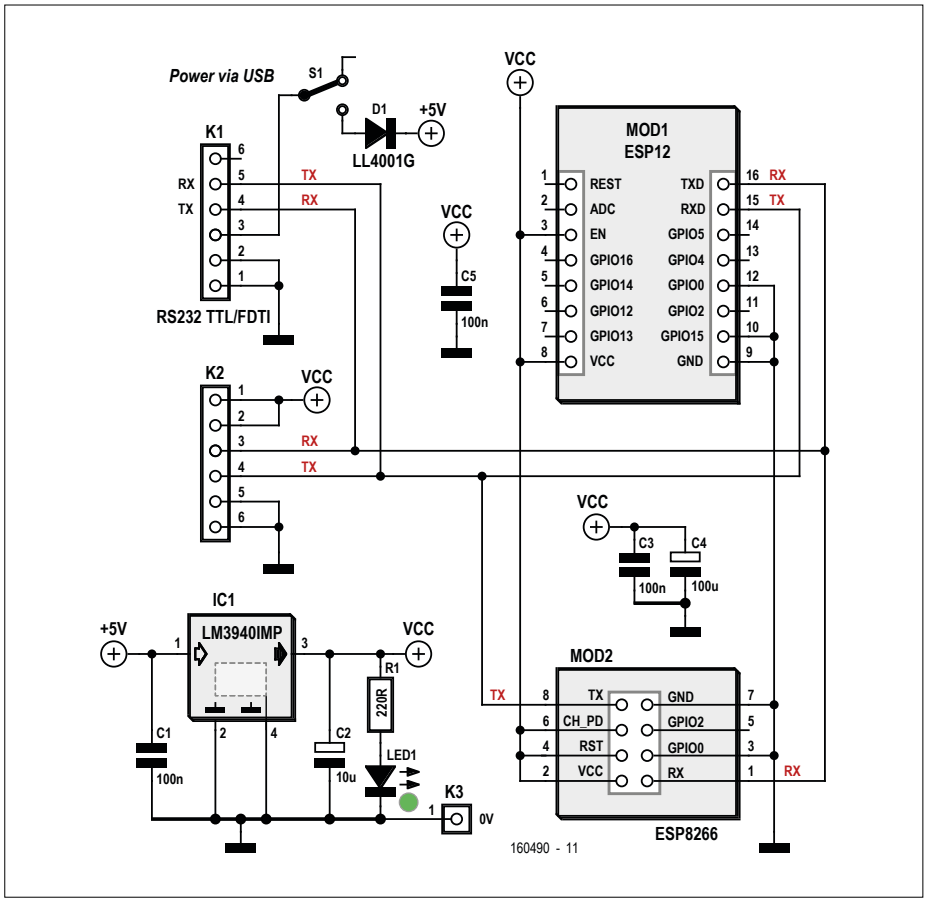


Figure 1. Adaptor circuit.

There's no need to configure the modules to put them into programming mode – everything is taken care of on the PCB. For ESP-01, the GPIO0 pin is connected to 0 V and the RST and CH_PD pins to 3.3 V. For ESP-012, the EN pin is connected to VCC, and the GPIO0 and GPIO15 pins to ground.

Female connector K2 provide access to the supply 0 V and 3.3 V terminals, along with the TX and RX signals. In this way, it is possible to program an external module with non-standard pinout by using a few flying leads.

Wiring presents no particular difficulty; solder the SMA components first and end with the connectors.

This board is versatile, I use it for testing prototypes. It offers the advantage of not being prone to the poor contacts or EMC problems associated with prototyping boards…

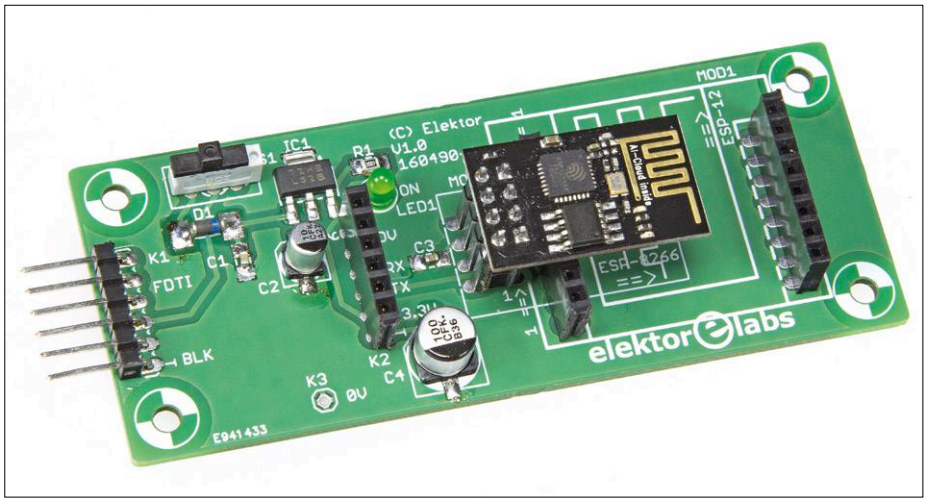


Figure 2. The ESP-01 module is ready for programming.

Software

Start by connecting the FTDI USB-TTL converter cable. Attention: the green wire on the interface goes on the switch side; the black is indicated by the silk-screening on the PCB (BLK). Ideally, you could put a coloured sticker on the visible part of the connector as an indicator. On the computer, install the FTDI driver, then select the COM port for the interface.

For programming the modules, I use the Arduino IDE. You have to first add the ESP8266 boards. Open `Files → Preferences` and at the bottom of the window, copy the following link into the `additional card manager URL` field:

`http://arduino.esp8266.com/stable/package_esp8266com_index.json`

Then open `Tools → Card type → Card manager`. Lastly, browse the list till you find `esp8266` and click `Install`. In the `Tools → Card type` menu you should see `ESP8266 Modules` with different variants. All that remains to do is select the mod-

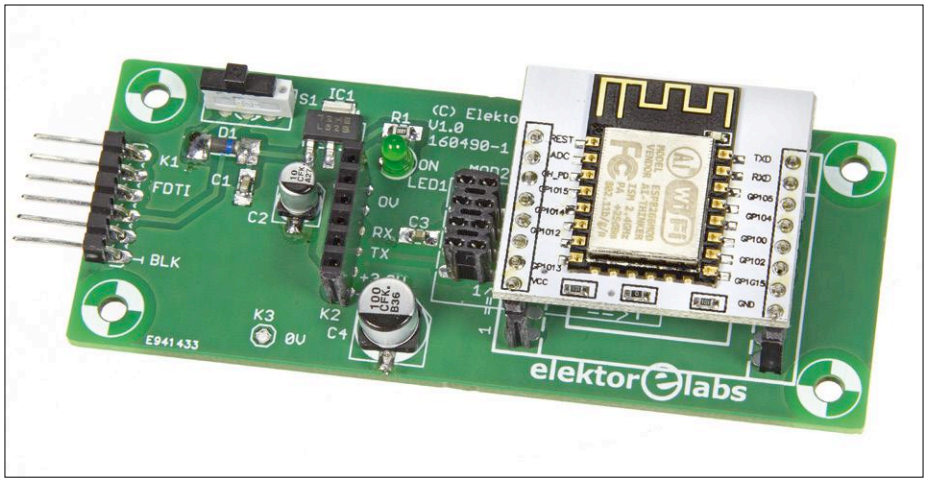
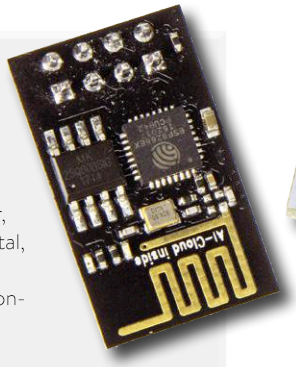


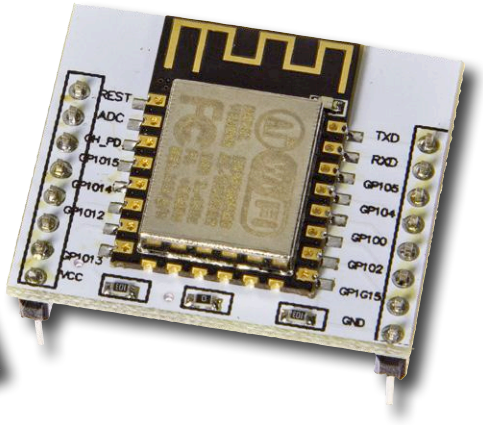
Figure 3. The ESP-012 module sits on the two rows of female connectors.

ule to be programmed: *Generic ESP8266 module* for ESP-01, *NodeMCU09 (ESP-12 Module)* for ESP-012.

Once the module has been correctly plugged into the appropriate socket (see **Figures 2** and **3**), wait for the green LED to come on. All you now have to do is download your program from the Arduino IDE. ⏮

(160490)





COMPONENT LIST

Resistor

R1 = 220Ω, thick film, 5%, 0.1W, 150V, SMD 0805

Capacitors:

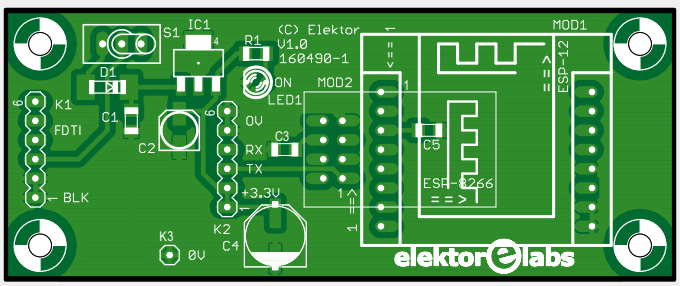
C1,C3,C5 = 100nF, 50V, X7R, SMD 0805
C2 = 10µF 16V, radial, SMD, Panasonic type EEEFK1C100R
C4 = 100µF 16V, radial, SMD, Panasonic type EEEFK1C101P

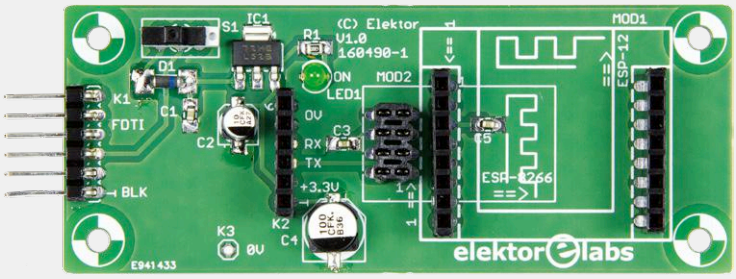
Semiconductors:

D1 = LL4001G diode, 50V, 1A
IC1 = LM3940, 3.3V, low-dropout regulator
LED1 = green, 3mm

Miscellaneous:

K1 = snap-off pinheader, 1×6 contacts, horizontal, 0.1″ pitch
K2 = snap-off female connector, 1×6 contacts, vertical, 0.1″ pitch
S1 = SPDT slide switch, through-hole, 500mA (EOZ 1K2 series)
MOD1 = 2× female connector for ESP-012, snap-off, 1×8 contacts, vertical
MOD2 = 1× female connector for ESP-01, snap-off, 1×8 contacts, vertical
1 FTDI TTL-232R-5V converter cable, 1.8m
PCB # 160490-1





Web Links

[1] FTDI driver: www.ftdichip.com/FTDrivers.htm

[2] Datasheet for FTDI USB-TTL converter cable: www.ftdichip.com/Support/Documents/DataSheets/Cables/DS_TTL-232R_CABLES.pdf

[3] Article page: www.elektormagazine.com/160490

reichelt elektronik

Technology connects.

- More than 45 years of experience
- 24-hour shipping
- More than 85,000 products

THE STYLISH RETRO VIDEO GAME CONSOLE FOR YOUR LIVING ROOM

RASPBERRY PI 3 IN NES DESIGN

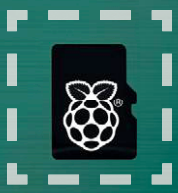
With this bundle you can put a stylish, high-quality retro video game console in your living room. For the implementation we recommend RetroPie and our “How-to” article in the reichelt magazine.

Order no.:
RPI NESPI BUNDLE

NEW £68.43

Our bundle includes:

- a Raspberry Pi 3 B
- a high-quality case in NES design
- a controller in NES design
- a controller in SNES design
- a fan for the case
- a high-performance power supply
- a 16 GB microSD card (Class 10) with pre-installed NOOBS (6 operating systems)
- a 1.2 m HDMI cable



+

RASPBERRY PI 3

Millions sold worldwide…
…and now less expensive than ever before!

- 1.2 GHz ARM Cortex-A53 64-bit quad core processor BCM2837
- 1 GB SD-RAM & VideoCore IV with dual core GPU
- WLAN, Bluetooth, HDMI, USB, LAN, microSD slot, CSI, DSI, 40x I/O, …

instead of 30.37 Order no.: RASPBERRY PI 3

SAVE 13 % **£26.57**

How-to article in the reichelt magazine

Retro gaming with the Raspberry Pi:

How to turn your mini computer into a retro video game console in just a few steps

In the reichelt magazine we explain step by step what you need to put a retro video game console in your living room and how to use it for playing truly classic video games.

LEARN MORE NOW ▶

Power supply for breadboards

- Maximum current: 700 mA
- Input 1: Hollow connector 5.5 / 2.1 mm, DC IN 6…12 V
- Input 2: USB 5 V
- Input 3: Micro USB 5 V
- Output: 3.3 V, 5 V, ground, USB

Order no.:
DEBO BREAD POWER

£3.73

Daily prices! Price as of: 23. 11. 2017

Onlineshop languages:

www.reichelt.co.uk

ORDER HOTLINE: +44 203 808 95 25

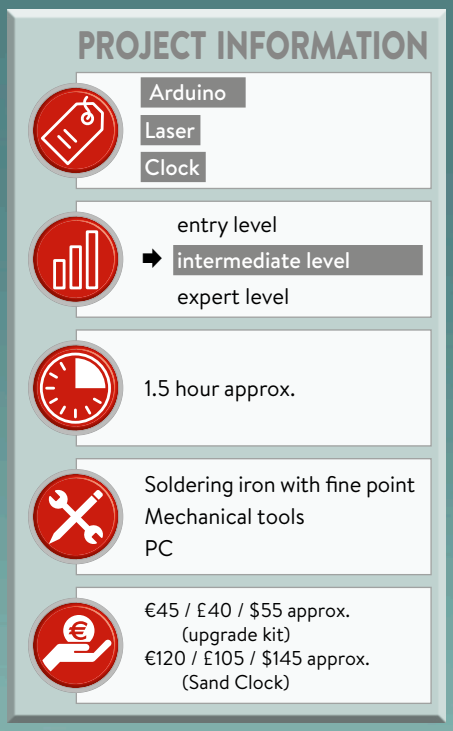
eLABS PROJECT

Laser Time Writer

Writing with light

By **Ilse Joostens and Peter S'heeren** (Belgium)

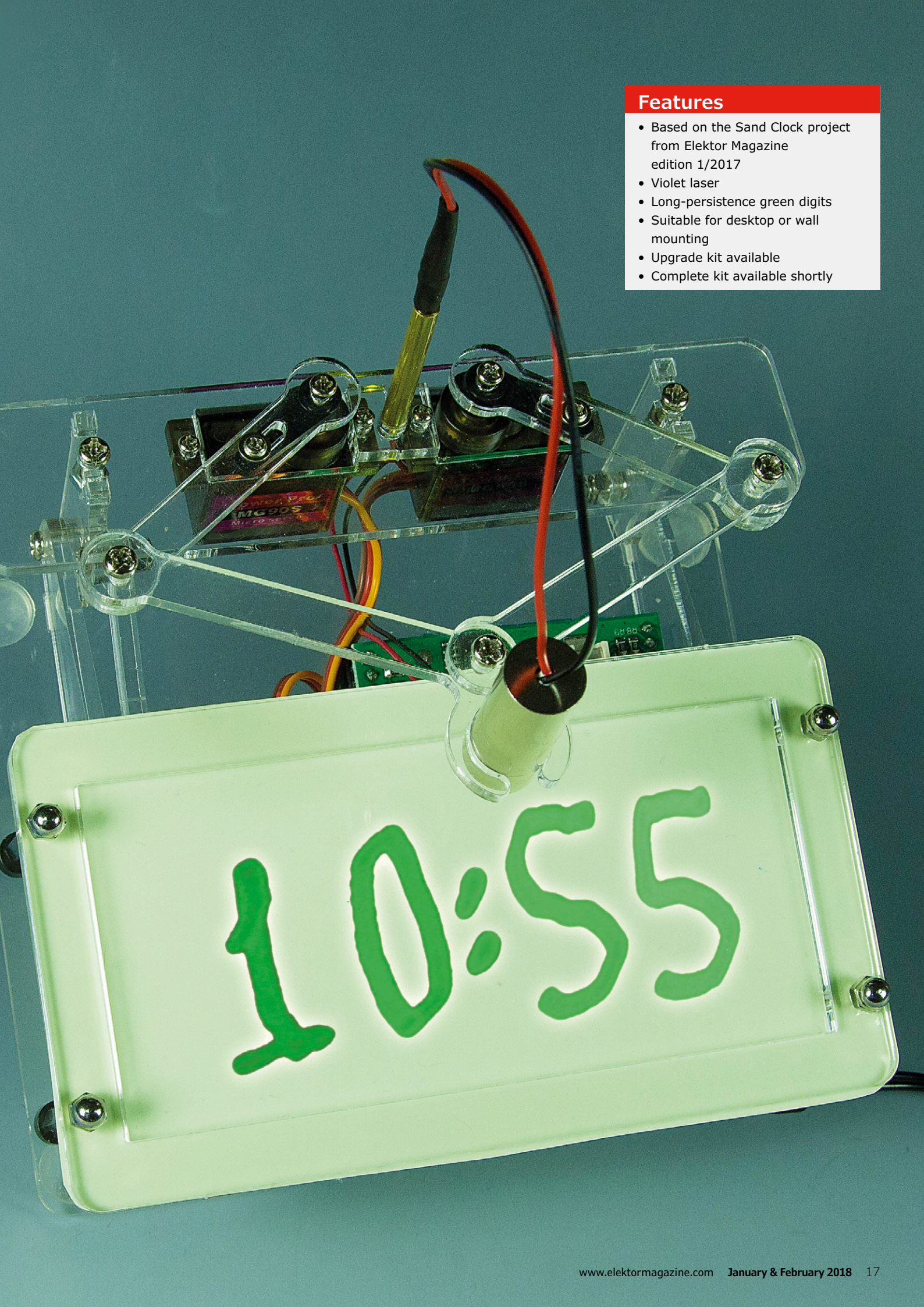
Based on an idea from **Nicholas Stock** (USA)



The popular Sand Clock project published here a year ago (Elektor 1/2017) was based on the plotter clock from FabLab Nuremberg, which wrote the time with a replaceable felt-tip pin. We have not been sitting still during the past year, and now we are proud to present the new Laser Time Writer. It writes the time with a violet laser on a piece of self-adhesive phosphorescent film.

Features

- Based on the Sand Clock project from Elektor Magazine edition 1/2017
- Violet laser
- Long-persistence green digits
- Suitable for desktop or wall mounting
- Upgrade kit available
- Complete kit available shortly



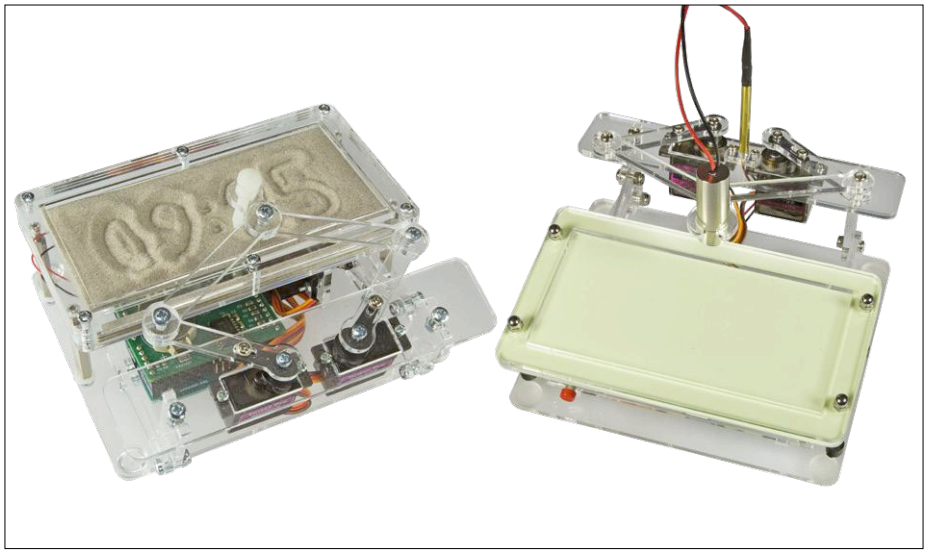


Figure 1. The Sand Clock before the upgrade (left) and after the Laser Time Writer upgrade (right).

Intermezzo 1: What's glowing there in the dark?

The simple explanation is that "glow in the dark" materials absorb energy from light shining on them and then release this energy over a relatively long period in the form of light. This light emitted over an extended time is usually greenish, but there are also materials that emit other colours. As usual, the detailed explanation is a bit more complicated.

Fluorescence and phosphorescence

The general term for the emission of photons (light) by a material in response to energy input into the material is *luminescence*. However, the term *glow* is used for materials that emit visible light when strongly heated.

Photoluminescence is a form of luminescence in which the energy is input in the form of photons, causing electrons to become excited. When these electrons return to their ground state (which is called decay), photons are emitted – usually at a longer wavelength than the incoming photons (this is called the Stokes displacement).

Three different mechanisms can be distinguished for photoluminescence: *fluorescence*, *phosphorescence*, and the *Raman effect* (which we will disregard here).

With fluorescence, which is named after the mineral fluorite (calcium fluoride) in which this phenomenon was first studied, the time between the excitation and decay of the electrons is very short (on the order of nanoseconds), so impinging light with a particular wavelength results in the virtually immediate emission of light with another wavelength. Some examples are dyes and pigments that light up when exposed to ultraviolet light ("black light"), fluorescent pens, fluorescent vests, etc. Fluorescent tubes, low-energy lamps, LED lamps and white LEDs also use fluorescent substances — which are confusingly called phosphors — to convert ultraviolet or blue light into a light spectrum that we perceive as white.

As early as the Middle Ages, materials that emit light were called phosphors — a name derived from the Greek word *phosphoros*, which means "bearer of light". The chemical element phosphorus, discovered by the German alchemist Hennig Brand in 1669, owes its name to the greenish glow emitted by white phosphorus in contact with oxygen in the air. However, this chemical reaction does not have anything to do with phosphorescence. With phosphorescence, quantum-mechanical effects delay the decay of the electrons from the excited state by times on the order of milliseconds to hours, so phosphorescent materials continue to emit light after the initial light source is removed. With glow-in-the-dark materials this effect is especially strong, and they can continue to "glow" (be visible in the dark) for hours after being exposed to light.

Although the Sand Clock [1] is super cool, it has a few drawbacks for everyday application. Due to the use of a sandbox, it can only operate in the horizontal position, which makes correct lighting essential for good legibility. The noise when the sandbox is being shaken smooth is unavoidable, but it can be very irritating — so the Sand Clock is not the ideal choice for a clock on your bedside table or nightstand.

The Laser Time Writer shows the time by writing it with a violet laser on a piece of self-adhesive phosphorescent ("glow in the dark") film. The written digits disappear after a short while, depending on the lighting conditions, allowing the cycle to start over again. The physical background of this phenomenon is described in more detail in the **Intermezzo 1** inset. We could have also used a bright white or blue LED instead of the laser, but the distance between the light source and the phosphorescent film is less critical with the laser — and it looks a lot more high-tech.

If you want to convert an existing Sand Clock into a Laser Time Writer, an upgrade kit is available in the Elektor Store. **Figure 1** gives an impression of the clock before and after the upgrade. A full Laser Time Writer kit will also be available in the near future — we'll keep you informed.

Phosphorescence in practice

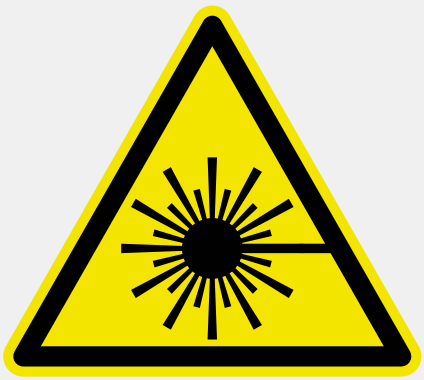
In the past, zinc sulphide doped with copper, or sometimes with silver, was used as a phosphorescent material. Nowadays alkaline earth metal aluminates doped with europium, dysprosium or other rare earth elements are used for this purpose. More recently, doped alkaline-earth silicates have been used for other colours, such as blue. Most of these materials are chemically and biologically inert and therefore non-toxic.

As far as we know, the self-adhesive film used here for the laser clock is only available with green phosphorescence, but if you want to experiment with other colours you can buy a variety of "glow in the dark" powders and use them to make your own phosphorescent paints. Suitable recipes can be found online.

Mechanical construction

The mechanical construction of the Laser Time Writer does not need much description. It is largely based on the Sand Clock, so conversion from the Sand

Intermezzo 2: Laser classification



Class 1

Lasers in this class are safe for the eyes under all conditions, either because the power is very low or because the laser light is screened off and cannot reach the eyes.

Class 1M

Lasers in this class are safe when viewed directly, but they can be dangerous when viewed through a magnifying glass or telescope. Lasers in this class have a diverging beam or a large beam diameter. The radiation from class 1 and 1M lasers may be visible or invisible.

Class 2

This class only applies to visible-light lasers, and lasers in this class are safe with incidental direct viewing. Intentionally staring into the beam can lead to eye damage.

Class 2M

This class only applies to visible-light lasers, and lasers in this class are safe with incidental direct viewing as long as the blink reflex time is not exceeded, the same as class 2. They can be dangerous under all conditions, including incidental viewing, when optical systems such as described under class 1M are used. Lasers in this class produce highly divergent or broad beams.

Class 3R

Lasers in this class are regarded as not dangerous (low risk), but they can potentially be dangerous if the laser beam is viewed directly. The limit of exposure for class 3R is five times the applicable limit for class 1 (for invisible light) or class 2 (for visible light). Lasers that continuously emit visible light (CW lasers) at power levels from 1 to 5 mW are normally regarded as class 3R.

Class 3B

Laser light radiation in this class is generally regarded as dangerous, and direct exposure can immediately cause serious eye damage. With a continuous (CW) laser the power may not exceed 500 mW. This radiation is dangerous for eyes, and to a lesser degree dangerous for skin. Viewing diffuse reflections of laser radiation in this class is not dangerous.

Class 4

This is the highest class. The light radiation is very dangerous for eyes and skin. Viewing diffuse reflections is also dangerous. Class-4 laser beams can cause fires if they strike flammable materials.

With an output power of 5 to 10 mW, the laser module used in this project is technically a class 3B laser product, but there is no danger in this application.
This output power level puts the laser module at the lower end of class 3B. The potential risks are additionally reduced by the relatively large beam diameter (several millimetres) and the writing motion. If a limited reflection of the laser beam from the self-adhesive film does occur, it can only reach the viewer's eye for a fraction of a second due to the motion of the laser module and is therefore not dangerous.

Never look directly into the laser beam, and never point the laser module toward a reflecting surface.

Clock to the Laser Time Writer is fairly easy and requires a minimum of new or additional parts. You can download extensively illustrated instructions in PDF format free of charge from the project page created for this article [2].
Along with replacing the sandbox and stylus by a piece of self-adhesive phosphorescent film and a laser module, we modified the baseplate of the Sand Clock. It now has four mounting holes for attaching the clock to the wall. The Arduino board can also be mounted in two different orientations, allowing the connectors for power and USB to face forwards or backwards as desired. The new baseplate is not included in the upgrade kit.
We made the arms of the pantograph mechanism a bit longer, so that the outer corners of the writing surface are also accessible now. This also makes it possi-

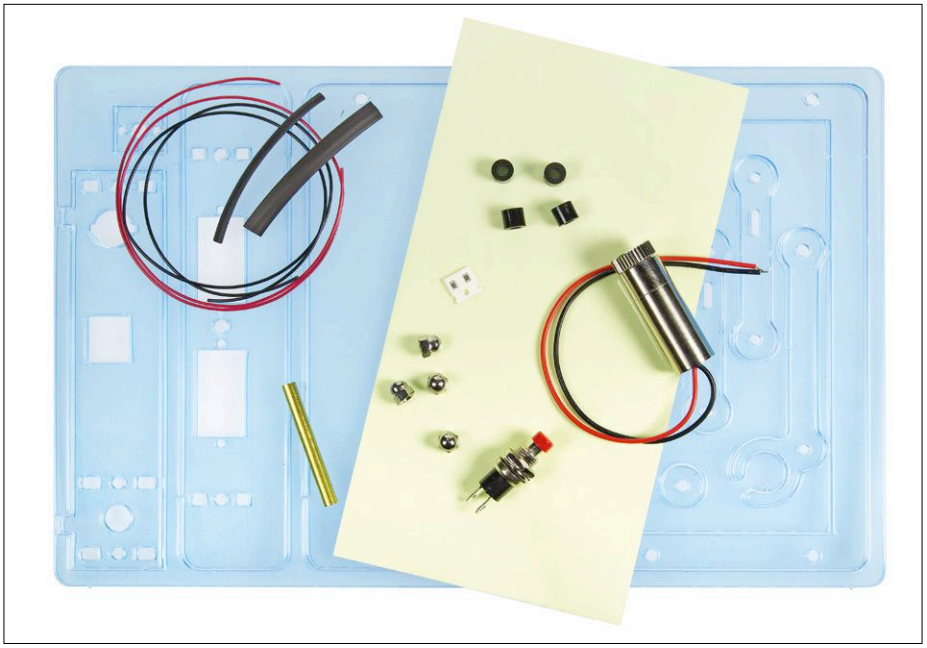


Figure 2. The contents of the upgrade kit.

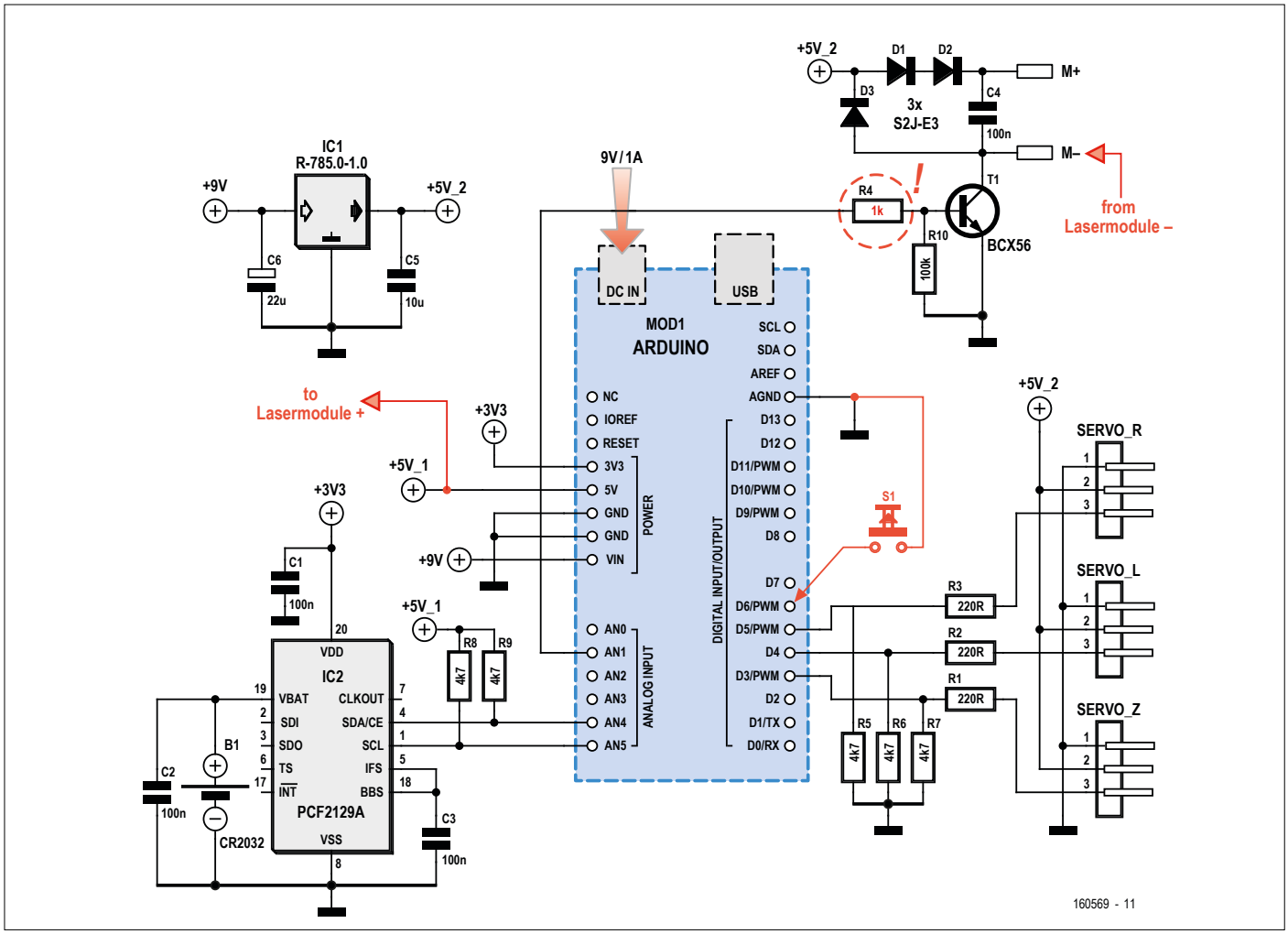


Figure 3. The schematic diagram of the Sand Clock electronics, showing what has to be changed to turn it into a Laser Time Writer.

Intermezzo 3: How do lasers work?

The term "laser" is an acronym of "light amplification by stimulated emission of radiation". Let's see what that means. The underlying principle of a laser is that excited electrons in a higher energy state than normal emit photons (light particles) when they return to the ground state. This is called *spontaneous emission*, but it is not enough by itself to make a laser.

If a spontaneously emitted photon hits another atom or molecule with an electron in the same excited state, then that electron will also return to the ground state and emit a photon with exactly the same phase and wavelength as the first photon. This is called *stimulated emission*.

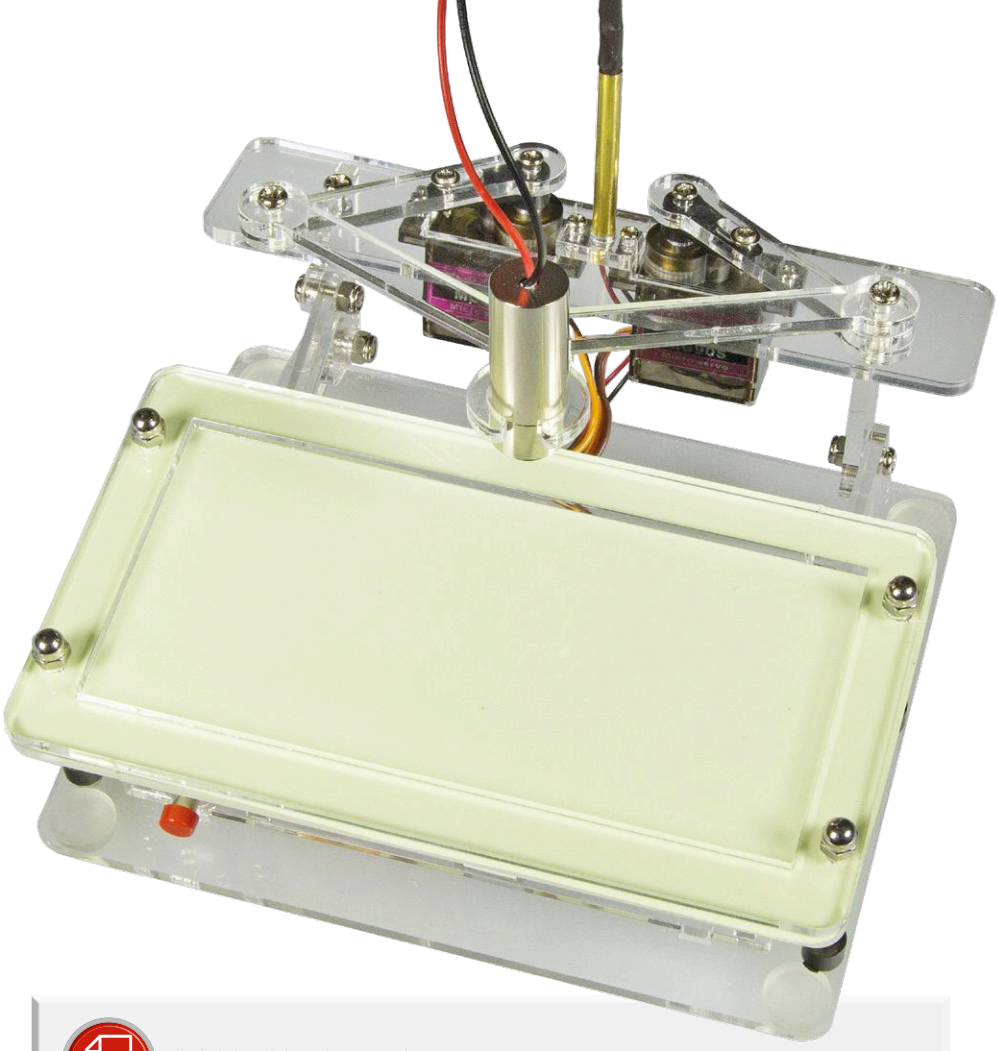
The material in which the laser radiation is produced is placed between two parallel mirrors so that the emitted photons pass back and forth through the material, producing more and more photons with the same phase. One of these mirrors is made partially reflecting, so the laser light can emerge at that end once the beam is strong enough.

In addition to this stimulated emission, some of the generated photons are absorbed by atoms that are in a lower energy state. In order to achieve laser operation, there must be many more excited electrons in the material than non-excited electrons. This condition is called *population inversion*, and it must be created and maintained in the material by adding energy – usually in the form of light from flash lamps, light from another laser, or with an electrical current in the case of semiconductor lasers.

Laser light differs from ordinary light in three important ways: it is monochromatic (single colour), it is coherent (all light waves or photons have the same phase), and it is highly collimated (a much tighter beam than can be achieved with ordinary light using lenses or mirrors).

The first working laser was built in 1960 by a researcher whose name is largely unknown to the general public: Theodore Maiman. The laser had been theoretically predicted earlier by the physicists C.H. Townes, A.L. Schawlow, N.G. Basov and A.M. Prokhorov. They later received the Nobel Prize for this, while Theodore Maiman was left empty-handed – an especially unfair situation.

ble to write the time upside down.
Finally, we added a pushbutton so that you can manually start a writing cycle. This option was also included in the latest version of the Sand Clock software.
An acrylic support that allows the clock to be positioned upright on a chest or desk is available as an option.
Figure 2 gives an impression of contents of the upgrade kit.



The electronics

The electronics for the Laser Time Writer are nearly the same as for the original Sand Clock. Transistor T1, which controlled the shaker motors in the previous design, now switches the laser module on and off. The laser module (described in more detail below) is powered from the 5 V supply of the Arduino board. We opted for this because the laser module draws only 60 mA at 5 V, and to avoid potential interference with the servo motors.
When T1 is driven into conduction by the Arduino, it acts as a current source. In the original Sand Clock that helped to keep the voltage over the shaker motors within reasonable limits. Unfortunately, the original circuit did not work properly with the laser module, so a modification was necessary. The laser module has a built-in switch-mode driver, so it draws more current at a lower supply voltage. That led to a lock-up situation at switch-on, with the laser module briefly drawing more current than transistor T1 could supply. As a result, the supply voltage to the laser module dropped and it started drawing even more current, up to the point where the voltage dropped very low and the current through T1 was much too high. We solved that problem by reducing the value of the base resistor R4 from 1.8 kΩ to 1 kΩ. With the upgrade kit, you have to solder a 2.2-kΩ resistor in parallel with the original base resistor. That makes the net base resistance approximately 990 Ω. Pushbutton S1 is connected between pin 6 of the Arduino board and ground.
The latest version of the PCB can be used with either version of the clock and allows the base resistance of T1 to be selected with a solder jumper. It also has solder pads to make it easier to connect the leads of the laser module and the pushbutton.
Figure 3 shows the full schematic diagram, including the above-mentioned modifications.

COMPONENT LIST

Resistors

Default: SMD 0805
R1-R3,R11 = 220Ω
R4 = 1.8kΩ *
R5-R9 = 4.7kΩ
R10 = 100kΩ
R12 = 2.2kΩ *

Capacitors

C1-C4 = 100nF, SMD 0805, MLCC
C5 = 10µF, 10V, SMD 1206, MLCC
C6 = 22µF, 16V

Semiconductors

D1-D3 = S2J-E3
T1 = BCX56
IC1 = DC/DC converter, SIP3, 5V/1A (Würth Elektronik 173 010 578)
IC2 = PCF2129A

Miscellaneous

K1 = set of pinheaders (1x10, 2x8, 1x6), 0.1" pitch
SERVO_Z, SERVO_L, SERVO_R = right-angle pinheader, 1x3, 0.1" pitch
B1 = battery holder, CR2032, with CR2032 button cell
1 pc Arduino Uno R3 or comparable
1 pc pushbutton, SPST, NO, 6mm mounting hole
1 set wires (red/black) and heat-shrink tubing
1 pc laser module, 405nm, 5-10mW, 12x35mm (eBay)

Mechanical

6 pcs machine screw, M2x10, Pozidriv/Phillips
6 pcs machine screw, M2.5x8, zinc-plated steel, Pozidriv DIN 7985A
2 pcs machine screw, M3x8, zinc-plated steel, Pozidriv DIN 7985A
15 pcs machine screw, M3x10, zinc-plated steel, Pozidriv DIN 7985A
6 pcs hex nut, M2, zinc-plated steel, DIN 934
7 pcs hex nut, M3, steel, DIN 934
3 pcs locking nut, M3, zinc-plated steel, DIN 985
4 pcs acorn nut, M3, chrome plated, DIN 1587
4 pcs spacer, 3mm height, polyamide, for M3
4 pcs hex spacer, 25mm, M/F M3, nickel-plated brass (min. total height 33mm, TME TFM-M3X25/DR213 or equivalent)
4 pcs spacer, 5mm height, plastic, for M3, outer diameter 6mm
1 pc brass tube, diameter 4mm x 0.5mm, length 35mm
4 pcs self-adhesive rubber foot
3 pcs micro servo, Tower Pro Tower Pro MG90S or MG90. metal gears
1 pc self-adhesive phosphorescent film, approx. 80x150mm (eBay)
Extruded (XT) acrylic sheet, 3mm, clear, laser-cut

* See text

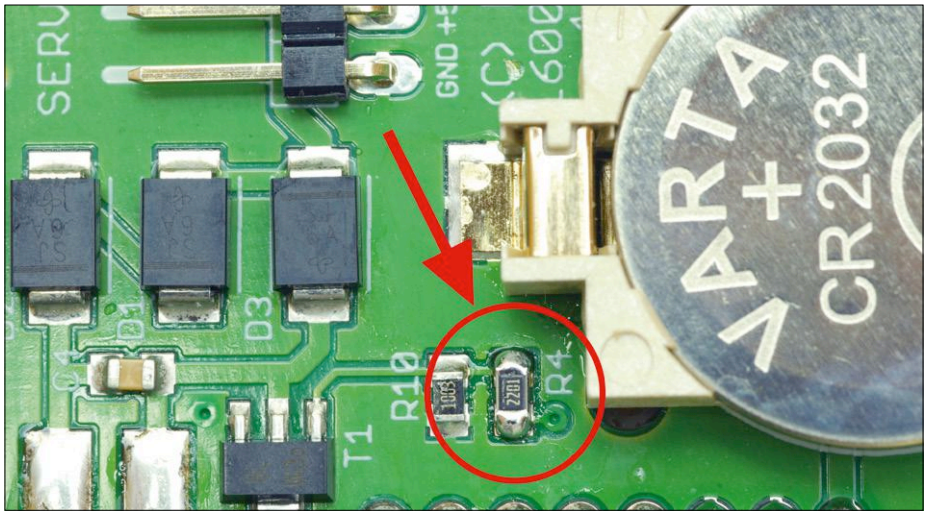
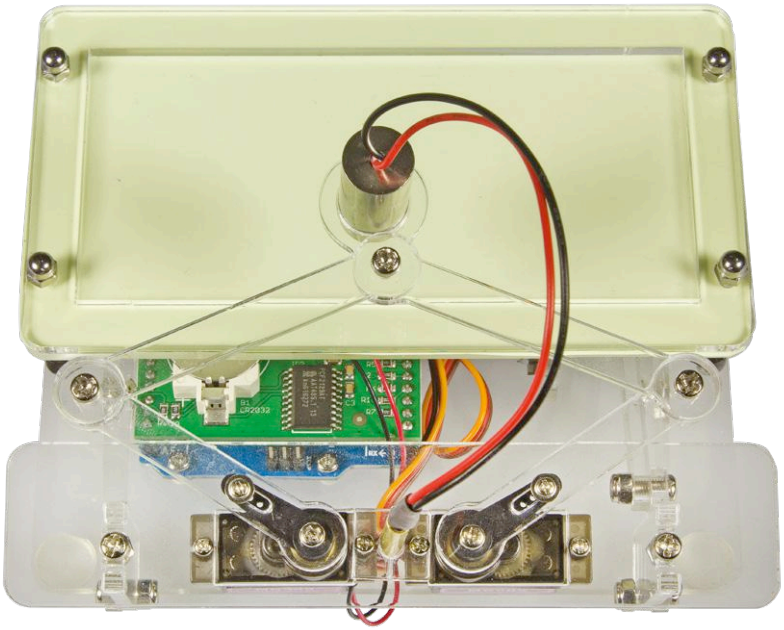


Figure 4. The PCB of the Sand Clock shield for the Arduino, together with a photo showing where the 2.2-kΩ resistor has to be soldered onto the existing 1.8-kΩ resistor.



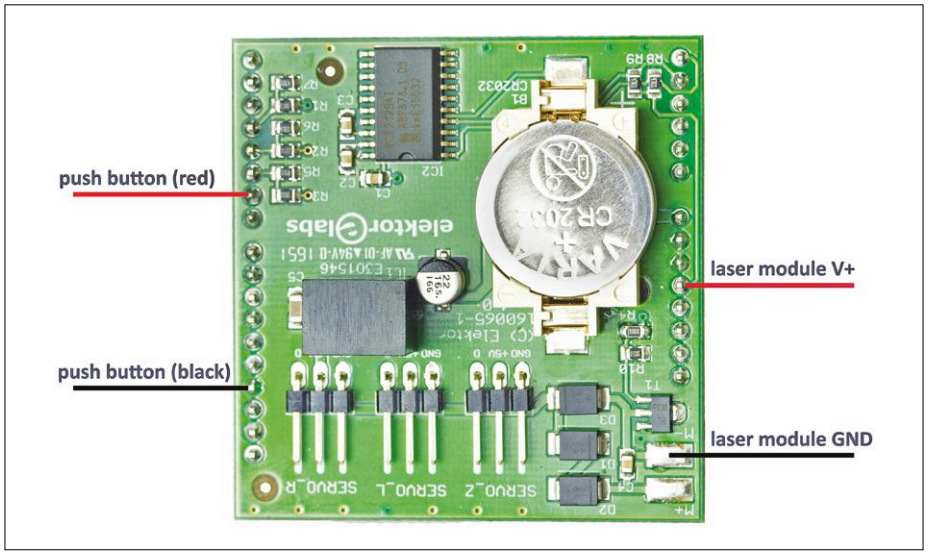


Figure 5. Here you can clearly see how to connect the laser module to the shield board.

Laser module

The laser module included in the upgrade kit is built around a violet laser diode with a wavelength of 405 nm (see also **Intermezzo 2** and **Intermezzo 3**). The work of the Japanese engineer Shuji Nakamura at Nichia Corporation in 1993 laid the basis for the development of blue LEDs. Together with Isamu Akasaki and Hiroshi Amano, he received the Nobel Prize for this in 2014. In 1997 he was instrumental in the development of blue laser diodes. Practical fabrication became feasible in 1999 thanks to ultra-pure gallium nitride crystals developed by Dr Sylvester Poroski at the Polish Academy of Sciences. That opened the way to low-cost mass production for devices such as Blu-ray players and recorders.

Naturally, the technology has been further refined and now both violet laser diodes (405 nm) and blue laser diodes (450 nm) are available, with power levels up to several watts The prices of these laser diodes have dropped dramatically, mainly due to commercial use in low-cost laser projectors.

Software

The Arduino sketch for this project is based on the Sand Clock software, but of course it is not identical. Along with the modifications for controlling the laser, we added the option of writing upside down. The lift servo motor still has only two configurable positions. All that results in a slightly modified command set. The initial calibration procedure is similar to the procedure for the Sand Clock.

The motion of the writing unit has been slowed down between the hours and the minutes, as otherwise the mass of the laser module causes the pantograph mechanism to vibrate too much.

The new sketch for the Laser Time Writer (LTW) is available as a free download on the project page for this article [2]. Software installation and calibration are described in detail in the assembly guide.

Assembly

Building the LTW is similar to building the Sand Clock. Detailed and extensively illustrated step-by-step assembly instructions are available as a download [2]. They apply to both the upgrade kit and the full LTW, for which a specific complete kit will be available soon. Before assembling the writing surface, you should moisten the acrylic with water containing a little bit of detergent. Then place the self-adhesive phosphorescent film on the surface and rub it smooth with a piece of stiff cardboard to remove the trapped air. You can use a paper kitchen towel to soak up the water that is pressed out to the edges. Leave the leads from the laser module to the Arduino shield long enough that the pantograph mechanism can easily be moved up and down by the lift servo motor.

For the sake of clarity, the PCB layout of the Arduino shield for the Sand Clock and the LTW is shown in **Figure 4**, along with a photo of the required modification: a 2.2-kΩ resistor is soldered on top of the existing resistor R4 (1.8 kΩ) as previously described.

Figure 5 shows how the pushbutton S1 and the laser module should be connected to the shield, and **Figure 6** shows several stages in the assembly process, so that you have an idea of what to expect. We hope you have a lot of fun with the upgrade!

(160569-I)

Web Links

[1] www.elektormagazine.com/160065

[2] www.elektormagazine.com/160569

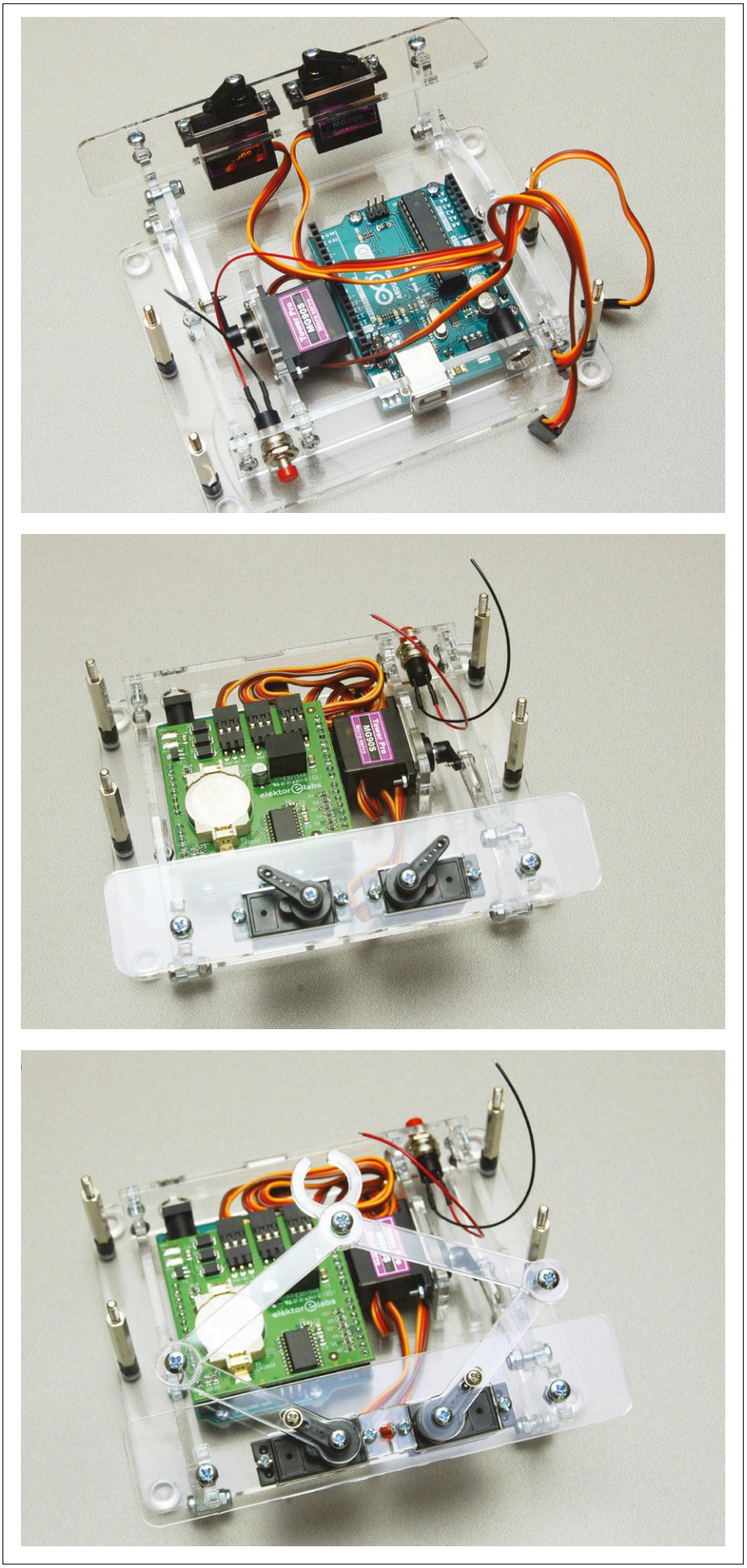


Figure 6. Three stages in the conversion from the Sand Clock to the Laser Time Writer.

The EEBUS is on its Way

A voice of sanity amid the babble of IoT

By **Tam Hanna** (Slovakia)

Smart homes are often built from a ragtag collection of devices from different manufacturers, and taming the mishmash of protocols spoken by these devices to get them to talk to one another successfully is a Sisyphean task. The EEBUS Initiative is a standardization body with its roots in Germany that aims to address this problem.

Let us start our discussion with a brief look at the market. Besides Samsung, LG and Haier there are not really any companies that can offer a complete IoT solution to their customers, allowing them to connect their tumble drier, their refrigerator and everything in between to their smartphone.

Smaller companies operating in niche areas are thus falling behind their more lavishly funded competition. This means that the market is unbalanced and that end users have fewer options open to them when assembling their collections of IoT devices. This is the problem addressed by EEBUS Initiative e.V., a standardization body based in Cologne (see the **text box** 'Who is behind EEBUS?').

Architecture

The principle behind EEBUS is similar to that behind other standards such as Bluetooth LE. A communications architecture must be in place that is as flexible as possible so that it can encompass as wide a range as possible of present and future applications. Also, as a standard, its job is to bring all the various device manufacturers to the same table.

One possible approach is a KV (key-value) store. On the one hand this is technically very straightforward, which means the bar to entry for developers is low. On the other hand, by specifying keys or formats for values, it is possible to create a reasonably rigid standard.

When talking about EEBUS we are usually thinking at the level of the SPINE protocol (short for 'Smart Premises Interoperable Neutral-message Exchange). The standard is highly application-oriented: the developer considers the requirements for inter-device communication in the application at hand and then decides on how the corresponding messages should look. **Figure 1** illustrates the overall architecture. At the top level of the hierarchy we have the physical device: this might be a washing machine, a smartphone or a light.

In the particular case of kitchen appliances or 'white goods' the standard addresses the situation where a single machine might encompass several distinct functions. The SPINE protocol includes the idea of an 'entity', which describes one function of an appliance. Take the example of a wine cooler that has a built-in ice dispenser: the cooling function and the ice dispensing function would correspond to distinct entities.

At the bottom level of the hierarchy the standard provides for individual 'features', each of which describes a single aspect of one function of an appliance.

The question of data formats

Bluetooth LE differs from SPINE in that it specifies both the physical and logical aspects of how communications are carried out. SPINE, however, confines itself to the seventh layer of the OSI seven-layer model: the exact process by which the data make their way from host to host is by and large left for the developer to worry about. EEBUS does, however, suggest using SHIP (which stands for 'Smart Home IP'), a specially-adapted protocol based on TCP/IP.

EEBUS is rather more strict when it comes to specifying how messages should be encoded. Messages take the form of plain text using an XML-based syntax (see the text box 'XML'). This simplifies the implementer's job, as pretty much any programming language will come with one or more libraries for parsing XML. As the processing power available in embedded devices

Who is behind EEBUS?

The history of the EEBUS Initiative can be traced back to the 'E-Energy' Initiative started by the German federal government. The stated goal of that project was to prepare Germany's energy infrastructure for the consequences of the transition to cleaner energy sources and the arrival of the Internet of Things.
The group itself was formed in 2012 at the Hanover Messe trade show; at the time of writing it boasts 65 members. Of particular interest is the diversity of the group: the list of members runs from energy suppliers and automotive trade bodies to individual domestic appliance manufacturers such as Miele.
The working procedures of EEBUS committee are inspired by the Deming PDCA (plan, do, check, adjust) cycle, driven by the identification of 'problems'. Embryonic specification documents are in the first step used to generate application use cases; these then guide the production of more technical specifications. Experience gained from practical implementations of these specifications then influences the next revision of the standard.

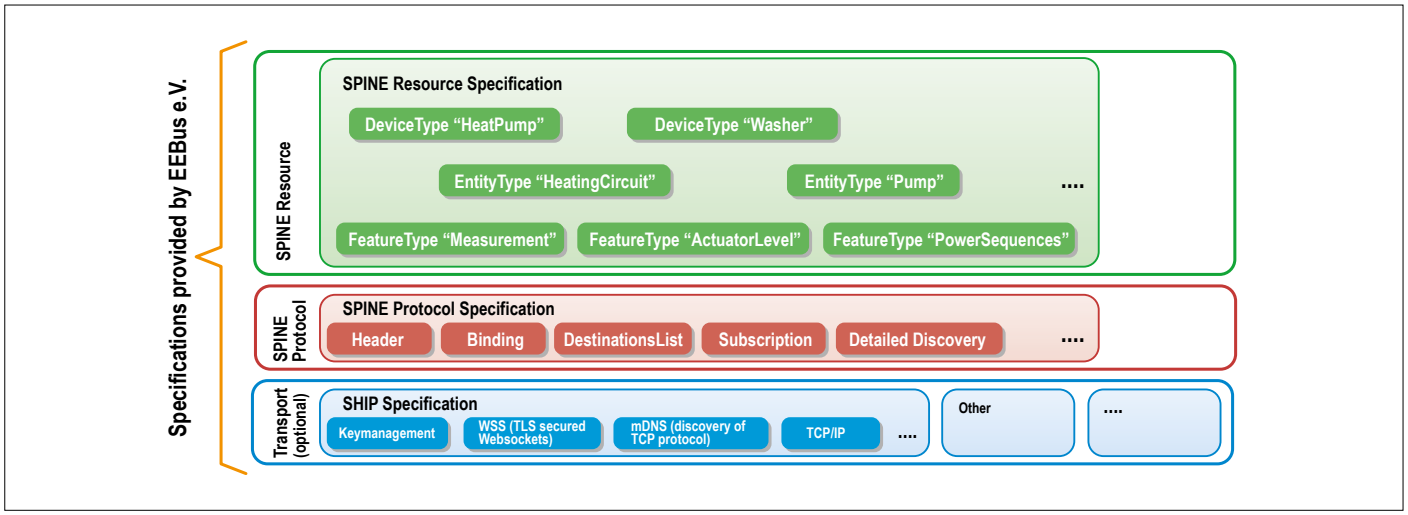


Figure 1. EEBUS architecture with three-level addressing.

becomes ever greater over time, the admittedly enormous overhead of using an XML-based format will become a decreasingly significant factor.

An example

We will look at a simple example to show the structure of a SPINE message. Suppose that a device wishes to transmit a reading from a sensor in response to a request from another device.
All EEBUS messages comprise two parts. The first part we will look at is the header, as illustrated in **Figure 2**, which contains getting on for a dozen different fields. The fields `addressSource` and `addressDestination` together specify the route the message takes from the transmitter to the receiver. Usually an address is specified using the hierarchy of device, entity and feature; in many particular cases it is possible to dispense with the need to specify an address, which reduces the size of the header.
The type of message is specified in the `CmdClassifier` field. Currently the standard provides for six different types of message, whose roles are listed in **Table 1**.
The other fields in the header are pretty standard stuff: they provide information about things such as the version of the specification in use and whether an explicit acknowledgement of receipt of the message is required. As a small example here is the header for a fictitious datagram that the EEBUS standardization organization makes available on their website:

XML

SPINE transfers information in XML format. You can think of 'eXtensible Markup Language' as a kind of HTML for resources: instead of items scattered around a web page the file describes elements of the real world. Although the format is certainly not economical in terms of bandwidth use, and is not managed well by eight-bit processors, it does have the advantage that it can be handled in more or less the same way on all platforms.
An XSD (XML Schema Definition) file is used to specify which tags can be used in an XML file. The XSD file itself uses XML to describe the structure of the XML messages. It specifies which fields can be used and how the information contained within them must be structured, and extends to including provision for ordinary data types such as 'double'.
Initially you may like to start by opening the XML and XSD files in an ordinary text editor: XML files are in general fairly self-explanatory. There is a wide range of XML editors available for more advanced users, but it is beyond the scope of this article to review them.
Programming you own XML parser is probably not such a worthwhile endeavor: experience shows that is a job best left so the software experts. Search on-line for 'XML parser' plus the name of your favorite programming language and you will almost certainly find plenty of candidates.

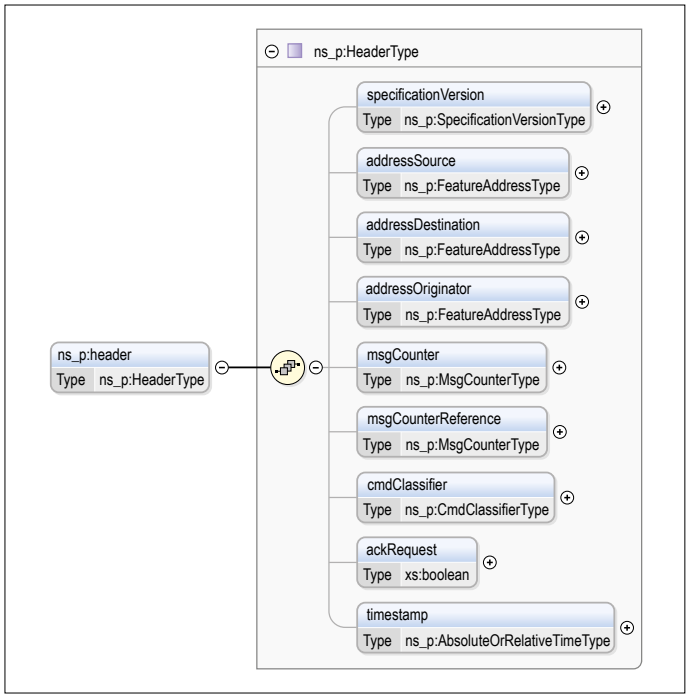


Figure 2. A SPINE header up close and personal.

```
<header>
    <specificationVersion>1.0.0</specificationVersion>
    <addressSource>
        <device>d:_i: 46925_TestTempSensor_1</device>
        <entity>2</entity>
        <entity>1</entity>
        <entity>1</entity>
        <feature>1</feature>
    </addressSource>
    <addressDestination>
        <device>d:_i:46925_TestSHM_1</device>
        <entity>1< /entity>
        <feature>1</feature>
    </addressDestination>
    <msgCounter>15</msgCounter>
    <msgCounterReference>8</msgCounterReference>
    <cmdClassifier>reply</cmdClassifier>
</header>
```

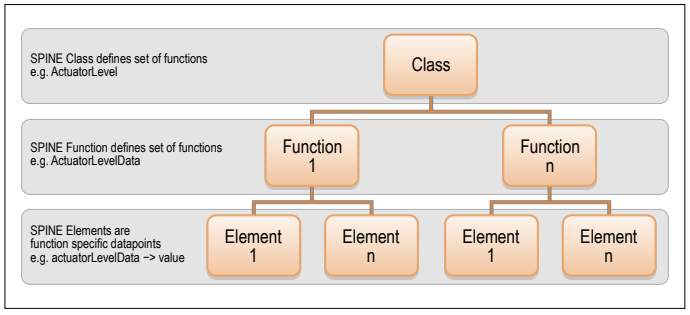


Figure 3. Functions and elements are grouped in classes.

An important aspect of the header is the pair of `MessageCounter` parameters. The value of `msgCounter` is incremented every time a packet is sent, and so functions as a message ID which uniquely identifies (within a short time period at least) each message transmitted. The `MsgCounterRef` field allows a message to refer to the message that caused it to be sent, which means that a reply can be matched up with its corresponding command. Communications protocols often specify their own formats for addresses. In the case of SPINE the elements of an address are separated by colons. The string 'd:' introduces the address (a kind of 'header within a header'), and the subsequent 'I' allows the addition of further manufacturer-specific information. The idea is that the addressing scheme should generate a globally unique string that can be used to identify a particular device.

Functions and filters

The second part of the datagram is the message body itself. This normally takes the form of a `payload` element. There is a potential trap in this declaration: even though the protocol allows for multiple `cmd` tags in a single message, implementations usually only pay attention to the first one.

```
<payload>
    <cmd>
        <function>measurementListData</function>
        <filter>
            <cmdControl>
                <partial/>
            </cmdControl>
        </filter>
```

Table: CmdClassifier.

`CmdClassifier`
specifies the role of the message being transmitted.

Name	Role
`read`	Command that causes the receiver to report a data value relating to its operation. For example, 'how many ice cubes do we have?'.
`write`	Command that instructs the receiver to accept a data value
`call`	Command that causes the receiver to execute a function or initiate an action. For example, 'eject five star-shaped ice cubes'.
`reply`	Message providing a reply to a read command. For example, 'I have six star-shaped ice cubes'.
`notify`	Message sent from the owner of a piece of information to an interested client. For example, 'attention, I have just been asked to supply four ice cubes'.
`result`	Message that acknowledges the successful or unsuccessful receipt or execution of a command. For example, 'sorry, I don't know what to do with that command'.

The first thing we see is the declaration of the task to be carried out; and the presence of a `function` tag also requires a subsequent 'filter' that delimits the region of the file associated with the command in question.
As we shall discuss below in more detail, EEBUS provides a 'Resource Specification' document that provides a reasonably comprehensive specification of most of the functions. SPINE gives developers a helping hand here with a kind of class structure that describes how the elements are grouped together (see **Figure 3**). The class itself is not normally given in the message; it is rather just a way of describing and grouping the functions and elements that belong to it. Depending on the function there will then be one or more groups of elements which in turn describe the data payload proper.

Data elements

The second part of the message contains the elements which carry the actual data payload. In our example this is a reading from a sensor.

```
      <measurementListData>
      <measurementData>
         <measurementId>1</measurementId>
         <valueType>value</valueType>
         <timestamp>2015-1105T10:14:00.0Z</timestamp>
         <value>
            <number>22</number>
         </value>
         <valueSource>measuredValue</valueSource>
      </measurementData>
      </measurementListData>
   </cmd>
</payload>
```

Requests

A request asking a device to read a sensor might look like the following.

```
<cmd>
    <function>measurementListData</function>
    <filter>

        <cmdControl>
            <partial/>
        </cmdControl>
        <measurementListDataSelectors>
            <measurementId>5</measurementId>
            <valueType>minValue</valueType>
        </measurementListDataSelectors>

    </filter>

    <measurementListData/>

</cmd>
```

The request causes the information itself to be placed in the filter. There are no data elements.

All about standards...

The structures described above provide a solid technical foundation, but do not address the basic problem we mentioned in the introduction: each manufacturer could end up implementing his own combination of entities and features, and we would have made no progress.
The EEBUS Initiative has followed the example set by the Bluetooth SIG. The organization obliges developers of parsers and other data processing tools to make their products ready for later versions of the protocol from the start. For a specific example look at **Figure 4**, which illustrates how the standard can be successfully extended by adding new child nodes at certain places in the data model. If a parser should encounter an element it does not recognize it can simply skip it and continue parsing at the next element it does recognize.
The idea behind this is simple: when all the manufacturers of a certain type of device implement the specified parts of the standard, a basic kind of interoperation becomes possible. The processes we describe here then allow the detection of manufacturer-specific extensions which (where they exist) enable additional functions.

... and detection

Last but not least we turn to the question of how a device can analyze and connect to a new communication partner. In the protocol this process is called 'functional commissioning'; and in the same section of the standard we also find a description of the 'notification' process.
The specification requires that the zeroth feature of the zeroth entity (that is, feature address 0 and entity address 0) provides an implementation of 'NodeManagement'. Any desired remote device can be completely analyzed by sending a `Read` command directed at the function `nodeManagementDetailedDiscoveryData`. The target device will react to this command by sending a relatively complex class structure which gives an overall picture of the available functions and entities.
The standards organization gives no further guidance regarding how to identify a device with which you are communicating; the text of the standard only makes rather tantalizing reference to 'a rather intelligent implementation' (page 57).

Get to know the documentation

The organizations behind EEBUS offer the specifications for download in exchange for your e-mail address; it is not necessary to join the standardization group.
The file ' EEBUS_SPINE_TR_Introduction.pdf' gives a high-level view of the EEBUS standard: anyone who seriously wants to get to grips with EEBUS should print this document out.
EEBUS_SPINE_TS_ProtocolSpecification.pdf and EEBUS_SPINE_TS_ResourceSpecification.pdf between them supply further technical information. If you find a concept in the first of these documents and want to investigate it further, you will need to turn to the second.
And finally there is a set of XSD ('XSL Schema Definition') files stored in a zip archive. These files describe the structure of the individual messages, and can be turned into graphical representation using various visualization tools. Some XML-based IDEs allow these files to be used to verify automatically-gen-

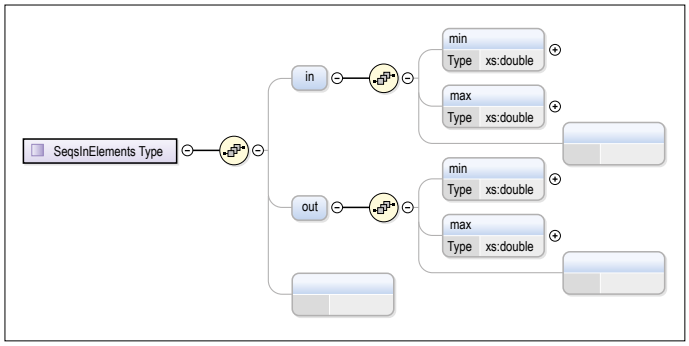


Figure 4. The parser must be ready to cope with extensions to the protocol at the specified points.

erated code: it is considerably easier to track down bugs if you know that your data file conforms to the standard. Be aware, however, that the XSD files are in many cases more tolerant than real-world implementations.

Conclusion

SPINE/EEBUS is important from a technical point of view because of its immense flexibility. There is hardly any appliance that cannot be described using the predefined key-value storage model. Thanks to similarities to Bluetooth LE the concepts behind the system should be more or less familiar to developers.

However, it must be made clear that SPINE provides just the basic scaffolding to support manufacturer-independent communication between devices; it does not provide any specific implementation guidelines for particular device types. To realize full interoperability, the manufacturers of each class of device must jointly decide which SPINE messages are used and in what form. These efforts have now begun, with the first standards emerging for washing machines and dishwashers.

As is the case with GPIB, whether a given EEBUS device will actually work together with another in practice remains to be seen. ⏮

(150583)

Internet Links

[1] Specification download: www.eebus.org/en/download-standard/

[2] List of members: www.eebus.org/en/initiative/members/

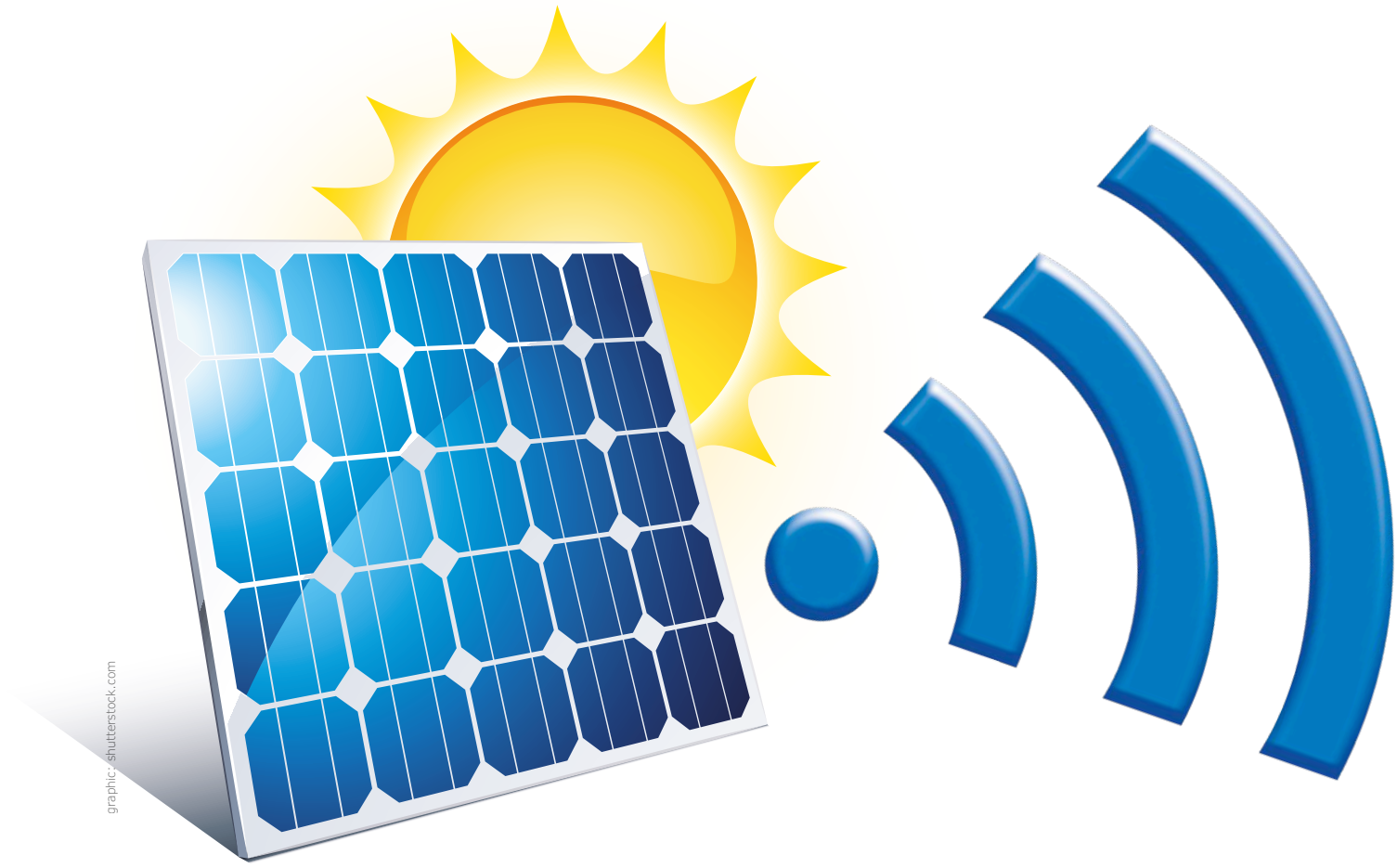
[3] Data model and details: www.eebus.org/en/technology/data-model/

Must have a connection

Observe that the process described here requires the existence of a transport connection between the two devices in question: it is not a substitute for DHCP, broadcasts and other, lower-level, detection processes.

Solar Power for Wi-Fi Repeater

Choice of analogue or digital versions



graphic: shutterstock.com

Some years ago I developed a solar-powered surveillance camera with data transmission by Wi-Fi or GSM network with SIM card. However, some users wanted a Wi-Fi model with greater range. That got me thinking about a Wi-Fi repeater powered by a solar panel.

By **Laurent Labbe** (China)

Outline

For the repeater function, I use a very small router from the company TP Link (such as the TL-WR802L) which offers two advantages: low power consumption (around 200 mA ; approx. 140 mA in repeater mode) and the absence of an on/off button. This miniscule box functions primarily as an Ethernet/Wi-Fi router, but it can be configured as a repeater. I have tested several versions of this model and it works perfectly — with the exception of those purchased in China (in which the ROM is exclusively in Chinese, and not flashable to English. My solar power project has two working modes: the first mode is purely analogue, using two comparators; the second mode is enriched by a dash of digital, thanks to a tiny 8-pin microcontroller from Microchip, the 12F1572.

The battery used is an assembly of two 18650 (Li-Ion) cylindrical cells, connected in parallel (so they must be identical!). During my (Chinese) search for high capacity cells, I did not have many surprises. There are a lot of 'fake' cells with good-sounding capacities. Finally, I found

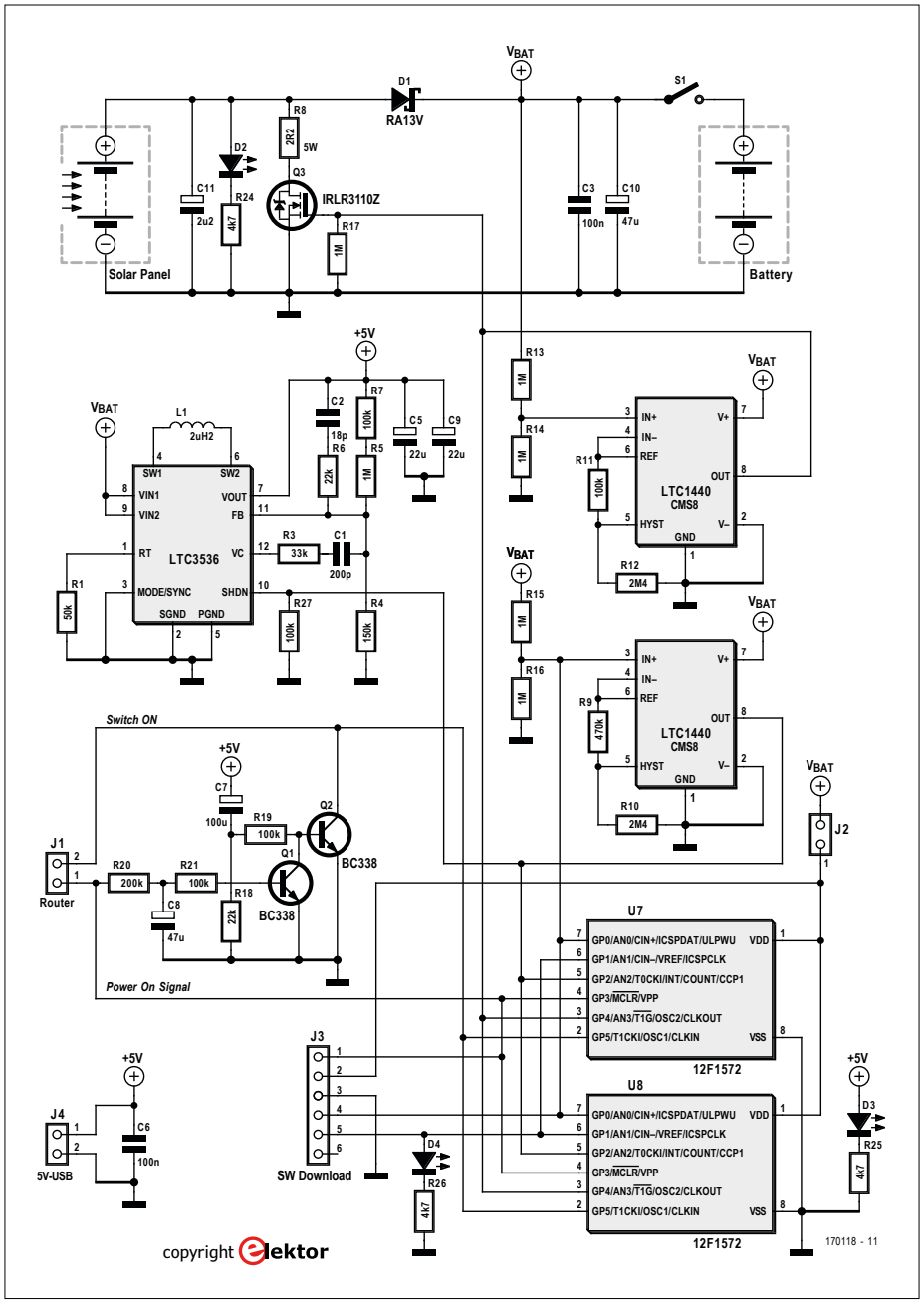


Figure. 1. Circuit diagram of the solar power supply for Wi-Fi repeater.

(and measured) some 3000-mAh cells. A trick to find the fake ones: weigh them. Choose the heavier ones. Now, before buying, I ask for the weight of the cells. To mount the cells in parallel, I modified a battery holder, initially intended to connect them in series.

To charge the battery, I adopted a simple, but not very orthodox solution. The solar panel charges the battery directly via Schottky diode D1. This diode is a type RA13V from Sanken Electric rated at 2 A. You could substitute another type based on the maximum current of your solar panel.

Finally, the router is powered by a DC-DC converter. I searched far and wide and I tested a lot of integrated circuits, and my final choice was the LTC3536. It can deliver 1 A, and above all it has a true 'power OFF' mode. Indeed, many DC-DC converters when shut down do cut the PWM, but a current still flows in the inductor. Consequently there is never truly 0 V at the output. LED D3 lights when a voltage of 5 V is present at the output, on the USB connector that connects to the repeater.

The solar panel has the following specifications:

- output voltage = 5 V;
- output current = 1 A;
- power output = 5 W.

LED D2 lights when the output of the panel reaches the voltage of 5 V and charges the battery.

Obviously the arrangement only works correctly when the solar panel is receiving full sunlight. The TP Link consumes around 200 mA and thus the autonomy of a fully charged battery is about 30 hours. Let's now look at how it works.

Pure analogue mode with the comparators

The two comparators are type LTC1440 (**Figure 1**), an improved, but pin-to-pin compatible version of the MAX931. Previously I always used the MAX931, but I had some concerns (some of them toasted…) so I changed to the LTC which is not only cheaper but seems less sensitive.

The first LTC1440 compares the battery voltage to a reference of 4.1 V. When the voltage exceeds that value (the battery thus being fully charged) the comparator switches on Q3 which shunts the solar panel with power resistor R8. Arguably the rating of R8 depends on the maximum current supply from the panel (P = R8 x I_{max}^2; with R8 = 4.6/I_{max} ohms. This arrangement avoids the addition of a transistor (and hence a voltage drop and a control circuit). Because of the hysteresis, the charging resumes at 3.9 V.

The second comparator monitors the battery voltage. It starts the DC-DC converter (via the shutdown pin) when the voltage is above 3.8 V and shuts it down when the voltage falls below 2.9 V.

These thresholds are determined by resistors R13-R14-R15-R16 (see **Table 1**). The hysteresis levels are determined by the values of resistors R9 (470 kΩ) and R11 (100 kΩ).

Why use two separate comparators and not a double one? Because the reference voltage must be connected to the

Table 1. Resistors R13-R4-R15-R16 for each working mode.

Variant	R13	R14	R15	R16
Analogue Mode	2.4 MΩ	1 MΩ	1.8 MΩ	1 MΩ
Digital mode	not used	not used	1 MΩ 1%	1 MΩ 1%

same input (IN−) whereas with a double internal comparator, the reference goes to IN− and IN+.

Digital mode with a microcontroller

In this mode, the software does the job of the two comparators: start/stop of the charge and start/stop of the DC-DC converter. LED D4 still indicates the state of the charge by the number of flashes.
I have chosen the 12F1572 for several reasons: low consumption, but primarily an internal reference voltage which allows the calibration of the analogue to digital converter. Usually, the power supply voltage is used, but I didn't want to add a voltage regulator for the microcontroller. The battery voltage is brought to a divider R15-R16 (1 MΩ each, preferably 1% types). A note here: the voltage to be measured must never exceed the reference voltage, here fixed at 2.048 V! With this divide-by-two circuit, the maximum measureable battery voltage will be 4.096 V, so this will be the maximum charge voltage. In the software, it is possible to define the ratio as 1/3 instead of ½, but we then have to recompile and also redefine R15-R16, by replacing R15 with two resistors of 1 MΩ for example.
The program is written in C and compiled with the free version of MikroC which limits the size of code. The printed circuit board allows either the DIP or SOIC version of the 12F1572 to be used.

Router option and Wi-Fi access point

On the circuit diagram, there is a whole group of components around transistors Q1 and Q2. This gathering is necessary for a particular working mode: MiFi, where the device is simultaneously a router and a Wi-Fi access point. In this case, it is necessary to press a button to activate the router. On the two types of MiFi routers that I tested (Huawei and Alcatel), starting up the router is commenced by taking a signal to ground. By taking the router apart and soldering a wire to the start/stop button, the router can be started by a pulse to ground. A second wire is needed to detect the voltage of the router's power (3 V or 5 V once the router has started). This allows, via the delay circuit C8/R20 and Q1, to avoid taking the line to ground which simulates pressing the Start button. There are two reasons for this. Some routers don't like the button pressed for too long (defined here by the components R18/C7) and so the signal goes high again on the startup of the router. The second reason is more important. The MiFi router has a built-in battery. It is thus possible that the router is already working although the solar charger is not charging. If the router is already working when the solar charger powers the 5-V line, the activation of the Start button might be taken as a stop by the router. With Q1 blocked, there will not be any pulse on the Start button line and the router will continue working normally.
In pure analogue mode, the startup of the DC-DC converter switches on Q2 and thus causes the grounding of the start signal for a time defined by C7/R18. After the router starts, the signal is released after a period defined by R20/C8.
In digital mode, the software takes care of everything. The program should be complied as either the repeater variant (no need for start signal), or the router variant (with startup signal).

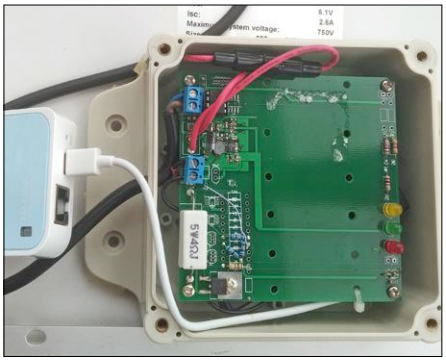


Figure 2. Board and cabling.

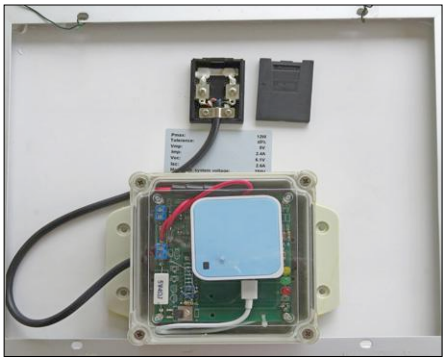


Figure. 3. Circuit assembly at the back of the solar panel.

The software

After the usual initialisation routines, the code sets the watchdog timer at 2 s. That's another advantage of this microcontroller; it allows dynamic adjustment of the watchdog timer.
The main loop starts by reading the battery voltage, As I have not succeeded in getting the MikroC function `adc_read` to work, I wrote the routine (`adc_read_ll`). The FVR (`Fix Voltage Regulator`) is set to ON only during the reading, to economise on the current.

As a function of the battery voltage (being read every 2 s by the watchdog) the LED will flash to indicate the state of charge:

- 1 flash: battery flat. Waiting for recharge. Repeater stopped.
- 2 flashes: battery under the threshold for starting the DC-DC converter.
- 3 flashes: battery above the threshold for starting the DC-DC converter. 5 V is present on the USB socket.
- 4 flashes: battery fully charged. Current from the solar panel diverted to the power resistor.

In the case where the router mode is used, with simulation of the start stop button being pressed, there is a regular check that the router is running properly. If it is not, a new start pulse will be sent. This might happen if the router stops automatically (which does happen with the Huawei).
I programmed the PIC with a PICkit 3.5 and a small printed circuit board for connection. I then placed this on the final circuit. The PIC can also be programmed while in-circuit on the board. J2 and J3 serve for the programming of the microcontroller on the card. The connection between the two pins of J2 must be removed in order to programme. In normal mode, the connection must be permanent. J3 is a PICkit 3.5 extension. The software is available at [1].

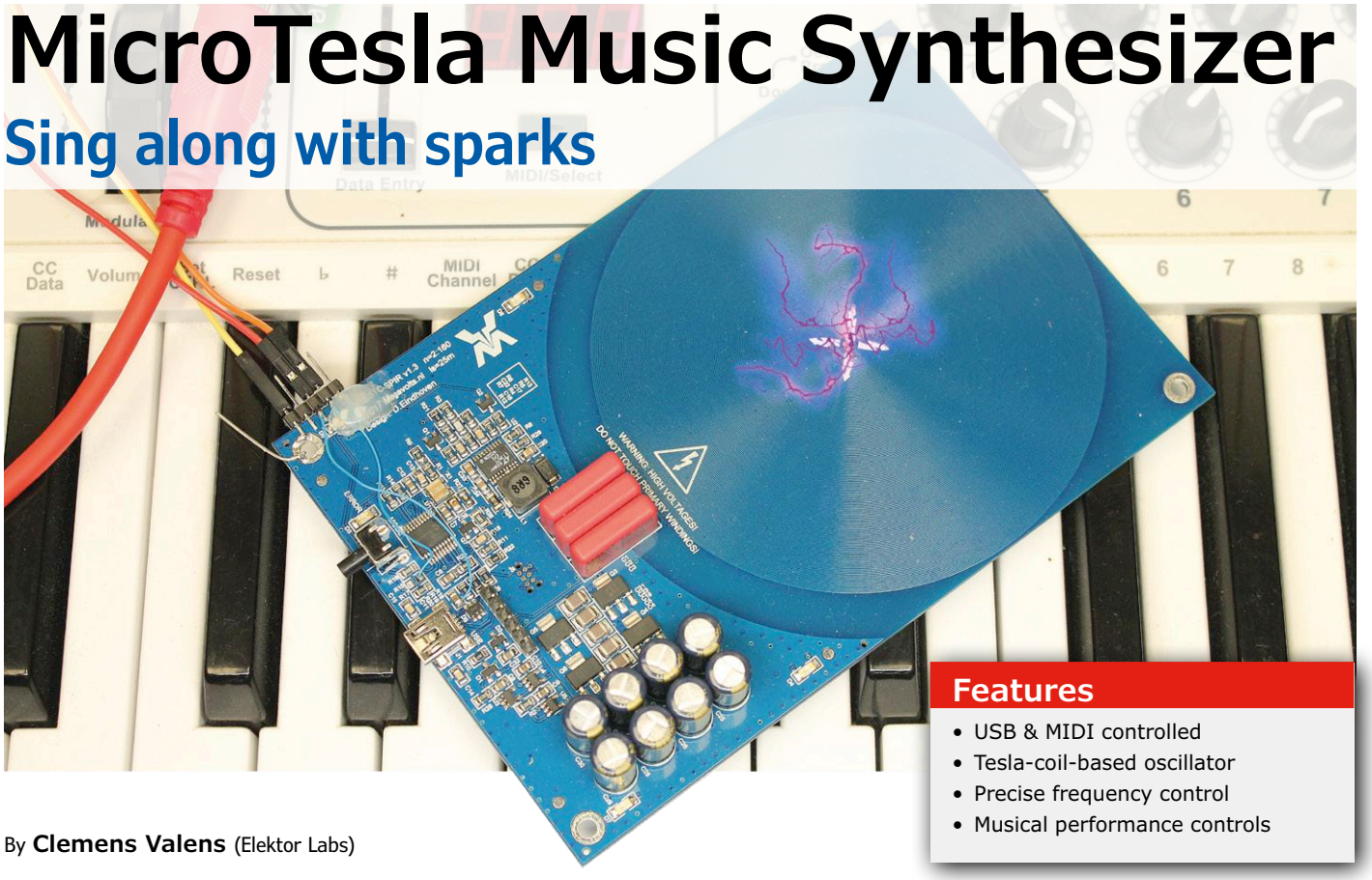
(160570)

Web Link

[1] Article page: www.elektormagazine.com/160570

MicroTesla Music Synthesizer

Sing along with sparks



Features

- USB & MIDI controlled
- Tesla-coil-based oscillator
- Precise frequency control
- Musical performance controls

By **Clemens Valens** (Elektor Labs)

In a previous article [1] we staged the Spiral MicroTesla, a USB-powered, flat mini Tesla coil capable of producing crackling sparks of up to about 4 cm. Although a great design, its usefulness is rather limited with its fixed 5-, 10- and 20-Hz spark rates. In this article we will show you how to transform the device into a MIDI-controlled music synthesizer.

Making music with Tesla coils has become popular after the American technological performance group ArcAttack participated in the TV show America's Got Talent back in 2010. The Tesla coils used by ArcAttack (**Figure 1**) are several metres high, but the concept of sparkly music translates perfectly well to our flat-coiled board and its 3-cm "needle".

In this article you will learn how to transform the Elektor MicroTesla into a Tesla music synthesizer with MIDI input. Sound manipulation will be limited to controlling the instrument's pitch since we have no way of digitizing the sound and bring it back into the microcontroller to do more processing on it. Therefore, do not expect fancy filtering or amplitude controls, because there are none. What you will get are a few frequency parameters that can be controlled in real time over MIDI:

- *vibrato* depth and speed (frequency modulation);
- pitch bending (frequency shifting);
- *portamento* (sliding from one note to another).

MIDI data can be sent to the Tesla Synth over USB or, after adding a suitable MIDI adapter (more on that later), over a traditional MIDI bus.

Before we can do this, however, we have to understand how the MicroTesla works, and especially its software.

Controlled by a real-time operating system

The original Spiral MicroTesla program is based on a small real-time operating system named Smalltask. It's a kind of cooperative multitasking OS where a scheduler runs through the list of tasks whenever it can and launches one by one those that are scheduled for execution at a certain point in time. Each task runs as long as it wants, and when it is done it returns control to the scheduler so that the next scheduled task can be launched. If a task never terminates or if it does not terminate quickly enough, it will block the rest of the program, exactly like a traditional, single-task program.

The Mini Tesla has three tasks running continuously:

1. `Led_task`: takes care of blinking LEDs in a pleasing manner;
2. `Check_switch` task: detects and

measures button presses that select the different operating modes;

3. `Control_task`: a state machine that controls the power-up sequence and the 5, 10 & 20 Hz and musical scale spark modes.

The task list also contains two tasks that only run on demand: `melody_task` and `error_blink`. The first one runs when the musical scale must be played, the second is only executed when the red LED must be turned off after leaving dark- or high-power mode.

The control task is started when the circuit is powered on. When it 'feels' that everything is fine, it will launch the LED and pushbutton tasks and then enter a kind of idle mode doing nothing. Pushbutton presses will make it cycle through its active modes and then back to idle.

Making sparks is really simple

The next step is to figure out how the software makes the MicroTesla spark. This turns out to be really simple: all that's needed really is a short pulse on the PWREN pin (IC5, port RC4, pin 6). In 5-Hz mode the pulse duration is 1,500 µs, in 10-Hz mode it is 750 µs and in 20-Hz mode 330 µs. High-power mode doubles the pulse durations and also clears the MODE pin (IC5, port RC2, pin 14). For the musical notes the pulse duration is set to a duty cycle of 0.7%, meaning that for the note C2 with a frequency of 65.41 Hz the pulse duration would be 107 µs. The program is a bit conservative and uses 100 µs instead. Again, high-power mode doubles the pulse duration and pulls the MODE pin low.

Surprisingly, that is all there is to it. No complicated control functions or strict timings to respect, in fact a 555 timer and some switches could easily replace the microcontroller.

The variable-frequency pulse generator uses two timers

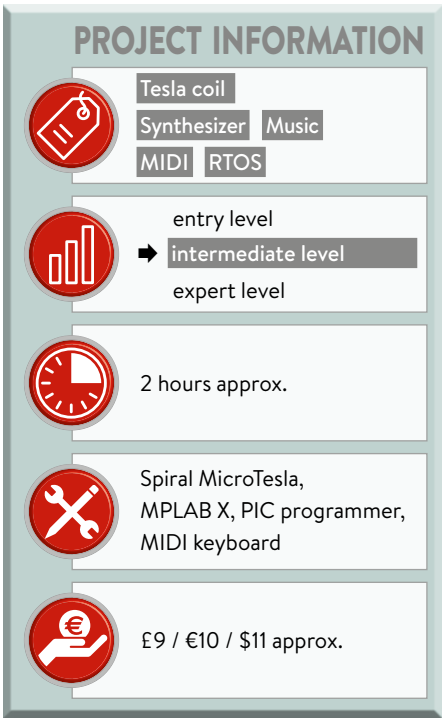
The objective is to create a MIDI-controlled variable-frequency pulse generator with adjustable pulse duration. Now that we understand the way the MicroTesla software functions, we can add these new features to it. To be honest, the original Smalltask operating system is not actually needed for this project, but I left it in to retain the MicroTesla's functions. That way, if you don't have MIDI data handy, you can still amaze your friends with crackling sparks and arcs.

For the oscillator I used two timers: Timer1 for the frequency, and Timer2 for the pulse duration. Every time Timer1 fires, i.e. when a period has been completed, Timer2 is started to time the pulse duration. Smalltask uses a timer too, Timer0, that fires every 125 µs (8 kHz), but this is not fine-grained enough for musical note generation. At 110 Hz (A2) the error would already be more than 1%. In this project Timer1 is therefore clocked from a 1.5 MHz source, giving it a resolution of 667 ns. This keeps the pitch error below 0.5% over the range from 0 to 8 kHz (even though we don't go that high). Timer2 that controls the pulse duration has the same resolution, setting the upper frequency limit to more than 10 kHz if we want to respect a duty cycle of 0.7%.

Interrupt-driven MIDI data reception is preferable

MIDI data is asynchronous and travels in groups; sometimes there is a lot, most of the time there is nothing. The data speed is 31,250 bits per second, and a byte takes 320 µs to complete. MIDI data reception could therefore be handled by polling the microcontroller's UART in a Smalltask thread. However, since the UART can produce an interrupt every time a byte is received, interrupt-driven data reception is preferable as it consumes much less processing time.

I did not bypass the Smalltask OS altogether as I added a worker task at a 1 kHz rate to synchronize the MIDI commands to the pulse oscillator. This task receives a message when the reception of a MIDI packet has been completed (up to three bytes, depending on the status byte). The MIDI command is then parsed and executed. Running status — when MIDI data is sent without a status byte — is supported, active sense — when keep-alive bytes are periodically transmitted — is ignored. Since the Tesla Synth is monophonic, only one note can be played at a time. Pressing a key while a note is playing will result in playing the new



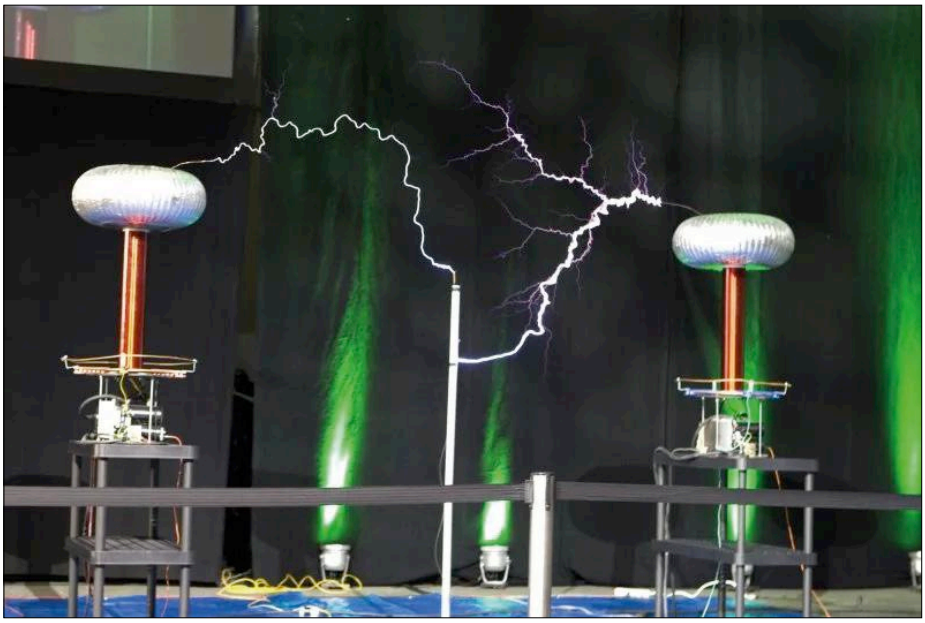


Figure 1. We had the chance to meet ArcAttack in 2010 during NIWeek in Austin, Texas.

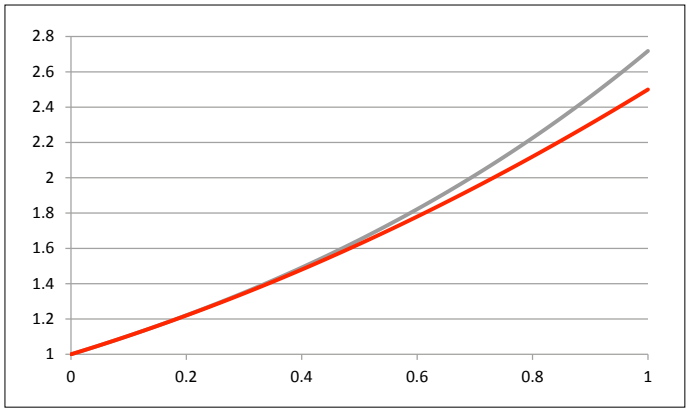


Figure 2. The function e^x (grey) and our naïve approximation (red) diverge quickly for larger values of x.

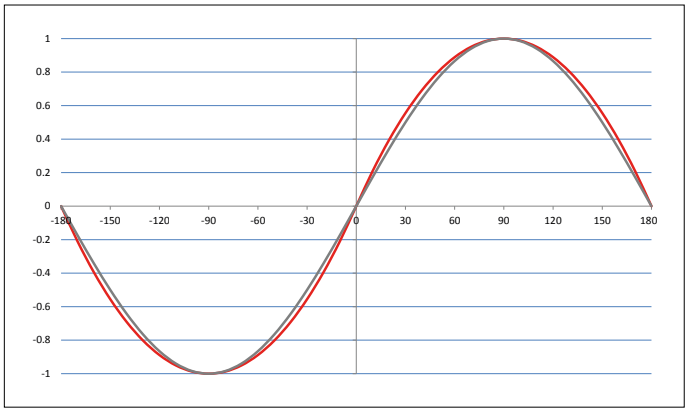


Figure 3. The quadratic approximation (red) of the sine function (grey) is more than good enough to be used as a low-frequency vibrato oscillator.

COMPONENT LIST

Resistors (0.25 W)
R1=1kΩ
R2=220Ω
R3=100Ω (see text)

Capacitors
C1=100nF, 5mm pitch (see text)
C2=100µF 6.3V, 2.5mm pitch (see text)
C3=1nF, 2.5mm pitch (see text)

Semiconductors
D1=1N4148
IC1=CNY17-3

Miscellaneous
K1=5-way 180° DIN socket
K2=3-way pin pinheader, 2.54mm pitch
PCB #160592-1

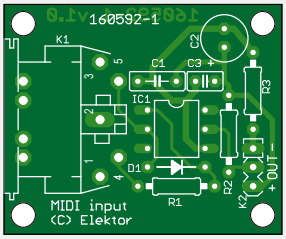


Table 1. Mapping of MIDI Control Change (CC) messages to Tesla Synth actions.

CC code	Action
1 (0x01)	vibrato depth
5 (0x05)	portamento speed
12 (0x0c)	vibrato speed
65 (0x41)	portamento on/off
72 (0x48)	release time
84 (0x54)	portamento control

note. Releasing the new key while the previous key is still being pressed does not restart the previous note.
Refer to **Table 1** for the MIDI implementation chart of the Tesla Synth. Where possible standard Control Codes (CC) have been used. For instance, CC 1 usually corresponds to the modulation wheel on a keyboard. The pitch bend control of a MIDI keyboard does not use a CC code, but has its own MIDI status (command) byte 0x0b. Note that this table shows a CC code for release time. This control sets the time it takes for a note to die away after it is being released. The Tesla Synth uses it to "fade out" the pulse width, but the result is not the same.

Portamento is only applied when playing legato-style

The worker task also runs the low-frequency oscillator (LFO) used for vibrato effects. This means that the LFO signal is sampled at 1 kHz, which is more than enough for the maximum LFO frequency of 20 Hz.
Being a real busy bee, the worker task makes sure that pitch bending is applied properly (±2 semitones) and it keeps track of portamento. In the Tesla Synth this effect is implemented as a linear slide from note A to note B. The slide duration is independent of the distance between notes A and B. Furthermore, it is only applied when notes are played legato, i.e. when notes are played without any silence between them. This allows mixing notes with and without portamento without touching any controls. The CC code for portamento control is not doing anything (yet).

Faster math with less precision

Pitch bending requires complicated mathematical operations in order to obtain musically pleasing results. The main problem is to calculate the pitch bend value raised to the twelfth power of two because there is no such function in C. Luckily there is a way around this:

$$2^{\left(\frac{x}{12}\right)} = e^{\ln\left(2^{\left(\frac{x}{12}\right)}\right)}$$

The natural logarithmic (ln) part can be simplified as follows:

$$\ln 2^{\left(\frac{x}{12}\right)} = \left(\frac{x}{12}\right)\cdot\ln(2) =$$

$$x\cdot\frac{\ln(2)}{12} = x\cdot 0.057762265$$

Instead of the twelfth power of two we must now calculate a power of e:

$$e^{(x\cdot 0.057762265)}$$

C has the function `exp()` to do this, but it is rather slow due to its high precision. Replacing it by a cut-off Taylor polynomial approximation, we obtain an approximation good enough for our needs (**Figure 2**):

$$y = x\cdot 0.057762265$$

$$e^y \approx 1 + y + 0.5\cdot y^2$$

A similar problem crops up in the LFO where we want to calculate the sine of a phase angle. The built-in function `sin()` is way too slow and must be replaced by something else. An often-used solution is a lookup table, but our microcontroller doesn't have enough memory for that,

at least not with all the Smalltask stuff included too. On the Internet I found a nice solution based on fitting parabola that does the job well [2], see **Figure 3**.

USB polling is yet another task

Adding a USB MIDI converter to the program means a lot of code, but almost everything of it can be copied verbatim from the Microchip audio USB example for the low-pin-count USB development kit for PIC18F14K50 included in the Microchip Libraries for Applications (MLA) package. I removed the MIDI send part and added the MIDI receive part.
USB MIDI messages are identical to traditional MIDI messages but prefixed with a package header byte. Also, MIDI messages are always four bytes long with the unused bytes set to zero. With the package header byte removed and the zero bytes ignored, what remains is a normal MIDI message that can be sent to the worker task described above.
The USB example requires USB driver polling at about 1 kHz which makes it a great candidate for yet another Smalltask task.

Your turn now

This completes our description of the Tesla Synth. The software can be downloaded from the webpage accompanying this article [3]. It is an MPLAB X project and, at the time of writing, compiled fine with version v3.60. To program the HEX file into the microcontroller a programming adapter like the PICkit 3, ICD3 or ICD4 is needed. Connectors K3 and K4 are intended for this.

(160592)

Web Links

[1] www.elektormagazine.com/160498

[2] http://forum.devmaster.net/t/fast-and-accurate-sine-cosine/9648

[3] www.elektormagazine.com/160592

Hardware, power & interference considerations

The MicroTesla hardware does not have a MIDI input; it doesn't even have a serial interface. All it has is a USB port. A real MIDI input is only needed if the Tesla Synth has to be controlled by a MIDI keyboard, for computer-controlled playing the USB port is enough. The MIDI adapter (**Figure 4**) must be connected to IC5, port RB5 (pin 12), and to make it work properly R15 must be removed (**Figure 5**). In normal situations R3 can be replaced by a piece of wire and C1, C2 and C3 may be omitted. No matter which solution you choose, MIDI, USB or both, without proper precautions you will run into problems due to the interference created by the Tesla coil. Its sparks may be nice to look at, remember that a century ago Tesla coils were used to build spark-gap radio transmitters. We don't do this anymore but the underlying physical principle is still valid. It is therefore necessary to shield the circuit with for instance bits of aluminium foil connected to ground. If you don't do this, you may crash your computer, or its USB port. Traditional MIDI is more forgiving but lack of response, missed messages and hanging notes will be your gain.

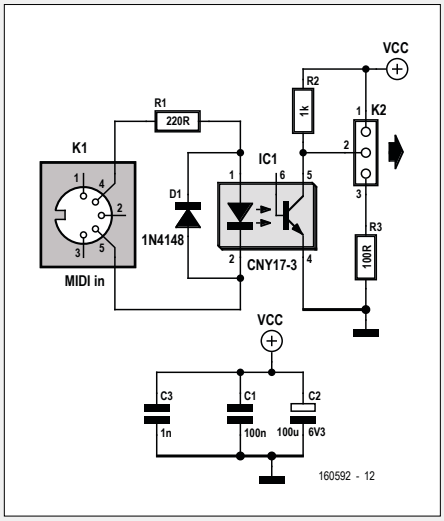


Figure 4. A simple MIDI input adapter with lots of protection and decoupling options.

The power supply is something else to be concerned about. The MicroTesla (synthesizer) sucks quite a lot of current when sparking and connecting it directly to a USB port on your computer is not a good idea. Use either a USB hub with external supply capable of delivering the required current or connect a second USB connector to K4 (D-, D+ and GND) and use a good USB power supply that does not occupy the USB data lines.

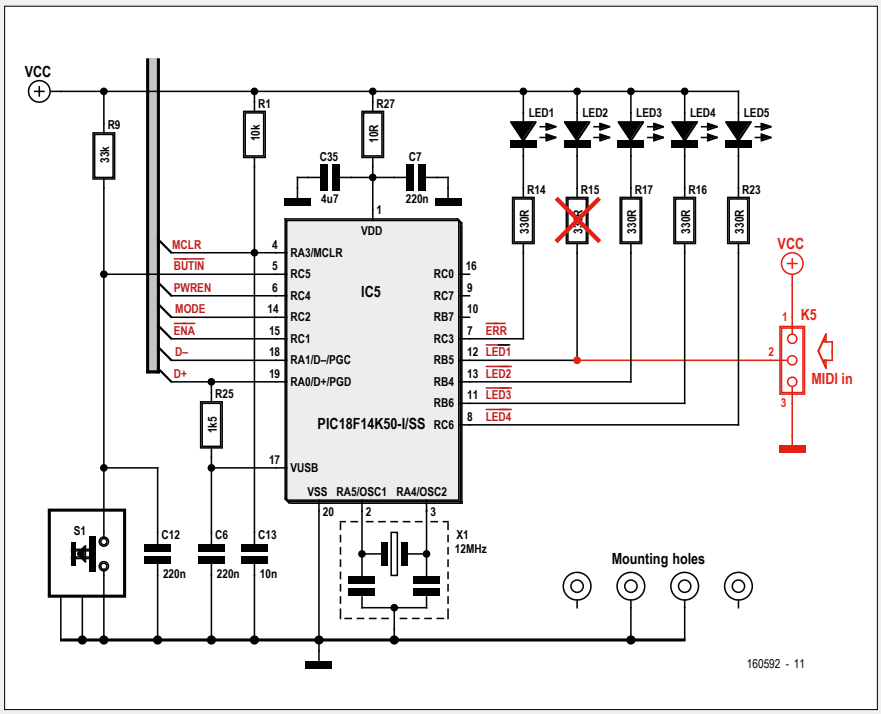


Figure 5. Connect the MIDI input adapter to port RB5. Do not forget to remove R15.

EPS — Easy Parking System

or how to avoid scratching your car

By **Olivier Croiset** (France)

Last year, my son arrived in his car with the doors all scraped. Now I know he's not terribly good at manoeuvring (he does, fortunately, have plenty of other qualities!). He explained that the carpark in the building where he lives is cramped and the underground passage leading to his parking space is narrow. It's not easy parking his car under these conditions. I soon had an idea for overcoming this situation: estimating the distance each side of the car as it enters the garage space. A central unit gives a simple visual warning (illuminated indicators) if the limit distances are reached or even exceeded.

There is no electricity in the garage, and the passage is poorly lit. So the EPS (Easy Parking System) has to be self-contained and consume very little, yet offer reliable help with parking difficulty; hence it is powered by 1400-mAh AA cells. This battery will be recharged regularly, approximately once every two months. The consumption in deep sleep (power down) is just 400 µA.

If the EPS is installed in a garage with line power, it is of course possible to dispense with the battery and power it using an AC line (mains) PSU; the stand-by function will however still be active.

I did consider powering the EPS using a portable USB battery (power bank). Unfortunately, it seems these require a certain current to be drawn so as to deliver their charge. This minimum current, as well as the time for which this current is available, varies from one manufacturer to another. What's more, this type of device goes into stand-by when the current is very / too low. Now my circuit consumes as little as possible: less than 500 µA in stand-by, a few tens of mA for 10 min. per day (the time to park). So I abandoned this idea, as it's hard to work with a power source you don't know the characteristics of, and settled instead for a battery comprising four 1.4-V (nominally) cells.

Circuit

The heart of this project is the ATmega328P microcontroller (**Figure 1**). The ultrasonic transducers are positioned either side of the entrance to the garage-space. They are powered from the central unit, which sends out the commands and receives the distances measured.

To economize the supply battery, the central unit goes into stand-by after it has been operating for 10 min. To wake

it up, all you have to do is flash your headlights when you arrive in front of the parking space! No need to get out of the car to turn the EPS on!

Note that I have not tested the circuit in surroundings brighter than a gloomy basement. In full daylight, flashing the headlights will undoubtedly not be enough to reactivate the circuit.

It must be possible to use an infrared detector (like the LHi 878 found in a presence detector). Unfortunately, I didn't manage to get any reaction from one of these (it may have been dead). I think that would be a solution worth investigating; in addition, there would no longer be any need to flash the headlights as the heat of the engine — at the front — would be enough to get the circuit going. The drawback with this detector is that when you leave the parking space — with the engine cold — the EPS remains asleep. There had to be some drawback to the rear-engined Porsche 911!

I studied my circuit using an Arduino board, ideal for this type of project; the quartz crystal and the two capacitors are not necessary, as millimetre accuracy is not needed. The important thing in fact is not to know the absolute distances with respect to the walls, but simply to know if the car is getting dangerously close to the walls on the left or right. If we use the internal oscillator, this frees up the crystal ports, and makes it possible to add a third "sensor + LED" combination – for example, at the front.

The use of the microcontroller ports is defined in the program by the #define directives. This makes it possible to easily modify the port assignments, and thereby make the work of the CAD designer doing the PCB routing easier.

As far as possible, I usually try to leave the RX and TX ports free, so as to be able to diagnose the operation of the circuits in case of problems. A few messages on this serial port, a serial/USB converter and HTerm allow you to find out what's happening within the Atmega328 and the program.

The three buttons could be wired in parallel on the ISP port; this would mean avoiding pressing the buttons during programming, but any good electronics engineer will understand this. This makes it possible to free up some ports.

Battery indicator

The red/green LED indicates the status of the EPS (on/off) as well as the state of the battery. The divider (R6/R7) in conjunction with one of the analogue ports (port PC4 — ADC4) measures the battery voltage. This value is then compared to the microcontroller's internal 1.1-V reference. If the battery voltage is high enough (>4 V approx.), the green LED lights; if not, the red LED lights. According to their data sheet, the minimum supply voltage for the ultrasonic transducers is 4.5 V, but they operate correctly down to 4.0 V.

This divider has to be adapted to the supply battery; the limit value is determined by the program, but also by the two HC-SR04 ultrasonic transducers which need 4 V!

Ultrasonic transducers

When the EPS has come out of stand-by, the car can advance into the garage-space; the distance between each trans-

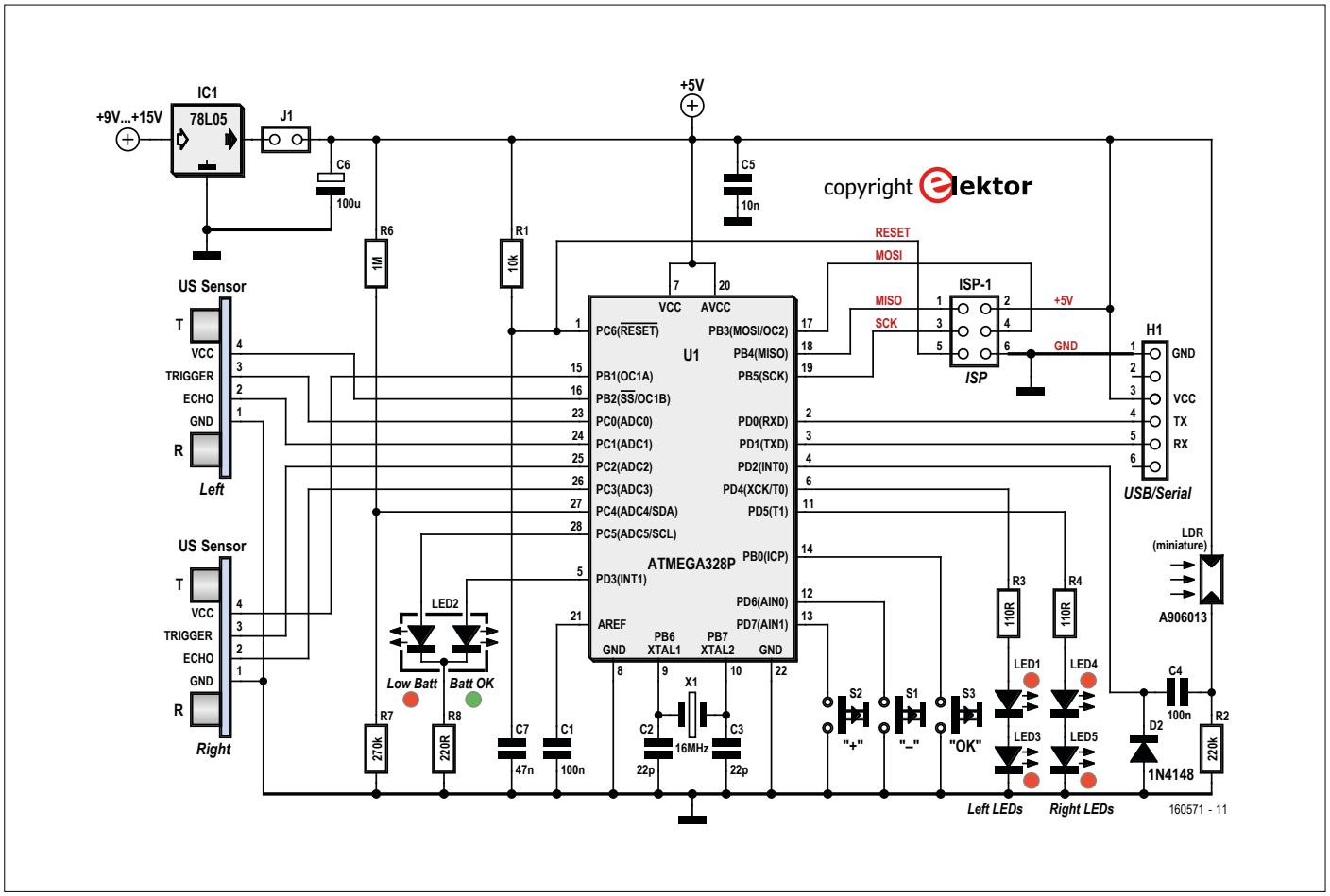


Figure 1. Circuit of the project using an ATmega328P microcontroller.

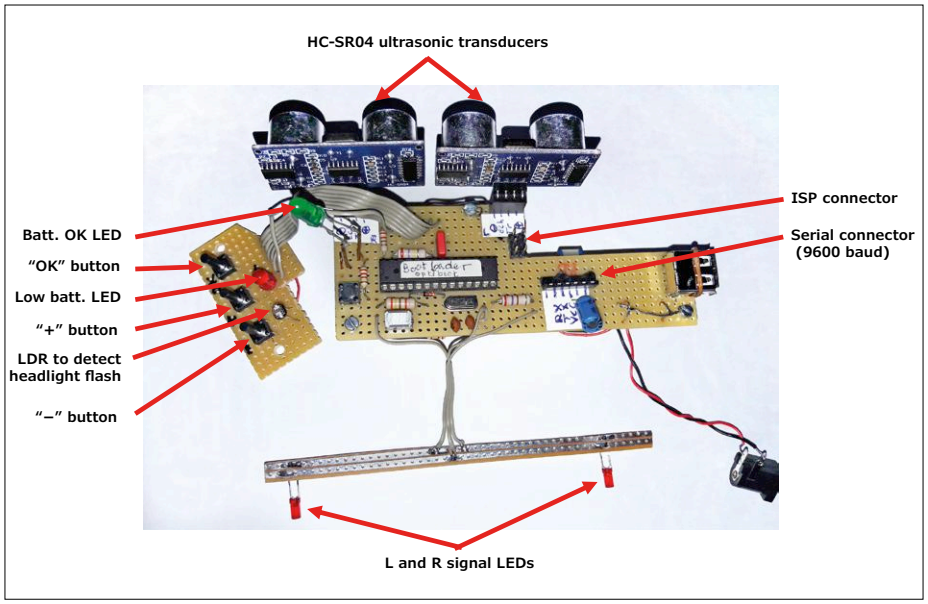


Figure 2. EPS prototype.

ducer and the bodywork is then measured continuously. The central unit powers the left and then right transducers in turn and sends them a trigger pulse. Depending on the distance measured by the left-hand/right-hand ultrasonic transducers (and the configuration, which we'll see later), the left/right indicator (each consisting of a pair of LEDs: LED1/LED3 and LED4/LED5) will light or not.
During this active period, the system consumes around 30 mA, depending on the LEDs that are lit. The most skilful drivers will save the batteries!
After operating for around 10 minutes the EPS goes back into deep sleep (power down mode). Of course it can be re-activated again for a new period of 10 min. by flashing the headlights again. But I'm sure even the least skilful will have managed to park the first time!

Activating the EPS

The EPS is started up by flashing the headlights. The LDR (ref. A 9060-13) and associated components convert this variation in brightness into a pulse sent to pin INT0. This pulse triggers an interrupt that takes the microcontroller out of deep sleep. Diode D2 is a flywheel diode, it absorbs the negative voltage during this pulse. The ultrasonic transducers are then activated, as previously described.

Getting going: applying power for the first time and configuration

When power is applied for the first time (and hence each time the supply battery is refitted), it is possible — but not obligatory — to set the distance at which the left and right LEDs are going to light to alert the driver. Here's the very simple procedure:

- Position an obstacle in front of the left-hand ultrasonic transducer at the desired distance so that no part of the car can touch the uprights of the garage. Remember to allow for the mirrors!
- Turn the EPS on (if the battery is correctly charged, the green LED will light); the left-hand indicator will flash slowly twice.
- Press the "+" and "–" buttons so that the left-hand LEDs light, then press "OK". The two left-hand LEDs and the two right-hand ones will flash twice to indicate that the setting for the left-hand channel is finished.
  Note: the "–" button reduces the detection distance right down to 0, the "+" button increases it up to about 1 metre.
- Just as for the left-hand channel, position an obstacle in front the right-hand ultrasonic transducer.
- Press the "+" and "–" buttons so that the right-hand LEDs light, then press "OK".
- The left-hand indicator LED1/LED3 and the right-hand one LED4/LED5 will flash twice slowly, indicating that the procedure of adapting to your needs is finished.

This setting is saved in the ATMega EEPROM, so it is not necessary to redo the setting each time you power up.
When you refit the battery, the EPS goes through the setting procedure; if the setting suits you, all you have to do is press the "OK" button while the left- and right-hand indicators go out (the monitoring function continues for 10 minutes).
I fitted the ultrasonic modules into KS28 G026 cases from KEMO Electronic. The most awkward thing is the two cables connecting the two modules to the central unit. Well, you can't have it all!

Here's a little circuit that less skilful drivers will appreciate, as well as older ones who sometimes have difficulties twisting round to judge all the distances in all directions. One possible improvement to the circuit would be to complement the illuminated indicators with an audible warning — of course loud enough to be heard when the car engine is running. ⏮

(160571)

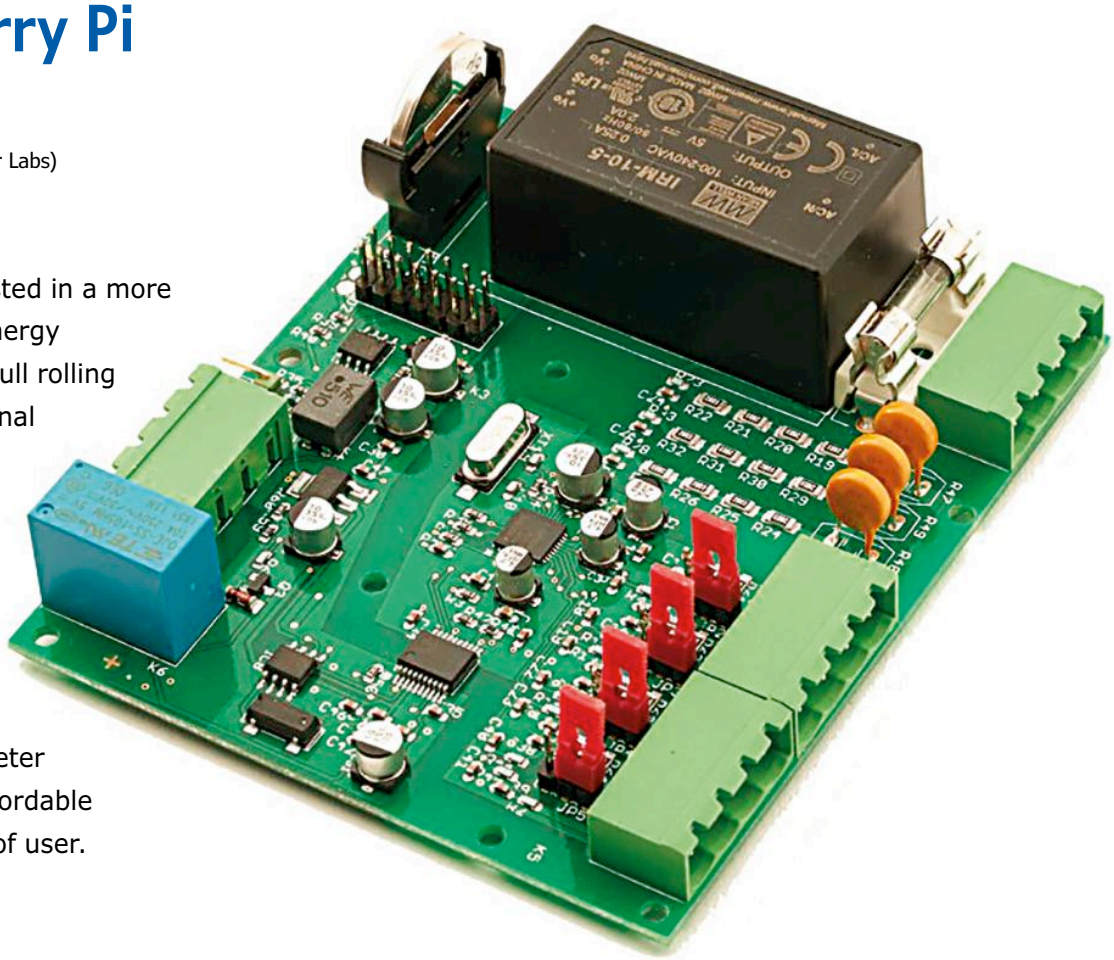
Web Links

[1] Article page: www.elektormagazine.com\160571
[2] Elektor Labs page: www.elektormagazine.com/labs/easy-parking-system-1
[3] Video of the prototype in operation: www.youtube.com/watch?v=79tF6HEACns&feature=youtu.be

SmartPi Reviewed

Smart energy meter extension for Raspberry Pi

By **Clemens Valens** (Elektor Labs)



Many people are interested in a more detailed view of their energy consumption than the dull rolling numbers on the traditional domestic watt-hour meter. Likewise, energy suppliers and network operators need better insight in "consumption patterns". The SmartPi "bidirectional" smart meter is an interesting and affordable solution for both types of user.

After a successful crowdfunding campaign back in 2016, the SmartPi was recently updated to revision 2.0. Compared to the previous version the device has become more practical with all the commonly needed connectors at the same side and a built-in power supply. All connectors now have screw terminals, improving flexibility, and an RS-485 interface is available for connecting to Modbus devices. The RPi's Real-time Clock (RTC) now boasts a back-up battery, and a relay port has been added. Finally, the new enclosure has a DIN-rail-compatible clip.

Specifications

The SmartPi measures current up to 100 A on four channels (three Phases plus Neutral), voltage (up to 390 V_{rms}) on three channels, and frequency; it calculates several types of power (active, reactive, apparent) and energy consumption. Besides measuring power consumption, it also measures energy and power generation, which is why the manufacturer rightly calls it a bidirectional smart meter.

Two types of current sensors can be used with the SmartPi: 50 mA and 1 A secondary-current devices, allowing currents of up to 700 A to be measured (this requires changing jumper settings inside the SmartPi).

The SmartPi add-on board can be bought separately, but also as a configurable kit or even fully assembled. The assembled version (reviewed here) is built on a Raspberry Pi 3 model B. The software, written in Go, is open source and available from GitHub [1]. It includes all the necessary drivers, a web server with webpage, and a REST interface for integrating the SmartPi into self-made applications and apps.

Where is the manual?

As is — unfortunately — so common these days, the manual that ships with the SmartPi is very succinct, more or less forcing you to go online. Unfortunately, things are not much better here. Note that the website is in German, but, at least for the SmartPi, webpages in English exist too. To get there you must either type in the URL yourself [2] or click the video (!) link in the shop. Actually, the Kickstarter page [3] is a much better source for information, nor forgetting the forum [4].

Following the installation link does not take you to a page explaining how to set up and install the device in your home, but instead provides detailed instructions on installing the SmartPi SD card image on a Raspberry Pi. How to hook it up to your domestic AC network is left as an exercise for the user.

The 'Use' link leads to a page with a connection drawing, but nothing is said on how to wire the device to the network. The page even suggests leaving the installation to a qualified electrician (**Figure 1**). Sound advice of course, but we would have

Figure 1. The SmartPi installed by a qualified technician (I am — really).

Figure 2. The pinout of the SmartPi voltage and current connectors.

liked to see a bit more detail here. As another point of concern, the page has not been updated and still refers to the SmartPi 1.0 that required an external power supply.

Connecting the SmartPi to a data network

The SmartPi exposes all RPi connectors, including the Ethernet connector. After cabling the SmartPi to my router, it showed up in its DHCP client list; point a browser to port 1080 at the address shown by the router and the SmartPi user interface appears.
Unfortunately my router is located far away from the electrical switchboard, so Wi-Fi would be handy. How to do this or even if this is possible is not mentioned anywhere. The RPi3 has on-board Wi-Fi, so I decided to try to configure it over SSH. The default username is 'pi' and the default password is either 'smart4pi' or 'raspberry'. In my case it was 'smart4pi'.

SmartPi & Wi-Fi

Connect to the SmartPi over SSH with PuTTY (or some other SSH client) and edit the file `wpa_supplicant.conf` using the command:

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

Add the following at the end of the file (replacing `mySSID` & `myPassPhrase` by your SSID & your pass phrase):

```
network={
    ssid="mySSID"
    psk="myPassPhrase"
    key_mgmt=WPA-PSK
}
```

Press Ctrl-O followed by the Enter key to save the file, press Ctrl-X to quit the editor. Disconnect the Ethernet cable and reboot the SmartPi. Now it's accessible over Wi-Fi. Why on earth didn't they document this?

Change the SmartPi password

Now that we are busy setting up the RPi, we might as well change the password. To do so is highly recommended by the SmartPi people, but, again, how to do it isn't explained. Here's the clue. Launch the SSH client (again) and connect to the SmartPi, then launch the Raspberry Pi configuration tool with:

```
sudo raspi-config
```

Select `Change User Password` and enter a new password.

Connecting AC power to the SmartPi

Once more, the documentation is not very helpful. Luckily the SmartPi's inputs correspond to the phases (P) and neutral (P) printed on a label stuck at the backside of the device (but not on mine, **Figure 2**).
The evaluation SmartPi came with three pre-wired current sensors. I live in France. My home has a single-phase electrical network, divided into two groups, hence I put a current sensor on each group. The third sensor was clamped on the wire going to the water heater (in the second group). Note that the sensors have an orientation, indicated with arrows. According to the manufacturer they should point in the direction of the power consumer (the building, the motor, etc.), but I found that I had to reverse them to get positive readings.
I connected the three voltage inputs together and wired them to a free fuse (I happened to have one).
Once you have managed to get positive currents and powers on all (three) channels, you can start trying to interpret the data.

Dashboard

What follows applies to software version 0.6.4.
The web interface (by default at port 1080, but it's configurable) showed three currents, three voltages and three frequencies (**Figure 3**).
The web interface has a settings page with a 'Measurement' tab where you can invert the current directions, select the

type of current sensor, and select the voltage channels to be monitored. Here you can also set the target line frequency and voltage. I had to reboot the SmartPi every time I changed a setting. Deactivating the Neutral current channel was not taken into account for some reason.

It seems that when the currents are small the measurements go haywire and the Cos Phi and frequency numbers start jumping around (I observed values from 0 to more than 50 GHz, see **Figure 4**). It should be easy to make the software handle this a little better.

When the SmartPi has collected some days' worth of data, it is time to look at its graphs and try to understand just what they show. It's also possible to export the data as a CSV file (you can set the decimal point character on the Expert tab of the settings page, but not the field separator…) and process it yourself in a spreadsheet.

The graphs are nice and scale automatically, which sometimes is nice, but sometimes it isn't, especially when there has been a large peak in the recent past. The graphs can display current, voltage, power, and energy production and consumption — just click on the buttons to activate or deactivate what you want to see. Hovering the mouse over the graph shows a cursor displaying the values and timestamp at the position of the mouse. Histograms are used for power consumption and generation over the last week.

Options and interfaces

As mentioned at the beginning the SmartPi exposes a REST API, allowing other applications to talk to it. Browsing the Settings section on the dashboard reveals more communication and interfacing options that show the intentions of the developers. Here it is possible to configure an UMTS connection, an FTP server and MQTT client support. Searching through the SmartPi repository at GitHub shows that MQTT is supposed to be working, but nothing about FTP which appears to be intended for automatic uploading of CSV files to an FTP server somewhere. Future extension will certainly also include the RS-485 port (Modbus) and the relay-switched contact.

A smart meter platform for makers

Having spent many hours fiddling with the SmartPi, its hardware and its software, what remains is a kind of mixed feeling. It is a product with a lot of potential that you would like to be a success, but for now (at the time of writing) it is hindered by unripe software and incomplete documentation. The SmartPi is (a Raspberry Pi extension board) for makers.

The software supporting it is full of good intentions, but lacks precision and finesse. New updates are released regularly though — check the forum [4] for the latest version.

The documentation is wildly insufficient. Users without RPi or Linux experience will be lost quickly. This is a pity as it should not be very difficult to add a more complete setup (Wi-Fi configuration, password management, etc.) to the configuration menu, add some Help pages, and add a few paragraphs with clear pictures to the manual. Doing so would make the SmartPi interesting for a much wider audience. On the other hand, if you are looking for a platform on which to build a smart energy consumption and/or production metering system, the SmartPi definitely is a good start. ⏮

(160485)

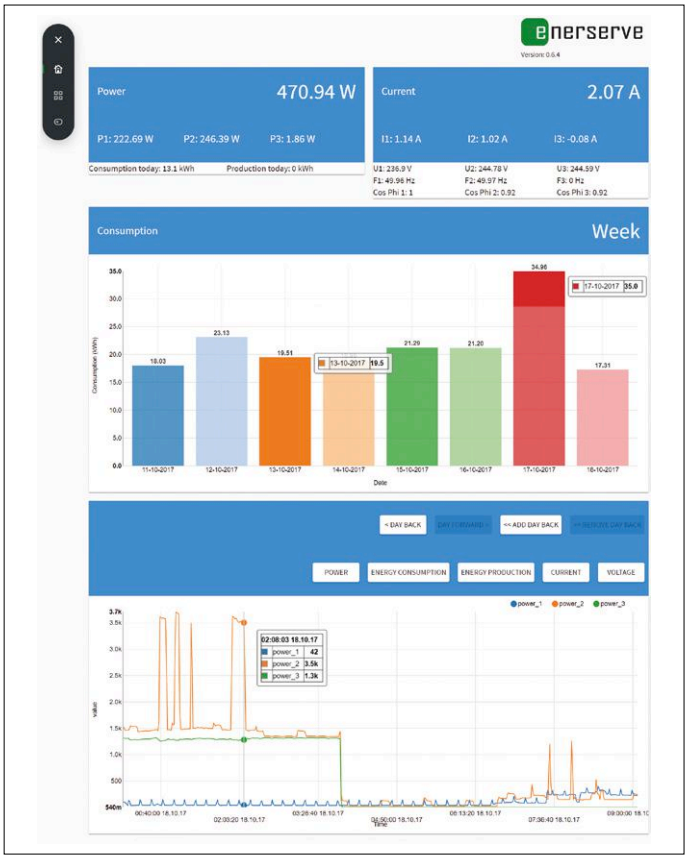


Figure 3. The dashboard shows graphs of past and current consumption (pun intended).

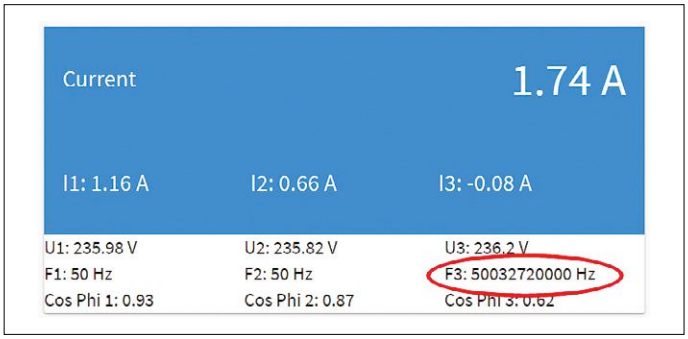


Figure 4. That can't be right. A line frequency of 50 GHz?

Web Links

[1] SmartPi on GitHub: https://github.com/nDenerserve/SmartPi

[2] Homepage: http://www.emanager.eu/en/products/smartpi

[3] On Kickstarter: https://www.kickstarter.com/projects/1240982104/smartpi-turn-your-raspberry-pi-into-a-smartmeter

[4] Forum: https://forum.enerserve.eu/

[5] www.elektormagazine.com/160485

The Intersil ICM7216

Peculiar Parts, the series

By **David Ashton** (Australia)

Our editor Jan's reminiscences in the September & October 2016 edition of Elektor on the venerable MM5314 clock IC [1] got me thinking about another IC of similar vintage but rather more peculiar — the unique Intersil ICM7216 series of frequency counter ICs.

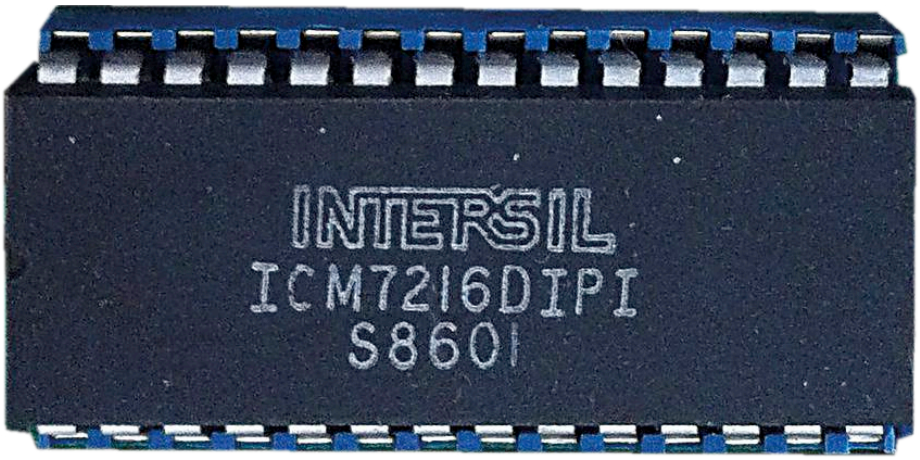


Figure 1. ICM7216 IC. Source: Eric de Paauw / CircuitsOnline.net.

These ICs qualify as some of the first Large Scale Integration (LSI) ICs — LSI was introduced in the mid-'70s. You can find datasheets dating from 1976 to 2004, from both Intersil and Harris (who were bought out by Intersil in 1999). A link to a datasheet covering all the versions mentioned below is given at [2]. Anyone who has built a frequency counter from TTL chips will know that it is not a simple task. You need an accurate and stable timebase and a generator for gate, latching and reset signals to allow the counter to count the input pulses for a precise period, then stop counting and latch and decode the signals to a display. And that's just for a basic counter. The ICM7216 series does all this and more for you in one 28-pin IC (**Figure 1**).

The ICM7216

The ICM7216 comes in four variants — A, B, C and D. The A and B versions are full featured 8-digit frequency counters which are also able to measure period, frequency ratio, units and time count. The C and D versions are basic frequency counters only. The A and C versions drive Common Anode (CA) 7-segment displays, the B and D versions drive Common Cathode (CC) displays. Their frequency range is DC to 10 MHz. The basic circuit for an ICM7216B counter is shown in **Figure 2**.

Apart from the IC, you need eight displays, a crystal, and odd other components. You can select from a 1-MHz or 10-MHz crystal for the timebase. The counter resolution is 1 in 10^8, but a simple 'xtal' will not approach this accuracy, and will drift with temperature and time.

So you can also use an external clock; a TCXO (Temperature Compensated Crystal Oscillator) would be better and an OCXO (Oven Controlled Crystal Oscillator) will have appropriate stability for the resolution. The stability from a rubidium or caesium clock (fear not, you can get

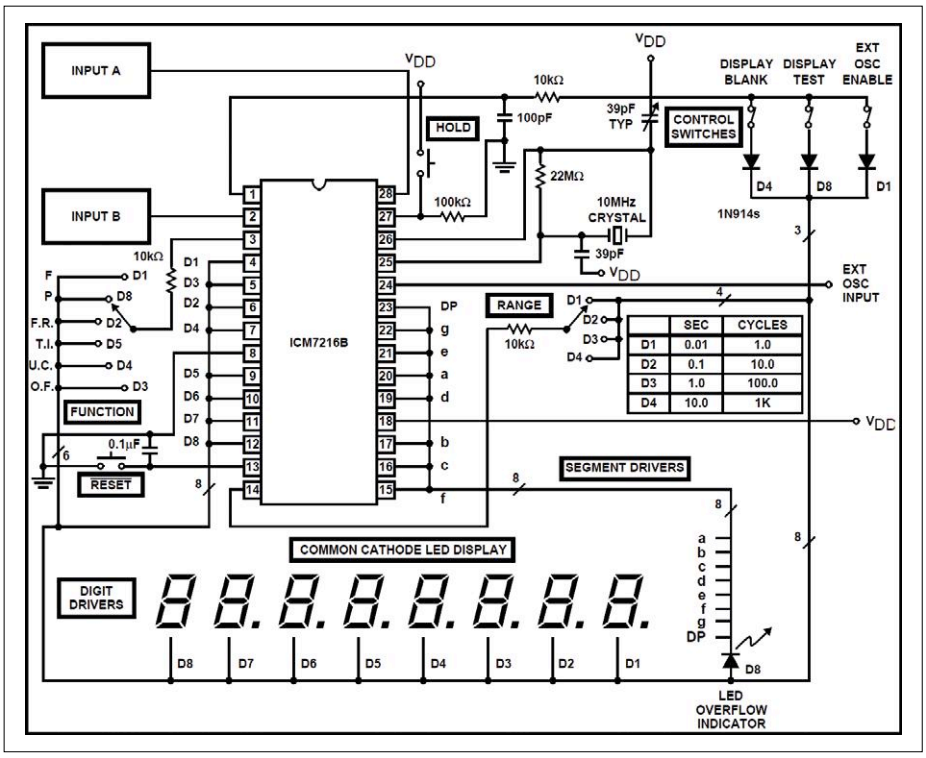


Figure 2. ICM7216 10-MHz Counter/Timer block diagram. Source: Intersil datasheet.

this accuracy from a GPS clock) would exceed the resolution. The IC will drive most small 7-segment displays directly, and automatically place the decimal point. Modes and settings are selected with cleverly multiplexed switches and jumpers. You can have switches for display test (all segments lit) and display blank (all display drive removed so you can use the display for something else). You can also add Hold and Master Reset buttons. All in all a pretty versatile little chip.

Upwards from 10 MHz

The ICM7216C/D types (which only do basic frequency counting) are great for built-in frequency displays. And the datasheets suggest a useful trick with them. If you use a 2.5-MHz crystal, instead of a 10-MHz one, the counter will overread by a factor of 4. So you use a 74S112 to divide the input by 4, and with those two trifling changes you have extended the range to 40 MHz. Very handy for testing crystals and oscillators — a lot of common ones are above 10 MHz but most of them are under 40 MHz. For the ICM7216A and -B types a better choice is to add a prescaler to get up to 100 MHz or even 1 GHz. In the era of these chips, the ECL (emitter coupled logic) 11C90 was a good 100-MHz prescaler, these days you may be lucky enough to find some 74S196 or similar fast divide-by-10 ICs. Above that, divide-by-10 ICs that go up to 600 MHz or even 1 GHz can be found without too much trouble. Depending on your input requirements, you'll probably need to add some amplification as well.

Used it, anyone?

The ICM7216 has been used in some commercial designs, notably the Sinclair (UK) PFM200 frequency counter. This was a miniature frequency counter using the same displays and cases as Sinclair's line of pocket calculators. **Figure 3** shows a PFM200 board. The high (for the time) component density was typical of Sinclair. The small 8-pin SP8660 IC is the prescaler *par excellence* for the 100-MHz range. The slider switch near the bottom selected the range. The ICM7216 — and possibly the 7226 — was also the centrepiece of a few Elektor projects of the 1980s and early 1990s.

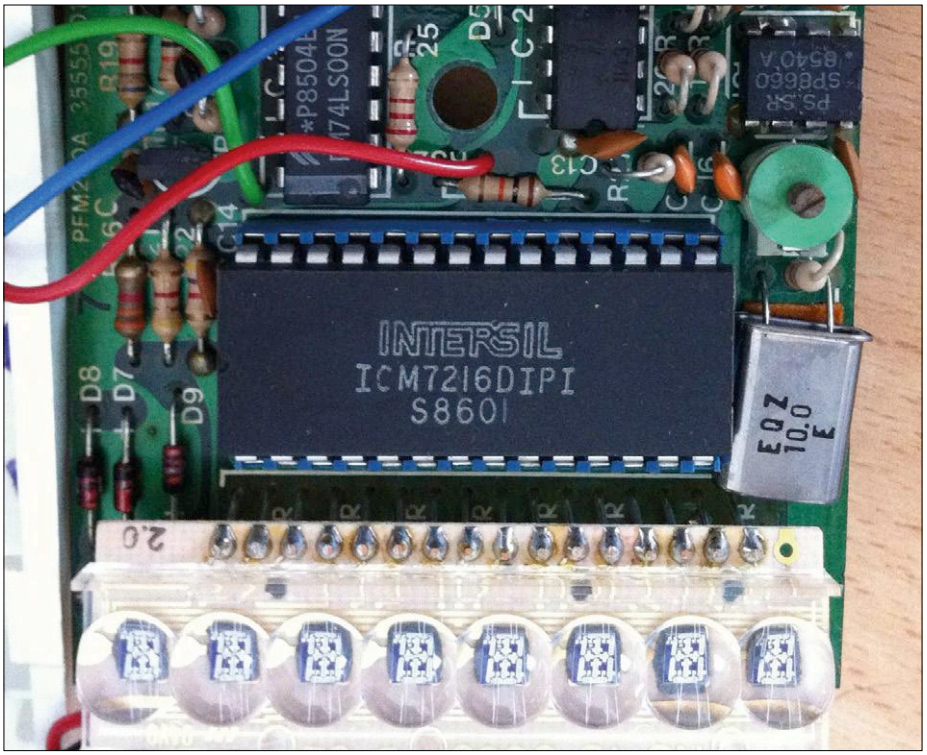


Figure 3. Sinclair PFM200 Frequency Counter board. Source: Eric de Paauw / CircuitsOnline.net.

Alternatives: yesteryear and today

These days, you'd probably use a small microcontroller, or configure an FPGA to measure your frequency. But for a quick & easy frequency counter these 40-year old chips still offer a good solution. Sadly these ICS are not cheap — expect to pay $25 or more on eBay for the -A version, or you can pick up the -D version for under $10 quite easily. But for a full-featured frequency counter, I don't think that's a lot?

There are very few other chips that even come close to these ICs. Intersil also offered a slightly later and almost functionally identical ICM7226 IC, which is a larger 40 pin chip which has BCD outputs (great for uploading or logging your readings). Apart from that there are the 3-digit 4553 and the 4-digit 74C925 series, however they are just plain counter chips and you'll have to organize any gating yourself, — and being basic CMOS they are quite slow. And to do it with standard TTL, you'd need a whole board full of ICs. So if you want to make a versatile counter, with no programming or hardware design, these ICs will do the job admirably and really you can't do much better than the reference circuits in the datasheets. I may be old-fashioned, but I have a very soft spot for these ICs of yesteryear. ⏮

(160398)

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

Web Link

[1] MOS Clock 5314 (1974), Retronics, Elektor 5/2017 p. 122; www.elektormagazine.com/160102

[2] ICM7216 -A, - B, -C, -D version datasheets: www.datasheetarchive.com/dl/255c6395adcb0ebce85a5719d13d509555f929/O/ICM7216

With a smartphone or an MP3 player it's easy to play audio files, but generally not at room strength or if you can, only in wretched quality. The Card Sound project makes it possible to fill the room with music **and** have excellent sound reproduction! This lightweight and compact player also stands out for its modest quiescent current consumption and desirable extra features such as I²C and RS-485 control.

Characteristics

- Analogue/digital supply voltages: 3 to 18 V_{DC} / 4.5 to 12 V_{DC}
- Analogue/digital current requirements: see *Measurements* panel
- Sinewave output power P_{out} into 4 or 8 Ω: 12.5 W or 62 W (measured at 10.0 V peak-to-peak)
- Output: short-circuit protected max. 2 A
- Harmonic distortion: see *Measurements* panel
- Zero-value noise: around 1.4 mW
- Data formats: PCM 16, 24, 32-bit for each of 32, 44.1, 48, 96, 128 kHz Stereo
- Dimensions (L x W x H): 89 x 33 x 30 mm
- Weight without SD card: 26 g

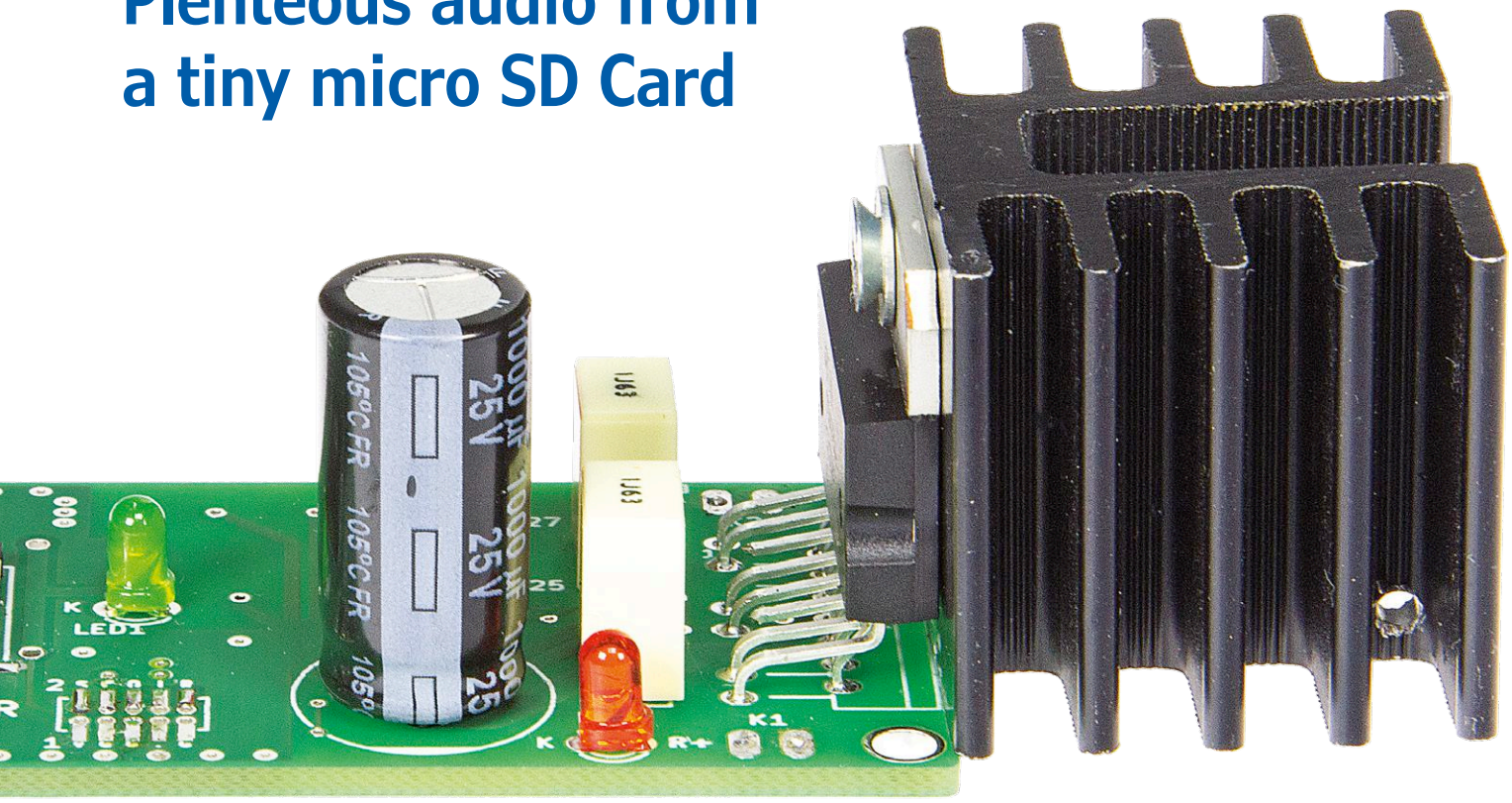
Figures valid with a supply voltage of 5 V_{DC} (digital) and 12 V_{DC} (analogue) for a 1 kHz sinewave signal with 16 bits and 44.1 kHz sampling rate

By **Falko Bilz** (Germany)

Card Sound is a small but highly capable audio player with numerous possibilities. For audio data storage it uses a micro SD card, which offers plenty of capacity at low cost. A card of this kind can be loaded with new audio material at zero cost from a PC or laptop. The project is not limited to use as an audio player; it functions equally well as the

Card Sound

Plenteous audio from a tiny micro SD Card



basis of a high-end front door bell, a melodic alarm clock or a sound module for scale models with two (!) independent loudspeaker connections. You can also adapt it as an audio add-on for your own projects (for instance home automation) and more. The number of audio files you can enjoy is virtually limitless. And last but not least, with this project you will learn how to communicate with SD cards and control a D-to-A converter (DAC) using an I²S bus. Here are the 'product highlights' at a glance:

- Good audio quality (Class AB output stage);
- 2 × 7 watts sinewave in stereo mode or with independent treble and base outputs;
- Multiple control interface possibilities: I²C, UART, RS-485, GPIO;
- 100% open source design of hardware and software;
- Single power supply voltage (optionally dual rails) for the audio output stage;
- Ability to store almost any number of music tracks of virtually any length in memory, limited only by the size of SD card used;
- Compact dimensions, minimal weight, low current consumption.

Circuit details

In the schematic given in **Figure 1** let's follow the flow of audio data, beginning with the micro SD card in its holder K5. The card is linked to the microcontroller IC2 via an SDIO data bus. The SDIO data bus makes bidirectional data transfer possible and includes a 4-bit wide data bus (D0 to D3), a clock (CK) and a command line (CMD). The finer points

PROJECT INFORMATION

Audio output
DAC
SD card

entry level
➡ intermediate level
expert level

4 hours approx.

SMD soldering equipment, TINA-TI, OpenSTM32 Compiler and STVP-STM32 Programmer, PC

£40 / €50 / $60 approx.

are available from the SD Association [1]. Two bits of each data byte are assigned to the four data lines respectively, which increases the read rate noticeably against the frequently used 1-bit data bus. The necessary pull-up resistors are enabled internally within the microcontroller by software.

The audio data on the SD card is converted into PCM (pulse code modulation) signals in the STM32F401 controller (with a Cortex M4 kernel by ST Microelectronics) according to user-defined instructions and is output to the I²S bus, a special interface for serial digital signals designed by Phillips. This data bus contains a bit clock (CK) line, a data (SD) line and a changeover line between the left-hand and right-hand channels (LRCK). Additionally we have a Master Clock line (MCK) that does not conform to the I²S specification. The fixed relationship of MCK = 256 × LRCK is defined by IC2.

The signals on the I²S bus control the CS4344 D-to-A converter (IC5), made by the audio specialist Cirrus Logic. The fixed ratio of MCK to LRCK means that IC5 can be driven only at the sampling frequencies of 32 kHz, 44.1 kHz, 48 kHz, 88.2 kHz, 96 kHz and 128 kHz (see equation at right hand side). This is a perfectly acceptable compromise, as this includes the well-known frequency 44.1 kHz used for CDs and also the high-quality 128 kHz. The software package [2] contains the file `calculations.ods`, which incorporates the Register values that are found also in the setup file `bsp_card_sound.c`.

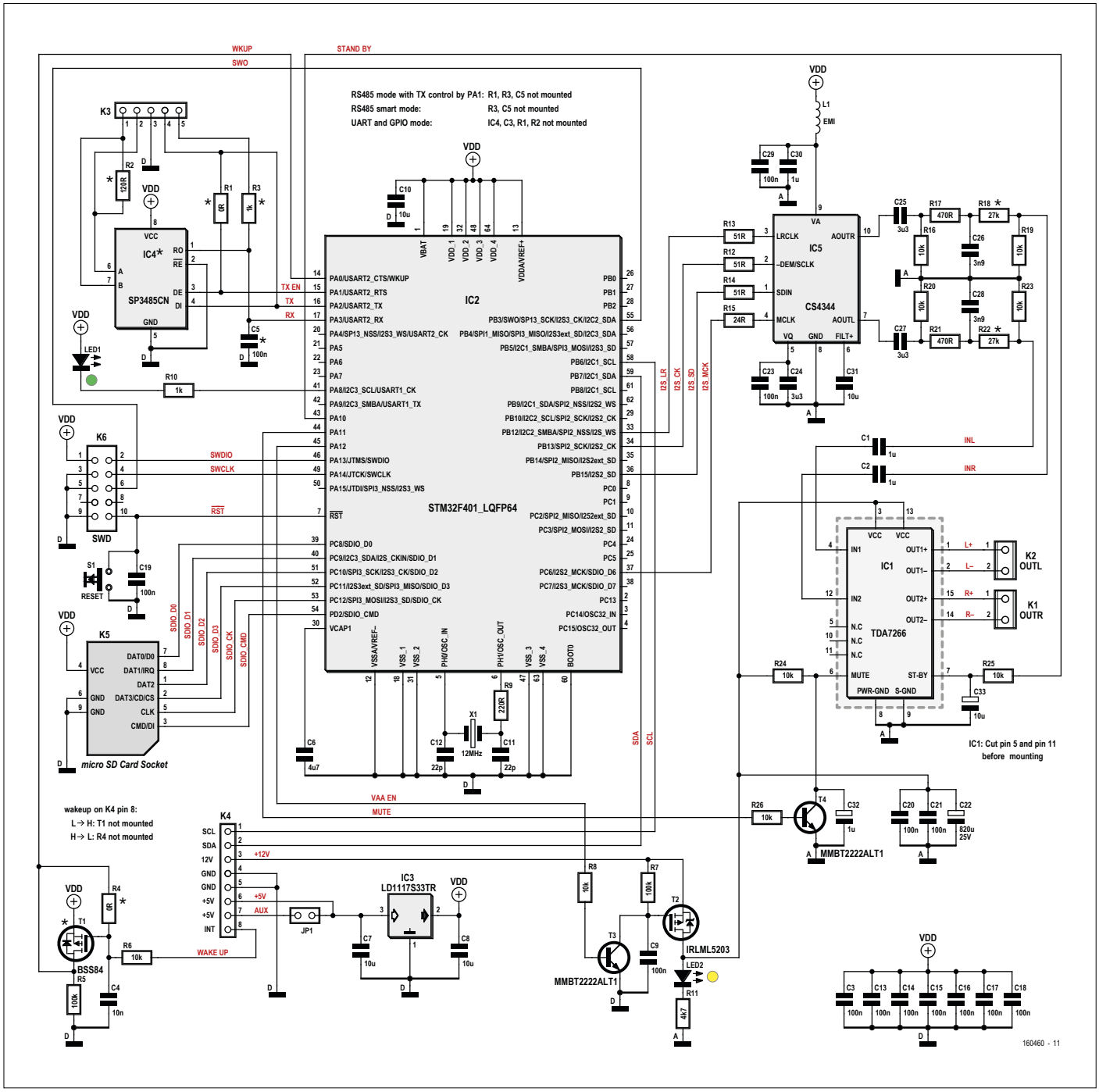


Figure 1. Schematic of the Card Sound module.

Note that the frequencies cannot be generated precisely with the 12 MHz crystal used. However, the maximum absolute frequency deviation is below 0.06%.
Resistors R12 through to R15 limit the current peaks that occur in particular on the edges of digital signals. These create lowpass filters in conjunction with the capacitance of about 8 pF on each input of IC5. The values of 51 Ω and 24 Ω are in line with the manufacturer's recommendations [3]. The same applies to the buffer capacitor C31 of the internal voltage reference and to C23 and C24, which handle smoothing of the zero-value voltage.
Since IC5 (like all other component parts of the circuitry) does not require a symmetrical power supply, for generating a sinewave of maximum amplitude we need a voltage base of half the supply voltage for the digital zero value. Conveniently this zero voltage is generated internally. The analogue output signals for both channels on pin 7 and pin 10 thus display a DC offset of approximately 1.6 V.
As you will know, interfering signals of more than half the sampling frequency arising on the output of a D-to-A converter need to be filtered out. For PCM frequencies of 44.1 kHz and above these are of course outside the audible range but they may nevertheless cause oscillation in the following stages of amplification, producing distortion. IC5 contains internal output filtering with a steep cut-off, which at a sampling frequency of 44.1 kHz suppresses interfering signals over 24 kHz by at least 50 dB [4]. At an output power of 5 W this corresponds to 50 mW.
Capacitors C25 and C27 decouple the DC voltage component mentioned. The first order lowpass filter that follows, comprising R17/C26 and R21/C28 respectively together with the input impedances R16 or R20 respectively, filters higher frequency components upwards of 100 kHz from the PCM signal by at least 5 dB. These component values are also in agreement with the data sheet for the D-to-A converter [4].
Finally the voltage dividers R18/R19 and R22/R23 attenuate the output signal (including interference products) by around 15 dB. Frequencies above 1 MHz are this suppressed by at least 70 dB in total. The resistor values are dependent on the analogue supply voltage, so that on one hand the analogue amplifier IC1

$$LRCK = \frac{12\,MHz \cdot PLLI2SN}{Bits \cdot Channels \cdot PLLM \cdot PLLI2SR \cdot (2 \cdot I2SDIV + ODD) \cdot 4 \cdot (2 - CHLEN)}$$

Using this formula you can use the Register values of IC2 to calculate the resulting sampling frequency.

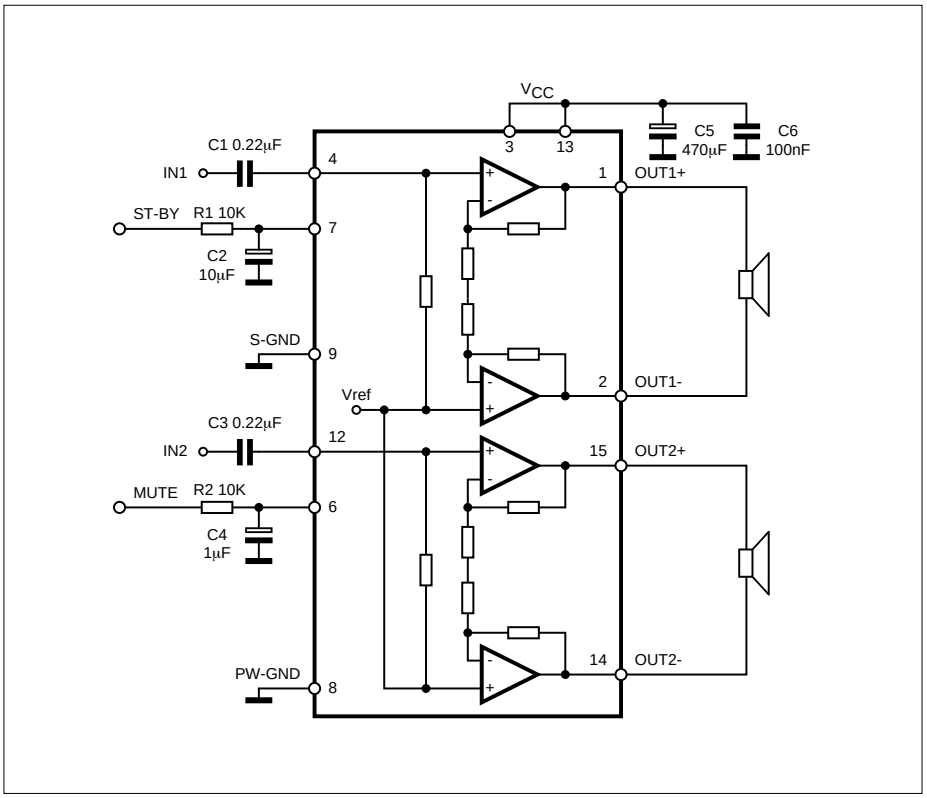


Figure 2. Internal circuitry of the TDA7266.

is not overdriven and on the other the full digital dynamic range of IC5 can be exploited. Relevant guideline values are given in **Table 1**.

Output stage

The integrated power amplifier IC1, a TDA7266 by ST Microelectronics [5], requires hardly any external components, as the block diagram in **Figure 2** shows. The degree of amplification is fixed at 26 dB. AC input voltages are thus amplified by a factor of 20. The IC limits output current to 2 A. IC1 in its multi-leaded package style (which the manufacturer calls *multiwatt*) with 15 connections also operates with an asymmetric supply voltage (V_{AA}) and generates internally a virtual earth (ground) at around half the level of V_{AA}. Unfortunately this voltage is not taken out to a Pin, even though the amplifier has three unused Pins. Consequently two additional decoupling capacitors (C1, C2) are necessary. Technically speaking the virtual earth is not created until at least +1.8 V is applied to pin 7 (ST-BY). A minimum voltage of 4.1 V (greater than the voltage of the digital section) may be necessary at Pin 6 (MUTE) and the outputs

Table 1. Optimal values for the attenuator elements.

$V_{analogue}$ [V_{DC}]	R18, R22 [kΩ]	R19, R23 [kΩ]
5	27	3.3
12	27	10
18	15	10

are enabled by T4. In the process R24 limits the current flowing through the transistor. The loudspeaker, as is normal with hi-fi amplifiers, is connected direct to K1 and K2 without decoupling capacitors. In purely arithmetical terms the amplifier produces an output power of 7 W, but only when using a supply voltage as shown in **Table 2**.

With the addition of the 3.3 V fixed voltage regulator IC3 in the digital section that supplies the microcontroller and the SD card, the circuitry is now fully functional. But in that case what role do the other components play? The supply voltage for the audio output stage IC1 is enabled as required by T2. This reduces the quiescent current requirement. LED2 lights up when the amplifier is powered. The FT Pins of IC2, in which PA12 also counts, are 5 V-tolerant; a maximum of 5.2 V is permissible. T3 takes care of level matching. If PA12 were to control T2 directly by R8, the voltage drop across R8 (especially in standby mode with only a small current flowing from/to Pin PA12) might drop very low, allowing the voltage on the controller Pin to rise high enough to destroy it.

The controller handles various power modes. The most frugal 'deep sleep' is known as Standby mode and according to the manufacturer consumes just 4 µA at room temperature. Note that this state can be terminated by a positive (rising) edge on pin PA0, a Reset pulse on Pin 7 or a 'hard reset'. A little trick: if you connect the SDA line with the 'Wakeup input' INT on K4, a start sequence on the I²C bus also rouses the controller from its standby slumber. The software also lets you activate other power modes for idle periods, for instance to allow control via RS-485. The I²C bus is available at pins 1 and 2 of K4 and the RS-485 bus on pins 1 and 2 of K3.

At around 10 mA the quiescent current of voltage regulator IC3 is relatively high. If you consider this too high for a specific application, you need to switch off the digital power supply by external means. Jumper JP1 is intended only for test purposes. The controller is clocked at 12 MHz by an external resonant circuit of X1, C11 and C12. R9 raises the *Q* of the resonant circuit and in the process helps minimise jitter (temporary variations in the MCLK cadence). The audio quality benefits from this.

The Card Sound module ought to be suitable for as many applications as possible. Some may not be simple to determine in software. To set certain control functions, you will be better off picking up the soldering iron and making a couple of changes to the hardware (**Table 3**). A small modification of component values enables you to optimise one channel in the base range and the second equally for treble. Sample component values are set out in **Table 4** together with the simulated frequency range.

Table 2. Output power at various supply voltages to the audio final stage.

Loudspeaker impedance [Ω]	P_{OUT} per channel [W]	$V_{analogue}$ [VDC]
4	7	7.5 up to 18
8	7	11 up to 18
16	7	15 up to 18
4	3	5
8	1.5	5

Table 3. Functions requiring specific soldering operations.

Function	Components needed	Components to be omitted
RS-485 with playout control via Pin PA1	IC4, C3	R1, R3, C5
RS-485 with hardware playout control	IC4, C3, R1	R3, C5
RS-485 bus termination	R2	./.
UART in 3.3 V mode/Voltage input at K3 Pins 4 and 5	R3	IC4, C3, R1, R2
Wake-up initiated by L→H (rising edge)/ switching function by K4 Pin 8	R4	T1
Wake-up initiated by H→L (falling edge)/ switching function by K4 Pin 8	T1	R4

Table 4. Stereo or treble/bass alignment for a supply voltage of 12 V_{DC}.

Operating mode:	Stereo	Bass	Treble
3 dB frequency range	20 Hz to 65 kHz	19 Hz to 1.7 kHz	560 Hz to 65 kHz
C25, C27	3.3 µF	4.7 µF	33n
C26, C28	3.9 n	22 n	4.7 n
C1, C2	1 µF	3.3 µF	33 n
R17, R21	470 Ω	4.7 k	470 Ω
R18, R22	27 k	18 k	18 k

Software development

The software is written in C, using the so-called CMSIS and FatFS libraries. It can be downloaded from the project page [2]. Most of the files were generated with the configuration program STM32CubeF4 [6], which largely eliminates the need to study the reference manual for the STM32F401. In this project the application itself has the file `main.c` as its entry point. The audio output is implemented in the file `waveplayer.c`: a data buffer is transferred by DMA across the I²S bus to the D-to-A converter IC5. When half of the buffer has been transferred, the PCM data for the next WAV file is written into the other half. In the process we interpose dynamic volume-matching by simple multiplication, using the so-called floating point unit (FPU). The 'fade out' function for sounds also makes use of this. The end result is a seamless I²S data flow, leaving additional computation resources available for your own needs or future program expansion (perhaps for a loudness function). The remainder of the files are largely additional ones with special subfunctions.

As mentioned at the beginning, we're dealing with 100% open hardware and

software design. Naturally the related development tools are also open-source and cost-free. In some cases it is necessary to register (at no cost) on the relevant web page before downloading. The tools are:

KiCad [7] for the schematic and the hardware layout;
TINA-TI [8] for special hardware simulations (this also functions under Linux in a Wine environment);
The SW4STM32 Compiler and Linker environment [9], also known as OpenSTM32;
STVP-STM32 [10] for programming IC2 with the Hex file generated previously.

The file `CardSound.hex` is generated in the folder `software/cube/` from the source files using a Make call. Suitable programming hardware includes the Nucleo Evalboard, on which an SWD interface is provide for external use. K6 is arranged with the customary 10-pin Cortex Debug Pin assignment [11].
There are almost as many bus protocols as stars in the sky. In our sample software we have implemented the Elektorbus [12] in so-called A0 mode without CRC checking. This bus stands out for its simplicity and understandability. That said, this project places greater emphasis on its audio capability than its bus protocols. This part of the software can be tailored to your special requirements, according to the particular application you have in mind. The program sections in which the Data Union U_ELB is used are easy to spot, for modification or replacing with protocol software of your own.

Configuring the audio output

Delivering the name of the sound file for playout by I²C or RS-485 command is obvious, although the actual software is less clear. This reads the allocation file `map.csv`. in which each line is constructed as follows:

```
Command value, Audio file name,
   Volume
```

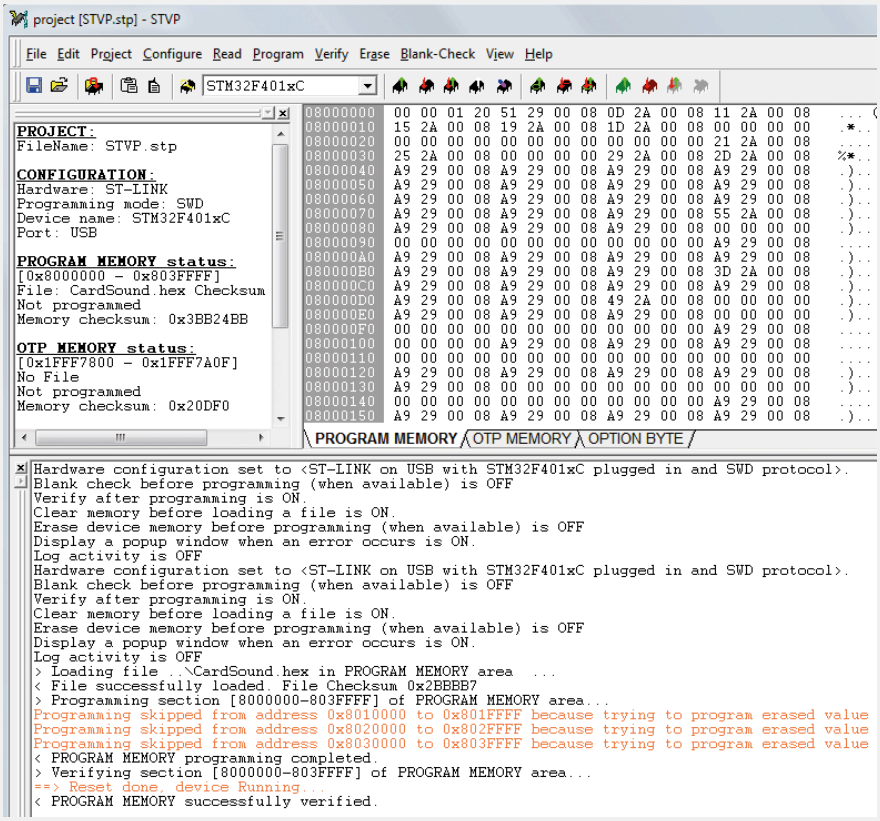
So in the following example a command with the value 1 the file `audio1.wav` is played out at full volume; command value 5 lets us play the file `audio2.wav` at half volume (the # symbol signifies a comment).

Programming the microcontroller

The STM32F401 Nucleo-64 board is a low-cost hardware device for programming the microcontroller. Remove the two jumpers marked **ST-LINK** and link the three CN4 connections **SWCLK**, **GND** and **SWDIO** to the same-named Pins on K6 of the Card Sound module.

CN4 Pin on the Nucleo-64	Pin at K6	Function
2	4	SWCLK
3	3, 5 or 9	GND(D)
4	2	SWDIO

First connect the power supply (5 to 10 V) to **Pin 6** of K4. After installing the ST Visual Programmer [10], link **CN1** of the Nucleo-64 to the PC using a USB cable. The driver is installed automatically. By using the command *Project→Open...* you can now open the project file `STVP.stp` from the software package [2] and select *Program→All tabs*. You can quit the programming operation successfully when you see the message *PROGRAM MEMORY successfully verified*.




```
# Index, file name, volume (1000
   = 100%)
1,audio1.wav,1000
5,audio2.wav,500
10,empty.wav,1000
```

There is no Stop command in the sample software; instead the command to play the audio file `empty.wav` is sent. This WAV file is stored on the SD card but it contains no sample data. The advantage of this 'configuration instead of programming' method is clear to see: less program code.

Card Sound can decode audio files in 16-bit, 24-bit and 32-bit (2-channel or stereo) format with sampling rates of 32 kHz, 44.1 kHz, 48 kHz, 96 kHz and 128 kHz. Your existing audio material can be converted into one of these

formats very easily with the programs Audacity [13] or ocenaudio [14]. In Audacity the method is *file→Audio export... →filetype*; you might for example select *WAV (Microsoft) signed 16-bit PCM*. To do this in ocenaudio you could use *file→Export... →WAV* to select the audio format Linear PCM.

Before conversion you should check the dynamic range of the existing audio file and, if necessary, upscale it to full modulation. As is well known, the 16-bit data format covers the range of values from −32768 to 32767.

The software supports so-called SDSC and SDHC cards. With these you will need to format a Partition in FAT format if this has not already been done by the manu-

COMPONENT LIST

Resistors

* : see Table 2
R1*,R4* = 0Ω, thick-film, 5%, 0.1W, 150V, 0805
R2* = 120Ω, thick-film, 1%, 0.25W, 200V, 1206
R3*,R10 = 1kΩ, thick-film, 5%, 0.1W, 150 V, 0805
R5,R7 = 100kΩ, thick-film, 5%, 0.1W, 150V, 0805
R6,R16,R19,R20,R23,R25 = 10kΩ, thick-film, 1%, 0.25W, 200V, 1206
R8,R24,R26 = 10kΩ, thick-film, 5%, 0.1W, 150V, 0805
R9 = 220Ω, thick-film, 5%, 0.1W, 150V, 0805
R11 = 4.7kΩ, thick-film, 5%, 0.1W, 150V, 0805
R12,R13,R14 = 51Ω, thick-film, 5%, 0.1W, 150V, 0805
R15 = 24Ω, thick-film, 1%, 0.1W, 150 V, 0805
R17,R21 = 470Ω, thick-film, 1%, 0.25W, 200V, 1206
R18,R22 = 27kΩ , thick-film, 1%, 0.25W, 200V, 1206

Capacitors

C1,C2 = 1µF, 63V PET, 5mm pitch
C3*,C5*,C13,C14,C15,C20,C21,C23 = 100nF, X7R, 10%, 100V, 1206
C4 = 10nF, X7R, 10%, 50 V, 1206
C6 = 4.7µF, Y5V, 16V, 1206
C7,C8,C31,C33 = 10µF, X7R, 10V,1206
C9,C16,C17,C18,C19,C29 = 100nF, 50V, X7R, 0805
C10 = 10µF, X5R, 10V, 0805
C11,C12 = 22pF, 50V, C0G/NP0, 0805
C22 = 820µF, 25V, radial, 20%, 10mm, 5mm pitch
C24 = 3.3 µF, 25 V, X7R, 1206
C25,C27 = 3.3µF, 63V PET, 5mm pitch
C26,C28 = 3.9nF, 250V, 5%, PPS
C30,C32 = 1µF, 16V, 10%, X7R, 1206

Inductors

L1 = ferrite bead 0.4Ω, 200mA, e.g. Murata BLM21BD102SN1D

Semiconductors

LED1 = LED, green, 3mm
LED2 = LED, yellow, 3mm
T1 *= BSS84 P-MOS, -170mA, -60V, 8Ω @ -10V, -1.5V
T2 = IRLML5203 P-MOS, -3A, -30V, 98mΩ @ -10V, -2.5V
T3,T4 = MMBT2222, NPN, 40 V, 600 mA
IC1 = TDA7266 (audio amplifier)
IC2 = STM32F401RCT6 ARM Cortex-M4
IC3 = LD1117S33TR (3.3 V LDO, 800mA)
IC4 *= SP3485CN (RS-422/RS-485 transceiver)
IC5 = CS4344-DZZ (24-bit 192 kHz audio DAC)

Miscellaneous

X1 = 12MHz quartz crystal, SMD, 5x3.2mm
K1,K2 = 2-pin pinheader, vertical, 0.1" mm pitch
K3 = 5-pin SIL pinheader, vertical, 0.1" mm pitch
K4 = 8-pin SIL pinheader, vertical, 0.1" pitch
K5 = Micro SD card holder, Hirose DM3CS-SF
K6 = 10-pin (2x5) pinheader, vertical, 0.05" mm pitch
S1 = Pushbutton, 6mm, single-pole NO, e.g. FSM4JRT (TE Connectivity)
Printed circuit board 160460-1 V1.5

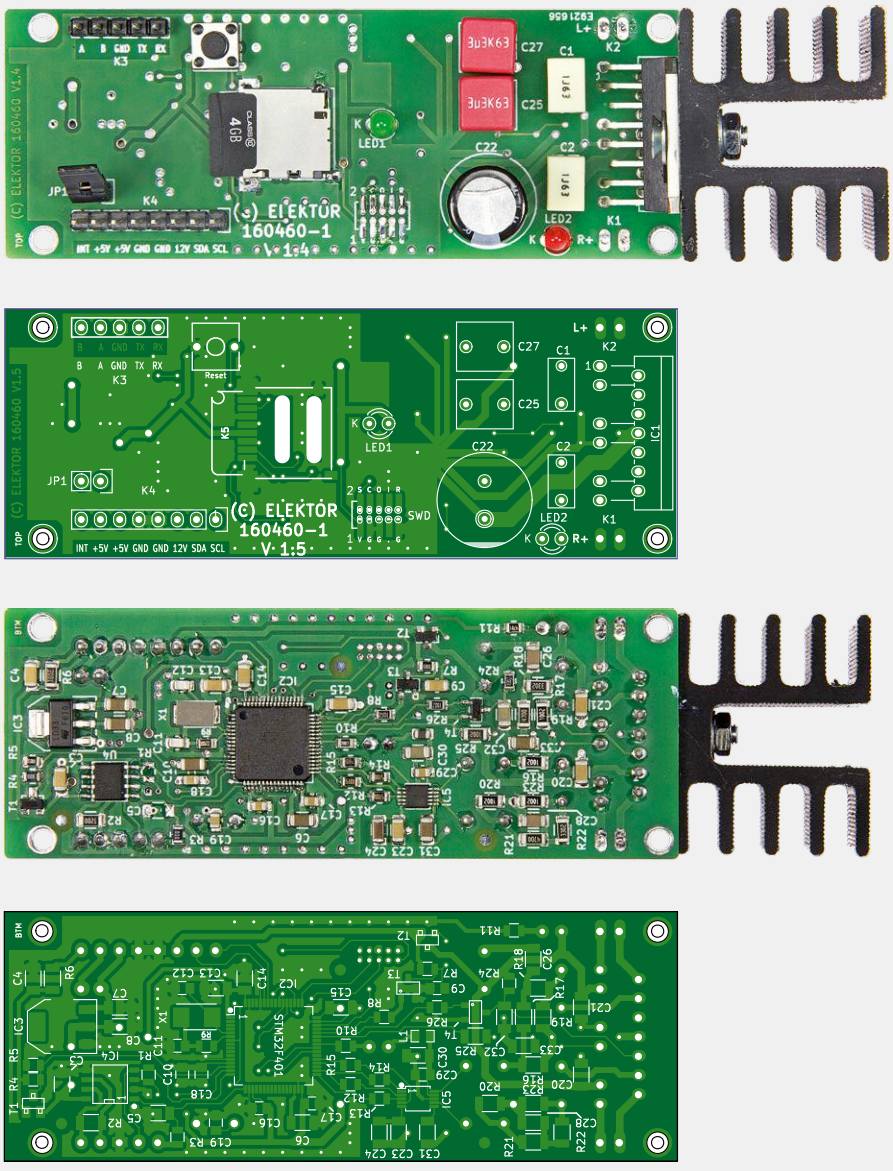


Figure 3. The double-sided printed circuit board for the sound module, with components mounted on both sides. (PCB version 1.50; images show version 1.40)

facturer. The CSV and WAV files can then be copied into this Partition.

Construction and commissioning

If you had some experience working with SMD components you will be able to solder the module yourself. The minimum distance between tracks (traces) on the printed circuit board available from the Elektor Shop (**Figure 3**) is dictated by IC2 and IC5, amounting to around 0.5 mm. The subminiature resistors and capacitors are of the 0805 type; if you solder these with an illuminated magnifying lens, a frugal amount of thin solder, plenty of flux, tweezers and a 0.5-mm or 1-mm width tip you should have success in populating the board. You must take ESD precautions without fail (gloves and grounding wrist strap) in order to avoid unexpected failures later on. As already described, not all of the components shown in the parts list need be installed, according to your intended implementation. Start first with the SMD components (but not the two somewhat heat-sensitive capacitors C26 and C28). Follow this with the wire-ended components and finally the SMD capacitors just mentioned.

Before installing IC1 you should cut short its connections 5 and 11 (**Figure 4**). The pin numbering follows sequentially from left to right (looking at the component side). If you offer up the IC into the holes as a test, you will see exactly which two connections we mean. You can re-use this trick in homebrew projects if it would lead to a more favourable printed circuit board layout.

When powering the project for the first time, you should initially connect only the power for the digital section, to pin 5 (GND) and pin 6 (+5 V_{DC}). If everything has been inserted correctly and the ICs are orientated the right way round, the current consumption should remain below 100 mA. If not, cut the power immediately and start searching for short circuits.

The next step is to program IC2 with the file `CardSound.hex` using the programming and debugging connector K6. There's a guide to this in the info panel *Programming the microcontroller*. After you press the Reset button S1, LED1 should light. Only now should you insert the SD card, onto which you have previously have copied the file `map.csv` and the sample WAV file from the software package. You can now connect power to pin 3 of K4 and a soundcheck loudspeaker to K1 or K2. After a second press on the Reset button S1, the test sound should ring out from the loudspeaker (hopefully). With that, the hardware has been commissioned successfully.

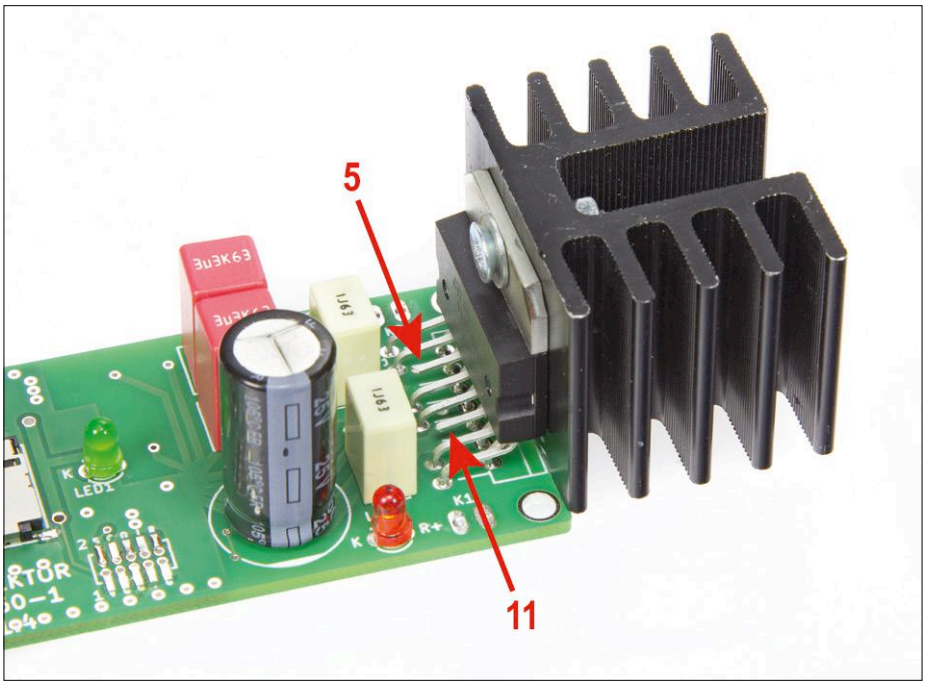


Figure 4. Amputated IC: To simplify populating the PCB we cut short two pins of the amplifier.

> With some experience in SMD work you should be able to solder the module yourself

Commands are received by the module both over the I²C bus and over the RS-485 data bus. The signal voltage for the I²C and the UART interfaces can range between 3V and 5 V. The ElektorBus protocol implemented specifies precisely 16 Bytes. Of those sent, the first Byte for synchronisation always has the Hex value 0xAA. The second Byte specifies the mode, which in our case is 0xA0h. The third Byte provides the Index in `map.csv` and can take on any value between 0 and 255 except 170 = 0xAA (this value is reserved for synchronising the data flow). For simplicity's sake the remaining Bytes have no function. For the I²C bus the device address 0x40 is included in 7-bit format in the software file i2c.c. In the software, shift the address 0x40 by one bit to the left (i.e. 0x80). This makes the device address compatible with the ST Micro software library.

It goes without saying that the software can be adapted to your own requirements. Examples might include simple polling of the voltages on Pin 4 and 5 of K3 (GPIO function) or sending acknowledgment telegrams at the beginning and end of the audio playout.

For short sound files under half a minute in length (with reasonably long pauses between these) you don't actually need any heatsink but it would work better if you screwed a heatsink (<3 K/W) directly onto IC1. That way it's guaranteed that nothing will burn out! The metallic part of the IC package has an internal conductive link to Pin 8 (Power GND), which makes it essential that when you finish construction, the heatsink is electrically insulated.

Measurements

Card Sound measurements were made in the Elektor Labs (with an earlier software version) using an audio analyser made by AudioPrecision. In the process it became clear that some of the possible file formats jarred badly at certain sampling rates. This has been corrected in the current version of the software. In the Card Sound module tested we had used ceramic types for C26 and C28. By using PPS capacitors instead the distortion values should turn out even better.

- Sampling frequency 48 kHz
- Format: Signed 24-bit PCM uncompressed
- R18, R22 = 33 kΩ (later changed to 27 kΩ in order that the amplification and also the current consumption would turn out somewhat higher)
- Modulation = 1000 (corresponds to 100%)

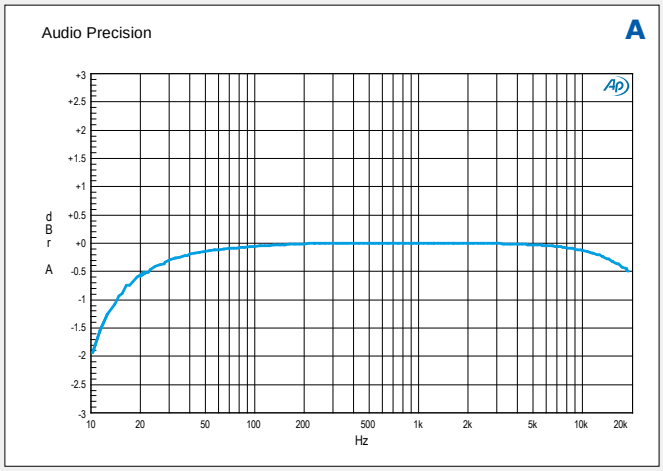
We also produced some graphs with the Audio Precision analyser:

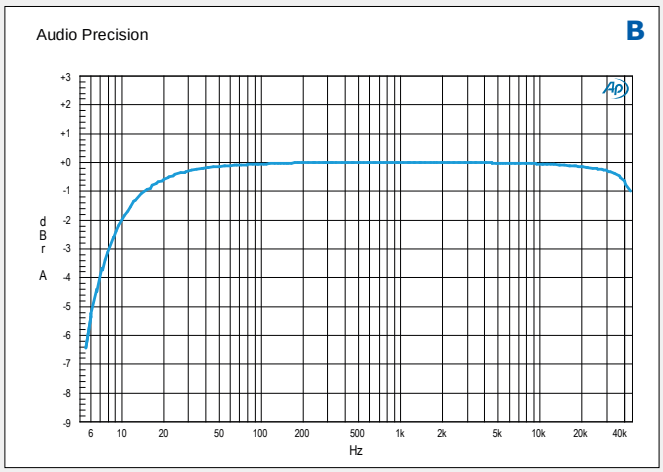
Plot A shows amplitude as a function of frequency. The test signal is a modified sinewave of 100 frequencies at 1 W output power into a load of 8 Ω and a sampling frequency of 48 kHz (signed 24-bit PCM). The result is a nice smooth curve, with attenuation of around 0.6 dB at both 20 Hz and 20 kHz.

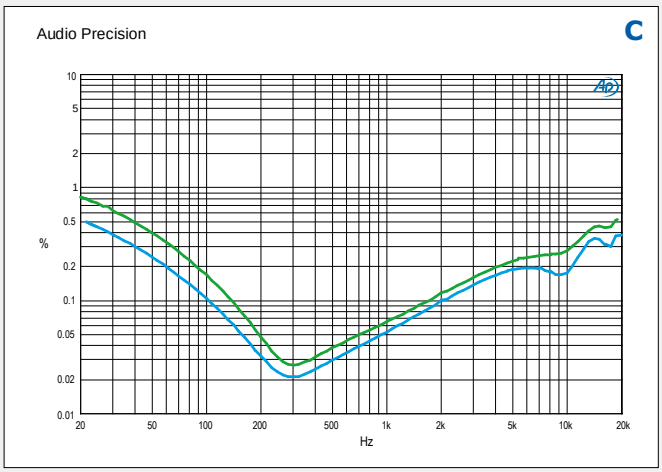
***THD+N*, 1.1 W/8 Ω**	100 Hz	0.12% (B = 22 kHz)
		0.12% (B = 80 kHz)
	1 kHz	0.023% (B = 22 kHz)
		0.049% (B = 80 kHz)
	7 kHz	0.041% (B = 22 kHz)
		0.18% (B = 80 kHz)
***THD+N*, 2.2 W/4 Ω**	100 Hz	0.18% (B = 22 kHz)
		0.18% (B = 80 kHz)
	1 kHz	0.045% (B = 22 kHz)
		0.063% (B = 80 kHz)
	7 kHz	0.16% (B = 22 kHz)
		0.26% (B = 80 kHz)
***S/N* (relative to 1.1 W/8 Ω)**		92 dB (B= 22 kHz)
		94 dB (A-weighted)
Current consumption	5 V	0.026 A (no signal)
		0.07 A (with signal)
	12 V	0 A (switched off)
		0.64 A (2 × 1.1 W/8 Ω)
		1.1 A (2 × 2.2 W/4 Ω)

Plot B again shows amplitude as a function of frequency. This time the test signal is a modified sinewave of 1 W output power into a load of 8 Ω, with a sampling frequency of 96 kHz (signed 16-bit PCM). Here again the attenuation at 20 Hz is around 0.6 dB, although at 20 kHz the value of around 0.15 dB is lower than for the sampling frequency of 48 kHz in Plot A. At 45 kHz the attenuation is 1 dB.

Plot C indicates Total Harmonic Distortion plus Noise (*THD+N*) at a bandwidth of 80 kHz as a function of the frequency. The test signal is a modified sinewave of 100 frequencies between 20 Hz and 20 kHz (signed 24-bit PCM). The curve lies well within the specified values of the TDA7266.







Circuit variants and scope for further optimisation

You can have double the output power (according to the manufacturer's specification), if you replace the TDA7266 with the pin-compatible TDA7297. The analogue section must now be supplied with at least 6.5 V_{DC}. The dynamic distortion now rises from 0.05% with the TDA7266 to 0.1% with the TDA7297 (at P_{out} = 1 W).

If you consider this power level is still too small, take a look at the mono amplifier TDA7396 (with up to 60 W into 2 Ω) along with the TDA8561Q (with an indicated 2 × 24 W into 4 Ω). These circuits also make do with a simple asymmetric supply voltage of from 8 V_{DC} or 6 V_{DC} respectively. Differing package types and in particular the symmetrical inputs of these components will, however, force you to make adjustments to the circuit that for instance might include a driver op-amp with differential outputs.

Higher signal-to-noise ratios than the CS4344 offers, with an indicated -90 dB at full power, are offered for instance by the WM8716 with –97 dB as well as the CS4361 with –94 dB. Even better values can then be achieved mostly only by using D-to-A modules with differential outputs. The follow-up analogue circuit design, which most likely will have to be constructed with multiple audio op-amps and a complex power supply, will then play in a totally different league!

Finally a word about the unused capacity reserves within the microcontroller: the 256 KB of flash memory and 64 KB of RAM should be able to support the decoding of compressed audio formats like FLAC or AAC in future. Write to **cardsound@gmx.net** if you would like to get involved in implementing this kind of decoding.

(160460)

Noise and distortion can be improved by a variety of measures, although with the penalty of higher component costs. The relatively basic power supply with an LD1117 produces around 100 μV_{RMS} of noise voltage at V_{CC}. A larger value for C8 can improve the filtering more or less audibly. Otherwise you can supply the digital section direct from a stabilised and technically clean-of-noise voltage source with 3.0 to 3.45 V_{DC} and replace IC3 with a wire strap or a 0 Ω resistor between Pin 2 and Pin 3. The jitter on the Master Clock line (MCK), which undoubtedly leads to distortion in IC5, is also worth closer consideration. The manufacturer of the microcontroller IC2 specifies 25 ps_{RMS} for the internal PLL at maximum clock frequency. This is certainly good, although with an external clock generator like the AW-11.2896MBE-T and corresponding adaptation of the circuitry you could achieve values below 2 ps. On account of the fixed clock speed the sampling frequency of the audio material used would then be defined as a fixed 44.1 kHz.

Web Links

[1] Physical Layer Specification Version 5.00: www.sdcard.org

[2] Labs project page: www.elektormagazine.com/labs/card-sound

[3] Evaluation board for CS4344: https://d3uzseaevmutz1.cloudfront.net/pubs/rdDatasheet/CDB4344-EB.pdf

[4] Data sheet CS4344/5/8: https://d3uzseaevmutz1.cloudfront.net/pubs/proDatasheet/CS4344-45-48_F2.pdf

[5] Data sheet TDA7266: www.st.com/resource/en/datasheet/tda7266.pdf

[6] STM32CubeF4: www.st.com/content/st_com/en/products/embedded-software/mcus-embedded-software/stm32-embedded-software/stm32cube-embedded-software/stm32cubef4.html

[7] KiCad: http://kicad-pcb.org/

[8] TINA-TI: www.ti.com/tool/TINA-TI

[9] SW4STM32: www.st.com/en/development-tools/sw4stm32.html

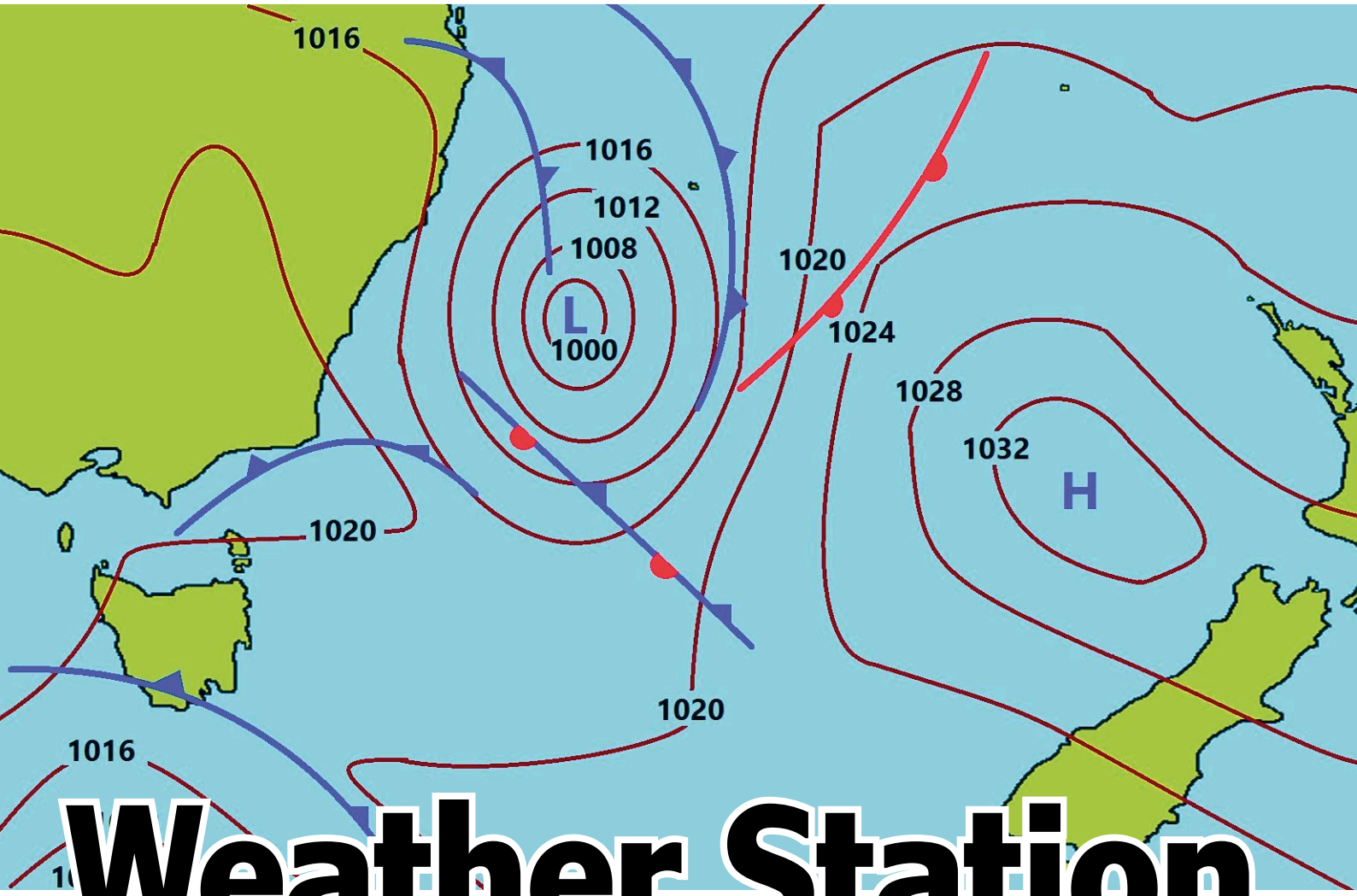
[10] STVP-STM32: www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-programmers/stvp-stm32.html

[11] Cortex Debug Connector: http://infocenter.arm.com/help/topic/com.arm.doc.faqs/attached/13634/cortex_debug_connectors.pdf

[12] ElektorBus: go to www.elektormagazine.com and search for ElektorBus

[13] Audacity: www.audacityteam.org

[14] Ocenaudio: www.ocenaudio.com



Weather Station

With no moving parts

By **Zeno Otten** (Netherlands)

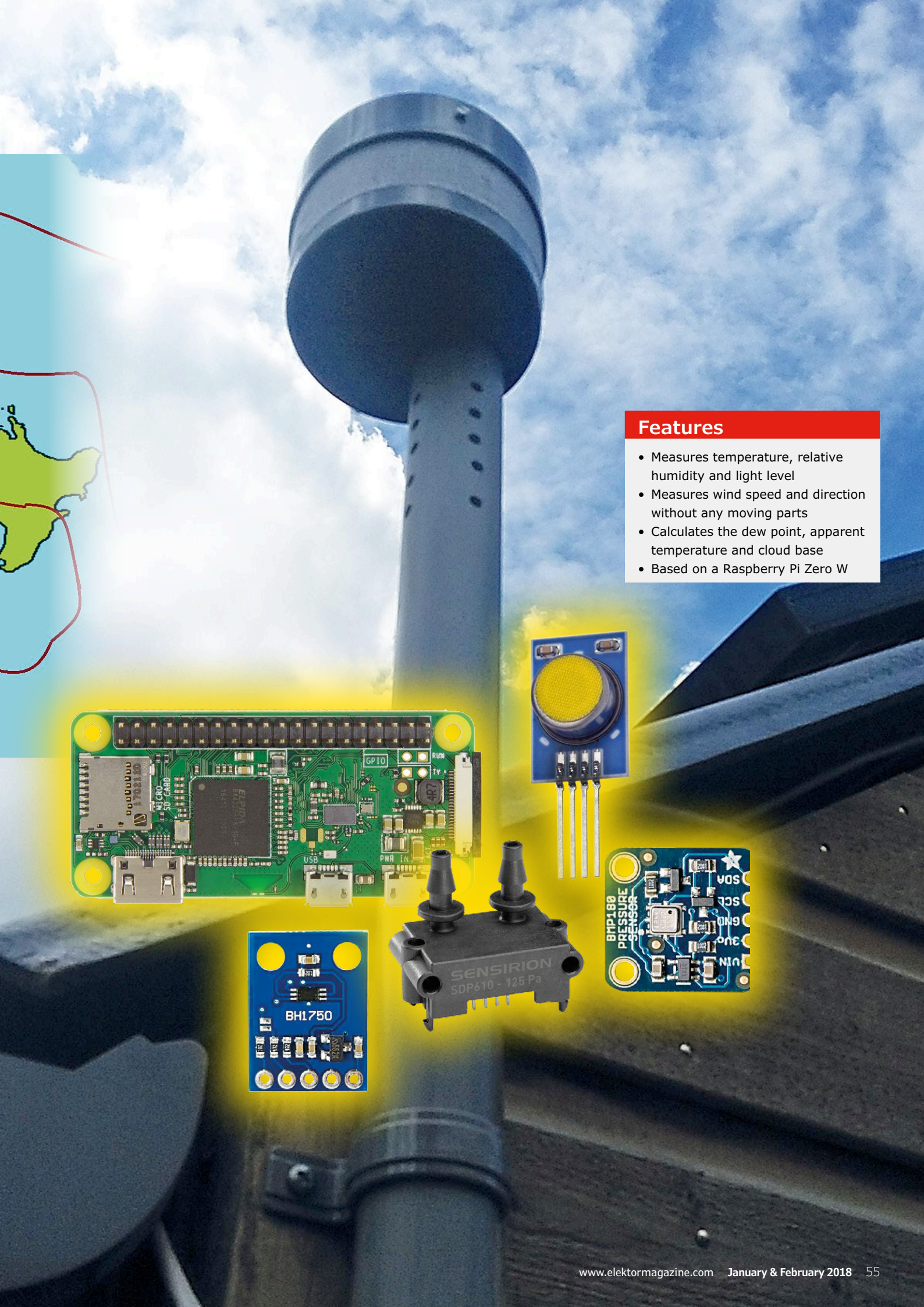
Many aspects of our daily lives are determined by the weather: whether or not you take an umbrella with you, whether you walk to work or take the car, whether you close the window or leave it open, whether you stay in bed or get up, and so on. Even though a glance out the window can tell you a lot about the weather at the moment, you need data to help you estimate what it will be later – and that's where the weather station described here comes in handy.

If cost is the only consideration, building your own weather station nowadays is hardly worth the effort. Commercial instruments made in Asia are so inexpensive (and in many cases remarkably accurate, despite the low cost) that an electronics hobbyist cannot compete with them. However, the ready-made devices have some shortcomings that justify a DIY project, and in any case building something yourself is a lot more fun than buying it.

For example, most instruments use an anemometer – usually a rotor with cups to catch the wind – to measure wind speed and a wind vane to measure wind direction. These are mechanical devices that wear out over the course of time

Features

- Measures temperature, relative humidity and light level
- Measures wind speed and direction without any moving parts
- Calculates the dew point, apparent temperature and cloud base
- Based on a Raspberry Pi Zero W



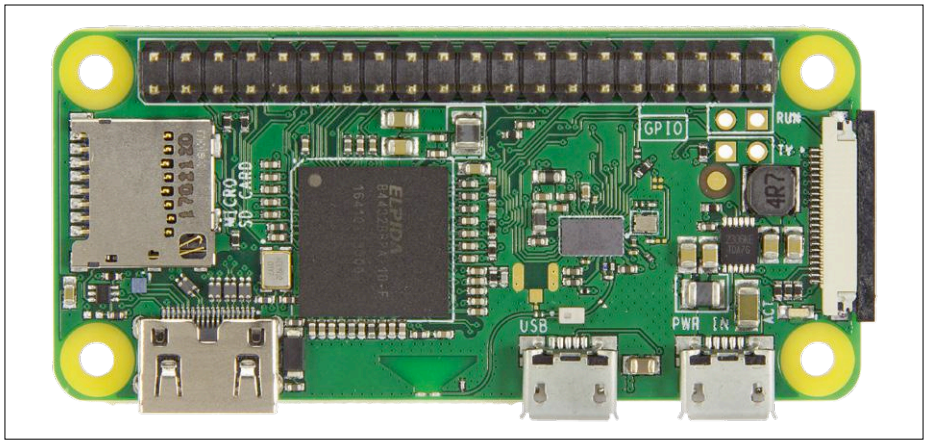


Figure 1. The Raspberry Pi Zero W.

(and depending on the quality, that can be a fairly short time) and need to be replaced. That's when the problems arise, because these mechanical components are usually not available separately.
Another thing is that communication between the sensors mounted outdoors and the base station indoors is almost always wireless, in many cases using a transmitter/receiver pair operating at 433 MHz with a non-standard protocol. That makes it virtually impossible to make any changes or extensions yourself. Finally, most weather stations do not show any derived quantities, such as the dew point or the apparent temperature.

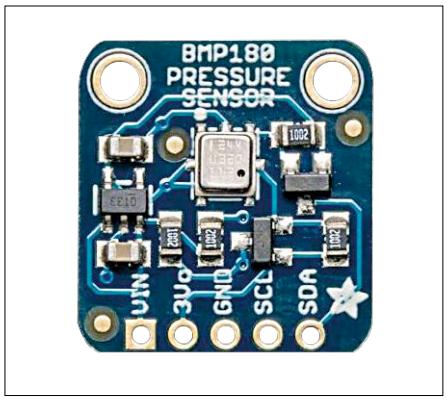


Figure 2. The BMP180 barometric air pressure sensor.

Figure 3. The HYT221 temperature and relative humidity sensor.

The challenge

For the author, all this was sufficient reason to tackle the challenge of building a weather station. The basic criteria for this were:

- A design with no moving parts
- Measurement of air pressure, relative humidity, temperature, light level, wind speed and wind direction
- Calculation of derived quantities: dew point, apparent temperature and cloud base
- All sensors connected over the I²C bus
- All sensors powered from 3.3 V
- Measurement data stored in a database
- Data presentation via a Web interface
- Communication over LAN or the Internet via WLAN
- Support for expansion (for example, with a rain gauge)
- Understandable and extensible software

The approach

A quick review of the above list of basic criteria makes the choice of the basic platform for the project fairly easy: the Raspberry Pi Zero W is perfect for this task. This compact single-board computer has sufficient I/O and processing power, and it has everything on board necessary for easy connection to the LAN or the Internet, either wired or wireless. When you also consider that this little marvel is smaller than a credit card (**Figure 1**) and only costs about 10 euros/pounds/dollars, the choice is absolutely clear. The RPi Zero W was described in detail in the September/October 2017 issue of Elektor Magazine [1], and for anyone who does not have previous experience with the Raspberry Pi or the RPi Zero, an excellent starter kit is available in the Elektor Store [12].
The various sensors are connected to the RPi Zero via the I²C bus, along with the lines for supply voltage and ground. Originally developed by Philips, the I²C bus is very popular for communication between microcontrollers, ICs or even entire modules. The I²C bus uses just two connection lines: SDA (serial data) for the data and SCL (serial clock) for the clock signal. Of course, additional lines are necessary for the supply voltage and ground, making a total of four lines. I²C employs a master/slave protocol in which the master (in this case the RPi) addresses a slave device (one of the sensors) and asks it to send the available data, such as a temperature reading. Each slave (sensor) has a unique address, which in many cases can be set (or partially set) by the user.
The bus uses pull-up resistors, so the SDA and SCL lines are at the high level in the quiescent state. These pull-up resistors have values in the range of 4.7 kΩ to 10 kΩ. In many cases these resistors are integrated into the sensors, so you have to be careful to ensure that the net resistance is within the stated range.

The sensors

With the right sensors, it's easy to measure air pressure, temperature, relative humidity and light level. It's a bit more difficult with wind speed and direction, but we can deal with this later.

Air pressure

For air pressure measurement, the author used a BMP180 [2] — a barometric sensor that can measure the absolute air pressure over a range of 300 to 1100 hPa (mbar). As you know if you have ever climbed a mountain, the air pressure depends on the altitude, so this sensor can also be used as a altimeter.

The sensor is mounted on a breakout board (**Figure 2**) and supplies air pressure data with a resolution of 12 bits, corresponding to an air pressure increment of 0.03 hPa. The supply voltage is 3.3 V, and the pull-up resistors for the I²C bus are mounted on the board. The sensor is connected to the RPi via the SCA and SCL lines.
Note: The BMP180 may be difficult to obtain. You can use the BME280 as an alternative, although some changes to the weather station firmware may be necessary. The BME280 is available in the Elektor Store [3].

Temperature and relative humidity
A robust HYT221 sensor [4] is used for this purpose (**Figure 3**). This sensor is ideal for outdoor use and can withstand condensation. The measuring range for relative humidity is 0 to 100% with an accuracy of ±1.8%, and the measuring range for temperature is –40 to +120°C with an accuracy of ±0.2°C.
This sensor also has an I²C interface with integrated pull-up resistors. The rated supply voltage range is 2.7 to 5.5 V.

Light level
Light level measurement can be used to detect darkness, overcast conditions or sunshine and convert them to numerical values. In the prototype the author used a BH1750 sensor [5], which features a large measuring range of 1 to 65,535 lux. This sensor is also mounted on a breakout board (**Figure 4**) with an I²C port, including pull-up resistors. The rated supply voltage range is 3.3 to 5 V. Typical light level values are 0.001 to 0.02 lux (night), 500 to 25,000 lux (overcast), and 50,000 to 100,000 lux (sunlight).

Wind speed and direction
The working principle for measuring these quantities with two differential pressure sensors is briefly described in the inset.

Table 1: Wind speeds	
Gale	>17 m/s
High wind	14–17 m/s
Strong breeze	10–14 m/s
Fresh breeze	8–10 m/s
Light to moderate breeze	1.5–8 m/s
Light air	0.3–1.5 m/s
Calm	0.0–0.2 m/s

The measuring range is the most important consideration for choosing the sensors. The wind speeds that can be expected in the Netherlands are listed in **Table 1**. If we assume an 'average' storm with a wind speed of 20 m/s, the formula mentioned in the inset gives a differential pressure of 260 Pa with an air density (ρ) of 1.3 kg/m³. This means that in theory you need a sensor with at least this measuring range. Fortunately, storms with such high wind speeds are not all that common here, so the author chose the Sensirion SDP600/610-125Pa [6], which has a range of 0 to 125 Pa (**Figure 6**). It has better resolution with smaller differential pressures, corre-

Figure 4. The BH1750 light level sensor.

Intermezzo: "The wind blows where it wishes"

Most wind gauges for household use consist of an anemometer (a rotor with cups to catch the wind) and a wind vane, which are moving parts. However, in the present era with the focus on software and microcontrollers, many electronics hobbyists have a strong aversion to messing about with mechanical construction, so we have to find something that suits them.
If you think about what keeps an airplane in the air, the solution is fairly obvious. An airplane wing is more curved on the top than on the bottom. As a result, the air flowing over the top surface of the wing has to travel further than the air flowing over the bottom surface, and that produces a pressure difference — the air pressure above the wing is lower than the air pressure below the wing. The net result is an upward force, or lift, that keeps the airplane in the air. This phenomenon was first described in the 18th century by the physicist Daniel Bernoulli.
Incidentally, the pitot tubes [11] used to measure aircraft speed work on the same principle. The pressure of a moving fluid (which is air in the case of aircraft, but the same thing applies to other fluids) is measured simultaneously parallel and perpendicular to the direction of motion. From this pressure difference, the speed v of the fluid (in m/s) can be calculated using the following equation formulated by Bernoulli:

$$v = \sqrt{\frac{2\Delta p}{\rho}}$$

Here Δp is the pressure difference between the measurements perpendicular and parallel to the direction of motion, and ρ is the density of the fluid (which in the case of air is 1.3 kg/m³).
For a static wind gauge (in other words, without any moving parts) you can use a vertically oriented cylinder, with the air flowing around it as shown schematically in **Figure 5**. Two differential pressure sensors are fitted in this cylinder, giving you four measuring points (ram pressure points) P1 – P4. The differential pressures between P1/P2 and between P3/P4 are measured.
The wind direction is determined by calculating how often the highest pressure is measured at each of these points during a given period and then averaging the result to obtain the actual wind direction.

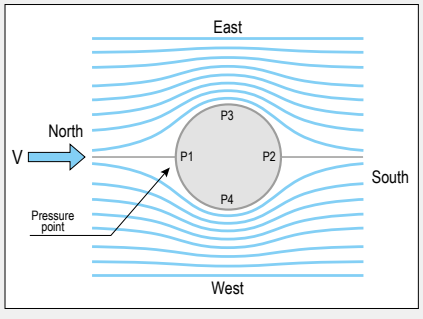


Figure 5. Air flow around a cylinder.

Figure 6. The SDP600/610-125Pa differential pressure sensor.

sponding to lower wind speeds.
The sensor operates with a supply voltage of 3.3 V, and the data is again read out over the I²C bus. However, there is a bit of a problem here. With most I²C sensors the address can easily be configured externally. However, with this differential pressure sensor the address is set to a fixed value of 0x40 in the factory. You need two of these sensors, which means you would have to use two different I²C buses. That's not very convenient.
Fortunately, the I²C address of the sensor is stored in an internal EEPROM. On request of the author, the manufacturer provided the information necessary to overwrite this address with a different value (at the cost of voiding the warranty). In this case the address of the second sensor was set to 0x21. This information, which you can also request from the manufacturer if necessary, is incorporated in a routine forming part of the weather station software (*weerstation.py*) included in the free download for this project [7]. This routine only needs to be run once, with just one of the two sensors connected.
The Sensirion differential pressure sensor does not have integrated pull-up resistors for the I²C bus, so they were added separately.

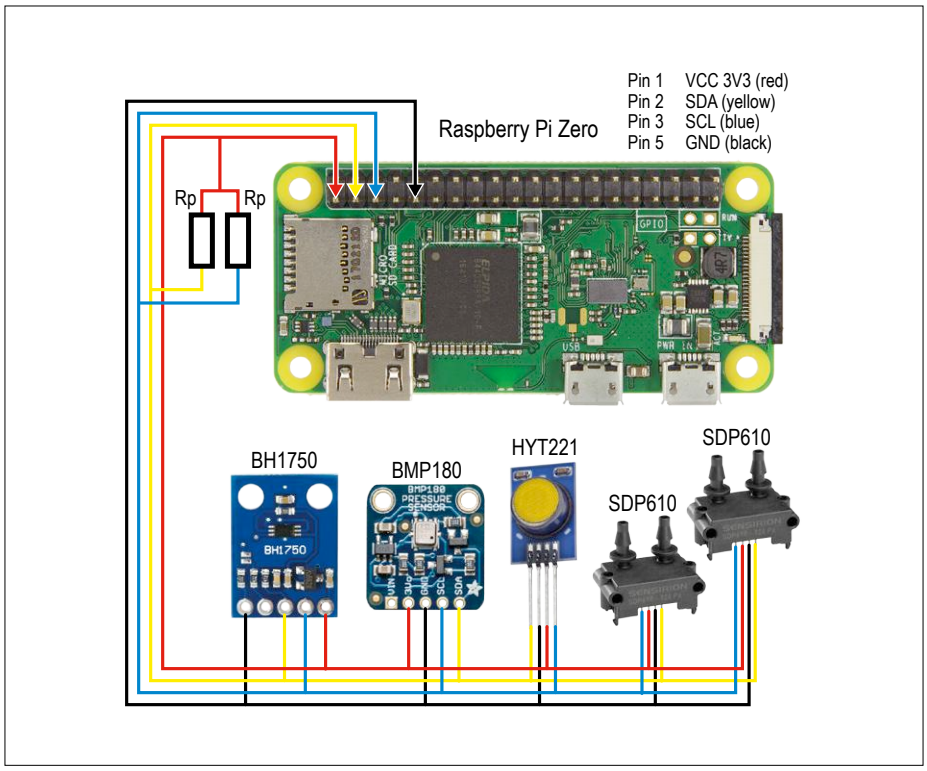


Figure 7. The sensors are connected to the RPi Zero W via a 4-wire bus.

Table 2: I²C components and addresses

Component	Address (hex)	Function
BMP180	0x77	Barometric air pressure sensor
HYT221	0x28	Temperature and relative humidity sensor
SDP01	0x40	Differential pressure sensor 1
SDP02	0x21	Differential pressure sensor 2
BH1750	0x23	Light level sensor

Derived quantities

The measured quantities mentioned above – temperature, relative humidity, wind speed and direction, and light level – are nothing special; virtually any ready-made weather station can also show them. Things get a lot more interesting when you calculate some quantities that are less ordinary.

Dew point

The dew point is the temperature at which the air is saturated with water vapour. The dew point is an important parameter in meteorology – it is used to calculate the cloud base, the chance of fog, and the chance of ice formation (which is important for aviation). A high dew point value also means that there is a greater chance of thunderstorms.
The dew point can be calculated from the measured relative humidity and the measured air temperature, as described in [8] and other sources.

Apparent temperature

There are two kinds of temperature: the actual temperature, which is what you measure with a thermometer or similar device, and the apparent temperature, which is what you feel. There can be a considerable difference between the two. For example, a measured temperature of –3°C can feel like –10°C (or even colder) when there is a strong wind and high relative humidity.

As you may have guessed, the apparent temperature is calculated from the measured wind speed and the measured relative humidity. There are various ways to do this; the author opted for the method used by the meteorological service of the Australian government [9].

Cloud base

Finally there is the cloud base, which is also a significant parameter: the lower the cloud base, the higher the chance of precipitation (as every farmer used to know). As described in [10], the cloud base can be calculated from the air temperature and the dew point.

The electronics

The electronics of the weather station are fairly simple, as long as you regard the RPi and the various sensors as black boxes. The layout shown in **Figure 7** consists of the RPi Zero W, the sensors, and the four-wire bus (two wires for I²C, two for the supply voltage and ground) that links everything together.

The power source for the RPi is not shown; the author used a USB charger/adapter connected to the power connector of the RPi board (at the bottom right on the photo in Figure 1). The resistors Rp shown in Figure 7 are the pull-up resistors for the Sensirion differential pressure sensors.

Figure 8 shows the author's prototype fitted in a small plastic enclosure. The USB power cable for the RPi exits through a strain relief, as does the four-wire UTP cable for the wind gauge. In the prototype this cable is 5 m long, which does not cause any problems for data transmission over the bus, and it is also protected by a strain relief. This allows the wind gauge to be mounted well above the roof, with the rest of the electronics located indoors. The addresses used for the devices on the I²C bus are listed in **Table 2**.

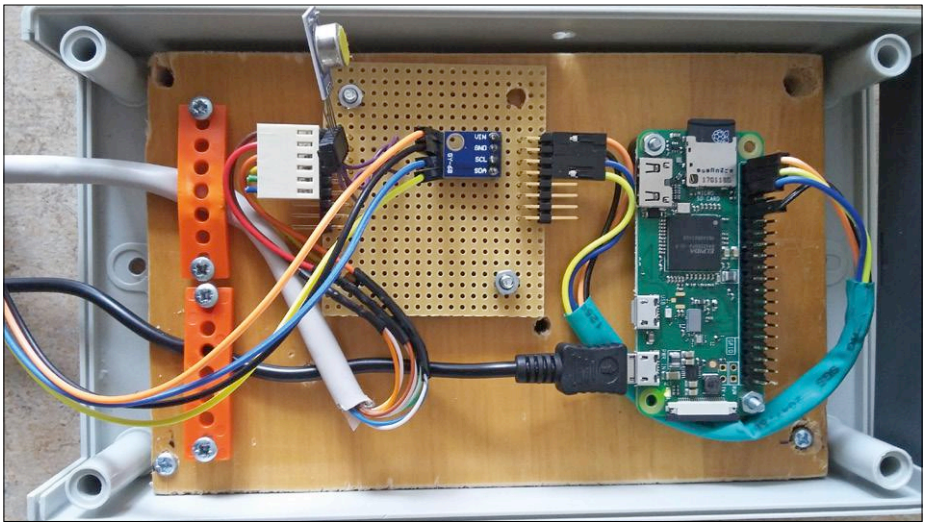


Figure 8. The author's prototype.

Mechanical construction

Building the wind gauge requires a bit of tinkering. For this the author used a piece of PVC pipe with a diameter of 10 cm. He drilled four holes in the pipe at right angles to each other (for the four compass directions) and glued a piece of plastic tubing (of the sort used for aquarium lines, for example) into each of these holes. These tubes lead to ports P1 to P4 of the two differential pressure sensors.

Figure 9 shows the wind gauge in its natural element. To protect the electronic components, the top and bottom of the wind gauge must be properly sealed to prevent water entry. Screening should also be placed in front of the tube openings to prevent insects from using them as hiding places. If you like tinkering, this is your golden opportunity.

Figure 9. The wind gauge in its natural element, well above the roof.

The firmware

The firmware for the weather station consists of two parts. The first part is the *weerstation.py* software. It is written in Python, which is a natural choice because the Python 2.6 programming environment is a standard part of the Jessie Linux distribution for the RPi Zero.

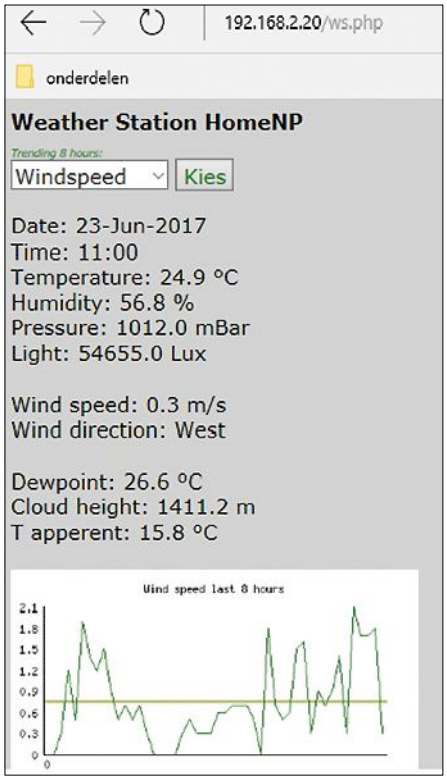


Figure 10. Visualisation of the measured and derived quantities on the web page.

This software reads the sensors every 10 minutes, performs the necessary calculations, and stores the results in a MySQL database.

The second part is a combination of HTML and PHP script. This script runs on the Apache webserver of the RPi Zero. It reads the data from the database and configures it for presentation on a web page (see **Figure 10** for an example). There's no point in showing a complete listing of the firmware here, since it can be downloaded with a couple of mouse clicks. The code is clearly structured and generally self-explanatory. For your interest, the following libraries are used in the *weerstation.py* software:

```
import MySQLdb as mdb
import sys
import time
import os
import smbus
from ctypes import c_short
import math
```

The web page *ws.php* (see the download) is published by the Apache webserver installed on the RPi Zero. The RPi Zero is connected to the author's LAN, so the data from the weather station can be viewed on every device within range of the network. If desired, the web page can also be published on the Internet via the LAN router.

Conclusion

This article is not intended as a detailed DIY project, but instead as a stimulus for interested readers to build their own weather stations. As always, we welcome your comments and suggestions, and we would like to hear from readers who have built weather stations.

(160566-I)

Web Links

[1] www.elektormagazine.com/160451
[2] https://learn.sparkfun.com/tutorials/bmp180-barometric-pressure-sensor-hookup-
[3] www.elektor.com/bme280-mouser-intel-i2c-version-160109-91
[4] www.farnell.com/datasheets/1643979.pdf
[5] http://domoticx.com/esp8266-wifi-lichtintensiteit-sensor-bh1750-gy-302-nodemcu/
[6] www.sensirion.com/en/environmental-sensors/humidity-sensors/digital-differential-pressure-sensors-without-zero-point-drift/
[7] www.elektormagazine.com/160566
[8] https://en.wikipedia.org/wiki/Dew_point
[9] www.bom.gov.au/info/thermal_stress/?cid=003bl08
[10] https://en.wikipedia.org/wiki/Cloud_base
[11] https://en.wikipedia.org/wiki/Pitot_tube
[12] www.elektor.com/raspberry-pi-zero-w-starter-kit

HomeLab Helicopter

Compiled by **Clemens Valens** (Elektor Labs)

In 1997 Opel (General Motors Germany) presented a connected concept car dubbed "Moon", based on its popular model Corsa B. Were these antennas smart or just bits of wire and aluminium foil? (Photo: Wikimedia, Lexicar)

Smart Antenna Protects Car Against Cyber Threats

For years, I managed to keep an old Opel Corsa B 1.2-litre City alive because the most sophisticated electronics parts inside were probably its lightbulbs (not counting the radio, which was optional). Today cars are so crammed with electronics that they get scrapped when one wire comes loose. How things have changed.

Filled to the brim with electronic systems modern cars turn previously non-existent problems into big deals, and we start reading things like "STMicroelectronics is leading the race to protect connected cars against cyber threats..."

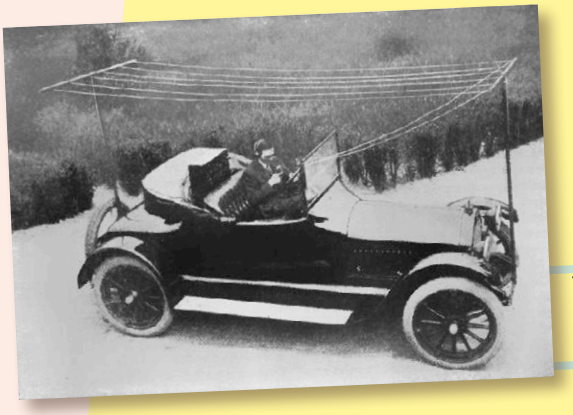
To fight these threats ST has created the STA1385 Telemaco3P telematics and connectivity processor family. Dual ARM Cortex-A7 devices integrating an isolated Hardware Security Module (HSM) subsystem based on an ARM Cortex-M3 to check the authenticity of received messages and external devices that try to connect.

The new chips are also very robust, with a 105°C maximum temperature rating for use in locations that can become extremely hot, such as on top or directly beneath the roof in a smart antenna.

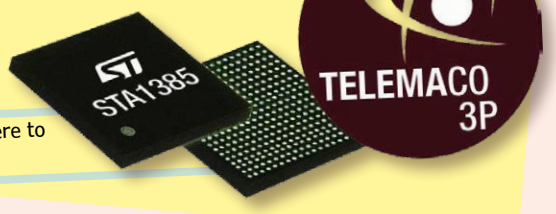
Smart antenna? Oh dear, even a piece of wire is getting "intelligent" and capable of running POSIX-compliant operating systems after adding three (!) 32-bit processors to it. Once a piece of copper wire the antenna now has CAN FD, Gigabit Ethernet, and 100 Mbit/s Secure Digital I/O (SDIO) interfaces to create its own interference.

Progress & technology, we all crave it, but boy, how they complicate matter(s).

Smart with dumb antenna.



This connected car sporting an inverted-L wire antenna didn't look very smart at all, even in 1919. (Photo: Wikimedia, Pacific and Atlantic)



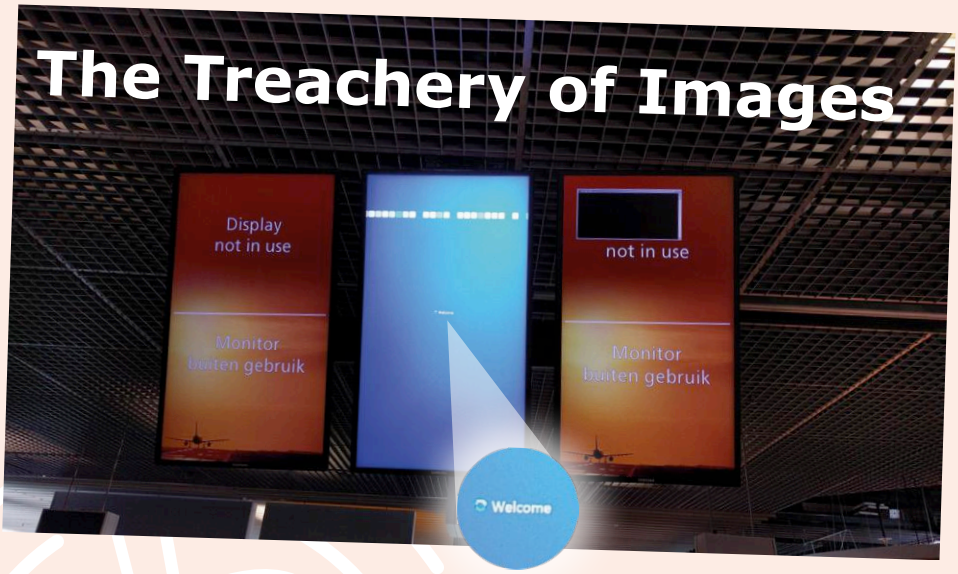
Telemaco3P advanced secure microprocessors by ST are here to protect our connected cars. (Photo: STMicroelectronics)

Must-have Homelab Tool

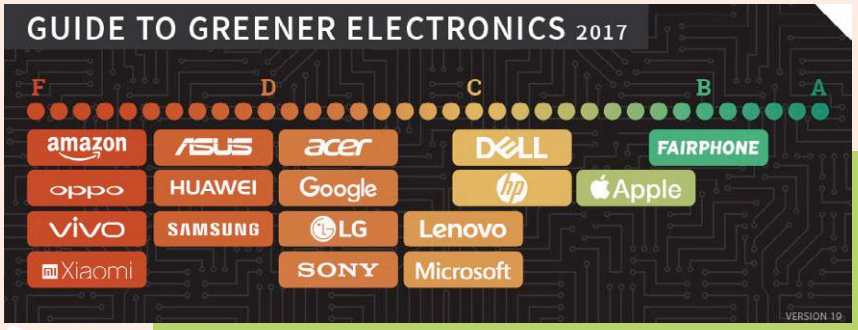
Fixr from True Utility is a bottle opener, nail cleaner, flat screwdriver (large & medium), small flat eyeglass screwdriver, Phillips screwdriver (medium & small), spanner/wrench (6, 8, 10, 12 & 14 mm), bicycle spoke wrench, cutting blade, wire stripper, measuring ruler, box opener, pry bar and file all in one. Finally, you can throw away all your other rusty space-consuming tools, and replace it by this 80 × 25 mm titanium-coated stainless steel pocket tool. Oops, almost forgot, it's also a key hanger.

Volvo's Polestar electric cars will be sold online only, and offered on a two- or three-year subscription basis.

The Treachery of Images

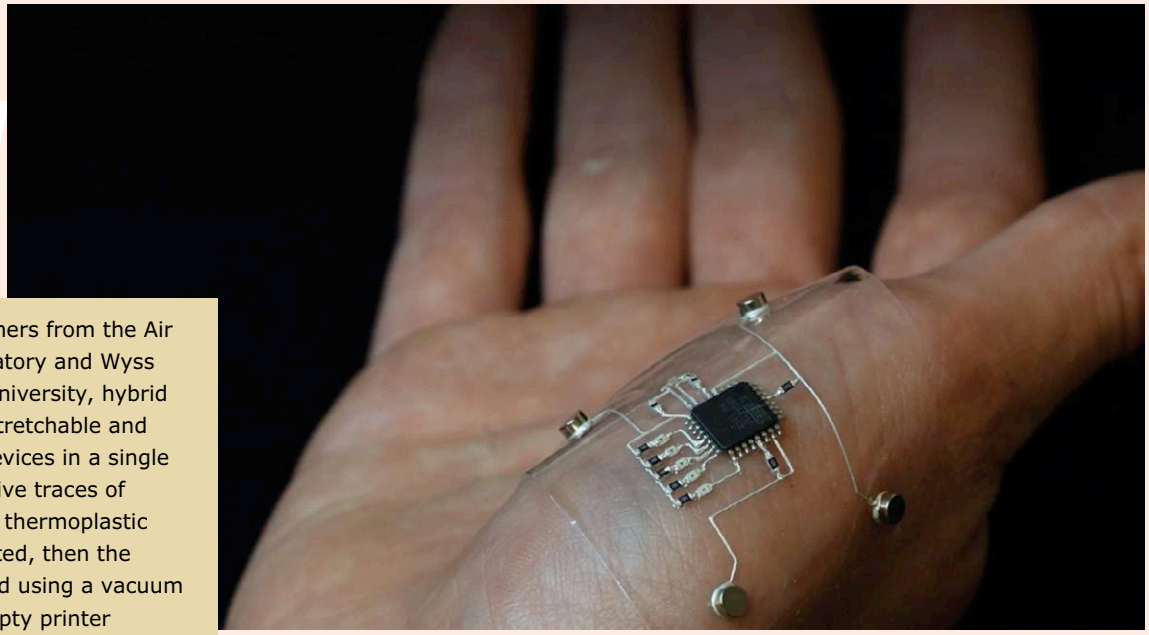


In an attempt to ease stressed travellers desperately looking for information when the displays aren't working, the designers of the flight announcement system at Schiphol Airport in the Netherlands came up with this solution. I say designers, because it is highly unlikely that a single person can be authorized to address millions of people every year. A team of designers and client satisfaction experts has brainstormed for hours about how best to deliver this message; expensive consultants browsed huge photo databases for a suitable background image to show on a non-active display. But which display is not in use? All three or only the middle one? If that is the case, do we need two displays to tell us that a third one is not being used? Wouldn't a few arrows pointing to the out-of-use display make the message clearer? Then again, although it is hard to see, the middle display does say "*Welcome*", so it is in use. This implies that the two other displays are not in use. And yet they do show an informative message. I almost missed my flight because I stared too long at this paradox. Damn useless displays, might as well switch them off.

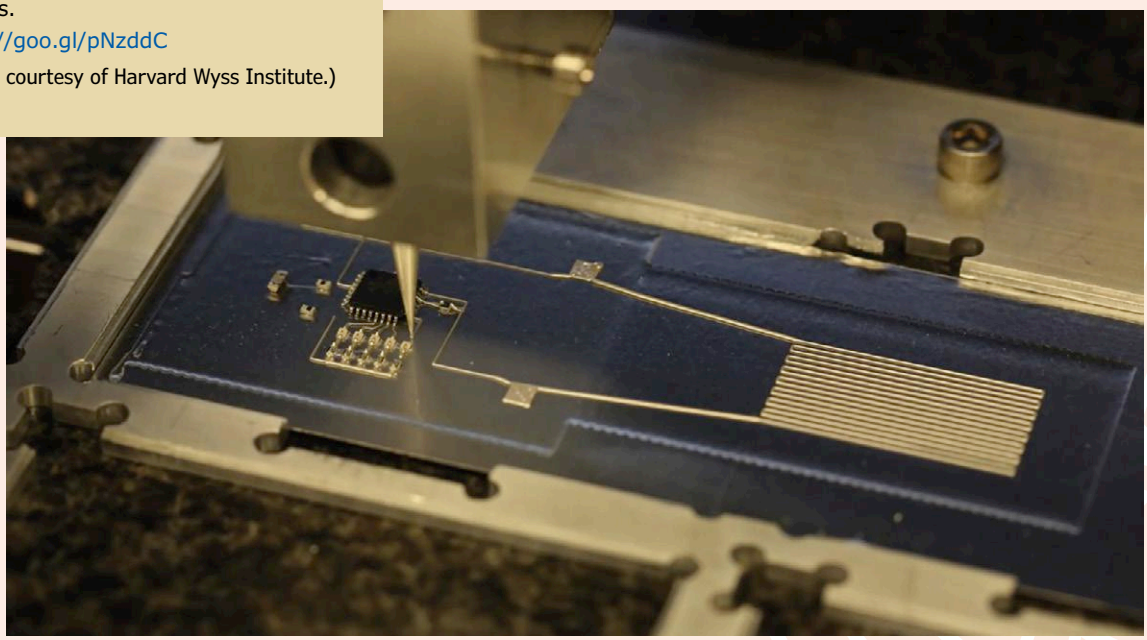


The Greenpeace Guide to Greener Electronics, version 19.

How green is your smartphone, tablet or laptop computer? Green is best, red is worst.
Read the full report at http://greenpeace.org/greenerguide.



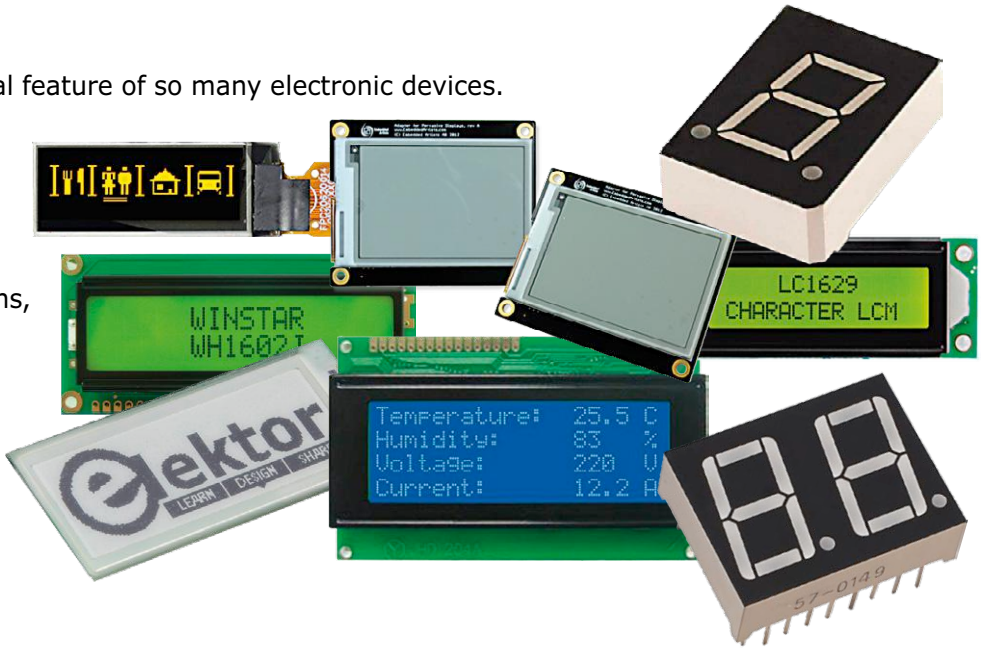
Developed by researchers from the Air Force Research Laboratory and Wyss Institute at Harvard University, hybrid 3-D printing creates stretchable and wearable electronic devices in a single process. First conductive traces of flexible, silver-infused thermoplastic polyurethane are printed, then the components are placed using a vacuum system made with empty printer nozzles.
https://goo.gl/pNzddC
(Photos courtesy of Harvard Wyss Institute.)



Want to participate? Please send your comments, suggestions, tips and tricks to labs@elektor.com

Everything (almost) you need to know about small displays

Some form of display is an essential feature of so many electronic devices. Not so long ago the only available option was to use 7-segment LED displays to let us know what the microcontroller was doing. Now there are many more display options, allowing a far more sophisticated representation of information. The question is: what can they do and how do they differ?



By **Dr. Thomas Scherer**

Q *Which type of display is best?*

A Just like life in general, there is often not one single best solution to any problem; it all depends… The best type of display depends on how it will be used and what it will be required to indicate. The main factors you need to consider are:

- What do you need to show on the display? Text or graphics?
- How big does the display need to be? The possibilities range from <1" up to computer-monitor size.
- How many colors does the display need to show? Will monochrome suffice or will full color be necessary?
- What type of interface can be used? 4/8-bit parallel or better still a serial protocol?
- Is power consumption an important consideration? Battery or mains operation?
- What level of display resolution is required? This will govern the type of display technology used.
- Cost? Price is dependent on the display technology and screen size.

These are the seven most important criteria to consider before you pick the type of display suitable for any specific application.

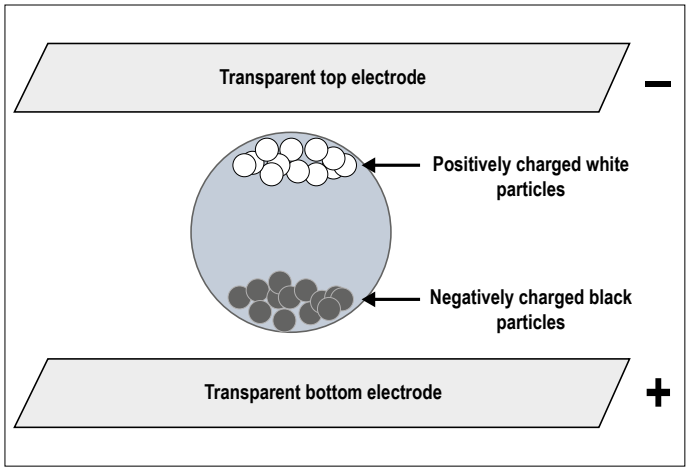


Fig. 1. E-Paper pixel construction.

Q *What are the pros and cons of E-Paper displays?*

A E-paper or E-ink displays first came to public attention when the Amazon Kindle e-reader device was introduced about 10 years ago. This type of 'electronic paper' display uses very little energy and is well-suited to applications where the displayed images are static for long periods such as occurs with e-reader devices. The display area is made up of a layer of tiny fluid-filled capsules (**Figure 1**) containing coloured particles. When a positive or negative electric field is applied to an individual electrode, the coloured particles with the corresponding charge will move either to the top or bottom of a capsule, making the surface of the e-paper display appear a certain colour. Particles now remain in this position and display the static image without requiring any more power, i.e. they have bistable properties.

E-Paper is a passive type of display so it doesn't emit light; the image has lower contrast and poorer resolution than some other types of flat-screen displays and the display has a much slower response time so the display of fast moving images is not possible. Colored E-Paper displays are possible but they are most often monochrome. Their low energy consumption makes them ideal for use in battery-powered equipment.

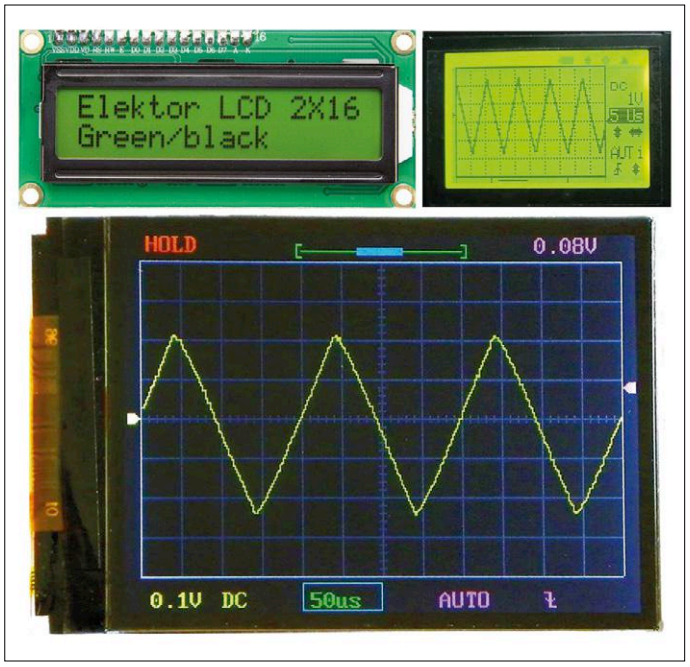


Fig. 2. Alphanumeric as well as monochrome and colour LCDs.

Fig. 3. The 'classic' 0.96-inch OLED display in white.

They come in a wide range of sizes including some smaller than the Kindle display. Special E-paper controller chips are necessary to drive them. For some hands-on experience there are several E-paper display development kits available starting from around $25. For more detailed information see the article in Elektor October 2016 [1] which features this display technology.

Q *What are the pros and cons of LC displays?*

A Small LCDs used in Microcontroller projects have been around for years now. This type of display technology is used in a wide variety of applications (**Figure 2**). Elektor has already published a wide range of informative articles on LCDs [2]. These display types can be driven by a range of different controller chips. The simplest displays consist of either two or four lines displaying 12 to 20 characters per line. They can be bought very cheaply and are generally controlled via 4- or 8-bit parallel bus control signals. They have the disadvantage of using up several precious microcontroller I/O pins, cannot display graphics and have a relatively poor viewing angle and contrast. They are however ubiquitous so practically every microcontroller IDE already contains library routines to control this type of basic display.

There are also of course many graphics-capable monochrome and colour LCDs. These are usually provided with an I²C or SPI serial interface which simplifies interconnection to the microcontroller. There are so many different types of display available it's important to determine before purchase, that the IDE of your preferred microcontroller supports the type of display controller you intend to use in your application, you don't want to waste unnecessary time re-inventing the wheel. The vast majority of touchscreens employ LCDs.
Overall one of the biggest disadvantages of these displays is the relatively high energy requirement for backlighting.

Q *What are the pros and cons of OLED displays?*

A The use of small OLED displays (**Figure 3**) in microcontroller projects has been described in detail elsewhere in this issue of Elektor. We can summarize the properties of these display types as:
Pros: High-resolution, high-contrast, low energy consumption, (almost) only one single controller chip and low costs (for small screen sizes).
Cons: The display properties do not remain stable over time. The cost of a large area display is relatively high and full-color OLED displays are very expensive.

Q *What about some of the other types of display?*

A There is not much to say about individual LED indicators and 7-segment LEDs. Displays based on this technology are easily readable but the 7-segment displays usually require additional hardware and involve a lot of wiring, they are also limited to displaying only numbers. If you are looking for an old-timer theme you can opt for Nixie tubes and add a real steam-punk vibe to your design. These types of displays have been featured in many Elektor projects in the past. We should also not forget to mention the more recent micro-LED (µLED) flat-panel displays. The display consists of arrays of microscopic LEDs and has all the advantages of OLEDs but is brighter, faster, with a longer life expectancy and offers better efficiency. Displays using µLEDs can achieve frame rates in the milliseconds range. Unfortunately this type of display is not yet freely available, we will just need to wait to see what the future holds… ⏮

(160563)

Web Links

[1] Elektor article on E-Paper: www.elektormagazine.com/160196
[2] Elektor article on LCDs: https://goo.gl/qrnxMs

Get Busy with the CAN Bus

In control of 4,000 lights in a multi-storey building

Thanks to the proliferation of Arduino and the Open Source and Hardware movements, setting up a CAN bus for experimentation is now child's play. Arduino-compatible CAN bus interface hardware can be bought online cheaply and supporting software libraries can be downloaded for free, all that remains to do is build the application. In this article we will show you just how easy that has become.

By **Clemens Valens** (Elektor Labs)
Based on an idea by **Bera Somnath** (India)

The CAN bus — officially known as Controller Area Network — allows connected devices to exchange short messages in an efficient way. Although originally designed as a robust vehicle communication bus using as little wiring as possible (twisted pair), today it has found its way into all sorts of other applications.

CAN is not a master-slave, peer-to-peer (P2P) or other connection-based network, but message-based. Consequently, connected devices — the nodes — send messages whenever they feel like it and without caring if there's anybody listening. Collision and error detection and prioritization schemes ensure that everything goes well as long as network traffic conditions remain normal.

Nodes do not have addresses or IDs, the messages are typed instead. Nodes listen to and send only messages of certain types; they cannot talk to each other directly without adding an extra layer to the protocol. However, since it is possible for a node to request a certain type of message, and because the CAN specification requires nodes to use unique message IDs, basic P2P communication is possible (see inset about Remote Frames). Standard CAN supports up to 2,048 message types (11-bit message ID), extended CAN supports more than 500 million (2^{29} to be exact, thanks to its 29-bit

Features

- No soldering required
- Supports over 4,000 nodes
- Easy to understand software
- Based on cheap modules

message ID). Both flavors of CAN have a payload of up to eight bytes.
Because on the CAN bus a '0' is stronger ("dominant") than a '1' ("recessive"), the lower the message ID, the higher its priority; ID=0 beats all. Experience has shown that in order to achieve high bus utilization, message IDs should be attributed according to message importance, not to message content.
The above is about all you have to know for the experiments described in the remainder of this article. For more information on how the bits travel over the wires and similar nerdy details we highly recommend the Internet.

Required hardware

As a basis for our experiments we will use the AVR Playground, but for the first experiments any Arduino Uno compatible board will do. Since the 'N' in CAN stands for Network, we are going to need at least two nodes, and so we will actually employ two AVR Playgrounds. Both nodes need a CAN interface. Several options exist; we chose a Microchip MCP2515-based solution because of its popularity. Since the AVR Playground has a mikroBUS slot, we opted for CAN SPI Clickboards (5-V version, though the 3.3-V version should work equally well) from MikroElektronika, leaving the Arduino shield connectors free for other uses. Almost identical modules, but with a different shape and requiring jumper wires to hook them up can be found online for less than €/$/£ 5.
For our more serious experiments (see below) we also recommend one of those cheap 4-channel relay shields that are so abundant on the Internet.
Once the CAN modules are connected to the Arduino Uno compatible boards, it's time to wire the network. Officially this has to be done with a twisted-pair cable but for short distances simple wires will work too. Twisted pairs can be found in Ethernet cables for instance (**Figure 1**). The nodes sitting at the beginning and end of the cable must terminate the cable; the CAN modules have a jumper for this. Intermediate nodes must *not* terminate the cable.

Required software

First of all an naturally you will need your Arduino IDE.
For easy access to the AVR Playground's peripherals you should install its Boards Package in the Arduino IDE. Refer to [2] for the details. Once installed, select the AVR Playground as the Arduino board and choose the COM port that goes with it.
You should also install the CAN bus library for Arduino — mcp_can — kindly brought to us by SeeedStudio [3].
That's it, here we go!

First steps

We will start simply by verifying that the CAN interface can be initialized correctly. The instructions refer to the Arduino IDE's main menu.

1. If needed, open a new sketch:
   File ➞ New
2. Include the CAN library:
   Sketch ➞Include Library ➞ mcp_can
3. Add the following below the `include` statements but above the function `setup`:

```
const int SPI_CS_PIN = 10;
MCP_CAN CAN(SPI_CS_PIN);
```

   This supposes that the CS pin of your CAN interface is connected to Arduino pin 10.
4. Add the following to the function `setup` (between the {}):

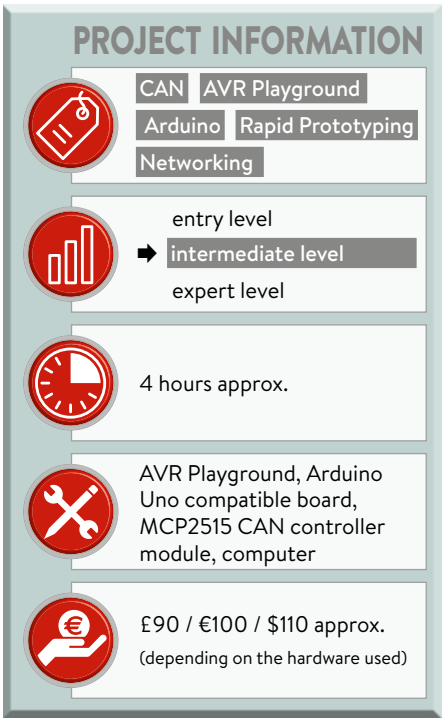
```
Serial.begin(115200);
while (CAN_OK!=CAN.
begin(CAN_200KBPS)) // 125 Kbps
with 10 MHz crystal
{
  Serial.print("*");
  delay(100);
}
Serial.println();
Serial.print("CAN init OK.");
```

5. Compile the sketch and load it into the AVR Playground (you will be asked to save it first):
   Sketch ➞ Upload
6. Open the Serial Monitor:
   Tools ➞ Serial Monitor
7. Set it to a baud rate of 115200 (bottom right corner).

If the text "CAN init OK." did not appear, verify the connections of your CAN interface and check your code carefully.
If you can connect both nodes to your PC at the same time, repeat the above steps in a second instance of the Arduino IDE connected to the second node. If you can connect only one node at a time, only repeat steps 5 and 6. Both nodes should initialize successfully (duh).

Sending and receiving messages

For the following experiment we will need at least two working nodes connected to the CAN bus. Do not forget to terminate the bus.
Listing 1 shows the program which is



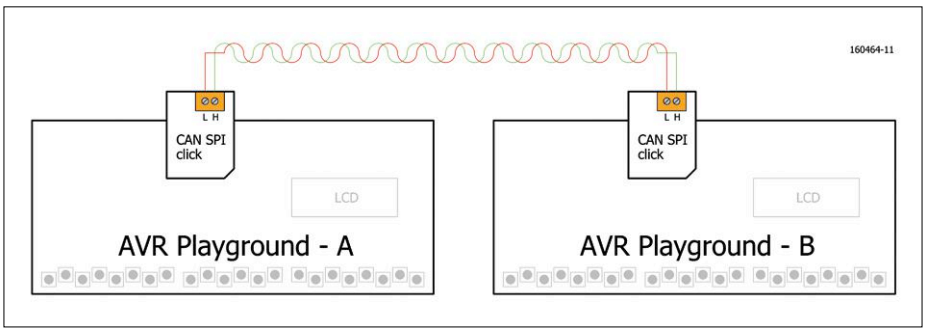


Figure 1. A basic CAN for our first experiments.

Listing 1. Send and receive short text messages. Make sure to use different values for my_message_id and my_message on each node.

```
/*
 * Elektor article 160464 Get Busy with the CAN Bus
 * Sending and receiving messages
 */

#include <mcp_can.h>
#include <mcp_can_dfs.h>

const int SPI_CS_PIN = 10;
MCP_CAN CAN(SPI_CS_PIN);
const int my_message_id = 100;
const char *my_message = "Node 1";  // 8 chars max!
uint8_t can_buffer[MAX_CHAR_IN_MESSAGE];

void setup(void)
{
  Serial.begin(115200);
  while (CAN_OK!=CAN.begin(CAN_200KBPS))
                                    // 125 Kbps with 10 MHz crystal
  {
    Serial.print("*");
    delay(100);
  }
  Serial.println();
  Serial.print("CAN init OK.");
}

void loop(void)
{
  if (CAN_MSGAVAIL==CAN.checkReceive())
  {
    uint8_t len;
    CAN.readMsgBuf(&len,can_buffer);
    uint8_t message_id = CAN.getCanId();
    Serial.println("---");
    Serial.print("Message ID: ");
    Serial.print(message_id);
    Serial.print(", received ");
    Serial.print((int)len);
    Serial.println(" bytes.");
    for (int i=0; i<len; i++)
    {
      Serial.print((char)can_buffer[i]);
    }
    Serial.println();
  }

  CAN.sendMsgBuf(my_message_id,0,strlen(my_message),(uint8_t*)
   my_message);
  delay(1000);
}
```

the same for every node, except for the values of `my_message_id` and `my_message` that should be unique for each node.
The function `setup` is the same as in the previous example, but `loop` has been substituted. It begins by checking if a message has been received. If this is the case, it is read and its contents are printed to the serial port, together with the message ID. Keep in mind that the message must be read before you can check its ID with `getCanId`. The function `readMsgBufID` can be used to read the message and its ID in one go. Message received or not, the function ends by sending the node's own message, then it waits one second.
Only the lines that contain '`CAN.`' (note that dot) are doing something with the CAN interface, all the other lines are for support and data (re)presentation.

Message filtering

In the previous example the function `loop` of every node received every message that cruised the CAN bus. This is fine for debugging purposes or when the host doesn't have anything else to do. However when it is supposed to handle other tasks as well, having to inspect every message to see if it contains useful information is not efficient. This situation can be greatly improved by exploiting the MCP2515's filter capabilities.

The MCP2515 is not a simple CAN bus interface, but a CAN controller with many possibilities. As such it features filters and masks that can be applied to incoming messages. Only the messages that pass through these filters will be made available to the host. The masks and filters are applied to the message ID, and since extended CAN uses 29-bit IDs the masks and filters are 29-bit long. However, standard CAN works with 11-bit IDs, which would leave the majority (18) of the filter & mask bits unused. For this reason, when using standard CAN, 16 of the remaining filter & mask bits are applied to the first two bytes of the payload.
When a bit in the mask is 0, the corresponding bit in the message ID is always accepted, irrespective of the corresponding filter bit. When a mask bit is set to 1, the corresponding message ID bit will only be accepted if it has the same value as the corresponding filter bit (**Figure 2**). On power-up the masks are cleared and thus every message will be accepted.

The MCP2515 uses three buffers for receiving messages. The first, the Message Assembly Buffer or MAB receives every message. When a message is complete, it is transferred from the MAB to either the RXB0 or RXB1 receive buffer. RXB0 has one mask and two filters (1 & 2); RXB1 has one mask and four filters (3-6). The RXB0 mask and filters are applied first, giving this buffer a higher priority than RXB1. The idea is that high-priority messages pass through RXB0 while less important messages transit through RXB1.

The mcp_can library that we use does not let the user specify which receive buffer to read, it simply checks RXB0 first and then RXB1. This implies that both masks must be set to avoid receiving every message anyway. Furthermore, the library does not support data byte filtering in standard CAN mode.

To activate message filtering, add the following lines to the end of the function `setup` of the previous example:

```
// Set masks & filters in standard
   mode (0).
CAN.init_Mask(0,0,0x3ff); // RXB0
CAN.init_Mask(1,0,0x3ff); // RXB1
// Only accept message IDs 100 &
   103
CAN.init_Filt(0,0,100); // RXB0
CAN.init_Filt(1,0,100); // RXB0
CAN.init_Filt(2,0,103); // RXB1
CAN.init_Filt(3,0,103); // RXB1
CAN.init_Filt(4,0,103); // RXB1
CAN.init_Filt(5,0,103); // RXB1
```

These lines set the masks so that only the filters determine which message IDs get through. Here we directed messages with ID 100 to RXB0 and messages with ID 103 to RXB1 (although you won't notice this). Note that filtering stops as soon as a match is found, so only filter 0 and filter 2 will produce matches. To see filtering in action you must, of course, program a node to send these message IDs and some more. You can download a sketch from [1] that allows you to do this. The sketch sends six messages with six different IDs and listens to the messages you specified for the filters. It can be used on every node on the bus as long as you make sure that each node sends messages with unique IDs. The variable `tx_id_base` is available to achieve this. A node's `tx_id_base` must be chosen so that the six message IDs it transmits (`tx_id_base` to `tx_id_base+5`) are unique on the bus.

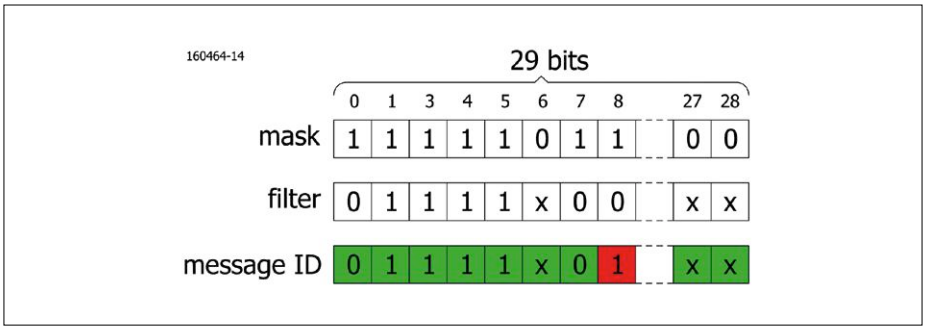


Figure 2. To limit the number of messages a host has to read, the MCP2515 first applies a mask and then up to six filters. If the mask bit is 0, then the filter and message bit don't matter; if the mask bit is 1 then the filter and message bit must be identical. This message ID is rejected because of bit 8.

Lighting control

Having come this far, we know almost everything we need to know to create an interesting application, all we have to do is add meanings to the messages and implement functions to do useful things like switching a relay or reading a pushbutton.

What about setting up a CAN in a building to control the lights in each room and on every floor? All the light switches in the building will become CAN nodes and at some strategic positions controller nodes can be placed to switch on or off these lights. The concept can, of course, be extended to heaters and air conditioners and what not.

We will simulate the system with our AVR Playgrounds, one of them equipped with an Arduino-compatible 4-channel relay shield, bought online for about €/£/$10, to switch the lamps. The AVR Playgrounds provide LCDs, buzzers, pushbuttons & LEDs, and trimmers that can be used to control the relays and provide feedback on what is going on (**Figure 3**).

Message ID mapping

Before we start programing, we must first think a little about network planning and message ID mapping. Assuming a building with ten floors, with ten rooms per floor, and with ten nodes (or sockets) per room, 1,000 IDs would be needed to address each node individually. Standard CAN has room for 2,048 IDs, giving us plenty of headroom. Unique message IDs are necessary for nodes that are allowed to send messages (like status information). In case this is not needed, a single message ID would be enough to switch on or off every light individually simply by storing its address and state in the 8-byte message payload.

Suppose our system requires that every node must be capable of sending (status) messages. Mapping 1,000 11-bit IDs to ten lights in ten rooms on ten floors would need a lookup table as it is not easy to come up with an intuitive ID assignment besides simply numbering them from 1 to 1,000. Such a scheme would also introduce important priority differences in a system where all nodes are supposed to be equal.

For the ease of installation it would be best if each node could come up all by

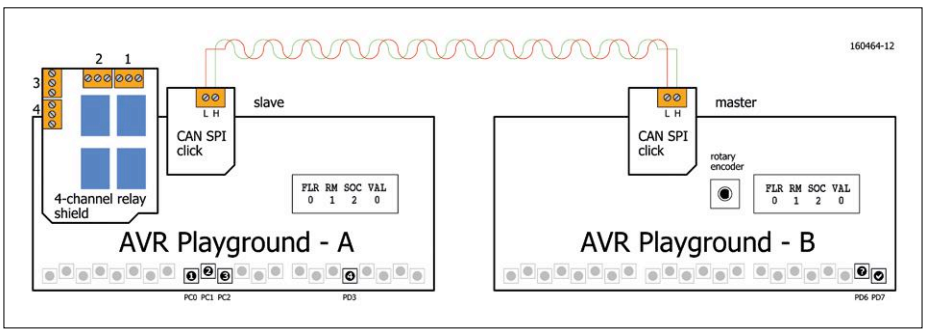


Figure 3. With this system we can simulate a lighting control system for a multi-storey building.

On bitrates and quartz crystal frequencies

The mcp_can library expects that the MCP2515 module is clocked by a 16 MHz quartz crystal. However, the modules found online can have other crystals mounted like 8 MHz; the CAN SPI click boards we used run from 10 MHz. Mixing these boards on one bus without adjusting the bitrate setting for each module in the firmware will result in incompatible bitrates on the network. Unfortunately, the mcp_can library does not expose a function to set the bitrate, only a few constants are available that are processed further internally. But there is good news too: without modifying the library, three pre-defined bitrates are available that happen to work with 8, 10 and 16 MHz MCP2515 CAN modules (see the table below). Use these constants in the call of the function begin() of the library (see for instance Listing 1). If you really need another bit rate, then you need to dive into the library and adapt it for your needs.

	8 MHz	10 MHz	16 MHz
25 Kbps	CAN_50KBPS	CAN_40KBPS	CAN_25KBPS
50 Kbps	CAN_100KBPS	CAN_80KBPS	CAN_50KBPS
125 Kbps	CAN_250KBPS	CAN_200KBPS	CAN_125KBPS

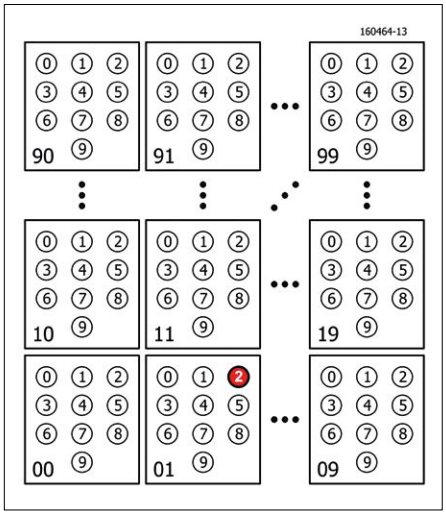


Figure 4. The payload <2><1><0><1> of the message with ID 0xffff will switch the highlighted node ON.

itself with a unique ID based on its location. The installer would only have to set the floor, room and socket numbers and the node would do the rest. Extended CAN with its 29-bit IDs can handle this quite easily: simply reserve four bits for floor, room and socket fields and add a large offset. This would map them all to a small (12-bit) address space with similar priorities in the huge pool of possible IDs and it would be easy for a controller node to figure out where a node lives. So let's use extended CAN with node addresses formatted as <s3s2s1s0><r3r2r1r0><f3f2f1f0> and add an offset of say 0x10000. Node (socket) 0 in room 0 at floor 0 would have address 0x10000 and node (socket) 9 in room 9 on floor 9 would have address 0x10999. (Where would node 0x10123 be? Exactly: socket 1 in room 2 on floor 3.) This leaves room for some 3,000 extra nodes… To use the mcp_can library in extended CAN mode, all that must be done is setting the 'ext' argument (of the functions that expose it) to 1 (instead of 0).

One additional ID (arbitrarily chosen as 0xffff) would be required for controlling the nodes. A switching command would then specify the floor, room, and socket in its payload, together with the switching action like ON or OFF, or, since RGB LEDs rule the world these days, an RGB value (or a fan speed, heater level, etc.). To support all these options we could code the action in three bytes and keep two bytes for future extensions. However, for the sake of simplicity we code the action as one byte with OFF=0 and ON=1 (**Figure 4**).

Of course, it is nice to be able to remotely switch a light on or off, but the real power lies in the possibility to control groups of lights at once. For this we will use the value 255 (0xff). When floor 255 is specified, all the nodes with the specified room and socket numbers on all floors will be concerned. Similarly, when room 255 is specified, all the nodes with the specified socket number on the specified floor will respond. When floor, room and socket are all set to 255, then every node in the building will listen and they can all be switched on or off together. **Table 1** shows a few examples to clarify the concept.

Table 1.
Groups can be controlled by using the group or "all" address, 255.

Floor	Room	Socket	Action
255	2	9	Socket 9 in room 2 on every floor will respond
5	255	2	Socket 2 in every room on floor 5 will respond
0	3	255	Every socket in room 3 on floor 0 will respond
255	6	255	Every socket in room 6 on every floor will respond
1	255	255	Every socket in every room on floor 1 will respond
255	255	255	Every socket in every room on every floor will respond

Table 2. DIP switch settings for the AVR Playground acting as the CAN master ('H' for High, 'L' for Low and 'M' for Middle).

DIP switch	Position (left-to-right)
S24	MMMMMMMM
S33	HLLHHHMMMM
S15	HLLLHHH
S25	LHHHLLH
S27	MMMHHHHH

Table 3. DIP switch settings for the (multi-node) CAN slave AVR Playground ('H' for High, 'L' for Low and 'M' for Middle).

DIP switch	Position (left-to-right)
S24	MMMMMMMM
S33	HLLHHHMMMM
S15	HLLLHHH
S25	LHHHLLH
S27	MMMHMMMM

Putting it in software

We have implemented the above description in three Arduino sketches. The first sketch, the master, to be run on the AVR Playground (see **Table 2** for its DIP switch settings), lets the user

select a floor, room and socket by spinning the rotary encoder (change value) and pushing it (advance to next parameter, note the cursor '>'), and toggle the node's state by pressing pushbutton PD7. A node's state can be toggled because its current state is requested first. This is done automatically as soon as the encoder has stopped moving for a while. If a node does not respond to this request within one second, because it does not exist for instance, the state is shown as '---' and a low-frequency sound is heard (if the buzzer is activated). Pushing down button PD6 will also send a state request. Pressing PD7 will not do anything for "dead" nodes. When a node does respond, its state is displayed and a high-pitched beep is produced. If a group address is selected for one or more parameters (indicated by 'All'), the state must be set with the rotary encoder and PD7 must be pressed to send the command.

Multi-message slave

The master sketch is complemented by a multi-node slave sketch, also intended for the AVR Playground (consult **Table 3** for its DIP switch settings) because it uses the LCD to show status information. The slave is equipped with a 4-channel relay shield where each relay corresponds to a node with its own floor, room and socket numbers (to be defined at the top of the sketch in the nodes data structure). These values are used to create the message ID for a node.

Every node also has its own beep frequency, allowing the multi-node slave to be used without the relay board.

Pushbuttons PC0, PC1, PC2 and PD3 (not PC3) toggle the nodes and their state (the most recently updated node) is shown on the LCD. Every time a node changes state, either locally through a button press or remotely by the master, a state-update message is transmitted on the CAN bus. Not only does this provide feedback to a master, it also allows scanning of the network in parts or all of it in one fell swoop simply by using the 'all' value for one or more of the floor, room and socket parameters. Since all the live nodes in the network will respond to this message with a status update, the master can now easily compile a list of them. Please note that we left this very last step as an exercise to the reader.

Simple slave

A second slave sketch was written too, it's much simpler and supports just one node (or message ID). This sketch features configuration constants at the top, making it suitable for use with an Arduino Uno (or compatible board) and other MCP2515 modules that may have different quartz crystals (i.e. frequencies) mounted.

Conclusion

In this article we have shown how to set up a CAN network without too much effort thanks to open source software and hardware and widely available MCP2515-compatible CAN controller modules. Because of this approach we could focus on the application, a system to control the lights of a multi-storey building.

The code presented in this article can be downloaded free of charge from [1]. ⏮

(160464)

COMPONENT LIST

Miscellaneous

2 pcs. AVR Playground
2 pcs. CAN SPI click
1 pc. 4-channel relay shield
Twisted-pair cable

On remote frames

The master sketch sends state request messages directly to a node, using the node's (message) ID, meaning that there are two nodes using the same message ID on the bus, the master and the node itself, a situation that the CAN specification tries to avoid. That's possible thanks to the Remote Frame specified by the CAN standard. In a remote frame the RTR bit is 'recessive' while in a normal data frame this bit is 'dominant'. Consequently if a data frame and a remote frame with the same message ID are transmitted at exactly the same time, the data frame would win arbitration because it has a higher priority, making the requester receive what it asked for, and everything is fine & dandy.

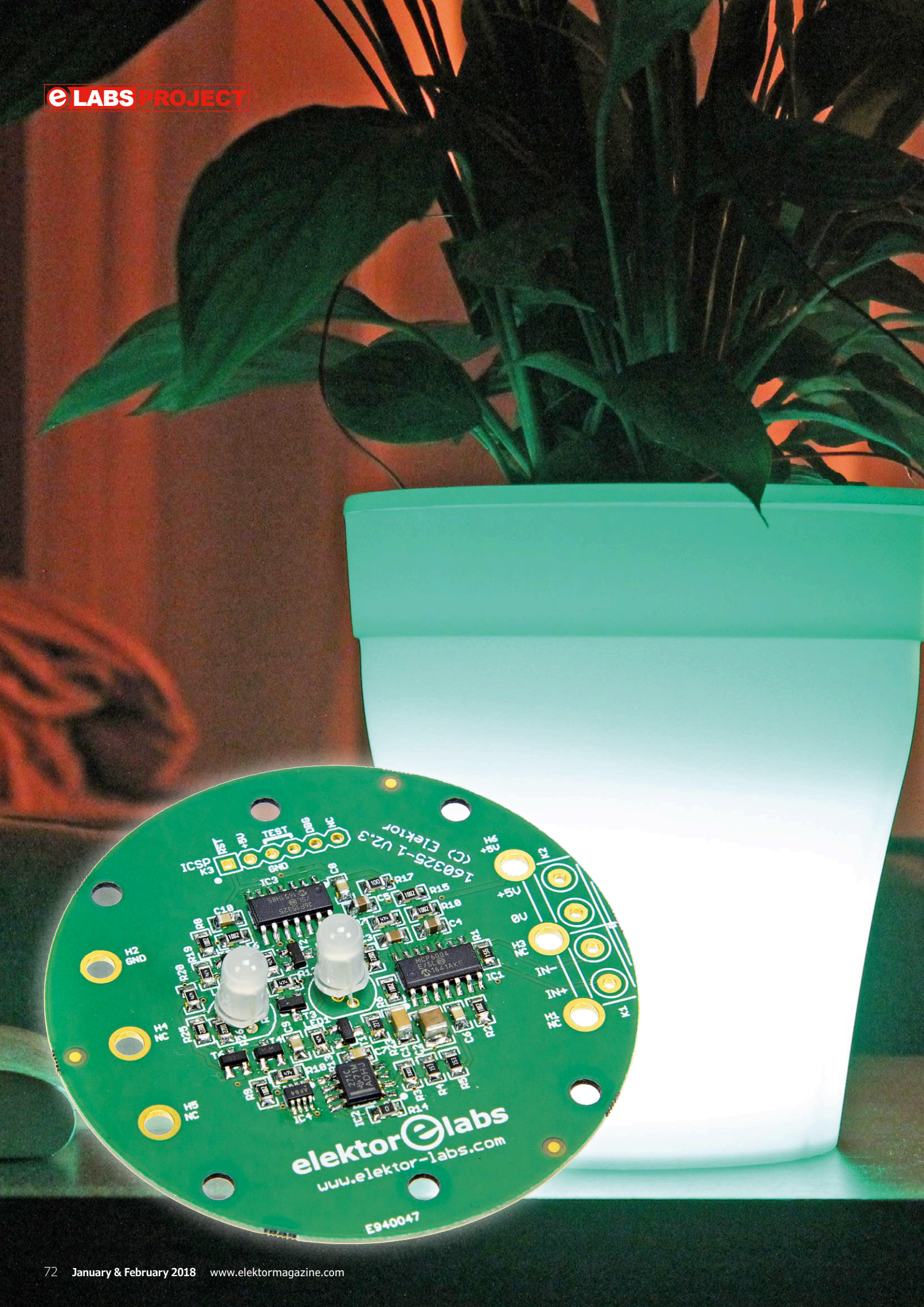
The main reason for the existence of remote frames is preventing nodes from flooding the network with messages nobody is interested in. This looks in contradiction with what we explained at the beginning of this article, but it helps reducing network traffic as they allow nodes to request only the data they really want.

Web Links

[1] This article: www.elektormagazine.com/160464
[2] AVR Playground: www.elektormagazine.com/labs/avr-playground-129009-2
[3] CAN library: https://github.com/Seeed-Studio/CAN_BUS_Shield

FROM THE STORE

→ 129009-2
AVR Playground
CAN SPI click (5 V)


ICSP
K3
RST
+5V
TEST
DBG
NC
GND
160325-1 V2.3
(C) Elektor
H6
+5V
X2
+5V
0V
H3
NC
IN-
IN+
H4
NC
X1
H2
GND
H1
NC
H5
NC
IC3
IC1
IC2
IC4
MCP6004
E/SL@
1641AKE
elektor labs
www.elektor-labs.com
E940047

Bio-Light

Plant expresses 'feelings' through colours

By **Walter Polleros** (Austria) and **Clemens Valens** (Elektor Labs)

We have always been in close contact with plants, trees, flowers, shrubberies, grasses, weeds and other vegetation, yet we have never figured out how to communicate with them. They are alive, we know that, but what do they think or how do they feel? Clueless, we created Bio-Light to visualize our green friends' emotions.

Features

- Translates biosignals into coloured light
- Discreet & stylish design
- Battery powered
- Works great with common indoor plants

Let's start this article by making it very clear that we do not claim omniscience about the secret emotional life of plants. In fact we don't even know if they have one. All we know is that plants produce electrical signals and — being electronics engineers — we know everything about electrical signals and what to do with them. Although we have no idea how to interpret the electrical signals produced by plants, we should be able to make them visible with full-colour LEDs. That is what Bio-Light is, a device that visualizes electrical signals (biosignals) generated (we think) by plants.

Although it has been known for many years that electrical biosignals exist in plants (from now on we will drop 'electrical'), our knowledge about their meanings and functions is minimal to be fair. Humans too produce biosignals; the electroencephalogram (EEG) and electrocardiogram (ECG) are two well-known examples. In this article a biosignal is an evolving potential difference between two parts of a plant.

To measure such a signal it is enough to stick two electrodes somewhere in or on the plant and amplify the potential difference between them. A microcontroller inside the Bio-Light digitizes the signal and maps it to a spectrum of colours that can be produced using an RGB LED.

Two signals

During experimentation it was found that the plant's signal is actually composed of two signals: a slowly evolving carrier signal onto which a small, faster signal is superimposed (**Figure 1**). Slowly here means several seconds to minutes or even longer — the really fast signals are in the range of 2 to 5 hertz. The Bio-Light visualizes the fast(er) signals, which are considered more interesting. However, we must admit that we are not sure if this signal is really produced by the plant or if it's caused by the environment, as it is sometimes possible to obtain the same effects without the electrodes connected to the plant.

Offset compensation

The signal of interest is very small, of the order of a few millivolts, and needs amplification. Unfortunately, doing this naively will also amplify the carrier that's much stronger (tens to hundreds of millivolts) with clipping as a result. In order to get useful results, the carrier must be removed before amplification, i.e. a common-mode rejection circuit is required. The Bio-Light solves this problem in an elegant way. Instead of adding or subtracting an offset voltage from the input, the Bio-Light treats the carrier itself as an offset voltage. Because the carrier's signal range is similar to an operational amplifier's offset compensation range, the carrier can be trimmed away by an opamp's external offset voltage nulling feature. This, of course, requires that the opamp used has such a feature, which is happily the case for many single-opamp ICs.

Offset nulling usually requires a trimpot, hence a digital potentiometer was called in.

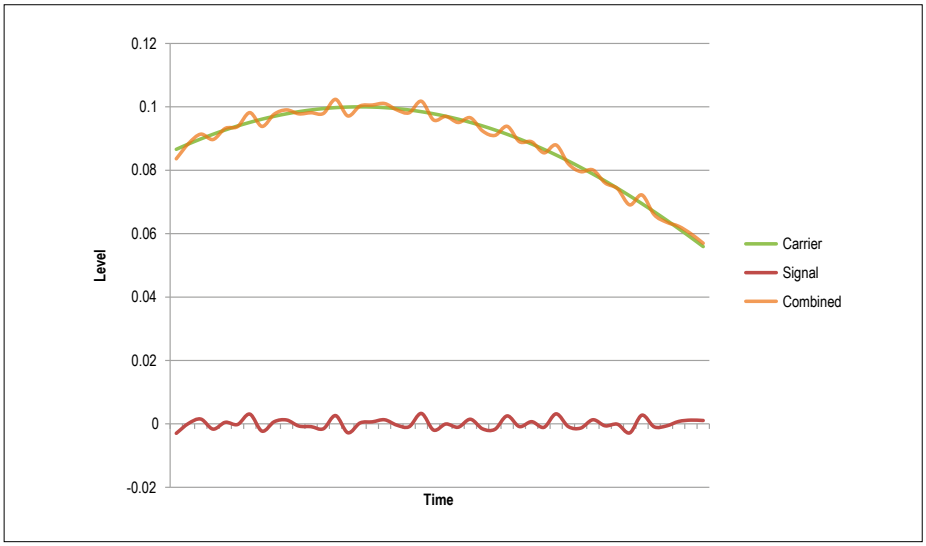


Figure 1. The plant's signal we are interested in surfs on a slowly evolving carrier wave.

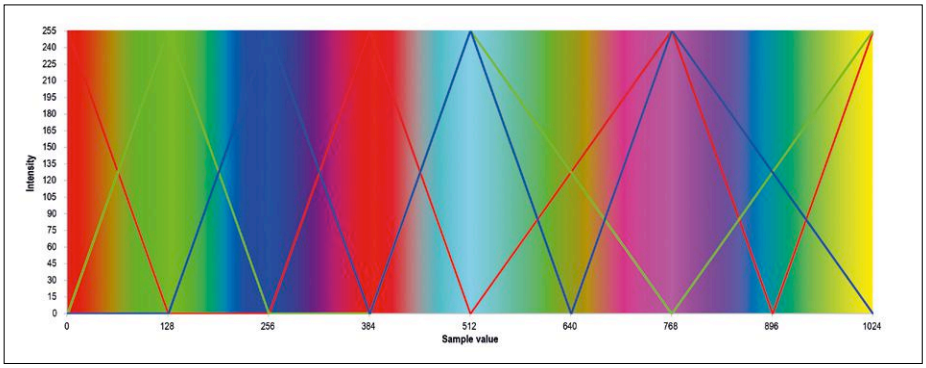


Figure 2. This colour map shows how a sample value is mapped to a colour. When the carrier has been eliminated completely, the plant's colour evolves around cyan.

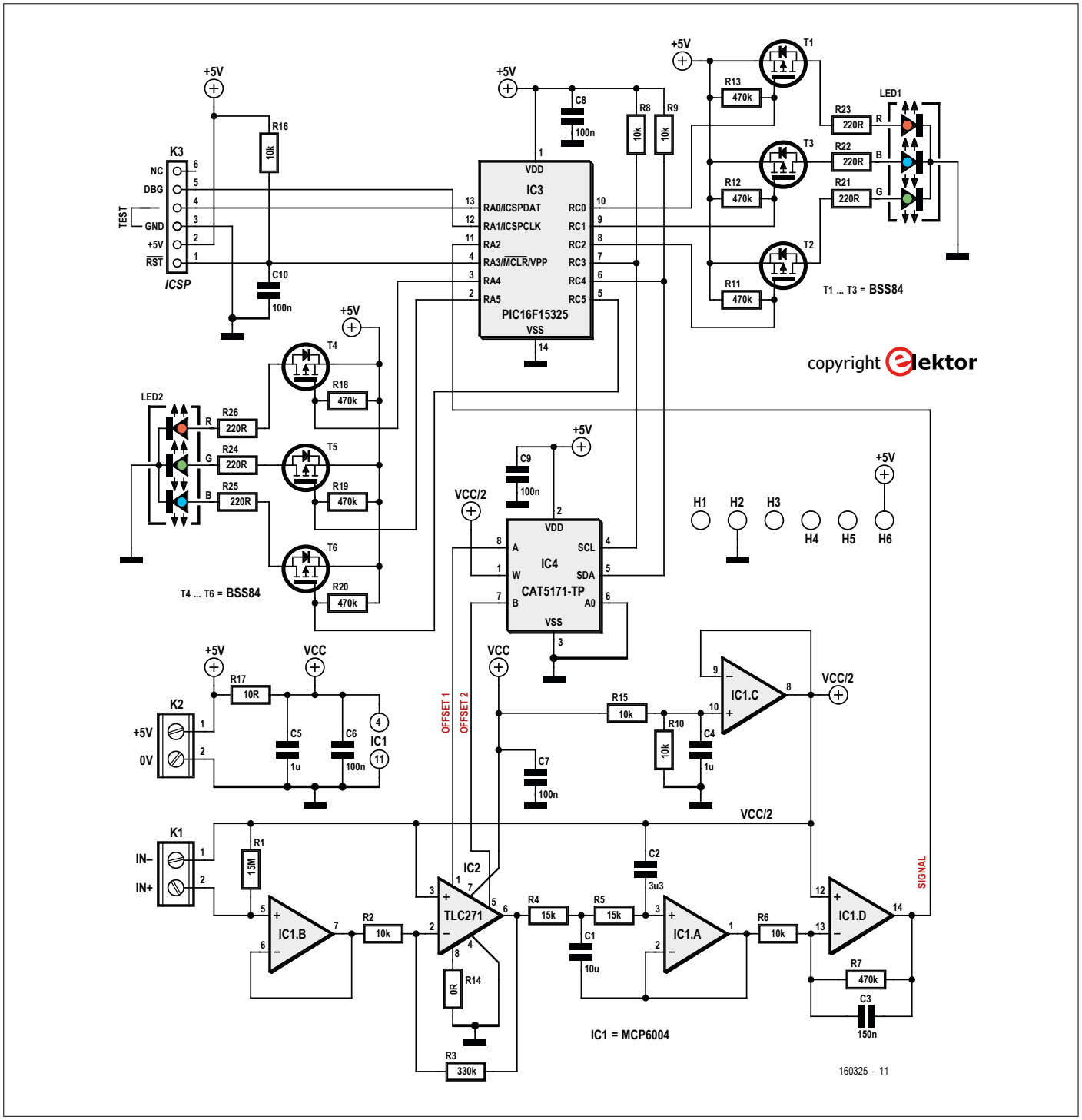


Figure 3. After the signal has been amplified and filtered, the microcontroller (IC3) converts it to the digital domain where it is mapped to a colour. Depending on the values of the samples it may then decide to adjust digital potentiometer IC4 to try to bring the signal back to the centre.

The software

After removing the carrier, the signal of interest can be amplified and filtered before being digitized by a microcontroller. The program running inside the MCU samples the signal at about 27 Hz, a rate chosen to avoid as much as possible any AC line noise (50 or 60 Hz) that may have made it through the input filters. The sampled value is then mapped to a colour in an optically pleasing way (**Figure 2**). When the signal value starts to get close to the minimum or maximum level, the offset compensation is adjusted to move the signal in the other direction. To avoid oscillations due to the filter's latency, the offset compensation adjustment rate has to be slower than the low-pass filter's cut-off frequency. If the offset is adjusted too fast, it may go too far as the filter needs time to catch up. Once it has gone too far, it will be adjusted in the other direction where again it may overshoot before the filter catches up. The result is an oscillating offset compensation, a highly undesirable effect.

The circuit

From the above high-level description to

The Evolution of Bio-Light

The Bio-Light concept was created by Walter Polleros and presented during the **Fast Forward Award** organized by Elektor at the electronica 2016 tradeshow in Munich. The prototype consisted of a box with a translucent ball on it, lighted from below by a powerful RGB LED. It was connected to a plant that influenced the colour of the ball. It was very interesting to see how the plant seemed to react to people passing by and how sometimes it was almost static, as if the plant was resting.

After the show Walter published his project on Elektor Labs and authorized us to develop our own Bio-Light while he continued working on his own design.

The original design used a high-power LED requiring a 24-volt power supply. We wanted our Bio-Light to be powered from a 5-volt phone charger or three AA(A) batteries, so we added a step-up converter. Then we discovered the existence of translucent flowerpots with an RGB LED inside to make them change colour. This discovery made us decide to drop the power LED and develop a board that would fit inside such a flowerpot.

Where Walter's design used a symmetrical ±12 V for the signal conditioning circuit plus 5 V for the microcontroller, we only wanted 5 V for everything. This meant adapting the analogue part of the design to single-supply operation and so Walter's TL061 & TL082 opamps were replaced by rail-to-rail MCP6004 devices. The TL061 originally used for carrier suppression was replaced by a TLC271.

The offset compensation in the original design was adjusted by an AD5220 digital potentiometer. Unfortunately, it turned out that this device was rather hard to buy and expensive, and so we replaced it by a CAT5171. Albeit less than half the price and much smaller, it does have an I²C interface instead of an increment/decrement counter, requiring a different software driver.

The original design was based on an old 28-pin PIC16F882 microcontroller. Because we felt that this MCU was too big for the job, we consulted Microchip and asked them to suggest a more suitable device. They recommended the PIC16F15325, a modern device in a 14-pin package from the enhanced mid-range family.

Replacing the microcontroller also implied porting of the firmware. Walter's firmware was written in assembly but today we prefer C and so the firmware was completely rewritten in C. An I²C driver was added and the PWM signals controlling the RGB LED were generated in hardware instead of in software.

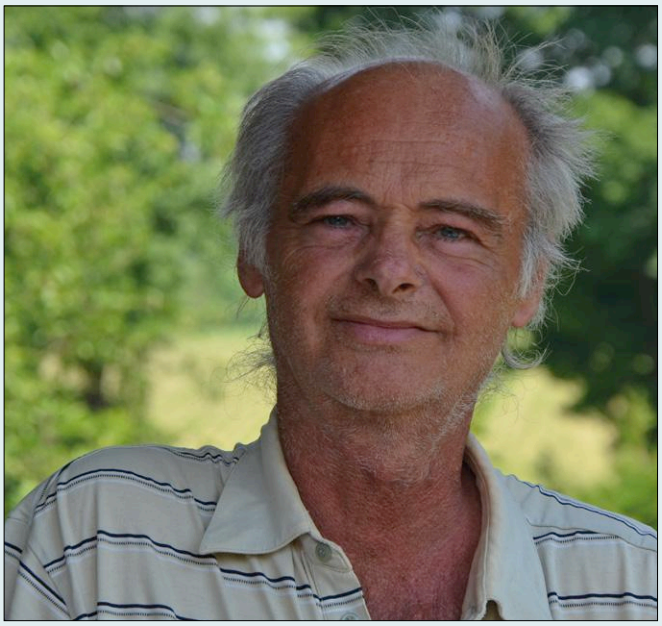
Although our new MCU only has 14 pins, it still has three pins too many (not counting the two pins dedicated to the ICSP connector), and so we added a second RGB LED. First we let this LED show the level of the carrier signal, but in the end we decided to make it do the same as the main LED to increase light intensity. In the firmware it has its own PWM driver (we squeezed six independent hardware PWM signals out of this controller) and it is available for other things for those who want to experiment.

In the end, even though our Bio-Light is based on the exact same principle as the original, almost every component has been replaced.

At the same time as we terminated our design, we received the news from Walter that he had started a crowdfunding campaign for his project. Unfortunately, when you read these lines that campaign will be over, but have a look anyway.

http://igg.me/at/Bio-Light

Bio-Light Deluxe from the crowdfunding campaign consists of a translucent ball on a walnut pedestal.



Walter Polleros, the creator of the Bio-Light

the circuit depicted in **Figure 3** is only a small step.

The input signal, i.e. the plant, is connected to K1 where it is buffered by IC1.B. It is then amplified by IC2 which also provides the offset compensation. R14 sets the opamp's bias current. When mounted as shown, the bias current is high. This provides the largest offset compensation range but also increases its power consumption. Those of you who want to experiment with this circuit in low-power modes should connect pin 8 of IC2 to VCC.

R1 is a whopping 15 MΩ, a value chosen to get a good signal out of the plant. However, it may be a hard-to-find value. It is possible to lower its value to about 4.7 MΩ at the expense of signal amplitude. The gain of the amplifier stage may have to be adjusted to compensate for this.

The next step in the signal processing chain is the low-pass filter constructed around IC1.A. It has a cutoff frequency of about 2 Hz. IC1.D adds more amplification while also suppressing frequencies above 2 Hz roughly (**Figure 4**). The total gain of the input stage is about 1,500.

IC1.C provides a virtual ground for the single-supply input stage.

Digital potentiometer IC4 (50 kΩ) adjusts the offset compensation. The IC itself is controlled over an I²C bus connected to the MCU, a PIC16F15325 in a 14-pin package (see **inset**). Not only does it control the digital trimpot, it also produces PWM signals to set the colours of the two RGB LEDs. LED1 is the main LED placed in the centre of the circular printed circuit board, LED2 is for background colour accompaniment. Transistors T1 through T6 provide the current required by the LEDs to shine brightly.

Connector K3 is intended for in-circuit programing of the MCU.

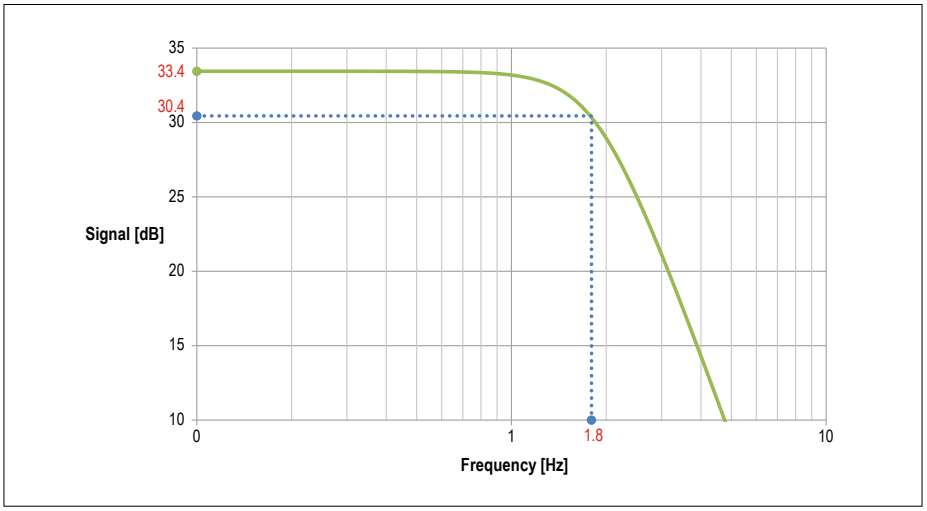


Figure 4. The transfer function of the filter and amplifier stage created with IC1.A and IC1.D. The gain is slightly over 33 dB and the cutoff frequency lies at 1.8 Hz.

Building a Bio-Light

To fit this circuit inside our translucent flowerpot we designed a suitable PCB with a thickness of 1 mm and an outer diameter of 66 mm. Although tightly packed with tiny SMT parts — they are all placed in an inner circle with a diameter of 36 mm — it is not too difficult to assemble for the more experienced solder iron warrior. The trickiest part is probably IC4, not because it is pretty small, but because you need very good eyes to see where its pin-1 marker is.

The LEDs must be mounted close to the board and their pins should be cut close to the board's underside. Try to keep this side of the board as smooth as possible. Do not mount any of the connectors.

Download the software from [1]. To program the MCU stick a pin header with long enough pins through the holes of K3, connect it to a programing pod and push the header to the side to ensure the contact is proper while the program is being uploaded. Remove it when done.

Self-test

Now it is time to test the board. On K3 short pins 3 and 4 with a piece of wire, do the same for pins 1 and 2 of K1, then connect a 5-V power supply to K2 (or the holes H2 and H6). When the supply is switched on, the board will execute a self-test. First LED1 will cycle through its colours red, green and blue, and then LED2 will do the same. This is followed by a white flash of both LEDs.

Next is the digital potentiometer; if you connect a voltmeter to pin 6 of IC2 it should show a slowly changing voltage. LED2 should fade along while this is in progress, indicating that I²C communication is alive and kicking. Then both LEDs will flash once again and finally, one second later, normal operation starts. After the board tests OK, remove the strap from K1. You can also remove the strap from K3, but it does no harm when left in place.

Flowerpot assembly

The translucent flowerpot we used is a two-piece job. In the bottom half sits a circular triple AAA battery holder with built-in On/Off switch. The plant goes in the top half. The battery holder can be taken out. On it sits a rusty RGB LED; remove it but keep the wire from the power switch.

The PCB has holes placed at the position of the connections of the battery holder. Place the board so that the negative pole of the battery holder sticks through H2 (GND). You may have to bend and squeeze it a little. Similarly, the positive pole should stick through H1 (NC). The board should rest as flat as possible on the battery holder. Solder the board to the battery holder. Solder the loose wire from the On/Off switch to H6 (+5 V), see **Figure 5**.

If you put fresh batteries in the battery holder and switch the assembly on, the LEDs should light up.

Drill a 3-mm (or so) hole close to the bottom in the side of the compartment that holds the battery holder (**Figure 6**). Solder two long wires to K1 and stick them through the hole you just drilled. They

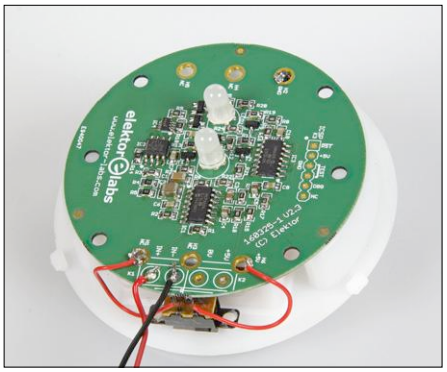


Figure 5. The Bio-Light board mounted on the battery holder. Note how the power switch is connected between holes H1 and H6.

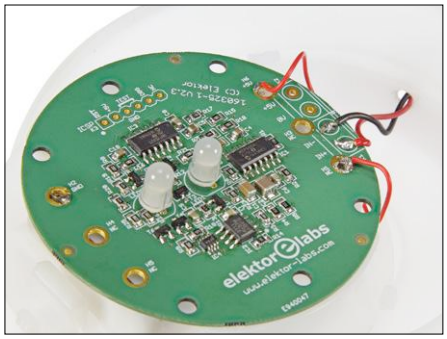


Figure 6. Drill a hole in the battery compartment close to the bottom of the flowerpot for the electrode wires to keep potential incoming water as far away as possible.

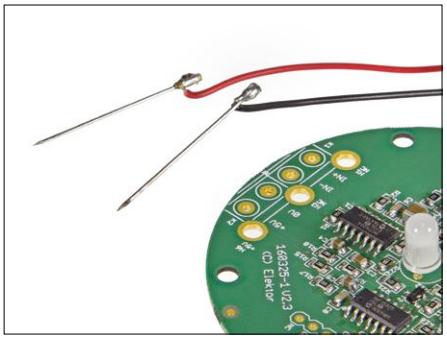


Figure 7. Dressmaker pins make for practical electrodes that are easy to attach to a plant. However, there may be better, less painful solutions like hairclips.

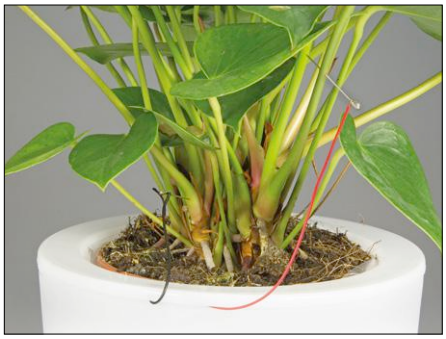


Figure 8. This is how we hooked up our Spathyphillum. Maybe the electrodes are too invasive?

should now be inside the bottom half of the flowerpot. Pull the wires up until you can click the battery holder back into its compartment. When done, verify that the assembly still works.

Hooking up the plant

You can drill a hole in the rim of the top half of the flowerpot and stick the wires through to hide them as much as possible, or you can let the wires come out in between the two halves (that's what we did). Solder a metal dressmaker pin or sewing needle at the end of each wire (**Figure 7**). If you feel that this method is barbaric, you may want to experiment with hair- or paperclips.

Assemble the two halves of the flowerpot, and insert the plant. Stick the needles in the plant at some distance from each other, for instance in two different branches, or one in a branch and the other in a leaf on another branch (**Figure 8**).

Switch on the flowerpot. At first the colours will probably change rather wildly. This is normal, it takes a while before the carrier signal is suppressed and the plant becomes its cool self again. Sometimes, for no apparent reason, the colour will change wildly during half a minute or so. (Why?) If nothing happens and the colour doesn't change at all, you may have to try another plant. We have obtained good results with a Spathyphillum, which happens to be a NASA-approved indoor-air quality improving plant.

Let's team up!

Because we do not know what to make of the colours produced by a single plant, it might be interesting to collect and compare the data from several plants. Maybe, if many people install a Bio-Light with the same plant species like the Spathyphillum, connected largely identically, and the share their observations, it might be possible to detect patterns and find a meaning for the signals? Let's team up and crack the secret of our vegetation's biosignals together. ⏮

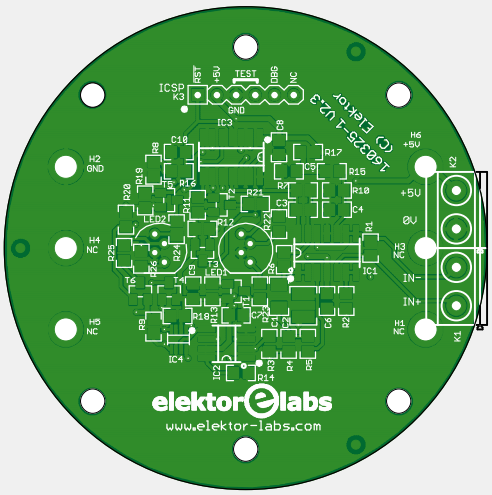
(160325)

Web Links

[1] www.elektormagazine.com/160325

[2] www.elektormagazine.com/labs/Bio-Light

COMPONENT LIST



Resistors

All SMD 0805
R14 = 0Ω
R17 = 10Ω
R21,R22,R23,R24,R25,R26 = 220Ω
R2,R6,R8,R9,R10,R15,R16 = 10kΩ
R4,R5 = 15kΩ
R3 = 330kΩ
R7,R11,R12,R13,R18,R19,R20 = 470kΩ
R1 = 15MΩ

Capacitors

Default: SMD 0805
C7,C8,C9,C10 = 100nF
C3 = 150nF
C4,C5,C6 = 1µF
C2 = 3.3µF, SMD 1210
C1 = 10µF, SMD 1206

Semiconductors

IC1 = MCP6004-E/SL
IC2 = TLC271ID
IC3 = PIC16F15325-I/SL
IC4 = CAT5171TBI-50GT3
T1,T2,T3,T4,T5,T6 = BSS84P
LED1,LED2 = RGB, CC, 5mm

Miscellaneous

K1,K2 = 2-way PCB screw terminal block, 0.2" pitch
K3 = 6-pin pinheader, 0.1" pitch
PCB # 160325-1 (1mm thick)
2 pcs. dressmaker needle or hairclip
Translucent (19cm high, 17cm diameter) LED flowerpot with 66mm diameter battery compartment

FROM THE STORE

→160325-1
Bare Bio-Light PCB

→160325-91
Ready assembled module

PlatformIO, the Universal Programming Tool

A Swiss Army Knife for microcontrollers

Embedded systems programmers who have to turn their hand to a wide range of boards will be familiar with the problem: the development environment supplied by a manufacturer will only run under one particular operating system and will only talk to a certain range of microcontroller devices. Other environments may be more universal, but may be limited in functionality or (sometimes extremely) pricey. The PlatformIO universal environment aims to overcome these disadvantages: it supports a wide range of operating systems, boards and development environments and offers many useful functions. With the help of some simple examples we show you how to get started.

By **Markus Ulsass** (Germany)

PlatformIO is an open-source project begun by the Ukrainian programmer Ivan Kravets, and has now been running for about three years. The aim of the project is to provide a common basis for programming microcontrollers from the widest possible range of manufacturers, running on several different operating systems and supporting a diverse range of development environments. This foundation is called 'PlatformIO Core', a command-line-oriented, free open-source programming tool written in Python 2.7. Different environments can be integrated through the use of plug-ins.
At the time of writing the project supports more than 400 development boards (including boards for 8- and 32-bit processors from Atmel, Freescale, Microchip, Nordic Semiconductor, Silicon Labs and STMicroelectronics, and the ESP8266 and ESP32), twenty text editors and development platforms (Atom, VS Code, Eclipse, Vim, Visual Studio and more), and ten frameworks including Arduino, CMSIS, ESP-IDF and mbed. On top of that, there are hundreds of libraries available for the environment, and the project continues to grow.
The PlatformIO GitHub repository can be found at [1], and the project website with more comprehensive documentation is at [2].

Two of the most popular programming environments and text editors that can be used with PlatformIO are Atom and Visual Studio Code. For each of these there exists a PlatformIO IDE plug-in that provides a graphical user interface on top of the PlatformIO Core.

We will look in this article at how to install PlatformIO and demonstrate a few functions and example programs with the help of the plug-in for the Atom text editor [3], which has the most advanced level of integration and which offers a wide array of application possibilities.

PlatformIO IDE for Atom

The first thing we need in the installation file for Atom, which is available from [3] for each supported operating system. After installation the program starts automatically.

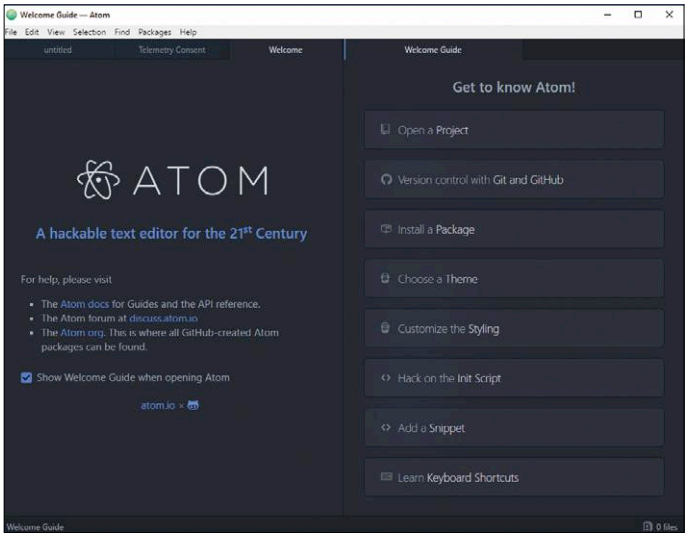


Figure 1. Additional modules (called 'packages') can be integrated into the Atom text editor.

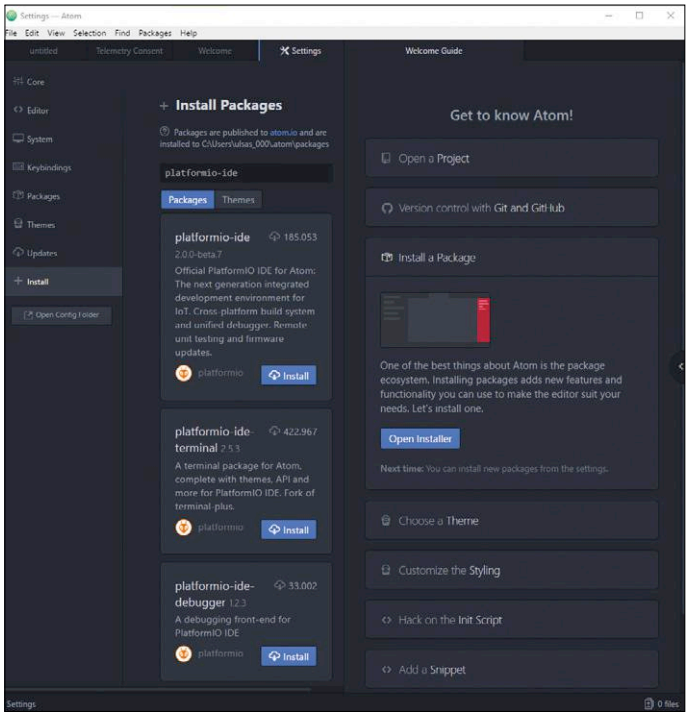


Figure 2. The PlatformIO IDE is available as a package.

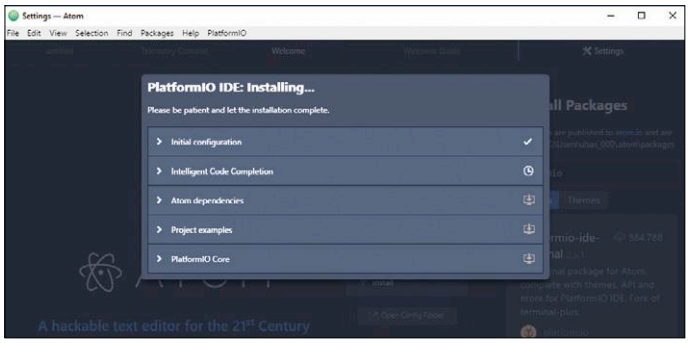


Figure 3. Patience, Hard Thing! (after Gerard Manley Hopkins).

It is possible to expand the Atom text editor with powerful additional functions by installing packages. In order to install the PlatformIO IDE package in Atom, first select the 'Install a Package' button in the right-hand pane of the welcome screen (**Figure 1**). Then click on the blue 'Open Installer' button and

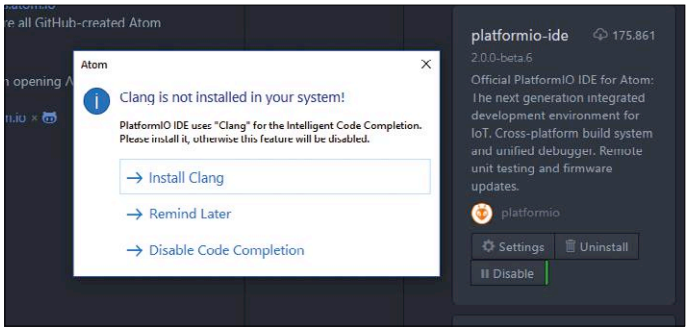


Figure 4. Clang and Python must be installed manually.

in the search box in the settings tab that opens enter 'platformio-ide' and press Enter. A list of options will appear showing various PlatformIO extensions: the one we want is called just 'platformio-ide'. Select that one and click on 'Install' (**Figure 2**). The installation process for the PlatformIO IDE package will now start (**Figure 3**), which will also require the installation of Clang (a C/C++ language frontend which includes code completion features) if you do not already have it, as well as Python 2.7. At the relevant points in the installation process you will be invited to click on 'Install Clang' and 'Install Python', which will take you to the relevant websites to download the install files and instructions. In both cases these programs must be downloaded and installed manually.

In the case of the Clang installation, when on the web page at [4] you will need to scroll down to the item 'II. Clang for intelligent Code Completion', where you will find installation instructions for the operating system you are using.

An important point when installing Clang on Windows is to make sure that the option 'Add LLVM to the system PATH for all users' is selected in the LLVM setup window; otherwise you may encounter errors later on (see **Figure 5**). All other options can be left at their default settings.

A similar step is necessary during the installation of Python. In this case the option 'Will be installed on local hard drive' must be selected beneath 'Add python.exe to Path' under 'Customize Python 2.7.12' during setup: see **Figure 6**.

Once the package and the other required software have been installed Atom must be restarted (see **Figure 7**).

After the restart all the components are installed and we can now get down to writing and testing some code. As always, our examples are available for download from the Elektor website at [5].

Ready to code

For our first outing we click on the 'Project Examples' button in the new 'PlatformIO Home' tab (see **Figure 8**). There we select 'espressif\esp8266-wifiscan' and then click on 'Prepare', as shown in **Figure 9**.

Because this example originates in a project created using the Arduino IDE, a notification will pop up telling you that the 'Smart Code Linter' C/C++ syntax checker for Arduino sketches is not enabled. At this point you can manually enable this function, and we will also be demonstrating it in one of the example projects described below. When you enable the function a web page will automatically open describing what is involved in using Arduino code and how to use the Smart Code Linter. The Arduino sketch appears on the right-hand side of the 'WifiScan.ino' tab: see **Figure 10**. If the file does not open automatically, you can select it manually from the 'Project' tab on the left.

In the interests of removing clutter, it is possible to close the tab on the left, which belongs to Atom, as well as the PlatformIO IDE 'Home' tab.

This view clearly shows the strengths of the PlatformIO IDE. The window is logically structured: on the left is a menu bar, a summary of the project files and code; on the right is a quick overview of your position in the code; and along the bottom edge are various status indicators and menus. Even in complex projects with many source files and libraries, this clear overview can be a very helpful thing to have.

A further special feature of the PlatformIO IDE is its config-

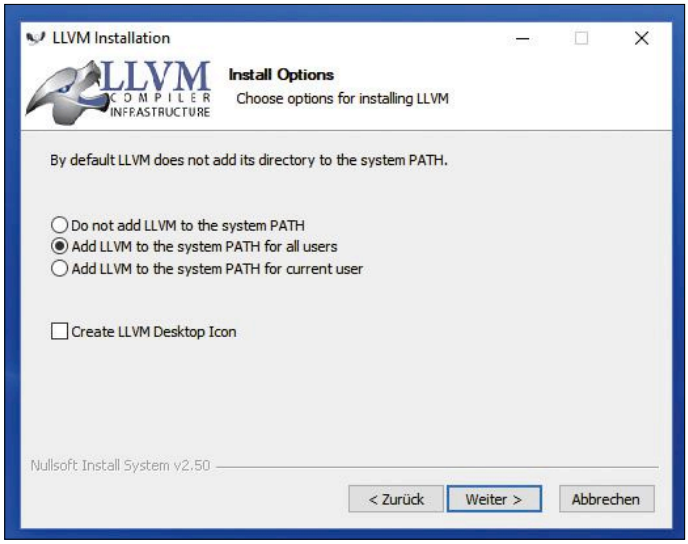


Figure 5. The option 'Add LLVM to the system PATH for all users' must be selected.

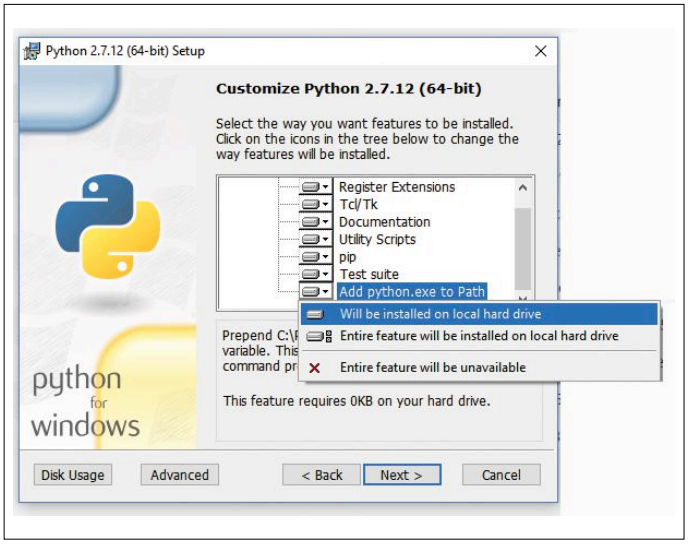


Figure 6. The option 'Will be installed on local hard drive' must be set when installing Python

uration file, called 'platformio.ini'. Here you can configure a wide range of different aspects of the system, including features to allow the same source code to be used with different boards, special directives regarding the use of libraries, and upload options.
In the example code we can see in the file 'platformio.ini' that 'board' is set to 'esp01'. Let's see how to change the project so that it can also work with a Wemos D1 mini board.

One program, many boards

A new board is added using the menu item PlatformIO -> Initialize or Update PlatformIO Project, or using the corresponding icon in the menu bar on the left (a stylized page with '<>' symbols). Under 'Selected Board' beneath the heading 'PlatformESPRESSIF8266' select 'Wemos D1 mini' and then click

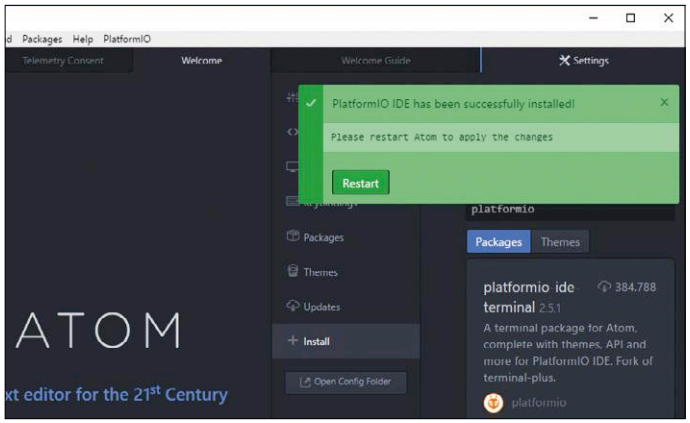


Figure 7. Restarting Atom.

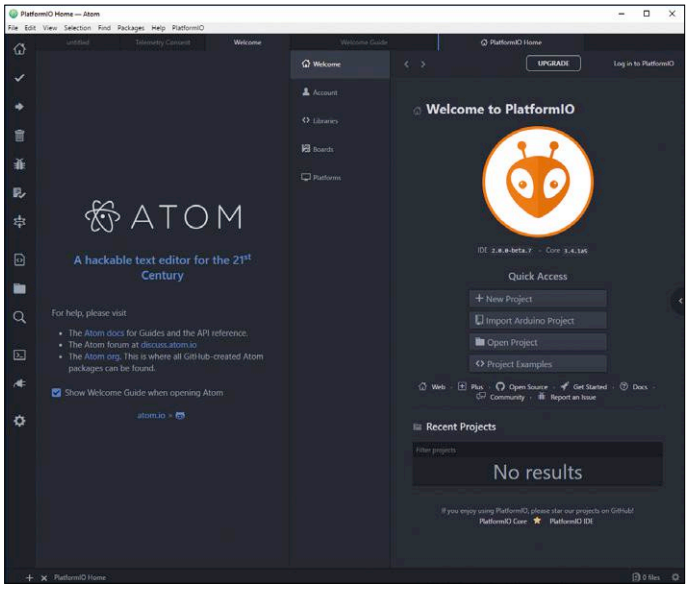


Figure 8. For our first example we click on the 'Project Examples' button in the 'PlatformIO Home' tab.

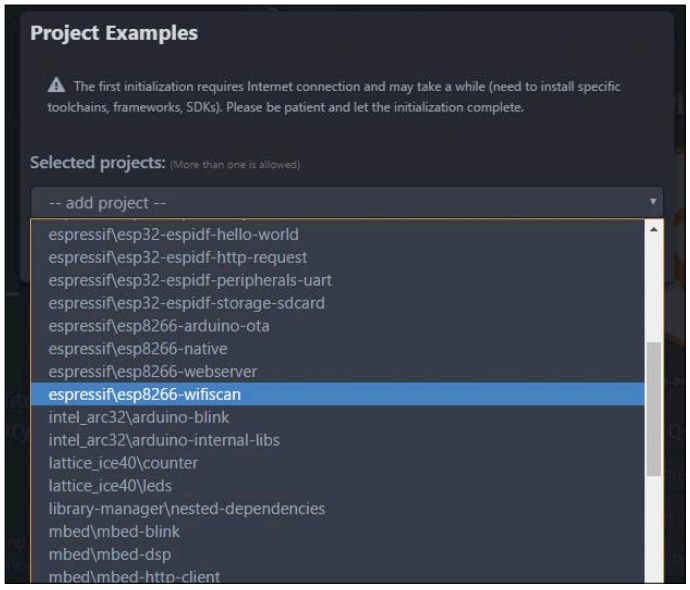


Figure 9. This is where the example program is selected.

on the blue ‘Process’ button. This will cause all the components required to use this board to be installed.
In the file ‘platformio.ini’ we can now see a new entry for the Wemos D1 mini board (**Figure 11**). We have to delete the entry for the esp01, as only the first board listed in the configuration file is taken into account by the Clang code completion functions and the Smart Code Linter syntax checker. We can now save the file using Alt-S.
We will now try out another interesting feature of the PlatformIO IDE: its automatic code completion. In the source file ‘WiFiScan.ino’, immediately after the line `Serial.begin(115200);` enter `Serial.pr`. A window will automatically pop up with the possibilities available. Use the cursor to select `size_t print(char)` and press Enter. The function name is automatically completed, and we can proceed to replace `char` with `Setup start - `, so that the line reads:

```
Serial.print(“Setup start - “);
```

As always, don’t forget the semicolon! And again, save the file using Alt-S.

To build the project click on the tick in the menu bar on the left or use F7 and choose the ‘Build’ option.
We now connect our Wemos D1 mini to a USB port and upload the code to it by clicking on the rightward-pointing arrow symbol in the menu bar on the left. The system will automatically select the correct COM port to use and send the code to the board.
Although the PlatformIO IDE can automatically detect boards connected to a USB port, it is also possible to configure individual settings, such as port and baud rate, manually in the ‘platformio.ini’ project file: see below.
Now click on the plug symbol in the menu bar on the left, verify the automatically-detected port, set the baud rate to 115200, and then confirm by clicking on ‘Start’. The serial monitor will open and a scan of your local WiFi networks should appear in the new console window.

Build your own project

Now that we have explored the basic functions of the PlatformIO IDE in Atom we can turn to writing our own example project. First select the menu item PlatformIO -> Initialize or Update PlatformIO Project, or use the corresponding icon in the menu bar on the left (a stylized page with ‘<>’ symbols), to create a new project. Select the ‘Wemos D1 mini’ board and the ‘Arduino Uno’ boards. Now create a new folder at any desired point on the local disk drive: this will become the project directory to contain our source code. This is done by selecting ‘Choose the directory’. Next click on ‘Process’, which will set up the toolchains for the two selected boards.
Once that is done the ‘Project’ area of the window will show the newly-created folder and files. Entries for both the selected boards appear in ‘platformio.ini’. Now, if we connect the two boards to the host using USB cables, we can start to look at what PlatformIO Core can do in more detail.
The PlatformIO IDE package for Atom is almost always used through its graphical user interface. However, we can use the console to dig down into the underlying PlatformIO Core. To do this, we open a new terminal window (by using the menu item PlatformIO → Terminal → New Terminal, by pressing Alt-Shift-T, or by clicking on the monitor symbol in the menu bar on the left). The new window will appear towards the bottom of the code window.
Typing ‘pio --help’ into this new terminal window brings up a list of the various options that the PlatformIO Core provides. Since we have two boards connected to the host, the first thing we would like to find out is which port each board is connected to. To see this, enter the command ‘pio device list’ in the terminal window.
The result is a list of the ports for the boards (see **Figure 12**). It is of course also possible to obtain this overview information through the menu system (using PlatformIO → List Serial Ports).
To ensure that the correct port is used when we subsequently upload code, we need to add entries to the file ‘platformio.ini’. Under Windows we add a line similar to the following

```
upload_port = COMX
```

for each board in question, with the ‘X’ replaced by the correct port number. So for our Wemos D1 mini board the entry might appear as follows.

```
[env:d1_mini]
platform = espressif8266
board = d1_mini
framework = arduino
upload_port = COM3
```

We now create a file called ‘main.cpp’ in the ‘src’ directory, by right-clicking on the directory and selecting ‘New File’. We enter the following lines into the file.

```
#include <Arduino.h>

// Blink sketch

void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  delay(1000);
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
}
```

This is the familiar ‘Blink’ sketch for the Arduino, with the sole difference that we need to include the Arduino header file ‘Arduino.h’. Now we save the file.
We can now look at another very useful function, syntax checking. If you deliberately introduce an error into the source code, for example if you change the function name `pinMode` to `pinModa`, then when you save the file the syntax error will immediately be highlighted with a red dot at the line in question. If

you now click on the red question mark in a circle in the menu bar at the bottom you will see further information about the error, including a description and the line number. In comparison to the Arduino IDE and other development environments this is a very helpful method of error reporting.
It is also very easy to upload the same code to different boards. To upload the code from our project to both boards, simply use 'Build' and then 'Upload'. If you want to upload only to one board, choose 'Run other target...' from the PlatformIO menu (or use the icon in the menu bar on the left) and then select the board you want to use for the build and upload process: you may need to scroll down. In this way you can test identical code on a range of development boards simultaneously.

Scratching the surface

Although we have explored many of the features of PlatformIO here, we have only really scratched the surface of this impressive project. Examples of areas we have not looked at are the possibility of using different libraries from project to project, the ability to import Arduino projects, and the integrated library manager with search facility.
PlatformIO also offers further powerful components including (automatic) firmware upload that can even be controlled remotely ('PIO Remote'), a debugger ('PIO Unified Debugger'), and support for unit testing and continuous integration (CI). However, not all of these add-ons are free of charge. An overview of the features and their costs can be found at [6].
A description of all these options, although very interesting even for less advanced users, is unfortunately beyond the scope of this article.
In conjunction with the services described above, the future will likely see the (paid-for) 'PIO Plus PlatformIO Core' service take a more central role. It should be possible to integrate this kernel more easily into other development environments. The open-source basis for the system should remain free of charge, however, with its support and further development funded by revenue from the paid services.

If you are a professional developer or programmer who is hitting the limits of the Arduino IDE or other simple development environment, or if you would like to develop in a more platform-independent way, you should take a look at this project and its wide range of features. But be warned: you may well find yourself unable to do without it!

(160061)

Web Links

[1] https://github.com/platformio

[2] http://platformio.org

[3] https://atom.io

[4] http://docs.platformio.org/en/latest/ide/atom.html#clang-for-intelligent-code-completion

[5] www.elektormagazine.com/160061

[6] http://platformio.org/pricing

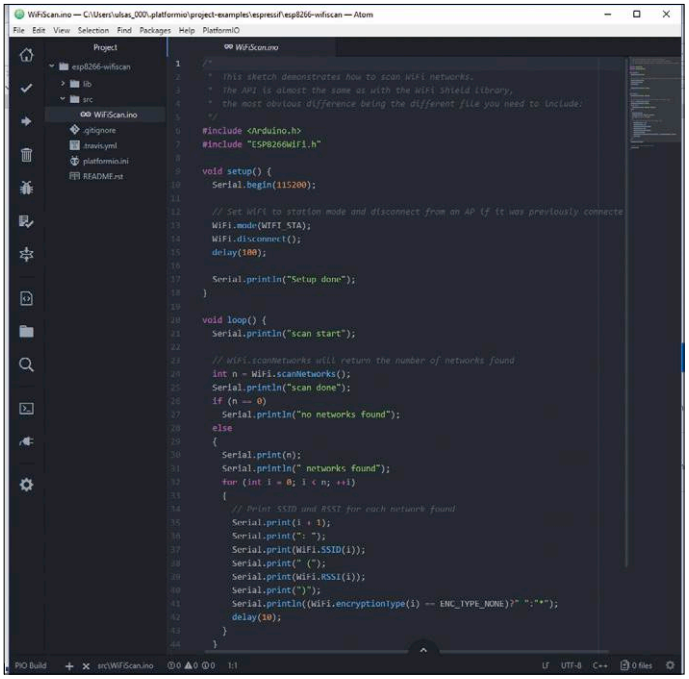


Figure 10. The Arduino sketch

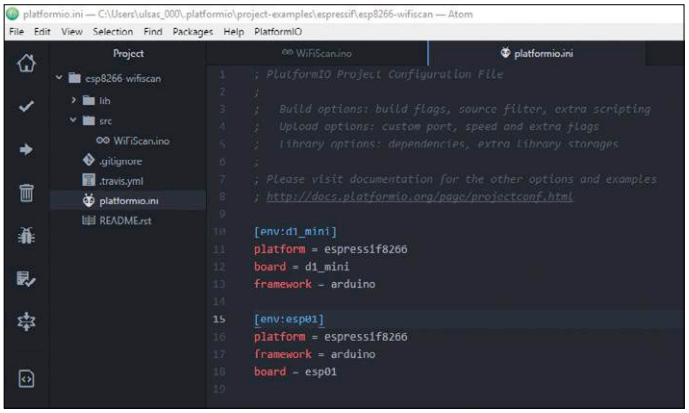


Figure 11. Board entries in platformio.ini.

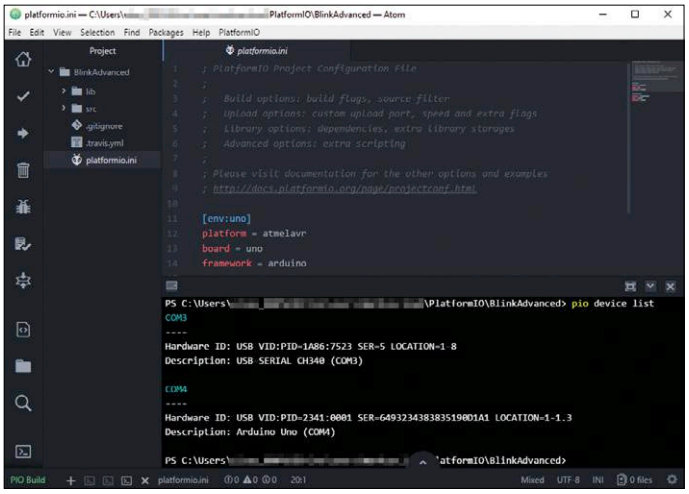


Figure 12. The ports attached to the boards that are being used can be displayed using a simple command.

Tips and Tricks

From readers for readers

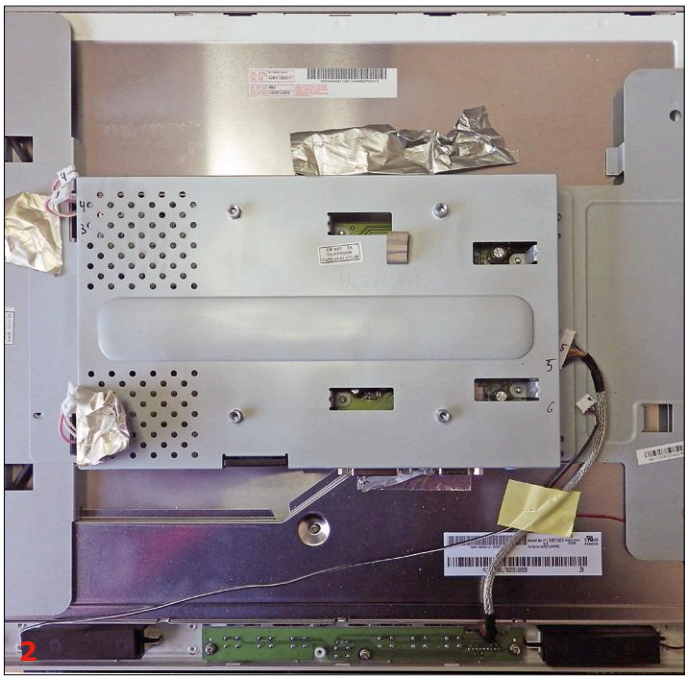
Another neat solution to a tricky problem.

New life for a failing monitor

By **Rolf Gerstendorf**

Some years ago I bought a good-spec, low-cost monitor featuring S-IPS screen technology with a wide color range. Even though I only use this 4:3 aspect display as my second, backup monitor I was slightly dismayed when it recently began behaving oddly, indicating that it may be on its last legs. After switch-on the display brightness began to pulse up and down at regular intervals and even the on/off switch (!) backlight illumination began flickering at the same rate until the circuit warmed up. At first the brightness fluctuations only occurred immediately at switch-on and lasted for about a minute, but as the days went on these periods of flickering lasted longer and longer. t's sad to think that the most common solution to the problem nowadays would be to ditch the monitor and just order another. After some research I discovered the replacement cost for a graphic-capable monitor is not trivial. I can think of better things to spend my hard-earned cash on, so I set about researching the web to see if there was a simple lower-cost repair procedure to fix the developing fault.

1


2

*It turned out that this was a fairly common problem for TFT displays generally and could be traced back to the low-ESR capacitors in the switched-mode supply section giving up the ghost (***Figure 1***). Amongst all the well-intentioned but somewhat nebulous advice given on some web pages I came across a site called Elko-Verkauf [1], this site is devoted to offering a life-line to dying electronic equipment suffering from dried-out and failing electrolytic capacitors. The website owner Andreas Grupe from Hannover told me how he got the idea:*

'I make a point of not throwing out electronic equipment on a whim, that way it helps to conserve our limited natural resources for the next generation to enjoy. My own 15" monitor was headed to the junkyard but was only suffering from faulty electrolytic capacitors. I needed low-ESR type electrolytics to repair it and it dawned on me that other people may also be searching for similar specialized capacitors to repair electronic devices. That was how my business 'Elko-Verkauf' got off the ground several years ago" ('Elko-Verkauf' translates as 'Electrolytic Capacitor Sales', Ed.)

A new business was born! The company can supply complete

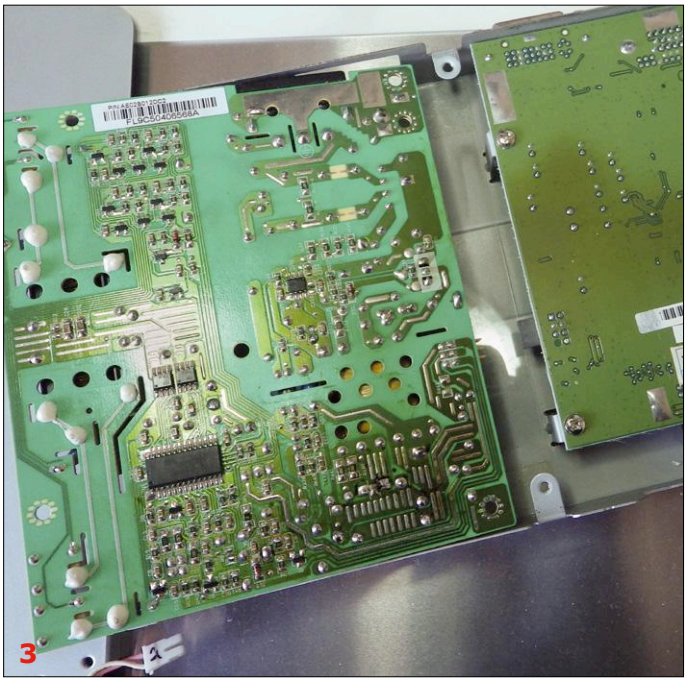
sets of replacement capacitors at relatively low cost and its web site lists hundreds of pieces of equipment ranging from beer dispensers through music studio mixer desks to plasma TVs. The service provided doesn't stop there; most of the equipment listed also has illustrated instructions on how to disassemble the equipment. This information can be really useful; it's sometimes tricky to take equipment such as my TFT monitor apart without causing damage. Andreas went on to say:
"What I personally care about is mutual support. Only through good cooperation with my customers was I able to produce these sought-after kits and instructions that make life easier for equipment repair purposes. Using customer feedback it was possible to create a well-researched knowledge data base for a wide range of equipment that can suffer from 'electrolytics breakdown'."

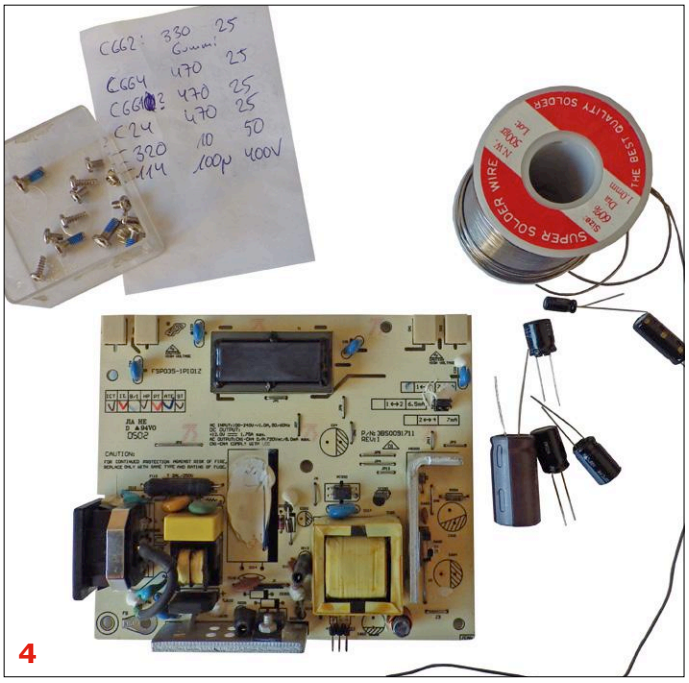
His philosophy sounds commendable and ticks the 'politically correct' box for sure. After a short search on his site I was able to locate my TFT model and download its corresponding disassembly instructions which cost just 2 euros. I must say that my particular model was not exactly the same as that described in the manual but the instructions were never the less applicable and showed how to take the monitor apart without breaking anything.

*The electronics are contained in a shielded metal enclosure (***Figure 2***). A few cables need to be carefully unplugged; the instructions contain information on the cables and suggest marking them for easier reassembly. The metal enclosure was now flipped over. The mains supply PCB (in* **Figure 3** *on the left next to the inverter PCB) was easy to remove and not long after, minus all its strips of foil, tape and fixing screws it was lying on my bench in the lab. The values and voltage ratings of all the suspicious electrolytics on the board matched those listed in the complete repair kit so I placed an order and after two days a jiffy bag dropped into my mail box containing the six electrolytics.*

*Now I had all the parts necessary to repair the monitor's power supply. First of all, it was necessary to identify the electrolytic capacitors and record their component numbers and values (***Figure 4***). The polarity of each capacitor is usually printed on the PCB but if not you can now mark it on using a fine-tipped Sharpie or similar marker pen. The old capacitors were now desoldered and removed without any problem. Only one solder pad was close to two SMD components so I needed to be careful not to disturb them.*

After 15 minutes the new electrolytic capacitors were fitted and re-assembly of the unit began. Three quarters of an hour later everything was back together and I was able to hook it up to my PC. It worked a treat!


3



4

To sum up: my total investment of around 15 euros and one hour's work was all it took to get the monitor working correctly again, thereby saving me a few hundred euros for a replacement monitor. ⏮

(160507)

[1] www.elko-verkauf.de

Have you come up with an inspired way of solving a really tricky problem? Perhaps you've discovered an ingenious but unconventional way of using a component or tool? Maybe you've invented a better or simpler way of tackling a task? You deserve a reward, write in — for every tip we publish you win £40 (or local equivalent)!

eLABS PROJECT

Wi-Fi Desktop Thermostat

Flexible and programmable temperature control

By **Roy Aarts** & **Clemens Valens** (Elektor Labs)

Thermostats are everywhere — in the fridge, in the freezer, on the wall, in the car, everywhere are thermostats except on my desk. Sometimes I want a thermostat on my desk to control, say, a fan because it's too hot in the room. It may also happen that I need a thermostat to switch off an electric stove when the liquid I am heating has reached a certain temperature, or start a pump when it becomes too cold.

Features

- 6 preset programs
- 3 custom programs
- Wi-Fi connectivity
- Normally-open (NO) and normally-closed (NC) contacts
- Switches up to 2500 VA

What I need is a portable and flexible stand-alone thermostat capable of switching a load. If you spend enough time on the Internet, it is possible to find what we may call "desktop thermostats" that come close to what I just described. Devices with programmable On and Off temperatures and capable of switching a fan or pump. Sadly they do just that, meaning they lack the flexibility I am looking for. So I decided to design one. To be honest, I subcontracted the design of the thermostat to my favorite trainee and let him do the hard work. To make the project more interesting I added Wi-Fi capabilities to the list of specifications. While I was busy with my other tasks, Roy our trainee worked quietly in a corner of the lab until after a few weeks he looked up and said that the thermostat was ready.

Checking Roy's work revealed that his thermostat worked as intended — he had done a great job indeed. Unfortunately it still lacked the desktop practicality and software flexibility I craved for and a second design iteration was needed. Because in the meantime Roy had moved on to new adventures I had to do this myself. The result of this two-stage design process is the device described in this article.

Specifications and features

Before we dive in the technical details, let's quickly look at the thermostat's specifications:

Table 1. The six thermostat presets. The column 'No.' refers to the table indexes in the firmware, which explains number 0.

No.	Switch ON	Switch OFF
0	$T < T_{LOW}$	$T \geq T_{LOW}$
1	$T > T_{HIGH}$	$T \leq T_{HIGH}$
2	$T > T_{LOW}$ and $T < T_{HIGH}$	$T \leq T_{LOW}$ or $T \geq T_{HIGH}$
3	$T < T_{LOW}$ or $T > T_{HIGH}$	$T \geq T_{LOW}$ and $T \leq T_{HIGH}$
4	$T > T_{HIGH}$	$T < T_{LOW}$
5	$T < T_{LOW}$	$T > T_{HIGH}$

- 250 VAC, 10 A (2500 VA) relay with normally open (NO) and normally closed (NC) contacts
- 6 switching presets
- 3 custom (user) programs
- Manual override
- User interface on smartphone, tablet or computer
- Wi-Fi network connectivity
- Micro-USB connector for 5-V supply

Switching presets

The firmware features six switching "programs" that cover the most commonly needed scenarios and some uncommon ones as well. Two programmable temperature levels T_{LOW} and T_{HIGH} determine the switching points of these programs. Refer to **Table 1** for details.

Custom programs

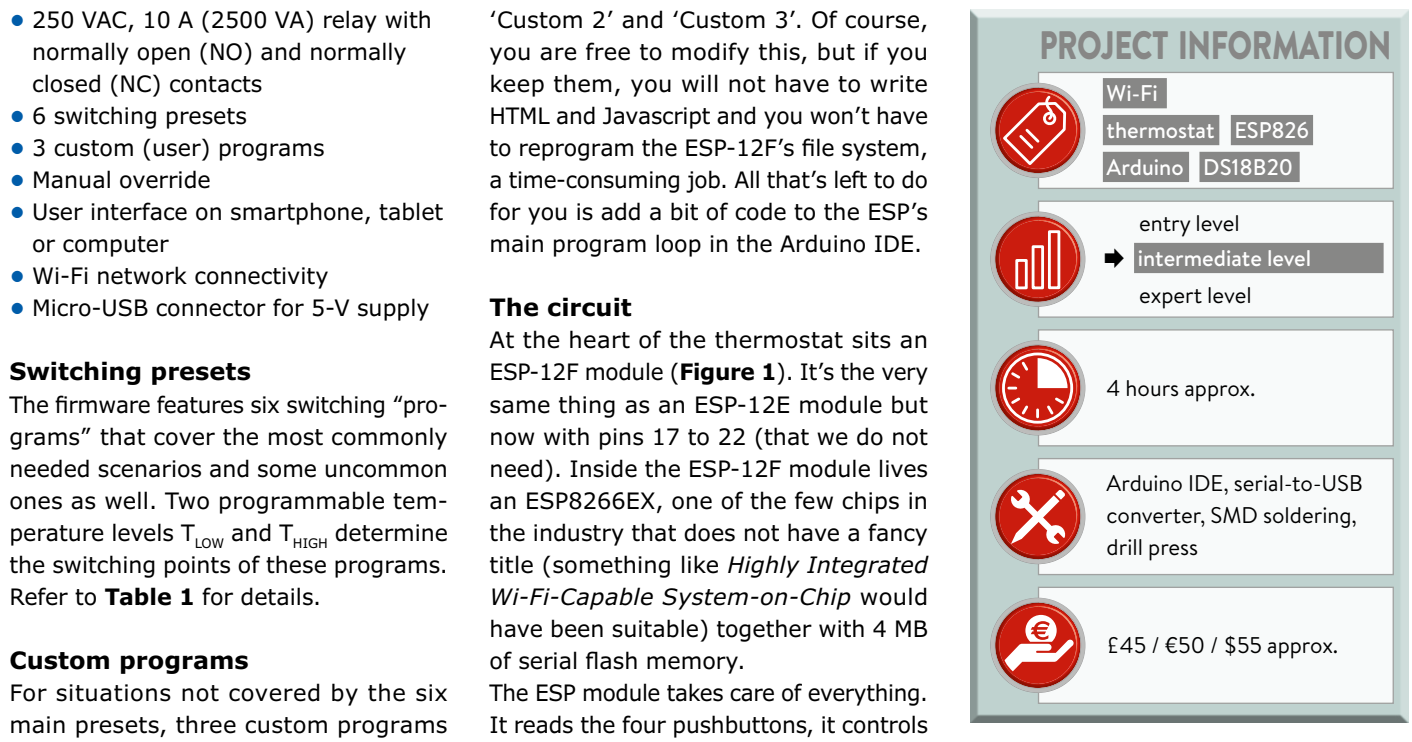
For situations not covered by the six main presets, three custom programs have been defined. They are empty in the firmware and should be written by the user. To make this exercise as painless as possible, the interface between the webpage and the firmware has already been implemented by reserving three entries in the webpage's preset list: 'Custom 1', 'Custom 2' and 'Custom 3'. Of course, you are free to modify this, but if you keep them, you will not have to write HTML and Javascript and you won't have to reprogram the ESP-12F's file system, a time-consuming job. All that's left to do for you is add a bit of code to the ESP's main program loop in the Arduino IDE.

The circuit

At the heart of the thermostat sits an ESP-12F module (**Figure 1**). It's the very same thing as an ESP-12E module but now with pins 17 to 22 (that we do not need). Inside the ESP-12F module lives an ESP8266EX, one of the few chips in the industry that does not have a fancy title (something like *Highly Integrated Wi-Fi-Capable System-on-Chip* would have been suitable) together with 4 MB of serial flash memory.

The ESP module takes care of everything. It reads the four pushbuttons, it controls the three LEDs, it communicates with the temperature sensor, and switches the relay on and off.

The power supply is built around voltage regulator IC1. D2 provides reverse polarity protection. The ESP module consumes quite some current when its Wi-Fi interface is busy and this can produce interference on the supply line. Components R11, C6, C7 form a filter to ensure that especially IC2 sees a relatively clean supply voltage.



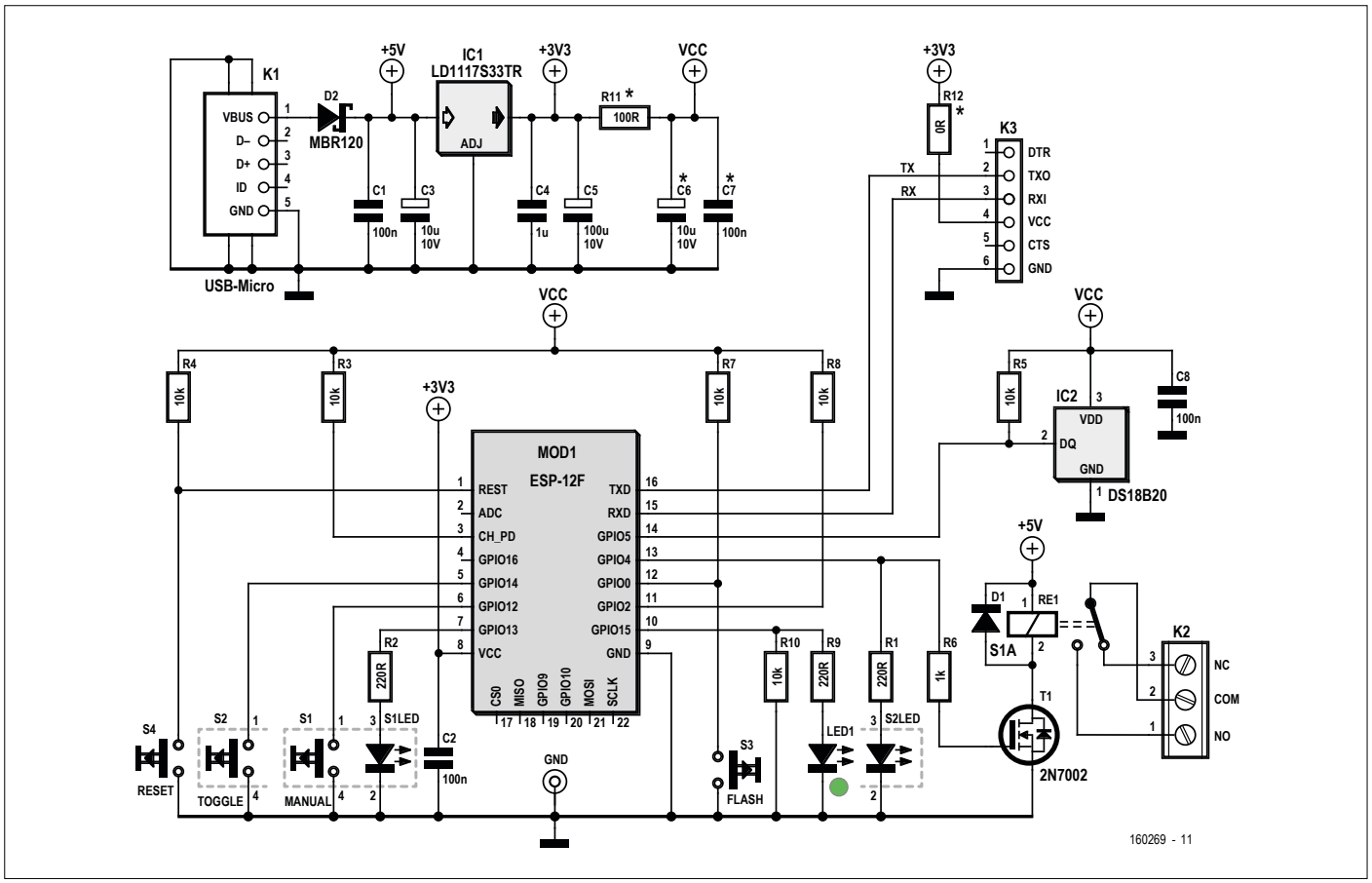


Figure 1. The heart of the desktop thermostat is not an Arduino, but IC1, an ESP-12F.

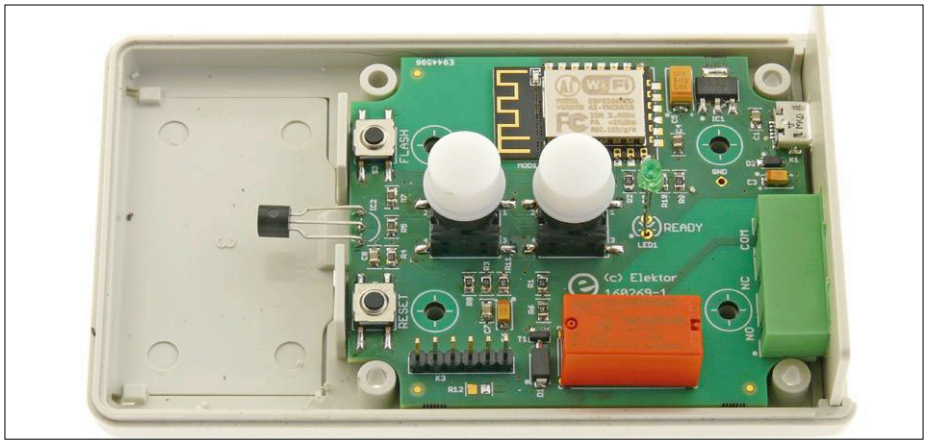


Figure 2. The PCB was designed for easy integration in a cheap enclosure.

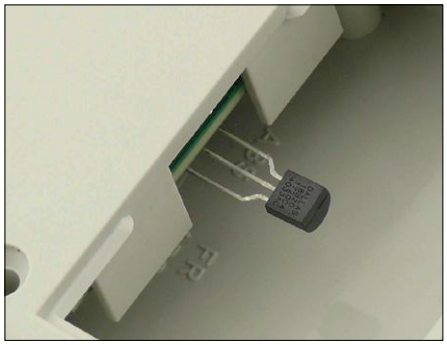


Figure 3. The battery compartment provides ample space for the temperature sensor, even when it is connected to the board with a length of wire.

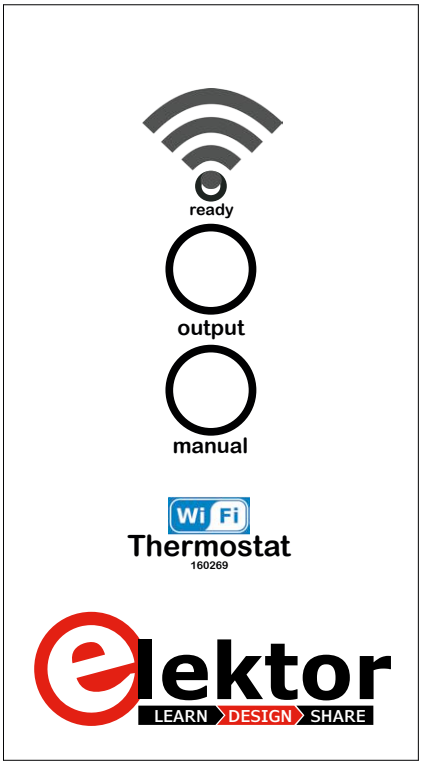


Figure 4. Suggested design for a front panel.

K3 is the serial port connector and is compatible with a 3.3-V "FTDI" serial-to-USB converter cable. Note that such cables carry 5 volts on their VCC pin even though their signals have 3.3-volt levels, which is why R12 is not mounted by default. It is available, however, in the case the serial port is to be used with some other circuit that is to be powered from the thermostat's power supply (or vice versa).

For our temperature sensor the popular DS18B20 1-Wire digital thermometer in a TO-92 package was selected. It can be mounted on the PCB, but it is also possible to connect it with wires for more freedom.

The relay has a standard footprint that can be obtained from several manufacturers. However, before buying one that fits on the board, make sure that its contacts are up to the load they have to switch. It must have a 5-V coil, and, for switching the specified 2500 VA, it needs contacts rated for a continuous (i.e. not *max.*) current of at least 10 A @ 250 VAC. Please note that the photos in this article show a previous board version with a 5-A relay which is much smaller. Because 2500 VA is a lot of power to switch the PCB has 100-mil-wide traces between the relay contacts and connector K2. Together with a standard copper thickness of about 35 µm this is good for some five amps while keeping reasonably cool. Since we want more, these traces have no solder mask and must be beefed up with a layer of solder. Not a lot is needed — 0.5 mm or so will do.

Pushbuttons

Although there are four pushbuttons on the board, only two, S1 and S2, are available to the user. With S1 the thermostat can be switched between Manual and Thermostat mode. In Manual mode the device functions as a simple On/Off switch, useful in e.g. startup or emergency conditions. In this mode the relay state is toggled with S2.

S3 and S4 are used to put the ESP module into programming mode using a secret handshake:

1. press S4 (Reset);
2. while pressing S4, press S3 (Flash);
3. release S4 (Reset);
4. release S3 (Flash).

If you executed this sequence correctly, the tiny blue LED on the ESP module will flash briefly, signalling that the module can now be programed over its serial port.

LEDs

LED1 indicates the state of the Wi-Fi connection. When it's blinking the thermostat is not connected to anything; when LED1 lights continuously the thermostat is connected. Two possibilities in that case:

1. the thermostat is itself an access point (AP) and a device (smartphone, computer, etc.) is connected to it;
2. the thermostat is connected to an access point (AP).

The fancy pushbuttons S1 and S2 have built-in LEDs. When S1LED is on, the thermostat is in manual mode; when the LED is off, it is in thermostat mode. S2LED indicates the state of the relay. When it lights the relay is active (NO contact is closed); when it is off, the relay is released (NC contact is closed).

Enclosure & construction

The board was designed to fit in a cheap ABS plastic enclosure with battery compartment, a Type 1593Qx from our friends at Hammond Manufacturing (**Figure 2**). This 112 × 66 × 28 mm (4.4 × 2.6 × 1.1") enclosure is available in grey (x = GY), black (x = BK) and translucent blue (x = TBU). When the temperature sensor is mounted on the PCB it can be pushed over so that its head will hang inside the battery compartment. With the lid removed the air can flow around it (**Figure 3**). If the sensor is mounted on wires, the battery compartment provides enough space to stow the sensor when the thermostat is not used.

A drilling template is available [1] together with a front panel inlay design (**Figure 4**) to help you build a nicely finished project. Only three holes have to be drilled.

The inside height of the enclosure is specified as 21.7 mm, the standoffs are 4.5 mm. The relay has a specified height of 15.7 mm. Mounted on a standard 1.6-mm-thick PCB you'll find that it is 0.1 mm too high. This can be adjusted by shortening the standoffs a little or by using a thinner PCB (1 mm for instance). Also, the top shell has four standoffs, one of which hinders the relay and consequently it must be removed. Luckily this is easily done with a 10-mm drill or near equivalent.

After mounting the relay do not forget to apply a 0.5-mm-thick layer (or more) of solder to the traces that run from the relay to K2. It may be necessary to shorten the standoff under the relay to compensate for this extra solder.

Extenders (5 mm and 8 mm) are available for the pushbuttons. With 5 mm extenders the pushbuttons just stick out of the top of the enclosure.

Mounting the LED requires some manual adjustments. To do this properly, first solder its anode pin — easiest to solder/unsolder — and adjust the height until you are satisfied. Then solder the other, grounded, pin.

The enclosure has a removable side panel which has to provide access to the connectors. Because we are dealing with AC line voltages it is **definitely** & highly **not** recommended to simply throw it away. The milling template also has a template for the cutouts needed for this small panel. Sure, it's small, but with a bit of effort it can be done and keeps the thermostat safe.

For the USB opening we found that a slot the size of the tip of the mating connector was enough to provide good contact. If it doesn't for you, be aware that micro-USB charger cables with extra-long tips (8 mm instead of 5 mm) are waiting in the stores.

Software

The program for the thermostat is written as an Arduino sketch with `setup` and `loop` functions. It is quite a large program due to the inclusion of a webserver and a Wi-Fi access point, but most of that can be considered black boxes. The thermostat functionality is implemented in the function `loop` and this is where you can add Custom presets. All pretty straightforward indeed. Everything can be downloaded from the article web page [1].

GET requests

More complicated to understand is the communication between the main program and the webpage that provides the user interface on the remote device. **Figure 5** shows the mechanism graphically. Here it is important to understand that the thermostat is a webserver that responds to so-called HTTP GET requests. Such requests expect an answer in return, an HTML file or an

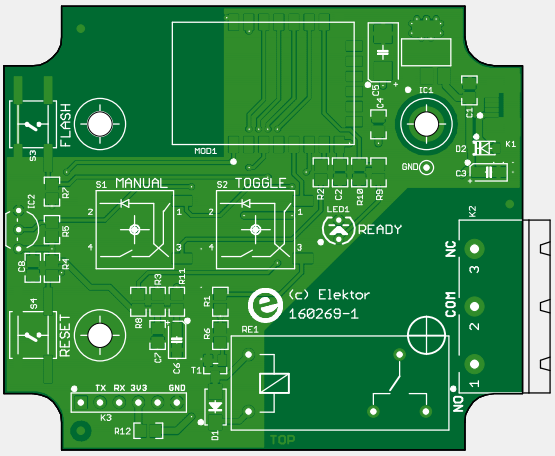
COMPONENT LIST

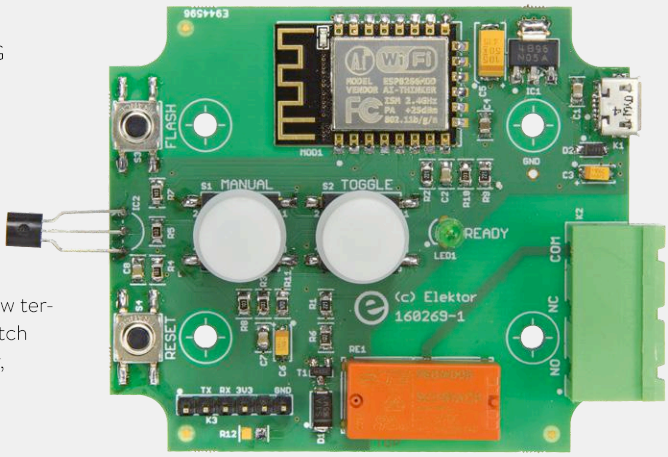
Resistors
SMD 0805, 0.1W
R12 = 0Ω
R11 = 100Ω
R1,R2,R9 = 220Ω
R6 = 1kΩ
R3,R4,R5,R7,R8, R10 = 10kΩ

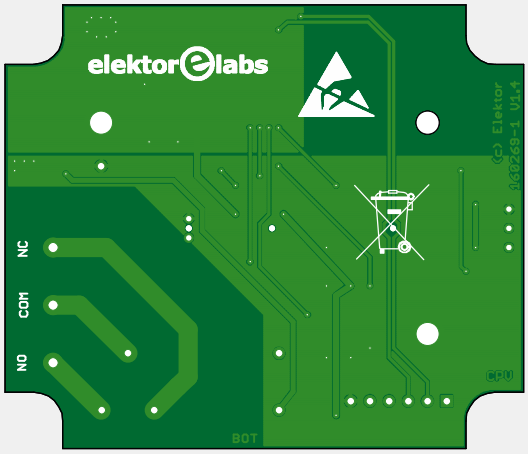
Capacitors
SMD 0805
C1,C2,C7,C8 = 100nF
C4 = 1µF
C3,C6 = 10µF, Case A
C5 = 100µF, 16V, Case C

Semiconductors
D1 = S1A
D2 = MBR120LSFT1G
IC1 = LD1117S33TR
IC2 = DS18B20
LED1 = green, 3mm
T1 = 2N7002

Miscellaneous
K1 = USB-B micro connector
K2 = 3-way PCB screw terminal block, 0.3″ pitch
K3 = 6-pin pinheader, 0.1″ pitch
MOD1 = ESP-12F
RE1 = SPDT, 5V coil, 10A/250VAC contacts (e.g. Omron G2RL-1)
S3,S4 = switch, tactile feedback, 6x6 mm, gullwing
S1,S2 = switch, tactile feedback, integral LED, Multimec 5GSH93542SMD
Cap for S1 & S2, Multimec 1US16
Extender for S1;S2, 2.5mm, Multimec 2SS09-5.0
Enclosure: Hammond type 1593Qx (x = GY/BK/TBU)
PCB # 160269-1







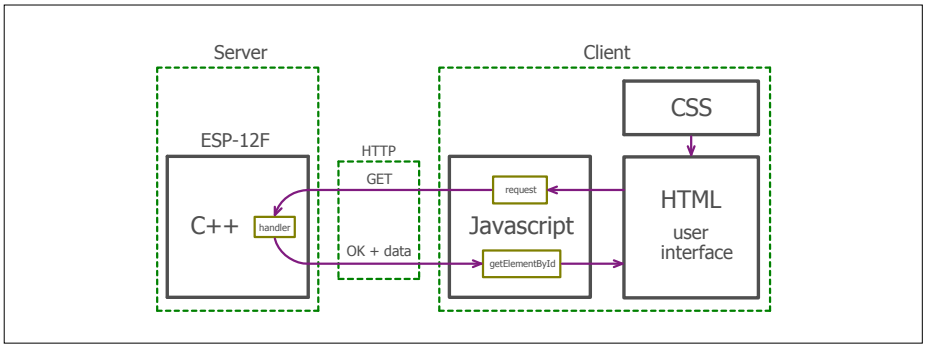


Figure 5. The thermostat is the server while the client is the phone, tablet or computer that displays the webpage. Four programming & scripting languages are used to make it all communicate.

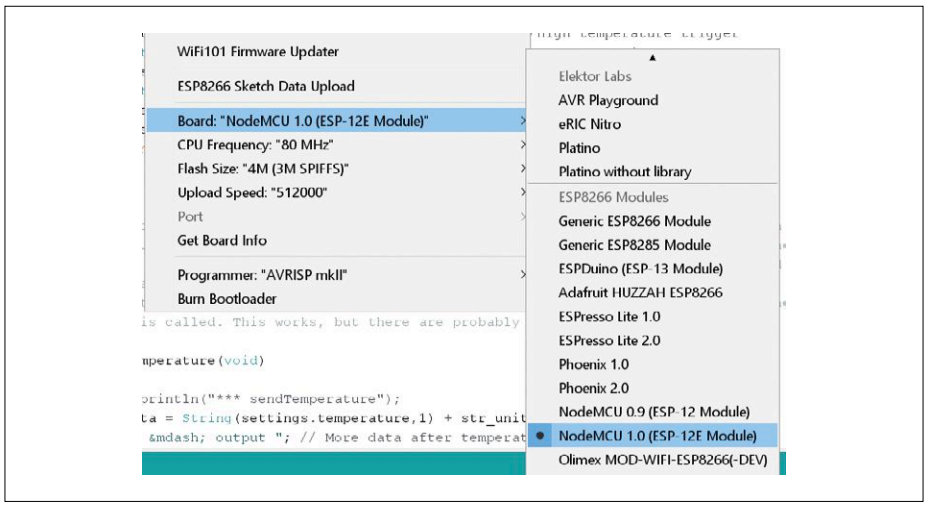


Figure 6. After installing the required Boards package and data programing tool, the Arduino IDE should know the ESP-12E/F module and show an 'ESP8266 Sketch Data Upload' in its 'Tools' menu.

image for instance, but it can also be data. At launch, the C++ function `setupWebserver` binds the supported GET requests to functions (handlers) inside the C++ program. For instance, when a GET request for '`curtemp`' — the current temperature — is received, the C++ function `sendTemperature` is called. How exactly this is done is hidden somewhere inside the webserver code inside the ESP8266 libraries. The C++ function `sendTemperature` collects and formats the requested data, and forwards it to the webserver which in turn sends it to the client where the GET request was found to originate from.

Javascript layer

Because HTML is somewhat limited and by itself would not allow everything we had in mind for the user interface, a layer of Javascript was added. This layer creates the desired GET requests and unpacks and distributes the data received in reply. To continue our example, when the webpage is loaded, a timer is started to request the temperature as measured by the thermostat at 5-second intervals. This timer, executed by the browser displaying the webpage, calls the Javascript function `request` to formulate and send the GET '`curtemp`' request. If an answer is received within a certain period, the Javascript function `showTemperature` is called with the received data as its argument. This function unpacks the data and updates the HTML page, and the user sees the temperature appear.
To make the user interface look sleek a CSS template or style sheet is used.

Setting up the software development environment

The thermostat program was written using the Arduino IDE. Because the ESP-12F is not an Arduino, the IDE has to be extended with a Boards package in order to add ESP8266 support. The procedure to do this is quite simple but requires an Internet connection. It starts from the 'File' menu by opening the 'Preferences' dialogue of the Arduino IDE. Copy the URL http://arduino.esp8266.com/stable/package_esp8266com_index.json (one line, no spaces, beware of typing errors) into the 'Additional Boards Manager URLs' box of the 'Preferences' dialogue. Close it when done. Open the Boards Manager (Tools → Boards). In the upper left corner of the window that opens select 'Contributed'. Look for 'esp8266 by ESP8266 Community' in the list that appears, click on it and then click the 'Install' button. The IDE will download the required files and copies them to the right location. When terminated, close the window. Now the 'NodeMCU 1.0 (ESP-12E Module)' will be available somewhere in the 'Boards' menu, under the header 'ESP8266 modules'; select it. Set the 'CPU Frequency' to '80 MHz' and the 'Flash Size' to '4M (3M SPIFFS)'. The 'Upload Speed' should be set as high as possible to reduce upload times. The maximum achievable speed depends on the serial-to-USB converter used.

Filesystem upload tool

After installing the Boards package a second IDE extension must be installed, needed to load the webpage in the ESP module's serial flash memory. The procedure is as easy as storing a file in a new folder and the file in question (esp8266fs.jar) and detailed instructions where to place it (<..>\sketchbook\tools\ESP8266FS\tool\) can be found at [3]. When done (do restart the IDE if you kept it open during the installation) you should have an entry 'ESP8266 Sketch Data Upload' in the IDE's 'Tools' menu (**Figure 6**).

Additional library

Finally, it is necessary to install a 1-Wire library to allow communication with the DS18B20 sensor. The library is available from [4] and can be installed with the IDE's library manager ('Sketch→Include Library→Manage Libraries…').

Flashing the firmware

Connect the thermostat by means of a serial-to-USB converter to the computer. Also connect the thermostat's micro USB connector to a 5-volt power supply. Download the thermostat sketch and data [1] and open it in the Arduino IDE. Do the flash-enable handshake

(see above: press S4, then S3, release S4, then S3). Select the correct board and serial port and launch a sketch upload. The sketch should compile & upload without errors & warnings.
The next step is to upload the webpage and everything else that resides in the sketches subfolder 'data': again do the flash-enable handshake and then select 'ESP8266 Sketch Data Upload' from the IDE's 'Tools'. Uploading should start and will take a while (several minutes) because the image is a hefty 3 MB.

Using it

Now that the firmware and webpage have been installed on the thermostat it is time to try it out. The first test is to power the thermostat and try the 'Manual' button (S1). Pressing it should toggle its built-in LED. Press it to make its LED light up. Now press the 'Output' button (S2) to toggle its built-in LED. This should also toggle the relay, which is audible. While you are doing this, LED1 should blink at a rate of 0.5 Hz.

Figure 7. The webpage served by the thermostat (in both access point mode and client mode) shows if the device is connected to a network or not. If it is, its IP address is shown in the Wi-Fi section in place of the text 'Network not configured'

On to Wi-Fi

Open the Wi-Fi settings on a smartphone, tablet or computer and look for the network 'Wi-Fi Thermostat'. When using a phone, it may be necessary to switch off mobile data first. Connect to it (no password is needed, LED1 should stop blinking and remain on), then open a browser and navigate to address 192.168.4.1. The thermostat's webpage should appear (**Figure 7**). Wait five seconds to allow it to receive and update the current temperature. If it does, all is fine and it is possible to change the settings. Click or tap ("clap") 'Submit' to send the new settings to the thermostat. It can take up to five seconds for the webpage to reflect the changes, so please be patient.

Hook it up to existing network

One section of the webpage permits configuring the thermostat for an existing Wi-Fi network. Enter its SSID and passphrase and *clap* the corresponding 'Submit' button. You must then *clap* the 'Restart' button to make the changes become effective. Note that the first restart using this button after programming the ESP module will result in a crash. This is not our fault but a known bug somewhere inside the ESP module and/or libraries. If this happens, simply power cycle the device and everything will be fine.
If, after a restart, LED1 lights up continuously, the thermostat has connected successfully to the network (to be sure, first disconnect all devices that may have reconnected to the thermostat's AP). To find out what the thermostat's IP address is, connect to its AP. The webpage will display it. You can also check your network's AP's connected devices list.

Forget

To remove the thermostat from the network, *clap* the 'Forget' button. This will erase the persistent memory of the device where its settings are stored. All its settings will be reset to their factory default values. *Clap* the 'Restart' button to restart the thermostat.

Where to go from here?

Although this article presented a thermostat, nothing prevents you from making something else out of it. What we have here is basically just a Wi-Fi-controlled relay applied to temperature control. It is, of course, only a matter of replacing the temperature sensor by some other sensor to transform the thermostat into something completely different. The software is easy to modify—all the hard parts have been premashed—and can be adapted to lots of other applications. Let your imagination run free, to throw in a cliché, and use this design as the starting point of a new project.

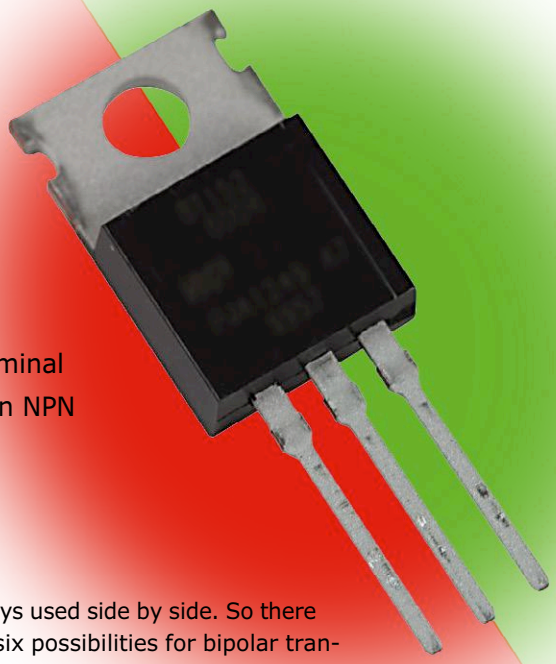
(160269)

Web Links

[1] Project support site: www.elektormagazine.com/160269
[2] ESP8266 Boards Package: http://arduino.esp8266.com/stable/package_esp8266com_index.json
[3] ESP8266FS file system upload tool: https://github.com/esp8266/Arduino/blob/d7044eceab4d4453e4d73ac49dcbbc8b8d0c9eb1/doc/filesystem.rst
[4] Arduino 1-Wire library: https://github.com/PaulStoffregen/OneWire

Bi-colour Transistor Tester

Green: NPN; red: PNP



By Hans-Norbert Gerbig (Germany)

Remarkably, in this transistor tester the TUT (transistor under test) flashes a bicolour LED. This allows you to see immediately whether the transistor is functioning, which terminal is the base, emitter and collector, and whether the trannie an NPN or PNP type.

As the experienced electronicists will notice at first glance, the circuit in **Figure 1** is an astable multivibrator consisting of a permanently installed NPN transistor (T1) in concert with the device under test. The only difference to the usual circuits is that the base of the TUT is not connected to the supply voltage via resistor R5, but to its own collector. The load resistance (R3 or R4) of the TUT is designed as a voltage divider, so that the collector (and thus also the base) is biased at half the supply voltage.

For this reason, it is possible to change the 3-mm duo LED with a simple switch from N(PN) = green to P(NP) = red. The test transistor makes the dual LED in the collector circuit flash. Resistor R6 (220 Ω) compensates the differences in brightness between green and red.

The transistor to be tested is connected to the 5-way socket strips E, B, C, E and B in such a way that three terminals are always used side by side. So there are six possibilities for bipolar transistors: E-B-C, B-C-E, C-E-B and, if you flip the transistor over, B-E-C, E-C-B, C-B-E. For unipolar transistors (FETs), the test combinations S-G-D, G-D-S, D-S-G and reversely, G-S-D, S-D-G, D-G-S.

If the connection sequence is correct — because this is the only way the multivibrator can vibrate — the Dual LED flashes green for N-transistors and red for P-transistors. Only one of the six options can be correct. If the connections E and C are interchanged, the LED may flash somewhat dimmer with some transistors. However, the basic connection is always fixed. For junction FETs (insulating layer FETs) with symmetrical structure however, only the gate is displayed with certainty. The S and D connections can be swapped without loss of power.

The small tester allows a quick and unambiguous connection determination with simultaneous functional testing. Important during testing: only the flashing LED (red or green) provides clear information. No lighting or continuous lightning means an incorrect connection sequence. Possibly the device under test is not a transistor at all but a voltage regulator, a thyristor or a triac, for example.

For the small circuit, the author has designed a breadboard and a ‘real’ printed circuit board. For both versions, corresponding files in Lochmaster or Sprint PCB layout format are available on the project page [1] for both versions. ⏮

(160542)

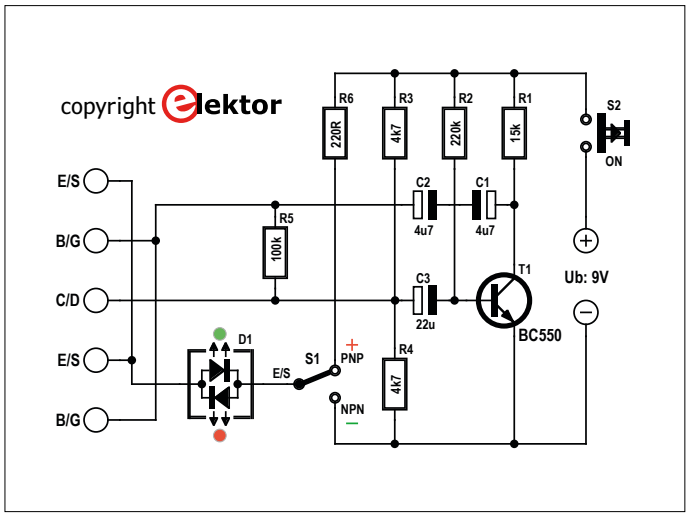


Figure 1. The transistor tester is a lightly modified multivibrator circuit.

Web Link

[1] www.elektormagazine.com/160542

Communicating Current Loops

Turning data signals into current affairs

By **Tam Hanna** (Slovakia)

Why do we find many industrial sensors equipped with a current output but no voltage output? This article provides the answer to this question and throws in plenty of other considerations.

It's rare for contact resistance to cause problems when you're experimenting in your home lab. The only time you might notice anything is if you use, let's say, a cheap and nasty breadboard or some elderly cables. But in industrial systems, unpredictable line resistances occur on a daily basis, creating a situation that pushes analogue systems based around voltage outputs to their limits.

The way to confront this problem is with so-called current loops, in which appropriate currents rather than voltages represent the signals being transmitted. **Figure 1** shows how the changes in current flowing through a wire, caused by the varying voltage drop across the sensor resistance R_{sense}, are independent both of the (ideally infinitely high) internal resistance of the current source R_i and also of the constant load resistance R_L (the measuring device). In this way the voltage drop across the meter is exactly proportional to the change in current I_Q caused by R_{sense}.

In industrial practice we implement current loops of this kind using a current range of between 4 and 20 mA, written as 4–20 mA. The maximum input resistance of the measuring device (measurement amplifier) can be calculated easily using Ohm's Law. Assuming the (sensor) amplifier in use has a maximum output voltage of 0.5 V below V_{cc}, then at a supply voltage of 5 V, we have a maximum resistance value of R_L = 4.5 V / 20 mA = 225 Ω. If the resistance turns out to be of lower value, this is not a problem. In practice higher voltages of 24 V are commonplace, by the way.

The question remains why the lower threshold is fixed at 4 mA and not 0 mA. Here's a little anecdote that actually happened just so. At an industrial installation a group of turbines were being monitored by a process control computer. An employee of the power company accidentally disconnected the encoder's input, which was Low at the time. The result was that the turbine controller sent the instruction for 'full throttle' and the turbines came within a hair's breadth of being blown up to destruction.

This near-disaster was only possible because the controller had no facility for detecting a cable failure. With the 4–20 mA system this could not happen, because any current lower than 4 mA would indicate an error and the machinery would be switched off. From a technical point of view, implementing a threshold of this kind is straightforward: when a current is measured, a fixed offset is simply added to the value to be processed.

Implementation with discrete components

Even if a current loop is very seldom constructed in practice using op-amps, this method is ideal for basic considerations. The first and most important question is how you reference the measured signal to ground. **Figure 2** shows two possibilities, in which the left-hand one may introduce a hardly-negligible error in practice, by making ground connections at more than one location. Within the confines of a small lab setup the ground potential should be more or less identical everywhere. However, this is unlikely to be the case in factory buildings, where you can find all matter of electrical disturbance superimposed on the ground potential, caused by all manner of heavy-current devices being switched on and off. These surges will falsify the signal. However, if each element in the control system is

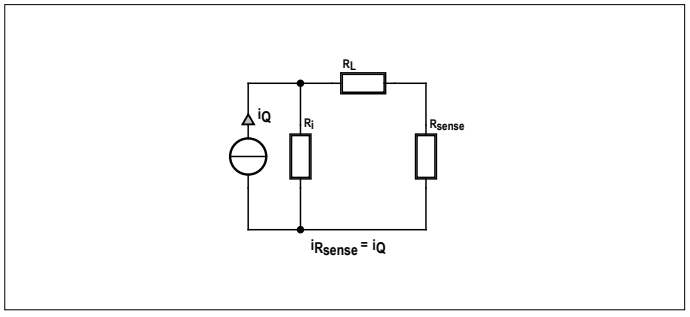


Figure 1. Current sources are not resistance-dependent.

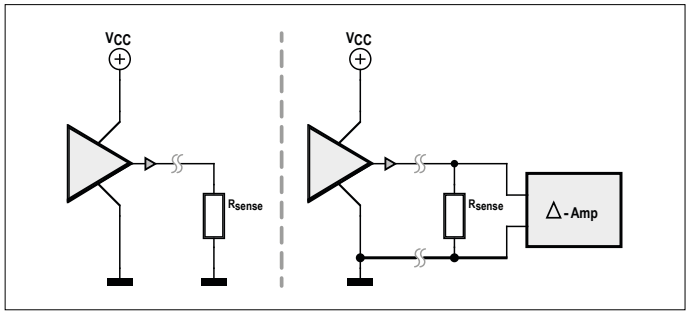


Figure 2. Multiple ground connections are not allowed under any circumstances.

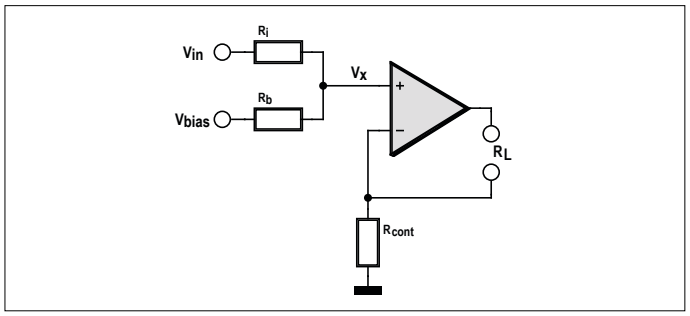


Figure 3. Dimensioning the input resistor is not particularly critical.

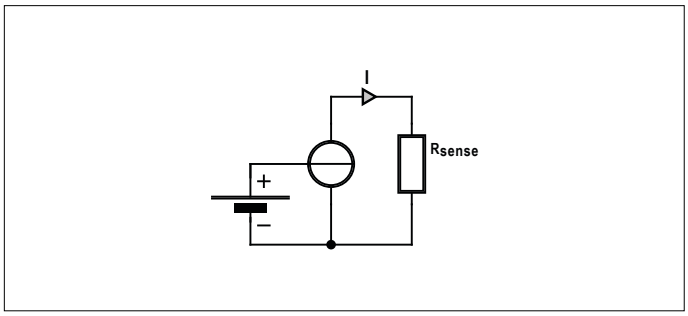


Figure 4. Primitive sensors draw their power from outside of the current loop.

equipped to handle differential signals, then these devices are definitely to be preferred. Bear in mind also that not all sensors require a power supply. There are also sensors that modulate an external power source.

A very interesting circuit can be found in a book by J. Michael Jacob [1]. This combination in **Figure 3** on the basis of an adder/incrementor provides a very effective means of converting the measurement voltage into a current output *and* of registering the 4 mA offset. As well as the voltage V_{in} to be processed we also need the bias voltage V_{bias} that ensures that the circuit generates the minimum current of around 4 mA.

Since the use of those operational amplifier circuits from the days of process computers is no longer the norm and therefore now considered 'alien', we need to do the dimensioning at least partially 'by hand'. First, we set R_i and R_b more or less arbitrarily, at 680 kΩ for each for the sake of simplicity.

Next we consider the two extreme cases, at V_{in} = 5 V [corresponding to I_l = 20 mA] and V_{in} = 0 V [equivalent to I_l = 4 mA]. In both cases the voltage applied to the inverting input can be calculated according to $V = I_l \times R_{cont}$. Since an operational amplifier with negative feedback always attempts to set both inputs to the same potential, we can say for the two loops:

$$V_{in} - V_R - V_R - V_b = 0$$
$$V_{in} - V_R - V_x = 0$$

As the inputs of an operational amplifier cannot draw any current, we can assume that the current through the two resistors is constant.

$$V_{in} - V_b = 2 \times V_R$$

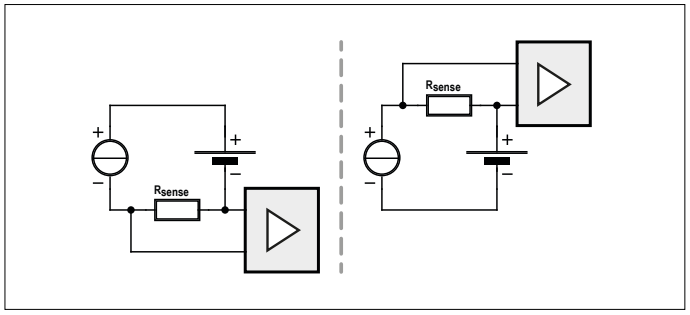


Figure 5. Two connections to a differential amplifier suffice simultaneously for power supply and for the measurement function.

$$V_{in} - V_b = 2 \times I_{in} \times R$$
$$I_{in} = (V_{in} - V_b) / (2R)$$

Consequently we can rewrite the shorter loop as:

$$V_{in} - I_{in} \times R - V_x = 0$$
$$V_{in} - [(V_{in} - V_b) / (2R)]) \times R - V_x = 0$$
$$(V_{in} \times 2R) / 2R - [(V_{in} - V_b) / (2R)] \times R - V_x = 0$$
$$(V_{in} + V_b) / 2 = V_x$$

We can now use this result in $V_x = I_l \times R_{cont}$. Make the two extreme values equal in order to determine V_b and R_{cont}. Also note that the accuracy of the circuit (ideally) is determined exclusively by R_{cont}!

Sensor systems in practice

Anyone interested in the design and dimensioning of op-amp circuits for current loops will find a good 20 pages on this theme in [1]. The book is readily available second-hand at low prices. However, right now it's more important to see how developers implement 4–20 mA systems in sensor technology.

By far the most common configuration is shown in **Figure 4**. Here we have a powered sensor or the output of a control in 4–20 mA format. So long as it's permissible to ground the output, the task is simple: all you do is to place in the current loop a resistor R_{sense} that's within the permissible range and measure the voltage drop at that point.

The group of sensors that we in the English-speaking world classify as 'primitive' can be further divided into sensors that are self-powered from the signal under measurement and 'three-legged' sensors equipped with a third pin for the power supply. Shunt sensors differ from their simpler brethren; as shown in **Figure 5** they are provided with an in-circuit precision resistor. Used in combination with a differential amplifier, they can be used to gather data not referred to ground. Whether the shunt is then installed 'above' or 'below' is not of importance. The decision for and against a specific architecture is normally down to the sensor manufacturer. If you want to capture a specific magnitude, you should always keep the various topologies in mind when choosing the sensor.

Ready-to-use ICs

An op-amp containing a few resistors and a transistor integrated onto a silicon substrate will always end up costing more the sum of the separate components, simply on account of its

more compact footprint. Of course this makes good business for the manufacturers of operational amplifiers and sensors with 4–20 mA current loops.

The AD5749 current driver from Analog Devices is a predominantly analogue component (other manufacturers offer similar products) and its basic structure can be seen in **Figure 6**. It differs from other components, in which a particular current range is preset, by letting users program the amperage using SPI (Serial Peripheral Interface). This avoids the need to use a microcontroller for selecting the current range.

Receiving data is problematic to the extent that it's entirely possible that you might need to connect a sensor in a current loop with two or three actuators. If this is not the case, inserting a series resistor is normally sufficient. If the output power is insufficient, an operational or similar amplifier can be connected downstream.

There are many references in the literature to multistage analogue circuits that convert the value ranges from 4–20 mA into a voltage range of 0 to 'whatever' volts. In this author's opinion, this approach is advised only conditionally or not at all; far better to digitise the values first and then perform the scaling (and the error detection) in software! If you're looking for ICs to convert current loop readings into voltages, there's a comparatively meagre selection. Texas Instruments has the RCV420 (inherited from Burr-Brown), which converts 4–20 mA into the voltage range 0 to 5 V. However, the chip requires a bipolar power supply. The data sheet recommends ±15 V, which is far from commonplace in the industrial domain. Furthermore the component is fairly expensive, priced in one-off quantities at €7.50 / £6.50 / $12.

In their Application Note 54 [2] Vishay Semiconductors present a family of completely isolated applications using their IL300 optocoupler IL300, which without exception are fairly complex.

Using safety measures and your brain

The folk at Maxim Integrated hold back on things like 4–20 mA sensors: they have an expensive temperature sensor and that's more or less it. Instead, they rely on various add-on parts for providing current loops with additional features.

One exception is their MAX14626, which is specially equipped for protecting current loops. This component, costing a mere €0.60 / £0.50 / $0.70 in 1,000-off quantities, is placed in the current loop and requires only a connection to ground. The protection lies in the fact that this component switches off at any output current of 30 mA or above. If you link it thermally to the sensor, you get the bonus of limiting the operating temperature to 150 °C (**Figure 7**). The snag is that the MAX14626 is supplied only in a TDFN (Thin Dual Flat No-leads) package, which was not designed for human manipulation.

Almost the same applies to the DS8500 from Dallas Semiconductor (now Maxim), which is offered only in the TQFN (Thin Quad Flat No-leads) package. This happens to be a transceiver for HART technology (Highway Addressable Remote Transducer), which permits current loops to have slow data-rate digital modems superimposed on them. HART is based on FSK from a technical perspective and operates by superimposing sinewave signals (**Figure 8**). Modems of this type demonstrate how modest sampling rates can sometimes work out to be the recommended option — mind you, anyone registering data at 20 kHz with HART would have to contend with any amount of noise.

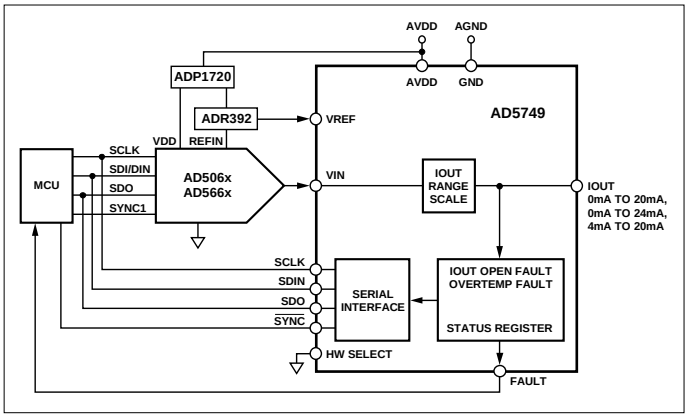


Figure 6. Just add an A-to-D converter and you have a complete controller-driven current loop (image: Analog Devices).

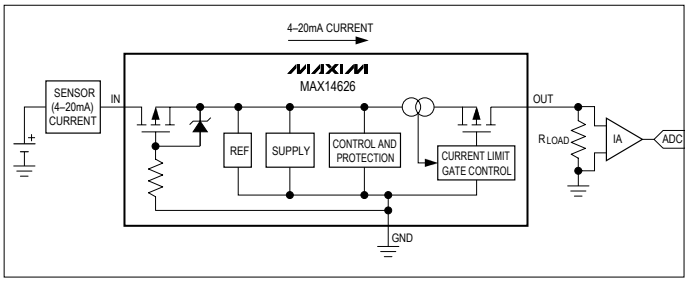


Figure 7. The MAX14626 protects both sensor and receiver alike (image: Maxim).

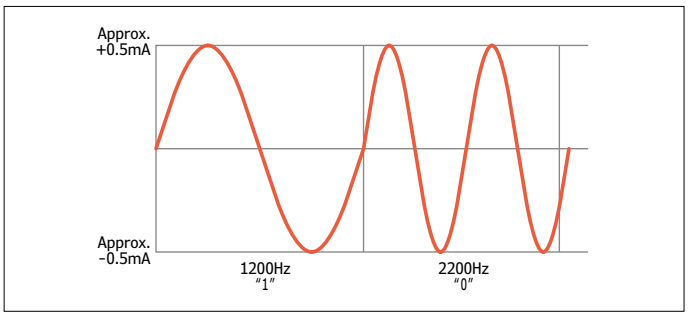


Figure 8. HART operates by superimposing sinewaves (image: Maxim)

And finally

A sensor current loop is definitely not applicable to every system. Nevertheless, in harsh industrial environments it can sometimes prove to be the optimum solution. ⏮

(160443)

Web Links and Literature

[1] J. Michael Jacob: Industrial Control Electronics: Applications and Design, 1988, ISBN 9780134593067

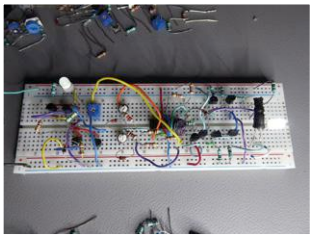
[2] www.vishay.com/docs/83710/appn54.pdf

Elektor Labs Pipeline

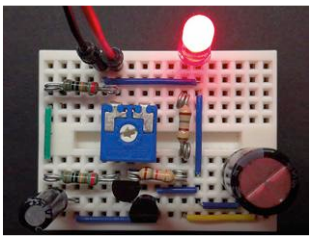
Our little Elektor Labs contest to design a circuit to fade an LED in and out without using a microcontroller produced a surprising number of entries. “Well done!” to all.

The winner just couldn’t stop

Roel Arits emerged as the Uncrowned King of Fading LEDs by posting various designs covering all sorts of techniques of fading an LED. His initial designs used an LDR and current mirrors. However, he kept going and came up with more circuits even after the contest was over. Logarithmic fading, sinusoidal fading and even tangential oscillators got explored through his designs. Indeed, no stone was left unturned as shown by his latest effort employing a motorized potentiometer.



https://goo.gl/rDWTLz



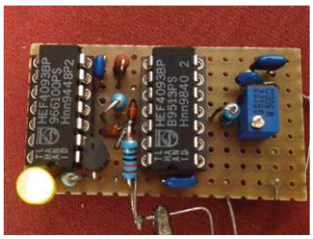
https://goo.gl/cPajhh

Put a PUT in it

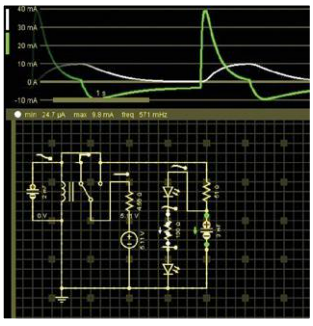
Contestant Anto won the Extra Prize with a simple but effective discrete programmable unijunction transistor (PUT) oscillator. We love PUTs and we would like to see more circuits using them (the 2N2646 seems still to be readily available). To avoid the problem of buying a PUT, Anto fabricated the PUT from two plain vanilla transistors.

Do it with a beat frequency

Most contest participants used some sort of frequency generator oscillating at the LED fading rate, but not contestant Arnoldus. Off the beaten track he used two slightly detuned square-wave oscillators to create a 0.5-Hz difference frequency to control the LED, similar to a radio technique known as beat-heterodyning. The beat signal obtained from the two square waves is a kind of pulsewidth modulated (PWM) signal. Due to the linear modulation, the fading is linear too and looks less natural than the logarithmic method.



https://goo.gl/awWMCi



https://goo.gl/1tMejd

Use a relay

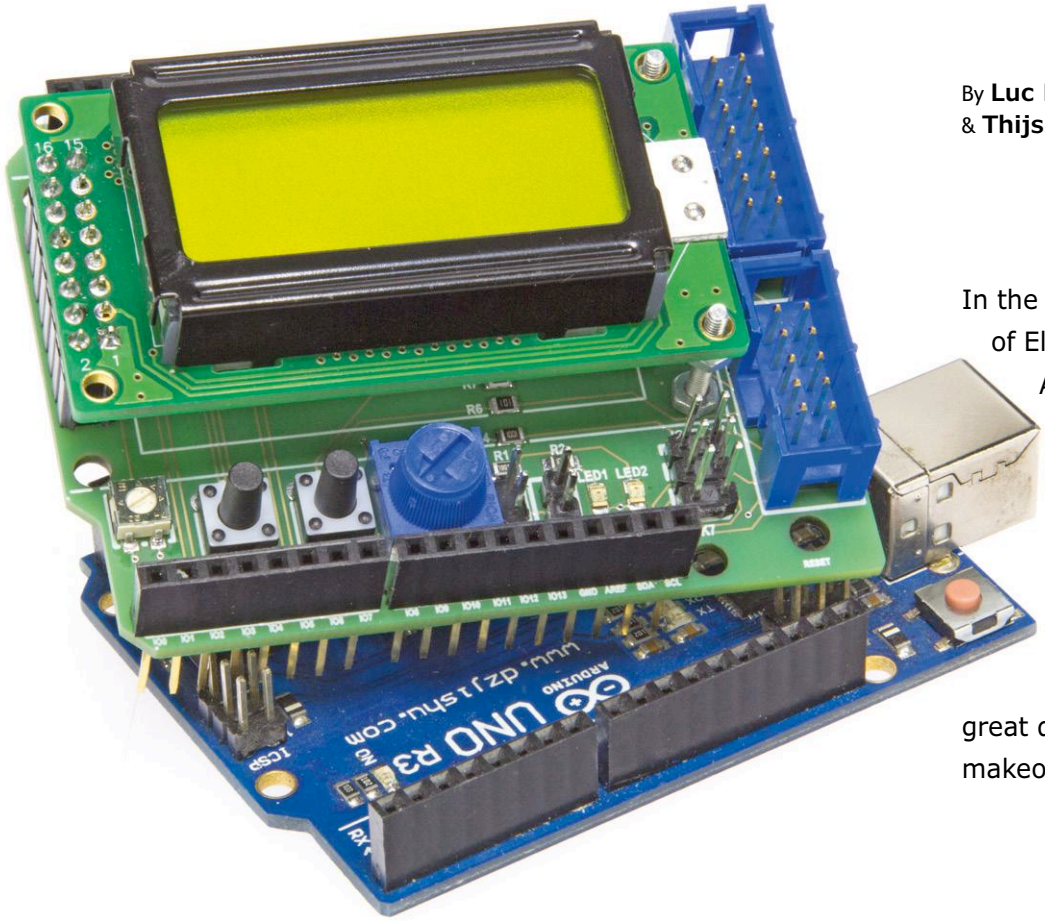
Totally used to solid-state semiconductors we often forget that switching was often done with electromechanical parts before the semis came into existence. Heribert Houben took us back to the basics and drew up an electromechanical (!) LED fade-in-fade-out circuit with only two semiconductors: the LEDs. The circuit is a relay oscillator, well known from mechanical buzzers and doorbells, but with a large capacitor added to create slopes rather than steep edges.

What you should know about Elektor Labs contests

Elektor Labs contest happen whenever we feel like running one. Contest rules are published on the contest’s main page found on the Elektor Labs website, there the entries are filed and the winners are announced also.
The Fading LED contest main page has a list of all entries received: https://goo.gl/JWo8oC (160558)

Arduino Experimenting Shield 2.0

Same functionality, better display options



By **Luc Lemmens** (Elektor Labs)
& **Thijs Beckers** (Elektor Netherlands)

In the 2014 July/August issue of Elektor we wrote about our Arduino Extension Shield [1] that, in particular for the Arduino Uno, provided additional interface options, LEDs, pushbuttons and an LCD. The shield enjoyed — and continues to enjoy — a great deal of interest. Time for a makeover!

The Arduino Experimenting Shield 2.0 offers largely the same features as its predecessor, but, after 3.5 years, quickly revisiting the specific details can do no harm.

Characteristics

- 2×8 character display
- 2 pushbuttons
- 1 potentiometer
- 2 indicator LEDs
- Arduino expansion headers
- ICSP header
- Expansion headers for ECC- and EEC bus

Schematic

When we look at the schematic in **Figure 1**, we see that the shield has to offer two freely available LEDs, two pushbuttons, a potentiometer and a 2×8-character LC display. In addition there are two connectors that allow further expansion boards to be connected using a ribbon cable; examples are the Wireless gateway [2] and the 16-bit ADC module [3]. This will all sound familiar to those acquainted with the 'old' Experimenting Shield.

New is the display and the connections to the display. The EA DIPS082 display chosen at the time has a somewhat exotic pinout. The new shield is suitable for use with any alphanumeric LCD with a standard 2×8-pin connector. As a consequence, the display choice is no longer limited to a two-line display with eight characters; it is now very easy to connect larger displays. Incidentally, we supply a standard 2x8-character LCD with the standard pre-built module [4], so that you can get started immediately.

Inputs and outputs

As is customary with many shields, the bus headers for the Arduino Uno on our Experimenting Shield are also repeated 1-to-1. Bus headers K3 through K6 serve this purpose. We therefore could not move these connectors when designing

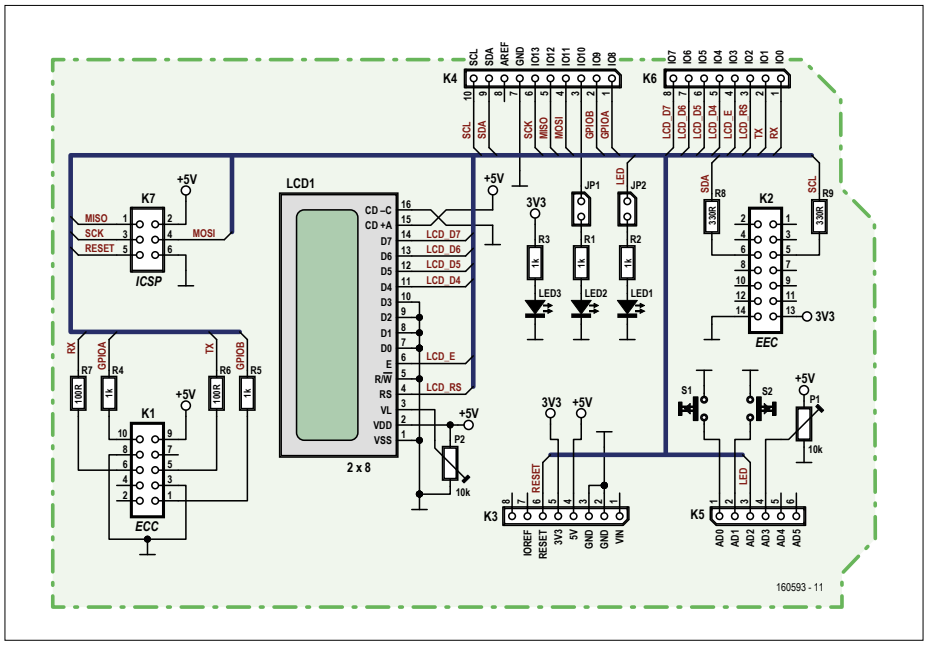


Figure 1. The schematic has barely changed compared to its predecessor. Only the connection to the LCD has been increased by two pins.

PROJECT INFORMATION

Arduino
Shield
LCD

entry level
intermediate level
expert level

30 minutes approx.

Soldering iron and solder

€ 40 / £ 45 / $55 approx.

Now any LCD with a standard 2x8-pin interface can be connected to the shield

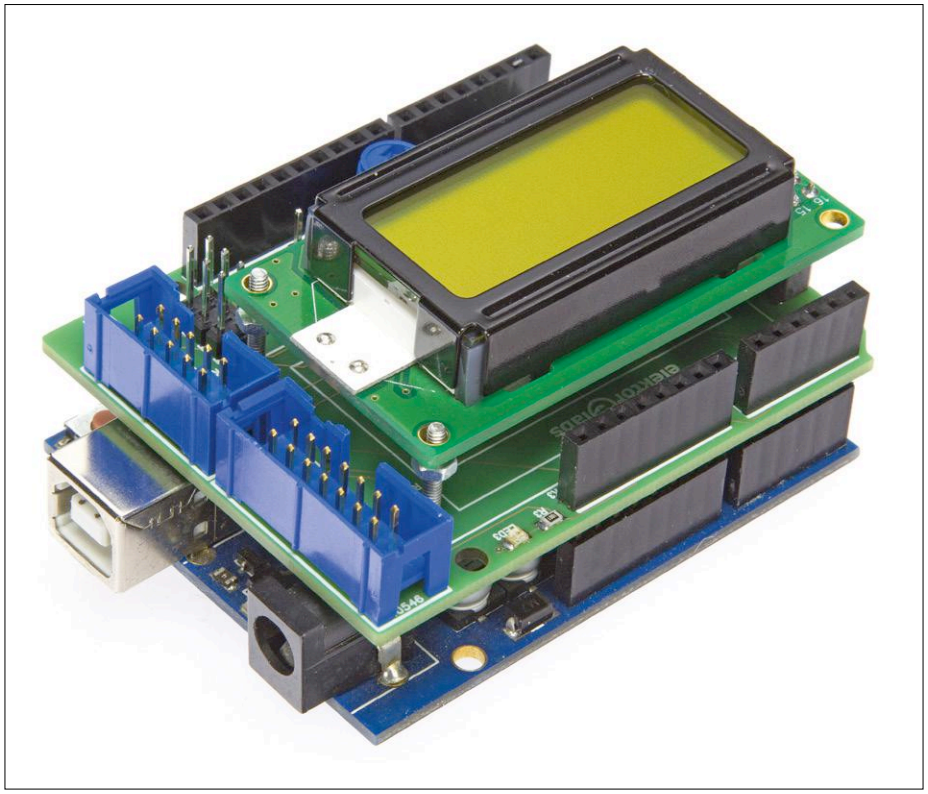


Figure 2. Take care when plugging in the shield: risk of short circuit.

the layout for the PCB. But practically every other component has been given a new position.

From the schematic we can see that the function of most of the Arduino pins is not affected by the Experimenting Shield. And when JP1, JP2 and the LCD are not fitted, all the pins are actually feely available. Do, of course, take note of the position of P1, which is connected to AD3. However, because of the LCD, pushbuttons and potentiometer P1, it is not really practical to plug another shield on top of the Experimenting Shield. But this implementation does however provide very easy access to the Arduino pins for breadboard wires.
The LEDs can be controlled via the AD2 (LED1) and IO10 connections (LED2). This requires that jumpers are placed on JP1 and JP2. LED3 indicates whether 3.3 V is present on the board. Pushbuttons S1 and S2 are connected to AD0 and AD1 respectively. These inputs can also be used as analogue inputs, so we have deliberately not implemented any pull-up resistors and debounce circuits. In the ATmega328 that is used in the Arduino Uno, internal pull-up resistors can be switching on via software.

Display

To drive the display we can use the library “LiquidCrystal.h” that is supplied with the Arduino IDE. If, for example, we

open the HelloWorld example sketch we only need to change the line

```
LiquidCrystal lcd(12, 11, 5, 4,
   3, 2);
```

into

```
LiquidCrystal lcd(2, 3, 4, 5, 6,
   7);
```

and change the line

```
lcd.begin(16, 2);
```

into

```
lcd.begin(8, 2);
```

Note

Take care when plugging the module onto an Arduino (Uno): to ensure that the size of the module stayed within the dimensions of the PCB for the Uno, we had to position the shrouded headers above the metal USB connector, see **Figure 2**. Make sure that you don't plug the module in too far, otherwise the pins of K1 could short-circuit to the USB connector. We have fitted the ready-built module with headers that have extra long pins for the connections to the Arduino (K3 through K6), so there should be no problems. Even safer would be to insulate the USB connector with a piece of sturdy tape or, even better, a piece of rigid plastic. ⏮

(160593)

Web Links

[1] www.elektormagazine.com/140009
[2] www.elektormagazine.com/130023
[3] www.elektormagazine.com/130485
[4] www.elektormagazine.com/160593

COMPONENT LIST

Resistors

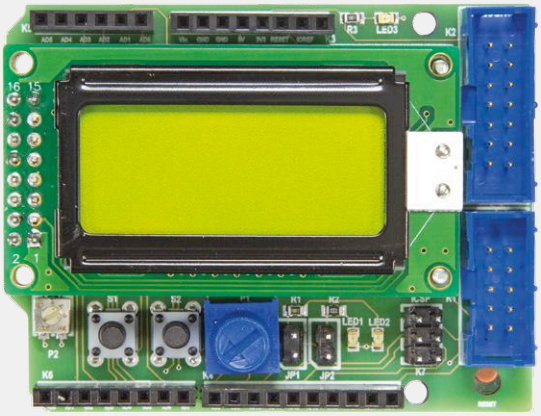
R1,R2,R3,R4,R5 = 1kΩ
R6,R6 = 100Ω
R8,R9 = 330Ω
P1 = 10kΩ trimpot with turning knob
P2 = 10kΩ trimpot (SMD, Vishay TS53YJ103MR10)

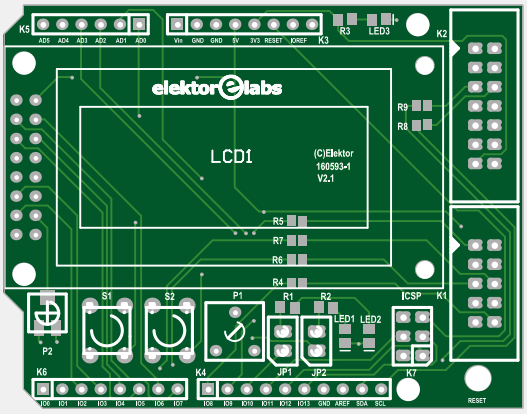
Semiconductors

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

Miscellaneous

S1,S2 = pushbutton
K1 = 10-pin (2x5) boxheader, 0.1″ pitch
K2 = 14-pin (2x7) boxheader, 0.1″ pitch
K3,K6 = 8-pin pinheader, single row*
K4 = 10-pin pinheader, single row*
K5 = 6-pin pinheader, single row*
K7 = 6-pin (2x3) pinheader, 0.1″ pitch
JP1,JP2 = 2-pin pin header, 0.1″ pitch, with jumper
LCD1 = LCD 2x8 characters, with backlight, TC0802B
LCD1 = 16-pin (2x8) pinheader, 0.1″ pitch
LCD1 = 16-way (2x8) socket for LCD1, 0.1″ pitch
LCD1 = 4 off M2.5x20 metal screws
LCD1 = 12 off M2.5 bolts





PCB no. 160593-1 v2.1 [4] or
Ready-assembled module no. 140009-91 [4]

* K3–K6 = Arduino-shield pinheader

FROM THE STORE

→160593-1
Bare printed circuit board

→160593-91
Read-built module

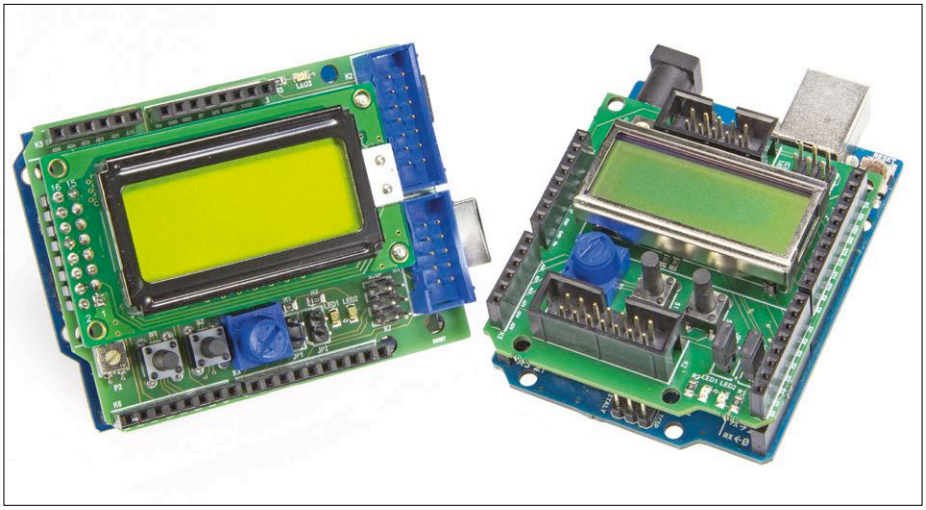


Figure 3. New (left) *versus* old (right).

ESP8266 on the Elektor Android I/O Board

Load new firmware yourself

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

The Android I/O Board makes it easy to control electronic devices from a smartphone or tablet. There is space on the board for a total of seven different wireless link modules, including the popular and inexpensive ESP8266. However, this requires loading different firmware into the module. In this article we give some background on the module and tell you how to load the firmware.

The ESP8266 is a very low-cost Wi-Fi module. In part 2 of the series on the Android I/O Board, published in Elektor 6/2015 (November & December), we said that once the firmware was more stable we would explain how the popular ESP8266 can be used on the Android I/O Board. Now the available firmware has matured to the extent of allowing the ESP8266 to be used with the Android I/O Board. Here we tell you how to do this.

ESP8266

The ESP8266 module has an Xtensa processor and can be used as a Wi-Fi access point (AP) or as a Wi-Fi client (station, STA). In AP mode the Android smartphone logs in directly to the Wi-Fi network of the ESP8266, and in STA mode the ESP8266 logs in to a Wi-Fi router. If the Android smartphone is logged in to the same router, it can connect to the module through the local network. An Android smartphone can also be used as an access point (this is called tethering), in which case the ESP8266 can log in to the Android smartphone in STA mode. However, in STA mode the ESP8266 needs to know the password of the network it wants to log in to. Later in this article we tell you how to set this up.

The ESP8266 is available in various PCB versions, each with its own type number. The Android I/O Board is designed for the ESP-01 version, which has a connector with two rows of four pins with 2.54 mm pitch (see **Figure 1**). The ESP-01 mates with connector MOD4 on the Android I/O Board. There are two LEDs on the ESP-01, which indicate when power is present and when data is being transmitted. The ESP-12 version is also popular and costs the same as the ESP-01. This version has more I/O pins fed out to the connector. The ESP-12 does not fit directly on the Android I/O Board. However, you can attach it to the board with a piece of double-sided sticky tape and use short wires to make the necessary connections, or you can make an adapter from a piece of prototyping board as shown in **Figure 2**.

With some ESP-12 models the GPIO15 pin must be connected to ground by a resistor with a value less than 1 kΩ, because this pin is sometimes used to detect the presence of an external SD card with start-up code. Incidentally, the ESP-12 module used by the author has only one LED, which remains dark even though the module works normally.

The ESP8266 is also available with different memory sizes. The older ESP8266 modules (usually blue) have 512 KB (4 Mb) of memory; the newer ESP8266 modules (usually black) have 1 MB (8 Mb). For the apps on the Android I/O Board you only need 512 KB, but you can also use the 1 MB version.

The ESP8266 produces a lot of noise on the supply line. The standard Android I/O Board does not have enough capacitance to ensure reliable operation of the module. An additional capacitor can be fitted between pin 2 (+) and pin 3 (–) of the MOD6 connector (for the six-pin HC-06 module), as shown in **Figure 3**. A 100 µF capacitor is more than sufficient.

ESP8266 firmware

A disadvantage of the standard firmware in the ESP8266 is that unlike the RN-141 and all other modules that fit on the Android I/O Board, it does not pass the data transparently. The firmware uses the AT commands dating from the days of analog phone modems. For example, to transmit data through the serial port you first have to send it an AT command that says how many characters will be transmitted. The baud rate of the serial port is also set to 115,200 by default. This is not compatible with the Android I/O Board, which operates at 9,600 baud.

However, the nice thing about the ESP8266 is that a lot of alternative

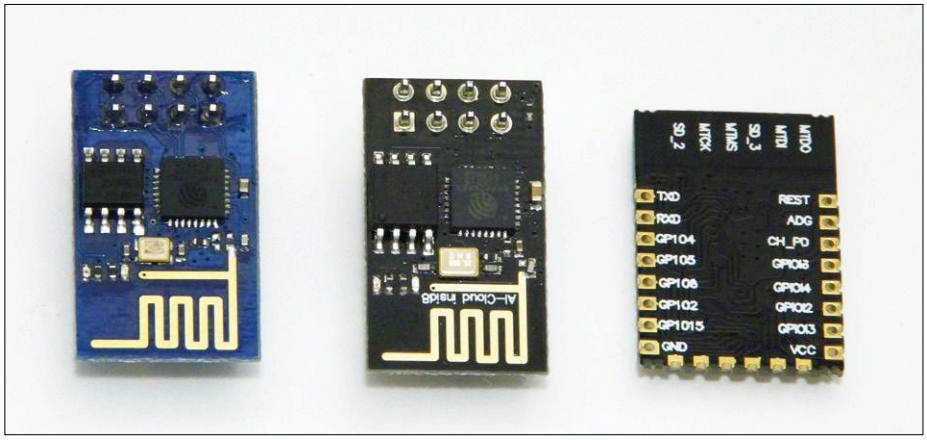


Figure 1. Three versions of the ESP8266, from left to right: ESP-01 layout with 512 KB, ESP-01 layout with 1 MB, and the rear of the ESP-12E layout with 1 MB.

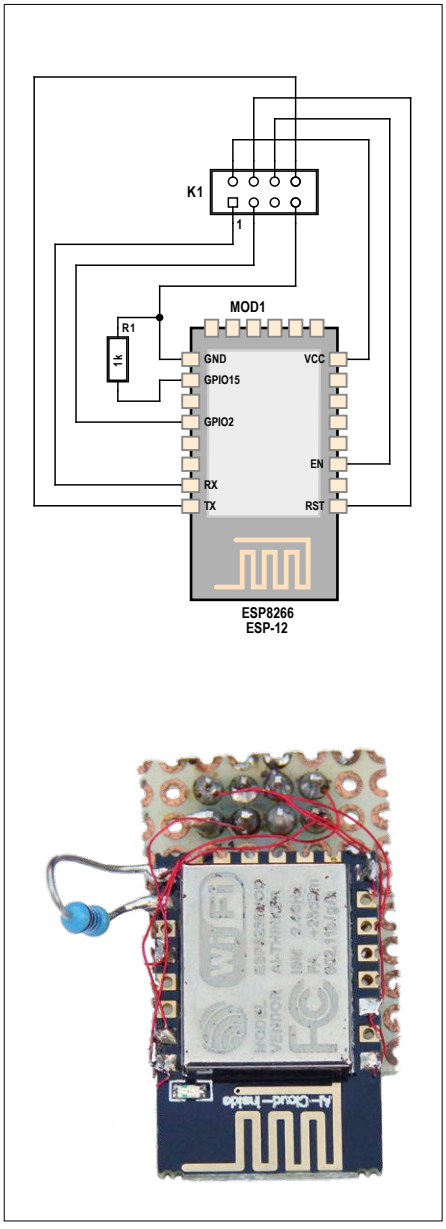


Figure 2. Schematic of a converter for ESP8266-ESP-12 to ESP8266-ESP-01.

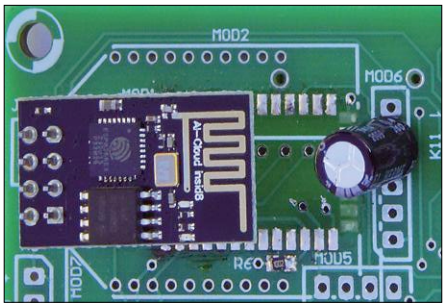


Figure 3. The additional capacitor wired to connector MOD6.

firmware versions are available for it. Some of these versions allow transparent serial data links to be set up. The author opted for the Beckdac firmware [1], which appears to be stable and is free from unnecessary bells and whistles.

If you think flashing the firmware is too much work or too difficult, you can order a preprogrammed ESP8266 from the Elektor STORE [2].

Standard AP and STA PM

You can send +++AT commands over the Wi-Fi link to configure the module. A complete list of the AT commands is available at [3]. The most important commands are:

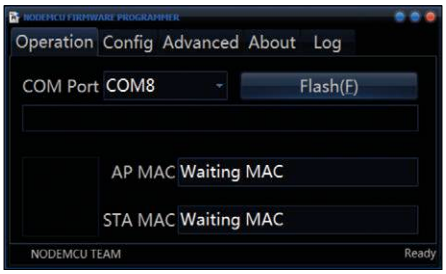
`+++AT BAUD baud_rate` — set the baud rate of the serial port. For the Android I/O Board: +++AT BAUD 9600. If you enter this command without any parameters (this applies to all of the commands), the configured value is returned. For example, "+++AT PORT" returns "BAUD=9600 8 N 1".

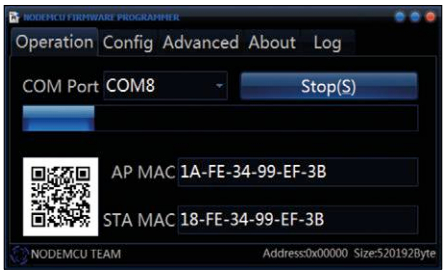
`+++AT MODE mode_number` — with mode number 1 the ESP8266 is in STA mode, with mode number 2 it is in STA mode, and with mode number 3 it is in combined AP and STA mode.

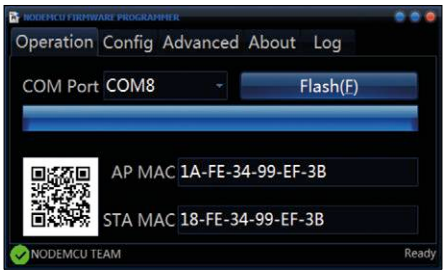
`+++AT STA SSID password` — configure the SSID and password of the router where the ESP should log in.

`+++AT AP SSID password_authentication_method` — configure the parameters of the ESP8266 Wi-Fi network. Authentication method 2 is WPA; method 3 is WPA2. The command "+++AT SSID password 3" protects the ESP8266 against login by unauthorized parties.

Incidentally, you can log in with several Android devices and/or PCs at the same time in either STA or AP mode. In that situation the ESP8266 combines all the connections and sends the results to all connected devices at the same time.







Figures 4–7. Screenshots of the esp8266flasher programming tool during flashing.

Table 1. Structure of the beckdac_4mbit.bin and beckdac_8mbit.bin files.

Adres	Bin-file	Bron
0x00000	0x00000.bin	Beckdac [1]
0x3E000	blank.bin	Espressif [4]
0x40000	0x40000.bin	Beckdac [1]
0x7E000	blank.bin	Espressif [4]
Extra for 8 Mbit (1MB) version		
0x80000	0x40000.bin	Beckdac [1]
0xFE000	blank.bin	Espressif [4]

Flashing the ESP8266

The firmware consists of several parts that must be programmed from separate initial memory addresses. This is complicated, and some programmers cannot handle it properly. The author suspects that as a result data is sometimes left in the memory areas between the programmed portions and interferes with the operation of the ESP8266. For this reason, the author created a single large binary file with the intermediate memory areas cleared.

Table 1 shows the structure of this file, which consists of a combination of Beckdac files and the original program code [4]. The files can be downloaded from the Elektor website [2]. You should use either the beckdac_4mbit.bin file or the beckdac_8mbit.bin file, depending on whether you have a 512 KB module (blue) or a 1 MB module (black), although in theory the 4 Mbit file is good enough for the 1 MB version.

Various programming tools are available for the ESP8266. The author tried three of them: FLASH_DOWNLOAD_TOOLS [5], esp8266_flasher [6], and esp8266flasher [7] (the only difference between the latter two names is the underscore). Although the author was able to load the firmware correctly with all three tools, the most reliable results were obtained with the last one together with the previously mentioned combined binary file.

Flash instructions

The basic procedure for programming the module consists of the following steps:

- Preparation: download the software and install the software and hardware.

- Optional: test the ESP8266 (before flashing).
- Flash the new firmware.
- Configure the appropriate settings in the ESP8266.
- Test.

1. Preparation

Download the firmware from [2] and unpack the zip file.
Download esp8266flasher.exe [7].
Obtain a USB/serial adapter module for 3.3 V (for example, the Elektor FT232R USB/Serial Bridge/BoB) and install any necessary drivers. Build a connector that connects the USB/serial module to the left side of JP2 on the Android I/O Board. Use only the two data lines and the negative supply line (but not the positive supply line for the module). Provide a separate power supply for the Android I/O board, because most USB/serial modules cannot deliver enough current.

2. Test the ESP8266 (before flashing)

Plug the ESP8266 into the Android I/O board and connect the external power supply and the USB/serial module to the corresponding connectors on the Android I/O Board

The red LED on the ESP8266 lights up when the supply voltage is present. The blue LED blinks quickly a few times during start-up.
On a PC, launch a terminal emulator program (for example, Teraterm) and configure a COM port for the USB/serial adapter with a baud rate (for most ESP8266 modules) of 115,200.

If you type "AT", you will receive "Error" in response. This is okay.

Look for Wi-Fi networks in your vicinity. There are various ways to find a new Wi-Fi network (depending on the supplier), such as "AI-THINKER_xxxxx" where the last six characters are the hexadecimal value of the last three bytes of the MAC address.
This tells you that the module is working.

3. Flash the ESP8266

Fit jumper JP1 on the Android I/O Board to put the ESP8266 in flash mode. If you use a resistor for this, chose one with a resistance less than 1 kΩ.

Connect the USB/serial module to JP2, and then connect the power supply to the Android I/O Board.
Run esp8266flasher.exe.
Touch the "Config" tab.
Touch the gear icon, load the beckdac_4mbit.bin or beckdac_8mbit.bin file, set the offset to 0x00000, and tick the checkbox in front of the file name (see **Figure 4**).

On the "Operation" tab, select the correct COM port if it is not already selected (**Figure 5**).
Click "Flash".

The MAC addresses should appear within a few seconds (**Figure 6**). If they do not, restart the Android I/O Board by briefly disconnecting power.
Flashing the firmware takes 90 seconds with the 4 MB version and 3 minutes with the 8 MB version. Wait until you see the "Ready" message (**Figure 7**).
Now the flashing is done. Remove jumper JP1 and disconnect the USB/serial module, replace the jumpers on JP2, and reset the Android I/O Board by briefly disconnecting power.

4. Configuration

Set the Wi-Fi network to ESP_xxxxxx.
Establish a Telnet connection (using Teraterm, for example) to 192.168.4.1 (port 23).

Type:

- `+++AT FLASH 1`
- `+++AT PORT 9600`
- `+++AT MODE 3`

Type (with the SSID and password of your local Wi-Fi network):

- `+++AT STA SSID password`

Type (with an SSID and password of your choice):

- `+++AT AP SSID password 3`

The ESP8266 will execute a reset and set up a new Wi-Fi network. Wait until the ESP8266 has restarted.

Now the ESP8266 is ready for use.

If you want to use the ESP8266 exclusively in STA or AP mode, set either MODE 1 or MODE 2 in the above instructions according to the desired mode. In this case you do not have to configure AP or STA.

5. Test

Using the login data you previously configured with the +++AT AP commands, you can log in to the Wi-Fi network of the ESP8266 from your Android smartphone or tablet.

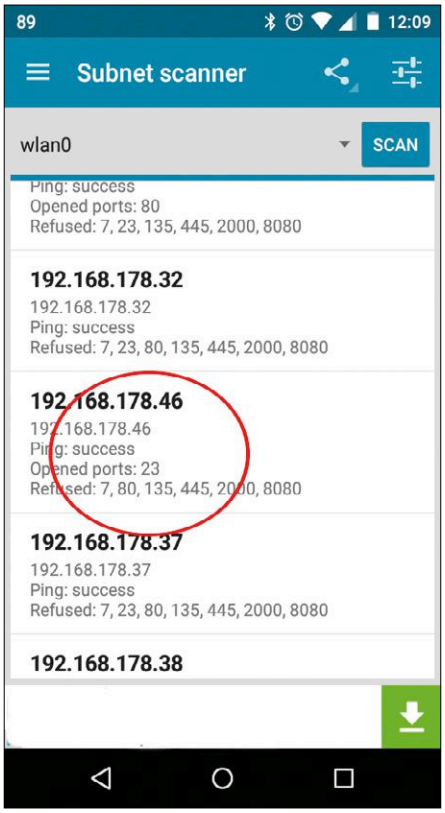


Figure 8. Subnet scan on port 23 to track down the ESP8266 on the network.

If the ESP8266 is within range of your local Wi-Fi network, it will log in to the network. You can use a tool, such as the Ping Tools app [8], to look for the IP address by scanning the network for port 23 (see **Figure 8**).

Now you can test the Android I/O Board, for example by running the demo app or sending a command (such as "G Z") with a Telnet app [9].

Things to check if it doesn't work

Everything should be okay if you follow the above instructions. However, if it doesn't work you should check the points listed below.

The ESP8266 produces a lot of noise on the supply line. Good noise suppression is essential. Fit the additional capacitor. The ESP8266 draws high peak currents (200 to 300 mA). Most FT232 modules (including the Elektor BoB) cannot supply

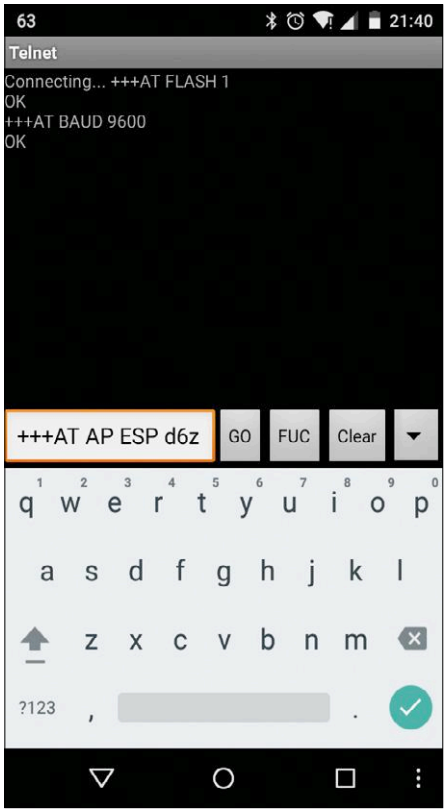


Figure 9. Configuring the ESP8266 with Simple Telnet.

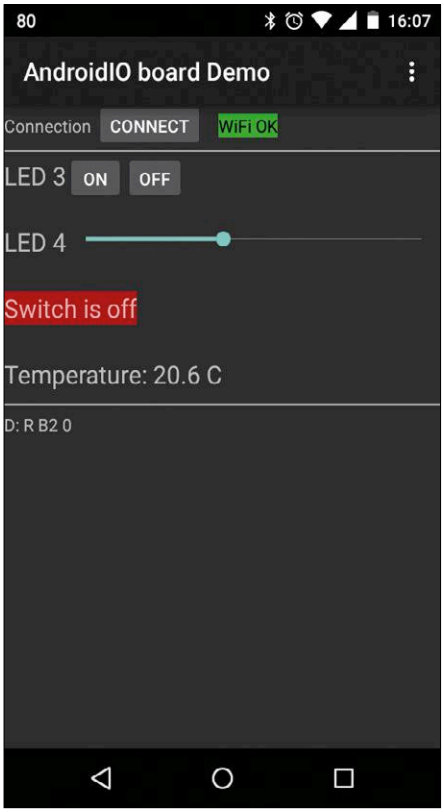


Figure 10. Demo app for the Android I/O Board with the ESP8266.

this much current. This means you have to use an external power supply when flashing the firmware.

Be patient. The author once spent two hours trying to load the firmware into the ESP8266. Each time there was a Wi-Fi signal, but the module did not respond to the outside world. The author decided to call it quits for the day. The next day he switched on the power and everything worked fine. The exact reason for this remains a mystery (the module does not have any large capacitors), but it could be that the USB/serial module can supply some current to the module over the data lines, preventing it from resetting properly. The lesson from this is: leave the module powered off for a while, and maybe it will reset by itself.

It is not possible to verify that the firmware has been flashed correctly. Errors can occur if the parameter settings for flashing are set incorrectly. You can manually configure the parameters with Flash_download_tools. The safest option is to configure writing at 20 MHz with DIO.
Logging in to the module on the ESP_xxxxxx Wi-Fi network often does not work under Windows 8.1. It usually works properly with an Android smartphone or tablet, or a PC with Linux, although it sometimes takes a minute or so to log in. The author therefore used Simple Telnet [9] from the Play Store to do the configuration (see **Figure 9**).

It looks like the Telnet connection is not established (you see the "Connecting..." message), but when you start typing the characters reach their destination properly.

The +++AT commands only work over the Wi-Fi link (through the Telnet connection), not through the serial port of the ESP8266.
Don't forget to set the baud rate to 9,600.

Demo app

Several apps for specific purposes are described in the series of articles on the Android I/O Board. A simple app that enables you to use the LEDs, NTC thermistor and touch button is provided with this project. **Figure 10** shows a screenshot of this app. LED 3 on the Android I/O Board can be switched on or off, and LED 4 can be dimmed over the range of 100% to 0%. A change in the state of the touch switch is passed directly to the app and displayed. Finally, the temperature of the NTC sensor is read out every second. As with other apps, in the Settings menu you can configure the communications module for Bluetooth, Wi-Fi, USB Accessory or USB Host. The APK file of this app and the source code can be downloaded from [2]. ⏮

(150703-I)

Web Links

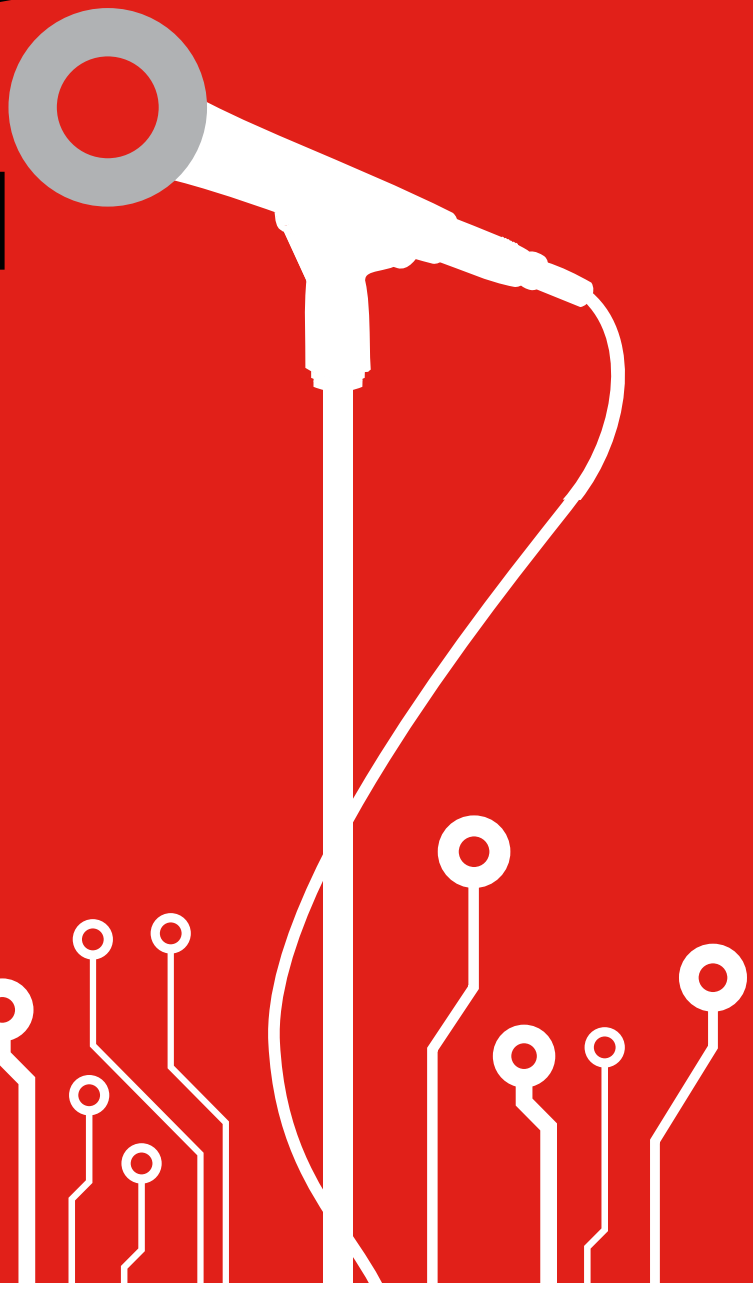
[1] https://github.com/beckdac/ESP8266-transparent-bridge
[2] www.elektormagazine.com/150703
[3] README.md at https://github.com/beckdac/ESP8266-transparent-bridge
[4] https://github.com/espressif/esp8266_at/tree/master/bin
[5] FLASH_DOWNLOAD_TOOLS_v1.2_150512.zip: https://drive.google.com/file/d/0B_ctPy0pJuW6V2ViNDR0NGdnYTQ/view?pli=1
[6] esp8266_flasher: https://drive.google.com/file/d/0B3dUKfqzZnlwVGc1YnFyUjgxelE/view?usp=sharing
https://github.com/Stadslab/ESP8266_example/tree/master/ESP8266_flasher_V00170901_00_Cloud%20Update%20Ready
[7] esp8266_flasher: https://github.com/nodemcu/nodemcu-flasher
[8] Ping Tools: https://play.google.com/store/apps/details?id=ua.com.streamsoft.pingtools
[9] Simple Telnet: https://play.google.com/store/apps/details?id=com.telnet

electronica
fast forward
powered by elektor

the startup platform

LAUNCH YOUR PRODUCT

ONTO THE
INTERNATIONAL MARKET PLACE!



Participate in 2018

November 13-16. 2018
Munich

For more information:
www.elektormagazine.com/e-ffwd

The electronica fast forward awards are brought to you by

electronica elektor

Skip!

Wireless 'Next Track' button for the Media Player

By **Luc Lemmens** (Elektor Labs)

It's always a bonus when you realize you can use part of a previous Elektor project to solve a completely different problem. In this instance we use the transmitter design from the recent 'Wireless Quizbuttons' project in combination with some clever Hoodloader software for the Arduino to provide a remote control function for a media player running on a PC.

Figure 1. Skip! You won't miss this button!

At home we ripped all of our audio CDs to one mighty NAS which together with Windows Media Player or the VLC player (still can't decide which one we prefer) we play in a random sequence as background music in the workplace. This usually works fine but sometimes the random nature of the selection can generate a jarring clash between two different music genres which can be quite disturbing. At this point you interrupt your work on the computer and select 'Next Track' from the media player options. Sometimes you're in the middle of something important and by the time you get to skip the song it's near the end anyway and all that trouble was in vain. A remote controller is an option but now you need to find the controller and as you know it's not always obvious where you left it. Sometimes in the rush, you can end up pressing the wrong button.

Wireless Arduino

We came up with a solution to the problem in the form of an emergency stop button (**Figure 1**), modified with the addition of a small ATtiny85 board to provide a remote 'skip' signal to the media player. This type of button is common in industrial environments and you can often pick them up cheaply at second-hand flea markets although they are quite expensive new.

The initial design used a cable between the button and the PC and I always promised that when time allowed I would develop the idea further and replace the cable with a wireless link. The opportunity presented itself during the time I was working on the 'Wireless Quiz button' [1] project in the Elektor Labs. It uses a battery powered transceiver board fitted with the nRF24L01 radio module which is also supported by easy-to-use Arduino library routines.

This quiz button has everything I need to generate the skip signal so I simply 'repurposed' the design. No changes necessary to either the hardware or software. The transmitter just sends the text string 'RED' over a so-called 'pipe' (one digital channel of the nRF24L01) whenever the button is pushed. Afterwards the controller reverts to the low-power sleep mode to conserve the battery.

The receiver listens on this channel and whenever the 'R' is detected it triggers the action. The only thing missing is a link between the receiver module and the USB-HID (Human Interface Device), to tell the media player to jump to the next title. Around this time I also discovered Nico Hood's 'Hoodloader' Arduino project on the Internet. This seemed worth exploring and could be the missing link. For my first experiments with the system I used an Arduino Uno fitted with an ELPB-NG-based prototyping shield. Not the most compact or prettiest setup but development was both quick and fun and it made good use of an Arduino Uno that would otherwise be gathering dust on a shelf in the lab.

The second heart

There's not too much to say about the Arduino Uno, it's been around since 2005 and there have been numerous successors and improvements. The basic configuration has a USB port with a dedicated processor intended for use only for programming the main processor, providing power to the board and as a UART interface to a computer; not for any real USB-based applications. Few users really pay any attention to this second processor chip on the board, it's there as a dedicated UART-USB bridge between the ATmega328 (and its bootloader) and the host Windows/Linux/Apple computer.

On most Arduino Unos this chip is an ATmega16U2 and with the aid of a special program such as Atmel's 'Flip' software it can be programmed via USB and has its own ISP header on board which can be used to flash its firmware. There are other versions of this firmware available which give the chip additional (mostly HID) functionality. The disadvantage is that the user needs to reflash the original serial-USB firmware before a sketch can be uploaded to the 328 processor using the Arduino IDE. Nico Hood has developed the Hoodloader, which is a special bootloader for the 16U2, it contains a USB stack and be switched into 'normal Arduino mode'. The Hoodloader effectively allows the Arduino Uno to be used as a single board with two independent microcontrollers both of which can be programmed from the same IDE.

The ATmega328 functions as a (normal) I/O microcontroller, while the ATmega16U2 provides the USB functionality. Both can run their own sketches and communicate with each other over the two-wire UART serial communication link between them. That all sounds nice and simple but I must say it has given me some headaches working out how you deal with this expanded Arduino Uno in the Arduino IDE. I needed to implement a small protocol to allow both sketches to communicate. The sketch running on the ATmega328 listens to the nRF24L01 receiver module and sends a byte over the local serial interface to the 16U2, when it detects a valid signal generated when the wireless push button is pressed. The 16U2 in turn listens for this received byte and then sends a 'Next Track' USB command packet to the computer and the media player software. The Hoodloader project has a lot more capability on offer than is used in this simple application but this is never the less a good introduction to it. To find out more check out the Github page [2].

Petite and modest

The hardware requirements for the Arduino-based receiver are really small, as you can see from **Figure 2** and the parts list. Using an ELPB-NG prototyping board we can have the three components fitted in minutes. We only need to download the sketches from the Elektor project page [3], load them to the board using the Arduino-IDE and the Skip! project is done and dusted.

The development so far has been pretty straightforward, we've been able to play around with the Hoodloader and build a USB-HID without too much difficulty. To be honest, what we've ended up with is not exactly an elegant solution (see **Figure 3**). It's really wasteful to use a board with two processors for such a simple application. The ATmega16U2 is in fact more than capable of controlling the nRF24L01 on its own without the help of the ATmega328. The SPI pins for the 16U2 are also available on a separate ISP header on the Uno board along with a few solder pads which connect to additional I/Os of the 16U2. As the 16U2 alone is up to the job I designed a small board for this controller which together with the nRF24L01 module fits into a USB stick enclosure.

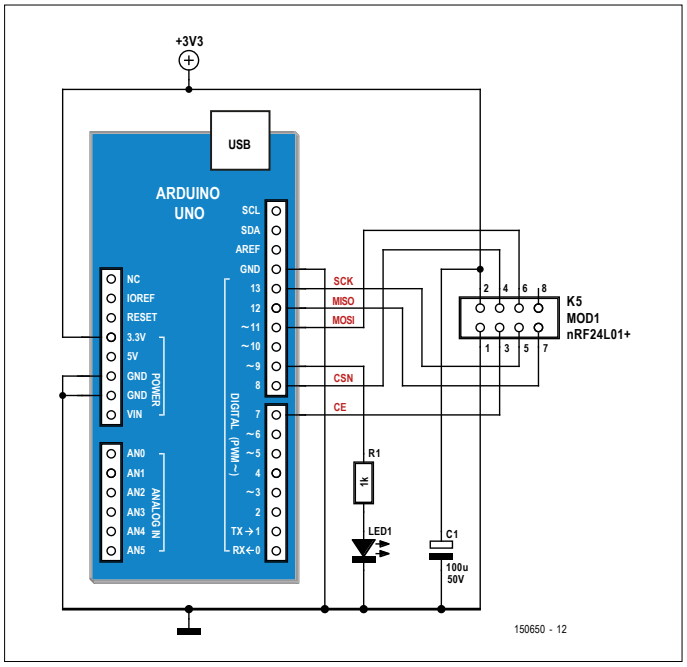


Figure 2. Components mounted on the ELPB-NG prototyping shield for the Arduino.

The schematic for this board (**Figure 4**) shows a striking resemblance to its configuration in the Arduino Uno schematic (never change a winning team!): the USB interface, the processor and oscillator, EMC (L1), overcurrent (F1) and ESD protection (D1, D2). Power is supplied from the USB interface and the +5 V connects to pin UVCC on the controller. No additional voltage regulator is necessary and the 16U2 includes an internal 3.3 V regulator to power the core. The output voltage from this internal regulator (UCAP) connects to the inputs at pins VCC and AVCC and also supplies the 3.3 V power to the nRF24L01 module. Note the 10 µF decoupling capacitor C5 on the sup-

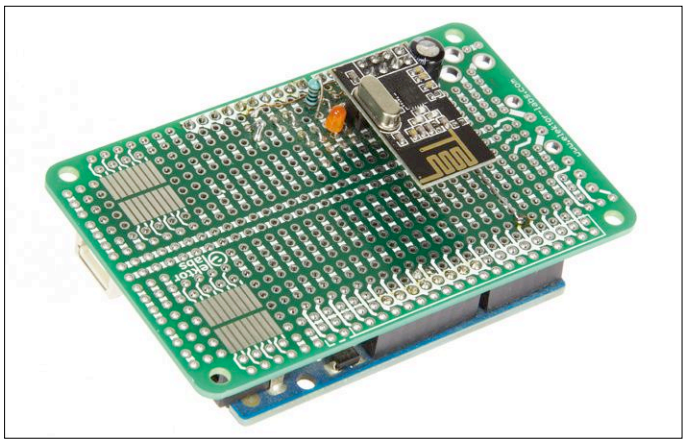


Figure 3. The Arduino Uno with shield fitted: functional… but not pretty.

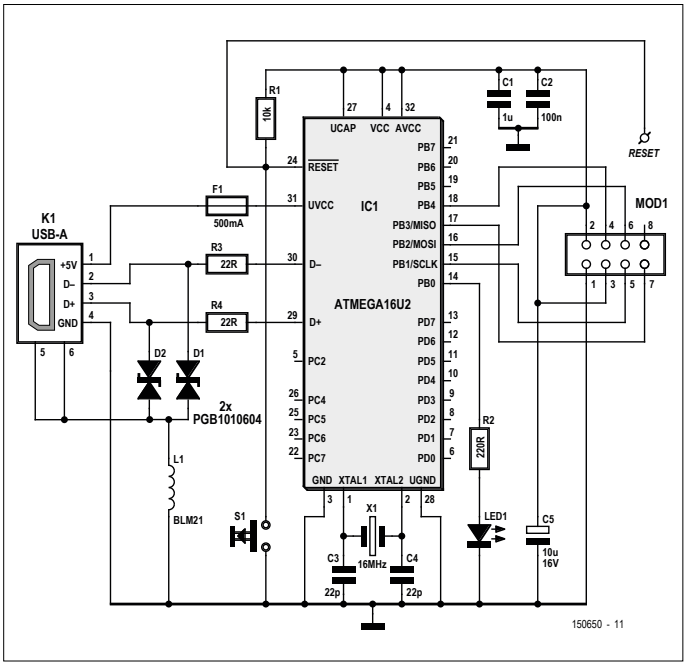


Figure 4. The USB-stick PCB schematic follows the original Arduino design.

ply to the radio module. This is essential for correct operation. With the current version of the sketch the LED will blink three

COMPONENT LIST
USB-STICK RECEIVER

Resistors

R1 = 10kΩ, 100mW, SMD 0603
R2 = 220Ω, 100mW, SMD 0603
R3,R4 = 22Ω, 100mW, SMD 0603

Capacitors

C1 = 1µF 25V, SMD 0603
C2 = 100nF 16V, SMD 0603
C3.C4 = 22pF 50V, SMD 0603 (C0G/NP0)
C5 = 10µF 16V, SMD 1206

Inductors:

L1 = ferrite bead, 0.4Ω, 200mA (Murata BLM21BD102SN1D)

Semiconductors

LED1 = LED, red, 3mm
D1,D2 = ESD suppressor diode (Littelfuse PGB1010603MR)
IC1 = ATmega16U2-AU, programmed, # 150499-41

Miscellaneous

F1 = 500 mA resetting PTC fuse, 15V (Bourns MF-MSMF050-2)
X1 = 16 MHz quartz crystal, 5x3.2mm
K1 = USB-A 2.0, right angled

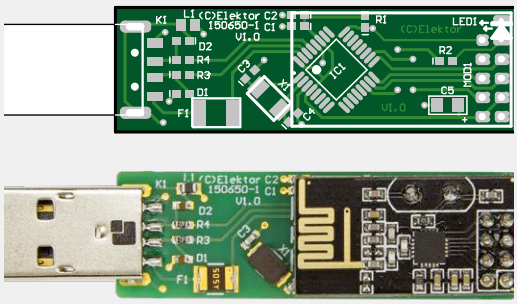


Figure 5. The tiny PCB for the radio receiver board.

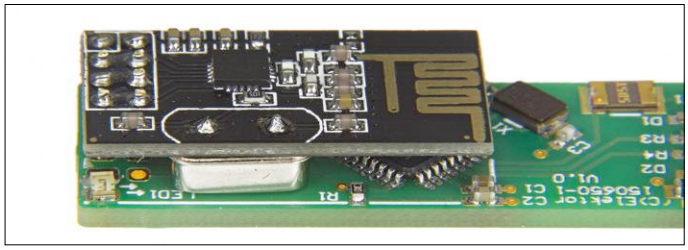


Figure 6. Swap the crystal to the other side of the board...

times when power is applied to the board to indicate that the application is running. After that it blinks once every time it receives a valid skip command from the button. That's all we need to control the media player. This solution has got to be neater than using the nRF24L01 plugged into a prototyping board, plugged onto an Arduino board. If you decide to buy the kit for this project (see box), the Hoodloader firmware is preprogrammed into the ATmega16U2. For a functioning Skip! button we just need a few lines of code in an Arduino sketch. Don't forget the original aim of this project was just to produce a big, obvious push button you could punch to advance the media player song selection onto the next title. This doesn't mean it has to stop there, you could modify the sketch to add controls such as mute or pause, triggered when the button is pressed for longer than, say one second. This is just a suggestion and you really can expand the system or use it for other applications as required. If you do develop another wireless ISB HID device with this tiny 16U2-Board let us know about it at **elektormagazine.com/labs**, we always look forward to your feedback.n...

Programming the Hoodloader

If you assemble the 16U2 board (**Figure 5**) from scratch you will likely be using a brand new 16U2 controller direct from the manufacturer which comes preprogrammed with a bootloader. This firmware allows users to program their own application via the USB connection using a PC programming tool such as Atmel's Flip. Then the problem is how to accommodate the Hoodloader into the controller. The same problem exists if it's necessary to update the Hoodloader or if for some reason you want some other firmware to run on the controller.
Unfortunately it's not possible to use the USB interface for the purpose of reprogramming the 16U2 firmware. For this we will need a parallel programming tool with a 32-pin ZIF IC socket (which you are unlikely to have lying around) or alternatively an AVRISP type of serial programming tool. This last option is considerably cheaper and the chances are you already have one in your home lab somewhere. However there isn't space available on this USB stick-sized board for the standard 6-way AVRISP header. Most of these connections are available at the MOD1 connector which connects the controller board to the RF module. The only connection missing is the RESET so an additional pad carrying the RESET signal is positioned next to the MOD1 connector. This allows you, for the purposes of programming, to fit a 10-way connector here instead of the 8-way connector and take the connections out to a prototyping board where the hook up to the AVRISP programmer can be made. Don't forget to use the correct fuse settings! The HoodLoader2 fuse values are:

Figure 7. ...so that it all fits in the USB-stick enclosure.

low_fuses=0xEF, high_fuses=0xD8 (boot from bootloader), extended_fuses=0xFC (no HWBE), unlock_bits=0x3F, lock_bits=0x0F.
The PCB is designed to fit in a Strapubox USB1 enclosure. The space available is a bit limited so it's necessary to take care assembling the components onto the board. First, all the components (except the radio module) are soldered to the board and then, when required, the Hoodloader firmware can be programmed. Here you will need to solder a pin header or a few leads to the board to provide connections to the AVRISP programmer. These wires will need to be removed once the programming is finished to make way for the nRF24L01 module so take care they can be easily unsoldered.

The 5 V power supply to the board comes from the USB connector K1 (i.e. it's powered from the host computer's USB port). Program the firmware, disconnect the programmer then unplug the USB cable and reconnect it. If all goes to plan Windows will now install the correct USB driver. Now in the Device Manager the stick will be recognized as a COM port with a name like 'Hoodloader' or something similar. If not, check the fuse settings and then reprogram the ATmega16U2 again.
The board now behaves as a standard Arduino Uno board (with the Hoodloader instead of the standard bootloader), and without an ATmega328 microcontroller on board. The board is recognized by the Arduino-IDE without any problem, just like our earlier version with the Arduino Uno. Now it's necessary to program just one sketch: Skip_button_6u2_only.ino (Contained in the 150650-11.zip download [3]). Unplug and reconnect the USB cable. The board is now recognized as an 'HID compliant device' in the Device Manager.
Now the Hoodloader and Skip! button sketch are running. Unplug K1 and remove all the wiring used to program the controller via the AVRISP programmer.

Crystal Repositioning

Take the nRF24L01 receiver module and carefully unsolder the (leaded) crystal and resolder it on the underside of the board as shown in **Figure 6**. Mount the receiver module on the component side of the 16U2 PCB, so that the crystal housing lies flat against the controller board. Make sure that both boards are now parallel to one another and as close together as possible before you solder connector MOD1. After soldering, use some sharp side cutters to remove any length of MOD1 pins protruding above the board surface. Finally you can fit the assembly into the Strapubox case.

Figure 7 shows that it all fits pretty neat!

(150650)

Project software:

free from www.elektormagazine.com/150650

Web Links

[1] 'Wireless Quiz Buttons RGB Style' Elektor 2/2016 (March & April), www.elektormagazine.com/150499

[2] https://github.com/NicoHood/HoodLoader2

[3] www.elektormagazine.com/150650

COMPONENT LIST TRANSMITTER

Resistors
Carbon film, 5%, 0.25W, 250V
R1 = 10kΩ
R2 = 6.8kΩ

Capacitors
C1,C2 = 100nF 50V, X7R, 5mm pitch
C3 = 100µF 50V, 3.5mm pitch, 8x11mm

Semiconductor
IC1 = ATtiny85-20PU, programmed, Elektor Store # 150499-42

Miscellaneous
Bt1 = Battery halter for CR2032
K1 = 2x3-polige Stiftleiste
S1 = pushbutton weith large cap, (Sparkfun, red: COM-09181, green: COM-11275, blue: COM-11274)
PCB, Elektor Store # 150499-2 v1.0
8-way (2x4) pinheader and socket pair, 0.1" pitch (for MOD1)
CR2032 Lithium battery
MOD1 = NRF24L01+ 2.4GHz Wireless Transceiver module (Elektor Store # 150499-91)

COMPONENT LIST ARDUINO SHIELD

Resistor
R1 = 1kΩ

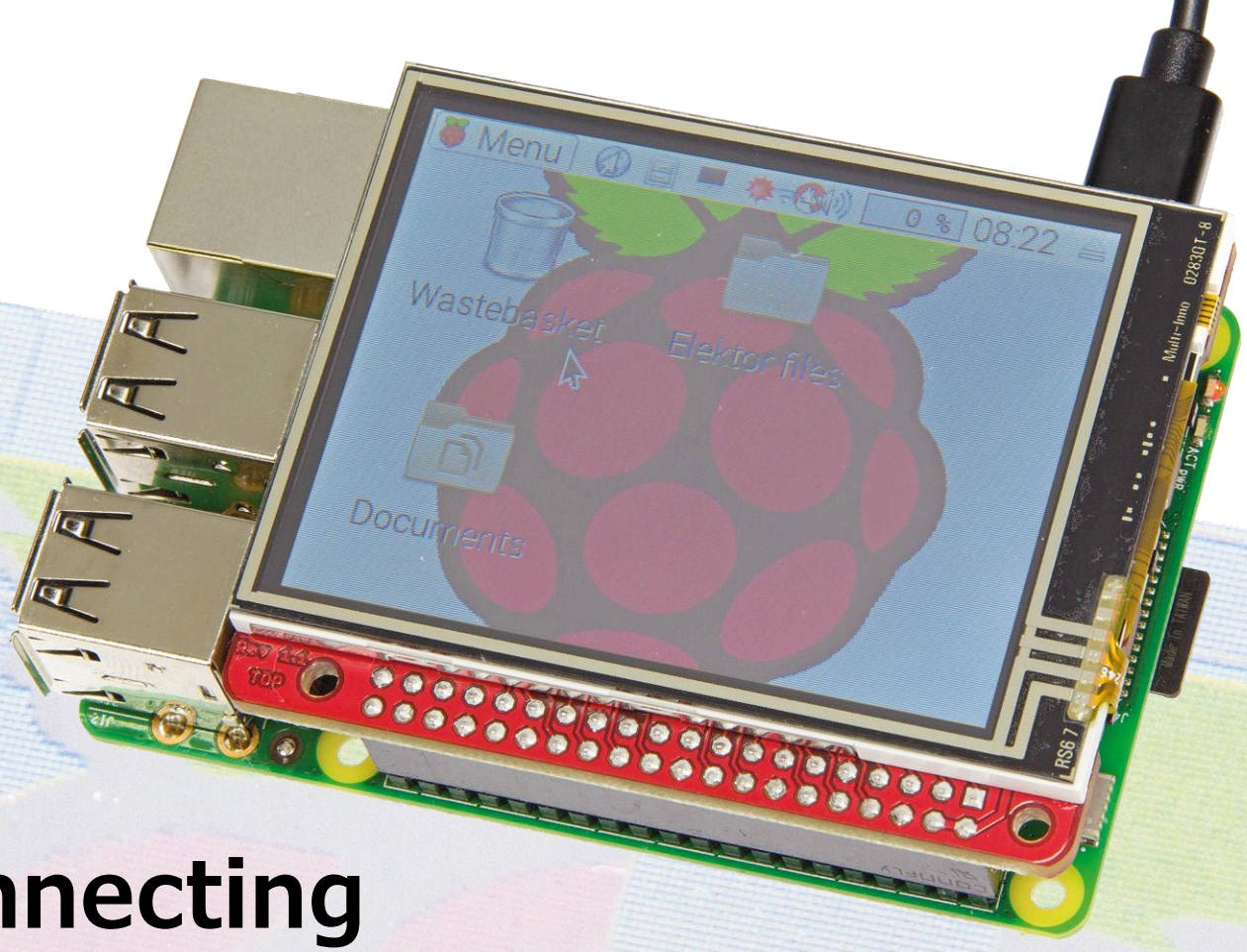
Capacitor
C1 = 100µF 50V, 3.5mm pitch, 8mm diam.

Semiconductor
LED1 = LED, green, 3mm

Miscellaneous
MOD1 = NRF24L01+ 2.4 GHz Wireless-Transceiver module, Elektor Store # 150499-91)
K1 = 6-pin pinheader, 0.1" pitch
K,K3 = 8-pin pinheader, 0.1" pitch
K4 = 10-pin pinheader, 0.1" pitch
K5 = 8-way (2x4) pinheader socket, 0.1" pitch
PCB # 150180-1 v1.0 (ELPB-NG)

FROM THE STORE

- **150650-1** - Receiver PCB in USB-stick form.
- **150650-41** - Microcontroller, Receiver.
- **150650-71** - Kit of parts, including Receiver PCB in USB-stick form, transmitter PCB (from the 'Wireless Quiz' project), all components including radio module and USB enclosure.
- **150180-1** - ELPB-NG prototyping PCB to build the initial receiver design.
- **150499-2** - Transmitter PCB (Wireless Quizbuttons).
- **150499-42** - Microcontroller, Transmitter (Wireless Quizbuttons).
- **150499-91** - Populated radio module.



Connecting an LCD to an Rpi

How easy (or difficult) can that be?

By **Luc Lemmens** (Elektor Labs)

There are large numbers of add-on boards available for the Raspberry Pi (the so-called 'HATs', Hardware Attached on Top) for a diverse range of applications. Some work immediately once you connect them, others have to be configured first or be provided with the correct software. This is not always immediately obvious...

Particularly in the case of a display module it would be handy to know what you need to do in order to conjure up an image on the screen, even if it is only for the purpose to show whether it is working or not. For some time now, Elektor have been selling a small touch-display for the Raspberry Pi [1] in their Store. The small color screen with 320 x 240 pixels and a diagonal of 2.8 inches is provided with a resistive touch panel and is very affordable. This display module is the same size as the Rpi. That makes this combination quite compact and can therefore be easily built into something.
Several buyers of this display asked us what software is required in order to work with this display; the standard Linux distributions from the Raspberry Pi organization apparently don't support this display.

The manufacturer of this display-module, the German company Watterott, has made a github website where the necessary information and software are made available [2]. Among the downloads from this page are ready-to-run SD-card images (Raspbian/Debian), with support for this touch display already integrated. With these you can simply make a new SD card. For an existing installation you can download a script from the same website that will install the FBTFT Framebuffer for the display. We have tested that script here in the labs and it works perfectly. In the event that you prefer to do everything yourself or if the script doesn't work, there is also an explanation as to how you can add the drivers manually. In any case, note that the touch screen needs to be calibrated.
A few other projects are also mentioned on the same website, built around the combination of Rpi/touch-display, such as a web radio and a mini TV. Certainly worth the effort to take a closer look!

(150824)

Web Links

[1] www.elektor.nl/touch-display-for-raspberry-pi
[2] https://github.com/watterott/RPi-Display

elektorlabs

We have moved!

By **Thijs Beckers** (Elektor Netherlands Editor)

Elektor has exchanged the 17th Century castle in rural Limbricht, Holland, for more contemporary surroundings. The Elektor Labs and editorial teams are located on the top floor in an office building opposite the RWTH technical university in Aachen, Germany (with lots of light; no more catacombs!).

Elektor has moved to a high-tech region and that has already borne fruit: we are currently cooperating with RWTH on an interesting project, which we hope to be able to publish in the course of this year.

So what's the lab situation now? Take a look at the pictures

Although the move was cross-border, it wasn't far really as Aachen is just a 30-odd drive from the previous location. Colleagues of Sales, Marketing and Memberships already occupied the ground floor of the building.

which say more than a thousand words.

For the sake of completeness: Elektor Logistics and Customer Services remain located in Susteren on familiar Dutch territory. ⏮

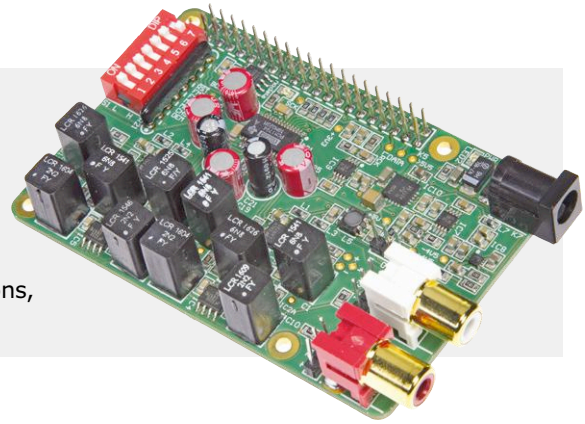
(160534)

Err-lectronics

Corrections, Updates and Feedback to published articles

Audio DAC for Raspberry Pi

Elektor 4/2017 (July & August), p. 22 (160198)



UPDATE. The latest version of the software for this project is now available for download from the /Labs page:
www.elektormagazine.com/labs/audio-dac-for-rpi-networked-audio-player-using-volumio.
It now uses RuneAudio which allows operation via Wi-Fi plus other functions, such as Internet radio (Dirble).

12-V LED Driver

Elektor 5/2017 (September & October), p. 30 (150314)

UPDATE. In the circuit diagram (Figure 1) of the above article, the common connection of jumper positions DIM1 and DIM2 are shown connected to LED + but should instead be connected to GND. The board is correct.

OBD2 Handheld using a Raspberry Pi

Elektor 2/2017 (March & April), p. 26 (160204)

UPDATE. Since the article was published, the associated OBD2 diagnostic software Pi-OBD Handheld GUI (HHGui) has been updated to make troubleshooting easier. The English error texts associated with the 4277 Diagnostic Trouble Codes (DTC) defined in the standard SAE J2012: 2007 are now implemented. For example, when a controller generates the error code P0301, the HHGui display indicates “Cylinder 1 Misfire Detected”. In addition, the software now displays additional Parameter Identifier (PID) of the SAE J1979 standard (identical in content to ISO 15031-5) in the Current Data and Freeze Frame Data menus if a controller supports these PIDs. Specifically, these are the following PIDs:

- 0x61 Driver’s Demand Engine — Percent Torque (engine torque setpoint, driver request via accelerator pedal or external request from cruise control, for example);
- 0x62 Actual Engine — Percent Torque (actual value of motor torque, calculated from usable braking torque + motor friction from PID 0x8E);
- 0x63 Engine Reference Torque (reference value of motor torque, constant value);
- 0x64 Engine Percent Torque Data, maximum 5 levels, where level 1 is idle;
- 0x84 Manifold Surface Temperature (Intake manifold surface temperature);
- 0x8E Engine Friction — Percent Torque;

The PIDs 0x61, 0x62, 0x64 and 0x8E are given as a percentage of the constant reference value of the motor torque (PID 0x63).
This now comprehensively supports the following PIDs:
0x00 to 0x64, 0x70, 0x80, 0x84, 0x8D, 0x8E, 0xA0, 0xC0, 0xE0
The new version 2.90 of the software is now available for download on the project /Labs page:
www.elektormagazine.com/labs/obd2-for-raspberry-pi.

Thomas Beck

Traffic Signals for Model Railroads

Elektor 6/2017 (November & December), p. 28 (160465)

FEEDBACK. I was very interested in this article and read it carefully but got a bit confused: In Figure 8 a total of 8 traffic light LEDs are connected to terminals K1 to K3, but an intersection needs 4 x 3 LEDs. What are the two LEDs on K3 for?

Helmut Wirth

Figure 8 only shows how the basic circuit looks. The lower 3 LEDs (K1) supply, as it says in the text, the north-south street and the next 3 LEDs at K2 control the east-west street. It is true that you need several traffic control lights; the second set of lights, pointing the other way at the opposite side of the junction (also consisting of 3 LEDs and 2 resistors) are wired in parallel. The upper 2 LEDs at K3 are for pedestrian crossing signals; here you only need red and green. Pedestrians will get a green light when S1 is pressed; for a complete intersection you will of course need 8 red, 8 green LEDs and 8 resistors (4 streets with lights on both sides!) these will again all be wired in parallel to K3. The circuit can drive this number of LEDs without any problems but it adds to the wiring so you may, for simplicity just omit the pedestrian signals. In this case the LEDs on K3 and the push button S1 are not fitted. Hope this helps.

Rob van Hest

How to Charge a Battery

Elektor 6/2017 (November & December), p. 6 (160526)

FEEDBACK. Why, in an article about rechargeable batteries was there no mention of NiZn batteries?

Peter Bitzer

I chose not to include this type of battery technology in the article because it is not (yet) widely available. I wrote a news item four years ago referencing NiZn batteries when they became available from Conrad Electronics. Meanwhile these batteries are now more widely available (see photo). NiZn rechargeables have a higher nominal voltage (1.65 V) compared to NiCd and NiMh cells. A fully charged cell tops out at 1.8 V and discharged drops to around 1.1 V. NiZn cells have very low internal resistance allowing them to be quickly charged at rates of 0.5 to 1 *C*. Rated at 2.5 Wh an AA-sized NiZn cell has about the same capacity as an Eneloop NiMh cell and offers more charge/recharge cycles.

The big advantage of this type of cell is its higher cell voltage. Some equipment does not cope well with the lower cell voltage offered by NiCd and NiMh cells. This feature can also be a disadvantage however; the equipment must be designed to accept the higher initial input voltage of NiZn cells and must also turn off when the cell voltage drops too low to avoid the deep discharge condition.

Charging the cells is similar to the regimen used for lead-acid or lithium cells: start with a constant current (0.5 to 1 *C*) and then use constant voltage level of 1.9 V per cell until the charge current drops below 0.05 *C*.

The cells can only be recharged by a suitable NiZn charger unit.

Dr. Thomas Scherer

Kinetic Sculpture

Elektor 5/2017 (September & October), p. 78 (160272)

FEEDBACK. I took the photo used in the article at Changi Airport, Singapore in 2016. It uses the same technology as described in the article.

Heinrich Lendle

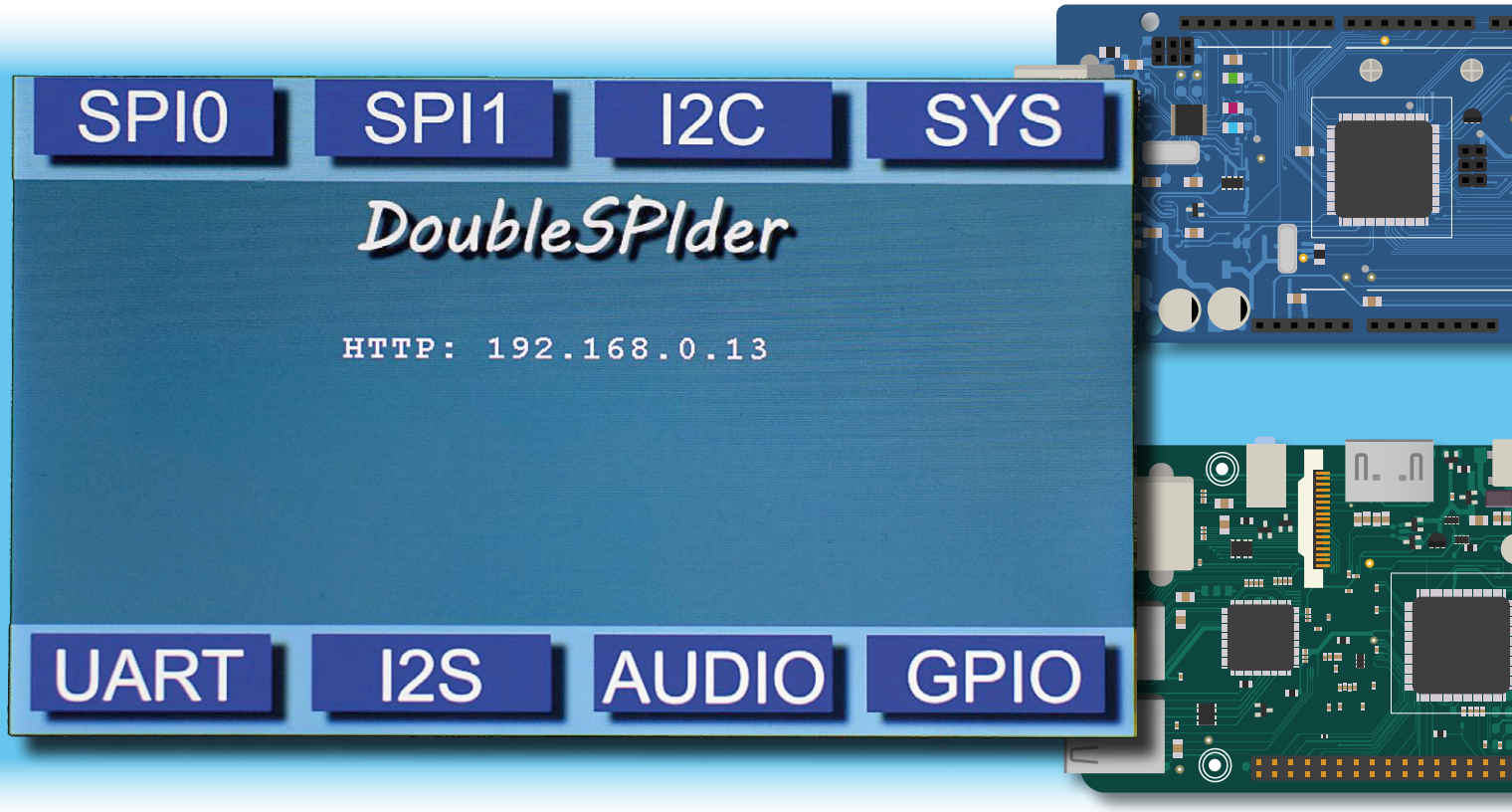
Editing Elektor circuit diagrams

One of our readers, Otto Bubbers, points out that circuit diagram files in PDF format available for our projects can be edited using freely available programs such as Libre Office Draw.

That works well but only on a purely graphical level so to draw a single resistor symbol for example you need to construct it from individual lines. The Elektor diagrams are protected by copyright so the material is restricted for personal use only (adding notes to the circuit for example) or for educational use where the original circuit source must be referenced.

DoubleSPIder

A universal interface converter for microcontroller projects



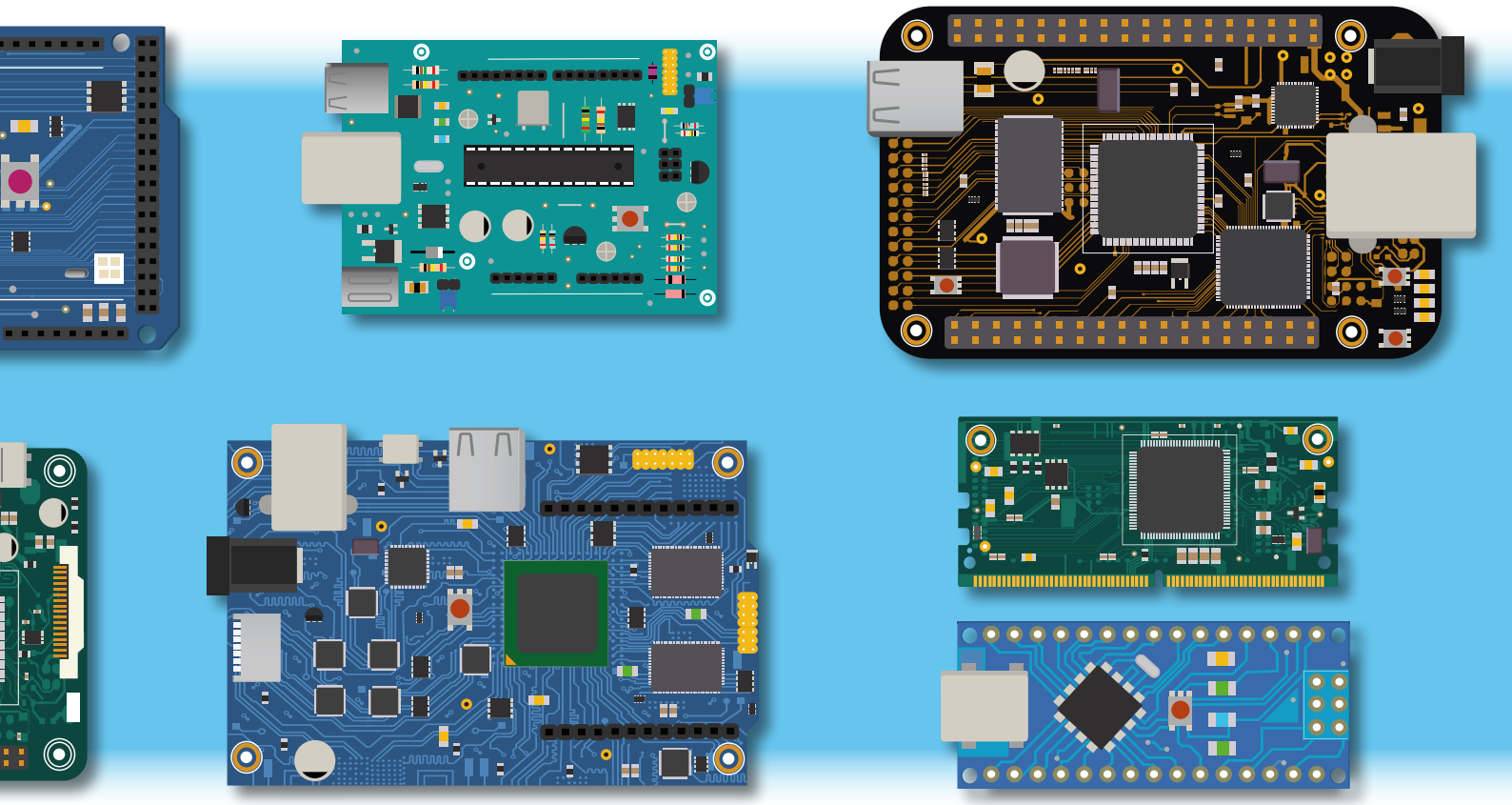
By **Torsten Jaekel** (USA)

When you want to try out a new microcontroller or a peripheral device, you are often faced with the question of how to control it over an SPI, I²C or UART interface or how to access it from a PC. Desktop and laptop PCs do not have these interfaces, so you need to look for a suitable host interface adapter. That also involves installing or adapting drivers. DoubleSPIder simplifies all this, because it can connect virtually all devices with each other.

DoubleSPIder is an interface converter in the form of an external adapter that provides three standard serial interfaces commonly used for microcontroller projects. It can also do a lot more: DoubleSPIder can be used directly over a network via TCP/IP without requiring any driver installation. This is supported by an integrated web server (**Figure 1**). That makes it easy to send commands or data packets to a microcontroller from the web browser on your PC via the SPI, I²C or UART interface.

DoubleSPIder facts and figures

Hardware platform:	STM32F769I-DISCOVERY development board with ARM Cortex M7 MCU
OS:	STM HAL driver, FreeRTOS, lwIP; open source firmware project
SPI:	3×, of which one direct and two shared with slave select; Aardvark pinout
I²C:	3× (single bus)
Internal trigger:	2× (alternative to I²C)
GPIO:	8×, freely configurable as input or output, including 1× open drain and 1× UART (RX, TX)
Network:	RJ45 Ethernet, 10 Mbit/s, web server, TCP port 8080, REST API server for Python, DHCP or static IP address
User interfaces:	LCD 800×400 touchscreen, UART command line (terminal), Pico C interpreter, web server
USB:	USB audio for data or external storage for SD card
SD card:	FAT file system, up to 4 GB, long file names



But that's not all — you can also send commands using a UART command line interpreter. There is also another TCP port available, which you can use, for example, to perform complex configuration tasks for external ICs using Python scripts, or to transfer any desired data in both directions between the PC and external devices via the SPI, I²C or UART interface.

DoubleSPIder configuration

Now that you know the purpose of DoubleSPIder, let's look at what it consists of. On the hardware side, DoubleSPIder is based on a ready-made evaluation board for a 32-bit ARM SoC from ST (more about this later), plus an extension board holding several connectors and a few electronic components. That makes construction very easy while offering a lot of convenience and performance. It also means that the interface adapter supports much more than just three serial interfaces. It provides the following:

- 3× SPI, of which one is a shared SPI bus with two Slave Select signals. The pinout is compatible with the Aardvark SPI adapters from Totalphase [2].
- 3× I²C, single bus with three physical connections and optional jumpers for pull-up resistors. Alternatively, the I²C signals can be used as interrupt triggers.
- 1× USART with logic-level RX and TX signal lines.
- 8× GPIO, freely configurable as input or output. One of these GPIO lines can also be configured as an open drain output, for example to generate a hardware reset on a connected board.
- 1× USB 2.0, which for example can be used as a driver-

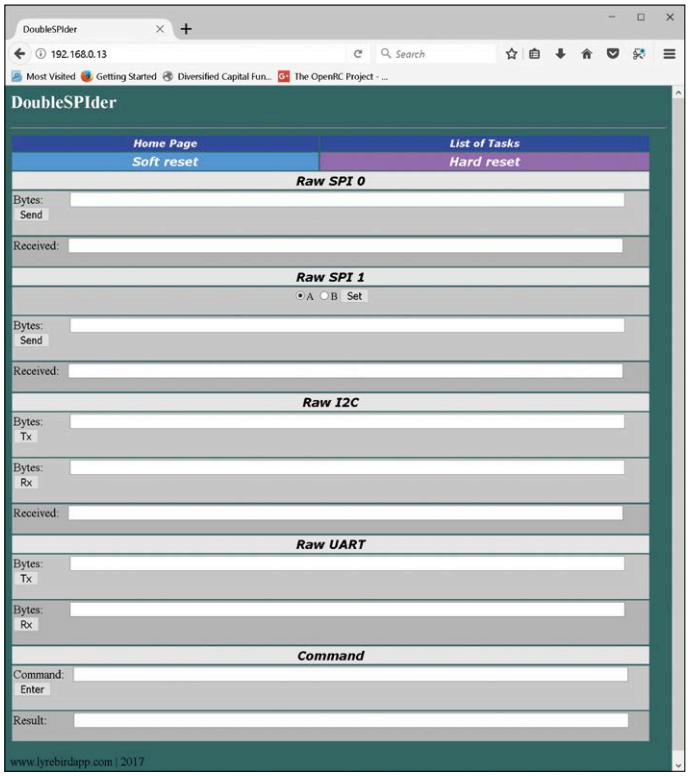


Figure 1. The landing page of the integrated web server, displayed by the browser of the host PC.

less sound card.
- 1x SDD card slot for bulk data storage.
- 1× Ethernet port, 10 Mbit/s, for communication with the integrated web server.
- 4” touchscreen for configuration and operation.

All of these features and ports are shown on the block diagram in **Figure 2**. As you can readily see, DoubleSPIder is a top-end, universal and network-based interface converter for SPI, I²C and USART serial interfaces as well as GPIO, which can be used for testing of test bare microcontrollers or development boards as well as direct connection of peripheral devices, such as serial sensors. No driver is necessary for installation. All interfaces can be accessed from a web browser on the PC.

Touch configuration

A display is particularly helpful for configuration of the interface converter. DoubleSPIder provides a touchscreen display, making manual configuration especially easy. All configuration parameters – SPI mode, SPI bit rate, I²C bit rate, GPIO pin direction, etc. — can be set using the touchscreen.
The firmware for the core MCU of the DoubleSPIder supports a graphical user interface (GUI) on the LCD (**Figure 3**). All settings can be made directly on the touchscreen. Configured settings can optionally be saved as a startup configuration file in the on-board QSPI memory, for automatic retrieval after a reset or the next power-up.

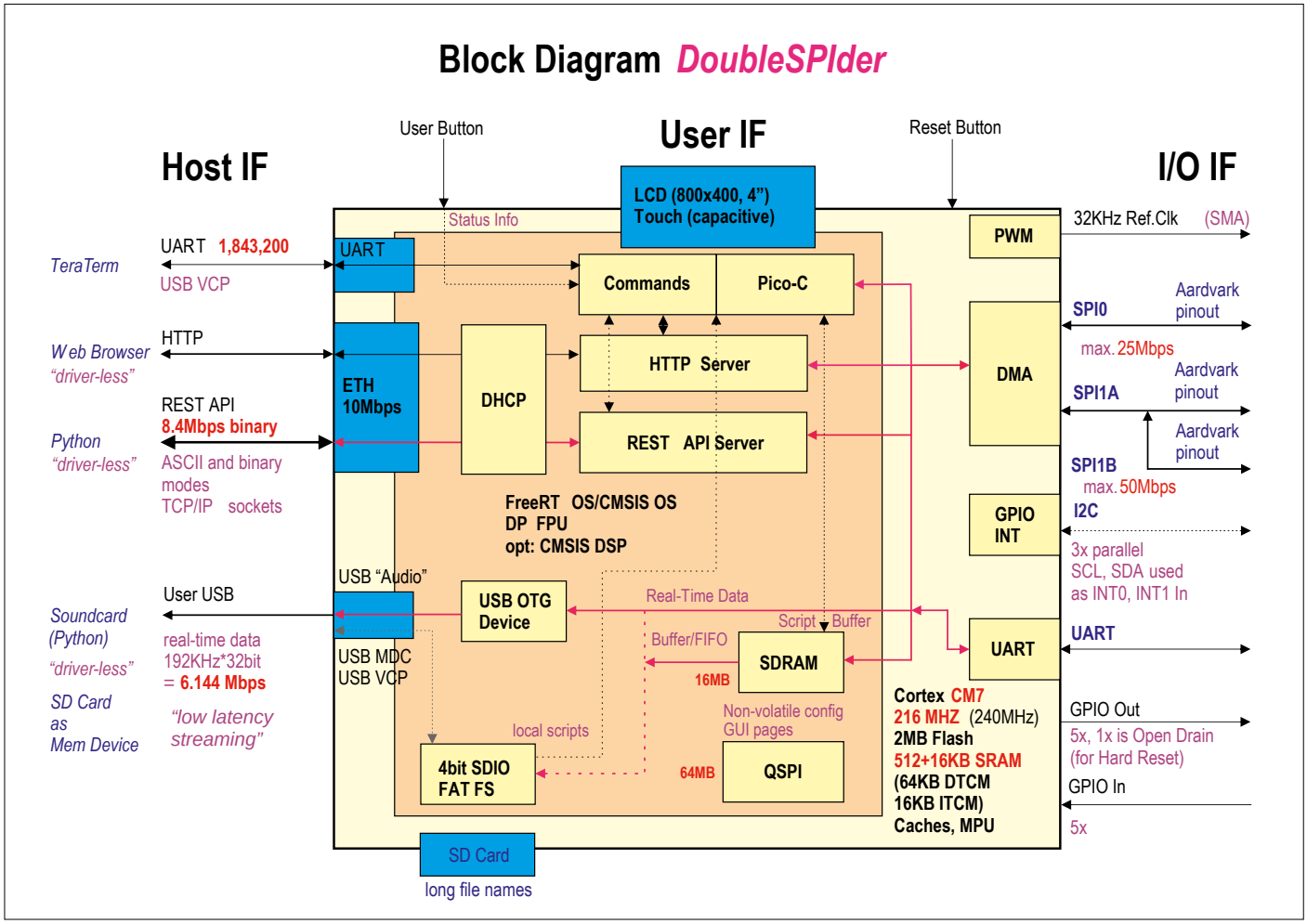


Figure 2. Block diagram and ports of the ST Discovery Kit board.

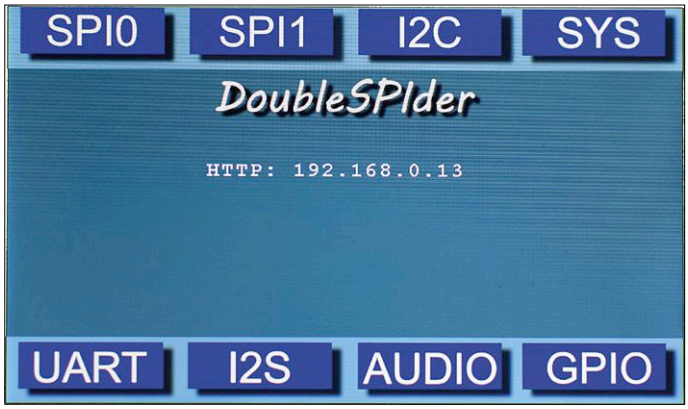


Figure 3. The user interface on the DoubleSPIder 4" touchscreen.

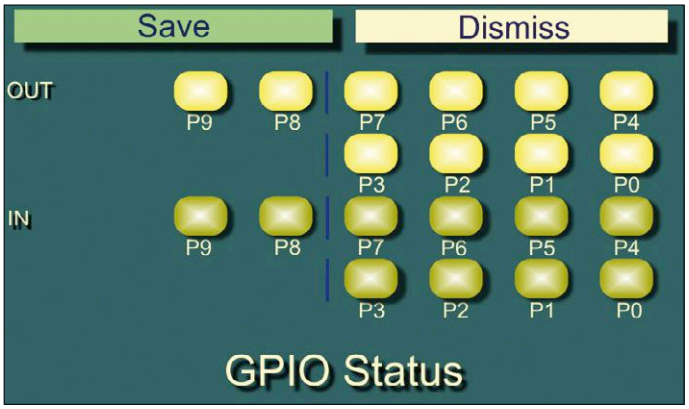


Figure 4. The GPIO status query and level change screen.

The programming of the GUI has been kept simple. The graphic components are predefined icons in BMP format (see **Figure 4**), which are stored in the QSPI memory. That leaves enough space available in the internal flash memory of the MCU to allow users to add more code for supplementary functions. The only disadvantage of not using a GUI manager or wizard is that handling the touch events (for example, for the buttons on the display) is a bit more complicated because the touch coordinates have to be decoded.

Along with the settings of many parameters and functions, the states of the GPIO signals can be displayed on the touchscreen, and the GPIOs configured as outputs can be changed. However, it is not possible to generate I²C or SPI commands or data packets on this screen. These tasks can be performed with the UART command line interpreter, with the web interface, or with Python via the corresponding TCP port.

Scripting

The main focus of DoubleSPIder is on scripting, for example with Python on a host PC. You can use scripts to configure and control (directly or remotely) devices and boards connected to DoubleSPIder and to exchange data, retrieve measurement data from sensors, and so on.

Along with the command line interpreter, which can be used interactively with a terminal emulator program such as TeraTerm, DoubleSPIder provides its own internal Pico C interpreter. Small programs can be entered directly as C code. This C code is interpreted directly without conversion, eliminating the need for compilation. This method can also be used to program functions directly in C for subsequent execution in the device. **Figure 5** gives an impression of how serial interfaces can be manually controlled using C code.

Python scripts running on the host PC can also send commands via the UART interface or use the Pico C interpreter. Even better, Python scripts can directly use the dedicated TCP port 8080. Python scripts can use network sockets to exchange commands and data much faster. A binary mode is additionally provided.

To provide optimal support for handling real interrupts (with the I²C signals configured as interrupt triggers), there is a Python accelerator that acts as a programmable state machine. Interrupt events can be programmed to initiate specific actions when they occur. That allows fast handling of time-critical events and offloads the Python scripts. Without the Python accelerator, constant polling of the corresponding pins from the host PC would be necessary — a very cumbersome procedure. That would require continuously reading an interrupt status register, for example over SPI, to determine whether anything has changed in the register. Polling should be avoided as much as possible because it is slow, costs processing power, and plugs the communication channels. This also applies in particular to the use of scripts intended to run in the background, which makes programming more difficult.

Audio via USB

DoubleSPIder also has a user USB interface. This fast serial port makes it easy to transfer relatively large amounts of data. Specific drivers are usually necessary for using USB on PCs, but that can be avoided by using drivers that are present on every PC by default. Consequently, the DoubleSPIder USB interface is simply configured as an audio interface. That works with every PC.

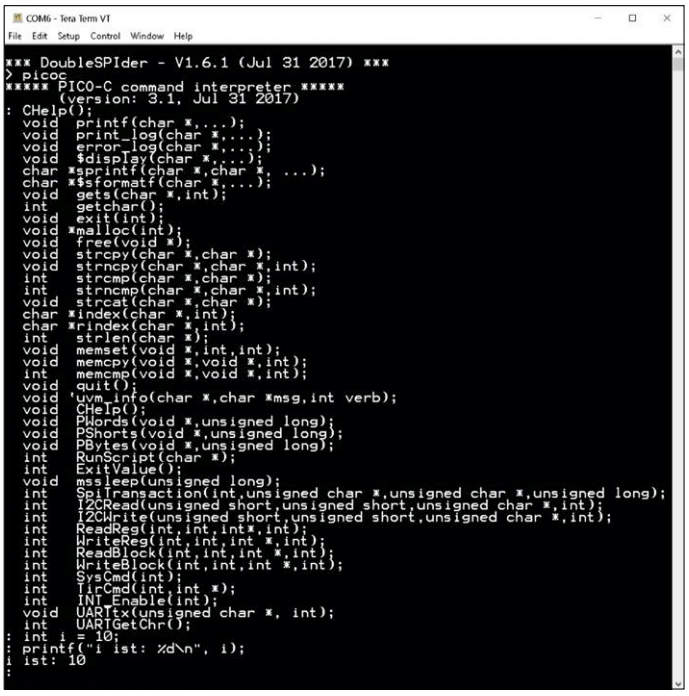
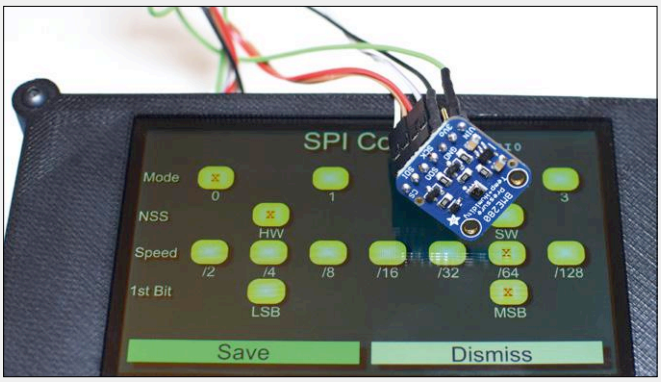


Figure 5. Code example for the use of implemented Pico C functions.

Connecting a BM280 sensor module

The BME280 is an integrated environmental sensor module from Bosch Sensortec. It combines a temperature sensor, a relative humidity sensor and an air pressure sensor in an 8-terminal LGA package with an edge length of just 2.5 mm. As this package size is a bit unwieldy, there are also small breakout boards fitted with this module for manual connection or soldering. The BME280 module supports the I²C and SPI interfaces. In this demo the breakout board, which is also available in the Elektor Store [6], is used in SPI mode. The connection details for DoubleSPIder and the various options for reading the chip ID are described below.



BME280 module connection details

The BME280 module is connected to the DoubleSPIder SPI2 (H1) interface. For the sake of clarity, the pin numbers from the BME280 data sheet [7] are listed below along with the usual designations of the module terminals.

Module	Pin	DoubleSPIder
VIN	6 & 8	3.3 V, pin 4
GND	1 & 7	GND, pin 2 or 10
SCK	4	SPI2_SCLK, pin 7
SDO	5	SPI2_MISO, pin 5
SDI	3	SPI2_MOSI, pin 8
CS	2	SPI2_NSS, pin 9

Chip ID readout over SPI

The chip ID can be read using an SPI transaction. The first byte is the register address 0xD0 (7-bit address with the read/write bit set to “read”). A second dummy byte must be sent in order to transfer the content of the register. The value of the second received byte should be 0x60.
The following readout methods are terminal protocols, with the exception of the screenshots. The relevant inputs are shown in blue.

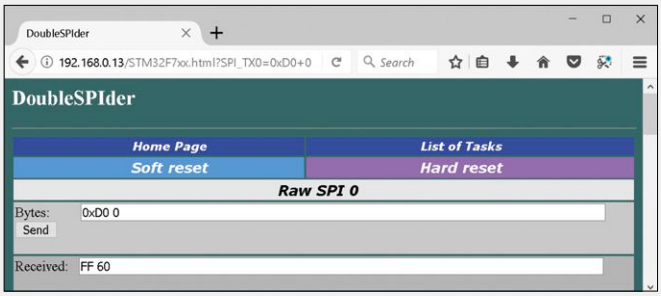
Via UART

```
*** DoubleSPIder – V1.1.1 (Jul 31 2017) ***
>
> spi0 0xd0 0
FF60
```

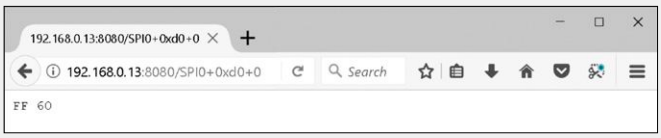
Via Pico-C

```
> picoc
***** PICO-C command interpreter *****
      (version: 3.1, Jul 31 2017)
: unsigned char spiTx[2] = {0xD0, 0};
: unsigned char spiRx[2];
: SpiTransaction(0, spiTx, spiRx, 2);
: printf(„BME280 ChipID: %x\n“, spiRx[1]);
BME280 ChipID: 60
```

Via Webbrowser



Via REST-API-Server, Port 8080



Via Python Script And REST API Server

```
#/usr/bin/env python

#Python network access to DoubleSPIder MCU board via REST_
API-IP: UDP port 8080
#using ASCII format (commands) on REST_API for SPI
transaction

import socket

def Main():
    host = input("Enter IP address shown on display: ")
    port = 8080

    message = 'GET /SPI0+0xD0+0'

    mySocket = socket.create_connection((host, port))
    mySocket.setblocking(1)
    mySocket.sendall(message.encode(encoding='UTF-8'))
    data = mySocket.recv(1024)

    #now print what we have received
    print(data)

    print('Good bye')
    mySocket.close()

if __name__ == '__main__':
    Main()
```

When you connect the USB cable to the PC, a new audio recording device appears on the screen. However, no audio data packets are actually transferred, but instead any desired data read from the connected devices or modules. That could be data from FIFOs or sensor data, for example, which is transferred continuously or intermittently.

This data port camouflaged as an audio channel is configured for stereo audio with 16-bit resolution and a sampling rate of 48 kHz and with ten channels. That yields a maximum possible packet size of 960 bytes. These packets can be transferred at 1 ms intervals, resulting in a maximum of 960 kB/s. The advantage of this is that USB audio is fast and has relatively low latencies.

It is fairly easy to eavesdrop on this "audio recording device" with Python and capture the USB data packets. The data in the packets can be quickly decoded and processed for as long as you like. **Figure 6** shows how to use a Python script to transfer this data (originally obtained from a serial data stream) and store it as "USB audio".

```
C:\Python27\python.exe
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import sounddevice as sd
>>> sd.default.device = 26
>>> sd.default.channels = 2
>>> sd.default.samplerate = 192000
>>> rec = sd.rec(5 * 192000, dtype = 'int16')
>>> with open('USB_recording', 'wb') as f:
...     f.write(rec)
...     f.close()
...
>>>
```

Figure 6. Python script for the USB interface configured as an audio interface.

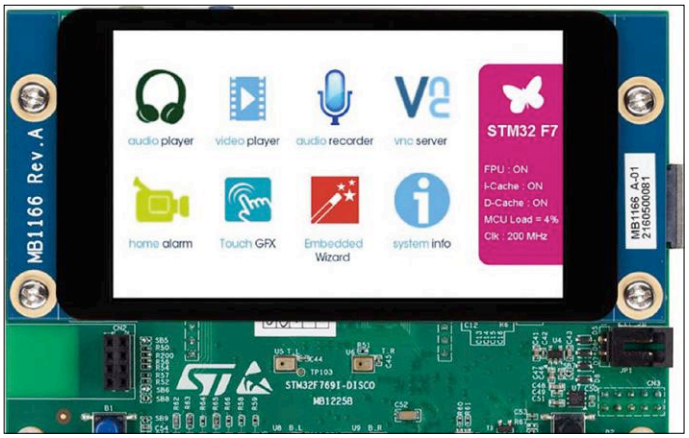


Figure 7. The ST evaluation board with the STM32F769NI-MCU costs approximately €80 with the 4" touchscreen.

More...

This project also has an on-board slot for an SD card. The integrated FAT file system allows Pico C scripts to be stored on the SD card and run from the card. Data can also be written to the SD card or read from it. To avoid the need for constantly removing the card in order to read or write it on the PC, the USB port can be configured as an external memory interface

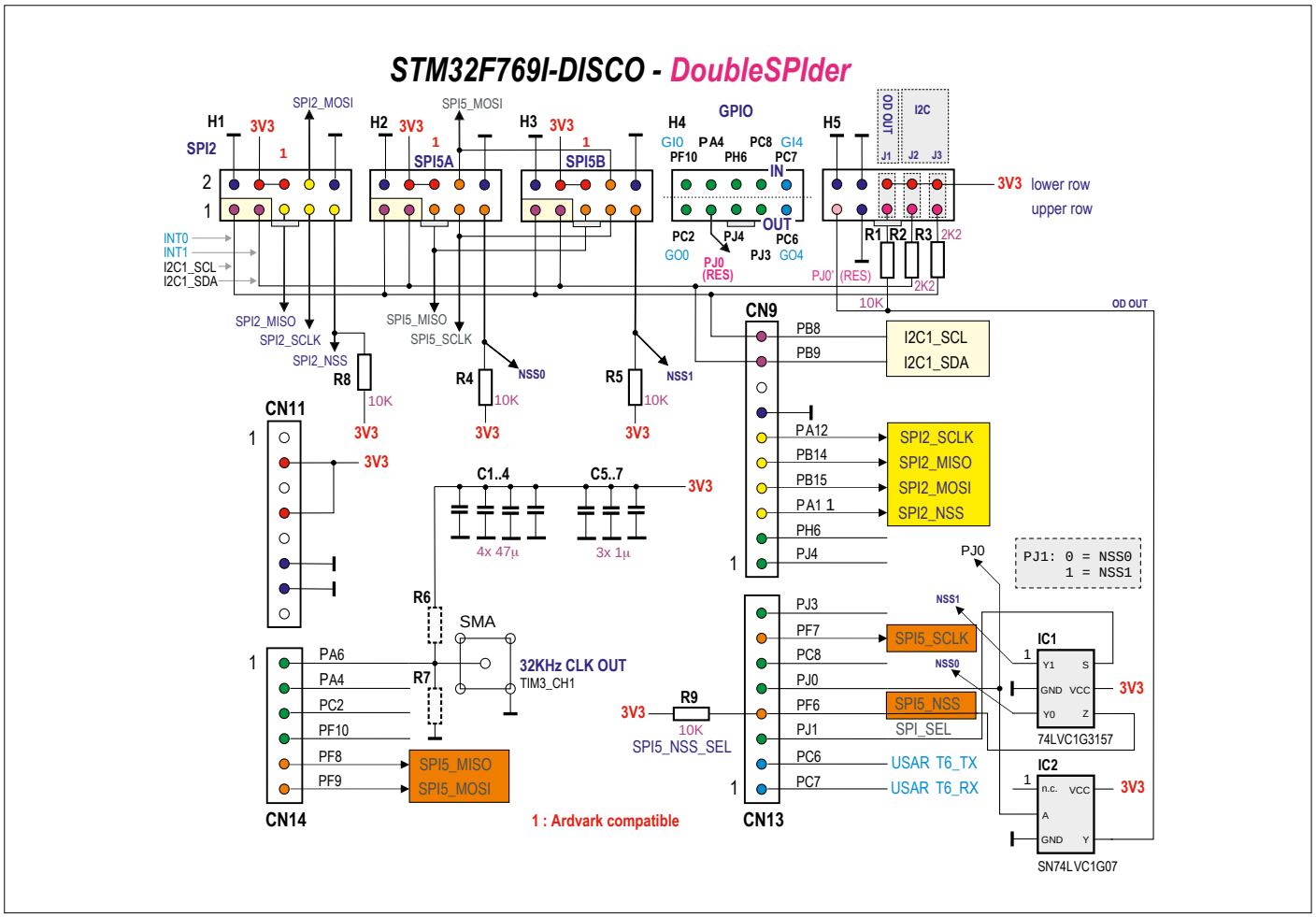


Figure 8. The circuitry of the extension board is very simple, consisting of just two ICs, several capacitors and resistors, and a handful of connectors.

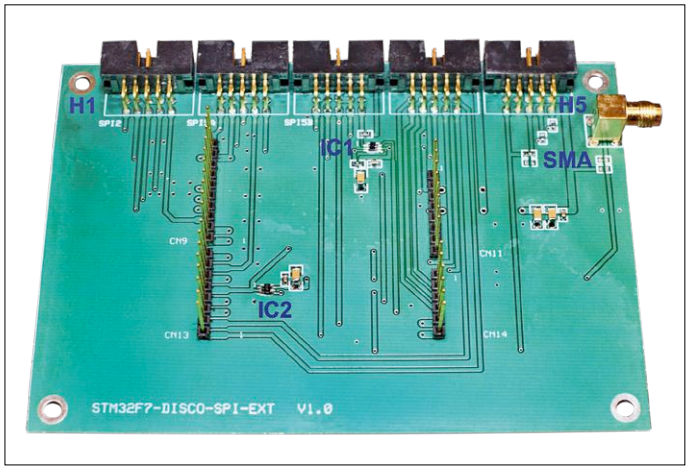


Figure 9. The fully populated development board.

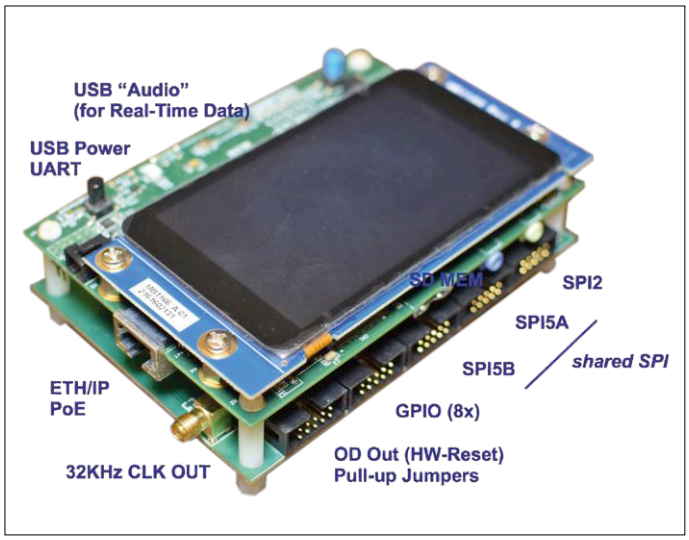


Figure 10. The evaluation board together with the extension board.

Figure 7. DoubleSPIder in a homemade enclosure produced on a 3D printer.

instead of USB audio. That makes the content of the SC card directly available to the PC. Then you can conveniently copy, edit or delete any Pico C scripts or other files on the card.
It is also conceivable to configure the SPI interfaces as I²S audio interfaces, which can be used together with the USB audio interface. Then you could connect a standard audio module, such as an ADC or DAC. The audio IC could be configured via SPI or I²C, with the data from or to the I²S audio port being transferred to or from the PC via the USB port.
The firmware for the MCU board is available as an open-source project on the Elektor web page for this article [3]. It uses the freely available HAL driver from STMicroelectronics, FreeRTOS as the operating system, and the lwIP network stack [4].
More details and documents on DoubleSPIder can be downloaded from [3]. A PDF document with more information on connection and use is available there.
As previously mentioned, DoubleSPIder is a project consisting of a development board from ST, namely the Discovery Kit with the STM32F769NI MCU (**Figure 7**), and an extension board with connectors and some supplementary components. **Figure 8** shows the schematic diagram of this extension board, and **Figure 9** shows the assembled board. The assembled unit is shown in **Figure 10**. The author also produced a matching enclosure on a 3D printer; **Figure 11** shows the prototype installed in this enclosure.
The 3D design files for the enclosure in Google SketchUp format and the design files for the extension board in the format of the free CAD program ExpressPCB [5], along with the component list, code and some demo files, are available at [3].

(160390-I)

About the Author

Torsten Jaekel has been working at Broadcom in California as a system engineer for more than 10 years. There he is involved in the development, commissioning and testing of sensors and touch controllers.

Web Links

[1] www.st.com/en/evaluation-tools/32f769idiscovery.html
[2] www.totalphase.com/products/aardvark-i2cspi
[3] www.elektormagazine.com/160390
[4] http://savannah.nongnu.org/projects/lwip/
[5] www.expresspcb.com
[6] www.elektor.com/bme280-mouser-intel-160109-2
[7] www.bosch-sensortec.com/bst/products/all_products/bme280

Photo: Anselm Rapp, Wikimedia Commons.

Figure 1. The Uher Report 4000L tape recorder (1964).

Uher Report 4000L (1964)

A portable tape recorder for frenzied reporters

By **Karl-Ludwig Butte** (Germany)

You probably think that reel-to-reel tape recorders are big and heavy, not suitable for mobile use. That's not entirely true — the Report 4000 family of tape recorders from Uher were masterpieces of precision engineering. Thanks to fully transistorised design and ingeniously compact construction, the Munich-based company succeeded in making reel-to-reel recorder technology portable as early as 1961.

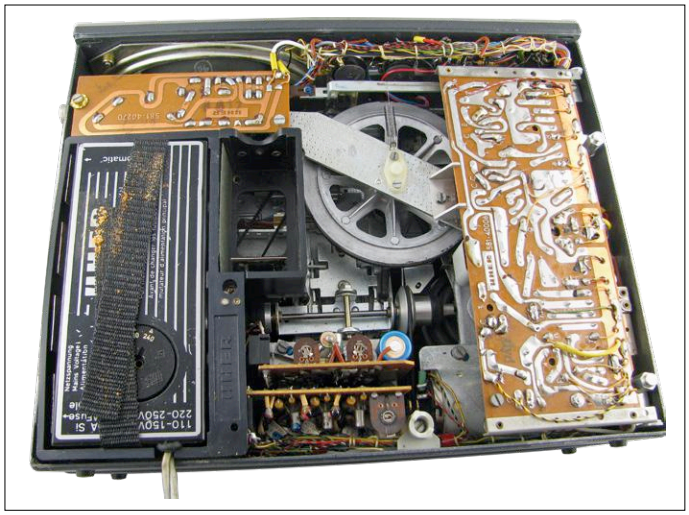


Figure 2. A first look under the bonnet of the Uher Report 4000L.

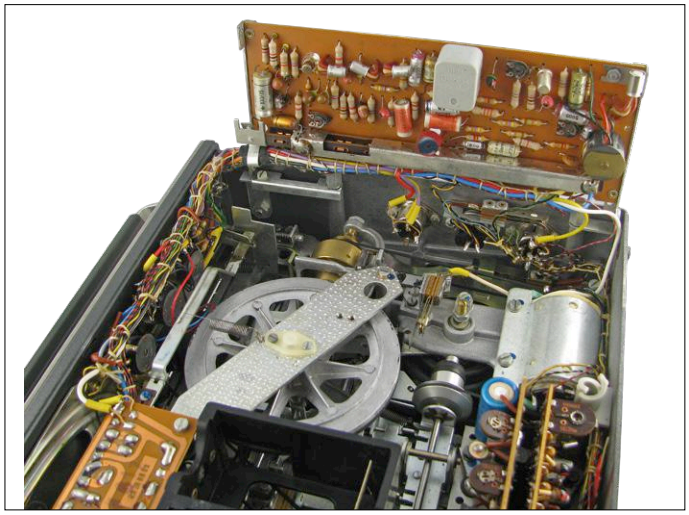


Figure 3. The record/playback amplifier board is hinge-mounted.

Figure 4. A tape head demagnetizer for removing undesirable magnetisation.

The Uher tape recorders were absolutely top quality, but they were expensive. In my youth, owning one of them was an impossible dream. When I was recently offered a Uher Report 4000L at an attractive price, I could not resist and took advantage of the opportunity to have a closer look at this legendary device. **Figure 1** shows the object of my pride with the unmistakable speed selection lever, a characteristic feature of this family of tape recorders. The tape reels are located beneath a protective hinged metal cover with a clear plastic window. They are secured on their hubs by spring-loaded clips. An accessory leather case and shoulder strap made carrying and using the portable recorder relatively easy. These sturdy devices were used not only by countless reporters, but also by radio studios, the German Meteorological Service and the FBI. The Uher Report family even made it into the movies – the tape recorders were used by Agent 007 of Her Majesty's Secret Service, among others [1], and Pepe der Paukerschreck and his cronies in a German classroom comedy used them to annoy their teachers.

Under the bonnet

If you're just as curious about what's inside this machine as I am, please join me on an exploratory tour. The rear panel is secured by a single screw, which can be opened with a coin or similar tool. This allows the power source to be quickly changed from battery (primary or rechargeable) to mains (AC line). The internals are additionally protected by a fixed black cardboard sheet. However, it is secured by just three snaps and is also easy to remove. The view in **Figure 2** is certainly impressive. Everything looks very compact and orderly. The wiring is nicely routed and bundled together, just like the wiring harness of a car. The mechanism is made from solid metal and is able to resist rough handling in the hands of reporters.

One thing you have to give the Uher engineers credit for: they clearly had a warm place in their hearts for service technicians and anyone else who had to work on the internals of tape recorders. Everything is easily accessible, and wear parts such as drive belts can be replaced without having to dismantle half of the device. The record/playback amplifier board on the right in Figure 2 covers part of the underlying mechanism. To provide access to the mechanism, the engineers mounted this circuit board on hinges (**Figure 3**). Only one screw has to be removed in order to flip up the circuit board.

In my device the three drive belts were completely worn out, which is hardly surprising after 53 years. Fortunately I had replacement belts on hand. Aside from that, thorough cleaning was of course necessary, especially the tape heads. My tape head demagnetizer was happy to be of service again. Let's see a show of hands among *Retronics* readers: How many of you still have a tape head demagnetizer? For the younger generations, **Figure 4** shows an example of this antique tool, which is used to remove undesirable magnetisation that gradually builds up in tape heads and tape transport parts over the course of time and can adversely affect recording and playback quality.

After these activities, it was time for the first functional test. It took a little while for the motor to start up, but then it purred away just like old times. Fortunately I still had an audio tape on a reel that was small enough to mount on the recorder. The maximum reel diameter of 13 cm is not overly generous, but it is sufficient for two hours of operation at 9.5 cm/s. Everything

ran perfectly; even the incandescent lamp for illumination of the VU meter was still okay.

Specifications

The specifications of the Uher Report 4000L are certainly respectable, even by modern standards. The frequency response depends on the tape speed. As shown in **Table 1**, the device not only meets the requirements of the German Hi-Fi standard (DIN 45500) for a frequency response from 250 Hz to 6,300 Hz [2], but also exceeds the requirements of its successor (DIN 45511) for a frequency response from 80 Hz to 8,000 Hz. The only speed that does not support Hi-Fi quality is 2.4 cm/s. The maximum wow and flutter are specified as ±0.15% at 19 cm/s, and the dynamic range is 56 dB. The transformerless push-pull output stage delivers 1 W into 4 Ω.

Table 1. Frequency response versus tape speed.

Speed	Frequency response	DIN 45500	DIN 45511
2.4 cm/s	40 – 4,500 Hz	–	–
4.75 cm/s	40 – 10,000 Hz	X	X
9.5 cm/s	40 – 16,000 Hz	X	X
19 cm/s	40 – 20,000 Hz	X	X

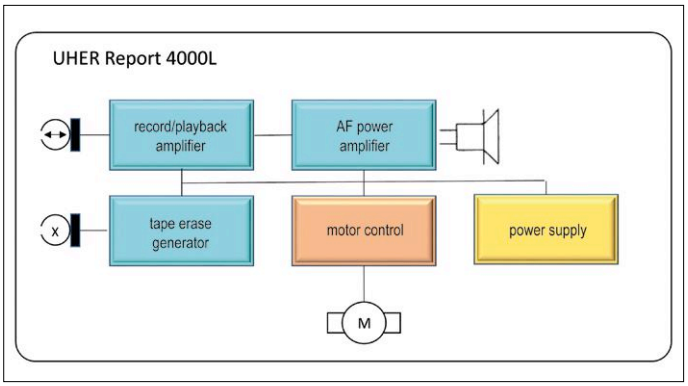


Figure 5. Block diagram of the Uher Report 4000L.

Circuit details

First let's have a look at the block diagram in **Figure 5**. The device consists of a combined record/playback amplifier, an audio output amplifier stage (which drives either the built-in loudspeaker or an external loudspeaker), a bias/erase signal generator, a motor controller and a power supply.

A short search on the Web [3] turned up the schematic diagram of the Report 4000L (**Figure 6**), which is reproduced here with the kind permission of ATIS Systems. Most of the transistors here are germanium types, which are now outmoded but were state of the art at that time: three each OC305, three each AC151, one AC153, and one AC176. The only silicon transistor in the audio portion of the device is an BCY 51. The motor controller employs an additional three each AC153, three each OC305, and two each BFY39 (the latter being silicon transistors). All in all, there are only seventeen transistors in the entire device. If you built a tape recorder now, you would easily have more than ten thousand transistors due to the inevitable microcontroller.

The combined record/playback amplifier in the upper left third of the schematic diagram is primarily built with common-emitter circuits, which enable very high voltage gain per stage. The complex equalisation network, which is adapted to the tape speed selected by the speed lever, is noteworthy. During recording, the signal can be monitored using the built-in loudspeaker or an external loudspeaker. Input connectors are provided for a microphone and a radio/phono signal source.

The push-pull output stage is shown in the upper right corner of the schematic diagram. It is formed by the AC153 / AC176 transistor pair, along with a driver stage built around an AC151.

The bias/erase signal generator in the lower left part of the schematic diagram supplies a 50-kHz HF signal for tape bias and erasing. Next to it you can see the VU meter, which is equipped with a switch so that it can also be used to indicate the battery voltage.

In the lower middle part of the schematic diagram you can see the motor control circuitry, which incorporates some clever features. However, to properly understand them you need to consult the service manual [4]. Here again the advantages of the Internet come to the fore: such easy access to operating and service manuals for devices was unimaginable in the pre-Web era. But let's get back to the motor and its control circuitry. The motor has fixed armature windings and a rotating permanent magnet. The AC voltage induced in the armature windings by the permanent-magnet rotor is used as a feedback signal for motor speed control. The motor is started with the aid of a special start-up commutator, which is keyed to the rotor position and has contacts controlled by centrifugal force. When the motor speed approaches the target speed, the centrifugal forces lifts these contacts free from the commutator and the motor control circuit switches to normal speed control. Pictures of the internal parts of the motor can be found at [5], among other places.

Finally, there is the power supply in the lower right part of the schematic diagram. Aside from a small number of electronic components, it mainly consists of a simple black plastic box (**Figure 7**). However, the underlying idea is very clever. The box is designed to hold five D cells to provide power for portable use. A mains (AC line) power supply (also shown in Figure 7) and a rechargeable battery, both of which fit exactly in the battery compartment, were available as accessories. A cable was also available for charging the rechargeable battery from a car battery. If you have a closer look at the schematic diagram, you will see that the circuit has the positive supply voltage connected to ground. That is now unusual, but at the time it was very common.

Summary

The Uher Report 4000L is still an interesting device due to its sturdy construction, easy access for servicing and compact form. The design is entirely oriented to the needs of the user and conceived for a long useful life. That appears to be out of fashion with modern devices, which are often permanently glue-bonded and thus end up as electronic waste when the internal battery has given up the ghost, since it cannot be replaced. It is sad to think that one of the reasons for the demise of traditional companies such as Uher is that their products lasted too long.

A lot has already been written about the varied history of the Uher company — much more than could be told within the

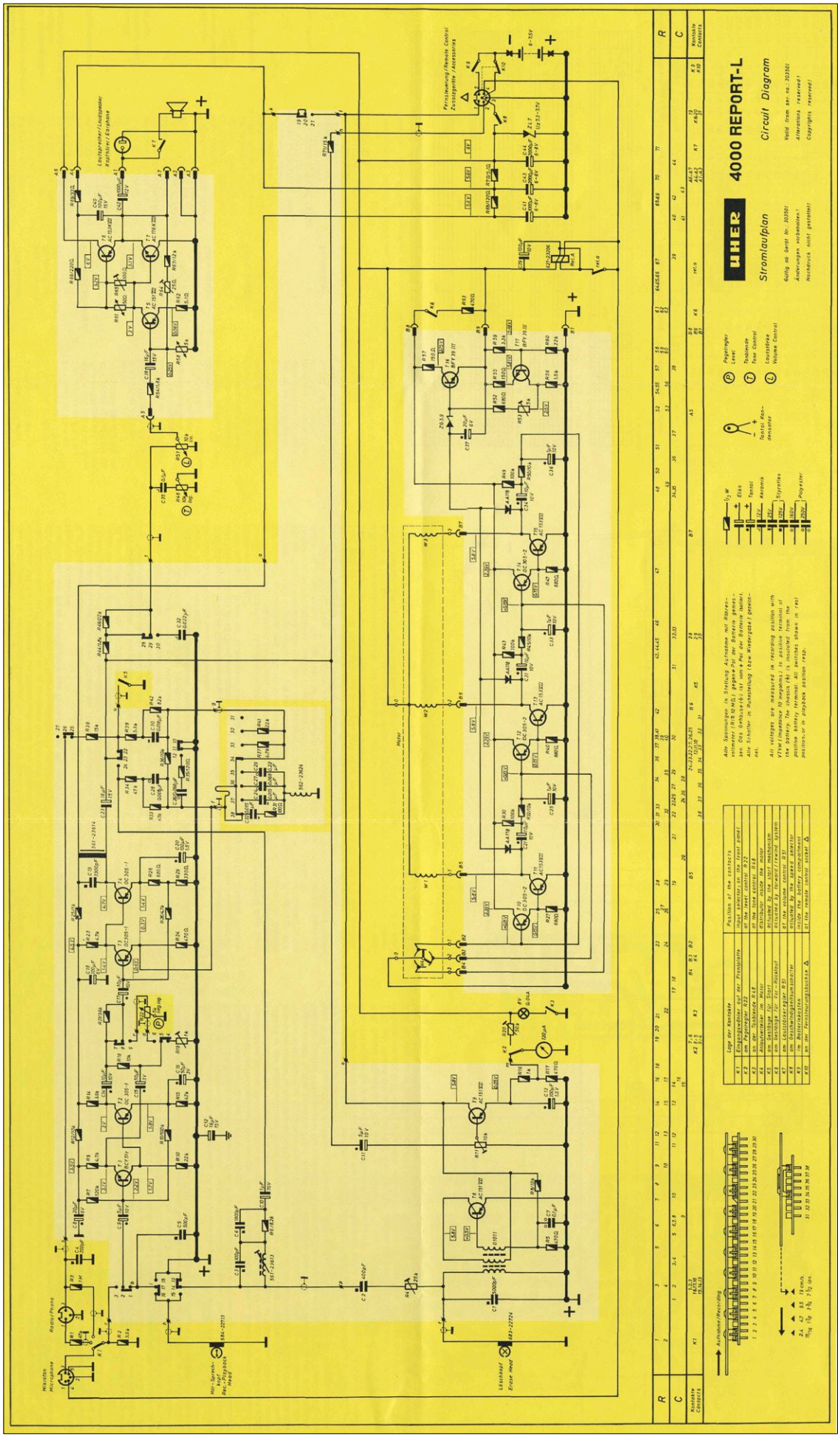


Figure 6. Detailed schematic diagram of the tape recorder. Reproduced with the kind permission of ATIS systems GmbH, which holds the copyright.

Figure 7. Battery compartment and accessory power supply.

scope of this article. However, for readers with an interest in history and the ability to read German (or wield machine translation), I can recommend links [6], [7] and [8], which give an impression of both the glorious and the less fortunate years of the Uher company. Link [7] also provides especially interesting information and photos from a long-term Uher employee.

(160559-I)

Web links

[1] https://en.wikipedia.org/wiki/Uher_(Brand)

[2] https://en.wikipedia.org/wiki/High_fidelity

[3] www.tonband.net/unterlagen/uher/sm_4000_l.pdf

[4] http://tonbandwelt.magnetofon.de/uher/4000/serva1.htm

[5] http://forum2.magnetofon.de/board2-tonbandger%C3%A4te/board20-tipps-und-erkenntnisse/16322-der-kollektorlose-b%C3%BChler-motor-des-report-von-innen/

[6] https://de.wikipedia.org/wiki/Uher#cite_note-7

[7] www.uher-erinnerungen.de/

[8] www.spiegel.de/spiegel/print/d-40350691.html

ESTD 2004

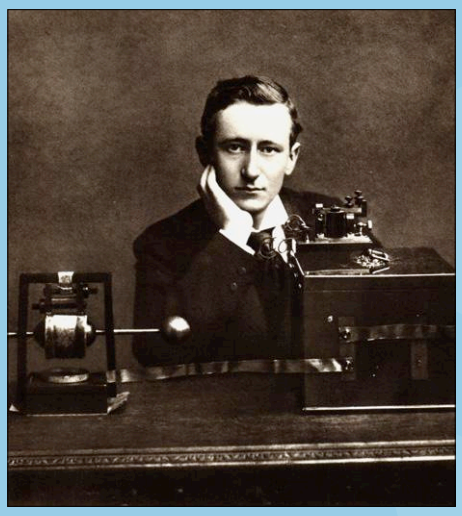
www.elektor.tv

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

ELEKTOR ETHICS

Open the Radio Spectrum!

When Guglielmo Marconi claimed a monopoly on the use of radio waves around 1900, he was thwarted right away by national states. In 1906, they signed an international treaty stating that radio spectrum be ruled by national sovereignty. Since then, the use of radio spectrum has been strictly regulated. Although there are good reasons to organise the spectrum, the way in which this is done hinders progress. New technologies make it possible to allow more users in certain frequency ranges. This is necessary to facilitate the explosive growth of wireless devices. In addition, wireless connections are an important link in the chain to give access to over 3 billion people who do not yet have an Internet connection.



Guglielmo Marconi. Photo: public domain.

By **Tessel Renzenbrink** (Netherlands)

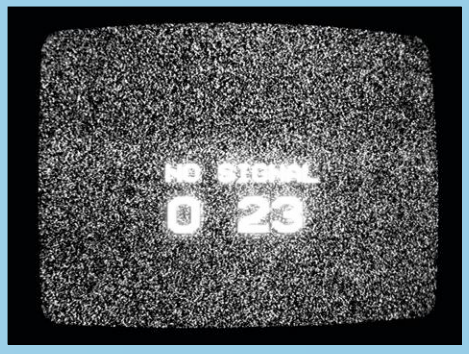
Marconi has the dubious honour of being the only person for whom an international treaty was drawn up to cut him off personally. The reason why nations acted in such a united fashion was their concern for national security. For the first time, radio technology enabled communication with ships at sea. An application for which states — especially in times of war — did not want to depend on Marconi's British company.

Monopoly

In an article [1] Robert Horvitz of the Open Spectrum Foundation gives a second reason why states demanded control over the spectrum: radio equipment was still very primitive at the time. In order to prevent malfunctions, it had to be operated by competent operators. Restricting the use of the spectrum should prevent amateurs from causing chaos and rendering it unusable for everyone. Everyone who wanted to broadcast had to apply for a licence, and so the strict spectrum regulation policy was born.

Horvitz says: "It is important to note that the claim to national sovereignty over spectrum is not based on the physical characteristics of radio waves. Radio is the same 'stuff' as light and there is no land in the world that claims ownership over the sunlight shining within its borders. It shows that government efforts to monopolise access to radio spectrum are not based on physics but are the result of a political agenda.

Radio operator receives a wireless message with Marconi's magnetic detector.
Photo: Archie Frederick Collins / public domain.



No Signal.
Photo: Liz Sullivan / Wikimedia Commons.

Spectrum auctions

Strict regulatory policies lead to inefficient use of spectrum. For example, the operating licences for certain frequency bands such as mobile telephony are distributed per auction. Telecommunications companies pay huge amounts of money for the licences and get an exclusive right of use for them. The exclusive right can mean that parts of the spectrum are hardly used or not at all. The position of governments is not neutral in this respect. The auctions generate billions of euros for the Treasurer. For example, the German government raised more than 5 billion euros in 2015, while in the Netherlands the figure was 3.8 billion euros in 2012.

A second point is that licences often cover a large geographical area. Telecommunications companies are particularly interested in urban areas because the costs of mobile network infrastructure can be recouped there. For sparsely populated areas, there is often no viable business model to pay for a finely meshed network. This leads to a lack of coverage or poor coverage in many sparsely populated areas.

TV White Spaces

Public policy can therefore lead to artificial spectrum scarcity. But technically speaking, it is now increasingly possible to share the spectrum in a smart way. An example of this is TV White Spaces (TVWS). These are unused TV frequencies that are used as a buffer to prevent different TV channels from interfering with each other. Because television broadcasters use high-power transmission equipment, these buffer zones are large. However, devices with lower transmission power can operate in these zones without disturbing primary users. Spectrum monitoring or geolocation databases can help white space devices find out which channels are free.

Everyone online

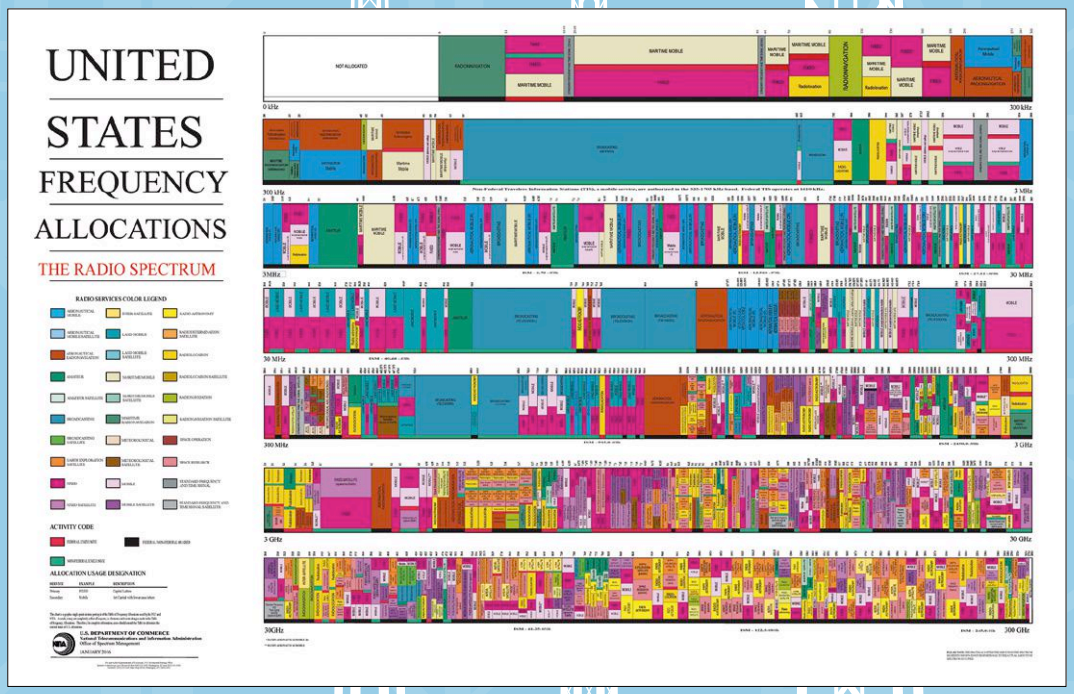
This kind of application to share spectrum is badly needed to realise Internet access. Currently, half of the world's population has no Internet access. And yet, more and more vital online services are emerging, such as e-health and e-government. To a large extent, it concerns people in sparsely populated areas where the construction of a cable network is not profitable. Wireless Internet is therefore an essential link in bringing more people online.
However, as already mentioned, it is often not profitable for telecom companies to roll out wireless networks in rural areas. This is called the 'last mile' problem: the high cost of connecting an individual end user to an existing network. The Internet Society (ISOC) therefore proposes an alternative way of working. ISOC was founded by 'the Fathers of the Internet' Vint Cerf and Bob Kahn, who invented the TCP/IP protocol. The association's goal is to make the Internet available to everyone and everywhere. In order to bring underprivileged areas online, ISOC has recently published a report [2] advocating community networks. These are networks set up and maintained by the community itself rather than waiting for a company to take care of them. In the report, this is called a paradigm shift because the last mile becomes the first mile. End users are suddenly no longer far away from the telecom business network, but are the centre of a local network. In order to ensure full access to the global Internet, community networks must ultimately be connected to the Internet backbone. And this is also possible because it is cheaper for a telecom company to connect a network than individual users.
Radio spectrum access is needed to set up wireless community networks. ISOC therefore calls on everyone to explore the possibilities of full use of spectrum sharing. As is often the case, technology is not the limiting factor, but stifled policy rules that have been drawn up for other times.

(160561)

Web Links

[1] Robert Horvitz: Geo-database management of white space vs. open spectrum. In: TV Whitespaces, A Pragmatic Approach, edited by E. Pietrosemoli and M. Zennaro. Available at http://wireless.ictp.it/tvws/book/.

Policy Letter: Spectrum Approaches for Community Networks. Internet Society. Available freely via: https://www.internetsociety.org/policybriefs/spectrum/



Unites States Frequency Allocations. Image: NASA / public domain.

welcome in your ONLINE STORE

EDITOR'S CHOICE

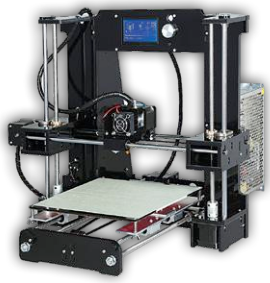
The Touch Board Pro Kit is aimed at creative people, who want to innovate with electronics, using primarily hearing and touch. As the kit describes itself, it marries programmed electronics (Atmel ATmega32U4) and touch sensors for immediate interactivity. The aim of this kit is thus the rapid development of simple applications. Fun, useful or just for pleasure, they use the conductive ink and other conductors such as the copper tape. These form switches and sensors (capacitive or proximity) controlled by the Touch Board, which is programmed like an Arduino. This kit seems to me perfect for anyone attracted to innovation and the use of electronic gadgets in a tactile environment. No matter their age or gender, all they need is to have some imagination.

Denis Meyer
Elektor Labs

www.elektor.com/touch-board-pro-kit

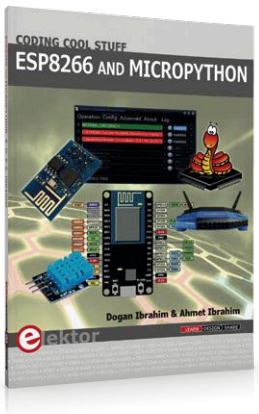
Elektor Bestsellers

1. Anet A6 3D Printer
   www.elektor.com/anet-a6



2. The Official ESP32 Book
   www.elektor.com/esp32-book
3. DIY Arduino Kitchen Scale
   www.elektor.com/ kitchen-scale
4. Motor Control Projects
   www.elektor.com/motor-control
5. HDMI Digital Microscope ADSM201
   www.elektor.com/adsm201
6. ESP8266 and MicroPython
   www.elektor.com/esp8266-book
7. Python 3 Programming and GUIs
   www.elektor.com/python3

ESP8266 and MicroPython



This book is an introduction to the ESP8266 chip and describes the features of this chip and shows how various firmware and programming languages such as the MicroPython can be uploaded to the chip. The main aim of the book is to teach the readers how to use the MicroPython programming language on ESP8266 based hardware, especially on the NodeMCU.

member price: £23.95 • €26.96 • US $32

www.elektor.com/esp8266-book

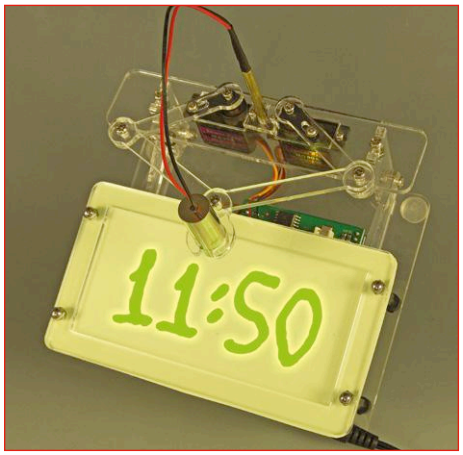
Motor Control Projects with Arduino and RPi

This book is aimed at students, hobbyists, and anyone else interested in developing microcontroller based projects using the Arduino Uno or the Raspberry Pi Zero W. The (tested) projects in the book cover the standard DC motors, stepper motors, servo motors, and mobile robots. Full program listings of all the projects as well as the detailed program descriptions are given in the book. Users should be able to use the projects as they are presented, or modify them to suit to their own needs.

member price: £26.95 • €29.66 • US $36

www.elektor.com/motor-control

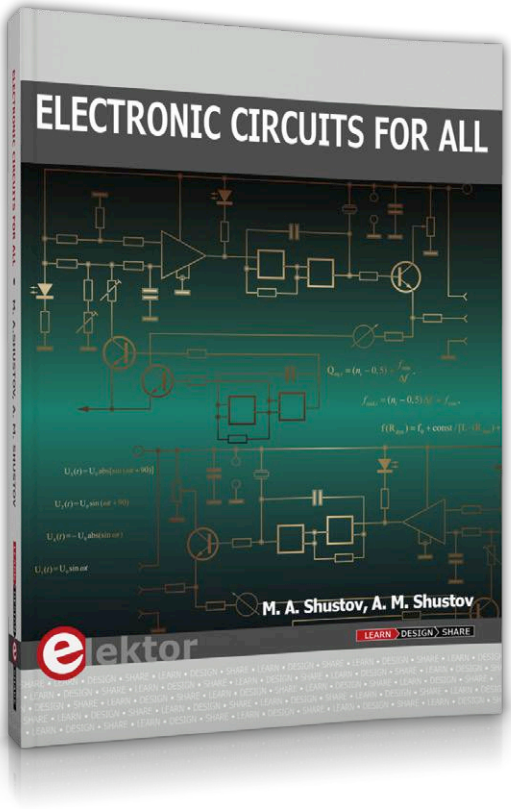
Laser Time Writer



This is an upgrade kit for Elektor's earlier published sand clock. Instead of writing the time in a layer of sand, it uses a laser module to plot the time onto a piece of glow-in-the-dark sticker material. The upgrade-kit comes with all parts required to transform your sand clock into a laser writer: all acrylic parts assembled in a precut plate, glow in the dark sticker material, laser module, pushbutton and wire, brass spacer and small mounting hardware.

member price: £35.95 • €40.46 • US $48

www.elektor.com/laserclock



400 CIRCUITS

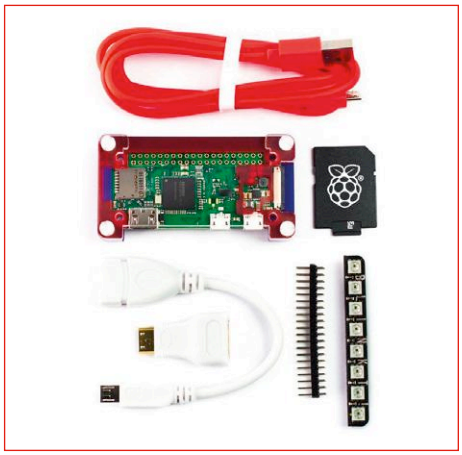
Electronic Circuits for All

This book contains more than 400 simple electronic circuits. The technical solutions presented in the book are intended to stimulate the creative imagination of readers and broaden their area of thought. This should allow readers to look beyond the horizons of possibilities and use ordinary electronic items in a new way. This book includes new and original radio electronic multipurpose circuits. The chapters of the book are devoted to power electronics and measuring equipment and contain numerous original circuits of generators, amplifiers, filters, electronic switches based on thyristors and CMOS switch elements.

MEMBER PRICE: £28.95 • €32.95 • US $39

www.elektor.com/electronic-circuits-for-all

Raspberry Pi Zero W Starter Kit

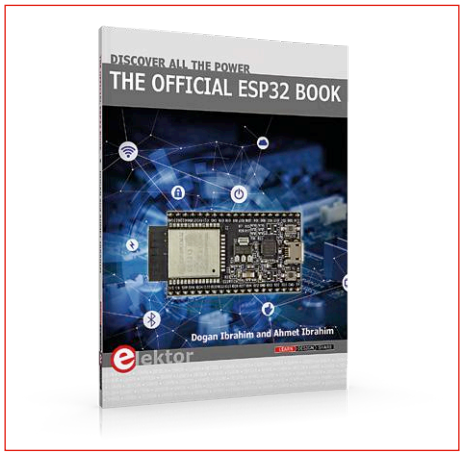


A great bundle of things you need to get up and running with the new Raspberry Pi Zero W! The kit includes a Blinkt! multicolour LED strip, an acrylic case, a 8 GB micro-SD card with NOOBS 2.2 pre-loaded, a male 2x20 pin header, 2 USB cables, a HDMI adaptor and of course a Raspberry Pi Zero W with single core CPU and built-in wireless LAN and Bluetooth.

member price: £41.95 • €47.66 • US $57

www.elektor.com/zero-kit

The Official ESP32 Book

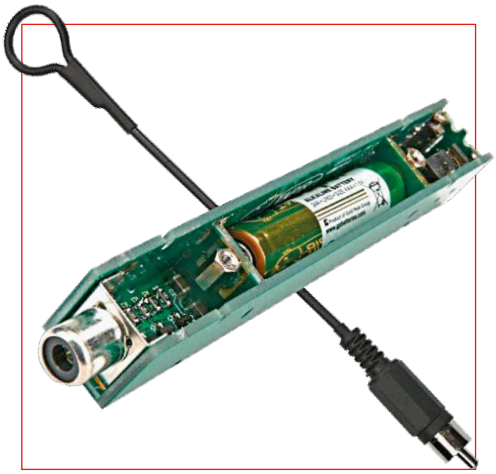


This book is an introduction to the ESP32 processor and describes the main hardware and software features of this chip. The book teaches the reader how to use the ESP32 hardware and software in practical projects. Many basic, simple, and intermediate level projects are given in the book based on the ESP32 DevKitC development board, using the highly popular Arduino IDE and also the MicroPython programming language.

member price: £27.95 • €31.46 • US $38

www.elektor.com/esp32-book

TAPIR E-smog detector



This ultrasensitive wideband E-smog detector offers you two extra senses to track down noise that's normally inaudible. Using the TAPIR (Totally Archaic but Practical Interceptor of Radiation) is dead easy. Connect the headphones and an antenna and switch it on. Move it around any electrical device and you'll hear different noises with each device, depending on the type and frequency of the emitted field.

member price: £19.95 • €22.46 • US $27

www.elektor.com/tapir

Hexadoku The Original Elektorized Sudoku

Traditionally, the last page of Elektor Magazine is reserved for our puzzle with an electronics slant: welcome to Hexadoku! Find the solution in the gray boxes, submit it to us by email, and you automatically enter the prize draw for one of five Elektor book vouchers.

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

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

Solve Hexadoku and win!

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

Participate!

Ultimately January 25, 2018, supply your name, street address and the solution (the numbers in the gray boxes) by email to: **hexadoku@elektor.com**

Prize Winners

The solution of Hexadoku in edition 6/2017 (November & December) is: **749AF**.

The €50 / £40 / $70 book vouchers have been awarded to: Jean-François Dedecker (Belgium); Monique Notot (France); Ricardo Claudino (Portugal); and Siem Oppe (Netherlands) .

Congratulations everyone!

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

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

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

Optimise Your Power Delivery Solutions

UPD360 - USB Power Delivery Controller

Microchip's Power Delivery (PD) Controller, UPD360, is a USB-IF certified USB Type-C™/PD Controller. UPD360 integrates the functional blocks required for USB Type-C and PD communications, which includes V_{CONN} FETs and port power controllers. UPD360 can operate in a standalone mode or a companion mode, interfacing to MCUs, embedded controllers or USB hubs over I^2C/SPI interface. UPD360 can be designed into applications that require USB connectivity, alternate protocols (viz. Display Port) and manage power (as a Source or Sink) up to 100 W over USB Type-C connectors.

Highlights

- USB Type-C and power delivery functionality
- Integrated power switch
- Integrated V_{CONN} FETs
- Dead battery support
- I^2C/SPI interface

microchip DIRECT
www.microchipdirect.com

MICROCHIP

www.microchip.com/UPD360

2ew18P

E-code for free admission

embedded-world.de/voucher

Nuremberg, Germany

27.2 – 1.3.2018

embeddedworld

Exhibition&Conference

... it´s a smarter world

DISCOVER INNOVATIONS

Immerse yourself in the world of embedded systems and discover innovations for your success.

embedded-world.de

Media partners

Markt&Technik – Die unabhängige Wochenzeitung für Elektronik

DESIGN&ELEKTRONIK – Know-how für Entwickler

Elektronik – elektroniknet.de – Fachmedium für industrielle Anwender und Entwickler

Elektronik automotive – Fachmedium für professionelle Automobilelektronik

Computer & AUTOMATION – Fachmedium der Automatisierungstechnik

SmarterWorld – Solutions for a Smarter World

MEDIZIN elektronik – Fachmedium für Elektronik in der Medizintechnik

computer-automation.de

elektroniknet.de

MEDIZIN-UND-elektronik.DE

Exhibition organizer

NürnbergMesse GmbH

T +49 9 11 86 06-49 12

F +49 9 11 86 06-49 13

visitorservice@nuernbergmesse.de

Conference organizer

WEKA FACHMEDIEN GmbH

T +49 89 2 55 56-13 49

F +49 89 2 55 56-03 49

info@embedded-world.eu

NÜRNBERG MESSE