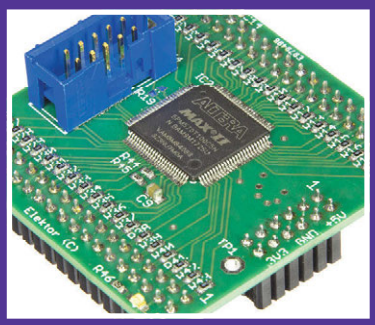
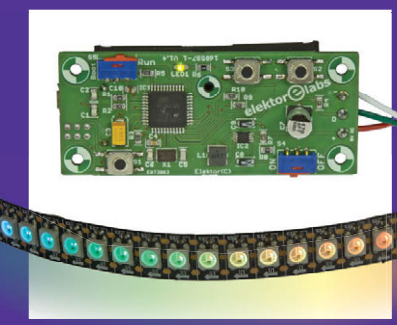
Edition 3/2018 | May & June 2018
www.elektormagazine.com

elektor

LEARN DESIGN SHARE



CPLD Breakout Board in DIL Format
For programmable logic projects



Wearable LED Controller
Light effects for the disco

10-MHz Reference
with satellite precision

VFD-tube Clock with ESP32

Internet time on vintage displays

10-MHz Reference • 3-Way Display Alarm • **About Battery Chargers, Choices and Electronic Designers** • Acoustics, GUI and Fingerprints • **Arduino board provides serial/USB adapter** • Close-to-Universal IR Remote Control • **CPLD Breakout Board in DIL Format** • electronica Fast Forward 2018 • Elektor Labs Pipeline • **EMC Limit Values and CE Declaration** • Err-lectronics • **Everything (almost) everything you need to know about… Bluetooth.** • Floranium • **For programmable logic projects** • Hexadoku • **Homelab Helicopter** • LoRa Telemetry Projects • **Ltank Lego Robot** • My Tiny Radio • **Our Vulnerable Digital Society** • PAM8302A Audio Amplifier • **Scrolling Message Display** • Simple LC Meter • **Struggling with LED Snake Lights** • The Connected Greenhouse • **The Elektor Mini Counter** • The NXP Cup 2018 • Universal I²C Bus Isolator and Level Shifter • **VFD-tube Clock with ESP32** • Wearable LED controller • **Wireless Protocols: a Panorama** • … and more

pico®
Technology

PicoLog® 6

Data logging

Straightforward from the start

NEW

Real-time data collection and display

Visual logger and channel setup for easy configuration and viewing

Available for Windows, macOS and Linux

Virtually unlimited logging capacity to PC

Robust database format minimizes data loss and corruption

Supports all current Pico data loggers

Try out the software for free!

Download now from www.picotech.com/A61

Elektor Magazine
Edition 3/2018
Volume 44, No. 489
May & June 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

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

www.eimworld.com
Printed in the Netherlands

In the pipeline, in the box

If you are not in the electronics industry, or without a formal qualification in the field, it may be good to realise that electronics as a pastime would be non-existent without its wide diversity of manufacturers of equipment, tools, PCBs, components, and software in many guises from CAD to ARM firmware.

Some readers have expressed a desire to see a version of *Elektor Magazine* with zero advertising and must have been happy to see the number of "ad pages" decreasing considerably compared to say 10 years ago. Be that as it may, advertisers and the industry in general remain a gateway to services, products and all manner of tools that are essential to enjoy electronics as a pastime, or pursue it at any level in education. Companies also continue to contribute to our turnover — the other strong factors being our loyal subscribers and product sales. So where are the suppliers if they do not appear in print?

I am happy to say that *Elektor Magazine* is attracting an increasing number of companies — big and small, "print" advertiser or online only — wishing to run projects in close cooperation with our labs and editorial staff. Without their input we would not have been able to publish about, or retail, hugely successful projects and tools like the Andonstar USB Microscope, the Nixie Clock, or the Anet 3D printer over the past months. All three deals, and more, have taken our labs, logistics and sales teams much effort to arrange exclusively for you — or should I say 'for your pleasure'.

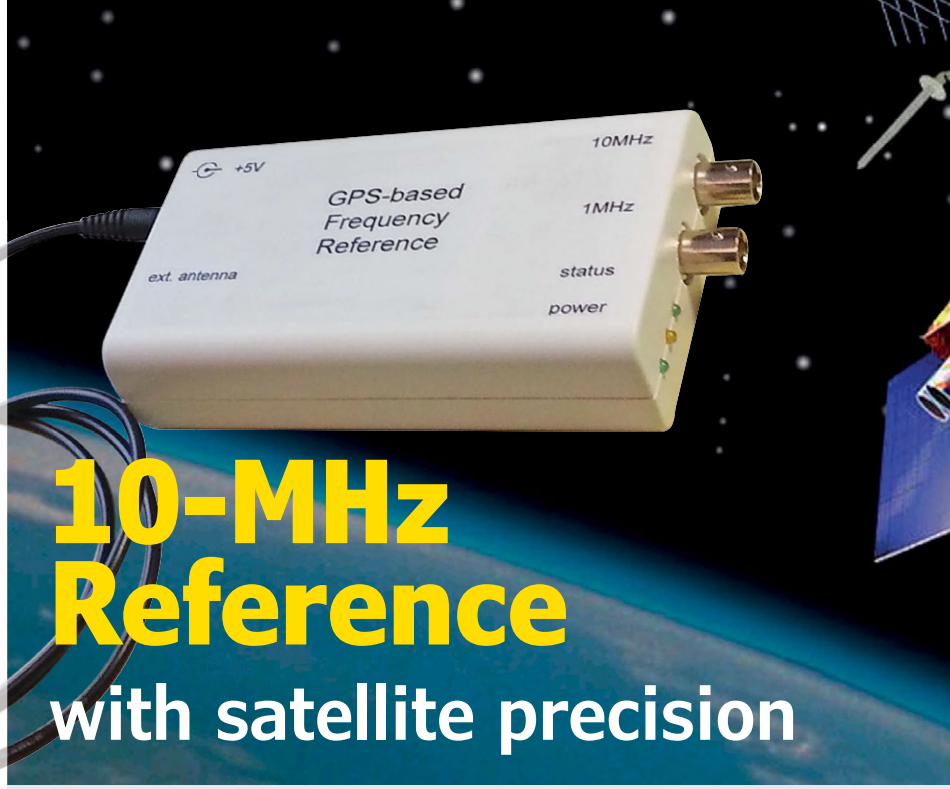
When in the past old hands with a fully loaded workshop had good reason to baulk at seeing basically the same ad pages for years on end, there is now a dynamic scene in terms of projects and lab tools promoted in our weekly newsletter and retailed through the Elektor Store. Product advertising still appears in good old print though, not just on the expected pages 128 and 129 but also in the "From the Store" textboxes found in the construction project articles. More is in the pipeline.

Jan Buiting, Editor-in-Chief

This Edition

Volume 44 – Edition 3/2018
No. 489 May & June 2018

Regulars

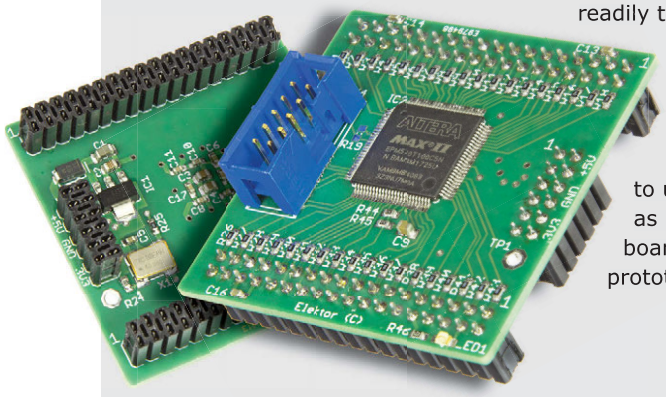


10-MHz Reference
with satellite precision

CPLD Breakout Board in DIL Format

For programmable logic projects

CPLDs tend to have a large number of pins practically inaccessible to the soldering iron and so do not lend themselves readily to use in homelab and experimental projects. The BoB described here brings out the connections to header sockets, allowing you to use the CPLD as easily as a DIL IC on prototyping boards or straight in prototypes.



Features

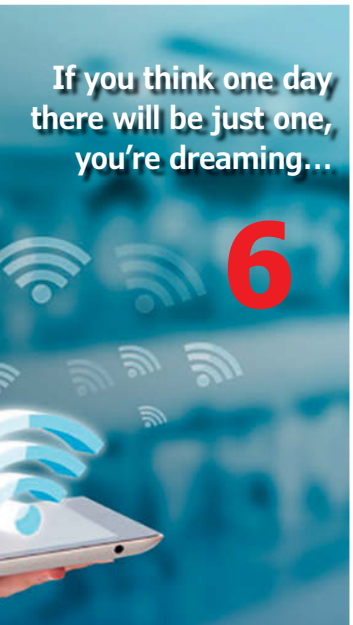
Projects

elektor magazine

Scrolling Message Display

512 LEDs controlled over Wi-Fi via an ESP-12F

This project lets you scroll text on a bank of eight 8 × 8 LED matrix displays. It uses a Wi-Fi module of type ESP-12F (based on the ESP8266), programmable with the Arduino development environment. Thus, from a smartphone or any other device that uses Wi-Fi, it is possible to send to an ESP8266 web server the text to display, the scrolling speed and the brightness.



Next Editions

Elektor Magazine 4/2018

Analogue Alternating Linear LED Fader • Opamp Circuits • VHDL Course • PWM as a DAC • Q&A: Soldering • Audio DAC Update • easy music • POV fidget spinner • DIY solder station • GPS-based 10-MHz Frequency Counter • Stencil Frame for Reflow Soldering.

Elektor Magazine edition 4 / 2018 covering July and August is published around 13 June 2018. Delivery of printed copies to Elektor Gold Members is subject to transport. Contents and article titles subject to change.

Elektor Business Edition 3/2018

Elektor Business Edition issue 3/2018 has a focus on **Sensors and Measurement**. Contributions come from companies, research institutions as well as private authors, covering professional approaches to analogue and digital measurement systems and techniques. Plus you'll find fresh instalments of all the EBE regulars like Infographics, Operation Marketing, Our Business, and Business Store.

Elektor Business Edition issue 3 /2018 is published around 5 June 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.

Wireless Protocols: a Panorama

If you think one day there will be just one, you're dreaming…

By **Robert Lacoste** (France)

You've probably thought (nay, dreamed) about getting several electronic devices to communicate between themselves via wireless links, right? You've probably asked yourself why there are so many protocols on the market: Bluetooth, Bluetooth Low Energy, Wi-Fi, Zigbee, LoRa, Sigfox, Thread, 6LowPan, MiWi, Z-Wave, ANT, plus a few others I've missed. Why can't these guys get their act together? Why can't they just confine themselves for once to the "best" protocol? This article looks at the advantages and disadvantages of the various wireless links.

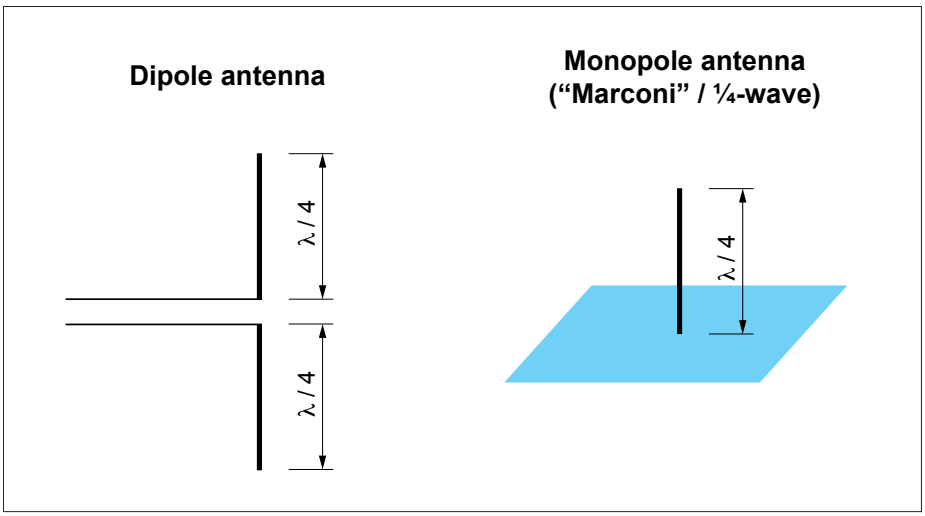


Figure 1. The length of an antenna depends directly on the wavelength of the carrier. Represented here are dipole and monopole antennas, the last needing a ground plane.

To talk about a concrete example, let's take a classic application case: home automation. Let's imagine that you want to design a system for your home, comprising a central control system using a microcontroller card (maybe an Arduino or a Raspberry Pi?), lots of sensors (temperature, light, movement detectors, etc.) and a few relays to control lighting or blinds. As we're in the 21st century, you'd also like to control all this with your smartphone, and also check the temperature of the aquarium from your work (during breaks, of course!). Naturally your partner does not want you running wires everywhere, so everything must communicate by radio, and the sensors will have to have batteries. What protocol will you choose for such a network, and why?

Radio?

Let's start with some basics so we know what we're talking about. I promise I'll keep it simple, you'll be able to read this without needing an aspirin. A wireless link will definitely need two antennas, which have the role of interfacing between an electrical signal and an electromagnetic wave. The electrical signal is a sine wave (the radio frequency 'carrier'), modulated to transmit zeros and ones. According to the type of modulation, the resulting signal occupies a certain range of frequencies around the frequency of the carrier. Why is this important? Because regulations don't let you do just anything you want: the frequency bands that you can use are precisely defined. In Europe, the reference document for SRD (Short Range Devices) and licence-free frequency bands has the lovely name of ERC/REC/70-03. You can easily find it on the Internet. This document defines the well-known 433 MHz, 868 MHz, 2.4 GHz, etc. ISM (industrial, scientific, medical) bands. Remember that this document defines not only the usable frequency bands, but also the permitted channel widths, the transmitted power and the maximum time of use of the channel. A value of 1% for example corresponds to 36 seconds of use per hour. It also precisely indicates the standards to be used. So "free" band does not mean rubbish band — its use is free, but only under restrictions which everyone must respect.

Wavelength?

Let's go back for a second to the carrier frequency f (in hertz). This frequency is directly related to the length of the electromagnetic wave in free space, called λ (lambda), because this wave propagates at the speed of light $c = 3\times10^8$ m/s. The relationship is simply:

$$\lambda = c / f$$

If you do the math, you will find for example that the wavelength of a signal of 868 MHz is 34 cm and for a signal of 2.4 GHz the wavelength is 12.5 cm. This wavelength is physically very important; in particular it determines the size of the antennas. In practice a classical antenna will have a length of a quarter of a wavelength (**Figure 1**), so 3 cm at 2.4 GHz or 8.5 cm at 868 MHz. It is true that antennas can be smaller than this, using materials like ceramics, but this is beyond the scope of this article. Just remember that the lower the frequency, the longer the antenna…

Link budget, what's that?

Lovers of radio, among many others, are very enamoured of a somewhat exotic unit of measure, the decibel. By definition the dB is the logarithm of a ratio of two powers multiplied by 10:

$$dB = 10 \log(P_{out} / P_{in})$$

A value in dB thus has no dimensions, it is a ratio. If you want to quantify an absolute quantity like a power, you have to specify it with respect to something, and indicate what by a lower case letter after dB. For example, a dBm is, by definition, a power relative to 1 mW:

$$dBm = 10 \log(P) \text{ [mW]}$$

But why do we complicate this with logarithms? Because this allows us to calculate easily the gain of a chain of circuits: all we have to do is add up the gains in dB. If you want to understand why, recall that the logarithm of a product is merely the sum of the logarithms…
Let's go back

to our foundations, and to the radio. Imagine you have a transmitter of power P_{tx}. The signal is transmitted by an antenna, which has a gain G_{tx}. This signal is attenuated between the two antennas by a factor $Loss_{Fspace}$, then it reaches the receive antenna with gain G_{rx} before being processed by the receiver. What is the level of the received signal? It's very simple thanks to our dB (**Figure 2**):

$$P_{rx,\,dBm} = P_{tx,\,dBm} + G_{tx,\,dB} - Loss_{Fspace,\,dB} + G_{rx,\,dB}$$

The receiver can correctly receive the message if this received power is sufficient, that is to say above what we call the sensitivity of the receiver.

The free space loss or free space attenuation, which is the attenuation of the signal between two antennas, depends on many factors: distance, obstacles, polarisations, etc. Let's stick with the simple case called "free space" which describes two antennas facing each other in space. Free space loss depends on the distance between the two antennas. If we double the distance, the loss quadruples (actually it increases by a bit over 6 dB, because 10 log(4) = 6.02). Why? Simply because the emitted power is effectively spread over the surface of a sphere, of which the surface area quadruples if the radius **doubles**. More surprisingly, this free space loss depends also on the frequency, with the same relation: 6 dB more loss if the frequency doubles. Why? not because the air "attenuates" higher frequencies more, but simply because the antenna gets ever smaller as the frequency rises. (For those who like formulae, look up Friis formula on the web).

At the same distance, a 2.4-GHz antenna captures a smaller area than an 868-MHz antenna, simply because it is three times smaller. So you can see that everything is important:

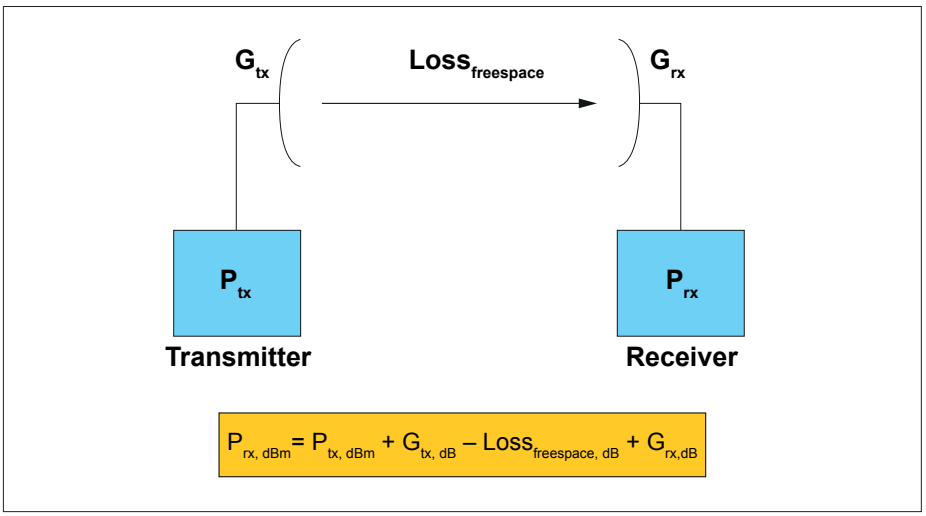


Figure 2. Expressing a link budget in dB allows simple calculation of the received power as a function of the transmitted power, the gains of the antennas and the free space loss.

rather choose a lower frequency to transmit farther, but with the constraint that it's more difficult to fit the antenna into a smaller device.

Sensitivity and data rate

Obviously with radio, you want the transmission to carry as far as possible, and in our example some of your sensors are in the cellar or the garage. How do you make a radio with maximum range? We have seen the lowering the frequency is a solution limited in practice by the size of the antennas. Of course we can always transmit a more powerful signal, but the transmit power is limited by the available power supply, and increasing the transmit power will reduce battery life. You can also use "better" antennas, that is to say those with a better gain, but it's a lot easier said than done, because an antenna with a high gain is also very directional. Our last solution is to use a lower received signal, which means **improving the sensitivity of the receiver**.

On what does this sensitivity depend? Is it possible to construct a receiver which is as sensitive as you wish, if you spend a bit of money on it? Well… no, and Claude Shannon showed that in the 1940's. Take a deep breath and cast your eye over the formula below which gives the sensitivity of any receiver:

$$\text{sensitivity} = -174 + NF_{(dB)} + 10\log(\text{datarate}_{(bps)}) + E_b/N_{0(dB)} \quad [\text{dBm}]$$

What does all this gibberish mean? Let's look at the different factors of this expression in simple terms. The number −174 is what we call the normalised thermal noise, and the only way to reduce it is to refrigerate the whole system, which is a bit difficult for your home automation installation. The term *NF* is the noise factor of your receiver, that is to say the amount of noise which it adds to the received signal. You guessed it, to make this better you need to use better electronics: more costly components, low-noise preamplifiers, etc. but at the very best you'll get this down to 0 dB. The last parameter, called E_b/N_0, is related to the quality of the demodulator and its ability to retrieve the 0s and the 1s from a noisy received signal. It's thus related the method of encoding the bits (with or without error correction bits), to the type of modulation used and also to the error rate which can be tolerated by the application. To improve it means using a more complex radio, with lots more signal processing, so more expensive.

Nothing very surprising so far, an expensive radio is better. The last parameter is a little more surprising: **the sensitivity of a receiver is directly related to the data rate**, that is to say the number of bits transmitted per second (and not, as you will often read, to the bandwidth). Reduce your data rate by a factor of 10 and you gain 10 dB of sensitivity, which corresponds to a nearly doubled range. Why? Intuitively, because to transmit a bit during ten times more

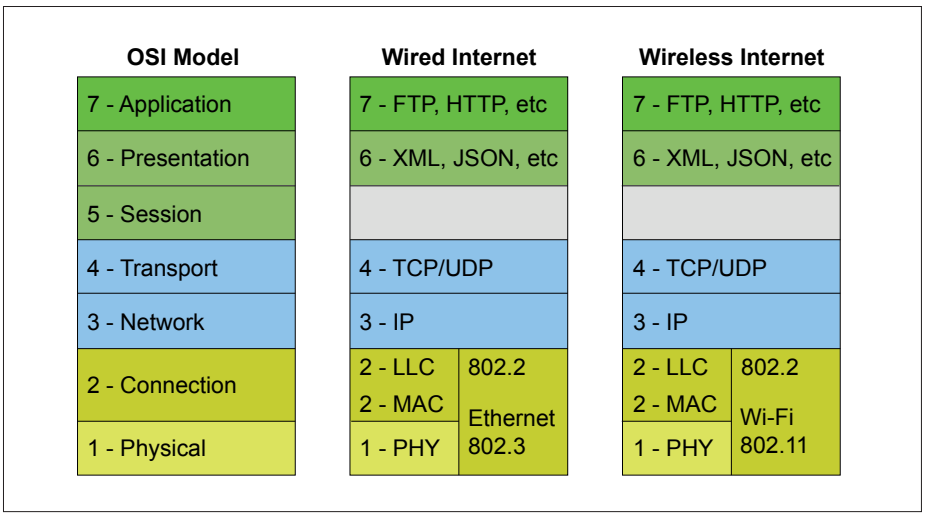


Figure 3. The OSI model is 40 years old, but it's still a reference, for example to represent the protocol layers of the TCP/IP and Wi-Fi world.

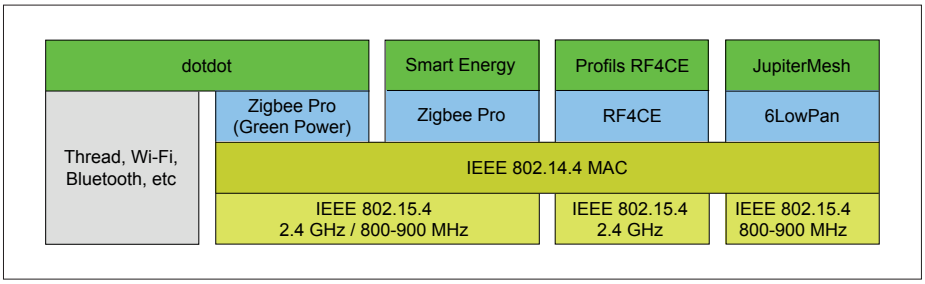


Figure 4. The Zigbee world is, let's say, a little complicated. Lots of variants, but that's maybe one of its strengths.

time means you are using ten times greater energy per bit, with the same transmit power. The receiver therefore receives ten times more energy to decide if it's a 0 or a 1.

As you will understand, to increase the range, you must reduce the data rate: it is a lot easier to send a message over 1 km if you must transmit 100 bits per second than if your data rate must be 10 Mbps. You may not have thought about it before, but this is why your smartphone surfs the net less and less fast as you move further from the town centre. The network detects that the communication is going to be cut because the received signal is too weak, and automatically changes the communication to a mode with lower data rate. This improves the sensitivity of the receiver and allows the communication to continue, even if you moan a bit!

The smartphone and the box...

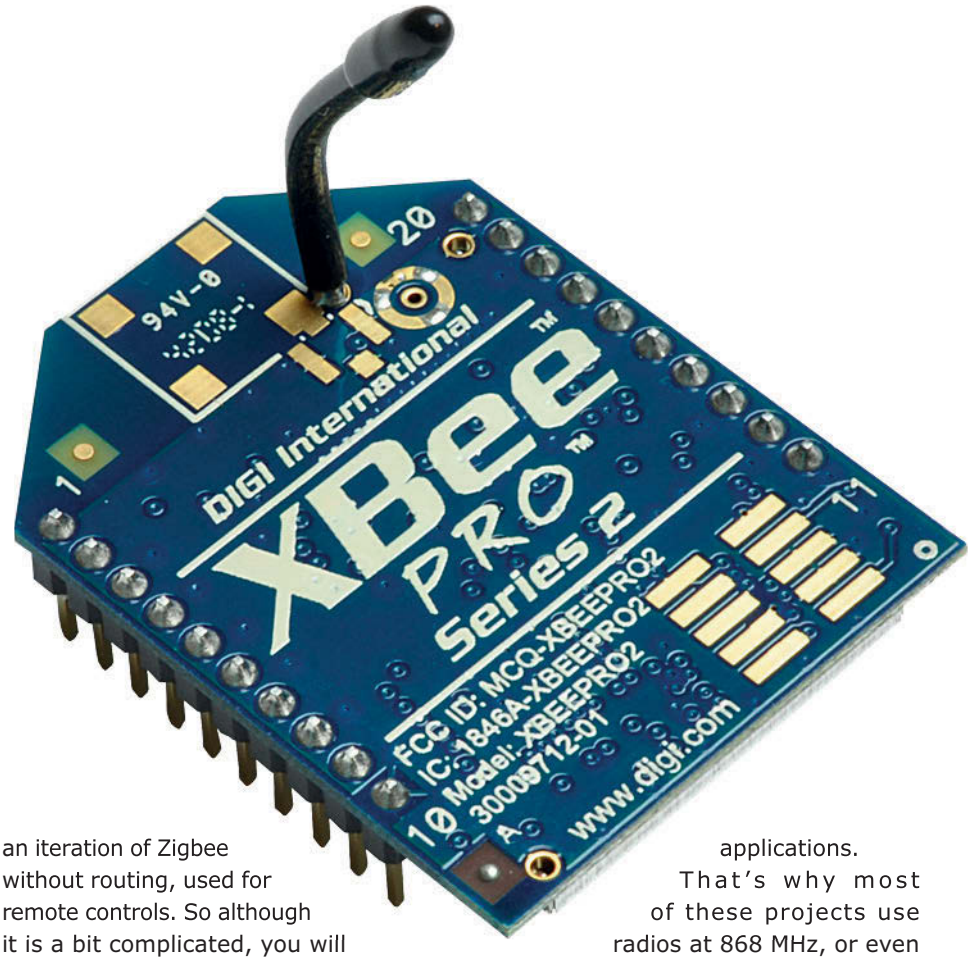
On the theory side, you now know enough. So let's get back to the practical stuff. We'll start where there's not much choice: how do you connect your wireless home automation installation to your smartphone or to the internet via the box from your service provider? You're restricted here to the protocols supported by these devices, and there are not many.

Your box probably only supports **Wi-Fi**. I'm not going to explain to you what it is, but just remember that Wi-Fi is the commercial name of an American standard, IEEE 802.11, available in several versions, from 802.11b (11 Mbps) to 802.11ac (up to 1.7 Gbps). On the frequency plan, Wi-Fi principally uses the 2.4-GHz band, but also the 5-GHz band, even 60 GHz. Wi-Fi is optimised for very high data rates, and thus does not have much range, a few hundred metres in open air and a few rooms in a house. Obviously, the more you need high data rates, the less range Wi-Fi has. Its advantage is its omnipresence in most devices and its high data rate, but you pay for that with a moderate range and a voracious energy consumption: embedded Wi-Fi components such as the CC3100 from Texas Instruments or the well-known ESP32 from Espressif easily consume 50 to 100 mA in receive and more than 200 mA in transmit mode. Wi-Fi is also a little limiting for the user: you must provide an identifier and a password to connect to the network, or press buttons to activate the WPS pairing mode. Finally, Wi-Fi is only a "low-level" protocol and does not specify what we call the application layer.

In this regard, communication protocols have long been specified in layers, based on the old OSI model (**Figure 3**). The advantage of such a model is to facilitate the specification and testing of a protocol, layer by layer, but also to replace one layer with another, without having to change the other layers. Wi-Fi for example is a layer 1 and 2 protocol like Ethernet, on which you can use TCP/IP (respectively layer 4 and 3) and applications like HTTP, FTP, etc. (layer 7). Thanks to the independence of these layers, you surf the internet in the same way whether you connect over Wi-Fi or Ethernet.

Back to the foundations again. To connect your home automation controller, you could use Wi-Fi (via your home box or directly), or almost certainly **Bluetooth**. Initially conceived for the transfer of audio between a mobile telephone and a headset, Bluetooth has since been used for a multitude of applications, in particular since the arrival of the **Low Energy** variant (BLE) with version 4.0. You will also find in this magazine an article of *Questions & Answers* on Bluetooth, so I won't say more about it here. Nevertheless you should know that this protocol uses the same 2.4-GHz band, with lower data rates (1 to 3 Mbps for classical modes). It consumes much less than Wi-Fi partly thanks to the usually lower power transmission and partly thanks to a very effective sleep mode. But even if a Bluetooth radio consumes several tens of mA while working, it's possible to design devices powered by a button cell, which only transmit a short message every few seconds for some months, simply

because the communication only lasts a few tens of microseconds for each message. In practice, the range of Bluetooth is from a few metres to maybe 100 metres in free air, mainly according to the transmit power. Unlike Wi-Fi, Bluetooth is a protocol which covers all the layers of the OSI model; hence it has a very high compatibility which is not the least of the attractions of this protocol. Finally the evolutions of Bluetooth version 5.0 are even more interesting, so cast your eye over the *Q&A* article.



Other short distance solutions

If you don't need a connection with a smartphone or a box, what are the other solutions to connect your devices by radio over a short distance? There's almost an embarrassment of choices. Still in the 2.4 GHz band (well, mostly), you will find **Zigbee**. This protocol is somewhat unique in terms of layers: it relies on another protocol for the lower layers, called IEEE 802.15.4 (**Figure 4**), with a rate of 250 Kbps. On top of this, Zigbee defines another protocol, called **Zigbee Pro**, which takes care of routing of messages via a mesh network algorithm. This protocol allows the transmission of messages from one node to another, relaying them in a transparent manner via intermediate nodes if a direct link is impossible. The transmission path is automatically reconstructed if necessary. Finally, Zigbee integrates an application layer which defines a standardised method of exchanging messages between devices: switches, lamps, electricity meters, etc. This application layer has just been renamed **dot.dot**. Zigbee assures a good interoperability and efficient transfer of messages thanks to its mesh network, but in reality the situation is a little more subtle. At first glance the idea of a mesh network is great, but it means the nodes which relay messages must be permanently in listening mode, which means they must be mains powered. This greatly reduces the applicability of ZigBee for some projects, even if it allows the use of products with very low power consumption if they don't have to relay messages. So Zigbee has had a bit of a chequered career: there are many variations of Zigbee at the routing and application layer levels, as well as frequencies used (Zigbee is also specified at 868 MHz et 915 MHz for the United States). For example **RF4CE** is an iteration of Zigbee without routing, used for remote controls. So although it is a bit complicated, you will easily find Zigbee chips or modules to use this protocol and create virtual links between products, for example Xbee modules at Digi [3].

The story of Zigbee gets ever more complicated, because other protocols use the same lower layers (IEEE 802.15.4), and also use the same integrated circuit transmitters and receivers as Zigbee without being Zigbee, if you see what I mean. In this family, you will find **6lowPan** which uses routing algorithms from the IP world on top of the IEEE 802.15.4 radio layers. The big advantage of this is that any 6lowPan device can be connected directly internet via a bridge, and have its own IP V6 address. The radio protocol touted by Google, **Thread**, is itself an application layer which sits above 6lowPan and IEEE 802.15.4. Confused? That's normal, have a look at **Figure 4**.

In summary, all the protocols built on IEEE 802.15.4 have the same radio characteristics (range in the order of one or two rooms in a house), but they differ in the network and application layers, so it's up to you to choose which one best meets your needs (interoperability, routing).

All these protocols at 2.4 GHz have just the right range for home automation applications. That's why most of these projects use radios at 868 MHz, or even 433 MHz. There, the choice is considerable. In fact a number of projects use generic integrated circuit transmitter/receivers and forge their own radio protocol. You will also find proprietary protocols offered by chip manufacturers, designed to make your life easier: **MiWi** from Microchip, **SimpliciTI** from Texas Instruments, etc. All these protocols have a better range than those at 2.4 GHz, but lower data rates (typically a few tens of kbps). Finally, **Z-Wave** has its adherents thanks to its strong interoperability. It's also a protocol at 868 MHz for its European version which supports relaying between nodes; it is rightly aimed at home automation applications. On the other hand, Z-Wave is only available from one integrated circuit manufacturer Sigma Design (ex Zensys).

LPWAN

Let's hammer it home once more: to get more range, the easiest solution is to reduce the data rate on your link. This approach, pushed to its limit, gave birth to networks collectively known as LPWAN (Low Power Wide Area Networks). The idea is to reduce the data rate sufficiently to have a range of several tens of kilometres over line of sight while

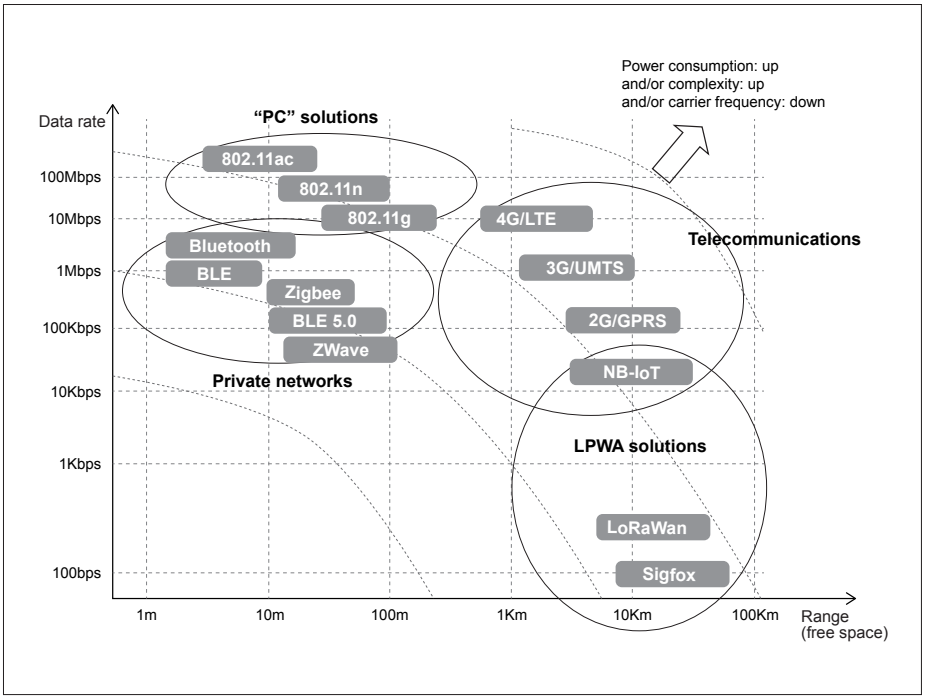


Figure 5. An attempt at synthesis. The curves show that you can swap range against bit rate, with associated electronic and power consumption costs

respecting the transmit power limits such as those at 868 MHz. The first to offer this approach commercially was probably **Sigfox**. More than a protocol, Sigfox is a telecommunications operator, and has deployed its communication bridges in multiple countries. With a subscription, a Sigfox compatible device can communicate with a server on the internet without installing receivers for each application. Details of the modulation used is a closely guarded secret, but the data rate is deliberately kept very low: 100 bits per second. Obviously the transmission of a message, even a short one, takes a finite time, so you're looking at an appreciable energy consumption per message, but that's the price you pay to get such a good range. The other LPWAN solution is **LoRaWAN** which lets you use the network of a LoRaWAN operator like Objenious or Orange in France, KPN in Holland, Proximus in Belgium or Digimondo in Germany, just to name a few. The model is thus similar to that of Sigfox in that you are working through an operator. But LoRaWAN lets you create your own private network: you can buy a LoRaWAN bridge for your home or your big company, and create your own private network without paying any subscription. Finally LoRaWAN relies, at the physical layer, on a modulation called **LoRa** which is owned by Semtech, until now the only supplier of LoRa components. There's nothing stopping you using these components without LoRaWAN, to create your own home automation protocol. Semtech has recently offered a component to do LoRa in the 2.4-GHz band, called SX1280.

In conclusion...

So there you are, I have finished my small tour around the better known radio protocols for home automation applications or the Internet of Things. To get back to my first question, you will now have seen that sadly, there is no such thing as the "best" protocol. Physics is physics, and you can only get better range if you make concessions on the data rate, or the size of the antennas, or the cost of the electronics. There are thus many solutions for every need. Need to transmit several tens of megabits per second? Wi-Fi is your friend, but the range is low and the power consumption is tall. Need to cover large distances, but only transmit a few bytes from time to time? Look at LPWAN solutions. You need an intermediate solution? One of the protocols in the 868-MHz band, more or less proprietary, will probably suit you. I've tried to summarise all this in a diagram, so look at **Figure 5**. Apart from the long range solutions, if you're looking for the most universal protocol, it's Bluetooth that wins. Maybe that's why three billion Bluetooth chips are sold every year!

(160653)

Web Links

[1] CC3100: www.ti.com/product/CC3100
[2] ESP32: www.espressif.com/en/products/hardware/esp32/overview
[3] Digi/Modules XBee: www.digi.com/products/zigbee
[4] Z-Wave: en.wikipedia.org/wiki/Z-Wave

reichelt elektronik

Technology connects.

- More than 45 years of experience
- High availability and reliable delivery
- More than 90,000 products

PROFESSIONAL SHOP AND LAB TECHNOLOGY
FULL POWER AT THE WORKPLACE!

RND lab

Programmable lab power supply

The output voltage can be set from 0 to 30 V DC and the current simultaneously from 0 to 5 A. The fine adjustment in steps of 10 mV and 1 mA allows precise working.

- Intelligent, temperature-controlled ventilation
- Constant voltage/constant current operation
- Low residual ripple
- Large LED display for output values, device status and memory

Order no.: RND 320-KD3005P

PRICE TIP £ 80.20 (£ 66.83)

(Please order compatible mains cable: NKSK UK 180, see below)

TRMS digital multimeter

- 0.09 % base accuracy
- AC/DC: 1 mV–1000 V
- 6000 counts
- Smoothing mode for more stable readings in case of variable input signals

FLUKE

YOU HAVE NEVER MEASURED THIS ACCURATELY BEFORE!

EN 61010-1 CAT III 1000 V

EN 61010-1 CAT IV 600 V

Order no.: FLUKE 179 instead of 308.71

£ 283.83 (£ 236.53) **SAVE 8%**

The power supply unit for professionals

Programmable lab power supply, 1–60 V, 0–15 A

Manson

PC SOFTWARE

USB

- 3 freely programmable memory locations
- USB interface
- PC software for programming cyclic operations
- Remote control switching
- Display accuracy: ±0.2 % + 3 digits
- Efficiency: 88 %

BROAD CURRENT AND VOLTAGE RANGE

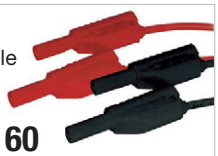
Order no.: HCS 3604 USB

£ 351.38 (£ 292.81) **BEST SELLER**

(Please order compatible mains cable: NKSK UK 180 ▼)

Safety measuring leads, 1 m

- 1 mm² cross section
- Stackable
- up to 20 A
- Connector: 4 mm



Order no.:

ML 2612-100RT red

ML 2612-100SW black

each £ 3.60 (£ 3.00)

ORDER NOW

AC power cable, 1.8 m

- for Great Britain
- UK plug > IEC socket

NKSK UK 180 £ 3.33 (£ 2.78)

Daily prices! Price as of: 29. 3. 2018

Onlineshop languages:

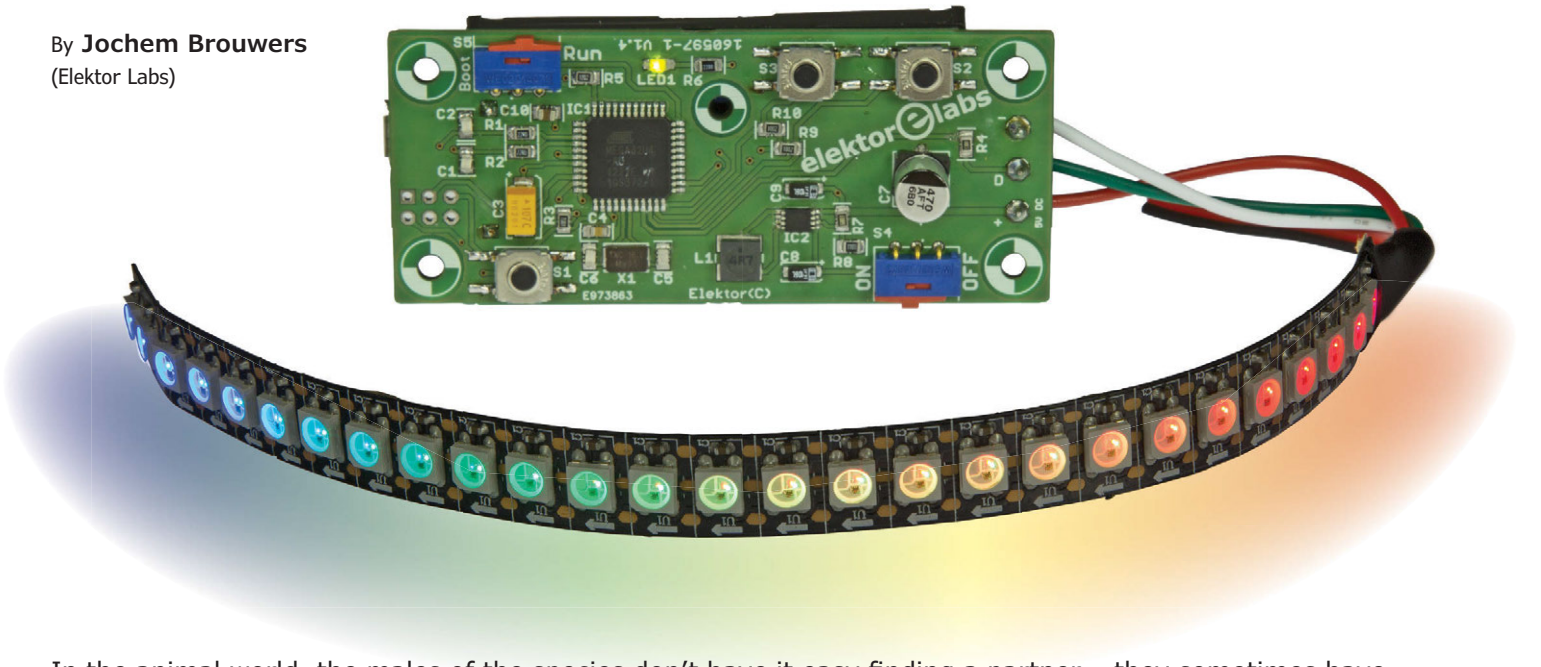
reichelt elektronik GmbH & Co. KG, Elektronikring 1, 26452 Sande (Germany), Phone: +44 203 808 95 25

www.reichelt.co.uk

ORDER HOTLINE: +44 203 808 95 25

Wearable LED controller
light effects for the disco

By **Jochem Brouwers**
(Elektor Labs)



In the animal world, the males of the species don't have it easy finding a partner – they sometimes have to prove in fierce (mock) fights with competitors who is the strongest, or need to show their prettiest side and/or need so sing as beautiful as possible… In comparison, Homo Sapiens, and then particularly Homo Electronicus, has it much easier these days – he calls in the support of his semiconductor friends.

All silliness aside – this circuit has not been specifically designed to make an impression on potential partners; the author needed a small controller circuit to drive LED strips. In the end, the final circuit turned out to be so compact and frugal with batteries that the designation 'wearable' is fully justified and the controller can be used without any issues for groovy brooches and such like.

Characteristics

- Programmable using Arduino IDE
- Compact circuit board
- Operates 5 hours on 2 AA batteries
- Supports several different LED strips (see text)

General considerations

To make a reasonably long, uninterrupted use possible, two AA batteries were chosen to serve as the power supply source. From that decision follows, more or less, that the circuit board for the controller should not be much bigger than the batteries.

Because the circuit had to be versatile, the author chose to add Arduino IDE support to the controller. It is then relatively straightforward to program the controller for your own applications.

The choice of controller itself, the heart of the circuit, has also received considerable consideration. Basically the choice fell on a microprocessor from Atmel, because these are easily programmed using an Arduino bootloader. The author first considered using an Atmega328P (the same one as is used in the Arduino Uno), but finally decided to use the Atmega32U4, which is used in the Arduino Leonardo. This processor also has a bootloader available, and also has a built-in USB/serial interface so that for USB communications no additional components are required.

The schematic

The schematic for the LED controller is a relatively straightforward affair, so we didn't feel it necessary to draw a block diagram first. In **Figure 1** you can see that the circuit comprises one large and one small IC, plus a handful of small bits. The author used 5-V LED strips with individually controllable LEDs; the power supply voltage for the controller may

be between 3.3 V and 5 V. Because the battery voltage drops a lot during use, a voltage regulator is indispensable. Let's start off with this.

We chose an old acquaintance: the MCP1642B, which we most-recently used in the GoNotify project [1]. Because of its very large input voltage range (650 mV to 5.5 V) it is ideal for our application.

This IC is available in an adjustable version and in versions with fixed output voltages. However, at the time of writing, only the adjustable version was easily available, so that is why we picked that one.

As is customary with this type of IC, the output voltage is set with the aid of a voltage divider that feeds back part of the output voltage. In the data sheet for the IC [2] it describes exactly how this divider (in **Figure 1** consisting of R7 and R8) should be dimensioned. It holds that:

$$R7 = R8 \cdot \left(\frac{5V}{1.21V} - 1 \right)$$

At Elektor we like 'round' resistance values, preferably from the E12 series. When we choose the value of 680 kΩ for R7, the value of R8 turned out the be a nice E12 value also:

$$R8 = \frac{680k\Omega}{\left(\frac{5V}{1.21V} - 1 \right)} \approx 220k\Omega$$

Pushbutton S1 is the reset button, but you will no doubt have already realised that. Pushbuttons S2 and S3 have been added for future use: you can use these in your own sketches to start certain actions. In the Arduino IDE these buttons can be found on pin 0 (S2) and pin 1 (S3).

PROJECT-INFO

- Portable
- Arduino
- Battery powered

entry level
intermediate level
expert level

4 hours approx.

SMD soldering tools
PC with Arduino IDE

€25 / £22 / $30 approx.

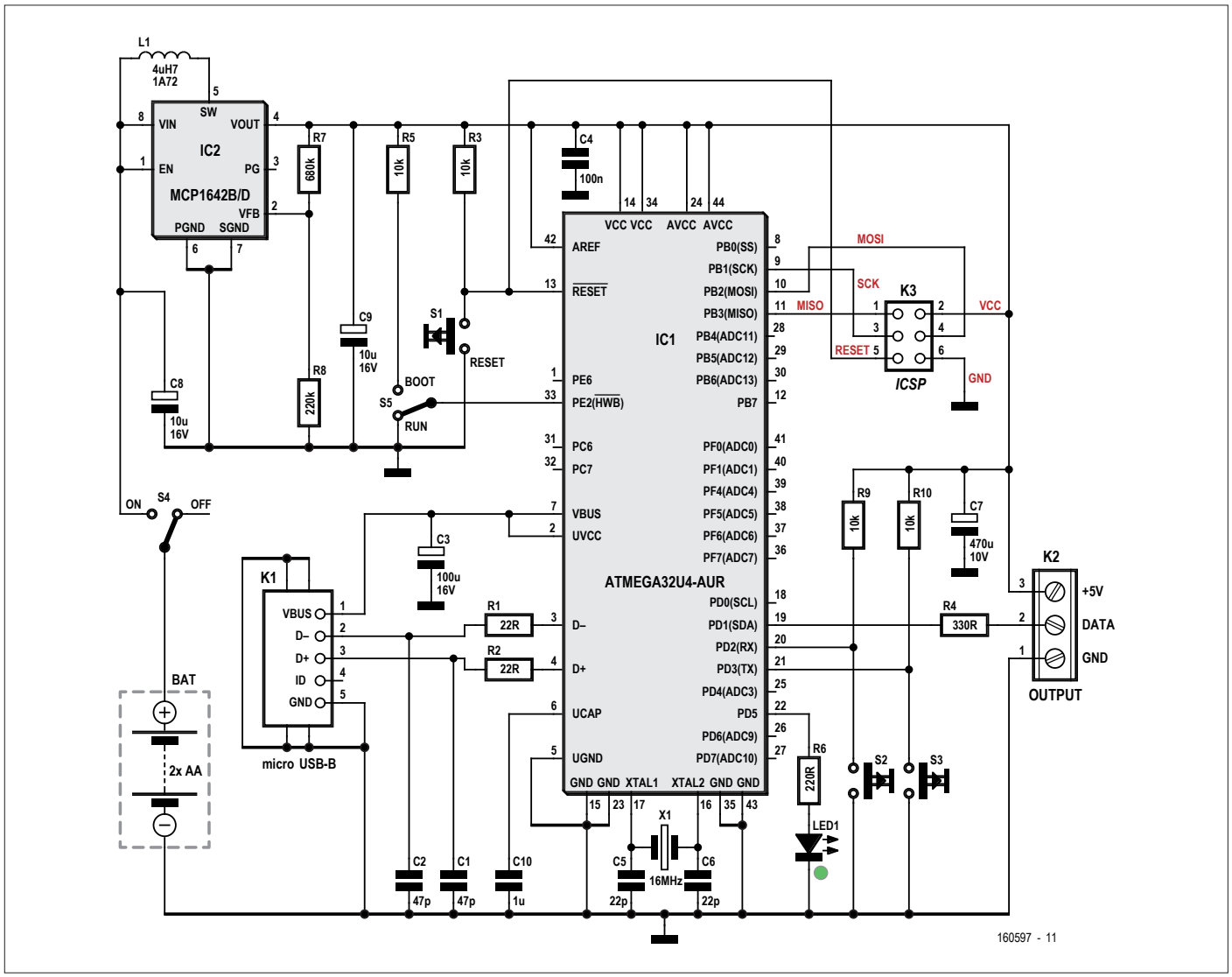


Figure 1. The schematic comprises some small fry around an Atmega controller.

S4 is the on/off switch; take note: this has to be switched ‘on’ even during programming.
LED1, a small green SMD LED, functions as the on/off indicator (it is then lit up continuously). When loading a program this LED flashes.

Connections

Connector K2 is the output of the controller. This is implemented as a three-way PCB terminal block to which the LED strips can be connected. This connector also has +5 V available, but note that this voltage can vary depending on the load. Note: check at [8] before you buy an LED strip whether it is supported!
To prevent interference on the data line (for example by reflections), resistor R4 with a value of 330 Ω is connected in series; this resistor also functions as protection for the first LED that is connected to K2. Another protection for the connected LEDs is offered by capacitor C7, which suppresses voltage fluctuations. In the Arduino IDE the data line can be defined via (digital) pin 2.
The Atmega32U4 in the LED controller is supplied pre-programmed with the Arduino Leonardo bootloader (plus a simple test program). The controller can be programmed via micro-USB connector K1 using the Arduino IDE.

If you, for whatever reason, would like to program a ‘virgin’ controller from the ground up yourself, then that can be done via ICP connector K4 (the abbreviation stands for *In-Circuit Programming*). With (for example) and Atmel AVRISP mkII this is easily done. Note: the Atmega first has to be placed in the boot mode. For that you need to put switch S5 in the position ‘BOOT’ and briefly press reset (S1).

Construction

A (double-sided) circuit board has been designed for the circuit, which is shown in **Figure 2**. Most of the part are of the SMD variety, but there is no need to despair: the circuit board is available with IC1 and IC2 already fitted (IC1 is then already pre-programmed). For the confirmed DIY-ers there is also a ‘separate’ pre-programmed controller available.
If you don’t shy away from a little bit of careful, finicky work, then the board will be finished in short order. Note: the battery holder and connectors K2 and K3 are fitted on the bottom side of the circuit board.

COMPONENT LIST

Resistors (all SMD0805, 5%, 0.1 W)

R1,R2 = 22 Ω
R3,R5,R9,R10 = 10 kΩ
R4 = 330 Ω
R6 = 220 Ω
R7 = 680 kΩ
R8 = 220 kΩ

Capacitors

C1,C2 = 47 pF, 50 V,C0G/NP0, SMD0805
C3 = 100 µF/16 V, tantalum, SMD2312
C4 = 100 nF, 50 V, X7R, SMD0805
C5,C6 = 22 pF, 50 V, C0G/NP0, SMD0805
C7 = 470 µF/10 V, 0,16 Ω, SMD electrolytic
C8,C9 = 10 µF/16 V, tantalum, SMD1206
C10 = 1 µF, 50 V, ±10%, X5R, MC Series

Inductors

L1 = 4.7 µH, 1,72 A, 1,65 A, shielded, 0.082 Ω, WE-TPC Series

Semiconductors

IC1 = Atmega32U4-AUR, 16 MHz, 32 kB, 25 kB, 44-pin
IC2 = MCP1642B-ADJI/MS, 650mV-5.5V in, 1.8V-5,5V out, 0.8 A out
LED1 = LED low power, green, SMD, 20 mA, 2.1 V

Miscellaneous

X1 = crystal 16 MHz, 30 ppm, 18 pF, SMD 5 x 3,2 mm
S1,S2,S3 = pushbutton, not illuminated, circuit board mount
K1 = micro-USB connector type B, USB 2.0, female
K2 = 3-way circuit board terminal block, pitch 0.2”
K3 = pin header 6-way, 2 rows, pitch 0.1”
S4,S5 = slide switch SPDT, on-on, through-hole
BAT = battery holder, AA x 2, through-hole

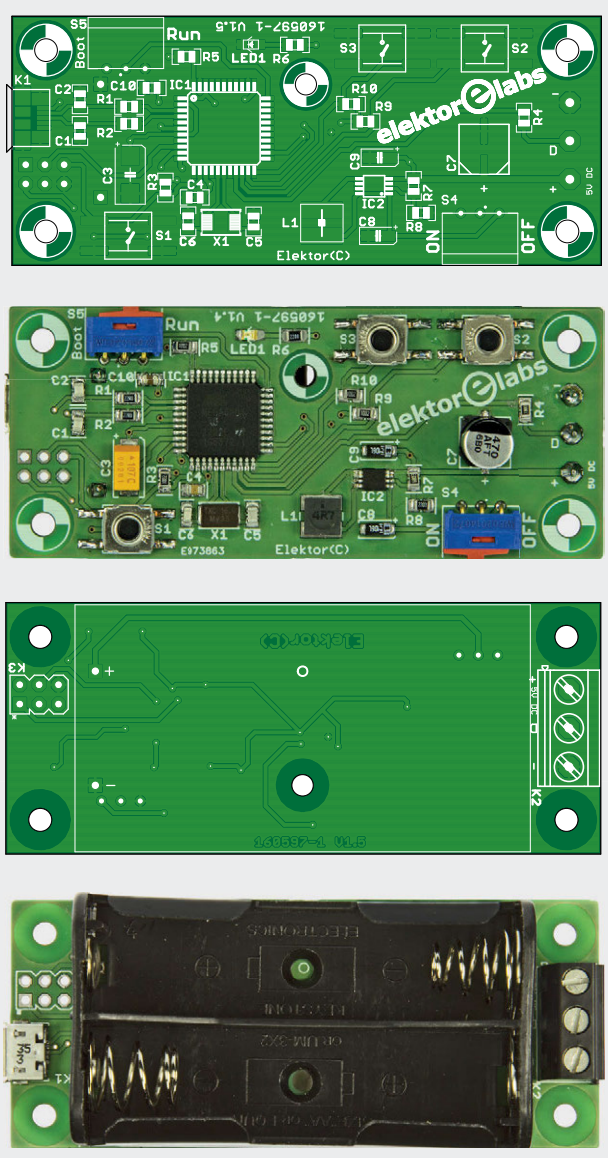
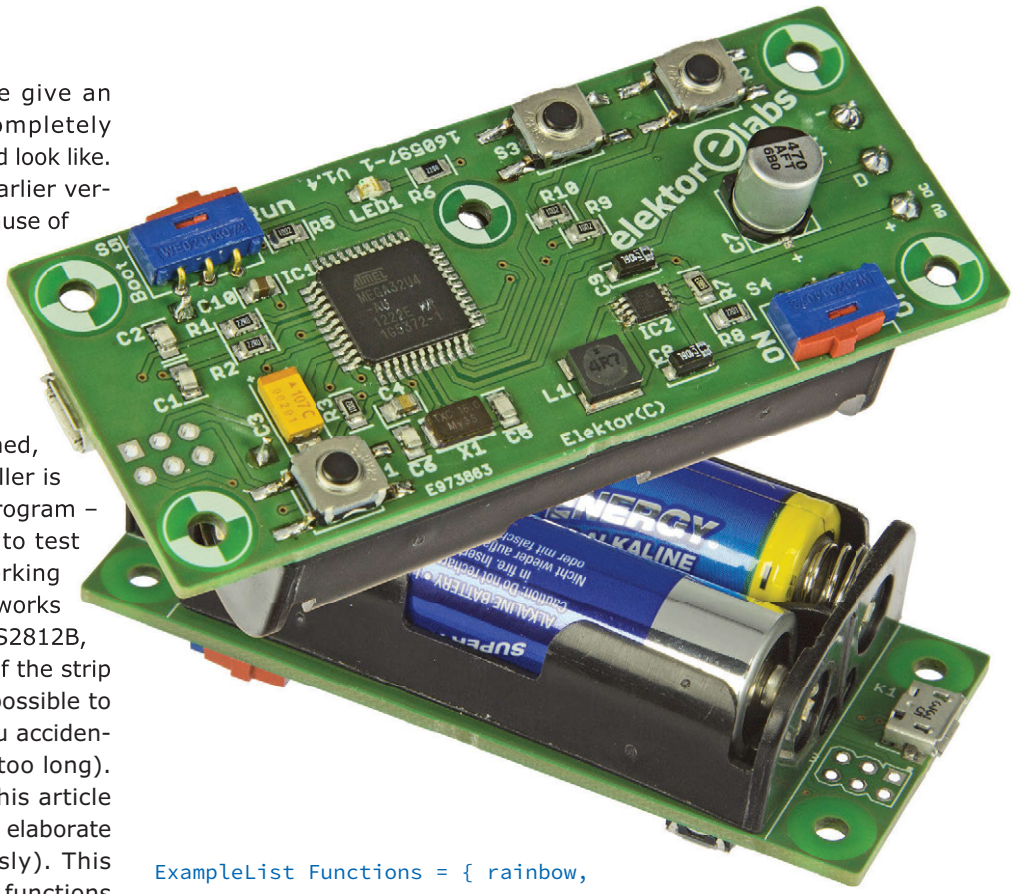


Figure 2. A compact circuit board has been designed for the circuit.

The photos with the article give an impression of what the completely assembled circuit board should look like. Note: the photos show an earlier version of the circuit board; because of technical production reasons, K1 in the final version was moved from the bottom side to the top side of the board.



Software

As has already been mentioned, the pre-programmed controller is provided with a small test program – that is really only intended to test whether the controller is working correctly; the program only works with LED strips of the type WS2812B, and only the first two LEDs of the strip light up (then it is also not possible to overload anything should you accidentally connect a strip that is too long). From the project page for this article [3] you can download a more elaborate demo sketch (free, obviously). This makes extensive use of the functions that are available in the FastLED library (see later on).

Before you can download the demo sketch in the controller you first need to select the correct board in the Arduino IDE. For this select in the Boards Manager of the IDE the Arduino Leonardo (because the controller contains the bootloader of the Leonardo).

Obviously the controller needs to be switched on, and also the FastLED library needs to be available to prevent errors during compilation.

The demo sketch contains 9 programs that you can 'cycle' through using a pushbutton – including a rainbow effect, confetti effect colour scenes and even two carnival programs (although carnival at the time of writing is already behind us).

For the reading of the pushbuttons, use is made of interrupts; these are also de-bounced in software.

You can add your own functions to the demo sketch to your heart's content; in that case do not forget to add these to the Example list: the various functions are called via this list. You also need to increase MaxFunctionNumber, otherwise the pointer jumps back to the beginning before your new functions have had a turn.

Assuming you have written a function with the name RGBChase, then you can add this to the demo sketch as follows:

```
ExampleList Functions = { rainbow,
   Green, WhiteGlitter, sinelon,
   RED, confetti, Blue, Carnaval,
   White, Carnaval2, RGBChase };
uint8_t MaxFunctionNumber = 10;
```

A new sketch

To give you an impression of the way in which a sketch for the LED controller is built, we give you a step-by-step example below. With this we make use of FastLED, which can be downloaded from the FastLED website [4]. There are also other possibilities, such as Neopixel. The choice is up to you.

The first step is including the library:

```
#include <FastLED.h>
```

Then a few other things need defining, such as the data pin (pin 2 of the controller). Also the type of LED strip needs to be indicated (in our case the WS2812B), the colour order, the number of LEDs in the strip, the brightness and the number of frames per second.

```
#define DATA_PIN    2
#define LED_TYPE    WS2812B
#define COLOR_ORDER GRB
#define NUM_LEDS    60
CRGB leds[NUM_LEDS];
#define BRIGHTNESS          100
#define FRAMES_PER_SECOND  120
```

In the setup section of the program a few functions need to be called. With this it is always a good idea to first clear any data that may potentially still be around. This can be done with FastLED.clear. Subsequently the LED parameters are called and the brightness is set.

```
void setup() {

  FastLED.clear();
  delay(1000);
  FastLED.addLeds<LED_
   TYPE, DATA_PIN, COLOR_
   ORDER>(leds, NUM_LEDS).
   setCorrection(TypicalLEDStrip);
  FastLED.
   setBrightness(BRIGHTNESS);

}
```

FastLED contains a large number of ready-made functions, so there is no need to continually re-invent the wheel. One of the functions is the Rainbow function; below you can see how this function is called in a loop. For this gHue needs to be initialised first; we do this by adding uint8_t gHue = 0 below the definitions of the LEDs.

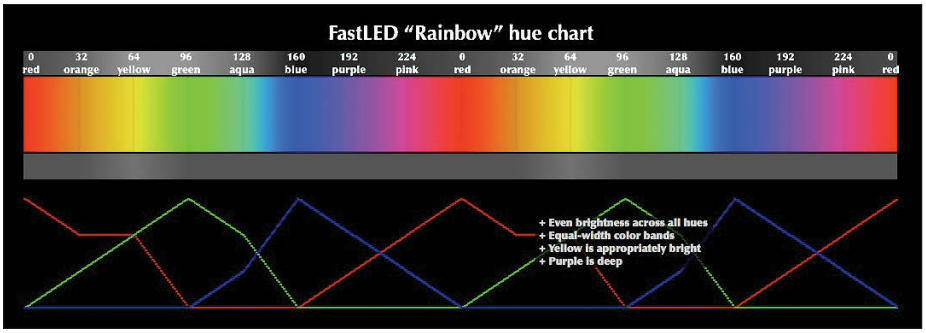


Figure 3. This 'rainbow' chart shows which hue corresponds to which colour.

```
    // FastLED's built-in rainbow
generator
    fill_rainbow( leds, NUM_LEDS,
gHue, 7);
    EVERY_N_MILLISECONDS( 20 ) {

        gHue++;  // slowly cycle
the "base colour" through the
rainbow

    }
```

FastLED supports two colour models: the traditional RDB model (Red, Green, Blue) and the HSV model (Hue, Saturation, Value). Here we show four different ways to 'fill' the pixels with yellow. With this 'i' indicates the relevant pixel.

This is the first 'traditional' way:

```
leds[i].red = 255;
leds[i].green = 255;
leds[i].blue =  0;
```

Another method is:

```
leds[i] = CRGB( 255, 255, 0);
```

It can also be done like this:

```
leds[i] = CRGB:: Yellow;
```

And when using hue values the instructions looks as follows (refer also the FastLED *hue chart* of **Figure 3**):

```
leds[i].setHue( 64);
```

To fill all pixels with a colour, it is best if you use a for construct. We demonstrate this in three different ways.
In the first example the strip is built pixel by pixel with a red colour. The second method build the strip with a blue colour; the difference is another way of calling. The third method finally changes the colour of the strip all in one go.
Note the position of the function FastLED.show(), this is crucial for the LEDs to light up at all.

```
void loop() {

    // Red build up
    for(int i = 0; i < NUM_LEDS;
i++){

        leds[i] = CRGB::Red;
        FastLED.show();
        delay(25);

    }

    delay(1000);

    // Blue build up
    for(int i = 0; i < NUM_LEDS;
i++){

        leds[i] = CRGB(0, 0, 255);
        FastLED.show();
        delay(25);

    }

    delay(1000);

    //Fill Full Strip Blue Violet
    for(int i = 0; i < NUM_LEDS;
i++){

        leds[i] =
CRGB::BlueViolet;

    }

    FastLED.show();
    delay(1000);
}
```

In the example programs from FastLED you will find countless other ways of controlling an LED strip.

We are, as always, curious as to the creative applications that you come up with for this LED controller.

(160597)

Links

[1] www.elektormagazine.nl/160333
[2] ww1.microchip.com/downloads/en/DeviceDoc/20005253A.pdf
[3] www.elektormagazine.nl/160597
[4] http://fastled.io/
[5] Information about the FastLED colours: https://github.com/FastLED/FastLED/wiki/Pixel-reference#chsv
[6] Tips related to the Neopixel library: https://learn.adafruit.com/adafruit-neopixel-uberguide/the-magic-of-neopixels
[7] Labs page for this project: www.elektormagazine.com/labs/wearable-led-controller
[8] https://github.com/FastLED/FastLED/wiki/Chipset-reference

STORE HIGHLIGHTS

Acoustics, GUI and Fingerprints

To each his own...

By **Elektor editorial team**

Juist in case you are wondering, all of the above are available from the Elektor Store...

Acoustics in Performance

In his book *Acoustics in Performance* Richard A. Honeycutt discusses the physics, scientific and psychoacoustic aspects that are essential to proper reproduction and perception of sound during performances and public events. For example, ambient noise, speech intelligibility and other aspects related to perception only.

The book succeeds in quantifying and qualifying many sensations associated with hearing in large rooms and even larger buildings, specifically concert halls and places of worship as the author Honeycutt calls them. Every chapter in the book first describes the ways in which the communication problem posed by poor acoustics is worded by the 'parties' involved: like the vocalist bothered by audience babble, the audience bothered by echoes and HVAC hum, and the hapless sound technician finding that his carefully wrought mixing console settings are useless once the hall is full of people wearing coats. Honeycutt's book addresses all these aspects and more.

Another remarkable thing about the book is the solid basis of theory it presents to address and dissect problems that at first blush appear to be an Gordian knot of physics, acoustics and electronics on the one hand, and human (= highly subjective) perception on the other. Like the sound technician starting out fairly quietly but constantly turning up each and every instrument and microphone level until a wall of sound is obtained simply to mask poorly performing artists. This is the "volume-overpowers-everything" myth which is carefully debunked in the book, first by explaining an anchoring all physics units involved, like the well-known watts and decibels but also %ALcons, a rarely seen but very useful unit to express articulation loss of consonants. Admittedly I had never heard of it.

In the future, instead of the boring "testing-one two-three", try "pa-ta-da" in the microphone and then ask a number of listeners what they actually heard. The outcome is sure to pinpoint a defect in the sound system being set up for the performance. For example, the test speaker's 'p' sound may be damped by velour curtains above the stage and loose some of its plosive characteristic sound to the extent of being heard as a 'b' by the audience. The same with the noise component so essential in

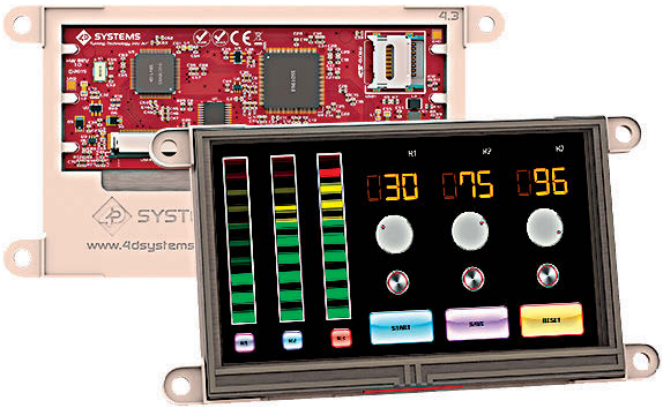
FROM THE STORE

→ **Acoustics in Performance**
Richard A. Honeycutt
Elektor International Media — ISBN 978-1-907920-64-6
124 pages, paperback — order no. 18334, price €24.95

the ‘t’ voiceless consonant — it may be filtered out by a piece of absorptive ceiling and approach the voiced consonant ‘d’.
The author is right in his steadfast approach to the problems, not by confirming subjective perceptions (in equally subjective wording) but by converging physics, facts and surprisingly simple remedies in a powerful way, supported by the purposely sober layout of the book. For example: turning down the PA volume in spite of the general excitement rising both on and off stage.
If you have an interest in optimising room, (concert) hall, studio, or even open-air acoustics aiming to give performers on stage the best tool to capture their audience, then *Acoustics in Performance* is for you. What appears to be a book serving a niche audience in fact has much wider appeal thanks to the wonderful convergence of so many disciplines including being able to formulate your perceptions in terms of acoustics.

Build a smart-looking GUI in minutes

The Australian company 4D Systems manufactures a range of easy-to-integrate intelligent display modules with and without touch capabilities. Workshop4 IDE is the tool of choice for programming the modules as it features four environments, from text-based to visual programming. Together they allow the user to have a graphical user interface (GUI) up and running in minutes rather than days.
We’re talking about a gen4-µLCD-43DCT-CLB module sporting a 65 Kcolour, 480 × 272 4.3” TFT screen with capacitive touch. It measures 123 × 84.5 × 8.4 mm, and has a 30 pin extension connector for wiring the module to other hardware.
At the heart of the display module sits a 4D Systems’ Diablo16 chip running a soft-core extensible virtual engine (EVE), which executes your 4D Graphics Language (4DGL) programs. The Diablo16 offers PWM audio generation together with GPIO, timers, PWM and quadrature encoder inputs. Serial ports are available; a microSD card provides data storage, so plenty of peripherals to play with.
Because of all these options, the display modules can either function as a slave display in a larger system or be the brains of the application.
Having a feature-packed graphic display is one thing, putting it to good use is quite another. For this reason the people at 4D Systems have come up with Workshop4 IDE, a tool that lets the user design smart GUIs in little time.
Workshop4 IDE comes in two versions: Free and Pro. Although we used the Pro version for this article, the free version is enough to follow along.
Four types of Workshop4 IDE projects exist. *Designer* — for users who prefer to type 4DGL code themselves and consequently have full control over the module; *ViSi* — a visual programming environment that allows the user to quickly set up a GUI before completing it manually with 4DGL code; *ViSi Genie* — a completely visual programming environment to design a GUI by drag and dropping GUI elements and then setting their parameters. Zero 4DGL coding required. A ViSi-Genie GUI is a slave which a host controls over a serial link; *Serial* — the module is a pure slave display, so a host must draw it by sending 4DGL primitives. For this mode to work, the SPE (Serial Platform Environment) must be loaded on the display first. This is the case for fresh out-of-the-box displays. Therefore, an ‘SPE Load’ button on the Tools menu is available to reload the SPE on a non-virgin display.
In a practical ViSi Genie GUI there will be input and output controls. Most or all output controls (LEDs, gauges, digits, etc.) will be set by the host. Similarly, most or all input controls (like buttons, sliders, knobs, etc.) will report to the host. When the GUI is ready, it can be loaded on the module.

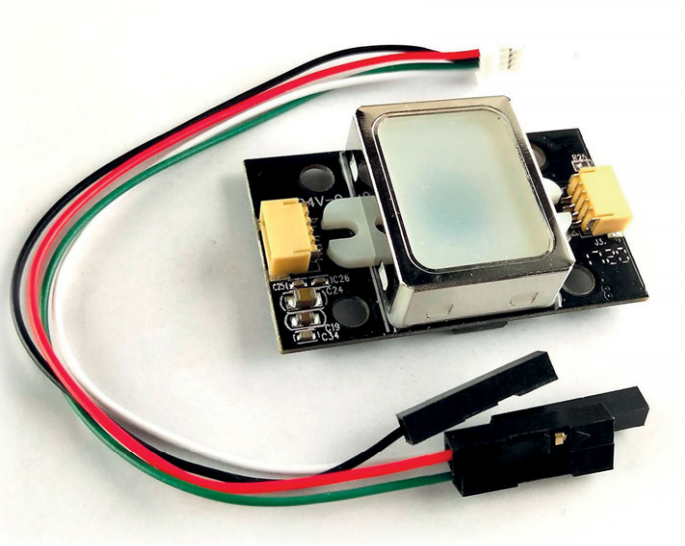


Exploring the IDE will reveal many useful tools, like for instance a microSD card partition wizard, making it a one-stop programming shop. Once you have become familiar with the module’s and Workshop4 IDE’s capabilities, you can go a step deeper. You may want to try the ViSi (without Genie) or Serial environments too. Seasoned and expert users will probably end up in the Designer environment where they have full control over the display.
Designing a nice GUI with the ViSi-Genie environment from

the Workshop4 IDE for a gen4-µLCD-43DCT-CLB module by 4D Systems really is easy. Time otherwise spent on creating and drawing touch-capable graphical elements before integrating them into the application can now be spent on other important tasks like drinking coffee. Projects will be finished faster, will look nicer and may even have more possibilities.

Fingerprint Recognition

The GT-521F52 is a small fingerprint scanner module with a serial interface. It can be easily interfaced to a microcontroller, an Arduino, a Raspberry Pi, a computer or, in short, to anything that can communicate over a standard asynchronous serial link. Adding such a module to a project would give it secure access capabilities, thus preventing pirates, children and cats from (mis)using your device.
The fingerprint scanner is manufactured by ADH Technology (there also exists a GT-521F32 which has less memory). Similar to the older GT-511 module, it is a one-chip optical sensor module with embedded recognition and storage functions. Only two wires are required to communicate with it. The tiny module measuring just 36.1 by 21 mm does all the hard work for you, all that remains to do is send it some commands and interpret the data it returned. Fingerprints are captured by and stored on the module itself. The scratch-proof optical sensor has a resolution of 450 dpi and produces 258 by 202 pixel images. The scans are converted by the module into so-called 'templates'. The GT-521F52 has room for storing a whopping 3,000 templates, the "32" can store only 200… Templates can be read from the module, one by one or the complete database in one fell swoop, but they can also be written to it, making sharing of templates between different modules easy.
The serial port accepts a 3.3 to 6 volts supply voltage. The serial port signals (default setting is 9600n81) are specified as 3.3 V signals and a level converter would be needed to connect it to for instance a 5-volt system like an Arduino Uno. Direct hooking up to a computer is possible with a 3.3-volt "FTDI" cable.
The "other" connector is for the touch interface and expects 3.3 volts maximum. The metal sensor frame functions as a touch sensor. The signal ICPCK available on this connector indicates if a finger is on the sensor or not. Touching the sensor can also be used to wake up the module from idle mode.
the manufacturer provides a precompiled Visual C++ 2005 project (including source code) that allows you to try out the module with a Windows computer. Each process starts by capturing fingerprints. This process is called 'enrolment' and consists of taking three scans of the same finger that are merged into one template (if you try three different fingers enrolment will fail). Once you have enrolled one or more users you can try to verify (ie, compare a scan to a given template) or identify (try to find a match in the database) them. You can also export them or import other templates The GT-521F52 fingerprint scanner module with scratch-proof sensor works well with dry, moist and even rough fingerprints. Although the documentation is insufficient, embedding the module in a project is not too complicated; a full-featured demonstration program including source code



is available for reference. Thanks to its high-level fingerprint processing capabilities and serial port communication the GT-521F52 is a very easy way to add a high-security access control or identification feature to a user project or product. ⏮

(160643)

Web Links

[1] www.elektormagazine.com/news/book-review-acoustics-in-performance

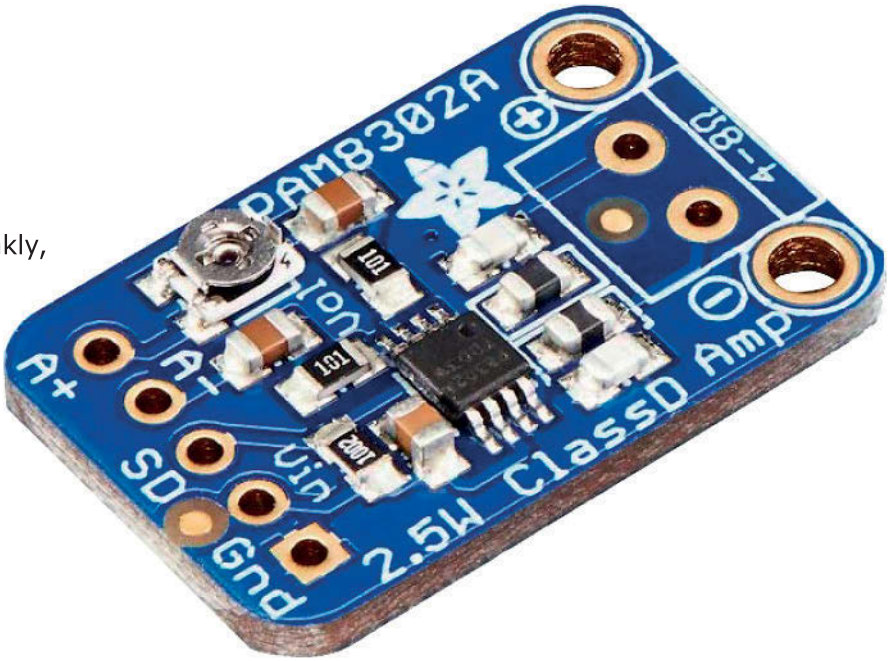
[2] www.elektormagazine.com/news/review-workshop4-ide/16883

[3] www.elektormagazine.com/news/gt-521f52-fingerprint-scanner-with-a-serial-interface-to-a-microcontroller-an-arduino-a-raspberry-pi-a-computer-or-to-anything-else-that-can-communicat/16494

PAM8302A Audio Amplifier

A modern replacement for the LM386

by **Tam Hanna** (Germany)



The LM386 universal audio amp is, frankly, well past its sell-by date now. Its feeble performance no longer meets current expectations. With the PAM8302 the author offers an up to date alternative and explains how this amplifier chip should be connected and handled.

Not every man has the desire or calling to be an audiophile, nor does everybody need an audio amplifier that can reproduce Puccini or Pavarotti in the best achievable sound quality. Yet even a process control computer (microcontroller or single-board computer) will sometimes require an audio amplifier, as for instance the author did for delivering status or warning notifications. In situations of this kind harmonic distortion is of minor significance and the key considerations are instead space requirement, simplicity, audibility, energy consumption and heat dissipation. Everything else is irrelevant.

Up to the end of the last century the go-to solution to this problem was always the LM386 integrated audio amplifier, which, because of its low efficiency, has gone rather out of fashion now. But there are several alternatives that meet the above requirements perfectly. The company Power Analog Microelectronics, now acquired by Diodes Incorporated, has developed the PAM8302A, an amplifier IC that requires only a handful of external components for practical use.

On and off

The primary method of raising the efficiency of an amplifier is by controlling its power stage. If this is not driven hard (fully) on or off, it consumes energy unnecessarily. Optimal efficiency is achieved only by a switching amplifier (Class D amplifier). The PAM8302A integrated amplifier employs digital techniques in conjunction a damping element to produce analogue wave forms. The PAM8302A is differentiated from similar ICs by interposing no filter elements between the output and the loudspeaker.

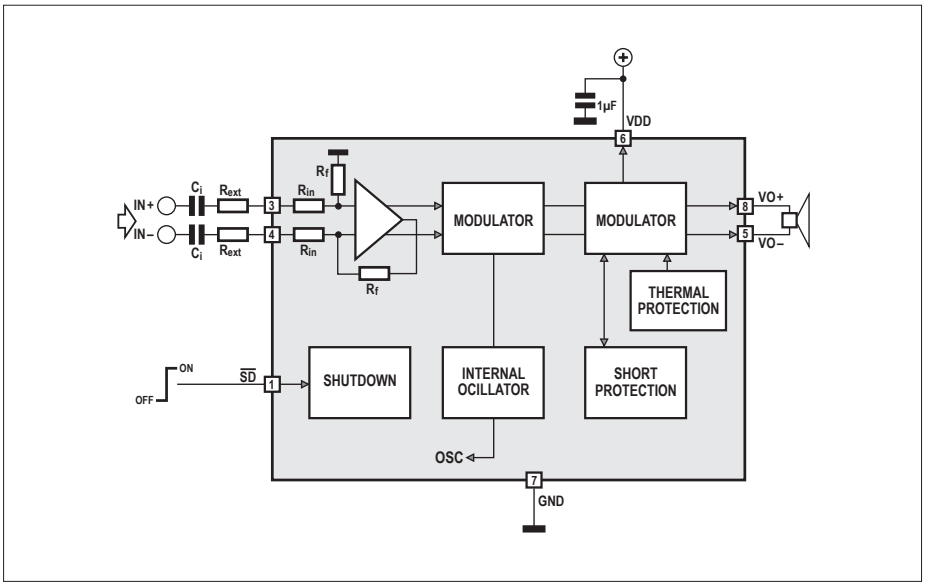


Figure 1. The datasheet application diagram for the PAM8302 is nothing less than spartan

The outline diagram of the application circuit in **Figure 1** is (absolutely) all that you need to know. The small number of external components and lack of heatsink (thanks to the outstanding efficiency) ensure the amplifier circuitry occupies a minimal footprint.
The PAM8302A has a symmetrical CMOS input, enabling you to connect a differential signals. If a symmetrical audio signal is not to be had, you will need to take pin 4 (IN-) to ground and connect the signal to the IN+ input (pin 3). The output driver, which is protected both thermally and against short circuits, is configured as a bridge (floating load) amplifier, the two outputs of which can be connected directly to the loudspeaker without further ado. According to the datasheet, using a supply voltage of +5 V the amplifier produced an output power of 2.5 W to a 4-Ω loudspeaker with a still tolerable distortion of 10%.
The active-low shutdown connection (pin 1) enables the amplifier to be switched on or off independently of the power supply. The datasheet [1] recommends that you power the amplifier on and off in this shutdown state, in order to avoid hearing plopping sounds. In the author's application this comfort feature was dispensed with, as despite this economy measure the IC functioned without any problems.
The only critical factor in this connection is that the PAM8302 offers as supplied no facility for adjusting or presetting the output signal. With the author's process control computer this is handled in the software, whereas for non-computer use you will need to add a potentiometer.

Higher mathematics

This topic of volume control brings us to the question of the total system amplification. There is no relevant information on this in the current datasheet issued by Diodes Inc. However, consulting previous versions of the datasheet [2], we read of two internal resistors, which are in fact visible in the internal block diagram (Figure 1). The feedback resistance R_f is given as 100 kΩ, the input resistance R_{in}, which can be increased by connecting an external resistor R_{ext} in series, being 10 kΩ. The maximum voltage gain a_{max} can be calculated using the formula:

$$a_{max} = 100{,}000 / (10{,}000 + R_{ext})$$
$$= V_{DD} / V_{in(p\text{-}p)}$$

Using one external 180-Ω resistor per channel the result is an amplification level close to the maximum of 10, as can be seen in the oscillogram of **Figure 2**.
The second fine adjustment affecting the use of the PAM8302 is the lower cutoff frequency of the input filter. This is relevant to the extent that small speakers have little capability of reproducing low-frequency signals. If you torment the loudspeaker with signals below 150 Hz, then, according to the original version of the datasheet, you will strain not just the speaker membrane but also the amplifier.
For filtering the input the PAM datasheet recommends placing a capacitance C_{in} and resistance (R_{in} + 10 kΩ) connected in series. This high pass filter has the following lower cutoff frequency:

$$f_{-3dB} = 1 / [2\pi \times C_{in} \times (R_{in} + 10{,}000)] \quad [Hz]$$

Our dimensioning results in an extremely low cutoff frequency, which nevertheless has not caused any problems in practice.

A question of energy

The author has made some measurements on the application circuit. Admittedly, the test adapters used do not have the same impedance as a conventional speaker, meaning that the results can only indicate trends or tendencies.
On the input side, the HP6624A lab power supply measured 45 mA of current drawn off-load, which increased to 210 mA under full load. The measurement is not totally accurate, because the cables drop a few volts and the LM2576 switching regulator, as part of the circuitry, also contributes to the current measured. At a supply voltage of 16 V the resulting power consumption is 0.72 W (off-load) or 3.36 W (under load).
A LeCroy 9354AM digital storage 'scope was used for measuring the output power. Using the formula $P = V^2 / R$, a value of 3.583 V results in an RMS power level of 1.6 watts leading to a total efficiency figure of 60%. Considering that the switching regulator does not work very efficiently, it comes close to the 88% value given in the datasheet.

From a thermal point of view there is nothing to criticise and whilst the precision resistors became uncomfortably warm, the PAM remained at room temperature. An amplifier based on the LM386 would have got into difficulties here; moreover it would require a considerably higher supply voltage with an 8-Ω load. The LM386 datasheet specifies a power of only 325 mW for a supply voltage of 6 V and a load of 8 Ω, whilst Diodes Inc. promises 'typically 1.5 W' for these conditions!

Don't upset the neighbours!

Magic bullets exist only in fantasy-land. In the real world of electronics every solution brings with it a new problem as well. With the PAM8302 this is electromagnetic interference (EMI). Rapid switching not only brings little joy to AC line-powered devices but it also turns cables into antennas. You need to take some serious measures to minimise EMI radiation, otherwise you'll be dealing your neighbours some bad cards.

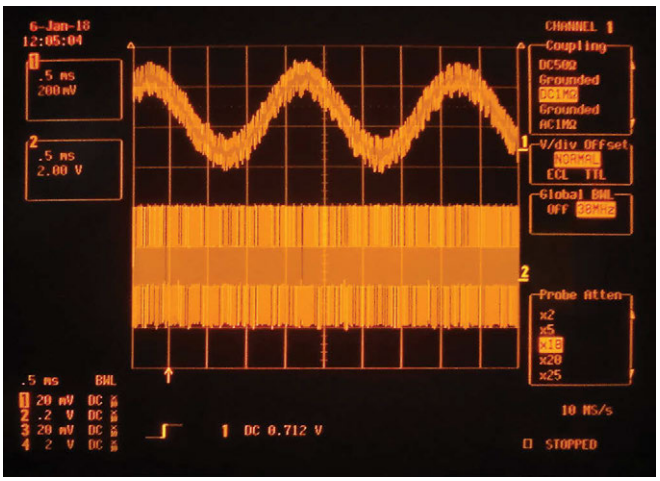


Figure 2. The oscillogram shows attenuation by a factor of 10.

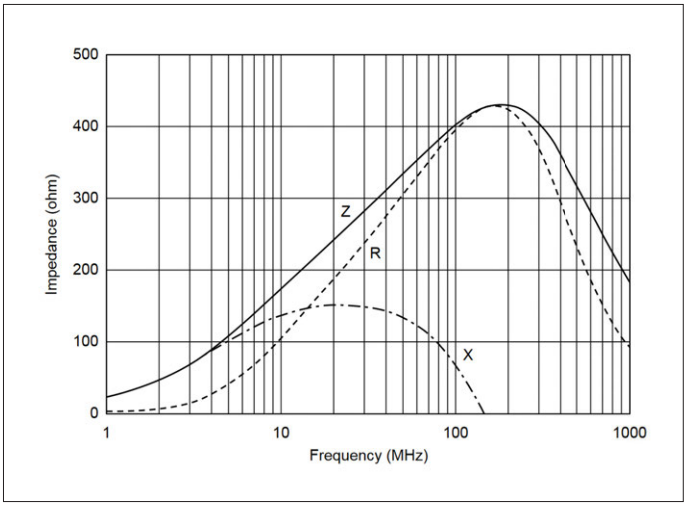


Figure 3. Frequency response of the inductance used.

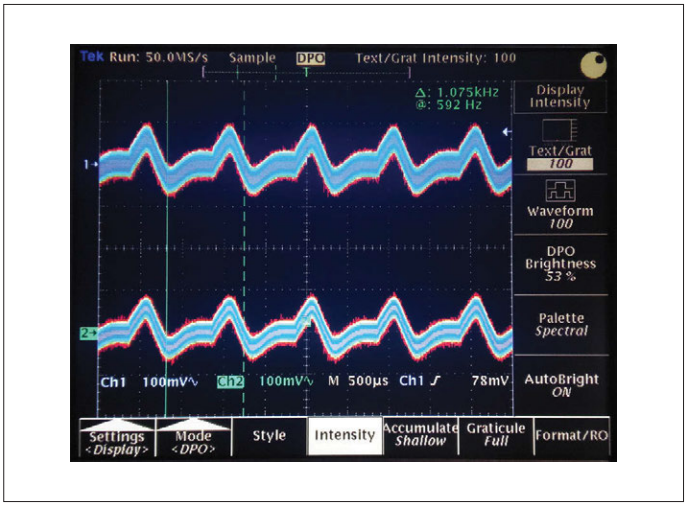


Figure 4. The second channel is 'narrower', proving that the L-C filter works!

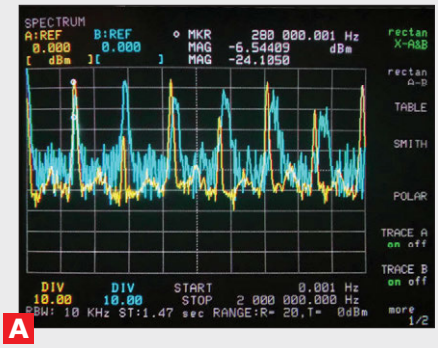
The first step in fighting EMC problems involves L-C filters on the output lines. The datasheet gives a guideline with L = 33 µH and C = 470 nF. The cutoff frequency is calculated as

$$f_{-3dB} = 1 / [2 \sqrt{(LC)}] \qquad [Hz]$$

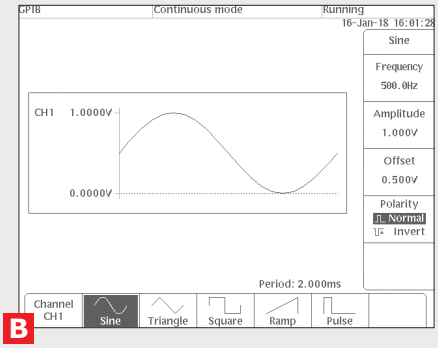
In practice this does not matter too much; you should attenuate in the range of 'a few megahertz'.
To minimize any repercussion from the PAM amplifier onto the power supply of the remaining electronics, such as sensors or digital circuits (which tend to be prone to interference in the range of several hundreds of kHz), we require an L-C input filter in the dimensions shown. The Murata inductance [3] used has a DC resistance of 0.85 Ω, which is low relative to the speaker impedance. In the relevant EMC range the impedance increases to over 400 Ω (**Figure 3**). In the oscillogram of **Figure 4** it can be seen that channel 2 equipped with a filter (below) is a lot 'slimmer' than the channel 1 seen above. The filter works!
The author has implemented the circuit in actual existing process control computers, which, to his general amazement, often turned into smoke. After some research, he learned that this was caused by turning off and on the speaker frequently during operation. This resulted in significant spikes in the output filter, which the author overcame with an additional inductor and a freewheeling diode in the supply voltage line plus a small 'bleeder' resistor on the output, by which the L-C networks are able to discharge. With this measure implemented, the amplifier has operated in hundreds of

Misoperation

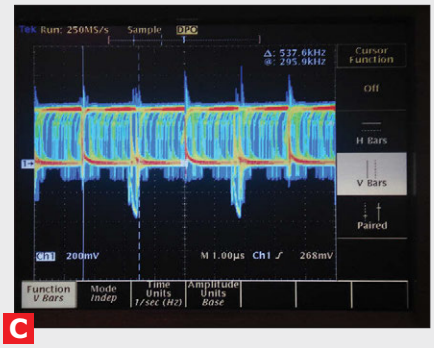
The author arrived at the best way of testing the operation of the PAM8302 purely by chance. **Figure A** shows a sweep in the range from 10 Hz to 2 MHz instead of from 0 to 2 kHz. Thanks to faulty operation, you can see a lot more clearly. The 4195A made live measurements (shown in yellow) of an amplifier carrying the signal seen in **Figure B**. The memory display (blue) contains another trace describing an idle (no signal applied) amplifier. It is noticeable that the output signals and their harmonics are subject to modulation. A digital phosphor oscilloscope is animated by this signal to output multicolour diagrams that have little informational content – Class D amplifiers are a problem that should be approached in the frequency domain. In any case, **Figure C** shows that we are dealing with classical pulse width modulation with a frequency of around 300 kHz.



A



B



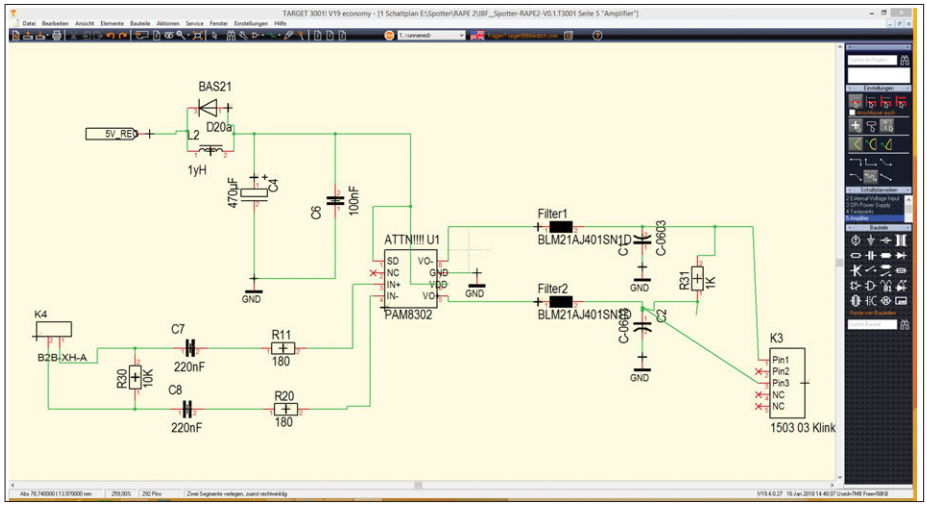
C

process control computers without further failures.

Next of kin

The PAM8302 is absolutely ideal for reproducing mono signals. In practice we frequently deal with stereo signals as well. The desire might also arise to adjust the volume digitally. Diodes Inc. also uses the technology it acquired from Power Analog Microelectronics in a multitude of other components listed at [4]. Technically the PAM8003 is closely related to the PAM8302A: the chief difference is that the PAM8003 is stereo but no longer a differential amplifier, with the inputs referenced to ground. The volume control is set by an analogue voltage applied to the volume pin, which is sampled in 64 steps. The analogue voltage can be generated using a D-to-A converter or a digital potentiometer. An equally interesting alternative is the PAM8407. With this you can set the audio level in 32 steps using two pressbuttons connected to the Up and Down pins. The rate of volume change increases rapidly when you hold down the button — luxury!

(160446)



Web Links

[1] www.diodes.com/assets/Datasheets/PAM8302A.pdf

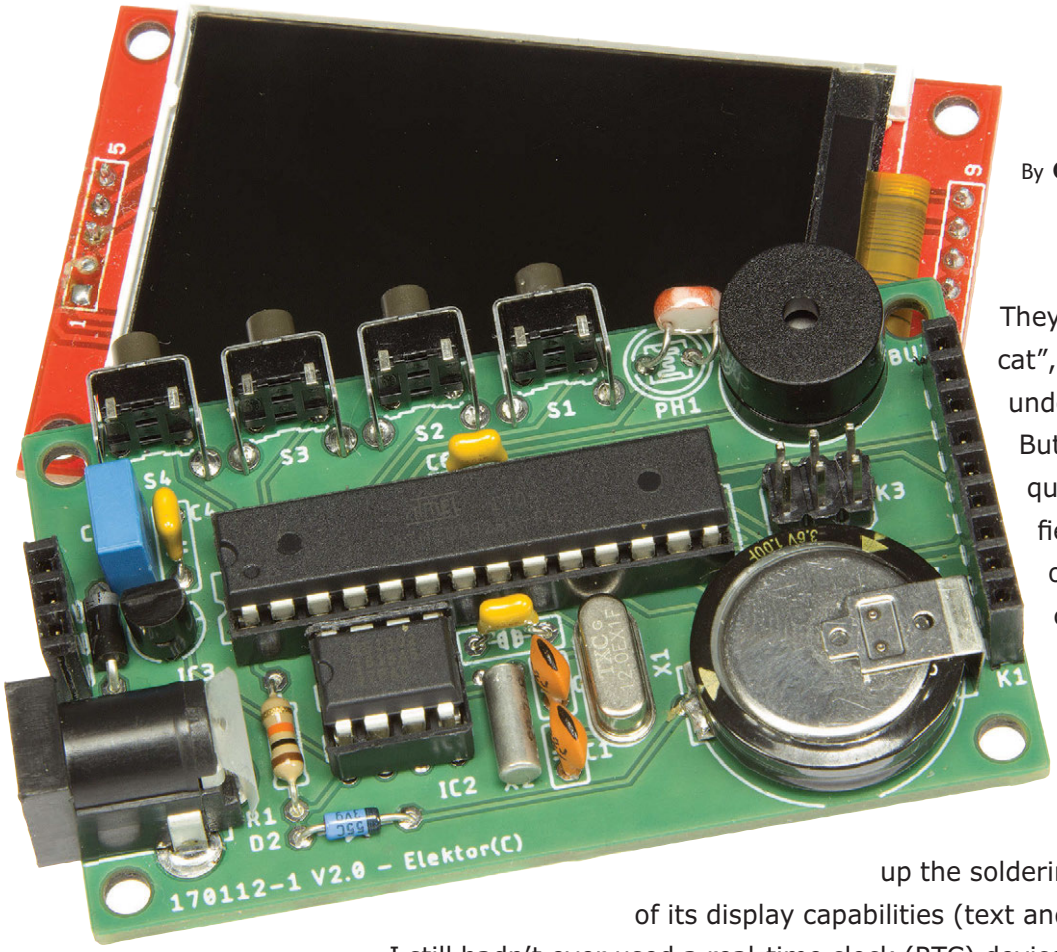
[2] https://hackstore.co.il/wp-content/uploads/2016/12/PAM8302A.pdf

[3] www.mouser.com/catalog/specsheets/murata_BLM21AJ601SN1D.pdf

[4] www.diodes.com/products/analog/audio/

3-Way Display Alarm

with 2.2-inch TFT screen



By **Olivier Croiset** (France)

They say “Curiosity killed the cat”, as if it were some kind of undesirable character defect. But I see it more as a positive quality, essential in technical fields. Digging around, going off in search of a specific characteristic… Quite by accident, on the web I came across a little 2.2-inch diagonal TFT screen (and all its larger fellows). I felt an irresistible urge to heat up the soldering iron and take advantage of its display capabilities (text and graphics). What’s more, I still hadn’t ever used a real-time clock (RTC) device. The combination of the screen and the clock led me to this alarm-clock.

Specifications

- 3 different screens for displaying current date and time
- Automatic display change
- Alarm function with ‘snooze’ button
- 2.2-inch TFT screen
- Real-time clock device
- Programmable using Arduino IDE
- Automatic screen brightness adjustment
- Choice of display language

As the circuit shows (**Figure 1**), the alarm is run by an ATmega328P-PU microcontroller ticking at 12 MHz. The software is written in the Arduino IDE. The microcontroller can be programmed on an Arduino Uno board or via the pins of the SPI connector on the alarm board itself. This lets you modify the program, in particular the screen colours.

The clock comprises:

- a 2.2” TFT screen (240 × 320 pixels);
- a DS1302 real-time clock IC using a 32.768 kHz crystal;
- three buttons (−, +, OK) for setting the current time and date and the alarm time;
- a button to enable or disable the alarm (e.g. at weekends);
- the alarm buzzer;
- a photocell (LDR) to automatically reduce the screen brightness at night (for light-sensitive sleepers like myself);
- an LM1117 regulator to power it all.

The 2.2" screen has a resolution of 320 × 240 pixels, with three colours (R, G, B) on 8 bits. It is controlled by the SPI bus MOSI/MISO lines. You'll note that it is possible to insert an SD card at the back of the display, but we're not going to be using this function here. Some larger screens are touch-sensitive, but an 8-bit microcontroller is generally not powerful enough to handle this (shortage of FlashROM).

The first problem to solve is controlling this TFT screen. I found it on eBay, where it is being sold by a Chinese site. At the low price (around €5.50 / £5.00 / $6.75), it's hardly surprising no documentation is provided. Its part number includes the mention ILI9346, which is a good starting point. A quick search on Github returned the `Ucglib` library developed by Oliver Kraus and very well documented.

Going through post-engineering steps in the Elektor Labs prompted a few changes to the original circuit. The project's initial power supply included two voltage regulators: 3.3 V and 5 V. As all the components can run at 3.3 V, only the 3.3-V regulator has been kept. As a result, the level change (down from 5 V to 3.3 V) between the Atmel328P-PU and the screen became redundant. This lower supply voltage meant that the clock frequency had to be reduced from 16 MHz to 12 MHz. And lastly, to improve the protection of the circuit, the circuit's positive power pin now has a diode to block any reverse voltage.

An external real-time clock (RTC DS1302) with a supercapacitor provides backup for three days in the event of a line power failure. The supercapacitor is charged via a Schottky diode, to isolate the external supply to the real-time clock from the rest of the circuit. A Schottky diode is needed because it offers the lowest voltage drop, which is important, as the

PROJECT INFORMATION

TFT screen
real-time clock
Arduino IDE

entry level
intermediate level
expert level

1–2 hours approx.

Soldering iron,
computer with Arduino IDE

€15 / £12.50 / $20 approx.

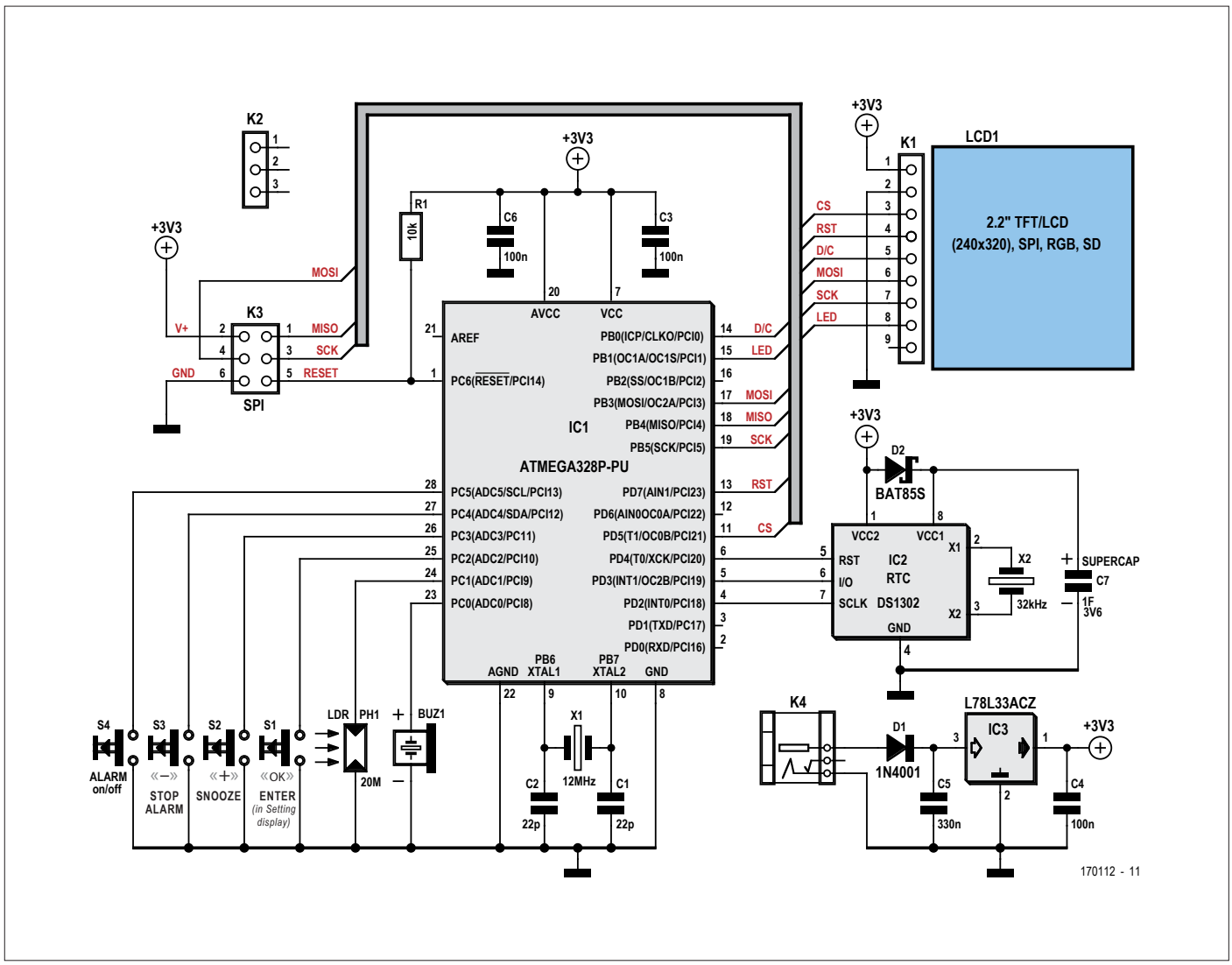


Figure 1. Circuit of 3-way Display Alarm.

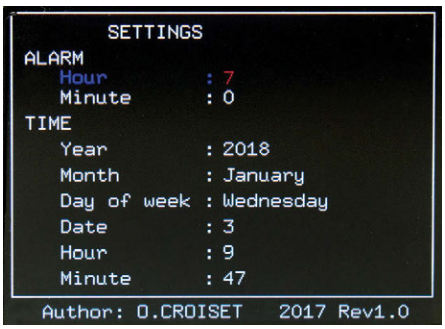


Figure 2. Alarm setting screen.

supercapacitor is being charged from a voltage of just 3.3 V.
The LDR measures the ambient light level, so the voltage powering the screen backlight LEDs can be reduced at night. People who need complete darkness to be able to get off to sleep will be glad of this; but I have made sure all the same that the display can still be seen in the dark.

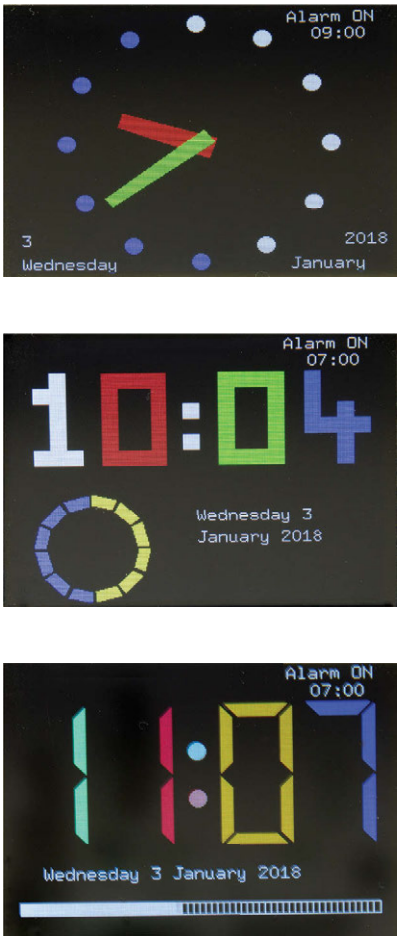


Figure 3. Various displays possible: hands, dots, or 7-segment.

Software

We've taken a look at the hardware, now let's look at the software. The program is based around a primary loop that performs the following tasks:

- Displaying the alarm time. Depending on the position of the alarm on/off button, this display is suppressed or not, and the buzzer will be enabled if necessary.
- Displaying the time (hh:mm:ss) and date (dd:mm:yy) according to the display mode set (7 segment, dots, or the very classic clock-face).
- Reading the −, +, and OK buttons.
- Reading the alarm time so as to activate the buzzer or not.
- Reading the LDR so as to adapt the screen brightness to the ambient light level.

The initialisation section of the code calls the various libraries needed, including `Ucglib.h` [1] used for displaying text and graphics elements in colour on TFT or OLED screens. This library includes numerous typefaces. There's a great temptation to include several of them to make the display more attractive. However, this library takes up nearly 16 KB, and each typeface 1–5 KB, which leaves only a little bit of room for the application. So you really have to choose the most suitable typeface. This is one of the limitations of this type of display controlled by an 8-bit microprocessor. Note that the complete program uses nearly 30 KB (the maximum if you want to keep the bootloader). The `Streaming.h` library [2] from Mikal Hart simplifies the display of character strings and successive numeric values. The "variable definition" part is fairly long. Graphic elements actually use up a lot, e.g. eight variables for a simple rectangle (x and y coordinates for all four corners). The round dial with hands may seem simplistic, but this is deliberate, as this type of display uses up a lot of resources. The trigonometric functions use "float" variables coded on four bytes. Drawing a hand (i.e. a quadrilateral) requires processing no less than 32 bytes, namely (4 x-co-ordinates + 4 y-coordinates) × 4 bytes.
The names of the days and months are available in French, English, German, and Italian. To display day and month in your chosen language, you just have to remove the "//" at the start of the lines for the language you want, and make the language block you don't want into a comment, and then recompile the program.
`Timer1` is used for reading the three setting buttons and to avoid contact bounce, as well as for producing the alarm using the buzzer. The alarm time is stored in the EEPROM: it is retained in the event of a power failure. In that event, the RTC will keep running, powered by the super-capacitor.
When the time and date display mode is to be changed, the whole screen has to be erased (78,600 pixels!), which keeps the ATmega fully occupied. In the first version of the code, the instruction `SWSPI` (software SPI) was used to reset the screen, but the microcontroller got behind and the seconds display exhibited some errors. This problem has now been solved using the `HWSPI` (hardware SPI) instruction.

Construction

The 2.2" TFT screen is hard to find in Europe, but the eBay.com website ought to enable you to find a suitable display. The Chinese suppliers are pretty professional, but delivery times can be a bit long… so just be patient while you're waiting for this key component in the project.
Assembly doesn't present any special difficulty. As this is for a household project, go for sturdy construction. Elektor offers a through-hole component style PCB, so no wiring problems.
Attention: pin 2 of the ISP connector (K3) is connected to +3.3 V; it must not be connected to a 5-V programmer. The device must be programmed before plugging in the screen, as connector K3 is located beneath the screen. This is not a problem as long as you are not using this project as a development platform.
You can produce the case for the alarm yourself (in your chosen colour) as the 3D printer files are offered for download. These are STL files produced using 123Design [4]. The case will transform the alarm circuit into a solid object that will not look out of place in your bedroom.

Operating instructions

This alarm draws around 300 mA, so it can be powered from a phone charger. However, all kinds of power adaptors can be used thanks to the jack connector. If you are comfortable working with SMD devices, you can replace it with a more compact mini or micro-USB connector

and position it as you see fit on the PCB. After applying power, go into 'Settings' mode by pressing the OK button. Modify the current parameter using the + and − buttons, and confirm with OK to move onto the next one. If you don't want to change the displayed value, just press OK.

At the preset time, if the alarm button allows it, the alarm will go off; the alarm will stop automatically after one minute. To stop it before this time, press the Stop button (−): the alarm will go off again in 24 hours. For a few extra minutes in bed, press the Snooze button (+): you'll hear the buzzer 5 minutes later. If you're still not ready to leave you bed, press + again (which makes a total of three alarms in 10 mins).

Note: if you decide to replace the alarm enable/disable button with a slide switch, you'll have to adapt the Arduino sketch.

Conclusion

This type of display is interesting if the display speed is not a crucial factor; it requires a great deal of resources for an 8-bit microprocessor. What's more, you mustn't be too greedy and go for a simple display using a single typeface. I've written the program as legibly as possible, providing information about the various displays. The current program changes the display every hour; it could be reprogrammed to change it only once a day. The colours can also be adapted to anyone's taste. Elektor and I will be interested in your ideas for variants on the display and colours: don't hesitate to tell us about them! Don't forget to consult the lab page for this project [5] to see the latest updates. For me, this project is a starting point for future projects, doubtless more complex; I'm certainly not short of ideas!

(160590 [170112])

COMPONENT LIST

Resistor

R1 = 10kΩ, carbon film, 5%, 0.25 W, 250V

Capacitors

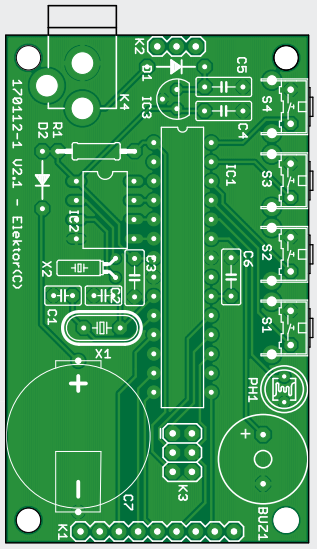
C1,C2 = 22pF, 50V, C0G/NP0, 0.1" pitch
C3,C4,C6 = 0.1µF, 50V, X7R, 0.2" pitch
C5 = 0.33µF, 50V, X7R
C7 = supercapacitor, EDLC, 1F, 3.6V (Panasonic type EECRG0V105H)

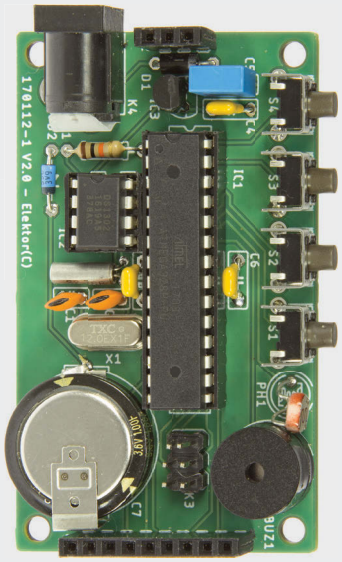
Semiconductors

IC1 = ATMEGA328P-PU microcontroller, 20MHz, DIP-28
IC2 = DS1302 real-time clock, DIP-8
IC3 = L78L33ACZ, 3.3V linear voltage regulator, TO-92-3
D1 = 1N4001-T
D2 = BAT85S-TR Schottky diode
PH = LDR, 20MΩ, 50mW, 100V (ref. A 906013)
X1 = 12MHz quartz crystal, C_L = 18pF
X2 = 32.768 kHz crystal, cylindrical, radial lead, C_L 12.5pF, 2ppm, 6.2×2mm (Raltron type R26-32.768-12.5)

Miscellaneous

BUZ1 = buzzer, 12mm
K1 = 9-way pinheader socket, 0.1" pitch
K2 = 3- way pinheader socket, 0.1" pitch
K3 = SPI board-to-board connector, 6-pin
K4 = male jack connector, centre pin 1.95mm dia., 12V, 3A (GND = centre pin!)
S1,S2,S3,S4 = tactile switch (Alps type SKHHLVA010)
28-pin DIL socket for IC1
8-pin DIL socket for IC2
2.2" TFT screen, 240×320 pixels, SPI, RGB, SD: see [2]





Web Links

[1] ILI9346 Library: https://github.com/olikraus/ucglib/wiki/reference
[2] Streaming Library: http://arduiniana.org/libraries/streaming
[3] TFT screen: www.ebay.com/itm/2-2-LCD-2-2-inch-SPI-TFT-LCD-Display-240x320-ILI9341-51-AVR-STM32-ARM-PIC/172671530762?ssPageName=STRK%3AMEBIDX%3AIT&_trksid=p2060353.m2749.l2649
[4] Article page: www.elektormagazine.com/160590
[5] Elektor Labs project page: www.elektormagazine.com/labs/3-displays-alarm-clock-with-tft-screen-1

FROM THE STORE

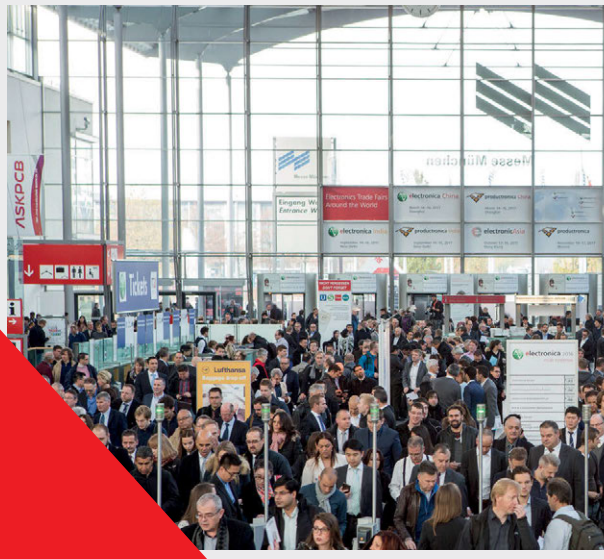
→160590-1
PCB, bare

→160590-41
programmed microcontroller

→18419
screen

→160590-71
kit avec circuit imprimé nu, écran TFT, µC programmé et tous les autres composants

electronica



powered by elektor

Fast Forward 2018

Whether you represent an established company looking to market an amazing new product, or a start-up hatching a groundbreaking innovation, getting a head start in today's highly competitive and ever-changing world is difficult. Luckily we have the solution: electronica Fast Forward Award (e-ffwd), the Start-Up platform powered by Elektor!

€75,000 should convince you to participate in the e-ffwd

No better place to present an innovative product to future business partners and clients than at electronica, the must-visit electronics trade show in the world. Join the e-ffwd and reach out to its expected 70 K+ visitors and 250 K+ Elektor readers throughout the world! (photo courtesy electronica).

The e-ffwd 2018 categories reflect electronica's tech highlights:

- automotive
- embedded
- artificial intelligence
- solid-state lighting / LED
- smart grid / smart energy
- industrial / IoT
- medical and healthcare (including wearables)
- cyber security

In 2016, with 35 participants from 16 countries, the first edition of the electronica Fast Forward Award (e-ffwd) , the Start-Up platform powered by Elektor, was a resounding success. It is therefore only logical to repeat the event in 2018 while making it even better.

Pitch your product at the e-ffwd stand and qualify to win one of the prestigious e-ffwd awards. The overall winner of the 2018 edition will receive a startup marketing package worth €75,000 from Elektor including an exhibition stand at electronica 2020. The second place will be awarded an Elektor media campaign with a value of 50,000 € while third place will benefit from media exposure via the international Elektor network worth 25,000 €.

Sponsors are welcome too

Visit **www.elektor.com/e-ffwd** and complete the sign-up form.
It is also the place to look for more information like the Terms and Conditions, and news.

We look forward to hosting the **e-ffwd 2018 edition** and welcome you at the **electronica** trade show in Munich this November.

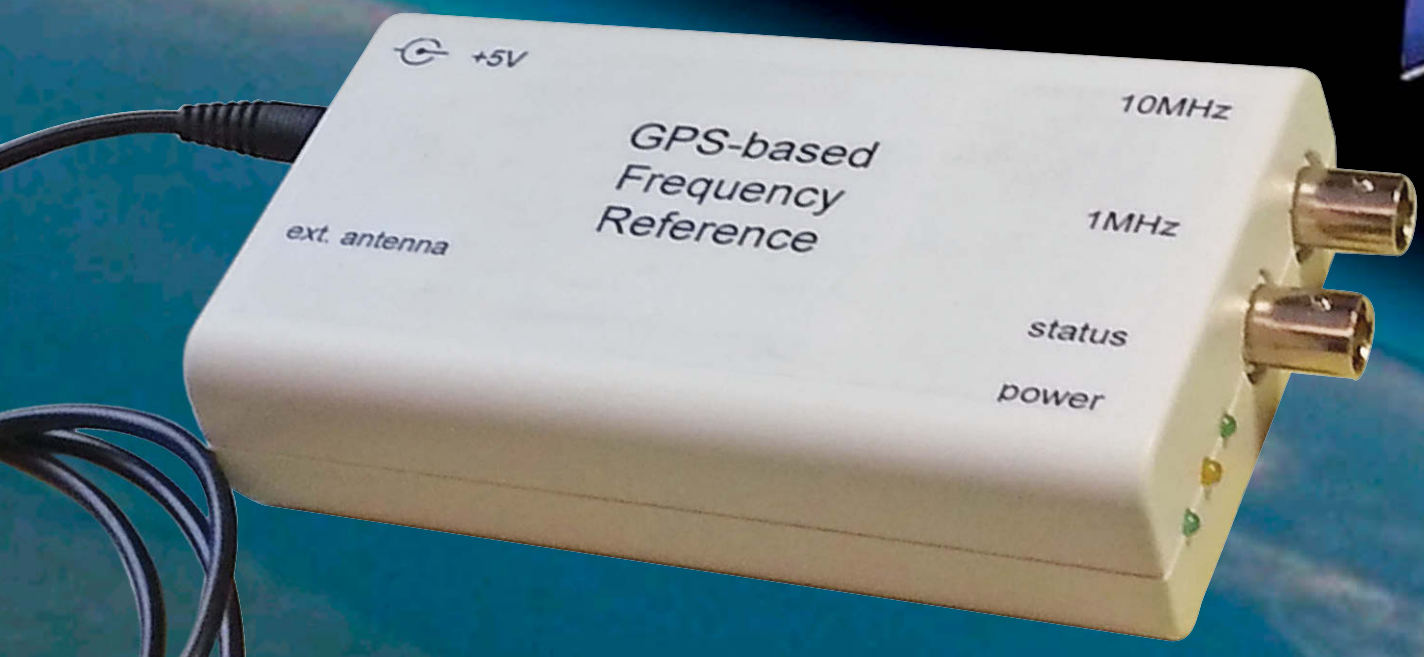
HOMELAB PROJECT

10-MHz Reference

with satellite precision

By **Willem den Hollander** (Switzerland)

Every now and then it would be nice if you had a stable signal available, the frequency of which was known exactly. For example, to finally calibrate your frequency counter properly, or as a reference when working on radio frequency circuits, or — well, anything else for that matter, since a good reference will come in handy sooner or later.



Characteristics

- 1-MHz and 10-MHz outputs
- Locks in 5 to 10 minutes with the GPS signal after identification of 3 satellites
- Accuracy 1:10^{10} under ideal circumstances

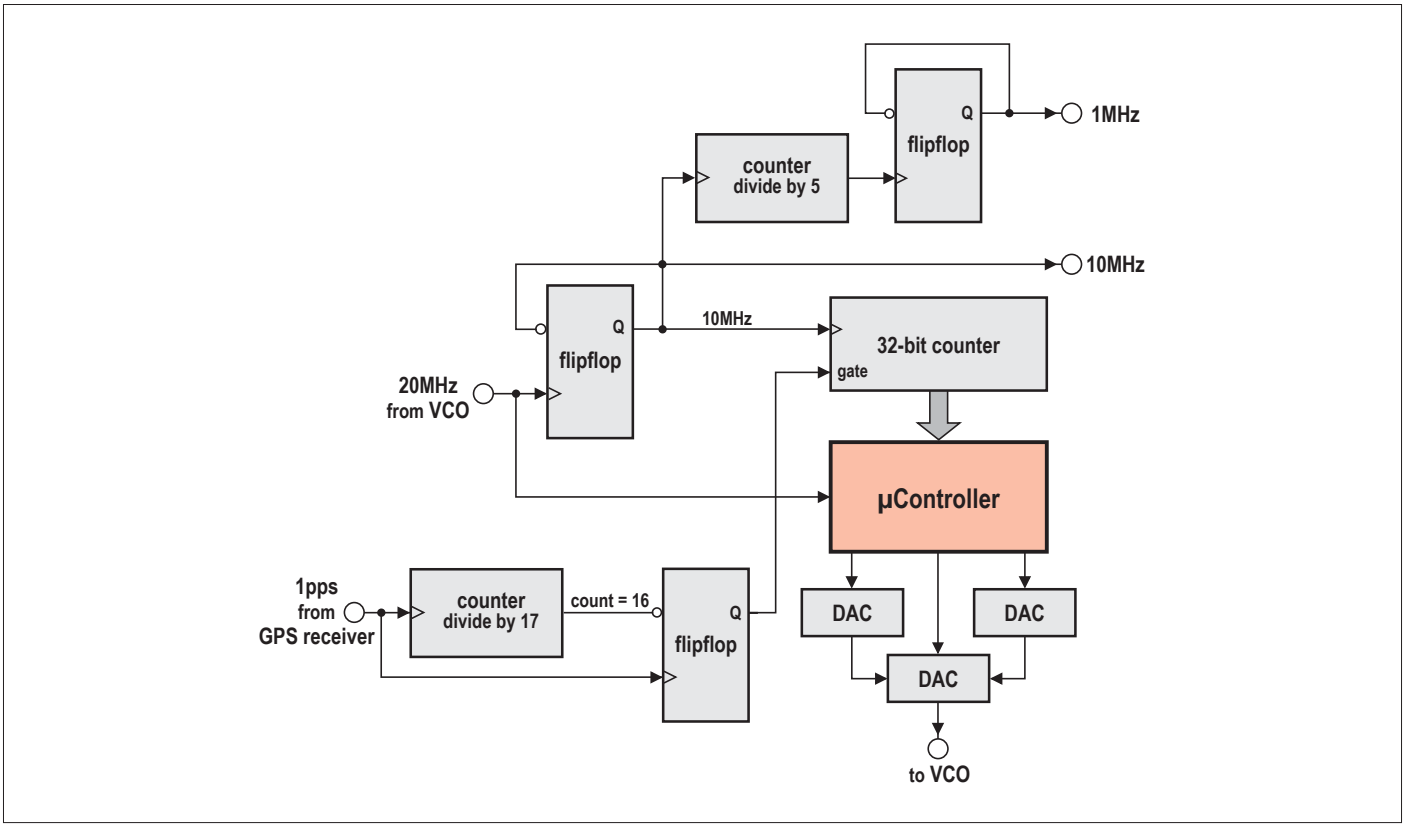


Figure 1. The block diagram of the 10-MHz reference looks complicated, but everything you see here fits in a single IC.

Not all that long ago this was quite a difficult problem. For simple hobby purposes you could just use an ordinary crystal — but no crystal oscillates at the exact frequency that is marked on its housing (although… years ago I bought, for not even 20 euros from a now long-vanished department store, a quartz watch the crystal of which happened to be ground so well that the magnitude of the error was about a minute per year) and in addition, the frequency is temperature sensitive. A temperature-compensated crystal oscillator, when it comes to stability, is already a big improvement, but it still does not solve the accuracy problem. In order to generate a really good reference signal we use a temperature-compensated, voltage-controlled oscillator, the output of which is locked to — a reference signal...

Chicken and egg

This is a little like the familiar problem of the chicken and the egg: which came first? For the large laboratories and metrological institutes, where it is not necessary to turn over every cent three times before spending it, the solution if obvious: buy a caesium atomic clock, use it to derive the standard second and voilà — the reference has been accomplished. For the hobbyist such a solution is naturally not an option — but that is not a reason to despair.

One possibility is to use the signal from the time transmitter DCF77; the frequency of which is a very accurate 77.5 kHz. The problem with this is that this (very) long wavelength signal is very sensitive to interference and cannot be

Figure 2. The hardware of the frequency reference comprises (apart from some ‘small fry’) not more than three components.

received everywhere with sufficient signal quality, at least not with acceptable methods. It is also true that with DIY or commercially available receivers, the best precision that can be achieved this way is of the order of microseconds.

Navigator

Fortunately there is a better solution these days. The Earth is surrounded by a swarm of navigation satellites (GPS, GLONASS, Galileo, Beidou) that send their signals ceaselessly, which are then used by those sat navs that no motorist nowadays would do without when venturing into unknown territory, to calculate the location on Earth. (As an aside — in the past *maps* were used for this purpose, those large sheets of printed paper that where always torn right at the place where you needed to go. The fear is that map reading is a dying skill.) Modern GPS receivers, which are available from any electronics mail-order store and also from the Elektor-shop, supply a 1-Hz signal (1 pps = 1 pulse per second) with a specified propagation delay (from input to output) of 10 ns (the actual accuracy is typically much better than this and is in the order of 1 : 10^{10}). Good enough for our application, and it is not at all difficult to lock a VCO to this signal.

PLL

The somewhat more mature electronics enthusiasts among us will probably fondly recall the CD4046 CMOS IC [1], the well-known combination of a VCO (Voltage Controlled Oscillator) and a phase comparator, and all this in a manageable 16-pin package. Using this it was possible, with relatively few external components, to build a real PLL circuit (Phase Locked Loop), where the phase difference between the VCO and an external signal results in a control voltage for the VCO, which in turn causes the VCO to remain nicely locked to the external signal.

This IC was, however, not all that easy to use. Particularly the correct sizing of the low-pass filter for the VCO control voltage required a considerable amount of 'Fingerspitzengefühl', and also the frequency range of the VCO, with an upper limit of about 1.4 MHz is very limited by today's standards. But — all those great things that is IC is capable of: FM-modulation and demodulation, frequency synthesis, voltage to frequency conversion, just to name a few of them...

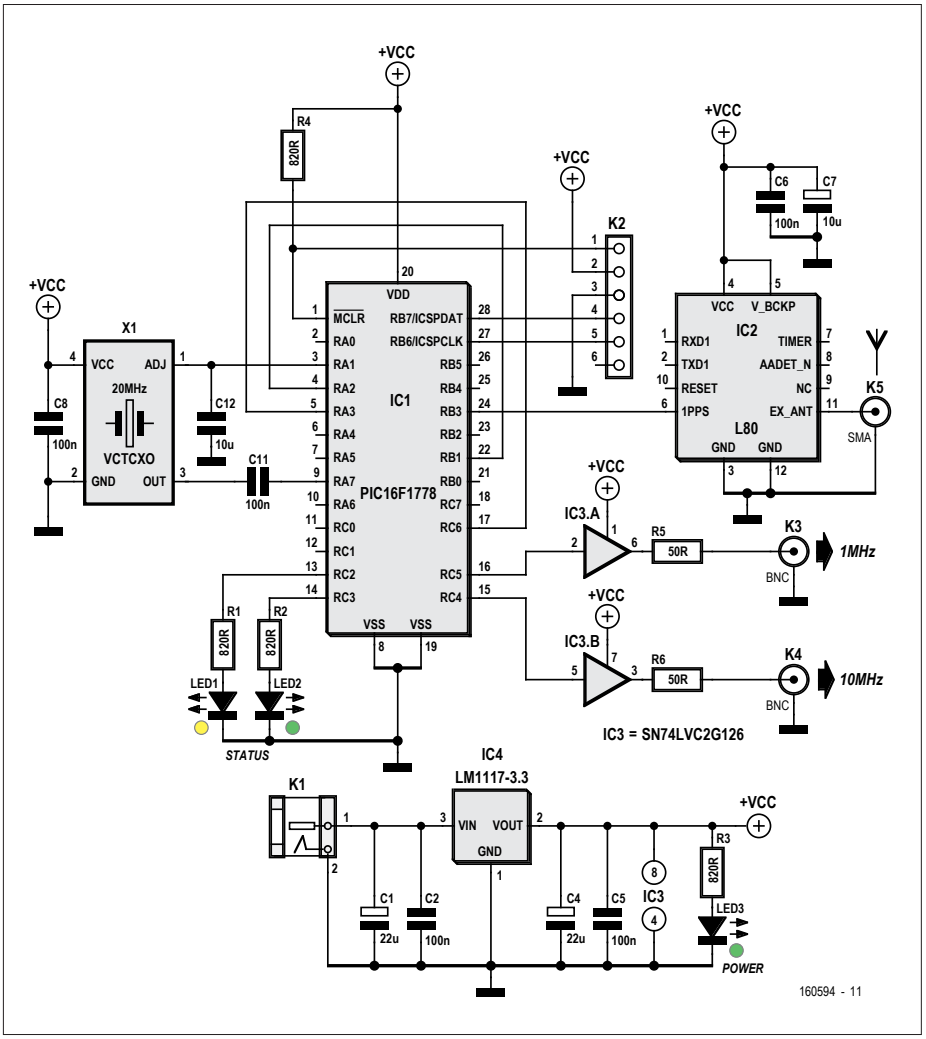


Figure 3. The complete schematic of the 10-MHz reference.

From the block diagram...

But let's not get too carried away with all the nostalgia: the author has designed a circuit for the 10-MHz reference that is much more contemporary, simpler and (to be honest) better. **Figure 1** shows the block diagram.

A VCO generates an input signal of 20 MHz. A flip-flop, configured as a divide-by-two, turns this into a 10-MHz signal — this also becomes the 10-MHz output signal from the circuit. Because we would also like to have a 1-MHz signal, we also drive the 10-MHz signal through a divide-by-five followed by a divide-by-two flip-flop; the result is a 1-MHz square wave with a duty cycle of 50%.

The GPS receiver supplies a 1-pps signal (one pulse per second); from this an accurate interval of 16 seconds is derived. During this period of 16 seconds, the 32-bit counter counts the 10-MHz pulses from the VCO. After this counting period has expired, the count value is compared with the ideal (theoretical) value — which at 10 MHz is obviously 10 x 10^6 x 16 = 160 million. From this difference a voltage is determined that is used to tune the VCO. The result of this is that after some time the VCO signal is locked with the GPS signal, and that is exactly what we wanted: the output signal is a very accurate 10 MHz that can be used as a reference.

... to the completed schematic

Fear not! If based on the block diagram you were led to believe that this would again turn into some darned complicated circuit, then you are completely wrong! In **Figure 2** we have put the three most important components next to each other — apart from some 'small fry' this is all we need: a GPS receiver, a temperature-compensated voltage-controlled crystal oscillator and a microcontroller. You can see the final schematic in **Figure 3**.

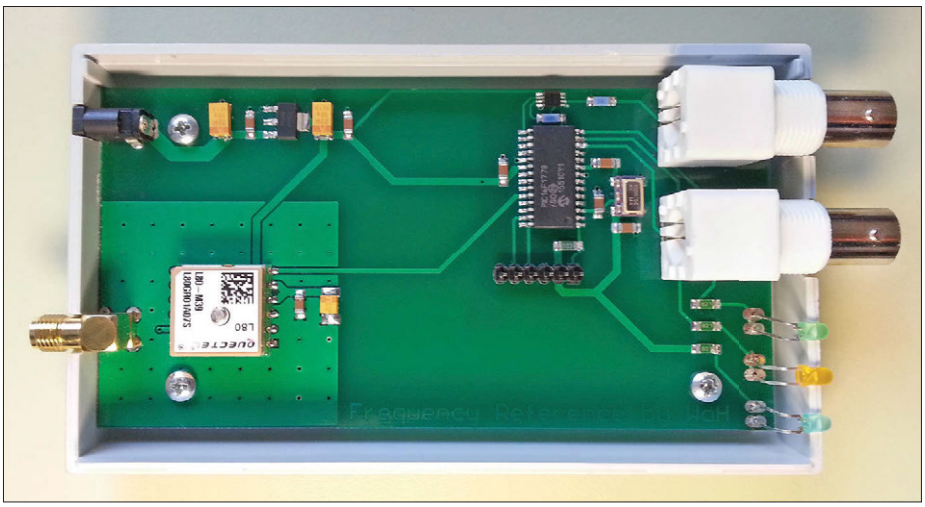


Figure 4. A view of the exemplary construction of the author.

The heart (and the brains) of the circuit is formed by a PIC16F1778 from Microchip [2]. This marvel in a 28-pin package has a few features that we gratefully put to use in this circuit. For example, the timers can be used completely independently of the processor core. Additionally, this IC possesses four configurable logic cells (CLCs); in this circuit three of these are used as flip-flops. The interconnects between the timer/counters and flip-flops are made through the use of configuration instructions.
The divide-by-five in this application is implemented with timer2; the divide-by-17 uses timer6 of the PIC.

32-bit counter

The 32-bit counter utilises timer1 of the PIC microcontroller. A small problem with this is that timer1 is only a 16-bit counter. Fortunately the solution is not particularly difficult: the overflow of this timer generates an interrupt; in the accompanying interrupt service routine (ISR) two registers are then incremented.

Synchronisation

For our purpose, only the lower two bytes of the 32-bit counter are important for synchronising the VCO; the two upper bytes are less important — we use these only to verify whether the past period of 16 seconds was indeed sixteen seconds long. If this is not the case then apparently some pulses from the GPS receiver were missed.
At the end of the 16-second period, the 32-bit counter stops, and at the same time an interrupt is generated. The accompanying service routine reads the counter value and calculates the difference between the actual and the theoretically ideal values. This difference is then used to control three digital-to-analogue converters (DAC) that generate the control voltage for the VCO. At the same time the counter is reset.

One DAC is not like another

In addition to all the other things, the PIC16F1778 has six DACs on board – three 5-bit implementations and three 10-bit converters. A resolution of 10 bits not enough to control the VCO with sufficient accuracy for our purpose; that is why we use the three 10-bit variants in combination. This works as follows: two of the 10-bit DACs are used for the high (DAC5) and low (DAC1) reference voltages for the third 10-bit DAC (DAC2). In a first step the reference voltages are set with a resolution of 8 bits. Subsequently, the absolute fine-adjustment is made using the third DAC. In this way we achieve a theoretical resolution of 18 bits.
As an aside: to prevent the proper operation of the DACs from being affected by the load, these are followed by opamps that are configured as voltage followers.
Once the DACs have generated the control voltage for the VCO, a period of sixteen seconds elapses so that the VCO can stabilise. After that the frequency is measured again and the VCO is adjusted again, if necessary — and so on. Consequently the frequency is measured every 34 seconds and the VCO gets adjusted.

A few details

The author in his implementation of the circuit used an L80-module from Quectel [3] as the GPS receiver. This attempts to receive 66 satellites, one after the other. After three satellites have been identified it takes 5 to 10 minutes before the VCO is locked to the GPS signal. The L80 has a connection for an external (active) antenna; this can come in useful when the receive circumstances are poor.
The voltage controlled oscillator is, to be precise, a VCTCXO (sometimes also called a TCVCXO) — a Voltage Controlled Temperature Compensated Crystal Oscillator with a frequency of 20 MHz (see also **Figure 2**). The output signal of which is also used as the clock source for the microprocessor.
Both the L80 as well as the VCO operate from 3.3 V; that is why the PIC is also powered from this voltage. The 3.3 V is supplied by an LDO (Low Drop Out) voltage regulator, which in turn is supplied with an input voltage of 5 V from a mains power adaptor (like one that is also used to charge phone batteries).
The 10-MHz and 1-MHz outputs from the PIC (see block diagram) do not go directly to the BNC output sockets; there are 3.3-V line drivers in between (IC3A and IC3B in the schematic of **Figure 3**). Resistors in series with the outputs of these drivers ensure an output impedance of 50 Ω.

The circuit is completed with three LEDs. The green LED D3 signals the presence of the 3.3-V power supply voltage. The other two LEDs (yellow and green) together indicate the state of the VCO. When both these LEDs (D1 and D2) are

Web Links

[1] www.ti.com/lit/ds/symlink/cd4046b.pdf
[2] ww1.microchip.com/downloads/en/DeviceDoc/40001819B.pdf
[3] www.quectel.com/UploadImage/Downlad/L80_Hardware_Design_V1.1.pdf
[4] www.elektormagazine.com/160594

About the author

Wilem den Hollander, who this year will celebrate his 74th birthday, has been busy with electronics since his 12th year. After his studies in electric eenginerring at Delft University, Holland, he moved to Switzerland to work on consumer electronics in the R&D department of an American company. During the course of his career he has obtained 20 patents. Even now in retirement he is still busy with electronics.

off then no GPD signal is being received. As soon as a GPS signal is received the yellow LED will flash at the rate of the 1-pps signal. Once the counter value differs by only 1 count from the ideal value the yellow LED will turn on continuously. And once the counter value is exactly equal to the ideal value the green LED turns on continuously.
Finally, in the schematic of **Figure 3** connector K2 can be seen; this is for programming of the microcontroller (In Circuit Serial Programming, ICSP).

Software

The author has written the firmware for the GDP-based frequency reference in assembler — not only because he has programmed only in assembler during his entire professional life, but also because this results in very compact, well-organised and easy to debug code.
As is customary, those who are interested can download the code (in the shape of a .asm and a .hex file) free from the project page for this article [5].

Construction

The author built his version on a double-sided circuit board with dimensions 116 x 60 mm (fitting an enclosure of 124 x 72 x 30 mm) — see **Figure 4**. You may think that the PCB is rather large for so few components (and most of them are SMD versions), but there is a good reason for that. It has to be prevented that the GPS receiver potentially receives interference from the VCO; that requires a considerable (physical) separation, combined with a generous ground plane (which is easily recognisable in **Figure 4**).

Performance

Once the GPS receiver starts to deliver second pulses, the output frequency (at BNC connectors J3 and J4) will be stable after about 5 to 10 minutes. With a clear sky or when the receiver is outdoors (which, however, in practice will not be that common…) or when it is close to a window, the reception of the GPS signal will be excellent. With bad weather and/or indoors an active antenna will likely be necessary, which is preferably installed as close to a window as possible.
The VCO is also sensitive to variations in the power supply voltage. When a 50-Ω load is connected the frequency will drift slightly. However, after a short time the output frequency will be stable (and locked) again.

To be continued...

Because there was so much space left in the memory of the microcontroller, the author has extended the circuit with a self-calibrating frequency counter. This requires only a little bit of additional hardware — mainly a display plus controller and a few operating buttons. This expansion is dealt with in the next edition of Elektor.

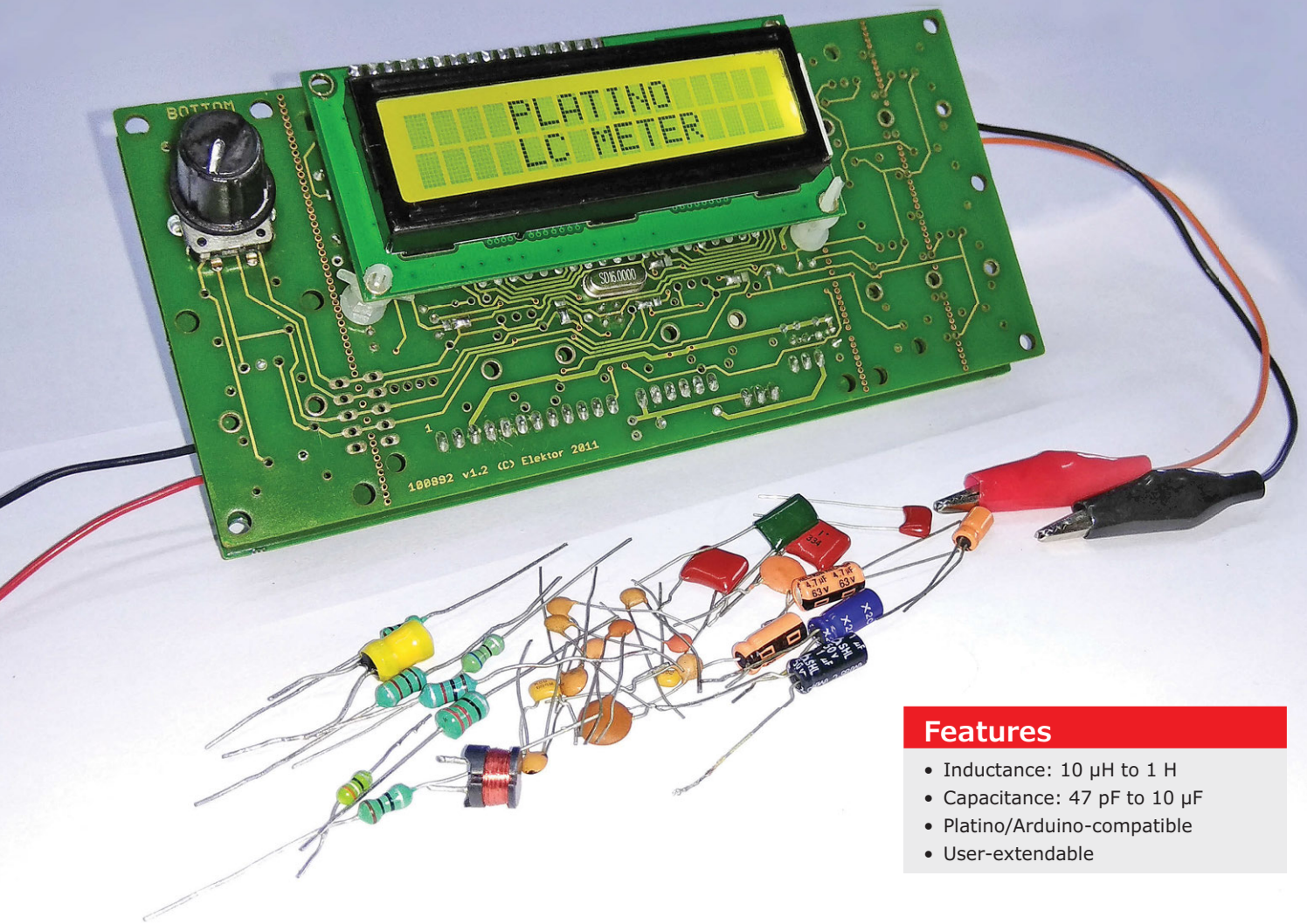
(160594)

Simple LC Meter

Measure inductance and capacitance with Platino

By **Sunil Malekar** (Elektor Labs)

Like it or not, using a microcontroller often helps to simplify a hardware design. Of course, what is left out of the schematic has to be done in software, but arguably that only increases the circuit's flexibility. Here is a small project that shows how to measure inductance and capacitance with a microcontroller and a few extra parts.



Features

- Inductance: 10 µH to 1 H
- Capacitance: 47 pF to 10 µF
- Platino/Arduino-compatible
- User-extendable

Inductors and capacitors are two types of components that have several things in common, yet they are very different. For instance, both can store energy even though it is stored as a magnetic field by the inductor whereas a capacitor stores electric charge. Another thing they have in common is that it is hard to know their exact value. Many multimeters can measure capacitance (*C* in pico-/nano-/micro-/milli-farad), but only a few do inductance (*L* in micro-/milli-henry). You have to buy an L meter to measure inductors. L meters usually also measure capacitance and resistance and therefore are often called LCR meters. We will limit ourselves to L & C and leave resistance to the trusty multimeter.

Measuring resonances

Various methods can be employed to measure inductance and capacitance; in this article we will use the resonance approach. Consequently we use the device under test (DUT) to create an oscillator and then measure the frequency of the oscillator's output signal. Because we know the oscillator's char-

acteristics it is possible to use this frequency to calculate the inductance or capacitance value of the DUT.

Two oscillators and clever software is all we need

The oscillators are shown in **Figure 1**. There are two, one for inductance and one for capacitance. IC1.A and IC1.B together with L_x, C1 and C2 form a digital Colpitts oscillator. When the two capacitors have the same value, as they do here, the oscillating frequency is calculated as:

$$f = \frac{1}{2\pi\sqrt{(0.5\,L_x\,C)}} \quad [\text{Hz}]$$

We can work this equation around to obtain:

$$L_x = \frac{1}{2\pi^2 f^2 C} \quad [\text{H}]$$

Because the frequency is a bit high when L_x is small, it is divided by 16 by IC2.A before being sent to the microcontroller. Resistor R1 prevents the oscillator from operating in the absence of L_x while P1 is available to calibrate the frequency with an accurately known L_x.

The second oscillator is of the relaxation type. The frequency of this oscillator is given as:

$$f = \frac{1}{2\,R_4\,C_x \ln\left[1+2\left(\frac{R_2}{R_3}\right)\right]} \quad [\text{Hz}]$$

Swapping f and C_x gives:

$$C_x = \frac{1}{2\,R_4\,f \ln\left[1+2\left(\frac{R_2}{R_3}\right)\right]} \quad [\text{F}]$$

This equation is pretty precise but the switching levels of IC3.A do affect it.

PROJECT INFORMATION

Inductance · Capacitance · Platino · Arduino · Test & Measurement

➡ entry level
intermediate level
expert level

4 hours approx.

Solder iron,
Arduino IDE,
USB-to-Serial cable

€50 / £45 / $63 approx.

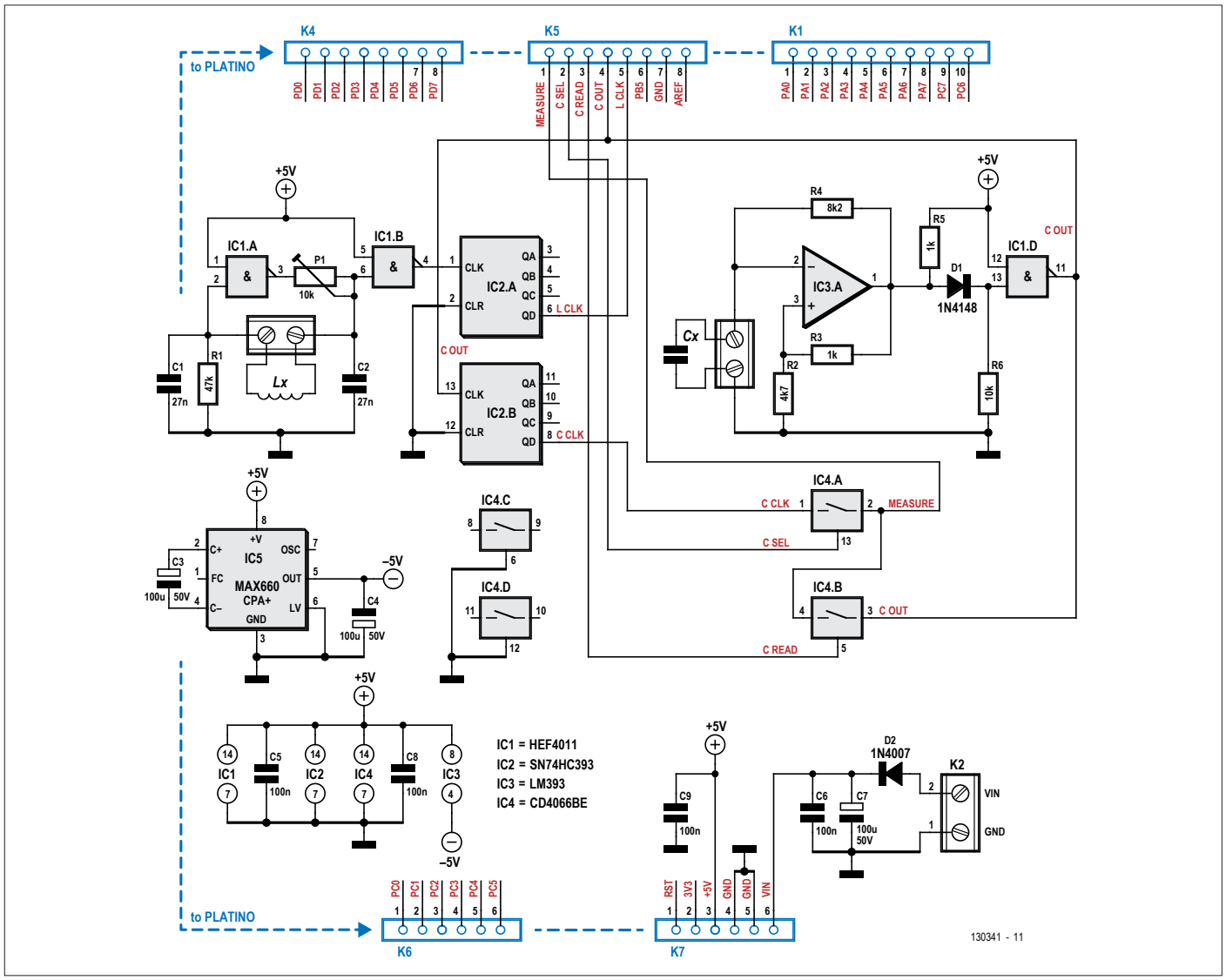


Figure 1. The Platino add-on board houses two oscillators and some support circuitry.

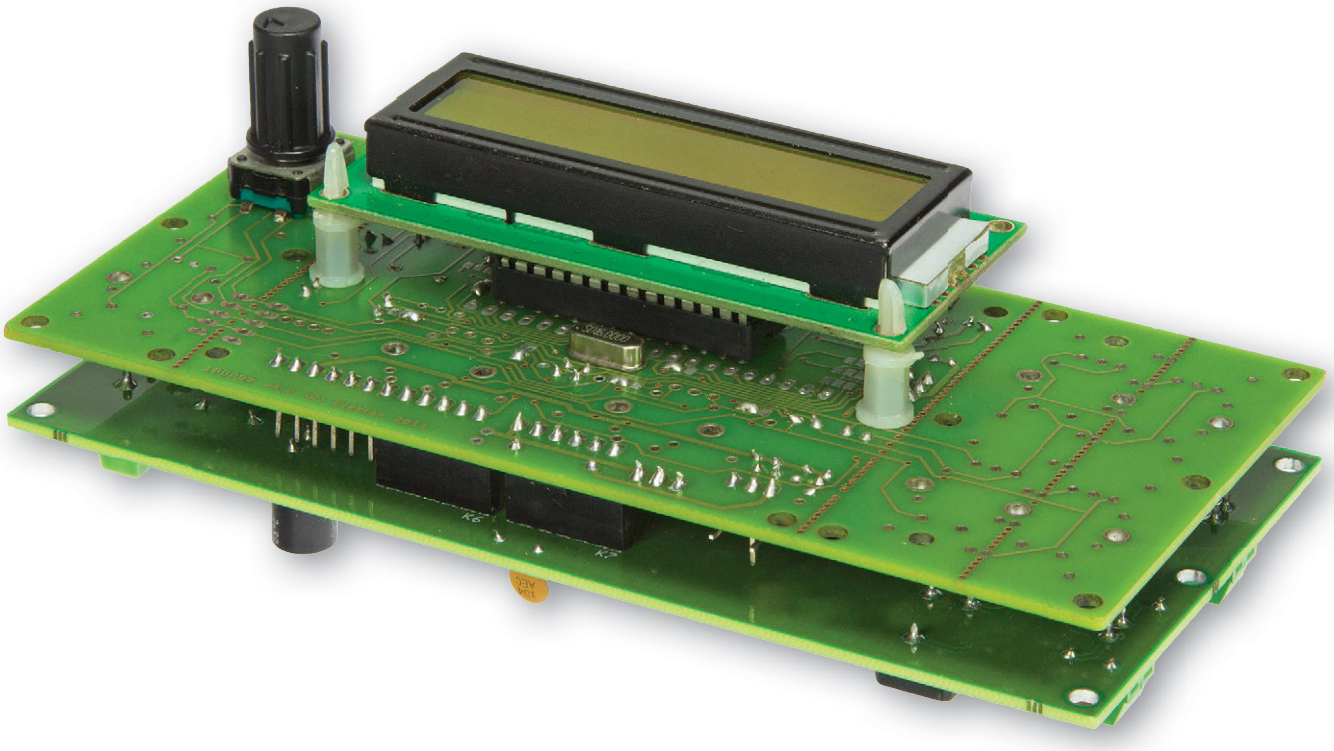


Figure 2. Platino plus add-on board. On Platino, 28-pin MCUs are mounted at the solder side, under the LCD.

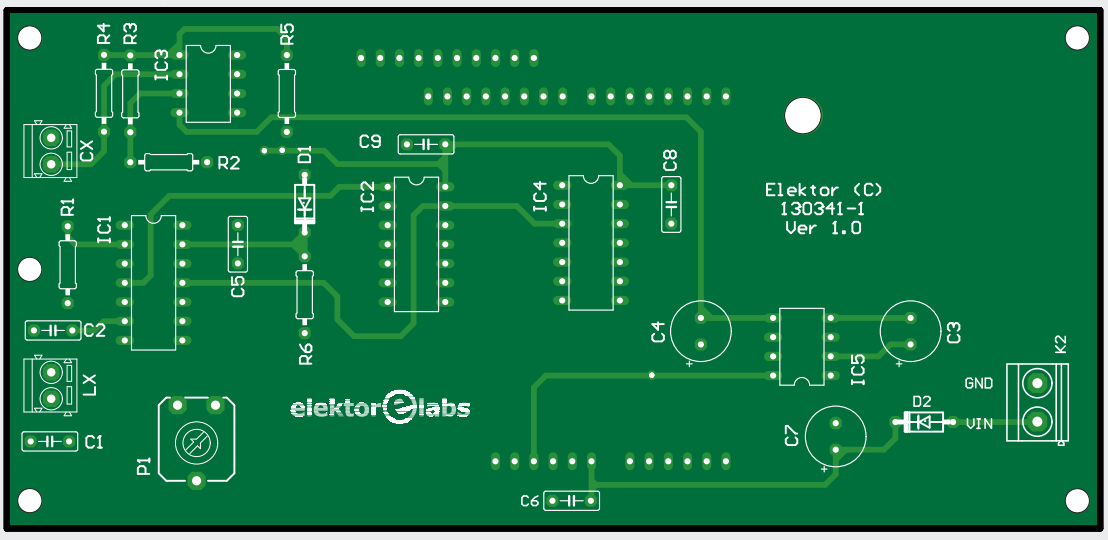
COMPONENT LIST

Resistors
R1 = 47kΩ
R2 = 4.7kΩ
R3,R5 = 1kΩ
R4 = 8.2kΩ
R6 = 10kΩ
P1 = 10kΩ trimpot

Capacitors
C1,C2 = 27nF, 5mm pitch
C3,C4,C7 = 100µF 50V, 3.5mm pitch
C5,C6,C8,C9 = 100nF, 5mm pitch

Semiconductors
D1 = 1N4148
D2 = 1N4007
IC1 = HEF4011
IC2 = SN74HC393
IC3 = LM393
IC4 = CD40676BE
IC5 = MAX660CPA+

Miscellaneous
LX,CX = 2-way PCB screw terminal block, 3.5mm pitch
K2 = 2-way PCB screw terminal block, 0.2" pitch
K1 = 10-way (1x10) pinheader socket, 0.1" pitch (optional, mount at solder side)
K4,K5 = 8-way (1x8) pinheader socket, 0.1" pitch (mount at solder side)
K6,K7 = 6-way (1x6) pinheader socket, 0.1" pitch (mount at solder side)
Optional: 14-way DIL socket for IC1, IC2, IC4
Optional: 8-way DIL socket for IC3, IC5
PCB # 130341-1



Even though the IC is powered from a symmetrical power supply this does not solve the problem completely. Luckily the software can compensate for this after calibration.

Rectifier D1 together with IC1.D afford an output signal between 0 and 5 volts. The negative supply for IC3 is created by IC5, a popular voltage inverter IC.
The multiplexer built with IC4.A and IC4.B lets the MCU select the best capacitor test frequency for both a rapid response (the larger the capacitor value, the lower the frequency and thus a longer measurement period) and best precision. Those who prefer simplicity can leave IC4 off the board and strap its pins 1 and 2.

The software does the hard work

From here on the MCU takes over. It is programmed using an Arduino sketch, making it easy for anyone to modify and play with. As is usual for a device with a nice user interface (UI) most of the program is dedicated to the UI. I mean:

- displaying values with the right units, at the right position on the display;
- handling the rotary encoder and its integrated pushbutton.

At least 80% of the programming effort is down to the UI.
To measure the frequency produced by the DUT, the sketch uses Arduino's `pulseIn` function. This is a rather precise way of doing things as it returns values accurate to the microsecond. The inconvenience is that it is a blocking function, meaning that the MCU cannot do anything else while it is executing it. In our case this is not a problem, because the MCU doesn't have to do anything else in the meantime.

Measuring only the pulsewidth therefore would suffice, but for better precision, and because things are never ideal, the signal period is measured also. The frequency value obtained, corrected for any division factors, is plugged into one of the equations above (L_x or C_x) and the value of the DUT (device under test) is output. It's that easy.

Build the LC meter

Finally, a word on assembling the LC meter. The add-on board is not particularly difficult to populate. The only thing to keep in mind is that K1 (optional) and K4 through K7 are to be mounted at the solder side of the board.
Also, an assembled Platino is required (**Figure 2**), see the **inset about Platino** for the details. Because the add-on board does not need K1 (it is there just in case), the board can also be used with an Arduino Uno. If so, you will have to change the user interface. Controlling the LC meter over a serial port is possible and makes for a good coding exercise. ⏮

(130341)

A quick word about Platino

Platino is a universal circuit board for AVR microcontrollers in DIP28 or DIP40 packages. It can be rigged with user interface objects like pushbuttons, LEDs, buzzer, and displays. Everything, and we mean everything, about Platino can be found on the Elektor Labs website (see below) and on the GitHub repository of Elektor Labs.
Experience has shown us that Platino users find it hard to configure its solder jumpers. This is understandable, as there are many. This table shows the settings for this project. The Position column corresponds to the labels printed on Platino PCB version 150555-1 v1.4. Note that we recommend installing solder jumpers before mounting any parts.

Jumper	Position	Function
JP1	C	PB4 used by add-on board
JP2	-	Non-existent
JP3	C	LCD backlight on PC5
JP4	C	Rotary encoder A on PC0
JP5	C	Rotary encoder B on PC1
JP6	C	Rotary encoder pushbutton on PC2
JP7	C	PB3 used by add-on board
JP8	DIP28	PC6 is reset input
JP9	XTAL	Use 16-MHz crystal
JP10	XTAL	Use 16-MHz crystal
JP11	DIP28	ISP SCK on PB5
JP12	DIP28	ISP MISO on PB4
JP13	DIP28	ISP MOSI on PB3
JP14	B	LED1 blue is standard Arduino LED
JP15	D	LCD RS on PD2
JP16	D	LCD E on PD3

And while we're on the subject of parts: for this project we need a fully assembled Platino with a 28-pin MCU (ATmega328P-PU) — hence no need for a 40-pin socket —, a 2x16 character alphanumerical LCD with backlight, a rotary encoder with integrated pushbutton on position S5A or S5C (depending on your preference). The Platino bill of materials (BOM) can be found here:

www.elektormagazine.com/labs/platino-versatile-board-for-avr-microcontrollers-100892-150555

Web Link

[1] www.elektormagazine.com/130341

About Battery Chargers, Choices and Electronic Designers

By **Jan Visser** (Elektor Labs)

A while ago someone in the lab suggested publishing another battery charger project. After all, our readers always find that interesting, and lots of them build our projects themselves. Some of our battery charger designs are still in demand decades years after the initial publication. So it seemed like a good idea, and given the current state of electronics technology, not especially difficult to do. We just had to brainstorm a bit on what sort of battery charger to develop.

That quickly turned into an animated discussion. One designer said it had to be able to charge lead-acid batteries, and another retorted that lead-acid batteries are outdated and we should make a charger for lithium ion cells. “And my NiMH cells?”, asked a third colleague. “Surely it has to be able to charge them too?” Colleague number four piped up with “What about NiCd and LiPo?” It quickly became clear that our charger had to be able to handle the full spectrum of batteries — from lead-acid to lithium polymer.

Then the discussion turned to the next question: how many batteries in series or in parallel? The first designer said, “I have a scooter with four lead-acid batteries, so the charger should at least be able to handle that.” The second said, “I don’t need that; I have model boats and they run on 14.4 V LiPo batteries.” Number three said, “My electric bike runs like a charm with twenty NiMH cells, but I want to be able to charge each cell separately.” And for my part, I have a Prius with a 202-V NiCd battery, and I might want to be able to charge it sometime.

There seemed to be no end to the discussion. Should we charge all the cells separately, or charge a group of cells in series and then balance them? And should we use the integrated battery management system of the battery pack, or use a BMS built into the charger?

The power supply question also led to a heated discussion. One designer wanted a 500 W supply because you need that if you want to charge a bunch of lead-acid batteries reasonably quickly, while another only needed 40 watts to charge their model boat batteries within one hour. Another issue was whether the power supply should be integrated into the charger or be a separate module. One designer preferred a heavy-duty charger for their home lab, while another would rather have a smaller, handier charger that is easy to carry.

Ultimately I decided to poll our readers on the Elektor Labs site to see what they wanted. Perhaps we were making things too difficult and our community had a definite opinion and a preference for a particular type of charger. That got an impressive 16 Kviews within a short time — and comments, I quickly realised that within our community there are also many different opinions. It’s nice to know that our lab is

a good mirror of our international electronics community and is aware of all the parameters, but that doesn’t make choosing any easier.

Of course, in our era of abundant electronics we have a wealth of components to choose from if we want to build a battery charger. Along with ICs specifically designed for battery charging, microcontrollers nowadays have more than enough resources and I/O to make very nice chargers. You might even say that we have too many options available now, and as electronics designers we are able to build a suitable charger for virtually every situation. Even so, designing a universal charger that can do everything for everyone looks like the impossible dream — because almost everyone has their own requirements.

In that situation designers tend to be guided by personal preferences, making objective choices difficult. That gives you an idea of the dilemmas we have to deal with before a design leaves the lab to help make the world a bit better. Whether the ‘ultimate charger’ will actually come to be, and in what form, is still an open question. If you have an idea or a suggestion about this, please feel free to let us know on our website [1]. ⏮

(160644-I)

Web Links

[1] www.elektormagazine.com/labs/batterycharger-new-style-which-what-why

VFD-tube Clock with ESP32

with an accurate Internet-derived time

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

Clocks built using vintage components are currently very popular. Most designs are based on a real-time clock, synchronized via radio time servers or GPS. The VFD clock described here uses an NTP server on the Internet to keep the time synchronized.

Features

- 9 V_{DC} supply, maximum 3 A
- displays hours, minutes and seconds, the date and the temperature
- several display formats for the time, date and temperature
- supports DS18B20, DS18S20, DS1822, LM75 and SE95 sensors
- user configurable RGB background lighting
- 1-wire and I^2C interfaces
- night mode
- complete kit available

Nothing is more annoying than a clock that shows the wrong time, although it's not really that much of a problem if the time is out by only a few seconds. It's a completely different matter when the discrepancy runs to tens of seconds or over a minute. Before you know it, all the clocks will show a different time, and you can no longer tell what the correct time is. Besides, it's not much fun having to continuously adjust all the clocks to the correct time.

Ignoring the notoriously badly calibrated clocks in microwave ovens, most clocks that are based on standard crystals with an accuracy of 20 ppm to 50 ppm will deviate from the correct time by about one minute to over two minutes per month. Even when an exceptionally accurate real-time clock IC is used with

PROJECT INFORMATION

VFD tubes
TH components
Espressif ESP32-DevKitC

entry level
➡ intermediate level
expert level

4 hours approx.

normal soldering tools
computer with Arduino IDE

€150 / £130 / $185 approx.

an accuracy of 3 ppm the deviation can be as much as 7.5 seconds per month. This may not seem like a lot, but over one year it adds up to one and a half minutes. To prevent time deviations, as well as for ease of use, some modern clocks use external sources to synchronize the time. Usually these are radio time servers (such as DCF77, NPL, WWVB, JJY, etc.) or GPS. Unfortunately, both these methods have disadvantages. A relatively large ferrite antenna is required to receive radio time server signals. These can be affected by interference from switching power supplies and the like. The reception of GPS signals inside the home can often be a hit and miss affair, which means that this is also not always a suitable option.

Our VFD clock (in case you're wondering, this stands for *Vacuum Fluorescent Display*; VFD tubes are covered in the Question & Answer section of this year's March/April issue of Elektor [1]) is built around an ESP32-DevKitC module made by Espressif. This Arduino compatible board not only has a powerful processor, but also has Wi-Fi and Bluetooth onboard. Most homes and offices nowadays have access to a Wi-Fi signal; for this reason we decided to use a WI-FI connection and an Internet time server (NTP) to synchronize the time. When the clock is switched on, it can make a connection to the network and synchronize with the NTP server in as little as one second, depending on the WI-FI network and the signal strength. This is significantly better than synchronization via radio time servers or even via GPS. A backup battery for keeping the time during a power cut is therefore completely unnecessary.

The clock has a modular construction and consists of a main board, a display board and a board for the RGB background lighting of the VFD tubes.

Circuit diagram

Display board

The display board is the most visible section of this clock. Apart from the VFD tubes, there are a few resistors and connectors for the connection to the main board. We use six Russian IV-22 7-segment VFD tubes for the display of the date and time, in combination with two DM160 (or equivalent) VFD indicators that provide the flashing lines between the hours, minutes and seconds.

The IV-22 VFD tube has an oval shape and is viewed from the top. Since its shape and size are the same as those of the IN-12 nixie tubes, we decided to use the same layout as used in the New Precise Nixie Clock, published in the May/June 2016 issue of Elektor [2]. Since standard box-headers have been used to connect the display board to the main board, it shouldn't be too difficult to connect a home-made display board using a different type of VFD tube.

Another reason why we selected the IV-22 VFD tubes is that they're widely available. However, in contrast to the

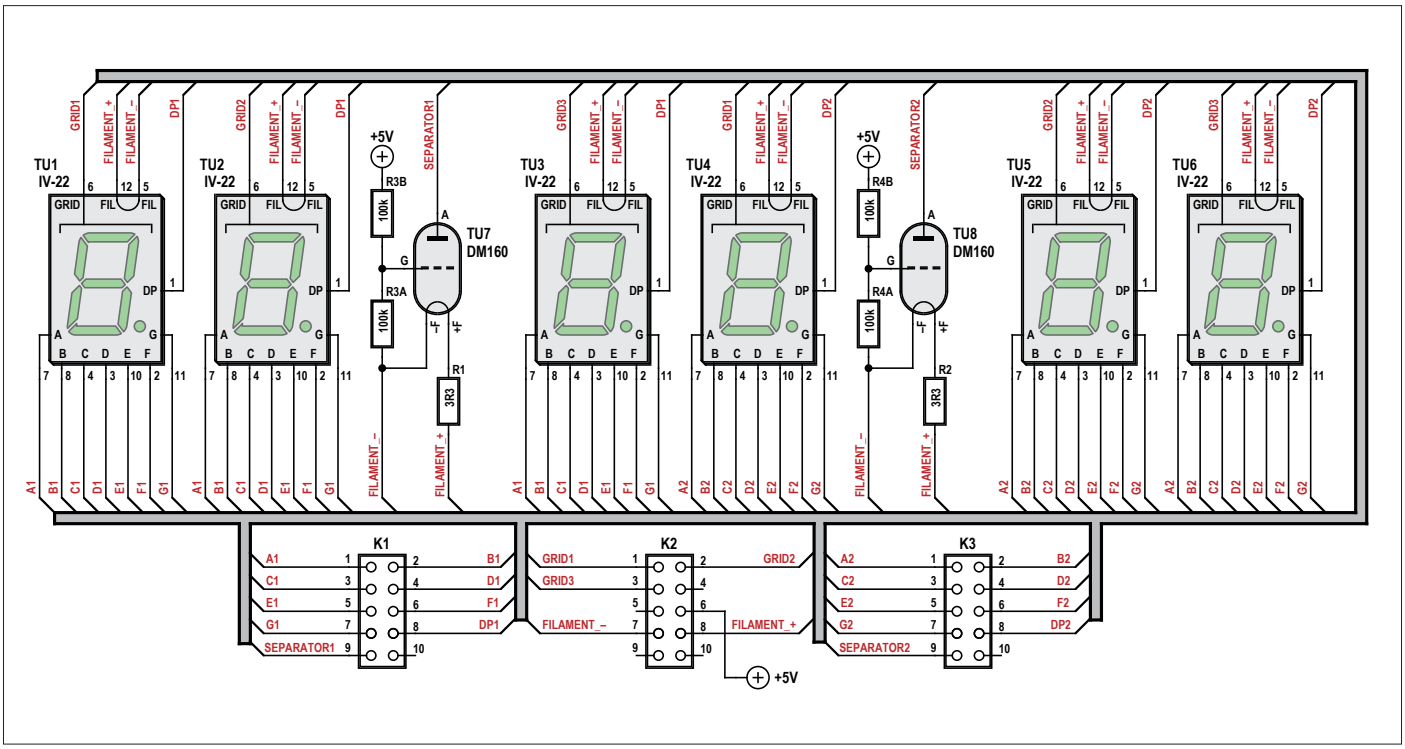


Figure 1. The circuit diagram for the display module.

COMPONENT LIST DISPLAY BOARD

Resistors

R1,R2 = 3.3Ω *
R3A,R4A,R3B,R4B = 100kΩ *

Miscellaneous

TU1–TU6 = IV-22 (Russian)
TU7,TU8 = DM160 or equivalent / IV-15 (Russian) *
K1,K2,K3 = right-angled pinheader 0.1" pitch
72 pcs tube socket pin 1mm
heatshrink sleeving, diameter 1.6mm 1:2 ratio
heatshrink sleeving, diameter 6mm 1:2 ratio
2 x (half) fuse holder 5x20 mm (Farnell 1866099)

Mechanical parts

2 x aluminum standoffs, round, 4.5x14mm (Ettinger 05.72.148)
2 x standoff, 15mm M3 F/F
4 x bolt M3x6 countersunk, steel, pozidriv DIN 7985A
8 x nylon bush M3

* see text

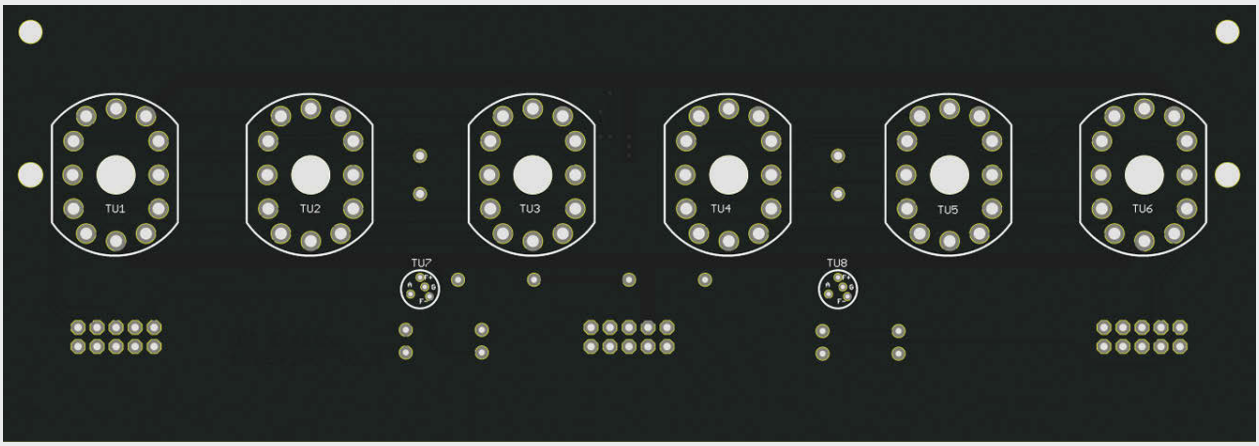


Figure 2. This board was designed for the display.

IN-12 nixie tube, the pins of the IV-22 are not easy to solder; for this reason we decided to use tube socket pins to mount the tubes onto the board. This has the advantage that the tubes can be easily removed in case there is a problem, without any tricky desoldering that could cause damage.
The DM160 is a small VFD indicator tube made by Philips, with a diameter of 5 mm and a length of about 26 mm. Other DM160-compatible tubes such as the CV5412, CV6094 and 6977, made by various manufacturers (Mullard, Amperex, Sylvania...), can be used without any problems. The Russian IV-15, which isn't fully compatible, can be used with a few modifications.
The DM160, introduced in 1959, is a triode; its anode is made from a spirally wound metal wire coated with phosphor. The intensity of the green light from this wire depends on the grid voltage (–3 to 0 V). The original application for the DM160 was as an indicator in transistor-based computer systems. As far as we could determine, such tubes were

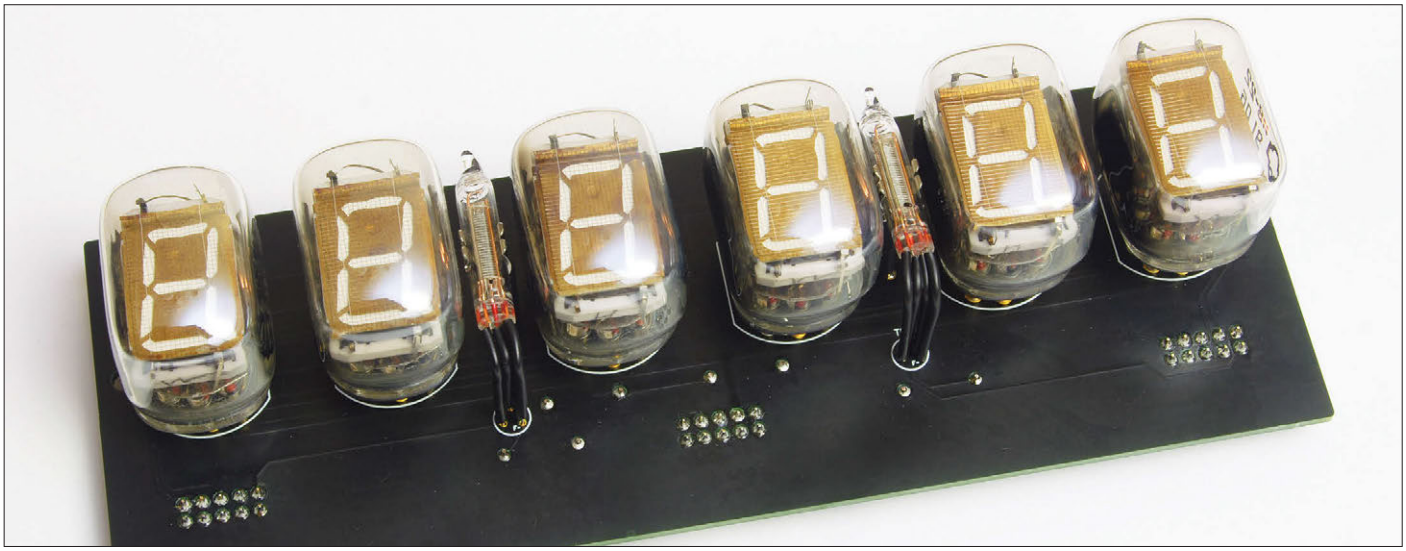


Figure 3. The fully assembled display module.

used in systems like the Bull Gamma 10, the Marconi TAC (*Transistorised Automatic Computer*) and the Elliott 803B. These days we would of course use LEDs for such applications.

The indicator tube is held in place on the board using (half) a fuse holder for 5 x 20 mm fuses, and an aluminum stand-off, 14 mm long. We found that it makes it simpler to control the tube when the grid is permanently connected to 0 V and the anode voltage is switched. Since the DM160 requires a slightly lower filament voltage than the IV-22 tubes, a 3.3 Ω resistor has been connected in series with the filament.

Figure 1 shows the circuit diagram for the display board, **Figure 2** shows the board with a fashionable black finish and **Figure 3** shows the fully populated board.

Main board

The circuit diagram for the main board may look a bit daunting at first, although it becomes much clearer once we've looked at each section in turn.

The anode voltage for the VFD tubes is supplied by a step-up converter built

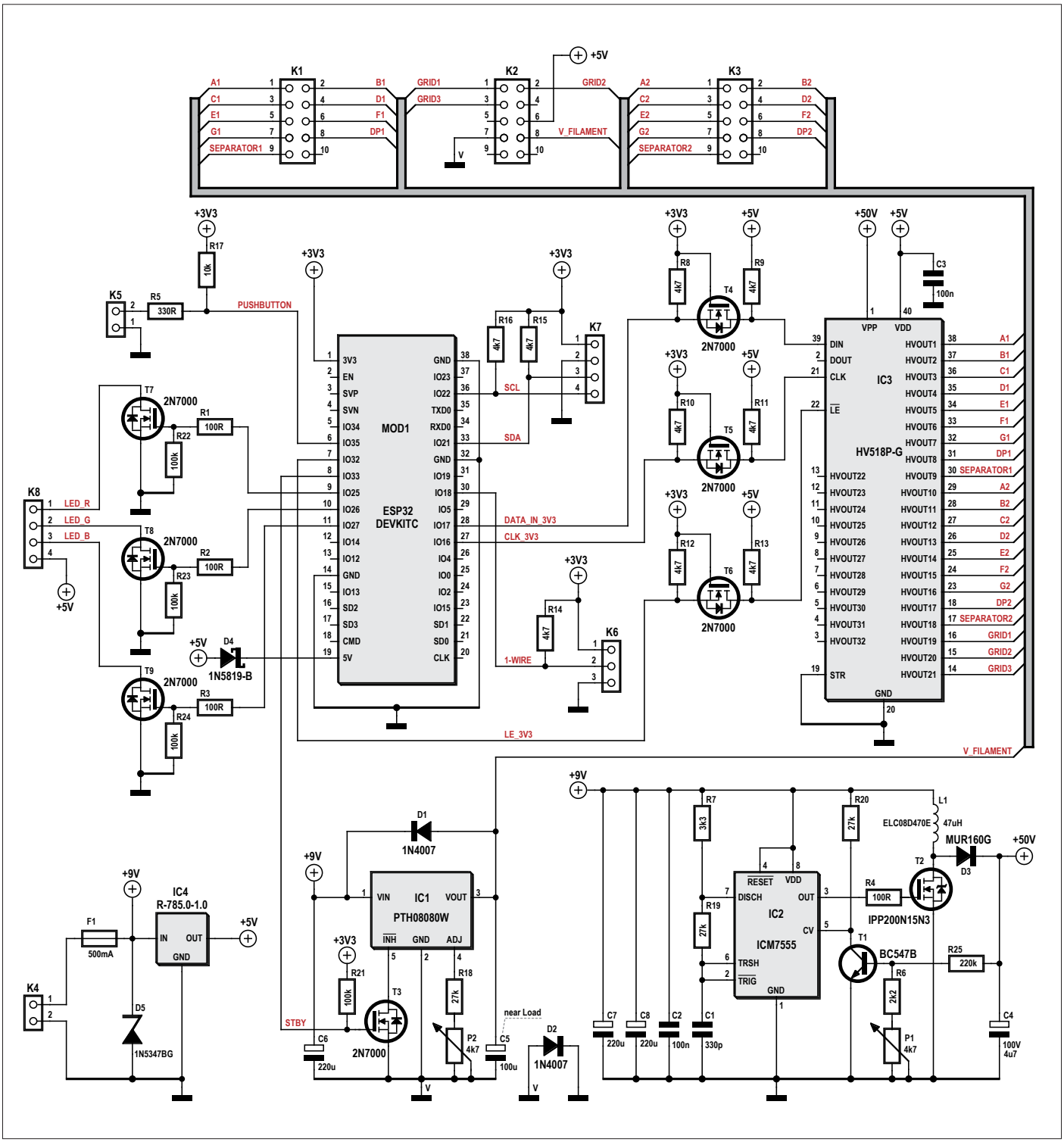


Figure 4. The circuit for the main board is less complicated than it appears.

COMPONENT LIST MAIN BOARD

Resistors

R1–R4 = 100Ω
R5 = 33 Ω
R6 = 2.2kΩ
R7 = 3.3kΩ
R8–R16 = 4.7kΩ
R17 = 10kΩ
R18,R19,R20 = 27kΩ
R21–R24 = 100kΩ
R25 = 220kΩ
P1 = 4.7kΩ preset (CB10LV472M)
P2 = 47kΩ preset (CB10LV473M)

Capacitors

C1 = 330pF, NP0
C2,C3 = 10nF
C4 = 4.7µF 100V, radial, low ESR
C5 = 100µF radial
C6,C7,C8 = 220µF, radial, low ESR

Semiconductors

D1,D2 = 1N400x
D3 = MUR160G
D4 = 1N5819
D5 = 1N5347BG
T1 = BC547B
T2 = IPP200N15N3 G
T3–T9 = 2N7000
IC1 = PTH08080WAH
IC2 = ICM7555
IC3 = HV518P-G
IC4 = Würth type 173010578, 5V stepdown regulator
IC5 = DS18B20 (optional, not on the board)
MOD1 = ESP32-DevKitC + 2x20-pin pinheader

Miscellaneous

L1 = 47µH, 1.2A, 0.067Ω, radial, ELC08D470E
F1 = polyfuse 0.5A, 60R050XPR
K1,K2,K3 = 10-pin pinheader, 0.1" pitch (2x5)
K4,K5 = 2-wayPCB screw terminal block, 5mm pitch
K6 = right-angled 3-way wire-to-board connector (Würth 61900319521) + female connector (optional)
K7 = right-angled 4-way wire-to-board connector (Würth 61900419521)
K8 = 4-way bus strip, 0.1" pitch
1 x SPST/NO pushbutton
1 x coaxial DC power connector 5.5/2.1mm (PC-010 TME)
red and black connecting wire
heatshrink sleeving, 3.2mm 1:2 ratio

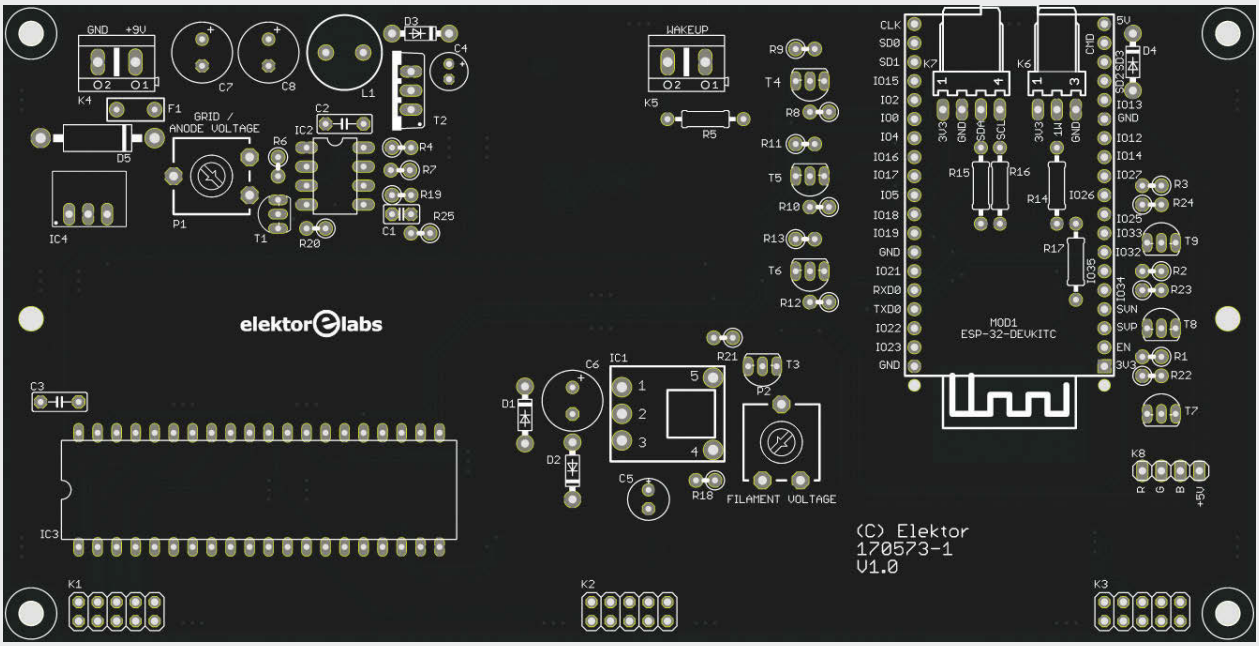


Figure 5. The main board is finished in a solemn black.

around a 7555 (IC2), a design that has also been used in earlier circuits. The advantage of this circuit is that the board layout is not critical and that the conversion efficiency is quite good despite the low cost of the components. The output voltage can be adjusted with P1 to a level between about 23 V and 71 V.

For the filament voltage we decided to use a ready-made module in the form of the PTH08080W, made by Texas Instruments (IC1). The price of this module is such that we found it impossible to build an equivalent circuit with through-hole components at a lower cost. It would have been possible if we used a modern switching converter IC, but these are generally only available in SMD packages, which are tricky to solder. P2 is used to adjust the filament voltage between about 1 V and 1.2 V. Via T3 the module can be put into standby mode, which is used when the filaments need to be turned off in night mode. The ground connection of the module is connected to the circuit ground via diode D2. This causes a DC-offset to be added to the filament voltage. As a consequence, when the anode segments (and grids) of the VFD tubes are turned off, they will be slightly negative compared with the filament, which guarantees that they will be fully turned off. According to the datasheet for the PTH08080W, the input voltage to the module must not be higher than ten times the output voltage. In our circuit that is therefore between about 10 V to 12 V, depending on the setting of P2. For this reason we've added an input overvoltage protection in the form of Zener diode D5 and polyfuse F1.

Since both the anode voltage as well as

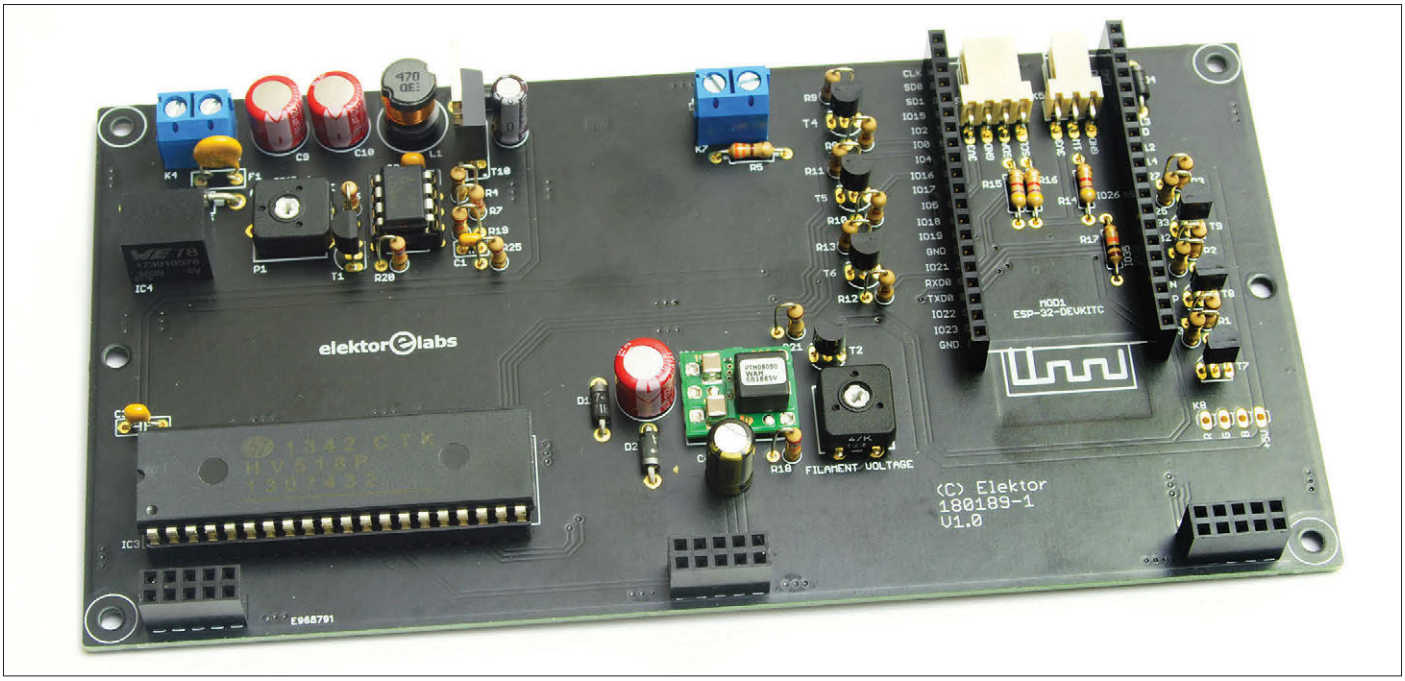


Figure 6. The main board, still without the ESP32 module.

the filament voltage can be adjusted, it should be possible to use other types of VFD tube with the main board.

The 5 V supply voltage for IC3 and the ESP32-DevKitC is generated by a switch-mode 7805 replacement (IC4). This can supply 5 V at 1 A.

The tubes are driven via IC3, a HV518P-G high-voltage shift register, specially designed to drive VFDs. The IC has 32 outputs, of which we use 21; it can switch voltages up to 80 V. At the input we have a data line, a clock and a latch signal.

The IC is supplied with 5 V, and the data lines require at least 3.5 V to recognize a logic 'high' signal. Since the I/O lines of the ESP32-DevKitC supply only 3.3 V (just too little), we've added transistors T4, T5 and T6 to convert the signal levels. This type of usage of a MOSFET is also often applied to an I²C bus.

According to the datasheet for the HV518P-G, the high voltage (VPP) should be applied *after* the 5 V supply voltage (VDD). The VPP should then remain and be at least 8 V (*recommended operating conditions*). We have therefore decided to keep the anode voltage switched on during the night mode.

The data sent to the VFD tubes has been multiplexed (3 x 2 tubes). This makes it necessary to continuously refresh the shift register in order to prevent any flickering of the display.

The ESP32-DevKitC module is supplied via D4 from the 5-V clock supply. This way, the module can also be connected to a computer via USB whilst the clock is switched on. The ESP32-DevKitC will then be simply powered by the higher of the two voltages.

There are also an I²C connector and a 3-way 1-wire connector for the connection of temperature sensors. Via the I²C bus you can connect an LM75 or an SE95. Via the 1-wire interface you can connect a DS18B20, DS18S20 or DS1822. You could modify the software to use the I²C bus for other creative applications and upgrades.

The RGB LEDs for the background lighting of the tubes are driven via three MOSFETs (T7, T8 and T9). And finally, there is a pushbutton that is used to temporarily activate the clock when it is in night mode.

Figures 4 to 6 show the details of the main board.

RGB background lighting

There is not much to say about the board for the background lighting of the tubes: it contains just six 5-mm RGB LEDs and

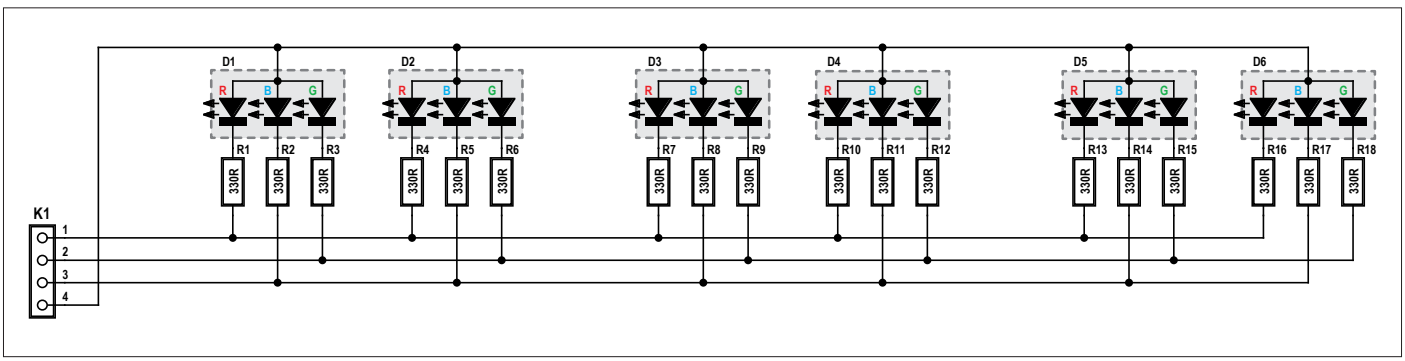


Figure 7. There is not much to the circuit for the background lighting.

some resistors. These (bright) LEDs shine through the corresponding holes in the display board. **Figures 7 to 9** show the details of the background lighting board.

Software

The program for the VFD clock was written using Arduino IDE v1.8.5. The sketch — an Arduino IDE project — is quite extensive and was therefore split into several .ino files. You should be aware that the Arduino IDE combines all .ino files into one large .cpp file during the compilation process. The .ino file with the project name comes first, which is followed by the other .ino files in alphabetical order.

The Arduino IDE v1.8.5 doesn't have native support for the ESP32. Espressif, the manufacturer of the ESP32, made the *Arduino core for the ESP32* [3] software package available. This is an extension to the Arduino IDE, which lets us develop software for the ESP32. The package contains the compiler and other tools for the Tensilica Xtensa LX6 microprocessor, which is at the heart of the ESP32. When this package has been installed, the Tools menu of the Arduino IDE will have many more entries in it. Select the 'ESP32 Dev Module' as your board.

The program uses libraries that are installed as part of the *Arduino core for the ESP32* package. These libraries add functions such as Wi-Fi functionality, a network stack based on lwIP and the essential interfaces for the ESP32 hardware in the Arduino environment. At a lower level there is an ESP32 version of FreeRTOS, which adds a type of multi-threading so that the Wi-Fi functionality and some other parts can work in the background.

The program has about 70 commands that you can send to it via the serial port or over the Wi-Fi connection. The serial port is configured at 115200 baud. If you use the serial monitor in the Arduino IDE, choose '115200 baud' and select something other than 'No line ending'. Once the VFD clock has established a connection to your Wi-Fi network, it becomes possible to send commands wirelessly via network port 5010 of the IP address of the VFD clock. The software can cope with a maximum of four simultaneous client connections. The file cmd_proc.ino contains the code and documentation for all of the commands. Many of the commands also have a shortened version: for example, the command *wifi start* can be replaced with *w s*.

When you have constructed the clock you will need to configure it via the serial port using a number of commands, including one to set up the Wi-Fi connection. We have created a text file for you, which you can use as a template to create your own configuration file. You can send this modified file to the clock using a serial monitor such as PuTTY, which will save you a great deal of typing.

The program has several tasks that are active at the same time. The tasks are executed from the main loop, the Arduino *loop()* function. The settings for most tasks can be modified using commands. The command *settings dump* (or the shortened version: *s d*) is used to request all settings.

The 'NTP' tasks requests the current date and time from the time server. The default server is *pool.ntp.org* [4]. An NTP server returns the UTC+0 time; this can

COMPONENT LIST RGB BACKGROUND LIGHTING BOARD

Resistors

R1–R18 = 330 Ω

Semiconductors

D1–D6 = RGB-LED 5 mm, common anode (eBay)

Miscellaneous

K1 = 4-pin pinheader, 0.1" pitch

white, red, green and blue connecting wire + extra pinheader

heatshrink sleeving, 3.2mm 1:2 ratio

Figure 8. This long board contains six LEDs and a handful of resistors.

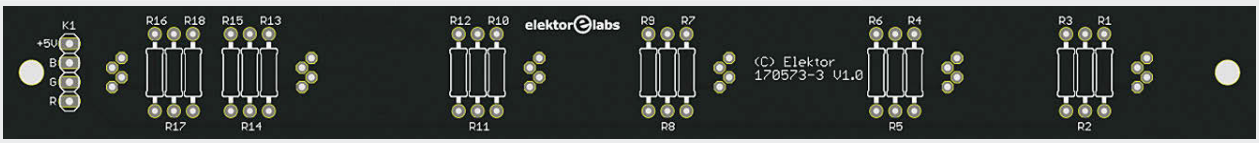


Figure 9. The background lighting.

be adjusted for the required time zone using the *clock set adjust* command.
The 'Clock' task deals with the actual clock functionality. This determines how and when the time, date and temperature are displayed.
The 'Display' task controls the shift register for the tubes and the 'RGB' task drives the RGB LEDs. Together, they take care of the visual aspects of the clock. Since the shift register needs to operate continuously to provide a flicker-free display, it is important that this task can't be 'blocked' by the rest of the software. The 'Main' task is exactly what it says: the main task for the clock. Each task makes use of one or more other tasks, with all tasks working together as one program.
You can freely download all the software and documentation from the project page for this article [5].

Modern Internet time with vintage display

Figure 10. The fully assembled clock, before it's been put in its enclosure. Note how the two DM160 tubes are mounted.

Construction

It doesn't matter in which order you populate the boards, so we decided to describe the boards in alphabetical order!

Display board

Mount resistors R1, R2 (3.3 Ω), R3**A** and R4**A** (100 kΩ) for DM160 or equivalent tubes. If you use the Russian IV-15, you should use values between 6.8 Ω and 10 Ω for resistors R1 and R2, and mount 100-kΩ resistors for R3**B** and R4**B** (R3A and R3B should *not* be mounted in this case).
Mount the right-angled headers (K1 to K3). Next, remove the 5x20 mm fuse holder from its cardboard strip without shortening its feet. Carefully cut the feet at the ends so that they have a uniform width, and remove the metal tab from the holder itself. Place a 14-mm aluminum standoff between the feet and put some heatshrink sleeving over it all. Make another tube-holder the same way and mount them on the board near TU7 and TU8. Then put thin heatshrink sleeving (20 to 22 mm long) over the leads of the small VFD tubes and bend them at right-angles 5 mm from the bottom of the tube. Thread the wires through the solder pads on the board and snap the tubes into their holders. Ensure that the orientation of the tubes and the position of the wires are correct. Solder the wires to the board and then use a multimeter to measure the resistance between pins 7 and 8 of connector K2. This should be around 6 Ω to 6.5 Ω (for the DM160); this indicates that the tubes have been wired correctly. You can see in the photos of Figure 3 and **Figure 10** how it should be done.
Then take the IV-22 tubes and put tube socket pins onto all the pins. Carefully put the socket pins through the solder pads of the board. If you prefer, you could put the individual socket pins through the solder pads first, and then push the tube into the socket pins. Once all six tubes are in their place, you should turn over the board and solder the socket pins to the bottom of the board. Don't use too much solder and avoid heating the pins for too long.
Finally, use a multimeter to measure the resistance between pin 7 of K2 and all other pins on the three pinheaders. With the exception of pin 8 of K2 you should not measure any resistance ('infinite' resistance, or open circuit). If you do measure a resistance somewhere, you probably have an IV-22 tube with a manufacturing fault.

Main board

Mount the parts in order of height from low to high and put IC2 (the 7555) in its socket. Mount Zener diode D5 several millimeters above the board, since this part can become very hot during fault conditions. Don't insert IC3 and the ESP32-DevKitC module into their sockets yet, and connect pin 8 of the ESP32-DevKitC connector (IO33/STBY) temporarily to ground using a piece of wire. Turn the power supply on and measure the voltage between pins 1 and 20 of the socket for IC3. This voltage should be adjusted to 50 V using P1. Measure the voltage between pins 2 and 3 of IC1 (PTH08080W module) and set this voltage to 1.1 V with P2. You should measure 5 V between pins 40 and 20 of IC3.

RGB background lighting

Mount all the 330 Ω resistors, followed by the RGB LEDs. Push the LEDs through the board up to the kink in the leads and solder a single lead on each one. Check that the LEDs are properly positioned and then solder the remaining leads. Don't let them get too hot and ensure they are correctly orientated. Solder two 4-ways

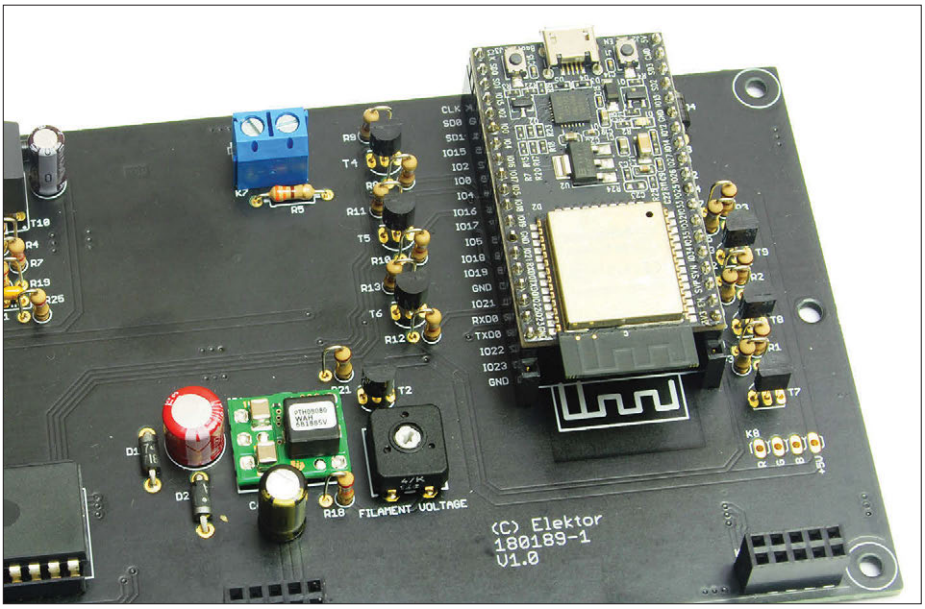


Figure 11. This shows clearly how the ESP32 module should be mounted.

pinheaders together, using four lengths of wire, and solder one of the pinheaders to the board for the background lighting.

Putting it all together

Connect the display board to the main board and switch on the power. Darken the room and check that the filaments of the IV-22 tubes are dimly lit. If a filament is not lit in one of the tubes, or if it's significantly dimmer than the others, you could try (carefully!) wriggling the particular tube. Sometimes there could be a bad contact, which can be a real problem at the very low voltage of 1.1 V. Turn the power off, remove the connection between pin 8 of the ESP32-DevKitC module and ground, and insert IC3 (the HV518P-G) in its socket. Also insert the ESP32-DevKitC module in the main board. Make sure that you have inserted the module correctly into the 20-way pinheader socket strips: the module has two rows of 19 pins! **Figure 11** shows how it should be done.

Turn the power back on and use a USB cable to connect the ESP32-DevKitC module to a computer that has the Arduino-IDE installed. Compile the sketch and upload it to the ESP32-DevKitC. You can then send the previously prepared configuration file via the serial port to the clock, using a program such as PuTTY. If all went well, you should see something on the tubes. Turn the power off again and mount the board for the RGB background lighting behind the tubes of the display board, using standoffs. Connect the connecting wires to the main board. If you want, you could also connect a wake-up button and a temperature sensor. When changing one or a few settings you could use a smartphone with a (serial) terminal app instead of a computer. It can of course also be done wirelessly via the Wi-Fi. We have successfully used the following Android apps for this: *Serial USB Terminal* and *TCP Telnet Terminal*. **Figure 12** shows the fully assembled clock in all its glory.

Your clock is now ready for use. A ready-made enclosure made with laser-cut PMMA is available from our store.

We would like to point out that we intentionally haven't included support for summer/winter time. The reason for this is that more and more people like to see an end to the twice-yearly changes, and that the rules for the summer/winter time are not the same round the world. ⏮

(170573)

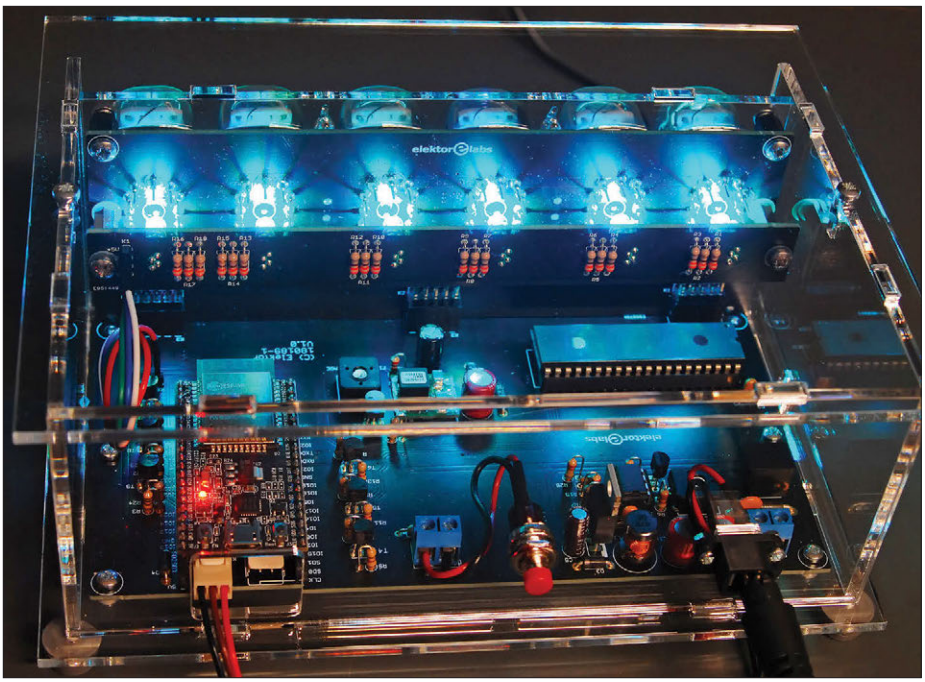


Figure 12. The VFD clock in all its glory (rear view).

Web Links

[1] www.elektormagazine.com/160618

[2] www.elektormagazine.com/150189

[3] https://github.com/espressif/arduino-esp32

[4] www.pool.ntp.org/

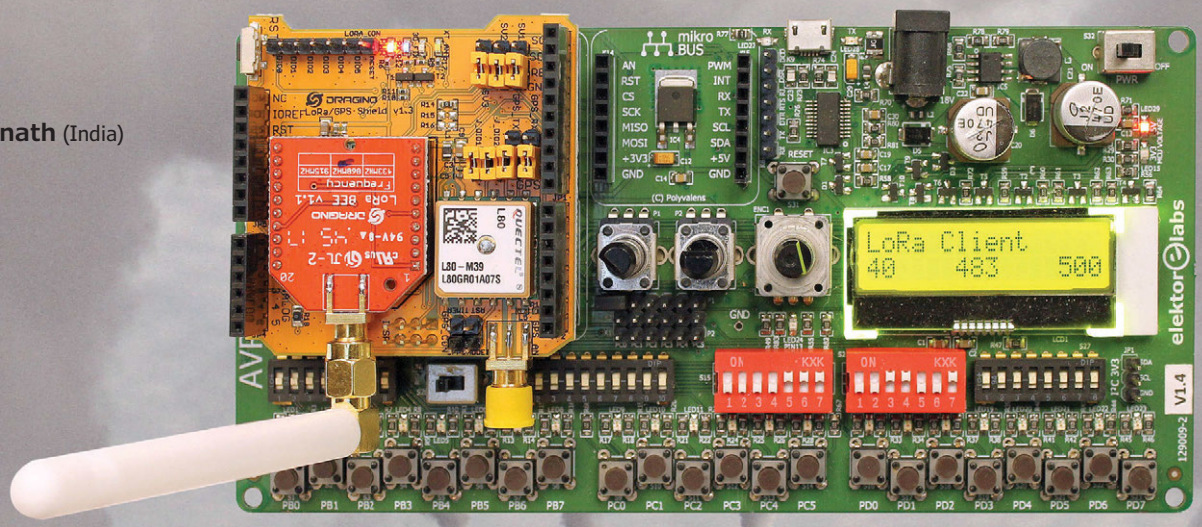
[5] www.elektormagazine.com/170573

HOMELAB PROJECT

LoRa Telemetry Projects

Using wireless peer-to-peer connections to fight industrial pollution

By **Bera Somnath** (India)



Features

- LoRa peer-to-peer communication
- Low-cost Arduino-based
- Easy to understand
- Helps to solve real-life problems

One of the difficulties of monitoring parameters in remote locations is getting the data over to a control room over long distances and 'overcoming' geographical and man-made obstacles. This article shows how LoRa together with Arduino can help to work around such problems.

While several communication technologies like Sigfox, LoRa, LTE-M and NarrowBand-IoT (NB-IoT) are competing to impose themselves as the standard for the Internet of Things (IoT) the market is being flooded with all sorts of low-cost radio modules. The good thing with these modules and technologies is that they are designed for (relatively) long distance operation, something that is difficult to achieve with Wi-Fi or Bluetooth.

The idea behind IoT is that objects ('edge nodes') send data to a (cloud) service, from where it is distributed to the data consumers. Although this is a nice concept, for many applications it is overcomplicated; one or more peer-to-peer (P2P) connections are sufficient in many situations. Of the above technologies, currently only LoRa supports P2P (with LTE-Direct hot on its heels). Also, LoRa is subscription-free, which explains why it is so popular among electronics amateurs. In this Homelab-category article I describe some experiments to help you to get started with LoRa peer-to-peer connections. Real-world

application examples will illustrate the experiments.

Chlorine leakage monitoring

After building a new chemical plant [2] in the new control room it was required to monitor the chlorine leakage along with the chlorine cylinder temperature of the chlorination plant located some 3.5 kilometres away. With a few 'obstacles' like a new 800-kV DC switch yard, an existing 400-kV switch yard, and a 30-meter-wide circulating water channel, in between the plant and the control room it was felt that a wireless data connection would be the cheapest and easiest solution. Unfortunately, the high-voltage switch yards produce so much RF interference and noise that most commercially available RF data links simply do not work. LoRa, on the other hand, does work in such hostile environments thanks to its use of spread-spectrum techniques (see **inset**). Two low-cost LoRa modules together with a microcontroller can easily create a reliable data link for the above-mentioned application.

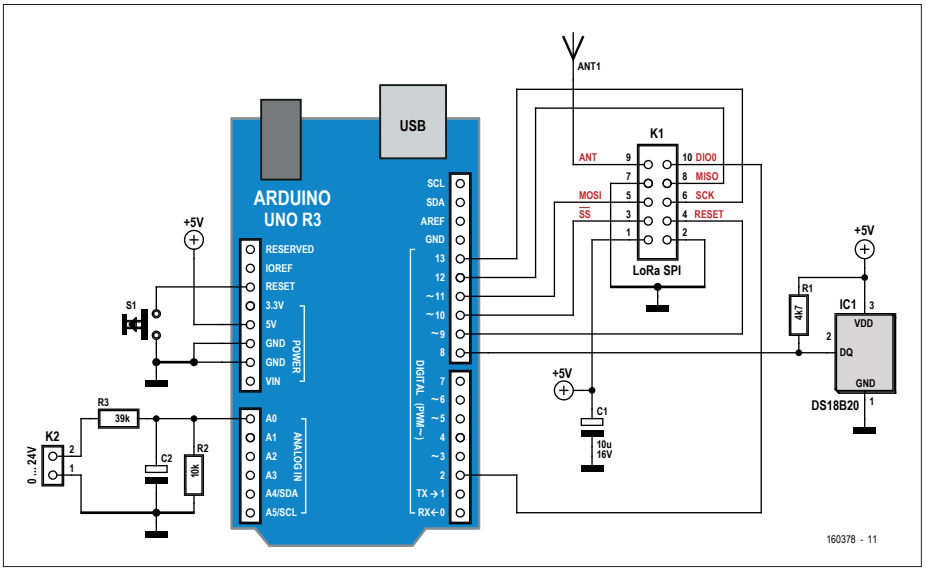


Figure 1. The LoRa transmitter (server) placed at the chlorine cylinder. The chlorine leakage signal is connected to K2, the cylinder temperature is measured by IC1 housed in a stainless steel enclosure.

Signals & sensors

In small doses chlorine can be used to kill microorganisms. However, for applications like water purification, high doses are required which can be lethal, hence the importance of keeping an eye on any chlorine leaking. In the event of a chlorine leak the chlorine cylinder gets emptied fast and the cylinder temperature will drop quickly. The measurement of chlorine cylinder temperature is therefore a vital parameter for identifying leaks.

The chlorine cylinder temperature sensor, which happened to be out of order, was replaced by a DS18B20 device in a waterproof stainless steel enclosure, capable of measuring temperature down to –55 °C. Since the atmosphere is chlorinated, corrosion is a big issue for everything metallic there.

Chlorine leakage is also measured as a 0–24 V analogue signal. It is converted to a 0–5 V signal by a resistive voltage divider to make it palatable to a microcontroller.

Cobble up an Arduino system

For the microcontroller an Arduino Uno was chosen (or a compatible board, see the **AVR Playground inset**). For this platform libraries for both the temperature sensor and the radio modules are widely available. **Figures 1 and 2** show the schematics of our prototypes. A relay to switch a loud alarm is connected to the MCU through a small transistor.

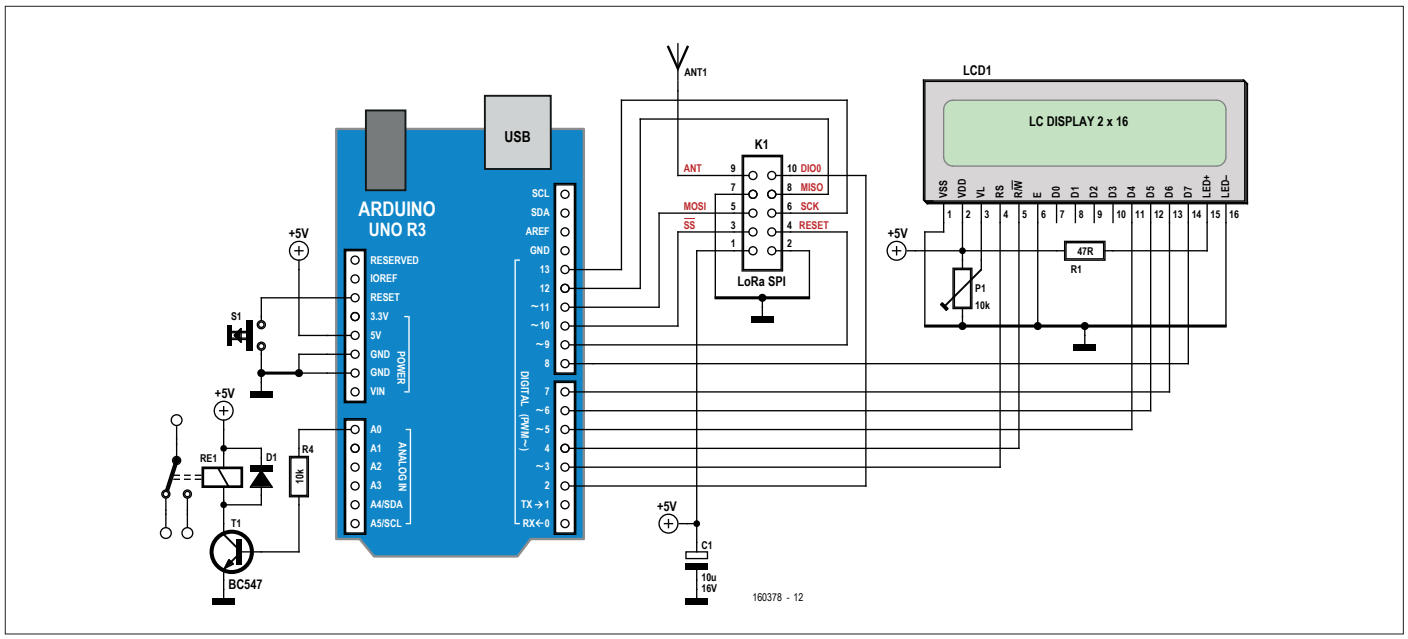


Figure 2. The LoRa receiver (client) is located at the control room, some 3.5 km away from the server. It displays the captured data and can activate an alarm by means of a relay.

The world's largest coal-fired power station is the Taichung Thermal Power Station in Taichung, Taiwan. Together with its gas-fired and wind generation units, the total installed capacity of the plant is 5,824 MW. (source Wikipedia)

For the LoRa link two 433-MHz modules were selected, based on the RFM98 from Hope Microelectronics (also known as Hoperf, which should be read as *Hope-RF*). Note that these modules have a 2-mm pin pitch, which makes mounting them on piece of perfboard a bit difficult.

You may want to look for modules with a regular 0.1" pitch (see **inset**). These transceivers are connected to the microcontroller's SPI bus.

Each LoRa module was equipped with a 6-dBi (decibel-over-isotrope) antenna with an SMA connector. All parts used should be easy to find online.

With these parts a transmitter and a receiver were constructed. The transmitter was installed on the corner of the chlorine plant roof on a 4-meter (13-ft.) high mast. The small 433-MHz antenna was placed strategically on the new chemical plant unit's rooftop, in line of sight of its brethren unit placed 3.5 km away across the high-voltage switch yard and the water channel. During a test run it was found that all data arrived at a rate of once every two seconds.

Open-source libraries

There are lots of Lora radio libraries available on the Internet. We opted for the impressive RadioHead package that provides neat and clean commands. Just a few lines of code are needed to get the radios up and running. The library offers many configuration possibilities to adjust the bandwidth, the spread factor and transmit power.

The chlorine plant data is transmitted as comma-separated values (CSV format). The receiver deciphers the data and prints the values on the LCD. It also generates an alarm when the values become dangerous.

The sender and receiver — a.k.a server and client — have different IDs, making it easy to set up connections between two devices without bothering other LoRa systems.

A step further

In our second case, a coal-based thermal power plant [3], the problem is slightly different, but can be solved in a similar way. Because coal-based thermal power plants (**Figure 3**) are huge polluters the Indian government, by means of the National Green Tribunal (NGT), insists on control of pollutants and requires continuous measurement and monitoring directly at the source of pollution.

One problem area proved the water level of the ash dyke overflow lagoon. Ash produced by burning coal is mixed with slurry and then transported to the ash dyke where it is stored in an underwater ash pond (**Figure 4**). The surplus water is spilled over into the adjacent overflow lagoon and from there it flows into the ash water recirculation pump house. Water gets pumped back to the power house again. Thus, besides saving water, this also helps reducing the discharge of polluted water into the natural drainage system.

During normal operation all water spilling

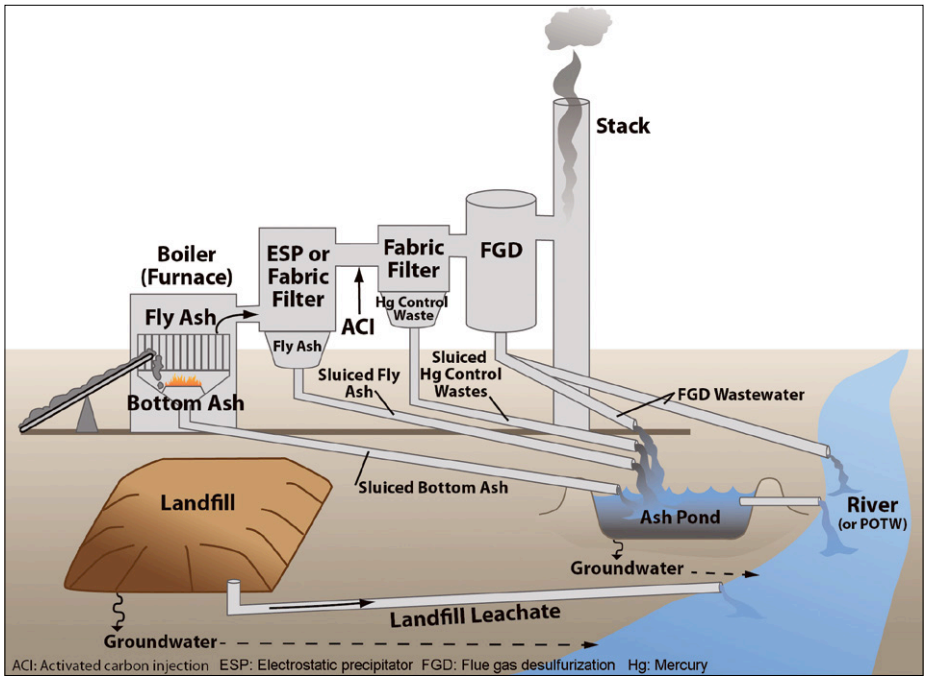


Figure 4. Waste streams in a steam electric power plant. In this article we are interested in measuring the water level of the ash pond. (source: www.epa.gov)

into the overflow lagoon gets pumped back to the power house. However, when due to heavy rain or pumping problems the water level increases there is the possibility that the lagoon itself overflows, letting polluted water enter the natural drainage system. Besides keeping the ash water recirculation system healthy, the NGT also wants to monitor the water level of the ash slurry overflow lagoon on a 24/7 basis.

The lagoon may be situated up to 10 to 15 kilometres away from the power house where the ash slurry control room is located. The ash water recirculation pump house is somewhere near the ash dyke but too far away from the overflow lagoon to monitor its water level as it gets its water either through a channel or through a long penstock. Besides the distance, the pump house is not permanently occupied; therefore, for 24/7 monitoring, the data is to be conveyed to the main ash slurry control room.

Transporting the water level over such a great distance is a challenge, but armed with LoRa technology we went forward to an achievable solution.

What's needed is a LoRa repeater

With 2 to 13 dBm radiated power levels and line-of-sight (LOS) LoRa can comfortably transfer data from the ash dyke to the boiler head, which is at a height of 65 meters (200 ft.). From there everything is LOS. However, the ash water control room is a low, one-story building located 1.2 km away from the foot of the boiler. Therefore, as a first step the data from the overflow lagoon (water level, position, temperature and in the future also wind direction), is made available at the 65-meter-high boiler (**Figure 5**). From there the data is transmitted over a second LoRa link to its final destination: the roof of the ash water control room. So, what is needed on the boiler unit is nothing but a LoRa repeater.

Such a repeater is easily constructed using an Arduino and two LoRa modules (**Figure 6**). For one side of the link we use the same modules as above, with SPI interface; for the other link we use a similar module but with an asynchronous serial (UART) connection. By using two sets of different server and client IDs the two links can operate simultaneously.

The third set, the one at the ash water control room, also uses the LoRa module with the UART port (**Figure 7**).

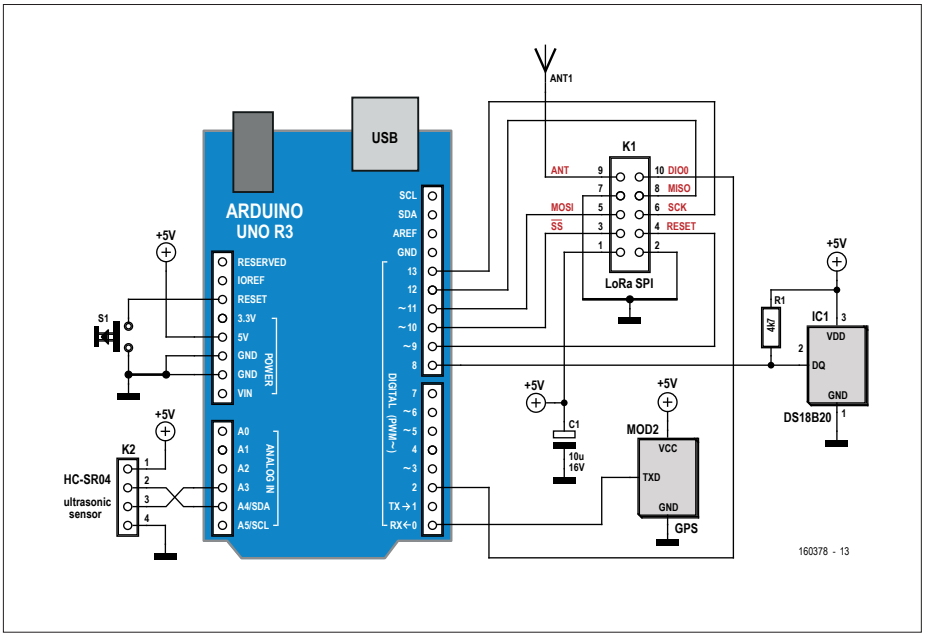
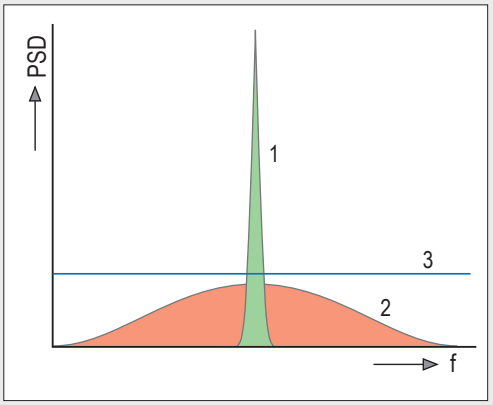


Figure 5. This LoRa server sends the water level of the overflow lagoon measured with a popular ultrasonic transducer to the repeater at the roof of the boiler. It also transmits the position calculated by the GPS module and air temperature. A wind direction sensor remains to be added.

Spread Spectrum

Since the beginning of World War 2 spread-spectrum techniques have been deployed, often to protect communications from being deciphered. It used to be an expensive technology but as semiconductor technology matured, the cost decreased and over the last few years companies like Dorji, Hope, and Semtech have released small radio modules which can produce spread-spectrum signals at very low cost.



Spread-spectrum in a nutshell.
1: narrowband signal; 2: spread-spectrum wideband signal; 3: noise floor.

Spread-spectrum is a method by which a signal with a particular bandwidth is deliberately spread in the frequency domain, resulting in a signal with a greater bandwidth. A spread-spectrum signal does not have a clearly distinguishable peak in the spectrum, which makes the signal more difficult to distinguish from noise and therefore more difficult to jam or intercept. The most common spreading techniques in use are Frequency Hopping (FH), which makes the narrowband signal jump around in the spectrum to make it occupy a larger bandwidth, and Direct Sequence (DS) which introduces rapid phase transitions to the data to make it larger in bandwidth. The receiver "un-spreads" the wide-bandwidth signal to obtain the original narrowband signal.

Because the signal can be spread to a wide bandwidth below the noise level it is difficult to detect. If somebody tries to eavesdrop on it, only some increase in the noise level may be noticed, nothing more. Even in the case that the signal is known to be present, without knowledge of the spreading or hopping sequence it is almost impossible to decode or jam it.

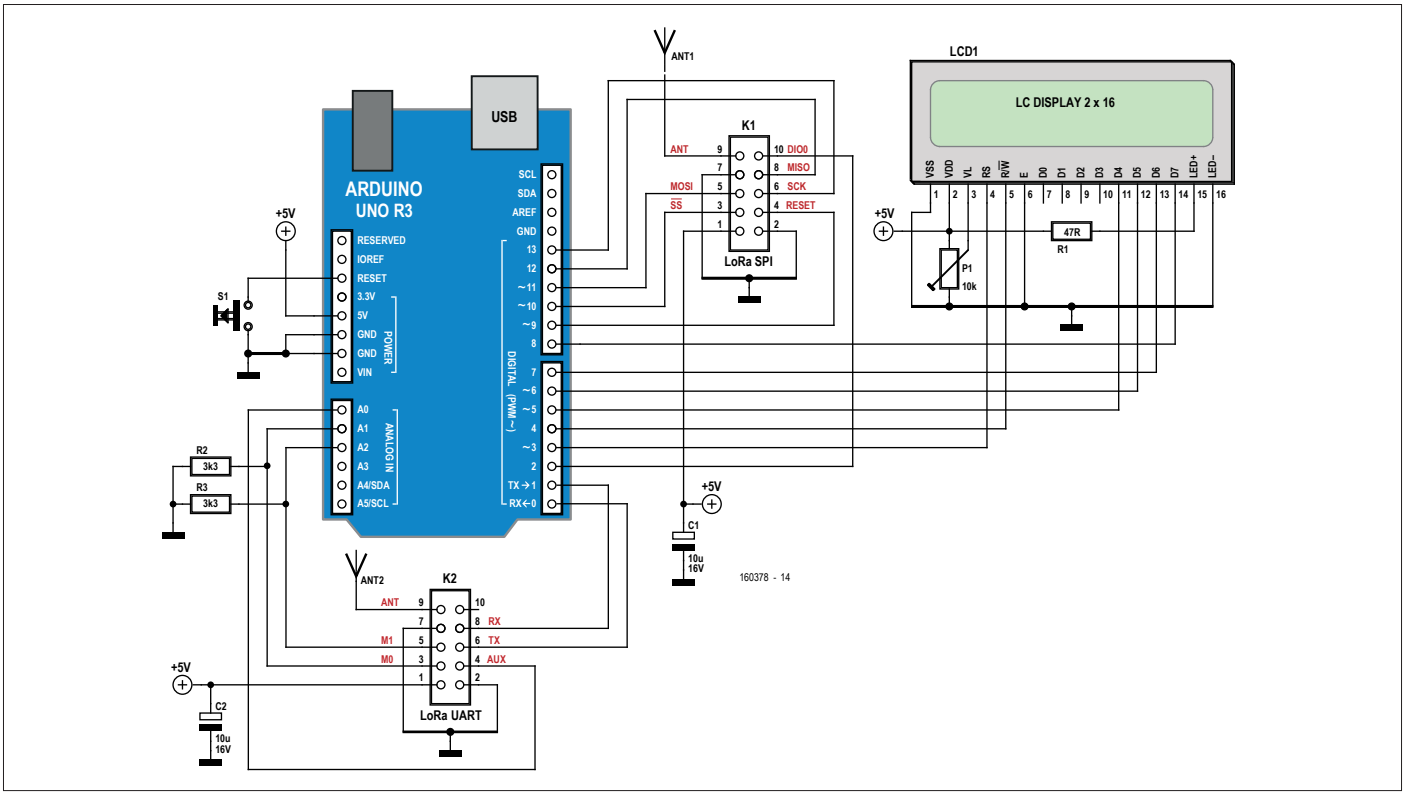


Figure 6. The repeater functions as a bridge between the SPI LoRa module (connected to K1) and the UART LoRa module (connected to K2). A display is available to see if everything is working as intended.

LoRa even works when it rains

Field tests have shown the correct operation of the system. My supervisor Shukla gradually moved with the first (uplink) LoRa set towards the farthest end of the ash dyke (14.2 km away according to my GPS calculator) while I stood at the boiler unit with the repeater module. My colleague Lalit went on the roof of the ash water control room with the third (downlink) LoRa set.

The signal started pouring in the moment Shukla switched on the uplink unit. Standing on the edge of the overflow lagoon, he pointed the ultrasonic level gauge towards the milky white ash water. Below, on the roof of the ash slurry pump house, I could see Lalit waving his

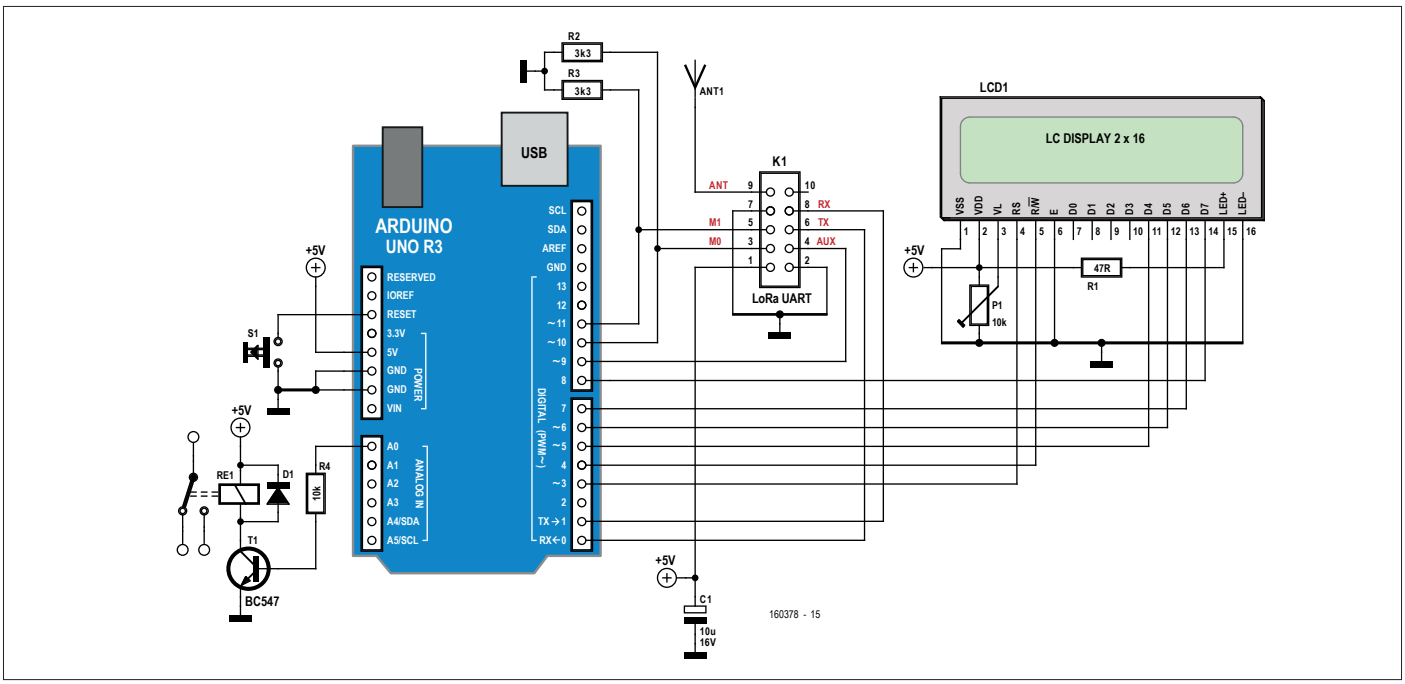


Figure 7. The LoRa client in the ash water control room is almost identical to chlorine leakage client (Figure 2) except that the LoRa module (K1) now is a UART type, the same as used by the repeater (Figure 6).

hands to indicate that he was receiving the data. The ash slurry pump house is about 1.5 km diagonally from my position. Putting the boiler unit in between Lalit and me slightly decreased the data rate from once per 2-3 seconds to once per 3-4 seconds.
When a heavy rain storm started at the dyke, Shukla scoured into his vehicle and had to shut the windows. The data stream stopped for a while but then started again at a slower pace of about once every 7 seconds. When the rain storm hit our position Lalit went inside the control room and, obviously, I was no longer at LOS with him. Still data continued to arrive but at a rate of once per 7 to 11 seconds.
Mission accomplished.

Conclusion

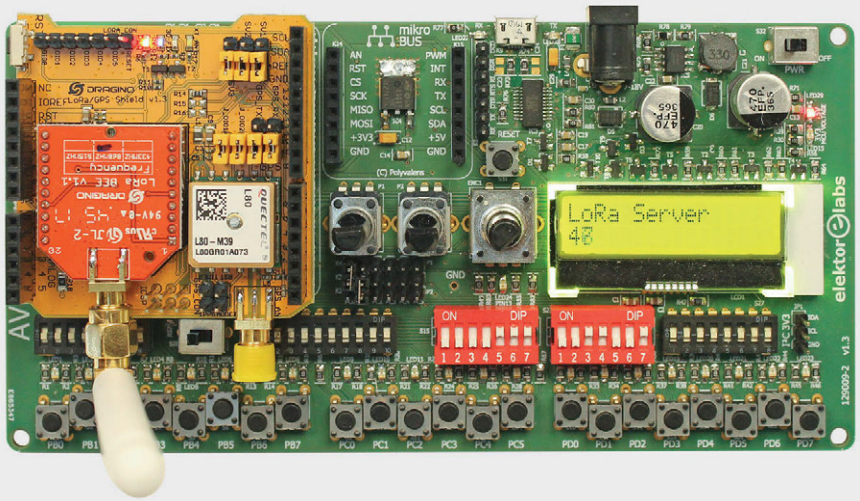
Using simple and widely available low-cost means it is possible to contribute to solving real-world industrial problems. Environmental hazards can be monitored remotely with the help of cheap LoRa peer-to-peer radio links and a bit of Arduino ingenuity. The projects presented can be used for many other generic remote telemetry operations.
Let's sound off with a word about the RFM95/96/97/98(W) transceivers from Hope. They can all be used for these projects. The main difference is in frequency band (868/915 MHz for the RFM95 and RFM97; 433/470 MHz for the RFM96 and RFM98) and spreading factor. In the RadioHead library (not Thom Yorke's, *Ed.*) they are all handled by the RH_RF95 driver. ⏮

(160378)

Web Links

[1] www.elektormagazine.com/160378
[2] www.elektormagazine.com/labs/remote-telemetry-on-lora-433-mhz
[3] www.elektormagazine.com/labs/very-long-range-remote-telemetry-using-lora-repeater
[4] www.elektormagazine.com/labs/zero-discharge-monitoring

Simulation with the AVR Playground



For simulation and development purposes we set up a peer-to-peer connection with two AVR Playground boards to which we added a Dragino Lora/GPS shield (with the jumpers in the default positions). One system simulates the plant (server), the other the control room (client, see opening photo).
The plant system simulates two analogue signals with its two potentiometers connected to A0 (PC0, K4-K3 pins 1) and A1 (PC1 K2-K3 pins 2). The signals are sampled and the values are sent as ASCII strings to the control room system. Here the values are displayed on the LCD. Both systems run a counter of which the value is displayed by the other. Doing so allows checking that the connection is still alive. In the case of a communication failure an alarm can be given.
The Dragino shield houses a Dragino LoRa BEE module which in turn is nothing more than a breakout board for a Hope-RF96-based LoRa module. The BEE module has a pitch of 0.1" and a handy SMA antenna connector. A special Dragino version of the RadioHead library is available to support these modules.
A GPS is also mounted on the shield, intended for applications that require positioning information. We did not use it here and to avoid Arduino program upload interference problems it is highly recommended to put the AVR Playground switch S28 in the position 'PRG' to disconnect the GPS's serial port.
Our test sketches and libraries can be downloaded from the Elektor Magazine website [1].

Table 1. AVR Playground client & server DIP switch settings.

Switch	Off	Centre	On
S24		1-8	
S33		1-10	
S15	1, 2, 3, 4		5, 6, 7
S25	1, 2, 5, 6		3, 4, 7
S27		1-8	

HomeLab Helicopter

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

Prepping for power surge attacks

The other day while on the train I stepped on a USB stick lying on the floor. All fellow travellers sitting nearby denied ownership of the stick so I put it in my pocket with the intention of examining its contents later. Of course, I know you have to be careful with USB sticks of unknown origin because they might contain malicious software, and you should even be careful with thumb drives from people you trust, which is why I did not plug it into my computer right away. Also I had been warned about the USB Killer, making me extra cautious. Do you know the USB Killer?

Now in its third incarnation, the USB Killer or "USB Kill" looks like any other USB pen drive. However, as soon as you stick it into a USB port, it places a high voltage (200 V) onto the port's data lines, destroying the port in most cases. The website supporting the product explains: "*The USB Kill 3.0 is a testing device created to test USB ports against power surge attacks. The USB Kill 3.0 tests your device's resistance against this attack.*"

Source: usbkill.com

It takes a vicious mind to come up with such a device, and an even more vicious mind to make a product out of it, but that is not where it stops. Special adapters are available to use the USB Killer with Lightning ports (used by recent iPhones), USB-C and micro USB. To top it off, they even sell a USB Killer Shield (CE approved!) to protect your USB port against the USB Killer. It is like world politics on a small scale: invent a danger and propose a solution. Oh, by the way, USBKill.com strongly condemns malicious use of its products. At least that is what it says on the website.

Those looking for ways of testing Ethernet and other ports in a potentially destructive way might want to consult www.fiftythree.org/etherkiller/

The future of electronics?
At 68 °C vanadium dioxide (VO_2) changes from insulator to conductor in less than a nanosecond...

Must-have Homelab Tool

Many SMT devices have markings that are either utterly cryptic or so small that you can't read them. As a result, it is often difficult to tell resistors, capacitors and inductors apart from their appearance only. This is where tools like the Mastech MS8911 can lend a hand. Looking like an overgrown tweezer with gold-plated contacts and a display, it can measure resistance, capacitance and inductance. Not only does it show the component type and its value, but it also gives the dissipation factor (*D*) for capacitors or the quality factor (*Q*) for inductors. Depending on the component value the tester automatically selects a series or parallel measurement method. An SMD tweezer is a great tool for anyone working regularly with SMD passives — get one now.
www.elektor.com/mastech-ms8911-smart-smd-tester-lcr-meter

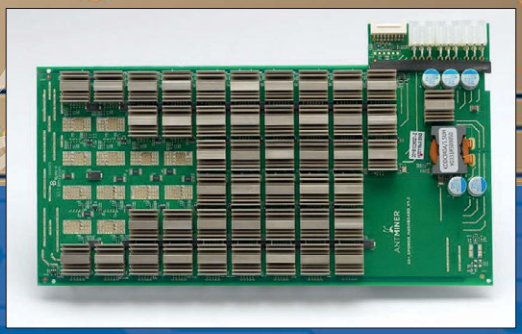
The future of electronics?
Renesas, Fujitsu and Panasonic may merge their semiconductor activities...

Real chips dig up virtual money

Special mining computers designed for executing the bitcoin validation algorithms can be bought; data centres specialized in keeping these computers up & running have come into existence, making bitcoin mines real places you can visit. The world's largest bitcoin mine is located in Ordos, Inner Mongolia (China), a former coal-mining town. It is operated by Bitmain Technologies, and consists of eight buildings housing some 25 K mining machines (of which 4 K are dedicated to litecoin mining, a bitcoin variant. These figures are from 2017). The estimated capacity of the mine represents about 3.5% of the global Bitcoin mining network. Not only does Bitmain run the mine, they also design the mining ASICs, build the special computers using them — the Antminer series — and sell these online. Bitmain is a very profitable company indeed.
The mine gets its energy from a coal-fired thermal power plant nearby. It is estimated to consume 40 MWh of energy, about 20% of which is required for cooling the computers. This is one of the reasons why many people think that the bitcoin will disappear in the near future: it simply is too power-hungry. According to the Digiconomist, a website dedicated to bitcoin and other cryptocurrencies, at the beginning of 2018 bitcoin mining consumed over 48 TWh of energy per year, which is equivalent to the energy consumption of Singapore or the power requirements of 4.5 million US households. Although many smaller mines are powered from green energy sources, other mines, like the one run by Bitmain, run on cheap electricity produced by burning coal and produce tons of CO_2 in return.
It is the sustainability of bitcoin mining that poses problems. In order to gain global acceptance and remain durably successful Cryptocurrencies must, among other things, strive to solve this sustainability issue. Not requiring mining is an important step in the right direction. The iota cryptocurrency targeted at the low-power Internet of Things (IoT) is an example of such a technology. Besides that it eliminates mining, iota is also free of the transaction fees used by bitcoin to motivate people to start mining in the first place. Rumour has it that the iota team is working on a dedicated real chip to support their virtual money.

"Engineers working for the Russian Research Institute of Experimental Physics at the Federal Nuclear Centre in Sarov have been arrested for using the power of a supercomputer to mine cryptocurrencies. […] In October 2017, two officials of the Crimea Council of Ministers were fired for mining cryptocurrencies using government computer facilities." [source: bitcoin.com]

Invented in 2009 by an unknown entity called 'Satoshi Nakamoto' and then released to the open source community, the bitcoin (BTC, symbol: ₿ has since become a huge success. It is a decentralized digital currency, a so-called cryptocurrency that allows transactions between users directly; no banks or other intermediaries are involved. The bitcoin is totally virtual money. Bitcoin transaction validation is done through a process called mining. This involves complex calculations and a lot of computer power, which is why miners are paid (in bitcoins, of course) for their work. Since mining brings in money and does not require any physical effort — all that is needed is a (very) powerful computer connected to the bitcoin network — many people have become miners.



The capacity of mining machines is measured in hashes per second (H/s) — hash rate — where a hash is the elementary calculation step of a bitcoin transaction validation process. The hash calculations are executed by specialized hardware like this hash board fitted with multiple hash chips. The Antminer V9 bitcoin miner specified for 4 TH/s contains three hash boards with a total of 135 hash chips and consumes just over 1 kilowatts.

The Antminer R4 is commercialized as a bitcoin miner "meticulously designed" for your home. *"With a height of 100 mm, it is slim enough to rest comfortably in your book rack or computer table."* Although it has an impressive hash rate of 8.6 TH/s, it usually produces less than 52 dB of ventilation noise. At best it's the only heater that pays for itself.

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

HOMELAB PROJECT

The Connected Greenhouse

IoT demonstration project using MQTT and Node-RED

By **Walter Trojan** (Germany)

Imagine that you have decided to build a complex IoT (Internet of Things) project, such as a system for monitoring and controlling the greenhouses in a nursery. Among other things, this might involve stabilizing the interior temperature and lighting at preset target levels and monitoring the humidity of the interior atmosphere and the moisture level in the soil in which the plants are growing. If these values go outside a specified range an alarm must be raised, for example by sending an e-mail to the nursery manager. Not a trivial undertaking, you might think; but these days the availability of cheap hardware and free software makes it easy to tackle an intricate IoT project of this type.

A professional approaching a project like this would start with a functional specification document, including a detailed definition of the solution and expected costs and delivery schedules. Unfortunately it is not possible at the outset to anticipate all possible difficulties and risks that might be encountered, and so it is worthwhile to carry out an initial feasibility study. This article describes just such a feasibility study, employing a model greenhouse populated with a real living plant, and looks at the hardware and software components that could be used. For reasons of space we cannot go into the full constructional details, but we can give some suggestions that might come in handy for your own projects. In the project we use familiar hardware such as the Raspberry Pi and the ESP8266 [1], along with cutting-edge software technologies such as MQTT and Node-RED. MQTT is a bandwidth-efficient IoT communications protocol, which typically runs on top of TCP/IP, and Node-RED [2] is a graphical programming system that is ideal for building prototypes and complex applications.

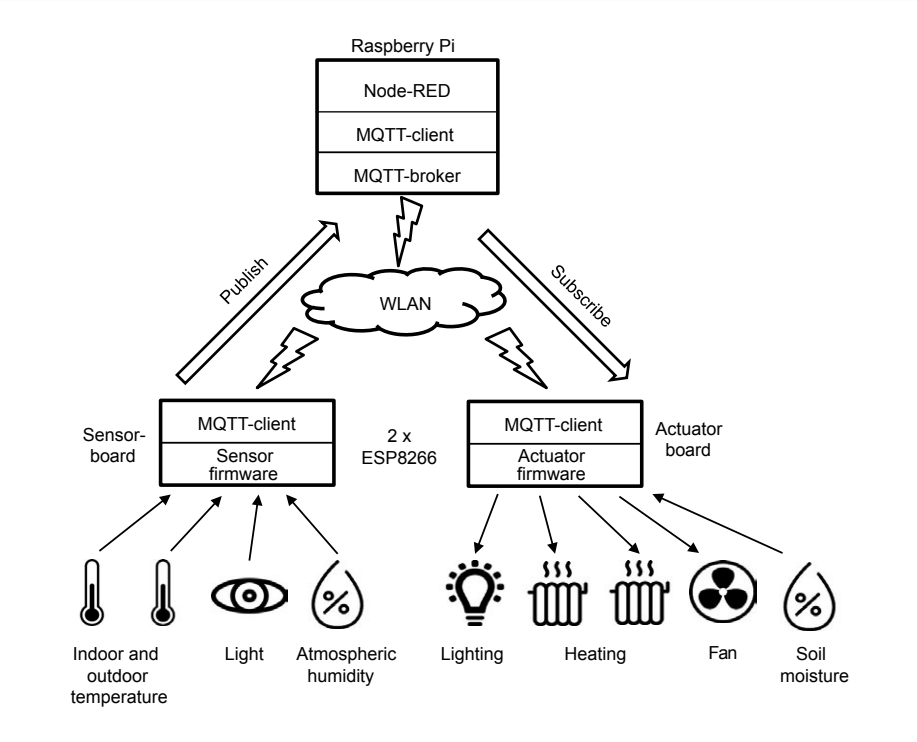


Figure 1. Configuration of the model greenhouse.

Project overview

It is of course well within the capability of a Raspberry Pi or an ESP8266 to monitor and control a model greenhouse. However, that would not capture the potential scale of the real project, which involves many greenhouses. For that reason we share the work among several devices, connected to one another using MQTT running over a WLAN.

Figure 1 shows the overall structure of the project. The active components (the Raspberry Pi and the ESP8266-based sensor and actuator boards) are connected together using a WLAN. These components exchange readings and commands using the MQTT protocol. The master of ceremonies is the Raspberry Pi, on which the Mosquitto MQTT broker software and the Node-RED application are installed. The control and monitoring process involves the following steps, executed cyclically.

- The sensor board measures the desired environmental parameters (temperature, light and humidity) and publishes the readings. This involves sending a message to the MQTT broker.
- The broker checks whether the message is of interest to other devices in the network and if necessary forwards it.
- The Node-RED application, which is subscribed to the message, processes it and, if required, publishes commands in response.
- The actuator board, which is subscribed to specific command messages from the Node-RED application, receives any commands that relate to it and executes them.

Even from this high-level view of the system it is possible to see how the MQTT network operates. At the center there is always a broker whose job is to mediate the transfer of messages over the network. The network can connect together a practically unlimited number of devices. Some network participants publish messages without knowing which participants will read it; others subscribe to these messages and can process their contents, without knowing from where they originated. The details of this process are explained in more detail below.

Our model greenhouse

The front view of the model greenhouse is shown in our lead photograph. It is mostly made of wood and acrylic. Its dimensions are such that a medium-sized houseplant will fit inside it: the base plate is a wood offcut 0.8 cm thick and measuring 50 cm by 30 cm, and to this are fixed 5 mm acrylic sheets 30 cm high to form the walls. The right-angle joints are made using lengths of 1.5 cm square cross-section wooden strips to which the various sheets are screwed. The detachable roof is also made from sheet acrylic, joined at the ridge using a triangular cross-section wooden strip. A handle fixed to the middle of the ridge makes the model greenhouse portable. The roof is firmly attached to the walls using mending plates and four M6 bolts. Two halogen heater lamps (12 V, 10 W) are mounted in diagonally-opposite corners of the enclosure, and the two processor boards are mounted in the middle of the short edges. A strip of white LEDs in the roof ridge provides bright illumination for the plant, while a fan on the side of the enclosure provides ventilation. The power supply is also mounted on the side of the enclosure. A two-pin sensor stuck into the soil measures its moisture level. In the photograph the heating lamps and the LEDs are at full intensity, which should be keeping the plant nice and cosy.

What do we need?

Having outlined the system we can now look at the particular hardware and software components that we will need in more detail.

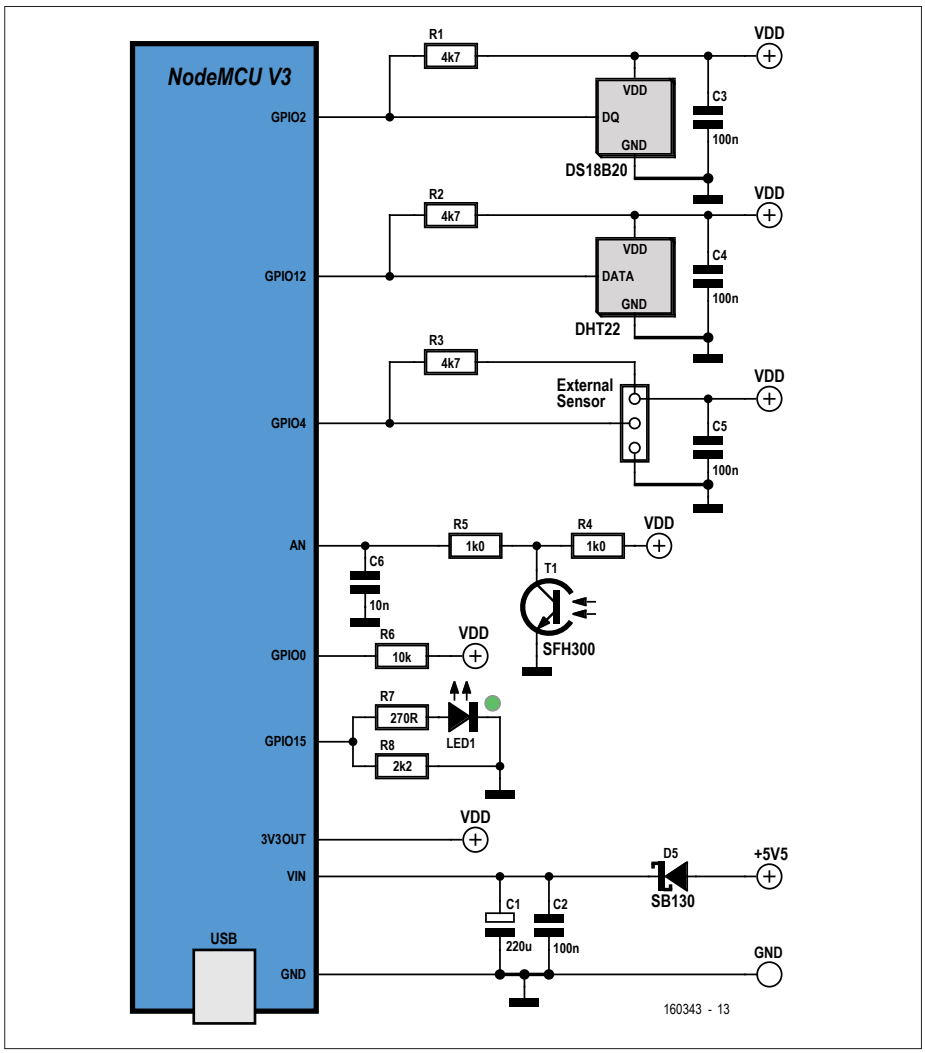


Figure 2. Circuit of the sensor board.

Raspberry Pi

We will be using a Raspberry Pi 3 with a four-core processor and 1 GB of main memory, which gives us plenty of processing power to spare. The most up-to-date Raspbian operating system, Raspbian Stretch, is stored on an 8-GB SD card. The system comes with a WLAN chip fitted and thus is easily connected to the wireless network. Fortunately Node-RED is already pre-installed in Raspbian so we only need to install the Mosquitto broker and the Node-RED application. When configuring the system, and for debugging, it is helpful to connect a USB mouse and keyboard and an HDMI monitor, but these are not needed for normal day-to-day operation: instead it is possible to log in to the Raspberry Pi using the 'ssh' command from an external PC to make any adjustments that might be necessary.

Sensor board

This board (see **Figure 2**) consists of an ESP8266 on a NodeMCU V3 board and handles the interface to the sensors to measure environmental parameters. The indoor temperature sensor is a DS18B20, the humidity sensor is a DHT22 and a phototransistor is used to measure the ambient light level. An external input allows a further DS18B20 to be added to measure outdoor temperature. If the outdoor temperature falls below a specified threshold, for example 4 °C, the operator can be sent an e-mail warning of the risk of frost. The temperature sensor built into the DHT22 is not used, although it would be possible to use it to check the measurement from the DS18B20.

The structure of the firmware for the ESP8266 is straightforward. The sensors are polled every 10 s within a loop, and the readings are published using MQTT. The sensors are read using their drivers.

The program was developed in C using Eclipse and in C++ using the Arduino environment: you can download it from the Elektor website at [3] and experiment further with it.

Actuator board

The ESP8266 NodeMCU board has four high-current outputs A1 to A4 to control the lighting, heating and ventilation of the greenhouse: see **Figure 3**. Each of these outputs is pulse-width modulated with a duty cycle from 0 % to 100 % and can run at up to 12 V. Lighting is provided by several LED strips populated with bright white LEDs. It is known that plants prefer blue and red light; since white light contains all colours, the plants should thrive. Heating is provided by two 12 V halogen lamps each with a power rating of 10 W, switched by separate outputs from the board. If the control algorithm demands only a small amount of heating power, the lamps are dimmed by adjusting the PWM signals. At elevated levels of temperature or humidity the 12 V fan is switched on to provide cooling and air circulation.

Since the analogue input to the NodeMCU on the sensor board is used by the phototransistor, the soil moisture level sensor must be connected to the otherwise unused analogue input on the actuator board. The sensor itself consists of two electrodes that are pushed into the soil; the resistance between these electrodes is measured and converted to an analogue voltage. A subsequent operational amplifier on a small adapter board converts the voltage range to 0 V to 3.3 V. The firmware determines the moisture level from this voltage and sends the result to the central controller using MQTT. In turn the central controller can send a warning e-mail to the operator if the soil is too dry.

The firmware for the actuator board was also developed in C using Eclipse and in C++ using the Arduino environment, and again the code can be downloaded from the Elektor website. The code receives commands from the Node-RED application over MQTT and generates the required PWM signals on the high-current outputs. It also periodically communicates the soil moisture level readings to the application.

Completing the circuit

The power supply is straightforward: 18 V DC from a laptop adapter is

taken to two DC-to-DC converters (see **Figure 4**). One of these produces 5.5 V to power the NodeMCU boards: the extra 0.5 V compensates for the forward voltage drop across the reverse polarity protection diodes on the actuator and sensor boards. The second DC-to-DC converter supplies the power outputs at 12 V. The input to the power supply circuitry is also protected against a voltage of reverse polarity being applied by diode D1. Since the heaters alone can dissipate as much as 20 W at full power, a reasonably capable laptop adapter is required: we could recommend one specified to deliver at least 3 A.

A quick look at MQTT

In contrast to the request-response architecture of the widely-known HTTP protocol, MQTT is based on a publish-subscribe architecture, based around a central broker. Devices which have a message to send simply 'publish' their message to the broker. The broker then forwards these messages to any other devices that require them, where they are processed. The nodes that wish to receive messages must 'subscribe' to them. Communication can happen in both directions: an individual device on the network can simultaneously be a publisher and a subscriber. A node might be a simple microcontroller, a PC or a Linux server. The only requirements are the presence of a TCP/IP stack and an implementation of the MQTT protocol.
The addressing scheme for publishing and subscribing to messages uses a mechanism called a 'topic'. This is a string which acts as a kind of 'subject' field for a message, and which takes a form resembling a URL. A temperature sensor in a study might, for example, publish its current reading using the topic string 'MyHouse/Study/Temperature', or using a shorter string such as 'Test1'. The payload and other parameters are transmitted along with the topic string. MQTT is data-agnostic: the payload can be binary data, text or even XML- or JSON-format data structures.

The reliability of message transfer can be configured at one of three levels as follows.

- Level 0: no particular guarantees ('fire and forget')
- Level 1: guaranteed transfer of at least one copy of the message (but multiple copies might be transferred)

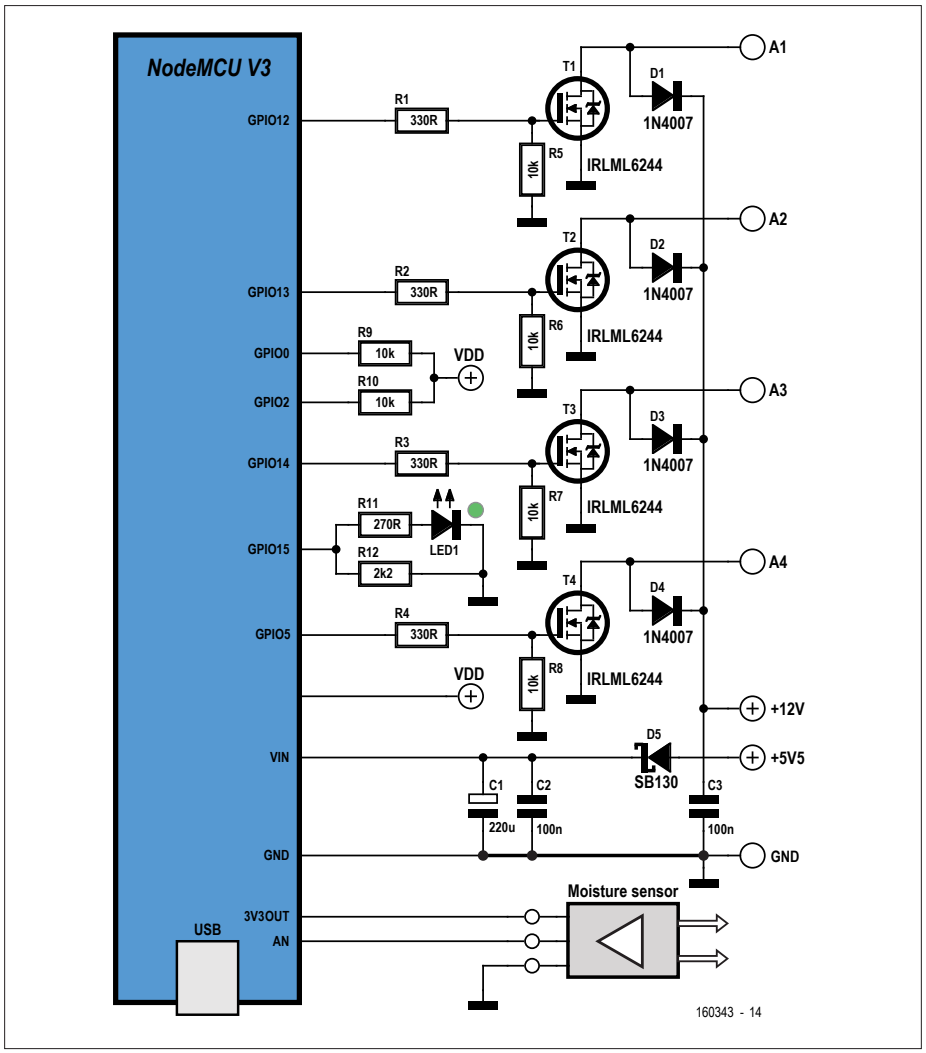


Figure 3. Circuit of the actuator board.

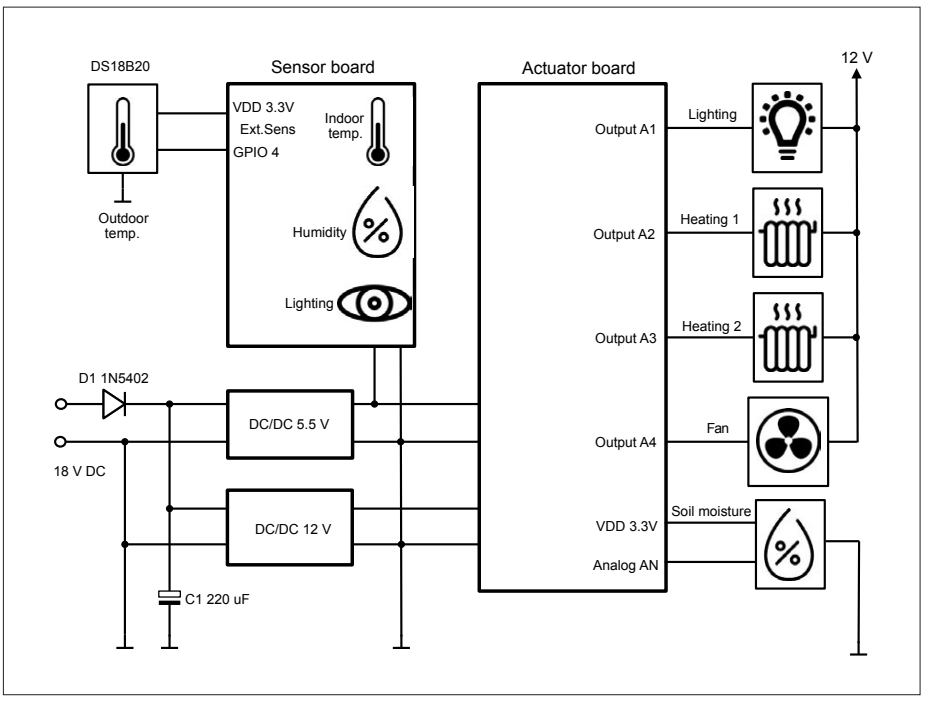


Figure 4. Block diagram of the greenhouse control system.

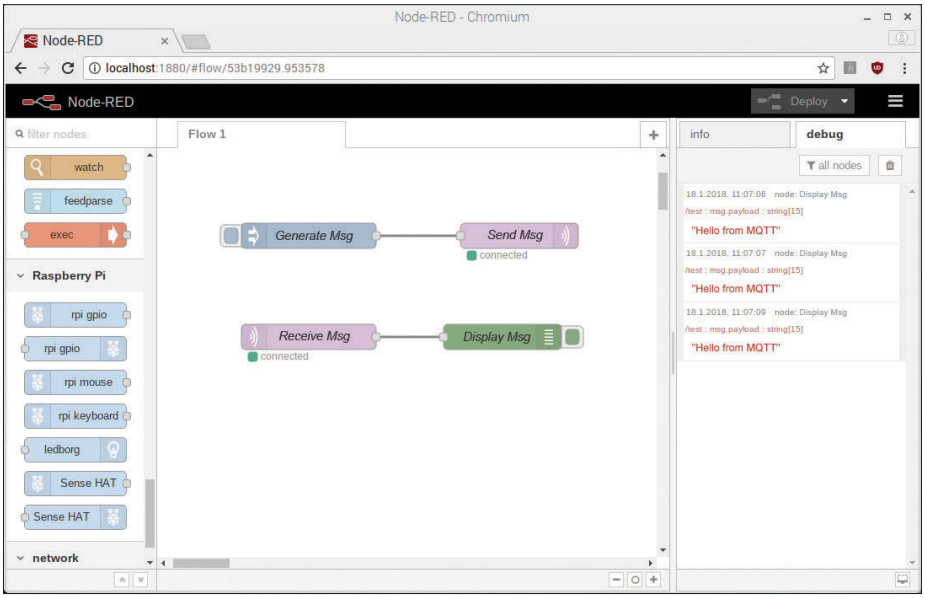


Figure 5. Our first Node-RED flow using MQTT.

- Level 2: guaranteed transfer of exactly one copy of the message

And, perhaps surprisingly, MQTT allows a node to lodge its 'last will and testament' with the broker! What this means is that should the node fail, the broker will execute a specified series of commands. This might, for example, generate an alert for the administrator responsible for maintaining the system.
Now, rather than waxing lyrical over many pages on the subjects of message filtering, multi-layered security and features for protecting against failures, I refer you to the wealth of information about MQTT available on the Internet.

MQTT on the Raspberry Pi

For this project I have selected the free Mosquitto [4] implementation by the Eclipse Foundation from among the many available MQTT brokers. The most recent version (1.4.10-3) can be installed on your Raspberry Pi using the following steps.

```
wget http://repo.mosquitto.org/
   debian/mosquitto-repo.gpg.key

sudo apt-key add mosquitto-repo.
   gpg.key

cd /etc/apt/sources.list.d/

sudo wget http://repo.mosquitto.
   org/debian/mosquitto-stretch.
   list

sudo apt-get update

sudo apt-get install mosquitto
   mosquito-clients
```

Mosquitto runs on the Raspberry Pi as a demon and starts up automatically. To test whether Mosquitto is running use the following command.

```
sudo service mosquitto status
```

The service can be stopped and started as follows.

```
sudo service mosquitto stop

sudo service mosquitto start
```

The following quick test will verify that everything is working. Open two terminal windows on your Raspberry Pi; in the first window subscribe to the topic 'Test' as follows.

```
mosquitto_sub -t test
```

In the second window publish the text 'Hello' under the same topic.

```
mosquitto_pub -t test -m Hello
```

The text 'Hello' should now appear in the first window. The subscription can be canceled by pressing control-C. The basic components are now in place, and we can start to build our greenhouse controller application on top of them.

Graphical programming using Node-RED

In order to minimize the effort involved in building the application a slightly unusual tool was employed. Rather than working with lines of program source text, I used a graphical editor to connect together functional blocks to form the application. Because in this approach messages 'flow' through a network of connections the programs are called 'Flows'. The (free) tool was developed in 2013 by IBM; it is now open source and is called Node-RED [2].
Node-RED is a convenient way to create applications for the Internet of Things as well as in other areas. The tool consists of a run-time environment, a graphical editor for program development, a dashboard for communicating with the application and a wide range of preconfigured functional blocks. These blocks, called 'nodes', can be rather powerful: for example, they can represent MQTT publishing or subscribing clients, web browsers or e-mail subsystems. The nodes have inputs and outputs, process incoming messages, carry out various actions and forward the messages (potentially modified) on their outputs to other nodes.
The structure of an application is described in the graphical editor by placing the required nodes on the canvas and then connecting them with message paths. The flow of messages is triggered by an event, and ends when a node does not produce an output message in response to its input.
Node-RED runs on the JavaScript-based platform Node.js, which in turn is built upon the Google Chrome V8 JavaScript engine. If a node must be configured to carry out some custom task, then this can be done using some JavaScript code: even in Node-RED a little bit of classical programming can be necessary. The graphical editor used for program development and the application interface are both implemented in the browser.
Node-RED has rapidly established itself as a general-purpose development tool for IoT applications. Over recent times a quickly-growing band of users have adopted Node-RED and the

highly-engaged developer community ensures continuous enhancements and maintenance. Today over 1400 special-purpose node designs and example flows are available for download. If you want to know more, again there is a wealth of information available on the Internet.

Our first MQTT flow

Fortunately Node-RED is fully integrated into Raspbian, and the run-time environment can be launched under 'Menu/Programming/Node-RED'.
This opens a terminal window with a few start-up messages that are not relevant to us at the moment. Node-RED can be stopped when it has finished doing its tasks using the control-C key combination. The graphical program editor runs in the Chromium browser: go to the address 'localhost:1880' and you should see the following windows.

On the left, the **nodes** window. This displays over 60 nodes under various categories including input, output, function, social, storage, analysis, advanced, Raspberry Pi and network. The special nodes for use with the Raspberry Pi are particularly useful as they allow convenient access to GPIOs, to the keyboard and mouse, as well as to LEDs and sensors.

In the middle, the **program** window. The required nodes can be dragged into this window, connected together, and then configured.

On the right, the **info** window. This has tabs labelled 'info' and 'debug': 'info' gives information about the selected node while 'debug' shows debug information.

Our first test will demonstrate the operation of Node-RED in the context of MQTT (see **Figure 5**). In this example we have four nodes connected in two flows. In the upper flow the Inject node 'Generate Msg' creates the MQTT payload text 'Hello from MQTT' and sends it to the MQTT Output node 'Send Msg'. This node publishes the message under the topic '/test' to the MQTT broker; and that is the end of this flow.
The lower flow starts with the MQTT Input node called 'Receive Msg', which is subscribed to the topic '/test'. When a message arrives under this topic, the flow forwards it to the Debug node called 'Display Msg'. In the example shown the button on the 'Generate Msg' node has been clicked three times, resulting in three messages being displayed in the debug window on the right.
The parameter settings for any of the nodes described above can be displayed for editing in a window by double-clicking on it. The parameters should be set as follows.
For the Inject node, select 'a-z string' for the payload and enter 'Hello from MQTT'. Set the name to 'Generate Msg' and then click 'Done'.
For the MQTT Output node, edit the server configuration to set the server to 'localhost' and the port number to 1883; then click on 'Add' or 'Update'. Set the topic to '/test', 'QoS' to zero, and the name of the node to 'Send Msg'. Then click on 'Done'.
For the MQTT Input node, set the topic to '/test', 'QoS' to zero, and the name of the node to 'Receive Msg'. Then click on 'Done'.
No parameters need to be set for the Debug node.
When this first test flow has been tested successfully we can start to experiment with more complicated flows.

Putting it all together

We are now at the point where we can define the parameters for our MQTT communications. There are four channels on the actuator board for controlling the lighting, the fan and the two heaters, each of which has a PWM signal which is controlled by a variable in the range from 0 to 255. With an eye towards future expansion to the full-scale system, we will name the corresponding topics 'actor1/x' where 'x' is the channel number. When we add further actuators their topics will be of the form 'actor2/x', 'actor3/x' and so on. The payload for each of these topics could just be a single number in the range from 0 to 255; but in the interests of clarity and to simplify debugging we will add the prefix 'chx_' where again 'x' is the channel number. **Table 1** shows a few examples.
The sensor board delivers readings for ambient light intensity, humidity and indoor and outdoor temperature. The topic strings we use will be 'sens1/' plus the sensor type. The range of possible values varies considerably from sensor to sensor: for example, the light sensor returns values from 0 to 1023, where 0 corresponds to maximum brightness and 1023 to complete darkness. The humidity sensor returns the relative humidity in percent, and the two digital temperature sensors return readings in Celsius, with negative values suppressed.
The actuator board hosts the soil moisture sensor: for this we use the topic 'sens2'. Again in this case the range of possible values runs from 0 to 1023, where 0 corresponds to maximum moisture content and 1023 to zero moisture content. The lighting and soil moisture readings are converted into percentage values in the central controller.

Installing the greenhouse flow

Before we can install the (somewhat complex) greenhouse flow a little more groundwork is necessary. First install the

Table 1. Example MQTT messages.

Function	Topic	Payload	Range	Example
PWM signal on A1 (light)	actor1/ch0	ch0_xxx	0 to 255	ch0_64
PWM signal on A2 (fan)	actor1/ch1	ch1_xxx	0 to 255	ch1_255
PWM signal on A3 (heater 1)	actor1/ch2	ch2_xxx	0 to 255	ch2_0
PWM signal on A4 (heater 2)	actor1/ch3	ch3_xxx	0 to 255	ch3_128
Light intensity	sens1/lght	lght_xxxx	0 to 1023	lght_500
Indoor air humidity	sens1/humi	humi_xxx	0 to 100	humi_80
Indoor temperature	sens1/tpin	tpin_xxx	0 to 100	tpin_27
Outdoor temperature	sens1/tpau	tpau_xxx	0 to 100	tpau_22
Soil moisture level	sens2/erde	erde_xxxx	0 to 1023	erde_300

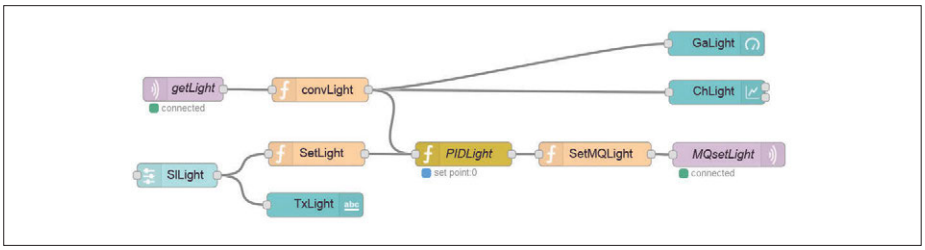


Figure 6. Flow for lighting control.

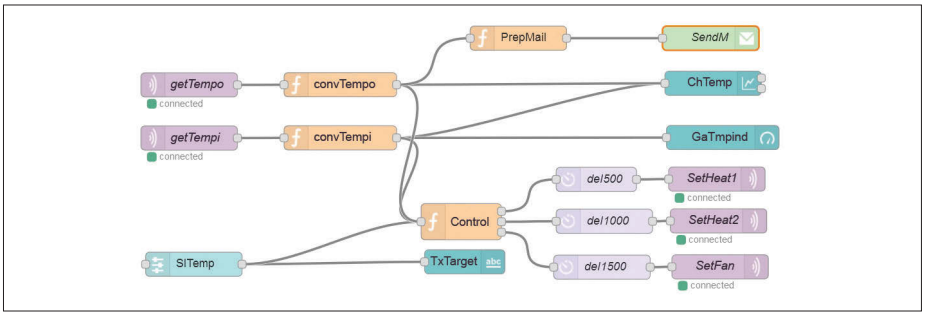


Figure 7. Flow for temperature control.

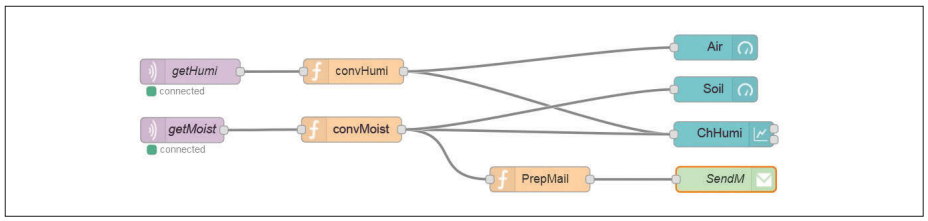


Figure 8. Flow for humidity and soil moisture monitoring.

npm package manager on the Raspberry Pi using the following command.

```
sudo apt-get install npm
```

Then move to the Node-RED directory (‘/node-red’) and install the Node-RED dashboard. This gives a user-friendly graphical interface to the Node-RED system.

```
npm i node-red-dashboard
```

And now we use npm again to install the PID controller node as follows.

```
npm i node-red-node-pidcontrol
```

The Raspberry Pi must now be rebooted and Node-RED launched again. In the Node-RED editor you will see a row of new nodes for creating graphical output. We can now proceed to load the flow for the greenhouse application. Node-RED flows are stored in a JSON-based format and they can be imported and exported via the clipboard. Copy the file Greenhouse_Flow2.json from the download page on the Elektor website [3] onto the Raspberry Pi, open it using the Geany editor, copy the contents of the file to the clipboard and then import it into Node-RED using the Import/Clipboard menu option as ‘Flow2’. You will now be able to inspect the overall flow with the component flows for handling light, temperature and soil moisture content.

PID control for lighting

The lights in the greenhouse are controlled smoothly using a PID controller to maintain a chosen constant overall ambient light intensity (**Figure 6**). ‘PID’ stands for ‘Proportional-Integral-Differential’, a control algorithm that, assuming its parameters are appropriately set, guarantees good tracking of the target value. The mathematics behind the controller is implemented in the PIDLight node; the behavior of the controller is determined by three constants giving the coefficients of the proportional, integral and differential terms in the control loop.

In this flow the current ambient lighting level is periodically received by the getLight node over MQTT. The value is normalized in the convLight node and then passed to the PIDLight controller. The target level is set using the SlLight slider node, and this value is fed into the controller where it is compared with the current reading. The output of the controller is a correction signal which is converted into an MQTT payload in the SetMQLight node. The actuator board receives this message and adjusts the brightness of the lighting accordingly. The ‘function’ nodes, which are marked by a script letter ‘f’, each include a few lines of JavaScript to implement their specific task. The blue nodes are graphical input and output elements that appear on the dashboard, such as sliders, text boxes, gauges and charts.

Two-input temperature controller

In this component flow (see **Figure 7**) the current outdoor and indoor temperature readings are received over MQTT. If the outdoor temperature falls below 4 °C a warning message is prepared in the PrepMail node and sent to the operator in the SendM node. Temperature readings, after normalization, are fed to the two-input Regulator node, where they are compared with the target value set by the SlTemp slider node. The Regulator node computes the control values for the heaters and for the fan and sends them to the actuator board using MQTT.

This controller does not use any difficult mathematics. Instead, a simple series of thresholds are defined and the power to the heating lamps is adjusted in discrete steps. Depending on the indoor and outdoor temperature levels the fan will contribute either to heating or to cooling the greenhouse. In order to avoid large spikes in current draw when several loads are switched simultaneously, the MQTT messages are sent with delays of between 500 ms and 1500 ms. Again, information about the status of the flow is displayed on the dashboard.

Soil moisture level monitoring

The air humidity and the soil moisture level readings are normalized and displayed on the dashboard. If the soil moisture level falls below 20 % a message is prepared in the PrepMail node and sent to the operator in the SendM node (see **Figure 8**).

System overview

The aim of this project is to display all the data relevant to the operation of the greenhouse in one place, and to test the behaviour of the controllers. To this end we make use of the dashboard and nodes available in Node-RED for creating graphical displays: these need to be specially installed as we described above. The dashboard can be displayed by opening a new tab in the browser and entering the following address.

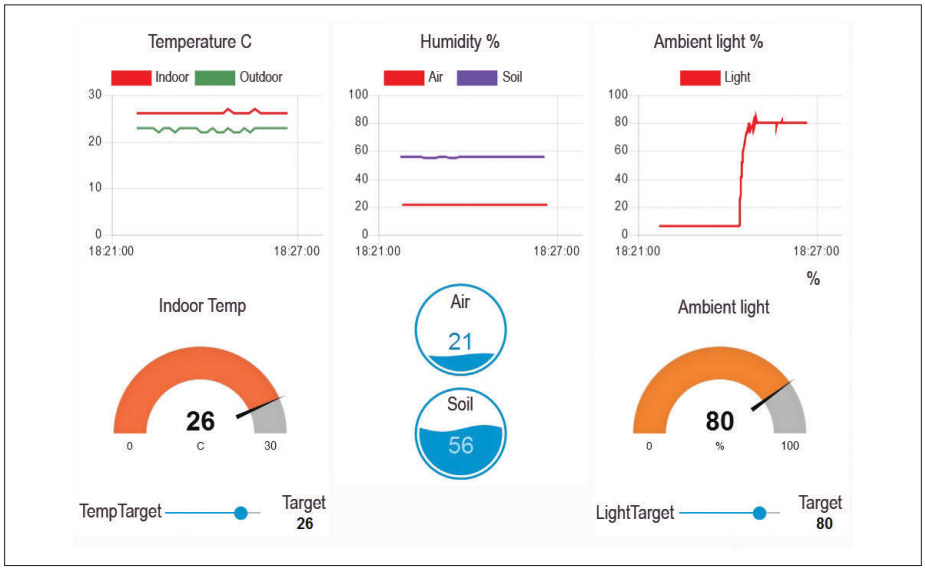


Figure 9. Operating parameters on the dashboard.

```
http://localhost:1880/ui
```

This window (see **Figure 9**) just shows the graphical elements themselves. To configure the parameters of a dashboard node double-click on it in the Node-RED editor window.

Across the top of the display there are three long-term charts showing the measured temperature, humidity and lighting levels. In this example we have set the time period displayed to just 5 minutes so that even small changes in temperature can be seen. In practice the period would be set to several hours, or perhaps one would implement a combination of short-term and long-term charts. It is possible to display more than one variable on a chart, and since version 2.3.2 of the dashboard it is possible also to specify the colours of the lines. The current indoor temperature, humidity and light levels are also displayed using a selection of indicator designs.

At the bottom of the display there are two sliders to set the target values for indoor temperature and light level. The screenshot shows how the dashboard appears on an Android tablet; on a smartphone the graphical elements are presented vertically, but the result is still perfectly legible.

Learning all the time...

The model greenhouse demonstrates that the full-scale project is well within the capabilities of the Raspberry Pi and ESP8266 hardware and the MQTT and Node-RED software. The combination of MQTT and Node-RED makes it easy and quick to implement the application logic. It is particularly useful for debugging to be able to see all the messages (or just the messages of interest) on the PC by simply using the MQTT subscription mechanism.

However, a couple of modifications will be necessary. The central Raspberry Pi controller is a 'single point of failure' and reliability will have to be improved by using multiple brokers with MQTT bridges or a cluster configuration. Means will have to be found to protect the network from external attack, which could compromise its operation. One approach to this would be to use TLS (transport layer security) encryption for all communications. These precautions are all perfectly feasible, but require know-how and extensive experimentation and testing. But that's what makes the job of an engineer so fascinating!

So that interested readers can quickly carry out some initial tests on a Raspberry Pi, I have created a complete Raspbian Stretch image with broker, dashboard and flow pre-installed. The image is available from the Elektor website [3]. It should be noted that this article is classified as Homelab hence was not processed by Elektor Labs. Consequently some parts of the program are in German. After unpacking the image it can be copied directly to an SD card from which the Raspberry Pi will boot. A C++ program for the Arduino IDE and a C version for the CHERTS SDK [5] are also available for flashing the ESP8266 firmware: make sure you read the notes in the README file. Happy experimenting!

(160434)

Web Links

[1] ESP8266: https://bbs.espressif.com/

[2] Node-RED: https://nodered.org

[3] Software download: www.elektormagazine.com/160434

[4] Mosquitto: http://mosquitto.org

[5] CHERTS SDK: www.esp8266.com/viewtopic.php?f=9&t=820

(almost) everything you wanted to know about...

Bluetooth

By **Robert Lacoste** (France)

Q *Let's begin at the beginning: where does Bluetooth come from?*

A Bluetooth was developed in the 90's following an initiative by engineers from Ericsson Mobile in Sweden. The objective was apparently to get rid of the cable which connected headsets to portable telephones. The industry association which specified and standardised this protocol, called **Bluetooth Special Interest Group** (SIG), was officially created in 1998. The first Bluetooth compatible portable appeared on the market in 2000, with the first associated headset. That was the start of the adventure...

Q *So why the strange name – "Bluetooth"?*

A It seems that it comes from the nickname of a Danish king of the 10th century, *Harald Blåtand*, who unified a good proportion of the Danish tribes… and who liked blueberries so much that he ended up with blue teeth. Also, the Bluetooth logo is a combination of his initials H and B in the runic alphabet!

Q *Bluetooth is now 20 years old, why are we still talking about it?*

A For two reasons. First of all, Bluetooth is, and has been for a long time, the most widely used radio protocol in the world: more than 3 billion Bluetooth-equipped products come out of factories every year! Secondly, because Bluetooth continues to evolve with new functions in each new version, yet retaining an impressive compatibility with older versions.

Q *Let's go back to basics. What are the characteristics of Bluetooth on a technical level?*

A Under the hood, it's a fairly complex protocol, but based on simple concepts. First of all it uses the 2.4 to 2.48 GHz ISM frequency band, available the world over without licensing. Also, it uses the principle of fast frequency hopping: the band is composed of 79 1-MHz channels, and a Bluetooth radio changes channel 1600 times per second, using a pseudo-random channel sequence. A mechanism called AFH (**Adaptive Frequency Hopping**), allows it to avoid to a certain extent the channels used by other radio systems such as Wi-Fi. A Bluetooth network, called piconet, is a network using star topology with a master (usually a smartphone) and up to seven slaves simultaneously communicating with the master. The basic bit rate is from 1 to 3 Mbps depending on the version. Lastly Bluetooth specifies high level application layers for transferring audio, data etc., with the necessary smarts for the products to discover the characteristics of their alter egos themselves, via standard profiles.

Q *What about Bluetooth Low Energy (or BLE), what are the differences with Bluetooth?*

A Bluetooth was invented for the transfer of audio data; BLE (which comes from an initiative of Nokia called Wibree) is aimed at battery powered devices, with low data rates and very low power: connected watches, sensors, etc. BLE is a variant of Bluetooth, available since version 4.0, and has been the object of numerous improvements in later versions. In a few words, BLE only uses 40 channels, of which only three are used to discover peripherals (broadcast channels), which considerably simplifies things. The application profiles have also been simplified with a model called GATT (Generic ATTtribute profile).

Q *BLE Beacons… what are they?*

A It's a sort of BLE product reduced to its simplest terms: a device that periodically announces itself on the three broadcast channels. This can be used to locate oneself (for example by installing a beacon in each room), to detect the presence of an object or a person equipped with a beacon, to broadcast sensor information, etc.

Q *We talk a lot about hacking. How secure are Bluetooth exchanges?*

A There was a lot of noise around a security failure of the classic Bluetooth protocol (non BLE), due to an error deep in the specifications, but which is rapidly being corrected. Basically, BLE and Bluetooth are reasonably secure, in particular since version 4.2 which introduced an LE Secure Connection mode based on solid cryptographic principles. But it's necessary to use these modes in products, as their use is optional. In concrete terms, the most problematic phase is the initial pairing: if there is no code demanded of the user (which needs a keyboard and/or screen in each product) then this phase needs to be carried out where hackers cannot listen in. If this is not done then the security of all further exchanges is com-

promised. So don't pair your new gadget to your smartphone in the middle of a shopping mall…

Q *It seems that with the latest version, Bluetooth 5.0, the communication will be faster and have a much greater range, is that true?*

A Yes and no. Bluetooth 5.0 introduces on one hand a fast mode (2 Mbps against 1 Mbps from BLE currently) and on the other hand a long range mode (coded PHY, at 125 Kbps but with an effectively doubled range). As often happens, it's a choice between cheese or dessert: either a faster rate at lower range, or the other way round. But note that both these functions are optional, so a product can well be specified as 5.0 without having any improvement over a 4.2 product. It's enough to conform to the latest version of the specifications while declaring that it does not support any of the optional functions. So beware of inflated claims, and read the fine print!

Q *Are any other big advances in the pipeline for Bluetooth?*

A Yes, Bluetooth Mesh is already specified and is beginning to become available. This improvement, independent of the actual version (it may be supported by Bluetooth 4.2 or 5.0 for example), allows for the extension of the range of a BLE network by relaying messages between the devices. This new version is particularly aimed at home automation applications: connected lamps and switches, etc., via new application profiles which are not compatible with standard BLE. This will be serious competition for Zigbee which has occupied this niche for many years.

Q *Let's leave the technical aspects. It seems you need to pay to sell a Bluetooth product, is that true?*

A Very much so. If you want to put the Bluetooth logo on your product, you must conform to the Bluetooth SIG (that part is free), declare your product on the website, fill out a long form describing in detail the Bluetooth functions that are used, undergo a series of compatibility tests (using an automatic test generator kindly provided by the Bluetooth SIG), and finally pay a fee between $ 2500 and $ 8000 depending on the type of product. You can add to that other costs like the purchase of a range of MAC addresses from the IEEE or for a number to create your very own GATT profile ($ 2500). The only time that it's not *a priori* necessary to pay is if you don't tell anyone that there is Bluetooth in your product. But that prohibits you from, for example, explaining how to connect to a smartphone. All this is over and above the costs of tests for conformity tied to the CE mark.

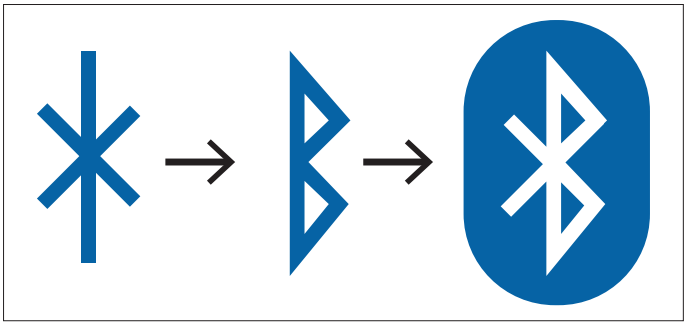


Figure 1. The letters H and B in the runic alphabet.

Q *Yes, but if you just use a Bluetooth module that's already been certified, then all that is not necessary, is it?*

A That's absolutely false, and it's been that way for several years now. Using a precertified module in most cases just means you don't have to undergo the Bluetooth testing, but you still need to confirm to the Bluetooth SIG, to declare your product and pay the $2500 to $8000 to have the right to say that your product has Bluetooth. Have a look at the FAQ at bluetooth.com.

Q *What does the future hold for Bluetooth?*

A Bluetooth is used in 100% of smartphones manufactured in 2018 and in a considerable number of other products: so it will remain the dominant protocol. Smartphones are becoming a sort of universal remote control, and even replacing PCs. Bluetooth is the essential protocol for interacting with smartphones. There's no other solution besides Wi-Fi or NFC, and these protocols are really for other applications. But it is probable that in the short term BLE will replace the classic Bluetooth for all applications, including Audio. It's an open secret, but everyone seems to be going that way. For its part, Bluetooth 5.0 will complicate BLE significantly, but will greatly increase the areas of application. Finally, the future will tell us if Bluetooth Mesh will be a success or not … ⏮

(160654)

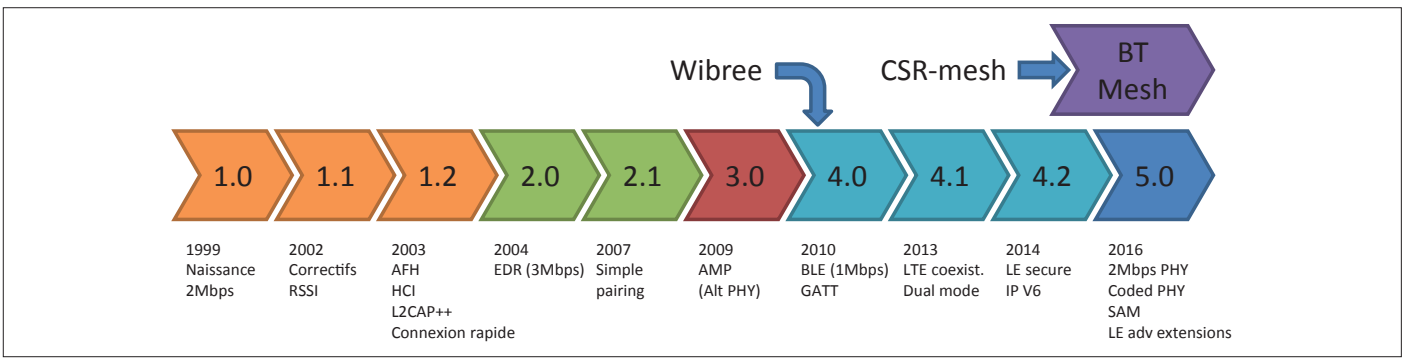


Figure 2. Bluetooth: from the beginning to today.

Tips and Tricks

From readers for readers

Another neat solution to a tricky problem.

Arduino board provides serial/USB adapter

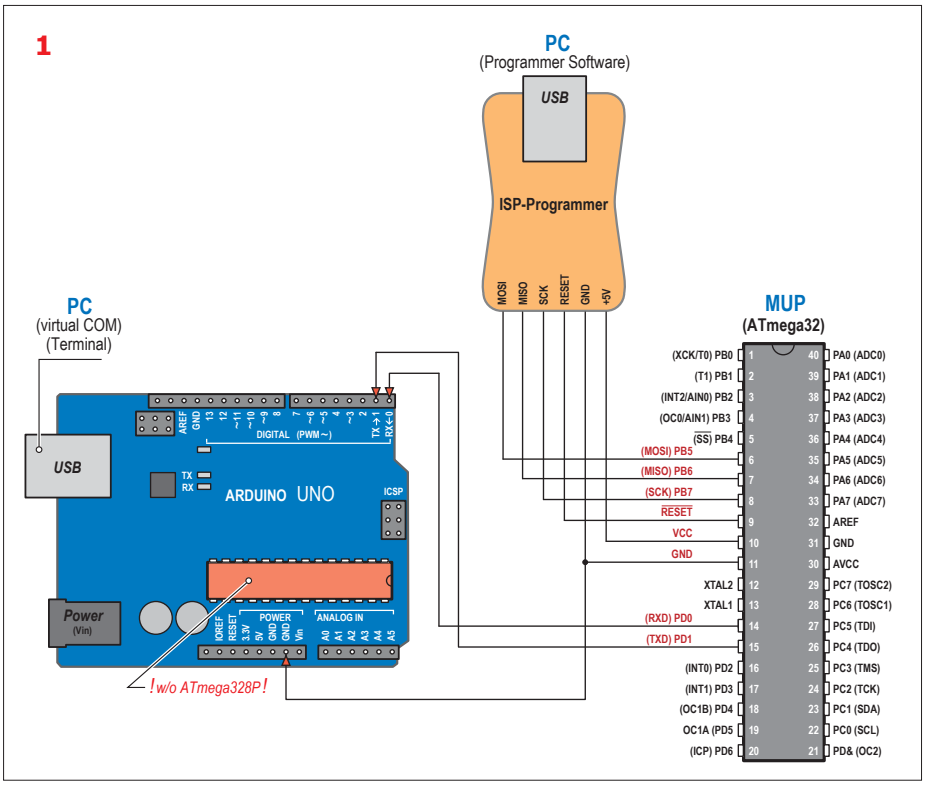
By **Friedrich Lischeck** (Germany)

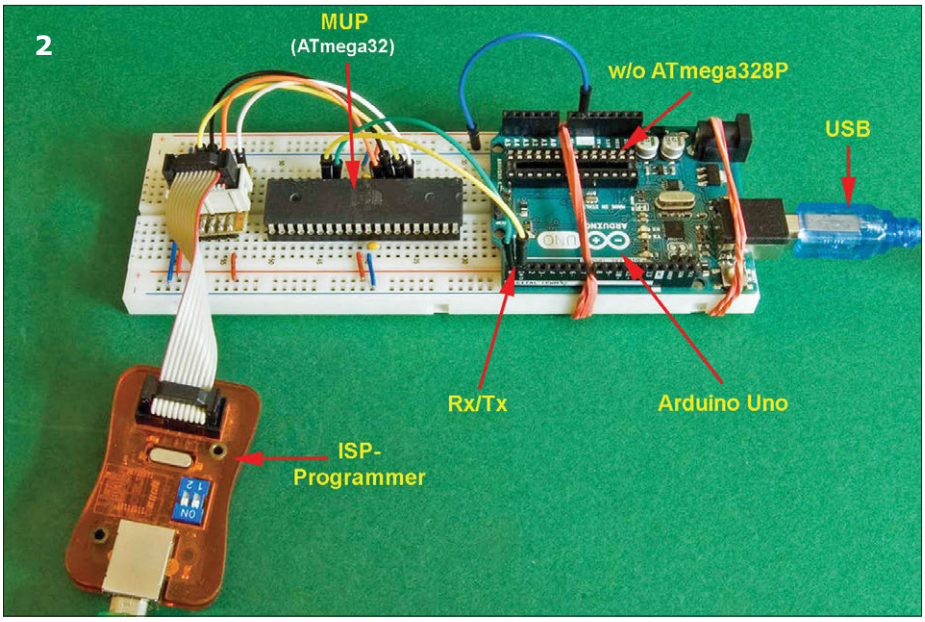
"Recently I was developing a program using a naked or stand-alone AVR microcontroller i.e. without the Arduino environment. For debugging purposes, a serial communication port would have been useful but most modern PCs don't provide the necessary hardware port. What I needed was an interface adapter to allow UART communications over a USB port. I did the job using a spare Arduino Uno board. This board has a built-in adapter and uses an ATmega328P in a DIL socket. It was necessary to extract the ATmega from the Arduino board, use some flying leads to link the Rx and Tx signals from the Arduino board to my MUP (MPU under Programming), fire up a terminal emulator program on the PC and I was in business."

Well that just about sums up the whole trick suggested here but don't worry, I have also included more useful information about its implementation. When you develop programs for an AVR MPU (or any microcontroller from a different manufacturer) seldom will the program run bug-free at the first attempt. When the target hardware is say an Arduino Uno then for debugging purposes its normal to insert code sections which output values of variables to the serial monitor built in to the Arduino IDE. This is useful to help spot bugs occurring during program execution. Programming tools with debugger functions are also useful but they are generally a) quite expensive and b) usually only applicable to just one family of controllers and therefore can't be considered universal tools.

Naked-controller debugging

The same is true when it's necessary to develop programs for stand-alone AVR-





About the author

Friedrich Lischeck has worked in IT sales; IT support and customer service for a telecommunications company and is now retired. Today his interests include analogue electronics as well as AVR controllers and also Raspberry Pi projects.

MPU applications or any other controller not plugged into an Arduino board or equivalent IDE. Without a dedicated debugger it is useful to output values of certain variables to a serial interface during program execution so that they can be displayed and analysed on a PC running a terminal program. Almost every type of microcontroller has a couple of pins dedicated to serial communications; if not the necessary code is usually already available to quickly implement a serial communication channel using two spare GPIOs.

The frustrating part of all this is that modern PCs can't really handle serial communication anymore; hardly any are supplied with serial ports these days. Fortunately there are many low-cost, off-the-peg solutions to the problem; a small module or adapter cable converts a USB port to a UART (see below). What happens if you don't have such an adapter or maybe it's being used on another project?

Adapter stand-in

Recently the author of this project found himself in exactly this situation. He quickly realized that the Arduino is so ubiquitous that almost every electronics enthusiast is likely to have a spare Uno lying around just collecting dust. Indeed he was no exception and was able to lay his hands on one without too much difficulty. The board has a main ATmega controller which runs the application code and also a separate controller dedicated to serial interface communications. Fortunately the original Uno board uses an AVR controller (the popular ATmega328P-PU or –PN) in a DIL outline fitted in a socket. That makes it super simple to just unplug the microcontroller and turn the Uno board into a dedicated interface adapter.

Go configure

The diagram in **Figure 1** *shows what needs to be done. The ATmega328P controller should first be removed from its socket on the Arduino board. Now it cannot affect or interfere with the Rx and Tx signals on the serial interface. It's also possible to gently bend pin 2 (RXD) and pin 3 (TXD) outwards slightly so that if you need to re-insert the controller these two pins will not make contact in the socket. Using some flying leads connect pin 0 (Rx) and pin 1 (Tx) of the Arduino header connectors (the strip running along the left of the board with the USB connector facing you) to the corresponding pins of the MUP. In the setup shown here I have used an ATmega32 as the MUP so that pin 0 (Rx) signal from the Arduino board connects with pin 14 (RXD) of the ATmega32 and pin 1 (Tx) of the Arduino board connects to pin 15 (TXD) of the ATmega32. Yes, you read that correctly; the Rx and Tx signals are not swapped here. Now complete an earth connection (GND) between the two boards as shown in* **Figure 2** *where pin 11 (and/or pin 31) of the ATmega32 connects to a GND connection on the Arduino board and also GND of the ISP connector. To program the MUP you just need to connect a low-cost ISP programmer to the ISP leads (SCK, MOSI, MISO, Reset, +5V and GND) of the MUP. Debugging communication takes place via the Arduino board and the whole setup requires two USB ports on the PC.*

And another thing

Clones of the Arduino Uno board are available from various manufacturers at very affordable prices. While we only recommend using genuine Arduino products it may be worthwhile considering a low-cost clone for the specific purpose of providing the serial/USB conversion. At the end of the day you have the versatility of using the board for serial/USB conversion or as a Uno development platform. Some Uno clones use an SMD version of the mega328P controller, in this case disconnecting the TX and RX signals will not be as simple as unplugging the controller from its socket. For simplicity use a board with a socketed DIL controller like an original Arduino Uno (available from the Elektor Store). It is of course always possible to use a standard dedicated interface adapter module to perform the serial/USB conversion (see box 'From the Store'). ⏮

(160509)

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)!

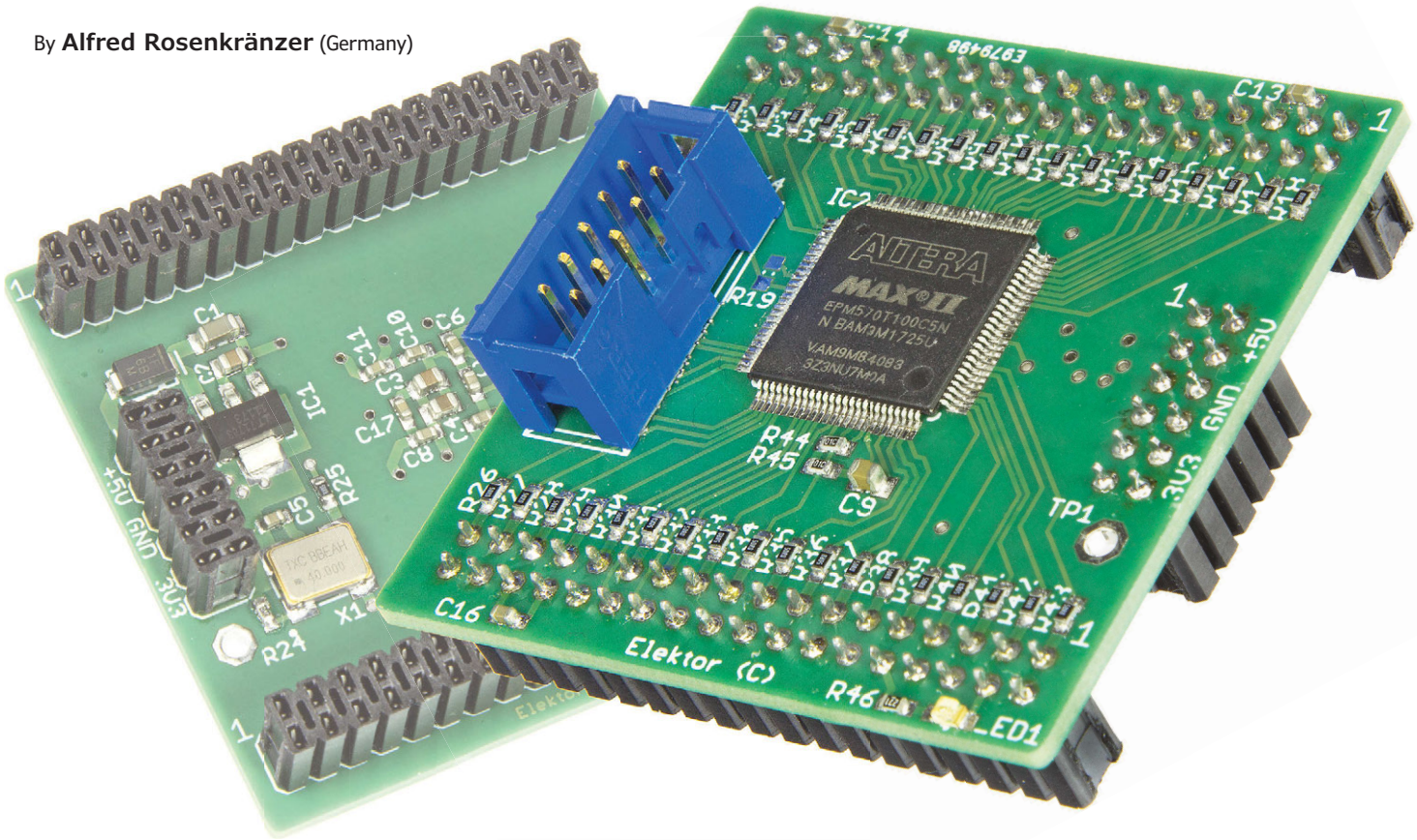
eLABS PROJECT

CPLD Breakout Board in DIL Format

For programmable logic projects

CPLDs tend to have a large number of pins practically inaccessible to the soldering iron and so do not lend themselves readily to use in homelab and experimental projects. The BoB described here brings out the connections to header sockets, allowing you to use the CPLD as easily as a DIL IC on prototyping boards or straight in prototypes.

By **Alfred Rosenkränzer** (Germany)



It has been a long time since digital logic circuits were routinely constructed using discrete TTL or CMOS parts. Such designs were large and very inflexible; any error or modification meant the addition of bodge wires and ultimately a new circuit board layout at the end of the development phase. The swathes of chips of past times have now been replaced by complex components that integrate many programmable logic functions. All very convenient, but the high pin counts of these devices and the difficultly of soldering them represent considerable obstacles to using them in experimental circuits and prototypes. The board described here makes modern CPLDs with 100 pins 'pluggable', just like a conventional (large) DIL IC.

The story of programmable logic

Older readers of *Elektor* will perhaps remember the PLA (programmable logic array) and PAL (programmable array logic) devices of the early 1980s. These devices contained eight or sixteen flip-flops, along with some logic gates, and they could be programmed once only. In the mid-1980s these rather awkward-to-use chips were replaced for circuit development purposes by GAL (generic array logic) devices. These offered the same functionality as their predecessors, but were electrically erasable and reprogrammable: a significant step forward!

However, a GAL would contain only the equivalent of a few hundred logic gates, which quickly became inadequate in the face of ever-increasing demand for capacity. So several GALs were integrated together in a single package, equipped with programmable interconnect: the result was the CPLD (complex programmable logic device). These components contained a large number of logic gates and flip-flops and could be reprogrammed. The number of available I/O pins grew steadily and high clock frequencies meant that multiple separate VCC and GND pins were required. Consequently it became impractical to program these high-pin-count monsters in their (very expensive) programming adapters; to combat this came the ability, which these days we take for granted, to program devices in the target circuit.

The MAX7000 series from Altera (now Intel) was perhaps the last range of devices that could feasibly be mounted on prototyping board using a PLCC socket. They have long since been technologically superseded and are no longer available.

The current MAX II, MAX V and MAX 10 series from Altera are very attractive and economical. However, they are only available in packages that are difficult or impossible for the hobbyist to solder. Even for prototypes it is necessary to design a complex circuit board and have it manufactured and populated.

In terms of power FPGAs (field-programmable gate arrays) have now eclipsed CPLDs. They are considerably more complex internally and include special functions such as a programmable processor core, a feature only found on the larger MAX 10 CPLDs. A significant disadvantage of FPGAs in comparison to CPLDs, however, is their volatility: when power is removed the programed configuration is lost, and when power is reapplied the FPGA must be programmed afresh from an external device acting as a 'boot ROM'.

A CPLD board for prototyping

This project enables you to exploit the advantages of the MAX II devices and build prototypes economically on perforated board. All the essential features for programming and running the CPLD are included on a four-layer board that we have made as compact as possible. As the block diagram in **Figure 1** shows, it comprises the following elements.

- the CPLD itself;
- a voltage regulator;
- a power indicator LED;
- decoupling capacitors;
- a JTAG port for programming;
- a clock source;
- double-row header sockets for connection to the main circuit board.

The whole thing can be used like a DIL IC, plugged into the prototype circuit and then, when you have finished with it, you can remove it from the board and use it in your next project.

Earlier revisions of the CPLD board included a DIL switch, LEDs and the ability to run the different I/O blocks of the device from different supply voltages. However, these features all add to the size of the board and do not have very many practical applications. Switches, LEDs and the like really form part of the application circuit itself and so belong on the main printed circuit board.

To allow the CPLD to be used at the highest possible clock frequencies, a four-layer board was designed. The two external layers, as shown in **Figure 2**, carry the signals, while the inner layers carry the supply voltage (layer 3) and ground (layer 2). The SMD decoupling capacitors are arranged under the IC. To keep the board dimensions compact not all the I/O pins are brought out to the header sockets.

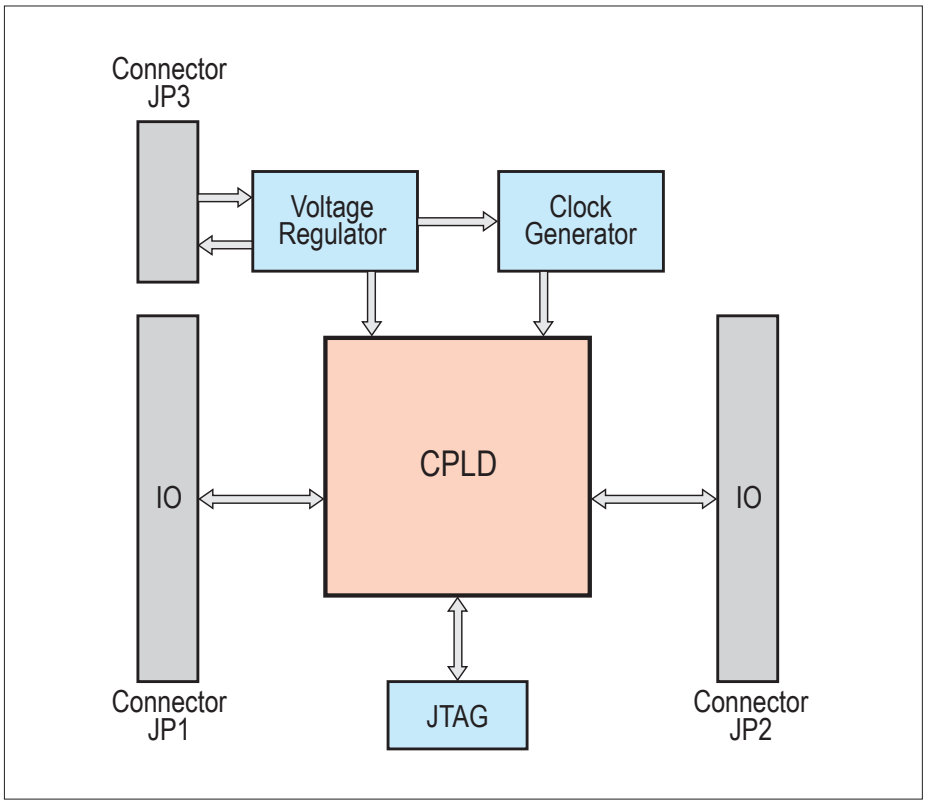


Figure 1. Block diagram of the CPLD board.

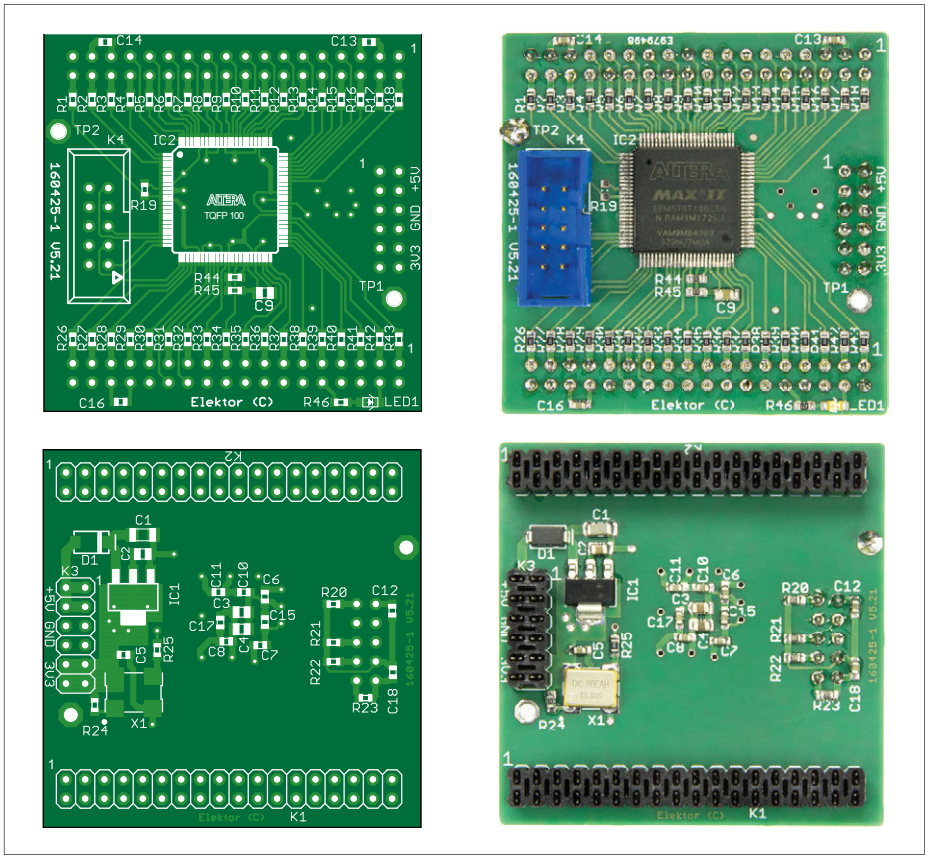


Figure 2. Top and bottom layers of the CPLD board. The inner layers of the four-layer board are power planes.

CPLD support circuitry

We chose the Altera EPM570 CPLD from the MAX II family for the board. This device is available in a range of different packages, of which the most user-friendly is certainly the 100-lead TQFP. The EPM570 is also available in a 144-lead TQFP and in various BGA packages, but the latter can only be soldered using reflow techniques. The EPM570 has an integrated linear voltage regulator that generates a core voltage of 1.8 V from a supply at either 3.3 V or 2.5 V. The board includes a small voltage regulator that can generate this 3.3 V or 2.5 V stabilized supply, which is also used to power the I/Os. There are also 'G' and 'Z' versions of the EPM570 which do not have the integrated voltage regulator. These devices can be used with our board, but then the external regulator on the board must generate the 1.8 V supply. This voltage will then also be used to power the I/O pins.

The CPLD has 570 LEs (logic elements). In the counting method used in the older MAX7000S family that would correspond to around 440 macrocells. Either way, that is a considerable amount of logic. The EPM570 is also pin-compatible with the smaller and cheaper EPM240, which has 240 LEs and which can also be fitted on this board. **Figure 3** shows the complete circuit of the module, centred on the CPLD.

Supply

A power supply at approximately 5 V should be connected to pins 1 to 4 of header socket K3. D1 ensures that the circuit cannot be damaged if the polarity is reversed. Voltage regulator IC1 generates the supply at 3.3 V (or 2.5 V) which powers both the CPLD and the clock generator. C1 and C2 are decoupling capacitors and for this reason are mounted close by the IC. The stabilised voltage is available on headers K1, K2 and K3 and so can be used on the motherboard as well if required. Of course, an adequate number of ground connections is also provided. LED1 indicates when VCC is available. A full six vias are provided to make a low-impedance connection between the output of the regulator and the third layer of the printed circuit board; they also help to dissipate some heat.

As noted above the board has four layers, of which layer 2 (GND) and layer 3 (VCC) are designed as planes in order to keep the power supply as clean as possible. The two layers form a small capacitor which helps with supply decoupling. The decoupling capacitors themselves (C6 to C8 and C10 to C12) are fitted directly under the CPLD and the connections to their vias are as short as possible. C13, C14 and C16 sit

COMPONENT LIST

Resistors

Default: 1%, SMD0603
R1–R18,R25–R43 = 50Ω (75Ω) optional (see text)
R40–R43 = 270Ω
R19 = 10kΩ optional
R20,R21,R22 = 2kΩ
R23,R24,R44,R45 = 10kΩ

Capacitors

Default: 25V
C1 = 10µF, SMD1206
C2,C3,C4 = 10µF, SMD0805
C5–C8,C10–C18 = 100nF, SMD0603
C9 = 2.2µF, SMD0805

Semiconductors

D1 = 40V, 1A Schottky diode, (Vishay VS-10MQ040-M3/5AT)
LED1= green (Kingbright KP-2012CGCK)
IC1 = LT1117CST-3.3 (Linear Technology)
IC2 = EPM570T100C5N (Altera) (see text)
X1 = 40MHz oscillator module, 50ppm, SMD 7x5mm (TXC 7W-40.000MBB-T)

Miscellaneous

K1,K2 = 36-way (2x18) header socket, 0.1" pitch
K3 = 12-way (2x6) header socket, 0,1" pitch
K4 = 10-pin (2x5) boxheader, 0.1" pitch
PCB 160425-1

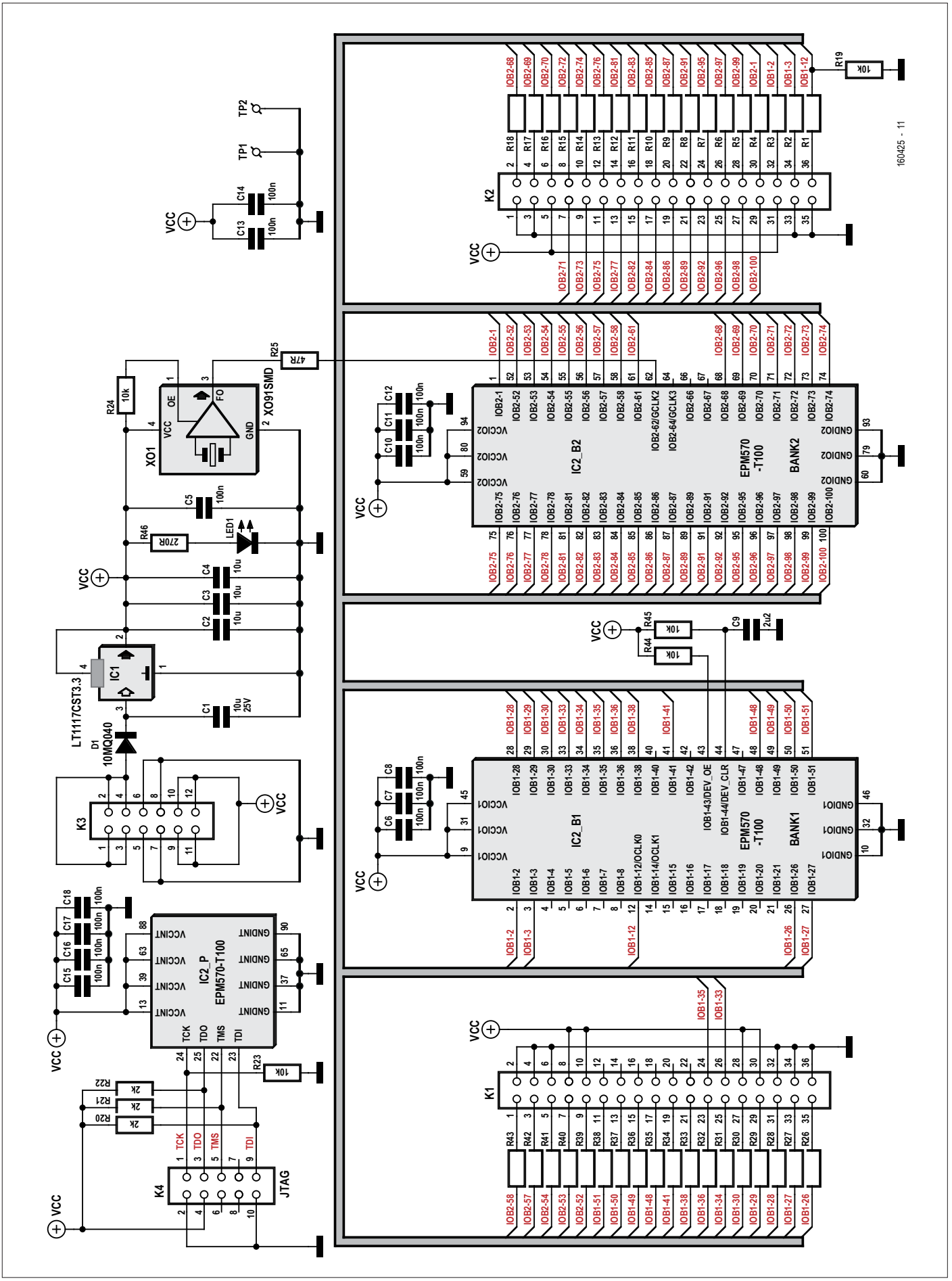


Figure 3. So many connections! Note that the CPLD is shown in three separate parts in the circuit diagram.

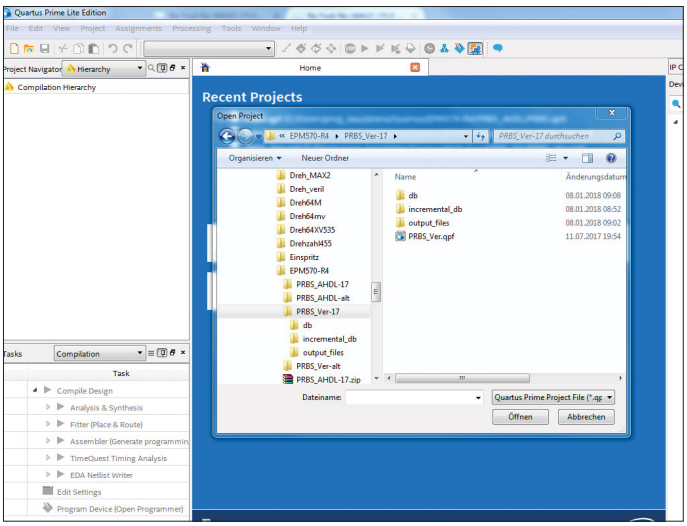


Figure 4. Opening the project file in Quartus.

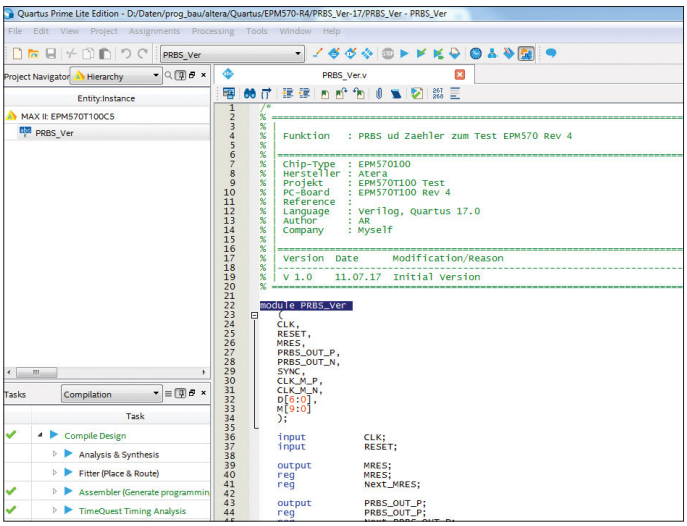


Figure 5. The PRBS project in the editor.

in the corners of the board, near to the VCC and GND connections on headers K1 and K2. Test points TP1 and TP2 provide ground points for connecting oscilloscope probes.

Clock generator

XO1 is a type XO91 integrated crystal oscillator which provides a clock signal to global clock input GCLK2 (pin 62) of the CPLD via resistor R1. Decoupling capacitor C5 is connected directly to pin 4. The OE input of the device is pulled to VCC by R24 (although in fact there is also a pull-up resistor in the IC). If the clock generator is not to be used it can be disabled by adding a wire from this pin to ground. This makes sense if it is desirable to minimize interference.

According to its data sheet the oscillator available in clock frequencies from 500 kHz to 125 MHz. Different versions are also available for different supply voltages (3.3 V for devices with the 'A' designator, 2.5 V with the 'B' designator), so be sure to select the correct part. The device is readily available and inexpensive at just over US$1. A second clock input to the CPLD, GCLK0 on pin 12, is brought out to pin 36 on header K2, so that you can connect an alternative external clock if necessary. In that case R19 can serve as a 50 Ω termination resistor, with a 0-Ω resistor fitted in position R1.

Device reset and output enable

DEV_CLR (pin 44) is used to reset the device and is connected via a resistor to VCC and via a capacitor to ground. This ensures that the CPLD is reset when power is applied. The global output enable (OE) on pin 43 is pulled up to VCC via a resistor.

Programming adapter

The CPLD is programmed over the 10-pin JTAG connector K4, which is connected directly to the JTAG port of the IC. In turn this can be connected to a PC using, for example, an Altera USB-Blaster or Ethernet-Blaster. Pull-up and pull-down resistors R20 to R23 keep the inputs at defined levels if no programming adapter is connected. Two pins of the JTAG connector are taken to the power supply in order to power the driver in the programming adapter. C12 and C18 decouple the VCC supply at the JTAG connector.

I/Os

As many as possible (but not all: 50 out of 74) of the I/O pins of the CPLD are brought out to the pins of K1 (20 I/Os) and K2 (30 I/Os). The interior pins are connected via series resistors to the pins of the IC. We do not specify the values of these resistors in the circuit diagram for a good reason: for example you could fit 50 Ω resistors on those pins for source impedance termination for high-speed digital signals, or higher value current-limiting resistors to drive LEDs. Seven pins on K1 are uncommitted, and they can be used to patch signals to and from the motherboard.

Functional test

Apply 5 V from a bench power supply to connector K3 to power the circuit. It is a good idea to set the current limit to 100 mA initially. On K3 +5 V is on pins 1 to 4, and ground is on pins 6 to 8. You should find 3.3 V at the output of the voltage regulator and on pins 9 to 12 of K3, and LED1 should light. You can check the clock signal on R25: the signal should be a 3.3 V squarewave with a frequency of 40 MHz. Use TP1 or TP2 as a ground point for the oscilloscope.

1, 2, 3, Quartus

How do we go about developing a program for the CPLD and downloading it? The dedicated tool, available for both Windows and Linux, for writing a program for Altera/Intel devices and for managing the download of data to the CPLD is called **Quartus Prime** [1]. For CPLDs in the MAX II series the free 'Lite' version can be used. The current stable version is 17.1. Linux aficionados will need to download the software packages in parts and then manually install all the files: how this is done is described in detail in [2]. Windows users have things a little bit easier: a download manager automatically loads and installs all the necessary files (as long as you click on the right button). We are still missing the connection between the computer and the CPLD board, a programming adapter called a 'ByteBlaster'. Sadly the original **ByteBlaster** is not exactly cheap at well over a hundred dollars, and in any case

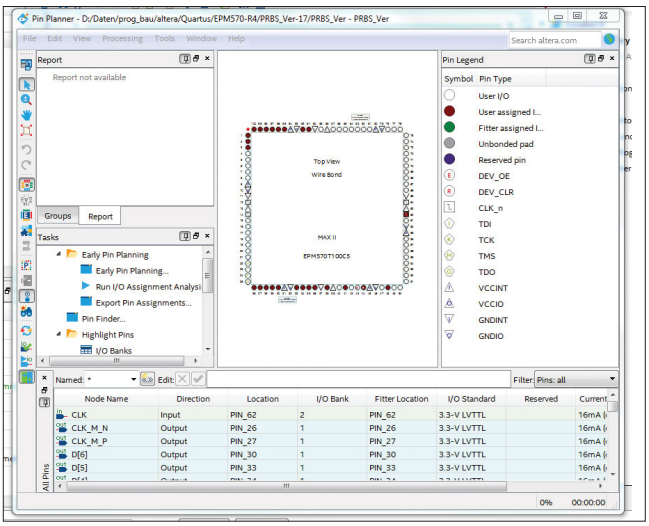


Figure 6. The Pin Planner tool is very convenient to use.

is only available on back order from suppliers such as Farnell [3]. However, the Chinese company Terasic [4] produces alternative devices, recommended by Altera, and a search of Ebay using the keywords 'USB Blaster' will turn up other Chinese clones for less than ten dollars!

The programming adapter is connected to the USB port on the computer and then you can launch Quartus. Select the 'Programmer' tab and verify that the adapter is recognized. If it is not, you will need to download the necessary driver manually. Then connect the programming adapter to the JTAG port on the CPLD board (K4) and turn on the power supply to the CPLD. Using 'Autodetect' will cause the Quartus software to analyse the chain of connected devices, which in this case just consists of our MAX II EPM570.

Now, of course, we need a program to represent the logic function that we want the CPLD to implement. We can choose between two popular hardware description languages **Verilog HDL** and **Altera HDL**: there are many introductions and tutorials on these languages to be found on the Internet to show how programs are written [5], [6]. Quartus includes its own editor for entering program code, and we will use that here.

The author has prepared a simple test project for demonstration purposes [7], comprising a ten-bit counter and a 2^7-1 PRBS (pseudo-random binary sequence) generator, in both languages. The two archive files PRBS_AHDL.zip and PRBS_Ver.zip are included in the project download and include all the files as created by the Quartus compilation process. The two archives should be unpacked in separate folders in a suitable place on your computer. Both examples were produced using Quartus 17.0; if you wish to use the newer version 17.1 then the design will need to be converted.

In Quartus, under File - Open Project, navigate to the directory you have chosen and double-click on the project file (for example PRBS_ver.qpf) to open it (see **Figure 4**). The design file, in this case PRBS_Ver, will now appear towards the upper left. A further double click will open the editor to show the code listing (**Fig-**

ure 5). Under 'Assignments' you can open the Pin Planner tool (**Figure 6**) to show the allocation of signals to pins and the characteristics of the pins; you can also make changes here. To recompile the design, under 'Processing' select 'Start Compilation'. The compilation process should complete without generating any error messages.

Finally we can download the .pof file into the chip using 'Programmer' in the 'Tools' menu (**Figure 7**). The USB-Blaster should have been detected by the system before compilation began; otherwise, it must be activated using 'Hardware Setup'. When you click on 'Auto Detect' the EPM570 should appear. Tick the 'Program' and 'Verify' boxes and then click on 'Start'. The progress bar towards the top right becomes more and more green and eventually reaches 100 %. Et voilà, your CPLD is programmed and starts carrying out the specified functions. **Figure 8** shows the output of the PRBS generator and **Figure 9** shows the outputs of the counter.

The .pof file, which is in the 'output_files' subdirectory, is the file which one would pass on to a contract manufacturer to enable them to program the chip on the board after assembly. The file is encrypted to an extent that prevents ordinary mortals from recovering the original design files from it.

For ease of reference we have produced an Excel spreadsheet [7] which gives the pinouts of connectors K1 and K2, which pins of the CPLD they are connected to, and, alongside, the signal names used in the design. This makes it easier to plan the connections to be used in a project and to enter the resulting data into Quartus' Pin Planner tool, and also simplifies the production of documentation. Which signals are found where can be determined from the Pin Planner tool or from the Excel file included in the project download. If no changes are made then a PRBS signal will be found on K1 while the outputs of the counter will be found on K2, dividing down the master clock. The .pin file in the 'output_files' subdirectory gives the exact assignment of pins to signals.

To start a new design in Quartus use 'Start New Project' and answer the questions that appear. Quartus will then set up a new project. Now launch the editor and enter the design. Then debugging can begin: keep recompiling the design until it builds without errors.

Quartus' Pin Planner tool will start with suggestions for which signals might be assigned to which pins. If you do not already have a printed circuit board layout then you can adopt these suggestions; otherwise you must reassign each signal to a new pin to match your existing board layout. The properties of the pins can also be changed at this point. Then the project can be rebuilt and the resulting .pof file finally downloaded into the chip as described above. ⏮

(160425)

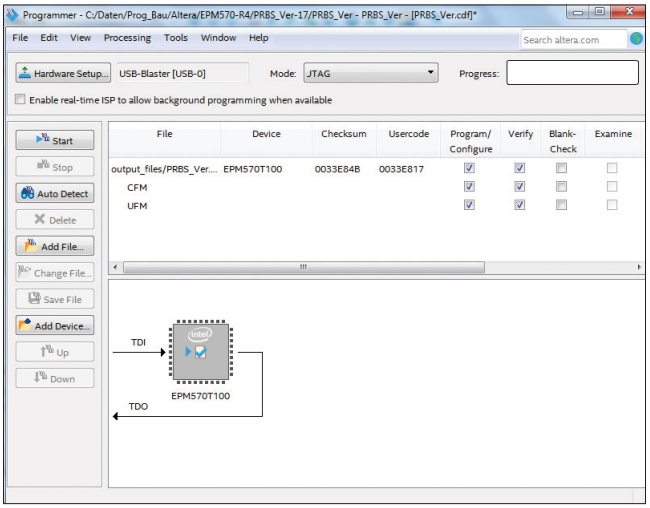


Figure 7. Downloading the program into the chip.

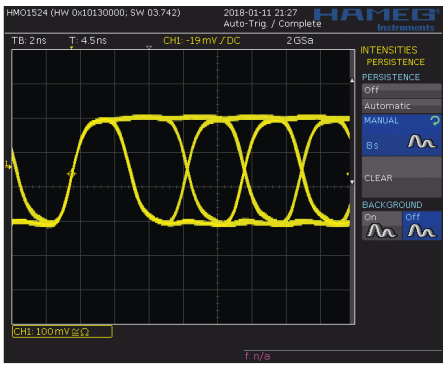


Figure 8. Output signal of the PRBS generator shown on the oscilloscope.

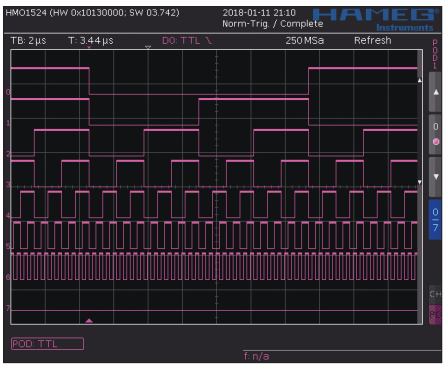


Figure 9. Output signals of the counter.

Web Links

[1] http://dl.altera.com/?edition=lite

[2] http://dl.altera.com/static/quick_start_guide/quick_start_guide_17.1_en.pdf

[3] http://uk.farnell.com/altera/pl-byteblaster2n/cable-programming-parallel-port/dp/1631973

[4] www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=74

[5] www.asic-world.com/verilog/veritut.html

[6] www.alterawiki.com/wiki/Altera_Hardware_Description_Language_%28AHDL%29_Language_Reference_Manual

[7] www.elektormagazine.com/160425

Elektor Labs Pipeline

Here is yet another small selection of projects posted at Elektor Labs. Some are more practical or useful than others; all kept their originators busy until they solved their problem or satisfied a desire.

Put it in a box!

Everybody welcomes a good idea. Here is one for those who construct small circuits on prototyping board and then let them lie around without any form of protection, gathering dust. Such contraptions usually end up dead because wires come loose or components get bent, creating short circuits or bad contacts. There is a simple solution to this problem: put the project in an enclosure. Nothing fancy or expensive — in many cases an old audiocassette case does the job. Transparent, easy to cut and open, they are perfect for the job. And they can be stacked, allowing your projects to be stored neatly in a drawer or cupboard.

https://goo.gl/uVVPUi

Build a microphone preamplifier

So last summer you spent your holidays somewhere in a hot and sunny place, where you enjoyed a hotel or beach resort with organized activities like morning aerobics in the swimming pool and loud parties at night, and where you karaoke'd your lungs out, and now you want to prepare yourself for the upcoming summer holidays? Then it is time to build this great all-analogue, all-through-hole microphone amplifier and impress your friends and family once again with your vocal capabilities.

https://goo.gl/UwzJG3

Make a POV fidget spinner

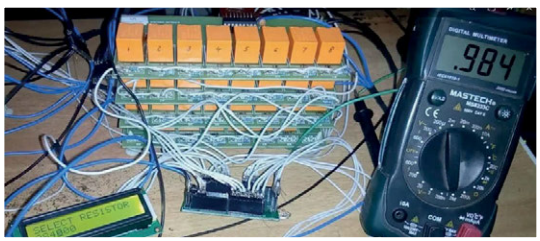
— "Hey guys, look what I made. I put a ball bearing in the centre of this flat multi-lobed structure so that it can spin along its axis. That's all it does, it just spins and I am going to sell billions of 'em."
— "Sure you are, dude. Beat it."
The rest is history and even at Elektor Labs we encounter fidget spinner projects. This particular project combines the fidget spinner with a persistence of vision (POV) LED display. Will it be ready before the hype is over? Or will it create its own hype? Decide for yourself.

https://goo.gl/QZYGf7

Build a resistor (or relay) box

At Elektor Labs we are confronted with all sorts of projects, from simple ideas to fully developed all-singing all-dancing multi-board constructions. Some projects are more useful or practical than others, but they all fill the need of at least the person who posted it. Here is one of those, a microcontroller controlled resistor decade box. According to the designer, it can produce any resistance between 0 Ω and 10 MΩ. If it goes all the way down to zero is questionable looking at the wiring involved; that it will help you consume your stock of relays, on the other hand, is certain. ⏮

(160620)



https://goo.gl/6MKVQY

The NXP Cup 2018

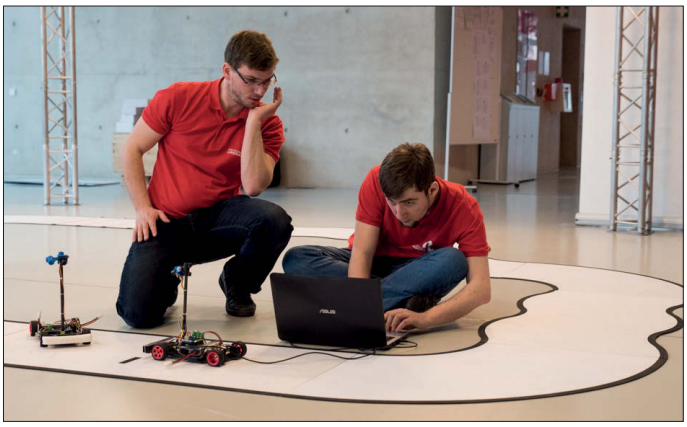
The Road to Erlangen

The NXP Cup is an automotive-based racing challenge that's open to highly motivated student teams seeking to gain experience and skills in the exciting world of automotive development. For the last seven years the NXP cup has been inspiring students to push their technology skills to the limit.

In 2017 more than 50 university teams across Europe, the Middle East and Africa (EMEA) signed up to participate in the 2018 NXP Cup and have been hard at work leveraging their technology to win. The goal is to pick up valuable skills in one of the hottest areas in technology: automotive development. We caught up with Iain Galloway of NXP to find out what the students have accomplished so far on the road to Erlangen.

Can you give us some background on the competition?

The NXP Cup started in 2003 in Korea with 80 teams. Since then the competition has expanded to China, India, Malaysia, Latin America, North America and most recent Europe in 2012. More than 500 schools and 15,000 students take part in the competition each year globally. The European finals will take place on 16-17 April in Erlangen, Germany with qualifying rounds in March.



How does the competition work?

The contestants place their vehicles on a black-and-white track. There are five qualifications throughout Europe. Each team can apply for one of these qualifications. During the qualification, there are three runs for each team. Only the time of the best run is retained. The fastest car that independently completes a circuit wins.

The top teams will be invited to the finals at the Fraunhofer Institute in Erlangen. The finals take two days. The first day is dedicated to training and mingling, as well as to get to know the sponsors and to discover opportunities within the industry. On Day 2 the final races take place. High level NXP representatives as well as camera teams and press are coming over.

The winning teams get great prices!

What have the teams accomplished to date?

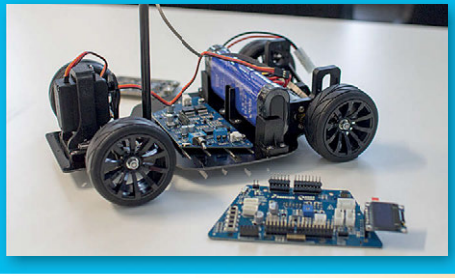
The teams signed up towards the end of last year, assembled their members and ordered their kits. The kits include the Alamak car kit, provided by NXP and its hardware partner Landzo motors, which is composed of a unibody chassis, motor control and power board and a single scan camera.

By this time, the teams have optimised their car, written algorithms and experimented with various configurations and

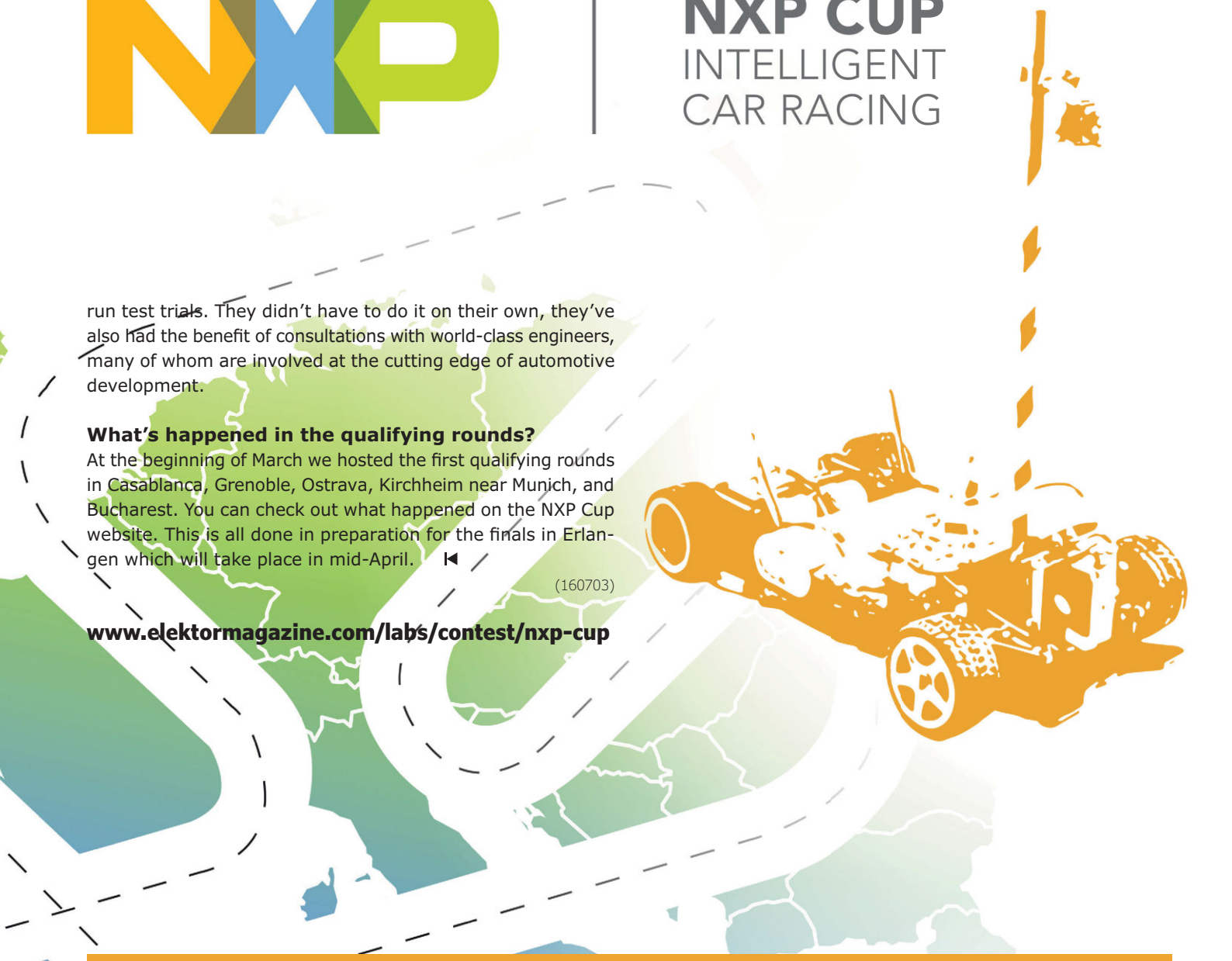
Landzo and Alamak

Landzo provides the car (the neutral platform for the challenges). They are providing both, the Chinese NXP Cup as well as the EMEA Cup with their car models. The 2017/18 season will run with the "Old-Model-C" car and the new Alamak model with more powerful motors, unibody chassis, a new enhanced motor control board and a new power board with preset inputs for encoders, multiple cameras and wireless Bluetooth.

www.landzo.com/index.php?route=product/product&product_id=95



NXP CUP
INTELLIGENT CAR RACING



run test trials. They didn't have to do it on their own, they've also had the benefit of consultations with world-class engineers, many of whom are involved at the cutting edge of automotive development.

What's happened in the qualifying rounds?

At the beginning of March we hosted the first qualifying rounds in Casablanca, Grenoble, Ostrava, Kirchheim near Munich, and Bucharest. You can check out what happened on the NXP Cup website. This is all done in preparation for the finals in Erlangen which will take place in mid-April.

(160703)

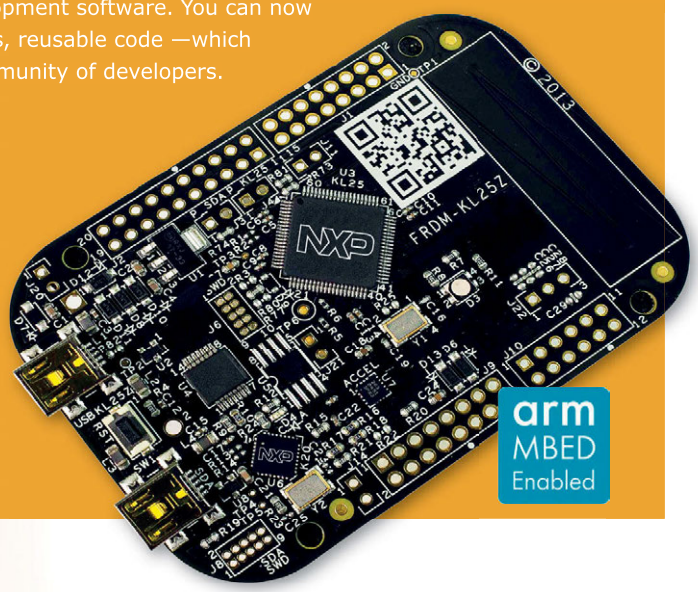
www.elektormagazine.com/labs/contest/nxp-cup

The Freedom KL25Z development platform

The Freedom KL25Z is an ultra-low-cost development platform for Kinetis L Series KL1x (KL14/15) and KL2x (KL24/25) MCUs built on Arm Cortex-M0+ processor. Features include easy access to MCU I/O, battery-ready, low-power operation, a standard-based form factor with expansion board options and a built-in debug interface for flash programming and run-control. The FRDM-KL25Z is supported by a range of NXP and third-party development software. You can now use mbed.org at no charge, with full access to the online SDK, tools, reusable code —which means no downloads, installations or licenses — and an active community of developers.

Features:

- MKL25Z128VLK4 MCU – 48 MHz, 128 KB flash, 16 KB SRAM, USB OTG (FS);
- Capacitive touch "slider", MMA8451Q accelerometer, tri-color LED;
- Easy access to MCU I/O;
- Sophisticated OpenSDA debug interface;
- Mass storage device flash programming interface (default) – no tool installation required to evaluate demo apps;
- P&E Multilink interface provides run-control debugging and compatibility with IDE tools;
- mbed enabled.



Struggling with LED Snake Lights

LED strips are flexible and functional but not entirely inconsequential

By **Dr. Thomas Scherer** (Germany)

The last issue of Elektor [1] covered EMC measurements taken with conventional LED lamp bulbs, revealing interference that can exasperate radio amateurs and affect radio reception in general. LED lamps for 230-V operation are one thing; LED strips are totally different, basically because they are (usually) powered by low voltages, right? True but they hide some sneaky problems up their sleeves.

Well over ten years ago, when the first white LEDs rated at 1 W and higher were not only available but also halfway affordable, I knocked up some home-made LED lamps for room lighting. For electronicists the advantages are particularly convincing: significantly better efficiency and longer working life than incandescent bulbs or indeed compact fluorescent lamps. They also survive switching on and off better and provide full brightness as soon as they come on. **Figure 1** shows a 'LEDified' lamp from the year 2006 — nicely analogue in terms of power supply, with the secondary winding of the transformer abused as a dropping resistor.

In the meantime, however, LED technology has moved on. Not all LED lights employ one or two couple power LEDs now; some are equipped with numerous little SMD LEDs, as demonstrated by the lamp in **Figure 2** (which, by the way, has served for ten years without complaint for staircase lighting). And by using multiple small SMD LEDs you can also create other products, namely by

chaining them on strips and in this way creating flexible lighting solutions in the truest sense of the word. The industry recognised this opportunity long ago and now you won't find any hardware store without at least one display offering LED strips of every kind. The Internet is also awash with them. The vital question is, what we need to consider when using LED strips in practice?

First purchases

Even tech-savvy people like me occasionally are caught out by electronic 'bargains'. In my case, it was one of the many Chinese websites, such as GearBest, that caught my eye when I was in fact looking for an ultra-low cost second smartphone. I spotted a 4-metre long LED strip in warm white, complete with plug-in AC adapter, which claimed low power consumption and would be sent to me across half the globe for a ridiculously low amount including postage. One click and three weeks later one of those typically grey plastic mailer bags dropped though my letterbox. Inside was a roll of 3528-format SMD LEDs (see **Figure 3**) for 12-V operation together with connector clips and a suitable power supply that could allegedly deliver up to 2 amps. Who'd have believed it?

My first act was now to connect these LEDs, while still on the roll, hooking up their leads to a laboratory power supply that was set to exactly 12 V. The roll lit up and the ammeter showed 1.2 A. So everything was in the safe zone. But what should I do with this new acquisition? I pondered a while and then I remembered that I had always wanted to have some indirect lighting in my study. And right on the wall, where my big PC monitor is. This would provide the advantage that no reflections would appear on its display, because in order to produce a crisper image, modern monitor screens are not as strongly anti-reflective as was previously the case.

Experience

No sooner said than done. Off I went to my local DIY centre and procured two U-profile channels of anodised aluminium, 12 mm tall and 2 m long. These suited my room (which is 4.10 m wide) exactly, without any need for shortening. I screwed both profiles to the wall without any gap between them, after which I was able to glue in the four metres of LEDs (LED strips are typically provided with self-adhesive tape on their rear side). The LEDs themselves would not be visible and reflections inside the U-profiles would (hopefully) camouflage the point-source nature of the lights. The only other work was to solder wires from one end to the power supply, with a cord switch in between. And that was it.

However, before executing my plan, I first unrolled the entire strip, as I wanted to know if all the LEDs were illuminated evenly. It would have been annoying to find any dim spots after completion. Removing everything off the wall just to replace some LEDs would have been a lot of effort. But another reason justified my caution: I noticed that the LEDs closest to the entry point shone a little brighter. Those at the far end were a bit darker. The difference was only faintly perceptible. However, the relationship of perceived brightness to actual light output is not linear but rather logarithmic [2], so the drop-off in brightness must already be significant if you can only just see it. And so it was in reality: at the feed point there were exactly 12.0 V, whilst at the other end of the power rails of the LED strip only 10.7 V could be measured. According to our friend Ohm, a mighty voltage drop of 1.3 V at 1.2 A indicates the power rails must have a resistance of at least 1.1 Ω. Not only is this inelegant, it reduces the efficiency unnecessarily. As you can see in the panel **Types of strips** in Figure 4a, in this case every three LEDs are connected to the 12-V power supply via a 150-Ω dropper resistor. Using 12.0 V around 2.7 V are lost across this resistor, giving rise to a current of 18 mA (and a consumption of around 56 mW per LED). However, at the 10.7 V mentioned I could measure a voltage drop of only 1.5 V. The LEDs at the end of the strip were passing only 10 mA and consequently consumed approximately 31 mW per LED.

The solution here was to divide the LED strips at the centre, gluing one of the two 2-m sections long into each of the two 2-m long aluminium profiles, feeding the two adjoining halves from the central point. **Figure 5** shows how the feed-in arrangements look. The result was an adequate voltage of 11.7 V at each of the outer ends. Which is fine. Because the LED strips looked too bright for me in fact, I connected a silicon diode in series, thus reducing the supply voltage to 11.3 V. Reduced voltage = reduced

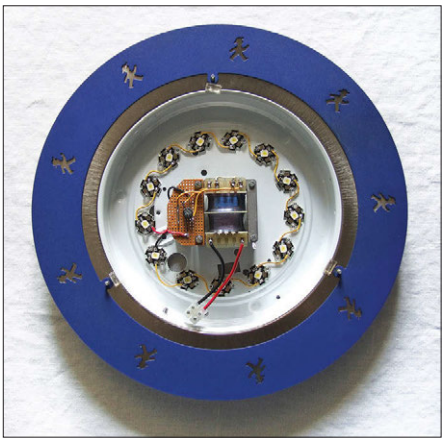


Figure 1. My first homebrew LED lamp using 12 white 1-watt LEDs, which were attached to the metal lampshade using heatsink paste.

Figure 2. Older type of 10-watt LED lamp with E27 fitting and SMD LEDs galore on the outer surface. Not to be grasped with bare hands when screwed into the fitting!

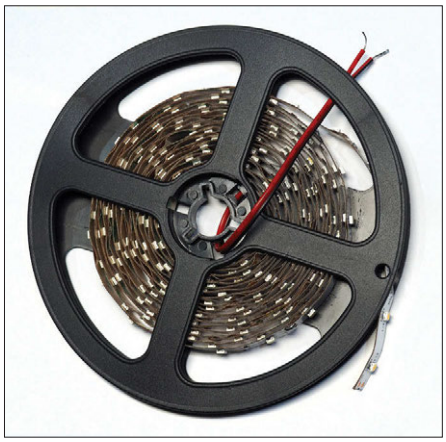


Figure 3. LED strips 4-m long, equipped with 3528-format SMD LEDs.

current = reduced power = increase useful life. I could now take extended pleasure from my homebrewed illumination.

Further insights

Should have, could have, might have... Unfortunately there were surprises in store for me. After revelling in my new indirect lighting every evening for a couple of weeks, I made a double-take one evening when entering my study. There before my eyes was a section an inch long that was dimmer than the rest. So I grabbed a chair, climbed up and took a closer look at the dark patch. In fact an LED had failed there. Curses! After such a short time!

For this reason the two LEDs that were connected in series with the deceased LED now lit up more brightly, but this was no consolation, because it affected the uniformity of my nice new lighting. So the dead LED was short-circuited. Time passed and the failed LED – I bet you have already guessed – proved to be not an isolated case. One time an LED would flicker for a while, then a few days later it would join its colleague in semiconductor heaven. On another occasion an LED that had gone dark for quite some time

Types of strips

LED strips come in a wide range of varieties. This article deals with only the low-voltage type that can be operated on 12 V or 5V. Practically all LED strips can be cut into sections of three LEDs each and are marked correspondingly. For 12-V operation we always have three LEDs wired in series with a dropping resistor. Their practical implementation varies only slightly. **Figure 4** shows circuits for two white and two RGB LED strips. In 4a we have an implementation with LEDs in the metric 3528 format. The LEDs are thus 3.5 mm long and 2.8 mm wide. Each triplet has a series resistor of 150 Ω. For better heat distribution, implementation variants using 5050 LEDs require two dropper resistors each of 30 Ω connected in series.

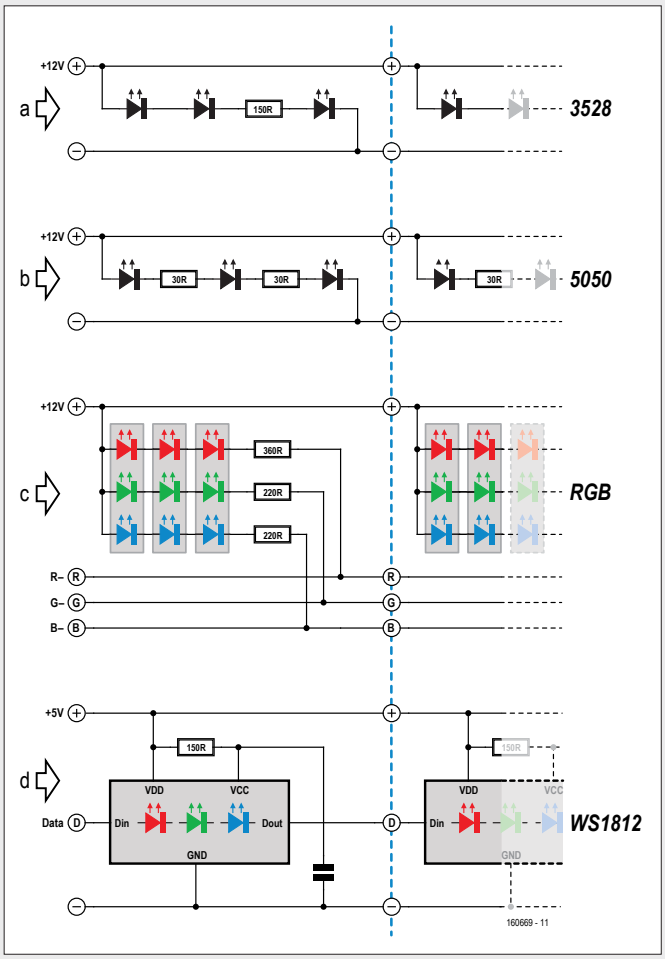


Figure 4. Schematic of various LED strips.
4d: Individually-controllable type WS2812 RGB LEDs.

Figure 9. Close-up of LED strips seen in Figure 4 connected in series.
9a: warm white 3528, 9b: warm white 5050 and 9c: RGB 5050.

Commercial RGB LED strips are fitted with 6-pin 5050 LEDs, on which the red, green and blue LEDs are again grouped in triads, each with a dropping resistor. As red LEDs exhibit a lower threshold voltage, the dropping resistor is increased here to 360 Ω, in order to reach the same current as for the green and blue LEDs.

The cropped images of **Figure 9** show the LED strips corresponding to Figure 4 in the raw. With one exception: Figure 4d shows the circuitry of the so-called 'intelligent' RGB-LEDs of the WS2812 pattern made by the Worldsemi company. Along with the three RGB LED chips they also have a controller on board. Because they conveniently they relay all incoming but redundant serial control data to their output, you can simply connect them in series and still address every LED in a daisy-chain or a strip individually. 256 brightness levels per colour are feasible, i.e. 224 colours in total. You might consider clock rate of 400 Hz per LED sub-optimal but if you do not hook up too many of these LEDs one after the other, the activation is virtually invisible to the human eye. This means you can realise fantastic running light (chase) and other dynamic colour effects limited only by your imagination. The LEDs are available either individually or on strips. The Internet is full of details on how to drive them.

Figure 5. The two 2-m long sections inside the aluminium U-channel are powered from the middle point.

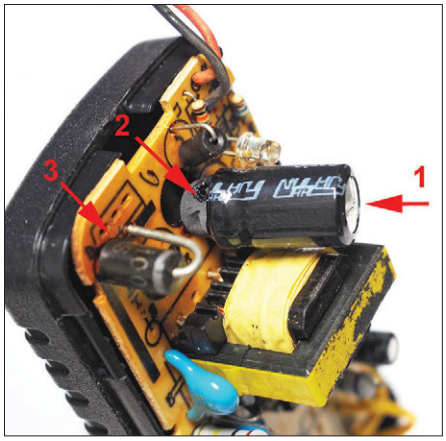


Figure 6. Opening up the AC power supply shows that the electrolytic has ruptured.

would suddenly light up again. So it must merely have been in a state of coma. But finally it could not suffer this earthly travail a moment longer and passed away just like its predecessors. In the course of six months and perhaps 1,000 hours of operation a total of eight LEDs failed. Yes, eight!

As an electronics engineer, I began to feel ashamed when I had a visitor one day who could not resist making a joke about this at my expense. So I wrote to the supplier and asked for one metre of this type of LED strips (plus an indication of the price). Simply buying a piece of any old strip or a pair of matching SMD LEDs to replace the defective ones would not have worked well. Not only were the LEDs on my strips of IES Class M quality but also of a colour temperature that was more than warm white — very, very warm and off-white. They were clearly yellowish and more likely to have a colour temperature of 2,200 K than the specified 2,700 K. Another type of LED would probably have replaced the divergent brightness with a divergent colour. Here, the principle of homeopathy does actually apply: replace like with like (*similia similibus curentur*).

Rather generously the Chinese supplier responded immediately that he would not only send me replacements but also at no charge. Two weeks later, I received not a 1-m LED strip but a complete 4-m roll, only without any power supply. Shortly after this I dismantled the aluminium profiles, marked the defective LEDs and desoldered them. Despite the cramped space in the aluminium profile, this worked quite well, using two soldering irons at once. Those who occasionally solder SMDs will soon gain some experience in ambidextrous soldering.
After changing out nine LEDs in all (the ninth one gave up the ghost while I was waiting for the replacements) and replacing the profiles on the wall, the world was put back to rights. Well, for the time being.

Worse to come

It seems that fate wanted to teach some more lessons. It goes without saying that more LEDs bit the dust in the months following the repair work. Among them were those that I had already replaced once. Remarkably all had gone short-cir-

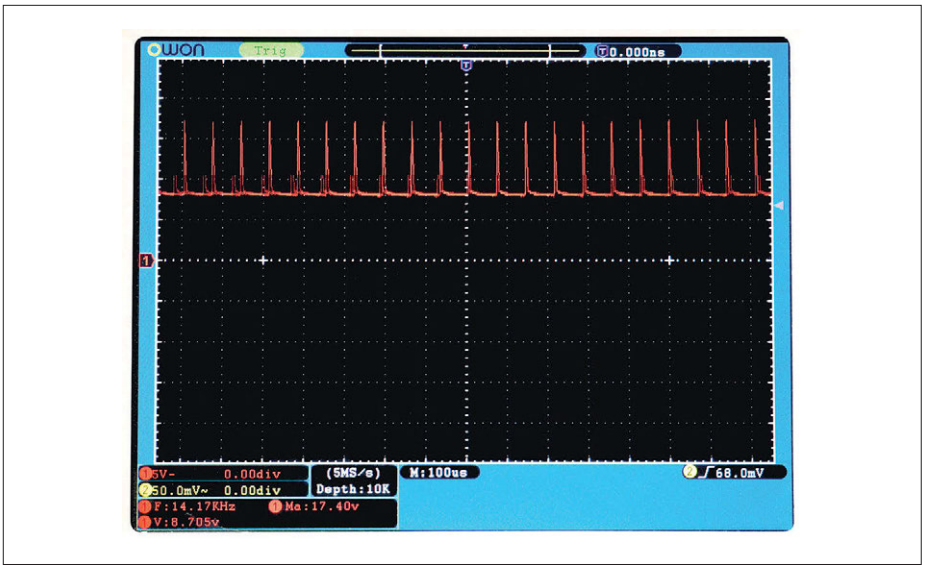


Figure 7. The oscillogram of the 12-V rail of the defective power supply indicates a hefty interference signal at around 15 kHz.

Figure 8. RGB LED strips from a hardware shop supplied with remote control.

Figure 10. A section of RGB LED strips, set to show 'white' at minimum brightness. The white has a definite blueish cast (very high colour temperature).

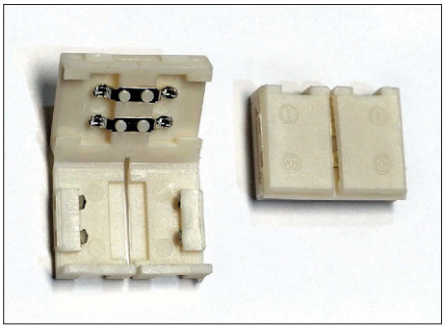


Figure 11. Clips like this let you extend LED strips without any soldering.

cuit; none of them were open-circuit. I had already proved to myself the old adage 'You get what you pay for' and had resigned myself to procure a new LED strip of better quality for more money soon, when a new phenomenon caught my attention.

The entire installation began to flicker. First imperceptibly. On one occasion the LED strips looked a bit darker, another time somewhat brighter, then flickered rapidly and irregularly. Both the degree and the frequency of incidence of this flickering increased over time. Interestingly the flickering sometimes also dropped off, with the light becoming more stable when dimmer. Eventually this all became too 'colourful' for me and I just had to get to the bottom of it. Failed LEDs could not be blamed, as an LED strip contained multiple series circuits, each made of three LEDs plus a dropping resistor in parallel. My multimeter did, however, indicate only 9.5 V on the power supply output under load. Consequently the power supply must have developed a fault.

With some poking and heaving with screwdrivers, the plastic case was removed to reveal the inner workings. **Figure 6** identifies the culprit and as you can see, the housing of the output electrolytic had boiled over (1). Beneath this the sealing compound had been burst out in addition (2). Last of all, you can see from the discoloration of the printed circuit board (3) that this spot became quite hot. Note this: in electronics, the electrolytic is always to blame and people don't call them 'smoothing bombs' for nothing!

Of the one-time 1,000 µF of this 'lytic only the smallest rudiments remained effective, if at all. The exploded capacitor was responsible not only for the non-uniform brightness but also for some seriously poor electromagnetic compatibility. How come? Of course, without filtering, the pulses of the switching power supply hit the output full on. Since the clock frequency is about 15 kHz in this case, you can see in the oscillogram of **Figure 7** beautifully steep-edged needle pulses of this frequency and an amplitude of a good 10 V peak-to-peak. And given that this ultra-wide signal — extending well into the radio-frequency range — was fed into the full length of the two 2-m long LED strips, I had created here a wideband jammer with a special dipole antenna. Although the LED strips are shielded on three sides by the aluminium profiles, I dispensed with determining the range of the interference and replaced the defective electrolytic capacitor immediately by an example having a low ESR.

Solution(s)...

This palaver verified another wise old saying: 'Buy cheap, buy twice'. So I bought a new LED strip that promised to be of higher quality. This time my choice fell on a 5 m strip (sold as spares!), fitted with the somewhat larger 5050-style LEDs. Their size suggested more efficiency per chip and consequently longer life. I was not to be disappointed either, as the (still affordable) cost of €15 (£13/$18) has paid off for a solid six months, with not a single failure so far. In addition, the 'warm white' designation is now also correct; 2,700 K is spot-on.

This new LED strip is significantly more current-hungry and for this (and other reasons) much brighter. Four metres of it gobble up a good 1.8 A at 12 V. In **Figure 4b** you can see that in this case the dropping resistor is composed of two resistors each of 30 Ω, probably to share the thermal load. Measurements showed that 1.5 V was dropped across each resistor. This indicates a current of 50 mA and thus 150 mW per LED – although the total length is slightly longer than with the old LED strips.

For safety's sake I obtained a new AC power supply that, according to the supplier, would now deliver 3 A. Following my experience with its predecessor, having some power in reserve struck me as no bad idea. Nevertheless I decided it was better not to rush things, especially as the new power supply lasted just two weeks before it also nuked its 'lytic. As a penance, I bought a third power pack. This time one for 4 A, which I opened up immediately and throttled back the output by modifying the voltage divider in the negative feedback section to 11 V. It is still working today.

More thoughts

On one of my forays at the local DIY centre I found myself unable to resist the flashing nonsense any more. Spontaneously I reached onto the shelf and grasped the blister pack seen in **Figure 8** with an RGB LED strips complete with power supply and remote control (!!!) in my hands. Amazing what you can get for a mere €20 (£18 / $25) in these parts.

In technology terms what you get with these strips are some 5050-format RGB LEDs with six pins, again in triads with three dropper resistors (see **Figure 4c**). The LED chips have separate cathodes and anodes, for which reason you can connect the individual colour LEDs independently and wire each colour in series. Protruding from the plug-in power supply were not only the connections to the LED strip but also a short cable with the IR receiver for the remote control signals.

So the PWM controller for driving the colour LEDs is located in the AC adapter case then. In the **Types of strips** panel you will find details on special strips using LEDs that are addressed individually and thus offer the highest level of lighting effects.

Figure 9 provides close-up photos of these three types of LEDs, corresponding to the circuits in Figure 4. The upper side of the RGB strips is covered in transparent and highly flexible plastic, which makes them unaffected by moisture. **Figure 10** shows a segment of RGB, illuminated at minimum brightness level to avoid dazzling image sensor in my camera. Generally all of the LED strips discussed so far can be separated at the boundary of triads, as can be seen in Figures 9 and 10. In this way they can be shortened to any extent you desire – and also connected together again. Suitable connecting clips (see **Figure 11**) are supplied normally; these can be used to join the conductive tracks of the strips together. Personally I do not recommend their use owing to possible resistance problems at the joints; instead I make the connection using blobs of solder. Also, extending the track longer than 4 m makes little sense, on account of the voltage drop. If longer strips are needed, you should provide multiple separate feed points along the strip.

But there are still other variants than those dealt with so far. Also widely available are strips using 5630-format LEDs, which with up to 500 mW per chip are very, very bright. Strips of this kind are always arranged so that you cannot look directly at the LEDs. Heat dissipation makes it advisable to fix them on metallic surfaces. For less powerful LED strips like mine (using 3538 and 5050 LEDs) wooden or even plastic surfaces are adequate, however.

Another word on quality. Philips was an early exponent of remotely controlled, multi-coloured LED illumination. Their well-known Hue range includes not only 230-V lamps but also low-voltage LED strips complete with matching control electronics. As regards the last-named, not only can you buy the brand-name product but you also have the opportunity to drive your indirect illumination using apps or Amazon's Alexa, etc. Quality comes at (high) price, though: for one single metre you'll be passing easily €20 (£18 / $25) across the shop counter — the control electronics cost extra. The starter outfit with 2-m strips and no 'bridge' (network integration) will make you around €60 (£54 / $75) the poorer. For the bridge alone you'll be forking out about €50 (£45 / $62). That said, there are also Hue strips with a special feature, namely the RGBW variant. The 'W' stands for additional white LEDs on the strip, producing also a stable white light without the conspicuous colour cast visible in Figure 10.

Branding techniques are a study in themselves. Besides Philips, other viable products have come onto the market recently, which if nothing else are more affordable. Worth looking at for instance are those from the LED manufacturer Cree or the Lightify series from Osram. ⏮

(160669)

Safety and 'static'

Safety restrictions apply with special LED strips, intended for direct connection to the Ac line supply effectively without AC adapters. With these you must pay attention to good insulation for metallic and other conductive substrates. It goes without saying that contact with LEDs and power tracks must be prevented constructively in every case.

Fingers must also be kept as far as possible from so-called 'electronic transformers' (**Figure 12**). Although they cost very little and provide plenty of power, they produce an unwanted AC voltage in the tens of kHz range at their output. Although LEDs are diodes and can 'rectify by themselves', they are not designed to do this and can tolerate only low reverse voltages. Not only do you put the LEDs at risk, with this kind of unrectified and unfiltered power supply you also create a high-grade jamming transmitter, which not only affects other radio signals but may also result in finding radio interference investigators on your doorstep and receiving a hefty fine.

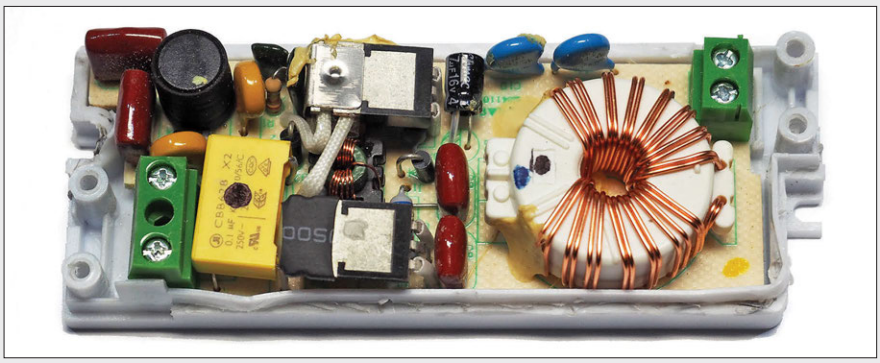


Figure 12. This kind of 'electronic transformer' is taboo for LED strips.

Web Links

[1] www.elektormagazine.com/160610

[2] https://en.wikipedia.org/wiki/Brightness

Scrolling

512 LEDs controlled

By **Pascal Rondane, Bastian Bouchardon** (Tours, France), and **Luc Lemmens** (Elektor Labs)

This project lets you scroll text on a bank of eight 8 × 8 LED matrix displays. It uses a Wi-Fi module of type ESP-12F (based on the ESP8266), programmable with the Arduino development environment. Thus, from a smartphone or any other device that uses Wi-Fi, it is possible to send to an ESP8266 web server the text to display, the scrolling speed and the brightness.

At the start, this display was connected via a network cable, not so practical. So Pascal decided to add on a module based on the ESP8266, using code written by Everett Robinson and improved by Pierre at the Fabriqueurs.com website [1]. He also integrated a web server, so that a client on a smartphone, a tablet, a PC, a Mac … can connect to the display.

LED Matrix

The thorniest component of this project is the most visible: the red LED matrix displays. A multitude of types are offered in online stores. Usually the main difference between these models is the orientations of the matrix of dots. It is only with difficulty that you can be certain of the type of module before you buy. When the Lab first mounted their displays on the Pascal's PCB, the text scrolled from top to bottom! Note that the circuit is designed to take the matrix displays marked 'FC-16'.

These matrix displays are designed to be mounted one next to the other; the data output of the first is adjacent to the data input of the second, and so on. The printed circuit board takes account of this logical layout, so the routing is simple and the traces short.

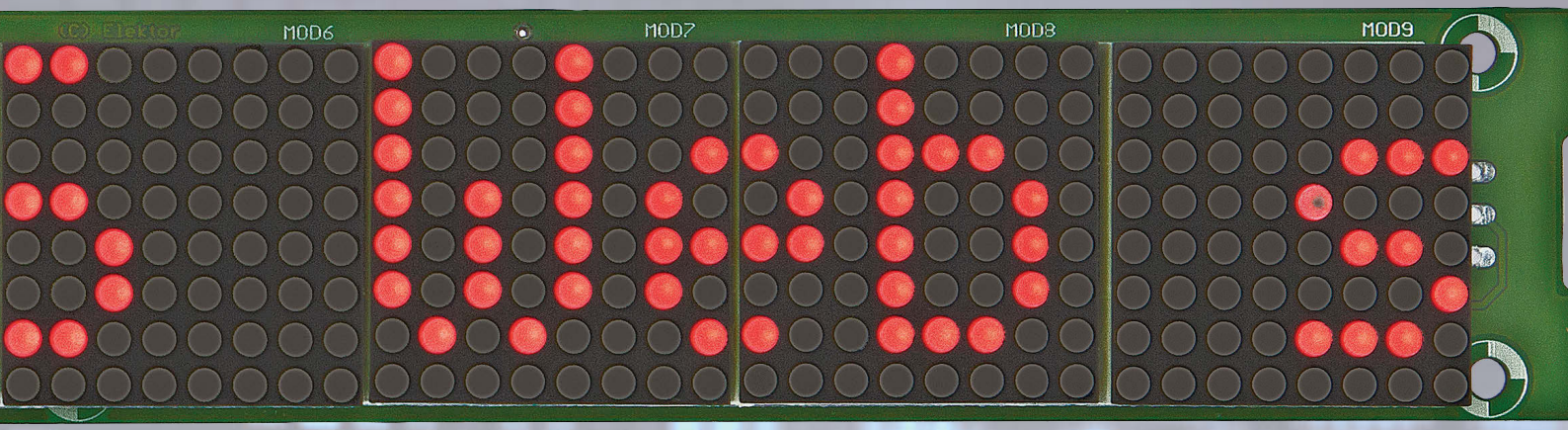
Parts

Figure 1 shows the schematic which breaks down into four blocks:

- 3.3-V power supply;
- ESP-12 Wi-Fi module;
- Eight LED matrix displays;
- Block outlined in blue dashed line: components not to be mounted, they are no longer used.

A 5-V, 2-A power supply should be connected at K2. Diode D1 protects against reverse polarity; fuse F1 blows in case of bad connections. The 5 V powers the matrix displays directly,

Message Display



over Wi-Fi via an ESP-12F

and regulator IC1 provides 3.3 V for the ESP-12 module. The two capacitors C7/C8 filter the ESP-12 supply to avoid unwanted resets and random functioning. Although the output pins of the ESP have 3.3 V and the matrix displays are powered at 5 V, the HIGH logic levels of GPIO pins 12/14/16 are sufficient to reliably drive the LED matrix displays. LED1 is lit when the ESP-12 module is powered.

Driver

Each LED matrix display contains a MAX 7219 SPI driver IC which takes care of the communication between the microcontroller and the 64 LEDs. On each of two connectors we find the following signals:

- +5 V;
- 0 V;
- Din and Dout: input and output of serial data;
- CS/LOAD: input for validation of data;
- CLK: clock input.

Of the sixteen bits of transmitted data, the eight most significant bits contain the address of the selected register and the eight least significant bits contain the value to transfer into the register. The data arrives on Din and the CLK signal (rising edge) transfers them into the shift register. After sixteen clock pulses, the first data is available on Dout and can be transmitted to the second MAX 7219 and so on for the rest of the matrices.

Software

The software calls on the `MD_MAX72xx` library. This has the advantage of supporting the various type of matrix LED with integrated MAX7219 driver. It also allows modification of the orientation of the displayed text.

This library is well documented. In the download [2] you can find many examples. There's even one which is almost exactly what we need, it has a different approach to the network connection. Effectively, the ESP8266

PROJECT INFORMATION

Wi-Fi
LED matrix
ESP-12

entry level
➡ intermediate level
expert level

3 hours approx.

PC,
soldering iron with fine bit

€50 /$60 / £45 approx.

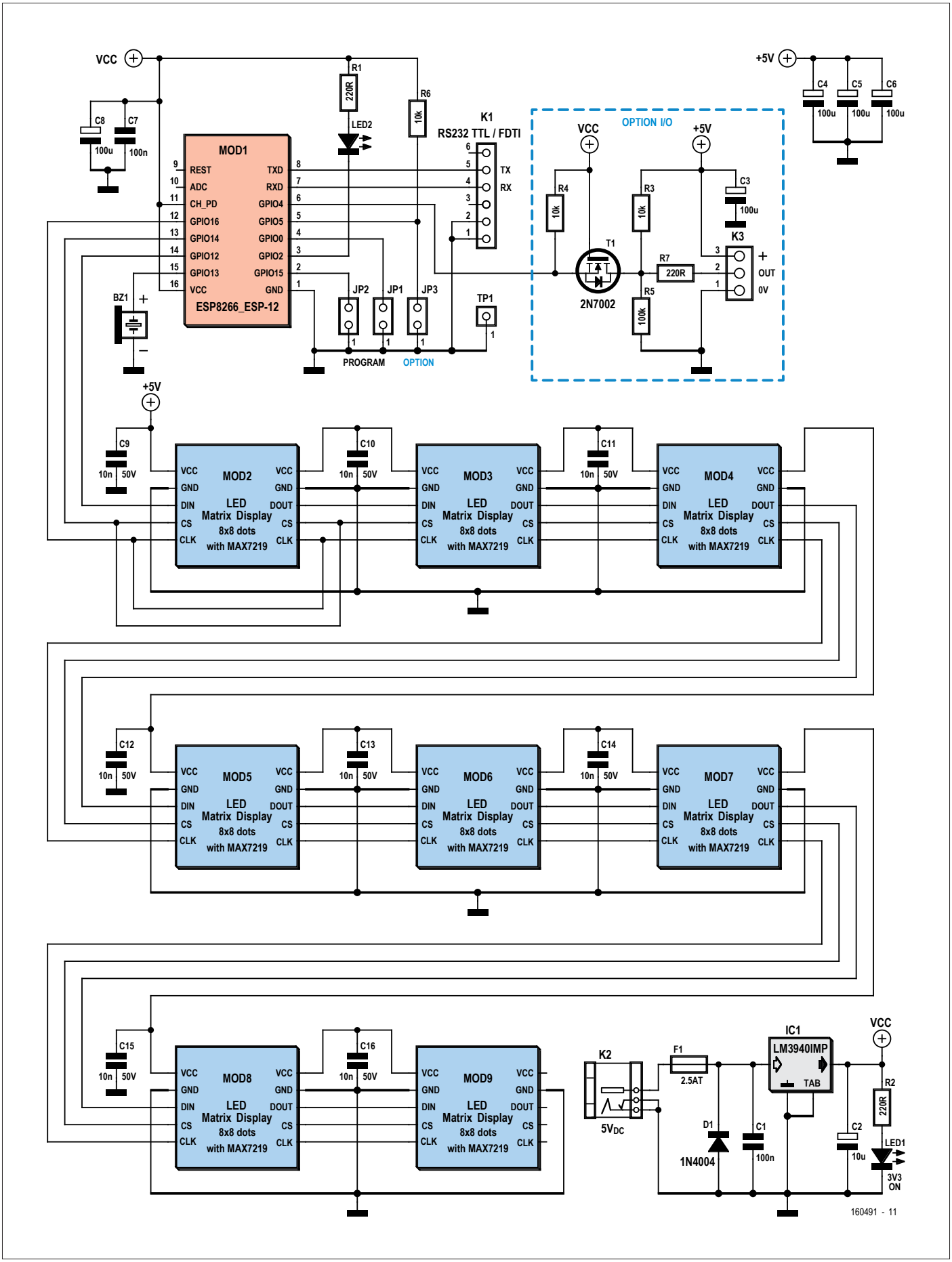


Figure 1. The circuit diagram shows the two principal functions: Wi-Fi communication with the ESP-12 and the display with the LED modules. The part enclosed with blue dashed line is no longer used.

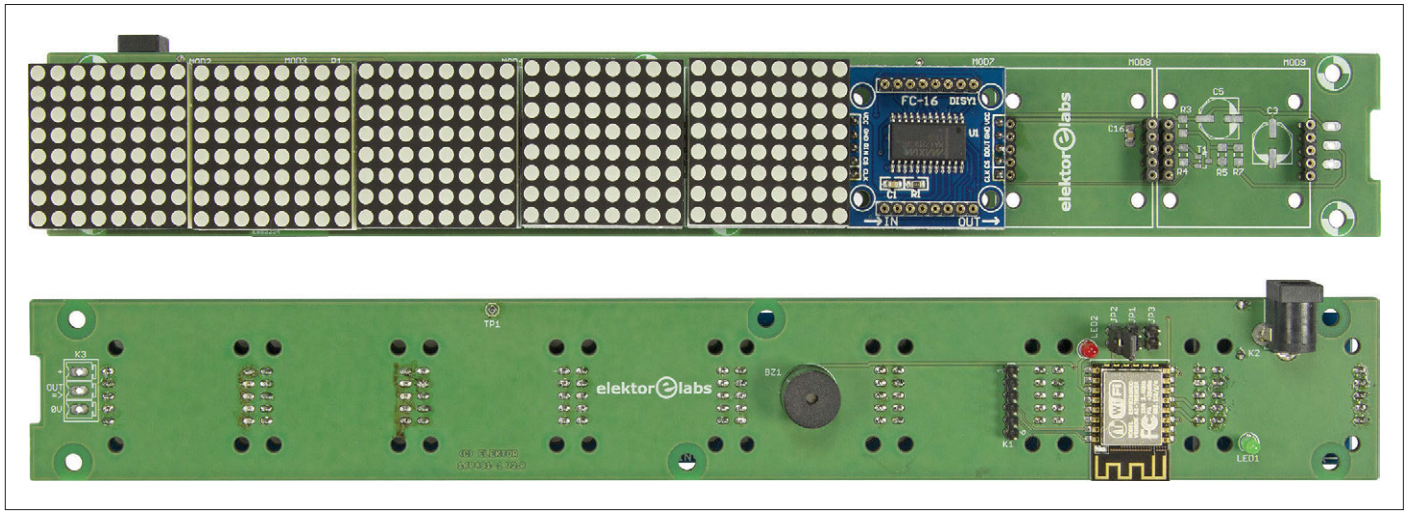


Figure 2. The different layers are clearly visible: the Elektor printed circuit board, a circuit board for the LED module and finally the displays. The Wi-Fi module and the indicators (buzzer and LED) are positioned at the back.

works in 'station' mode which connects to your Wi-Fi without a password, so any other network user can modify the message or the parameters. We prefer to use 'AP' mode (Access Point) where you need a password to access the display.
The font character data are stored in the file `fontdata.h`. Each character is defined by a line like this one:

```
5, 0x1c, 0x2a, 0x49, 0x49, 0x22,
// 150 - Euro sign
```

The comment contains the index of the character in the table (for information only), the sketch automatically calculates the index of the character to display. At the end of the file, we have added the accented characters.
The message to display from the web page is coded **ISO-8859-1**, the characters are thus converted into coded characters in **fontdata.h**. For ordinary characters, it's easy, as the ASCII code corresponds to the index in the table: line 64 for the character @ for example. For the other characters, you pass through a series of `switch case` instructions, it's not elegant but it does allow the addition of special characters. The maximum length of the character string is fixed by `const uint8_t MESG_SIZE = 255;`.
The sketch is well commented in French and English.

Wiring

The printed circuit board is double sided. Some of the tracks are very fine, so it might be better to call on a professional for the assembly. Even though there are some SMD components, it's possible to do the soldering with an iron. The through-hole components are placed from the side with the white lettering.
Start off by soldering the ESP-12 module, carefully align the module on its footprint and solder a corner. Check the alignment, correct it if necessary and solder the opposite corner, followed by the other fourteen pads. Continue with the other SMD components, then the through hole components and finally the LED matrix displays (**Figure 2**).
The 8× 8 LED matrix displays are delivered as a kit, with the SMD components already soldered to the printed circuit board. To connect the displays we leave aside the elbow connectors from the kit and mount the 5-way connectors onto the printed circuit board. The side with the larger pins is soldered onto the printed circuit board of the display, the other side with finer pins is inserted into the sockets on our main circuit board (**Figure 3**).
Each display is installed in the printed circuit board of its kit, on top of the SMD components. The direction of the display is not clearly shown. One of the sides of the case has a bump and the printed type number; this must be at the top of the printed circuit board. The assembled displays need to be mounted on bases to allow sufficient space for components beneath them (electrolytic capacitors).

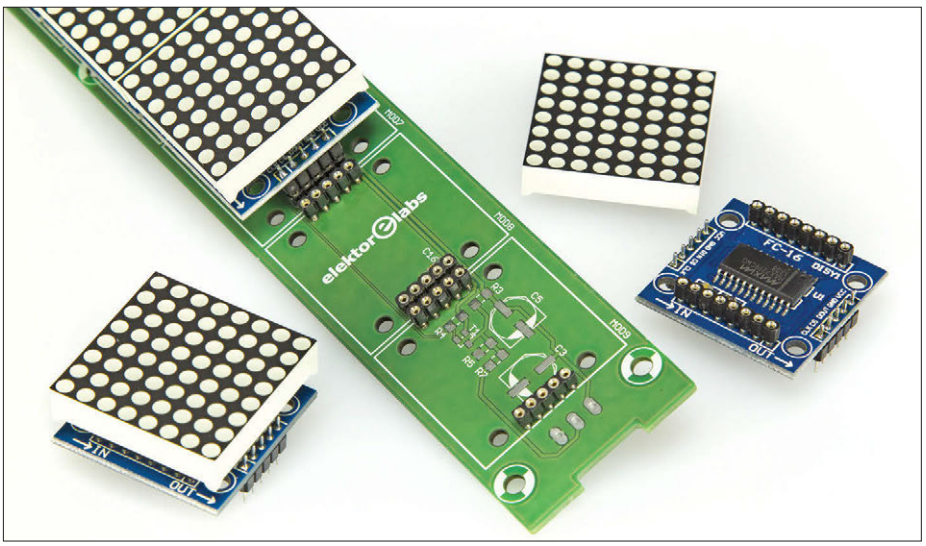


Figure 3. Detail of the stacking of the circuits.

Figure 4. Web page for configuration of the display.

It's possible to improve the rigidity with screws and spacers (if your modules have fixing points) or with a drop of silicone sealer.
You can power up your board before fitting the displays to check the presence of the 3.3 V and that LED1 lights.
Pascal bought some amber displays the same size as the red ones, but the connections are not quite the same and some signals are inverted… and to put the cherry on top, the display was rotated 90 degrees! Advice: if you change the display colour, do a quick check on the test bench.
It only remains to connect the displays to our printed circuit board. The assembly is now ready to be programmed.

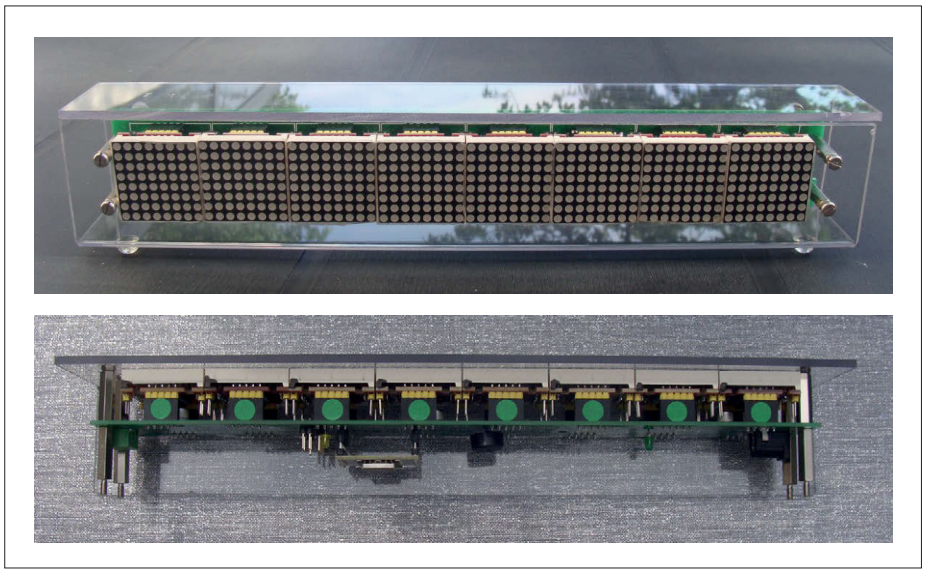


Figure 5. The polycarbonate case of Pascal's prototype.

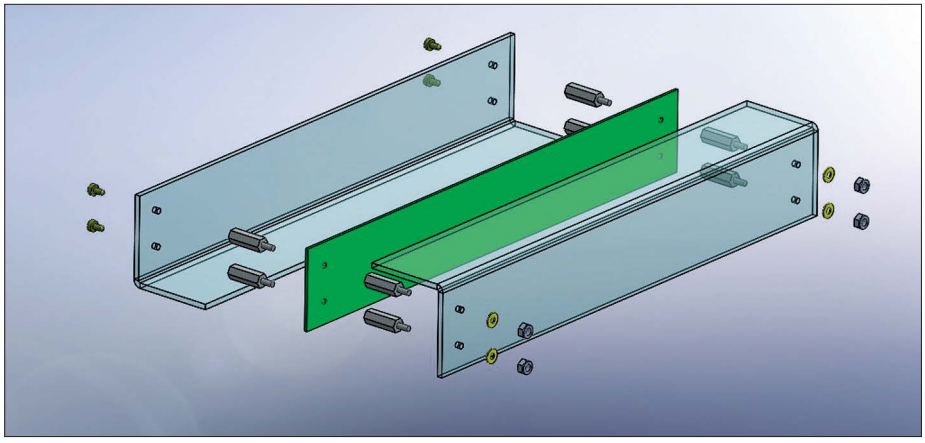


Figure 6. The author provided plans for the case.

Programming the ESP Module

In normal working, jumper JP1 must be out on switch-on. Jumper JP3 does not have any function at this stage. To load the software, insert the jumpers JP1 and JP2, install a 3.3-V USB-UART interface between K1 and your PC, then power the circuit up by connecting a 5 V power supply to K2. Note: the card is not powered via K1 to avoid an accidental short-circuit if K2 is also connected. If that happens, you risk overloading the programming interface.
The zip file available at [3] contains the complete file for the Arduino development environment, including the MD_MAX72xx library modified for this project:

- `MD_MAX72xx.h` : collection of definitions (from line 217, with `#define USE_PAROLA_HW 0`) to select the make and model of the matrix displays. All the definitions are set to '0', except for `#define USE_FC16_HW 1`.

- `MD_MAX72xx_lib.h` : from line 498, the following constants define the orientation of the display:

```
#if USE_FC16_HW
// tested MC 23 Feb 2015
//#pragma message «FC16 HW selected»
#define       HW_DIG_ROWS     1
///< MAX72xx digits are mapped to rows in on the matrix
#define       HW_REV_COLS     0
///< Normal orientation is col 0 on the right. Set to 1 if reversed
#define       HW_REV_ROWS     1
///< Normal orientation is row 0 at the top. Set to 1 if reversed
#endif
```

The file `fontdata.h` will be found in the same folder as the sketch `140491-11_V3.0.ino`. Open this sketch in the Arduino development environment.

The sketch creates a web page which can be opened on a mobile device to control the display. Line 41 lets you select the language of this page (French '0' or English '1'). You can easily add another language.
In the Tools menu, select `Generic ESP8266 module` as the target card, not forgetting to choose the correct COM port for your programming interface. Compile and upload the sketch to the ESP module, open jumpers JP1 and JP2, and reset power to the card: the text should start scrolling.

Modification of the displayed message from a screen

The ESP8266 appears as a Wi-Fi access point (AP) with the SSID: Scroll 8 Matrix

Elektor and password: Elektor2017. You can modify these parameters on lines 71 and 72 of the sketch. The library `ESP8266WiFi.h` controls the access point, `WiFiClient.h` and `ESP8266WebServer.h` control the web server.

Connect to the network with your mobile device, open your browser and go to the IP address 192.168.4.1. The ESP will display its web page (**Figure 4**). There you can set the text to display and set the scrolling speed and the brightness of the display. When you click on the Apply button, the new message and/or the new parameters will be sent to the display. The buzzer BZ1 beeps to confirm reception. The pixel test function tests the proper functioning of the displays.

When you change the message, if the web server is busy processing the request, there will be a latency time before the appearance of the new message and you will hear two beeps instead of one. You will have to figure out how to better manage the web server.

When power is cut to the display, the message is lost and the display will start up with the default values when powered up again.

COMPONENT LIST

Resistors

Default: 5%, 0.1W, 150V, SMD 0805
R1 = 220Ω
R2 = 330Ω
R3*,R4* = 10kΩ
R5* = 100kΩ
R6 = 10kΩ
R7* = 220Ω

Capacitors

C1,C7,C9–C16 = 100nF, 50V, X7R, SMD 0805
C2 = 10µF, 16V, Panasonic FK Series
C3* = 100µF, 16V, Panasonic FK Series
C4,C5,C6,C8 = 100µF, 16V, Panasonic FK Series

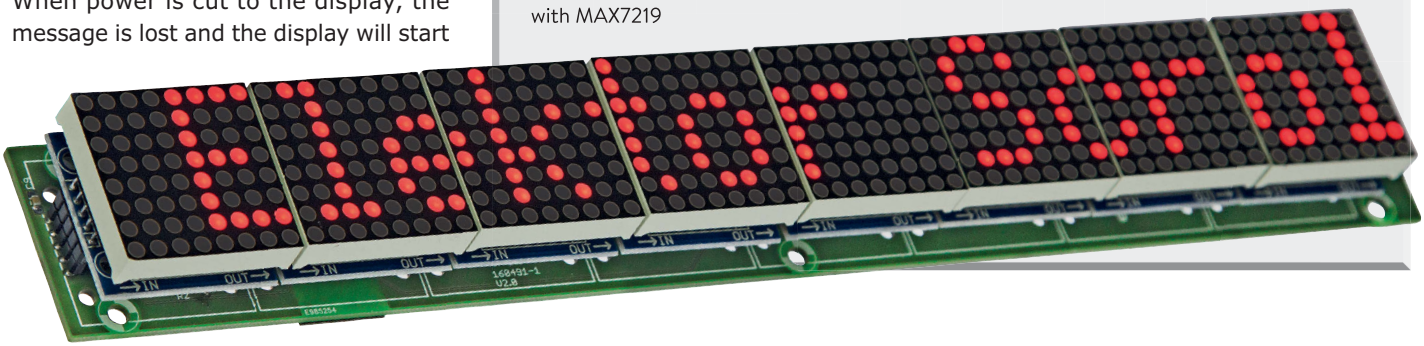
Semiconductors

D1 = 1N4007 (1000V, 1A)
IC1 = LM3940IMP-3.3 LDO regulator, SOT-223-3 case
LED1 = green, 3mm
LED2 = yellow, 3mm
MOD1 = ESP-12F Wi-Fi module
MOD2–MOD9 = FC-16 8× 8 LED matrix with MAX7219

Miscellaneous

BZ1 = buzzer, 5VDC, 12mm
F1 = 2A fuse, type 1206
JP1,JP2,JP3 = 2-way jumper pins, 0.1" pitch
K1 = 6-pin pinheader, 0.1" pitch
K2** = DC power connector, female, 2.35mm, 4A (MJ-180-PH)
K3* = PCB screw terminal block, 0.15" pitch, elbow
K3'* = plug-in terminal block, 0.15" pitch
MOD2–MOD9 = board-to-board connector, 5-way, 0.1" pitch, D01-9922046
MOD2–MOD9 = IC-type socket, 5-way, 0.1" pitch, D01-9972042
Power supply 5V/2A with 2.35 mm plug
Printed circuit board, Elektor Store ref. 160491-1

* Components not used in this project; reserved for other functions
** Attention: rated for 4A; not a standard connector!



Final touch

Pascal fitted his display (**Figure 5**) in a polycarbonate box, composed of 'L' shaped pieces, assembled with screws and spacers. To bend them, he had access to a bender. But it is possible to DIY bend bought Plexiglas™ (acrylic sheet) by heating it very gently with a heat gun and bending it a little bit at a time.

Start by assembling the display block and the front panel. Then line up the back panel before drilling to ensure that the markings are in the right places. You sometimes have to cheat a bit on the drilling. You can download the supplied mechanical plans at [3] (**Figure 6**). ⏮

Pascal thanks Jacques and the makers David, Vincent, Jean Louis and Pierre.

(160491)

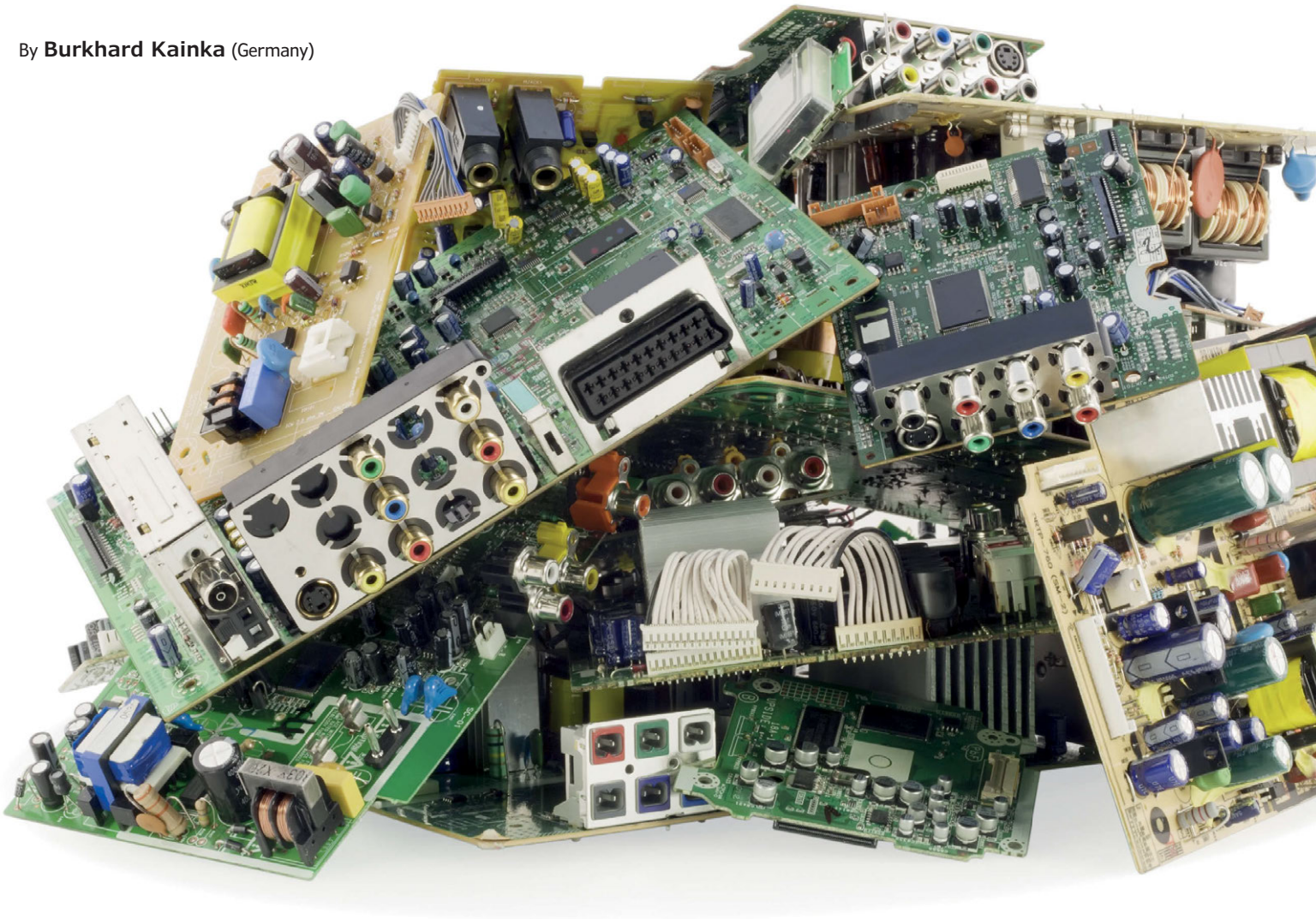
Web Links

[1] Fabriqueurs Project (in French): www.fabriqueurs.com/arduino-et-matrices-de-leds-un-afficheur-pilote-depuis-internet-v2
[2] LED matrix display library: github.com/MajicDesigns/MD_MAX72XX, majicdesigns.github.io/MD_MAX72XX/index.html
[3] Support & resource page for article: www.elektormagazine.com/160491
[4] Labs page for project:www.elektormagazine.com/labs/scrolling-text-display-160491

EMC Limit Values and CE Declaration

Simplified measurements for private individuals and small companies

By **Burkhard Kainka** (Germany)



Electronics enthusiasts keen on marketing their wonderful prototypes are faced with a myriad of regulations. Two of the biggest hurdles are the CE mark and EMC testing. However, they can be mastered.

If you basically do electronics as a pastime, you are usually under the radar of the authorities. Only really big mistakes can have painful consequences. If your product unintentionally radiates a considerable amount of RF energy, you can expect to receive unpleasant questions. And if you build something that hurts someone, you can face criminal charges.

Statutory requirements

Statutory requirements apply to companies of all sizes, and there are more and more of them all the time. Here is a brief summary of the situation in Europe, and specifically Germany, with no claim to completeness.

Every product that is publicly marketed in Europe must have a CE mark as evidence of the manufacturer's declaration that it conforms to all relevant regulations [1]. For this there must be a formal declaration stating which regulations those are. Most of the requirements relate to electromagnetic interference and passive noise immunity. Now there is also RoHS conformity (including lead-free solder, etc.), as well as general requirements regarding the materials used in the product (REACH, the

European chemical regulations), which are aimed at avoiding toxic substances. Many other standards can also be relevant, depending on the application. The manufacturer must be able to present this CE declaration upon request. There must also be a test report containing the measurement data used to verify compliance with the requirements.

When you see all this in concentrated form for the first time, you may be tempted to give up — and there are also many additional minor issues. Where can you find out which standard is relevant for your product? And where can you read all that? On the one hand there are the European standards, which you can find on the Internet. They are implemented as national standards, such as the DIN standards in Germany [2] or the BS standards in Great Britain. These standards are not freely available; you have to buy them — and they aren't cheap. Occasionally you might know someone who knows someone else who can loan you a copy of a relevant standard.

And then there is the electronic waste regulation WEEE. If you market products, you must also look after disposing of them so they don't end up in a landfill. A refuse bin symbol with a red line through it must be printed on the PCB or the product to indicate that it must be turned in to a public take-back point and that the manufacturer is participating in this take-back system. The details of all this have been discussed on the Elektor Forum site (in German) [3]. In any case, the whole thing involves significant costs and considerable red tape. Is that all? No — there is also a similar system for all types of batteries, and a system for packaging materials. And you always have to be on the lookout for new regulations that may affect you.

To mark or not to mark

A logical question here is who can or must comply with all this. If you want to sell a transistor, you don't have to stick a CE label on it because a transistor is not considered to be equipment. If you build a small circuit board with a voltage converter, it could be classified as a component or as a finished product. As you can see, there is a grey area.

On the back of a Raspberry Pi board (**Figure 1**) you see a CE mark, an FCC mark (the US equivalent of the CE mark), and a refuse bin symbol with a line through it. On an Arduino Uno board (**Figure 2**) there is a CE mark and an FCC mark but no refuse bin symbol; in its place there is a notice regarding RoHS conformity. Many microcontroller evaluation kits from large semiconductor companies do not have any marks at all. They probably rationalise this by saying that these products are not intended for the general public, but instead only for other companies. That's another grey area.

What would much smaller companies have to say about that? If you have invented a very specific measuring device for use by electronic hobbyists and want to somehow market it, and you have no idea if you will ever sell more than 50 of them, do you have to put yourself through all that stress? The official answer is "Yes". Or suppose you have designed an experimental tube radio and want to sell it as a kit via eBay. Some would simply say that it is not a finished product, just a set

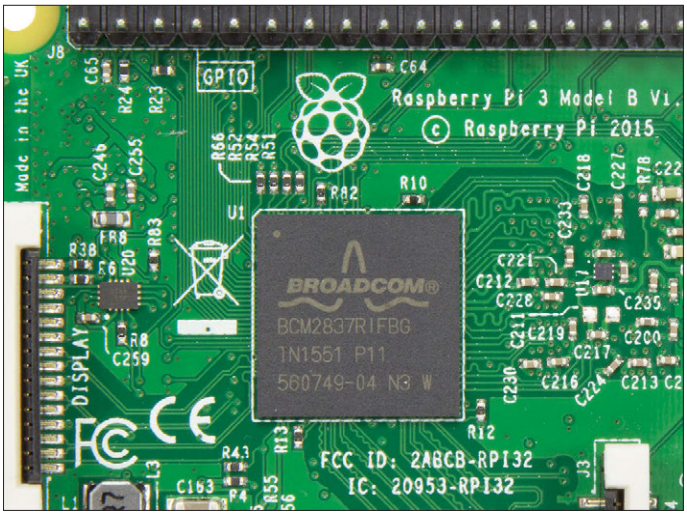


Figure 1. Markings on the Raspberry Pi.

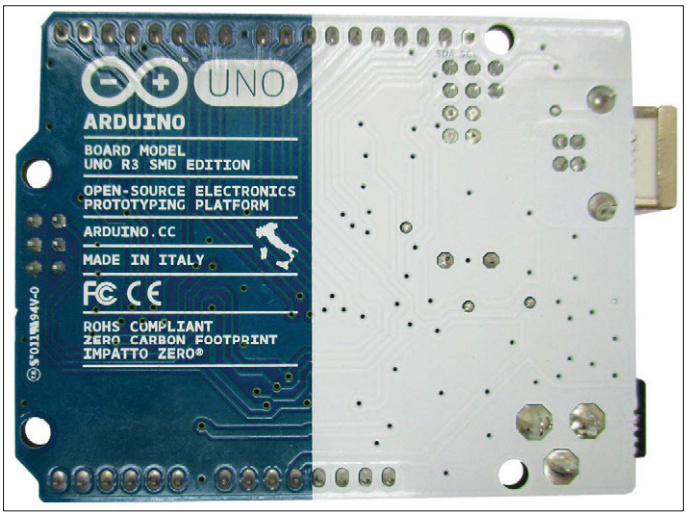


Figure 2. Markings on the Arduino Uno.

of components, because the customer has to put everything together. However, it's very unlikely that the authorities would see it that way.

Another question is what is the worst that can happen to you if the authorities are alerted and decide that you are not complying with the regulations? In that case they will usually prohibit you from continuing to sell the product. Then you would either have to comply with the requirements somehow or be stuck with a bunch of unsold products. That is not pleasant if you have made a large number of them, and it can also lead to a recall campaign. However, a start-up company just getting off the ground might be tempted to take the risk. In fact the chance of getting into trouble is lower with small product volumes.

In the USA the situation is even more difficult for small companies. Doing an FCC certification yourself is not possible; it has to be done by an authorised body and it is expensive. Many small companies in the maker community cannot afford that and simply ignore it. The issue is collectively swept under the

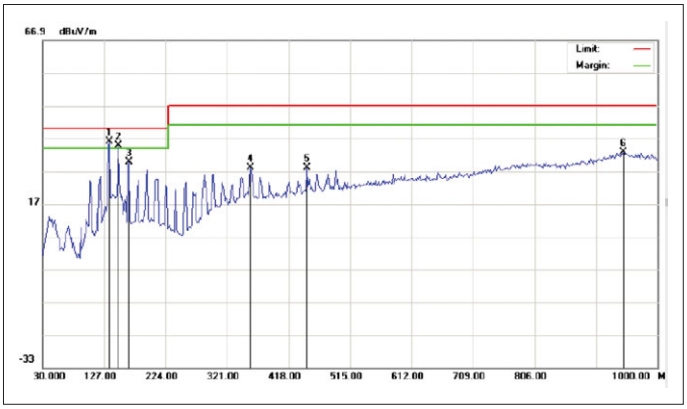


Figure 3. EMC measurement of a microcontroller.

carpet and kept quiet. Everybody hopes that as long as nothing bad happens, nobody will talk about it.

Contact with the authorities

What everyone fears actually happened to me. I received a complaint from the Bundesnetzagentur (the German regulatory body for telecommunications and other public services) regarding one of my products. Their staff had taken several products off the shelf for routine checking to see if they complied with the regulations, and they weren't happy with one of the kits. Fortunately it did not involve a violation of a limit value, but I had failed to observe an important directive. The product was a crystal oscillator with several interchangeable crystals, intended to allow simple short-wave radios to receive DRM signals. Purely as an aside, I also showed how to inductively couple a 2-MHz AM signal to a radio. I thought that with a range of less than two metres, interference would not be a problem, so there was nothing to worry about. But the authorities had a different opinion. In their view, AM at 2 MHz is not allowed. Other frequency bands have been allocated for that. We had to go cap in hand to the authorities and jointly consider how to resolve this. The end result was a sticker with a printed notice and a change to the next edition of the manual. All in all, that was a relatively good outcome. However, another very important point was also raised. The people at the authority said to us: "Don't think we haven't noticed all the other products being sold without CE marks. That's not allowed." In other words, kits and development boards should also have CE marks. And fully assembled products or type tests are essential for verification, as described in the guidelines.

We then asked how that should work, because verification by an authorised body is not affordable for small product volumes. Then we discussed a couple of cases as examples. With that it became clear that self-declaration is certainly possible, and in most cases we could do the assessment under our own responsibility. Once a CE mark is there, nobody takes a closer look. Also, the declaration does not have to be as elaborate as is often the case in some areas. And I could send them an initial CE self-declaration for checking. That was adequate, and it could serve as a template for other products.

EMC limit values

One of the key aspects of CE declarations for electronic products is the EMC limit values. Manufacturers must ensure that their products do not cause excessive radio frequency interference, which means they must comply with statutory limit values. The usual measurements for this are often complicated and expensive. Many developers have an uneasy feeling because they are not able to properly estimate whether their products will comply with the requirements.

For many electronic and microcontroller products, one of the relevant standards is EN 55022 [4]: "Information technology equipment — Radio disturbance characteristics — Limits and methods of measurement". An interference signal in the range from 30 MHz to 300 MHz may not exceed 30 dBµV/m at a distance of 3 metres; in the range from 300 MHz to 1 GHz the limit is 37 dBµV/m. Interestingly enough, nothing is specified for the frequency range below 30 MHz — as though the traditional shortwave bands no longer matter. However, the lower frequencies certainly do matter if anything manages to penetrate into the power grid.

If you have testing performed by an authorised body with suitable equipment, you typically receive test results like the chart in **Figure 3**. There you can see the usual picket fence of harmonics of the CPU clock frequency emitted at different signal levels. The test report notes which peaks were found at which frequencies and the corresponding signal levels. The measurement is made in a shielded chamber, so all the signals must originate from the device under test.

To be able to correctly assess the situation myself, a few years ago I bought a spectrum analyser with a measuring range up to 1 GHz. The only other thing I actually need is a fully shielded test chamber. However, there is a simpler alternative. I take a very small antenna in the form of a wire loop and hold it very close to the device under test. On the spectrum analyser display I see all sorts of known signals, such as FM broadcast signals in the VHF band, DAB, DVBT and so on. Between them I also see the signals that actually matter to me: the ones that become stronger when the probe antenna gets closer to the device under test. By comparing them to the levels of the known signals, I can estimate whether the interference signals generated by my device exceed the limit values.

Of course, this all depends very much on the antenna you use. But roughly speaking you can say that if a dipole tuned to 100 MHz produces a 1-µV signal with a 50-Ω load, the electric field strength is about 3 to 4 µV/m. In many cases all local FM transmitters will have very similar signal levels, which you can note for the location concerned. They are typically well above the EMC limit levels, as otherwise noise-free reception would not be possible. For example, if the signal level of an FM transmitter at 100 MHz is 1 mV on a 50-Ω dipole, that is 60 dBµV, which corresponds to a field strength of about 70 dBµV/m.

Those signals are my reference levels. Then I can say that my interference signals must be a certain number of dB below the reference signals. If I see that the interference signals at a distance of 50 cm are sufficiently weak and they are not even detectable at a distance of 3 m, then I can be reasonably certain that my device complies with the limit values. And in the vast majority of cases, relatively small devices easily comply with the requirements.

An AVR microcontroller such as the one on the Arduino Uno board, makes it easier to build devices with low noise emissions. The RAM and ROM are integrated into the chip, so there are no external lines with fast signals. The V_{cc} and GND pins are next to each other. If you put a 0.1-µF ceramic capacitor right

next to these pins, the supply lines will be virtually noise-free. With the right fuse settings, the crystal oscillator operates at a very modest amplitude of around 1 V_{pp} instead of maximum amplitude, and what's more, the waveform is virtually sinusoidal. That keeps the harmonic levels low. The only other thing you have to consider is the signals on the port pins. There you have watch out if relatively high frequencies are present and a PCB track or wire could start acting like an antenna.
A continuous ground plane on the back of the PCB also minimises interference. The Elektor SDR shield is a good example of how well that works. There you might expect that the Arduino would cause a lot of interference to shortwave reception. In fact it doesn't, because the Arduino Uno and the SDR shield both have ground planes and tight decoupling.

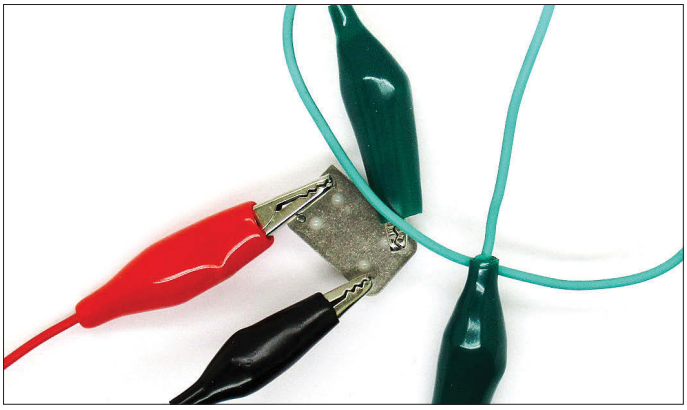


Figure 4. Here undesirable emissions are observed.

EMC estimation with simple resources

Sometimes you can draw conclusions even without a spectrum analyser. What matters in most cases is the harmonics of the clock frequency. With a microcontroller clocked at 16 MHz, you have to measure the harmonics at 32 MHz, 48 MHz, and so on. At 96 MHz there might be a signal in the FM broadcast band. With a signal level of 30 dBµV/m at a distance of 3 metres, you have a strong signal that distinctly suppresses the noise. However, in most cases this harmonic can only be seen very close to the microcontroller and is not detectable at a distance of 3 metres. In that case you can at least say that this harmonic is clean.
Some modern radios display the antenna voltage in units of dBµV. That includes the Elektor DSP Radio with the SI4735. Using a perfectly ordinary rod antenna, you can make measurements with that sort of radio that are comparable to measurements made with a spectrum analyser. For example, I measured a signal level of 60 dBµV from a local FM transmitter at 99.2 MHz. I could also receive a signal from the Arduino at 96 MHz. For that I had to bring the antenna to within 20 cm from the board to clearly suppress the noise floor. The indicated antenna voltage was 20 dBµV. That is roughly equivalent to the limit value of 30 dBµV/m. At a distance of 50 cm I was no longer able to determine whether the Arduino was on or off. In other words, the emitted interference was already below the limit value at about 20 cm, compared to the specified distance of 3 m. With that sort of result, it is unlikely that any of the other harmonics will exceed the limit value.
For comparison purposes, you can also occasionally do something intentionally wrong just to get a feel for the magnitudes involved. For example, I took a packaged 10-MHz crystal oscillator, connected a 30-cm wire to its output, and simply laid it on the bench (**Figure 4**). The signal levels were distinctly higher than the limit values. The steep signal edges generate lots of harmonics, and the wire is a suitable antenna. Signals were visible every 10 MHz, with the odd harmonics stronger than the even. The 90-MHz harmonic was certainly there, so I could check it with an FM radio. The range was amazingly large. An FM radio showed 40 dBµV at 3 m distance for the 90-MHz harmonic, which is much too high.
That meant I had to switch it off right away, because I could not know which of the many harmonics was already interfering with a radio service. In that situation the least interference occurs at the fundamental frequency, since the wire would have to be 7.5 m long to act as a quarter-wave antenna at 10 MHz. However, there is a good chance that one of the many harmonics will be radiated especially well because it matches the resonant frequency of the antenna.
This experiment shows that the signals usually present on typical circuit boards can certainly generate interference signals above the limit values. You might think that a port pin cannot output much power, but that is easy to underestimate. For comparison, radio amateurs have what they call the 'whisper net' (WSPR), which they use to see how much power is necessary to reach a given range. A Raspberry Pi with suitable software is all you need for a transmitter. The RF signal comes directly from a port pin. After that, you only need a good low-pass filter to get 10 mW at the antenna without any amplifier. In the shortwave bands, that is enough to reach everywhere in Europe. A range of 2,000 km is easily possible with just 10 mW. This also means that an unintentionally radiated high-frequency signal can easily reach the International Space Station at a height of 400 km.

Summary

If you develop a feel for the potential risks and take known precautions seriously, such as ground planes and tight decoupling, you should be on the safe side. Then a few simple measurements can confirm what you already know: your device complies with the limit values. It often helps to have the radio on while you're working. Sometimes you run into problematic signals, and then you hear interference on the radio. That way you never forget what matters. ⏮

(160435-I)

Web Links

[1] Searching for directives: https://ec.europa.eu/commission/index_en

[2] Overview of available standards: www.beuth.de/de/themenseiten/ce-kennzeichnung

[3] WEEE discussion: http://forum.elektor.com/viewforum.php?f=167782

[4] A bit outdated, but readily available: http://cq-cq.eu/EN55022-2006.pdf

[5] Elektor DSP Radio: www.elektormagazine.com/100126

HOMELAB PROJECT

Ltank Lego Robot

High-tech meets toytech

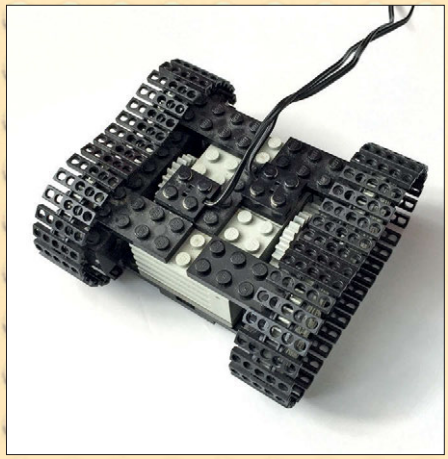


Figure 1. Ancestor-wired vehicle.

Figure 2. Lego smartphone robot platform.

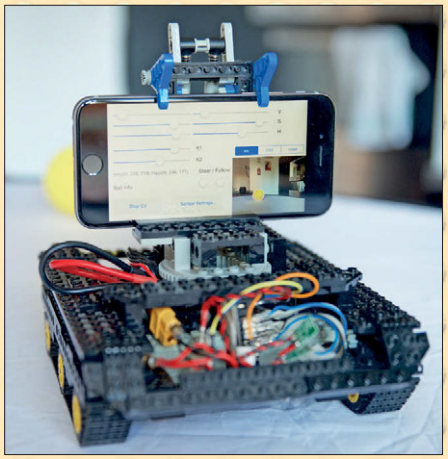


Figure 3. Final form of the robot.

This little Lego™ robot likes to play ball. It has a Smartphone for a brain, the reflexes of an ARM Cortex-M0 processor with integrated Bluetooth, and the strength of four classic Lego motors powered by a LiPo battery.

By **Dömötör Gulyas** (Germany)

A little while ago, a friend and I set out to design an autonomous robot that could detect and follow a tennis ball for an assignment while studying for our Master's degrees. We quickly agreed to use a smartphone (with a decent camera and plenty of processing power) running the OpenCV [1] library's rich algorithm toolset to locate the ball visually, but what we also needed was a capable mechanical platform to be able to chase the ball. The phone, weighing 130 g, is a little heavy for the smaller desktop robot platforms. The larger platforms easily exceeded our budget, and we needed a quick and cheap way of securely attaching the phone to the robot.

Maybe, like me, you have a box of Legos hidden somewhere, with various parts from the ages, collected by passing on pieces from aunt to nephew, friend to friend, or hitting the jackpot at a garage sale. Some of those 20-year-old parts fit in very well with modern electronic gadgetry — with a little help from our bag of tricks. Thus, we started building.

The legs

We based the platform's design on the venerable, classic "2838" 9-V motor from 90s' Lego Technic™ kits, with which I have previously built a quite minimal, wired-remote controlled, tracked vehicle, pictured in **Figure 1**. One motor for each side, with not much else to weigh it down, made it very nimble, if hard to control manually (with digital switches for each motor's direction).

That vehicle was too small to carry a battery, electronics, and a smartphone, and thus we expanded the platform by making it longer and wider, creating enough space to house the battery and electronics, and added a turret on top to give the smartphone's camera a perched view.

The resulting Bricksmith [1] design is pictured in **Figure 2**. The turret has a quick-release mechanism to easily attach the smartphone in landscape orientation, and can even be rotated electronically via its own motor, should we want to scan the landscape without moving.

For a little extra oomph, the larger platform has an additional motor on each side, with two motors driving each track. That little extra power gave the robot enough speed to run off the table in the blink of an eye quite a few times, when the software yet again locked up in full throttle.

The fully dressed robot can be seen in **Figure 3**, complete with phone attached, running the computer vision software.

The guts

Having decided on a high-level control platform via smartphone and mechanical platform with Legos bits, it was time to figure out what had to go in-between to

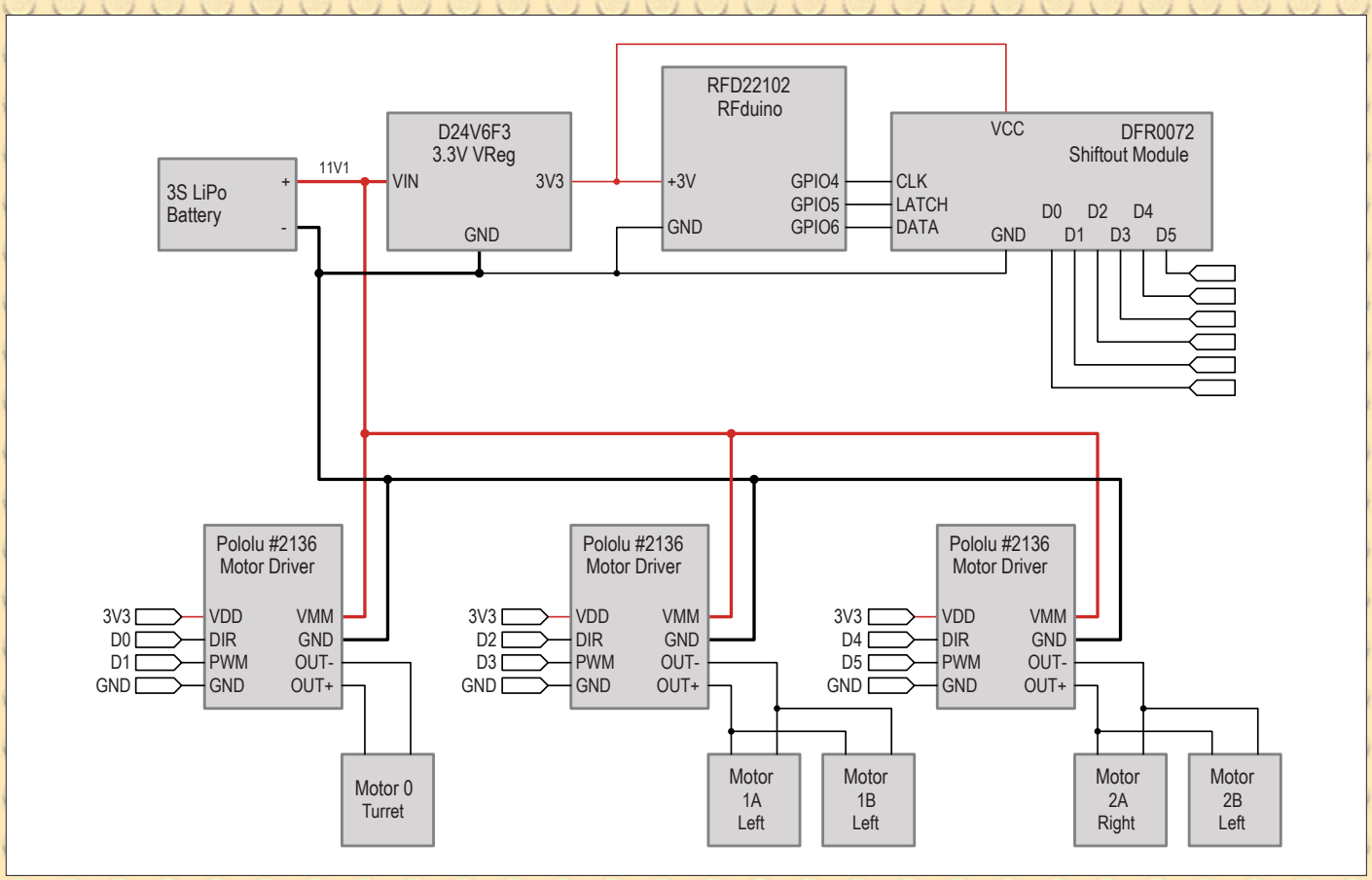


Figure 4. Motor driver electronics diagram.

make it all come together. The iPhone we planned to use provided only a single reasonable connection mode: Bluetooth 4.0 Low Energy (BLE). Bluetooth has been around for a while, being used for wireless headsets, or PC keyboards and mice, but the 4.0 'Low Energy' revision really hit the mark for Internet-of-Things type devices, enabling easily-built, battery-powered active radio devices that last months or years without recharging. It nicely fills the niche between Wi-Fi, with its high-power, high-bandwidth data transfer, and RFID, which is passive and uses no energy on the device side, but is also very limited in range and bandwidth. BLE combines very little power with comparatively large bandwidth and range, and a hobbyist-friendly device ecosystem has developed in a relatively short time. One example is the Nordic NRF51822 SoC, which has a 32-bit ARM Cortex-M0 processor core bundled with a BLE radio. The ARM Cortex-M0 is the power-saving kid sibling to the high-powered ARM Cortex-A series processors used in most current smartphones and it can also be programmed with an open-source GCC based toolchain, available for most PC platforms as ready-to-go binary packages [2].

The minimalist module of our choosing with the NRF51822 chip is the RFduino [3], not coincidentally rhyming with Arduino. The RFduino comes with an Arduino-compatible firmware and can be directly programmed from the Arduino IDE, or it can be loaded with custom firmware, if one wants to eschew the advantages, and restrictions, of the Arduino libraries. They are great for newcomers and prototyping, but may need to be circumvented if one has to (or wants to) get down to bare-metal C or C++ programming. The RFduino module has everything on board and only needs a 3.3-V supply to get started.

The RFduino is very tiny and has just a few I/O pins. Discounting the pins used for programming, five digital I/O ports are left. With one motor driver per side of the robot, plus the one for the turret, we need a total of six digital outputs (two for each motor driver). For this purpose, we employ a digital shift register that is attached to the RFduino with a 3-pin serial interface (clock, data, and latch): a DFR0072 shift register module, expanding three digital outputs to the six required for our purposes. If we had the need for further digital output signals, several shift registers could be daisy-chained to expand the number of outputs even further, at the cost of reducing the frequency at which the signals can be updated.

The DC motor drivers are controlled via a PWM and direction signal pair from the RFduino, for smooth analog control of the motor movement. Using the shift register means that the hardware PWM generators of the RFduino cannot be used, and the PWM has to be generated in software. PWM stands for pulsewidth modulation, and in this case it simply means that the motor driver connects the supply voltage to the motor while the PWM signal is High, and disconnects it when it is Low. Repeating in quick succession, varying the time the motor is on or off with each repetition, resulting in smooth control of motor motion. See also the PWM vs. PWM **inset**.

The NRF51822 microcontroller on the RFduino is targeted at extremely low-power applications and runs at just 16 MHz, with the processor being shared

PWM vs. PWM

The motor driver PWM signal is a bit different from, and not to be confused with RC-Servo PWM commonly found in radio-controlled planes, trains, and automobiles. The RC PWM controls the servo position (or motor power) by varying the width of the HIGH pulse between 1.0-2.0 ms, with 1.5 ms encoding a neutral position, and is sent with a fixed frequency, the exact format being a historical artefact of early radio control modulation techniques. This PWM, however, relies on the duty cycle only (not caring for the absolute pulse width, but only which percentage of time it is HIGH in a given period). The signal also varies the pulse width, but the two interpretations are quite different.

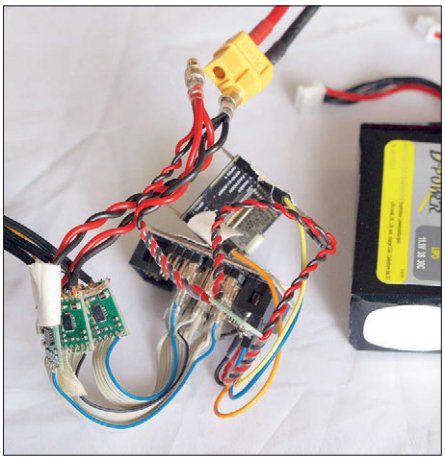


Figure 5. Motor driver electronics spaghetti.

by our motor driver software as well as the supplied Bluetooth-Stack, but that still allowed us to generate an 8-bit resolution PWM signal with a sufficiently fast 15-Hz cycle time for each motor.

The Pololu #2136 is built around a Texas Instruments DRV8801 driver IC and can handle 1 A of continuous current (our motor's stall current is 0.7 A @ 12 V). By attaching two mechanically coupled motors to each driver, we can hope for the driver's thermal management to protect the motors from excess current if something unexpectedly goes wrong. The #2136 driver also has an analog current sense output that could be used as feedback to the RFduino (which has two analog inputs) to implement current control, for even finer motor protection and movement precision.

The 2838 Lego motor is a simple ungeared DC motor with a no-load speed of 4100 rpm and 0.85 Ncm stall torque, requiring up to 8 W of electrical power when attached to a 12-V supply — which it can only handle for a short time before burning out. But, with matched drivers like the Pololu #2136, it can be run off a 3-Cell LiPo battery (with 11.1 V nominal voltage) without danger of combustion. We gave our robot a 3-cell 1200-mAh/13.3-Wh LiPo battery, which is a good mechanical fit, and provides more than enough juice for a day's worth of experimenting.

Figure 4 shows a diagram of the whole setup, including the battery and 3.3 V for the logic supply. The end result in **Figure 5** looks less neat, but fits in the spare space of the robot's trunk, as shown in Figure 3.

To round it all off, the RFduino is programmed to provide a custom service via its Bluetooth radio, which the phone uses to set the desired motor power and direction.

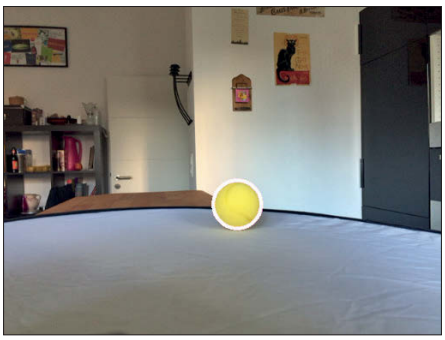


Figure 6. What the camera sees.

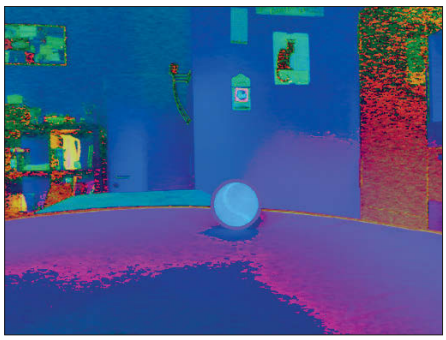


Figure 7. Converted to HSV color space.

The brains

Ordinary humans can locate a brightly colored ball very easily, but Computer Vision is not an easy problem, though thankfully made easier by OpenCV [4]. OpenCV comes with many algorithms that we could throw at the problem of detecting a brightly colored ball, or doing entirely different things entirely, for example face detection.

We quickly prototyped our ball detection pipeline on a PC with the Python scripting language and OpenCV, and later ported it to C to run on the smartphone. Using Python at first allowed us to quickly develop and test a rough sketch, disregarding performance and resource usage while experimenting. Once everything was working, we converted the ball detection to C code, in order to make better use of the phone's more limited resources. Even though a modern smartphone easily keeps up with a desktop computer from a few years ago, there is still a performance gap to a current PC. OpenCV has bindings to both the Python and C languages, greatly simplifying our work.

We implemented only a relatively simple image filtering pipeline, using a chain of OpenCV-provided image transformations that filtered out tennis ball colored and shaped areas of the image, like our target ball, or bystanders' bright neon sneakers. Starting with the basic captured image in **Figure 6**, it is first converted into the HSV (hue, saturation, value) color space visualized in **Figure 7**, which is equivalent to an RGB (red, green, blue) image in information content, but makes it easier to mask a specific color (hue) of varying brightness, the result of which is shown in **Figure 8**. This step masks large swaths of the original image from our search for the ball, and all we have to do is look for the right shape. This was implemented via the Hough Circles algorithm, a clever mathematical transformation that results in a list of circular objects.

In Figure 8, this operation results in only a single candidate which is correct, but as shown in **Figure 9**, it is not always that simple. Looking at the color mask in **Figure 10**, it can be seen that the color-masking step results in a large amount of noise, and the apparent shape of the target ball is literally lost in the shadows, not leaving much for the Hough Circles algorithm to work with. Under such conditions, no single right candidate circle can be reliably determined.

We also put together a user interface

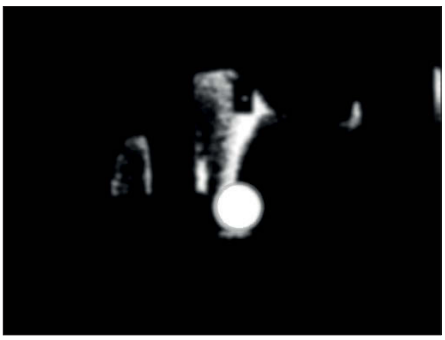


Figure 8. Color masked image.

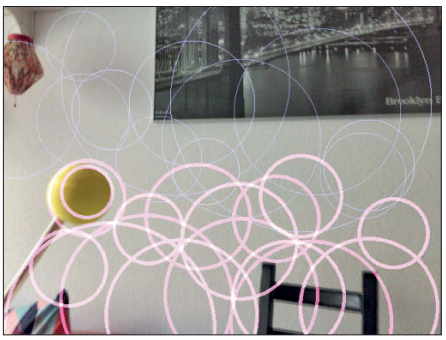


Figure 9. Confusion in adverse lighting.

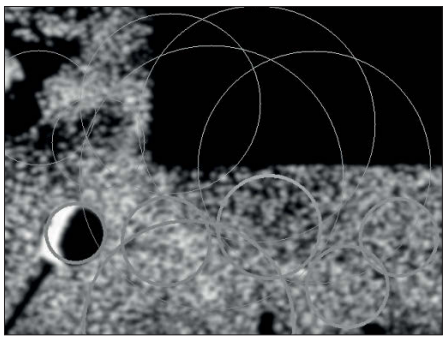


Figure 10. Confusion in adverse lighting, color mask.

on the smartphone, to be able to adjust various parameters, shown in **Figure 11**. It's not pretty, but gets the job done, with the top three HSV sliders controlling the color masking, the bottom K1, K2 the Hough Circles algorithm parameters. If a valid circle is found, we can convert its pixel coordinates into movement commands for the motors. We wanted the robot to approach the ball, but not run it over, so that it could follow a ball if it rolled along without running into it at speed and bouncing it away, out of sight. The x,y pixel coordinates in **Figure 12** can be converted into angles relative to the camera, resulting in a horizontal angle that can be used for steering, and a vertical angle γ, from which the distance can be estimated via **Figure 13**. Knowing the height of the camera h_c, and the radius of the ball r, the distance to the ball d can be estimated from

$$d = \frac{h_c - r}{\tan\gamma}$$

This being a tracked robot, there is no steering mechanism, just differential movement of the left and right tracks. Forward motion is achieved by driving both tracks in the same direction, and rotation by driving them in opposite directions. Rotation and forward movement are blended to rotate and move towards the target at the same time. When the desired distance is reached, the robot closes in no further, but keeps tracking the ball by rotating in-place. We implemented the forward motion as simply setting the approach speed proportional to the estimated distance, but chose a more complex PID-controller for the steering. This was necessary because the camera has a limited field of view, and if driven by a simple proportional (the *P* part of PID) controller, the robot can turn fast enough to overshoot the target from one vision frame capture to the next, with the ball no longer being in the field of view. We did not implement a scanning strategy for when the target is not in view, that task is left for a future rainy afternoon.

Concluding bits and pieces

A systematic overview is shown in **Figure 14**. There are a few details that didn't quite make it into this text, such as use of the phone's gyroscopic sensor to improve targeting, or the exact way in which the Bluetooth communication works on the Microcontroller. But, all the source code and documentation (so far as it exists) are available to download [5], peruse, and use, for all interested.
Also worth mentioning is the utility of tools such as the Open Bench Logic Sniffer, an open-source logic analyzer invaluable when one has to figure out why the shift register driving everything doesn't quite do the right thing 4000 times a second.
It is easier than ever before to tinker with hardware and software, with powerful wireless computers that fit in the palms of our hands, and all kinds of sensor and actor modules we can attach to them. All the electronics we used for our robot were ordered as fully assembled little modules: we only had to solder a few wires together, and it didn't break the bank, and turned out to be a great learning experience. ⏮

(160002)

Web Links

[1] http://bricksmith.sourceforge.net
[2] https://launchpad.net/gcc-arm-embedded
[3] www.rfduino.com
[4] http://opencv.org
[5] www.elektormagazine.com/160002

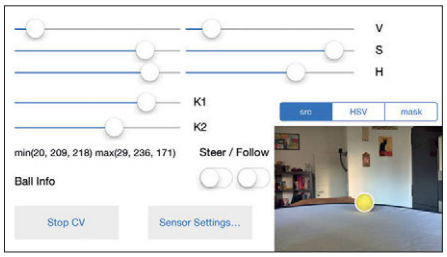


Figure 11. Smartphone UI.

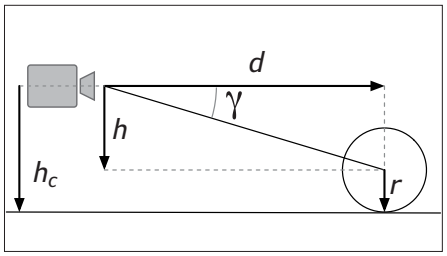


Figure 13. Distance estimation.

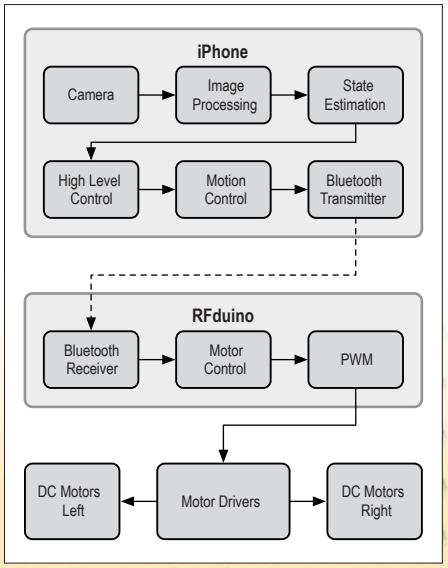
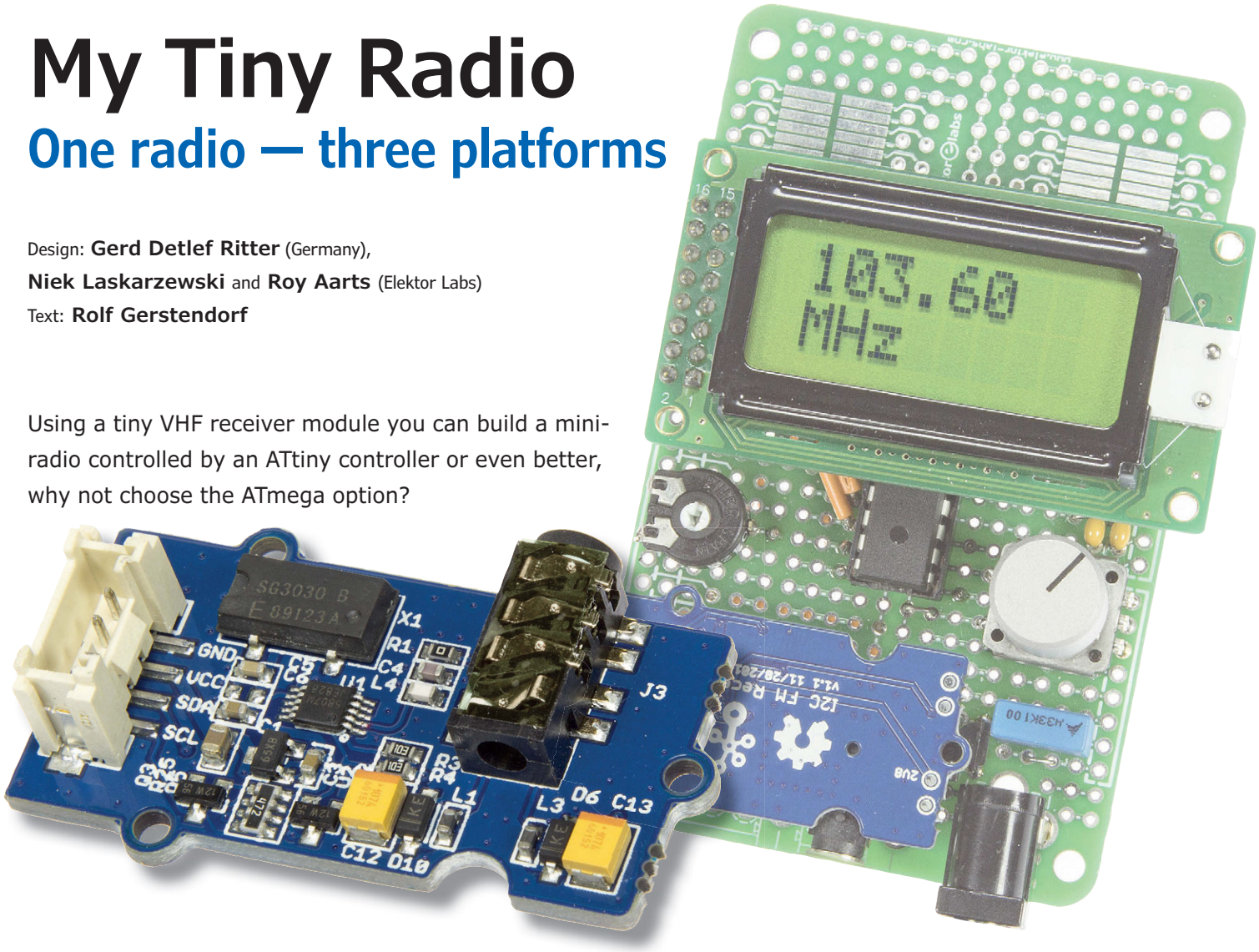


Figure 14. System Overview.

My Tiny Radio

One radio — three platforms

Design: **Gerd Detlef Ritter** (Germany),
Niek Laskarzewski and **Roy Aarts** (Elektor Labs)
Text: **Rolf Gerstendorf**



Using a tiny VHF receiver module you can build a mini-radio controlled by an ATtiny controller or even better, why not choose the ATmega option?

Many exotic and wonderful things come from The Orient. These days it's not just rare spices and hand-crafted ornaments but also electronic goods. One company with its headquarters in China's Silicon Valley in Shenzhen, Guangdong, situated just to the North of Hong Kong are specialists in producing all things electronic to gladden the heart of any serious hacker or Maker. The company is called Seeed Studio [1]. You just need to go to their website to realize you've struck gold: There are a wide variety of boards on offer including Shields, Hats and Capes relating to Raspberry Pi, Arduino, BeagleBone, IoT, ARMmbed, Wearables *etcetera etcetera*.

One of our readers, Gerd Detlef Ritter was impressed by a tiny FM receiver module with an I²C interface he discovered there amongst the selection of 'Grove' modules. This module is a complete VHF receiver controlled fully via its serial I²C interface. The receive frequency bandwidth extends from 50 MHz to 150 MHz including support for RDS (and its US variant, RBDS) and with a stereo jack socket for direct headphone connection. The module works with a supply in the range of 3.3 V to 5 V and uses so little power that battery operation is feasible (especially when coupled with a low-power processor for controlling the module).

Seeed Studio are well known for not treating their products as just anonymous black boxes; they also provide backup in the shape of software and hardware information (including Eagle PCB layout files) useful for anyone developing their

Features

- 50–115 MHz tuning range
- Single-Chip Stereo FM Tuner with I²C interface
- Compact Break-out-Board
- RDS support
- Automatic station scanning
- Headphones socket (32 Ω minimum)
- Operating supply 3.3 to 5 V at <100 mA
- Control via ATtiny, ATmega or Arduino

own project based on these products. So it shouldn't have been so surprising, not long after Gerd received his VHF receiver module in the post he had hooked it up to his Arduino Uno, tinkered with the code examples, and was sitting back enjoying a cup of coffee while listening to 'Radio Ga Ga' by Queen on his headphones.

A fully integrated FM tuner

The module is based on the FM stereo tuner chip type RDA5807M from RDA Microelectronics based in Shanghai. Its datasheet [2] shows the IC's internals (**Figure 1**), just try to ignore the large pink 'Confidential' watermark on every page. The IC has the following features: The receiver has a tuning range of 50 MHz to 115 MHz which makes it suitable for the reception of FM broadcasts in all countries. It contains a low-noise amplifier (LNA), a programmable gain amplifier (PGA), high resolution A/D and D/A converters with an audio DSP in between.

The synthesizer uses a 32.768-kHz reference clock to generate the local oscillator signal (VCO) for the mixer. The multi-phase mixer provides good image-signal rejection and a limiter stage reduces overloading and limits intermodulation products generated by signals from powerful adjacent channels.

The DSP core takes care of channel selection (using the CHAN register), FM demodulation, Stereo-MPX demodulation and produces a digital signal for the audio-output. If the received signal strength is marginal the MPX decoder automatically switches to mono to reduce noise.

The stereo digital output signal from the core is converted to left and right audio using two D/A converters with the resulting signals output at L_{OUT} (pin 10) and R_{OUT} (pin 9) which can be used to drive headphones (with an impedance >32 Ω) directly. The D/A converter stage also controls the output volume and includes low-pass filters for the audio output signals with a cutoff frequency of around 30 kHz. The RDA5807M also features an integrated low-drop voltage regulator to power all the stages in the radio module. Supply voltage is in the range from 2.7 to 3.3 V. The chip performs an auto-reset function on power-up and a soft-reset can be issued using commands over the I²C bus.

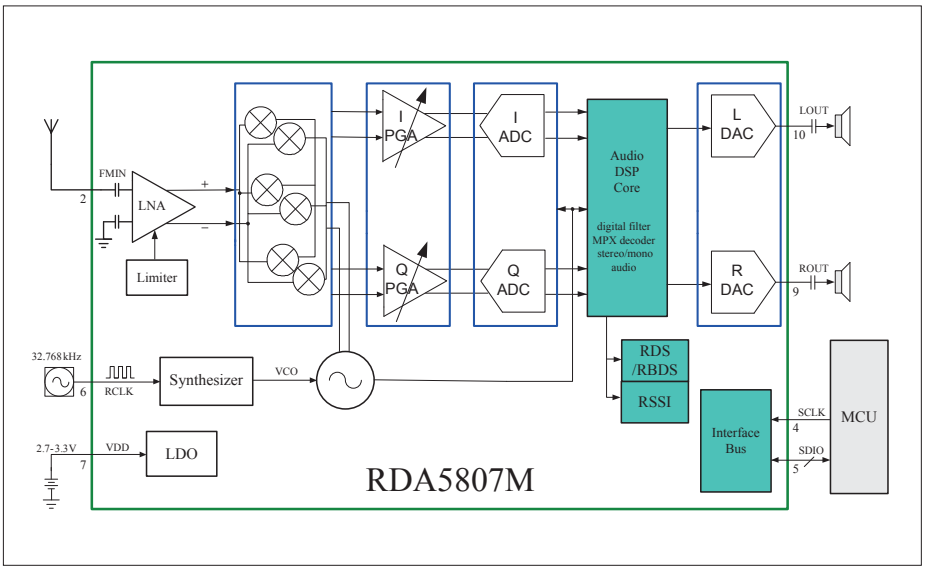


Figure 1. Internals of the integrated FM tuner chip. (Source: RDA Microelectronics).

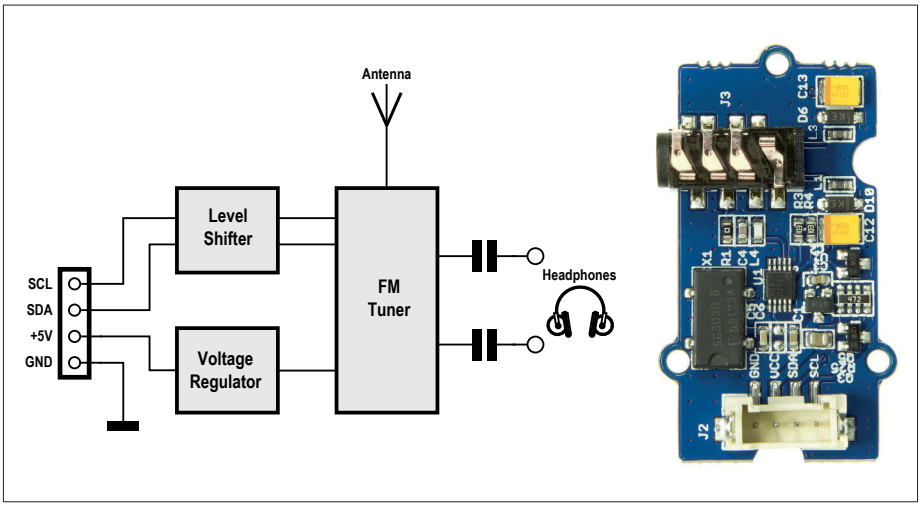


Figure 2. Block diagram of the FM module with the integrated FM tuner.

The FM module Break-out-Board

The small Grove I²C FM receiver board is effectively just a breakout board for the RDA5807M chip; a block diagram of the board is given in **Figure 2**, along with the RDA5807M is a 4-way pinheader. The pin assignment is the same as with all the Grove expansion boards with I²C interfaces: Ground pin 1, positive supply pin 2, SDA pin3 and SCL pin 4. The supply voltage to the chip is a stabilized 2.8 V. The on-chip regulator can accept an input voltage in the range of 2.7 to 3.3 V and the external on-board regulator allows the module to interface with 5-V systems. It is of course necessary to use a level-shifter chip to convert the 5 V I2C signals to operate at the 2.8-V onboard voltage level.

Following the application sheet suggestions, both Left and Right outputs to the headphone socket pass through coupling capacitors and inductors (L1, L3) to suppress any unwanted VHF components on the audio signals. The RCLK pin on the IC connects to the crystal and its loading capacitors while the antenna input FMIN comes out to a solder pad.

From Arduino to Tiny up to Mega

For the first trials we can use the relatively bulky Arduino Uno board — it works well but is not really designed to be built into mobile equipment. Alternatively we could use a smaller Arduino board but that will be quite expensive and also relatively power hungry!

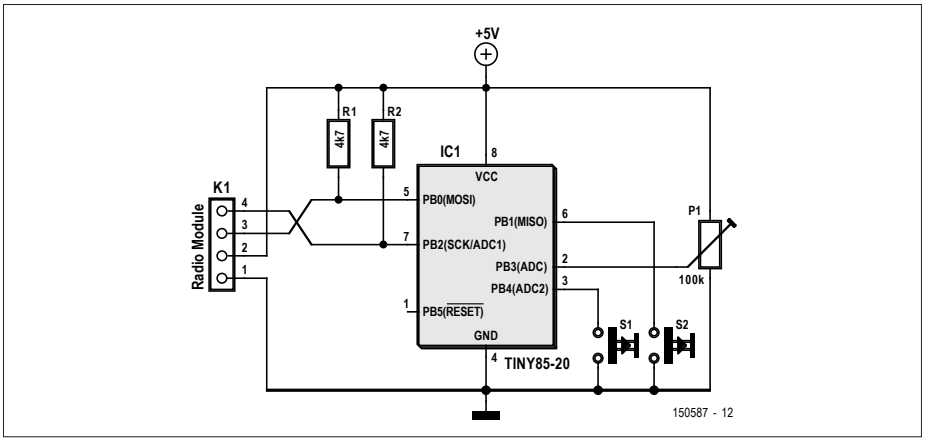


Figure 3. Not much to see here! Along with the 8-pin controller there are just two pull-up resistors and the user-frontend consists of two push buttons and a preset.

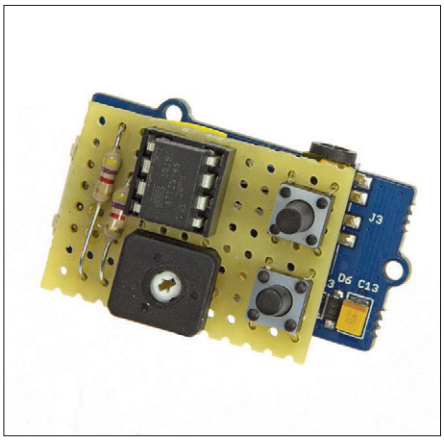


Figure 4. The minimalist version: The perf board with the controller is smaller than the radio module!

COMPONENT LIST FOR ATTINY VERSION

Resistors

R1,R2 = 4.7kΩ
P1 = 100kΩ trimpot

Semiconductors

U1 = ATTINY85-20PU, programmed, Elektor Store # 150587-42

Miscellaneous:

S1,S2 = pushbutton, 6x6mm
K1 = 4-way pinheader socket, 2mm pitch
U1 = DIP8 IC socket
I2C FM Radio module (Seeed Studio)
Holder for 3 AAA batteries

For the small radio module the ATtiny85 microcontroller seems like a good option. It costs practically nothing; it's available in a DIL8 outline for easy mounting and has an I²C interface (or more correctly, a Universal Serial Interface (USI) which we can use as an I²C interface). On top of that there are two free I/O pins we can use to read the state of two push-buttons for I²C Up/Down register setting. The ADC input at PB3 can be used to read the value of the trimmer to control out-

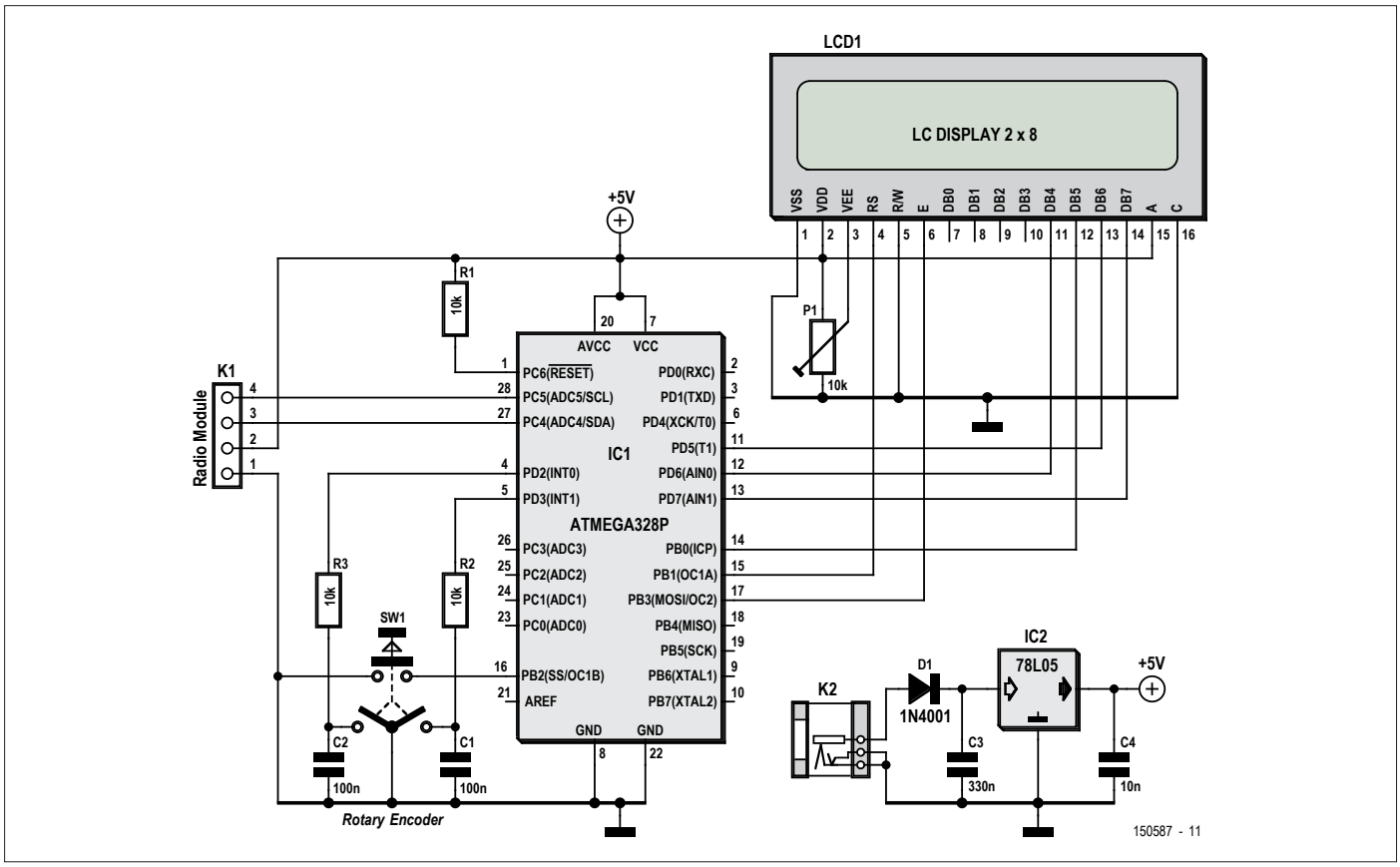


Figure 5. Schematic of the larger ATmega controller board with the rotary encoder and LCD.

put volume. The tiny controller also has 8 K of flash memory space for our code! The few through-hole components needed for this circuit shown in **Figure 3** can be mounted on a small square of perfboard and piggy-backed onto the radio module. You can see from **Figure 4** that it shouldn't take more than a few minutes to fit and wire up all the components. A separate antenna isn't needed; the headphones act as an antenna. Power can be provided by a 3-V battery or a small solar cell. The circuit draws 100 mA max.

The tiniest version certainly exemplifies the minimalist approach but as a result, the user interface is not exactly what you might describe as user friendly. A small 2 x 8 line LCD would certainly be useful to convey information to the user and a rotary encoder instead of an analog rotary pot and a pushbutton would also be an improvement. Unfortunately the ATtiny85 does not have enough I/O lines and memory space to accommodate the improvements. Here in the Elektor lab two of our lab interns took up the challenge and set about replacing the ATtiny with an ATmega328. This version certainly has enough I/Os and it's even available in the classic 28-pin DIP package for simple mounting on a breadboard.

Looking at the circuit diagram for this version (**Figure 5**) we can see the obligatory Grove-Interface (K1), the display (with P1 for contrast control) and a rotary encoder with push button function (together with components to debounce the switch signals) a low voltage DC supply socket (K2), followed by a 5-V fixed voltage regulator IC2 and decoupling capacitors C3 and C4. Diode D1 prevents any damage if a supply of incorrect polarity is connected at K2.

Even this pimped-up 'luxury' version still uses a perfboard but it looks much neater than a standard breadboard. In fact this is our new-generation prototyping board type **ELPB-NG** which you can also order from the Elektor Store. As you can see in **Figure 6** the overall appearance is much less cluttered than a standard breadboard. The FM module is flipped and plugs into the pin header K1. To the right of center is the headphone socket and further to the right the rotary encoder. The LCD contrast trimmer is to the left and the ATmega328P is in between, under the LCD. **Figure 7** shows the position of all the components and solder bridges necessary on the board; assembly and soldering really should be a cinch!

COMPONENT LIST FOR ATMEGA VERSION

Resistors

R1,R2,R3 = 10kΩ
P1 = 10kΩ trimpot

Capacitors

C1,C2 = 100nF, 2.5mm pitch
C3 = 1µF 50V, electrolytic, 2mm pitch
C4 = 2.2µF 50V, electrolytic, 2mm pitch

Semiconductors

D1 = 1N4148
U1 = ATMEGA328P-PDIP, programmed, Elektor Store # 150587-41
IC1 = LP2950

Miscellaneous:

K1 = 4-way pinheader socket, 2mm pitch
LCD1 = LCD (2x8) with 16-way (2x8) pinheader socket, 0.1" pitch
U1 = DIP28-narrow IC socket
SW1 = rotary encoder, Alps EC12E2424407
K2 = DC adaptor socket, 2.5mm (Lumberg NEB21R)
I2C FM Radio module (Seeed Studio)

Miscellaneous:

S1,S2 = pushbutton, 6x6mm
K1 = 4-way pinheader socket, 2mm pitch
U1 = DIP8 IC socket
I2C FM Radio module (Seeed Studio)
Holder for 3 AAA batteries

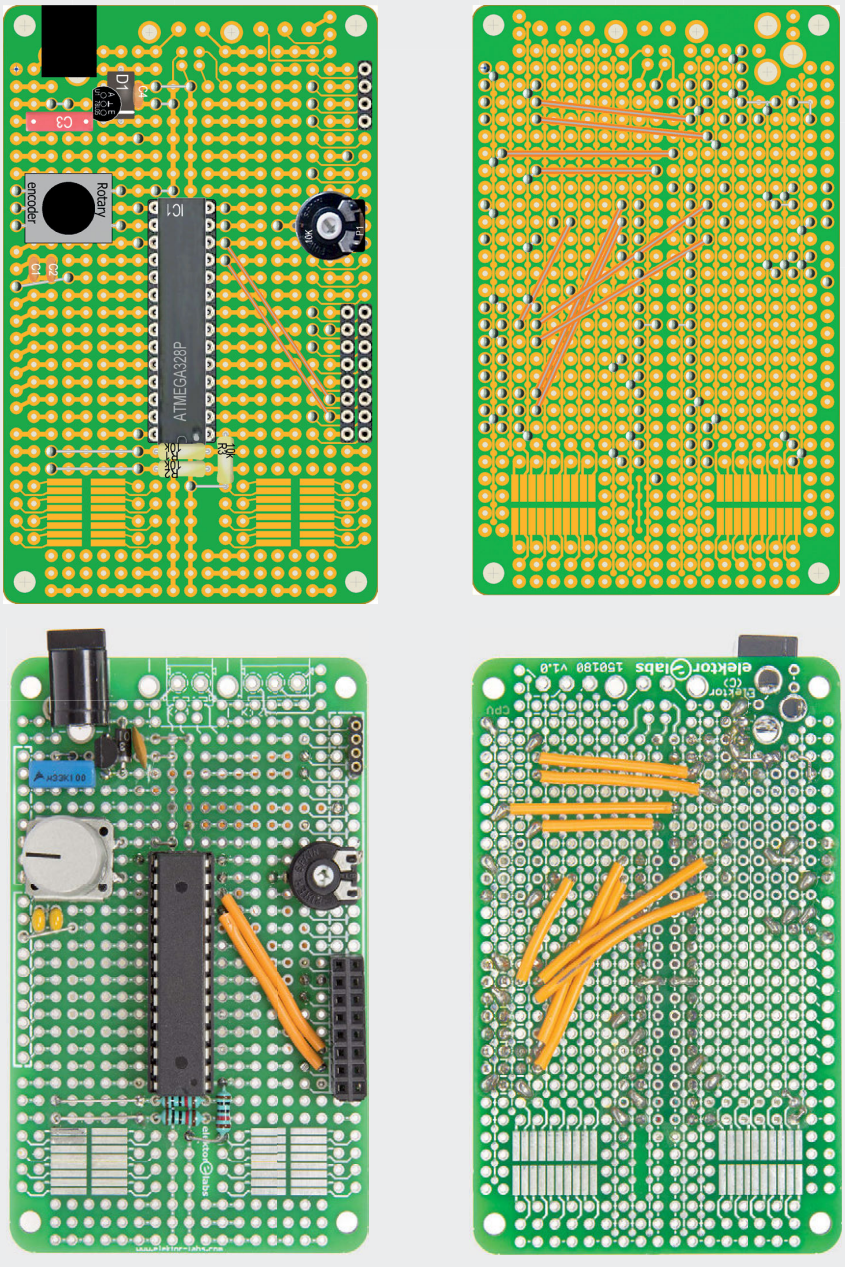
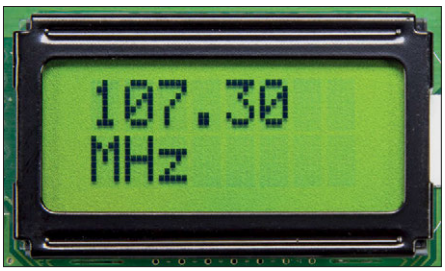
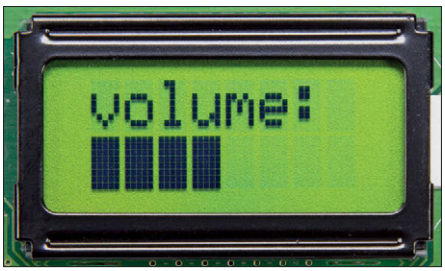


Figure 7. Overview of component positioning and wiring using the ELPB-NG board.





When the module is switched on the display shows the frequency the radio is tuned to. Twiddling the rotary encoder allows you to select a different frequency up or down. By pressing the rotary encoder knob the display changes to show menu options where you can use the rotary encoder to change the volume, mute the audio or start frequency scanning for strong stations.
Twiddling the rotary encoder now tells the radio in which direction it should start scanning (either up or down in frequency). The settings and station selected are stored in the controllers EEPROM and reloaded when the radio is switched on again.

Software for the Arduino, Tiny and Mega

So far we haven't touched on the software. Indeed if you have no interest in this area you can simply order either one or both of the two controllers pre-programmed along with the other materials directly from the Elektor Store (see later) and you'll be up and running in no time. Our technical manager here at Elektor labs Clemens Valens gives a short video walkthrough on both versions of this project [6].
It is interesting to compare the differences with the Demo software from Seed Studio available via Github [3]. This hardware design is like our ATtiny radio with a pot to set the output volume and two pushbuttons for tuning but the firmware is written for the Arduino IDE so needs some adjustments to run on our ATtiny version. The core however remains largely the same no matter which controller is used to control the radio module.
In the demo version the Wire.h library is used to communicate with the radio IC module. As we already mentioned the ATtiny85 doesn't have a 'real' I²C interface but instead has a USI. You can't therefore use the standard 'wire' library with this controller. Fortunately at the Arduino-Playground you can find code snippets [4] for implementing an I²C interface on an ATtiny85. Just like in the demo version we must, in addition to the new library, also define the port pins. Apart from that, there is very little difference to be found between the firmware for the ATtiny (from [5]) and the Arduino software [1].
And so to the ATmega firmware. You will need to download the 'large' software version for this project from the project page [5] to follow this description; the source code is also well documented.
In the code for the ATmega we can of course use the Wire library and we also need to include the LiquidCrystal.h library for interfacing to the LCD. After defining and initializing variables (minimum signal strength, Frequency steps for scanning, BassBoost, Mono/Stereo...) are also a number of variables defined and initialized for the LCD menus and for the rotary encoder control (and its pushbutton).
After all the 'includes', 'definitions' and initializing there is a whole series of functions. In the setup section the pins used for the rotary encoder are defined, the LCD is started and the settings written to the FM module. In the Loop function the receiver frequency is written to the display, the rotary encoder contacts read and the volume menu started or the Autoscan and mute bits written into the 'settings' variables in the EEPROM. When Autoscan is active the search for another station is initiated.
Two more functions for the main menu and volume menu then follow the Interrupt-Service Routine which reads the rotary encoder contacts and establishes the direction of rotation and depending on the active menu, takes care of the station frequency, the cursor position or updates the volume setting. The last function clears the LCD by writing eight space characters to the display.
The functions setVolume and fmSeek in both software versions differ slightly because of the use of the rotary encoder and LCD.
The function to control the FM module provided by the demo from Seeed Studios comes after the showRadioStation and showSignalStrength function (after `// FM functions`). In this demo just some of the more basic information held in the RDA5807 I2C registers is read and displayed. The data sheet details other information held in the internal registers and with some additional software (there is space enough in the ATmega memory) we could for example display things such as RDS information banners on the LCD. ⏮

(150587)

FROM THE STORE

- 150587-11 Software
- 150587-41 Preprogrammed ATmega controller
- 150587-42 Preprogrammed ATtiny controller
- 150180-1 Prototyping board ELPB-NG
- 150587-71 Kit containing Prototyping board, preprogrammed ATmega and radio module

Web Links

[1] Grove I²C FM module : www.seeedstudio.com/depot/
[2] RDA5807M Data sheet (confidential!): www.seeedstudio.com/wiki/File:RDA5807M_datasheet_v1.1.pdf
[3] Demo software for Arduino: https://github.com/Seeed-Studio/I2C_FM_Receiver
[4] Code snippets for I2C/USI using an ATtiny85: http://playground.arduino.cc/Code/USIi2c
[5] Elektor project webpage: www.elektormagazine.com/150578
[6] Tiny Radio video: https://youtu.be/c2T2jd9OFVg

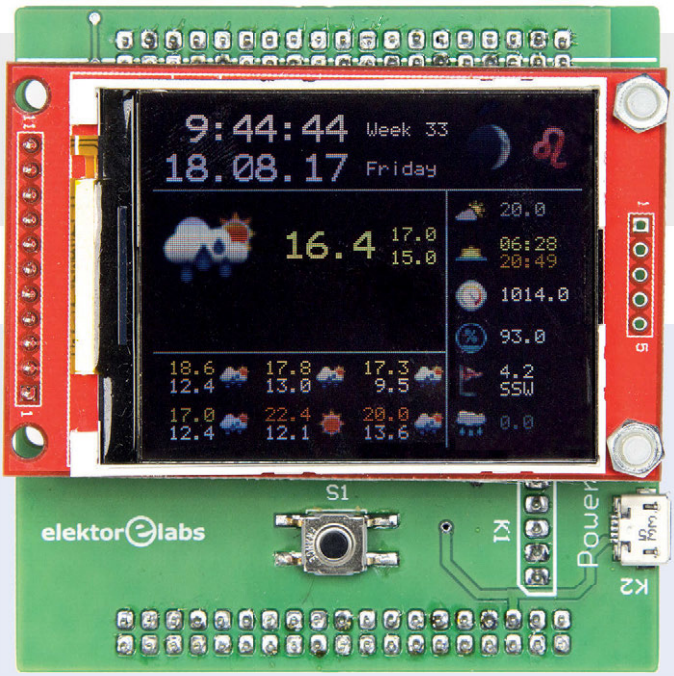
Err-lectronics

Corrections, Updates and Feedback to published articles

Weather Display

Elektor 6/2017 (November & December), p. 90 (160157)

UPDATE. Table 3 'FTDI cable connections to Nucleo' has an error. TxD goes to CN10-31 on the Nucleo board and not CN7-31 as given in the table.



Weather Display

Elektor Magazine 6/2017 (November & December), p. 90 (160157)

FEEDBACK. I'm having trouble generating the machine code for the ESP-12E module. The software consists of the following files: *weathertimeget_el.ino, parse.ino, config.ino, wget.ino, wgetu.ino*. When I compile the *weathertimeget_el.ino* in the Arduino-IDE it generates an error because some of the necessary program components are contained in other files.

Apart from that I'm unsure if the entry in the Arduino-Sketch *weathertimeget_el.ino* `Const StringLocation = "Schweinfurt"` is sufficient or do I need to also enter the digit associated in the location list with the place name?

Jürgen Rieger

Reply: All five .ino files must be in the same folder. When you open *weathertimeget_el.ino* in the Arduino IDE, all the files will be opened and (eventually) compiled together. For your location just enter „Schweinfurt" that will be enough.

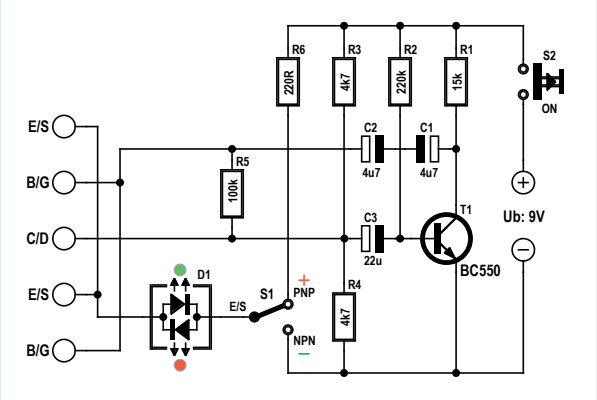
Luc Lemmens (Elektor Labs)

Bi-colour Transistor Tester

Elektor Magazine 1/2018 (January & February), p. 92 (160542)

FEEDBACK. Should C1 and C2 really be connected back to back in series as shown in the schematic for this project or should there be a connection shown at their centre to R2 and the base of T1?

Dieter Glatz



The author replies: When polarized electrolytic capacitors are connected back to back as shown in the diagram they become 'AC capable' and can be used in place of a non-polarised capacitor. This is important here because when switched, the collector and base of the test transistor change state alternately between positive and negative. This would be an unacceptable operating condition for a single polarised electrolytic. The diagram is correct and there should be no connection between the centre connection of these two capacitors and the base of the transistor. The value of these two capacitors connected in series becomes C = 2.35 µF approximately. In this application you could substitute two non-polarised 1-µF capacitors here; in this case they should be connected in parallel.

Hans-Norbert Gerbig

Universal I²C Bus Isolator and Level Shifter

Simple and compact

By **Andre Jordaan** (Switzerland)

When you want to connect two or more I²C devices together there may, in some cases, be the requirement to change the voltage level of the signals. And while you're at it, it may not be such a bad idea to build in galvanic isolation at the same time.

Characteristics

- Suitable for operating voltages from 3.0 to 5.5 V.
- Bidirectional
- 2 channels
- Ground and power supplies are galvanically isolated

In order to allow I²C devices operating from different power supply voltages to be interconnected, we here present a simple level shifter, which is intended as a converter for signal levels from 3.3 V to 5 V and the other way around. The circuit also provides galvanic isolation between the signal lines at the same time. The idea arose when I wanted to connect an Arduino (5 V) to a Raspberry Pi (3.3 V).

No optocouplers

I initially looked at using optocouplers, but because the signal that needs to be

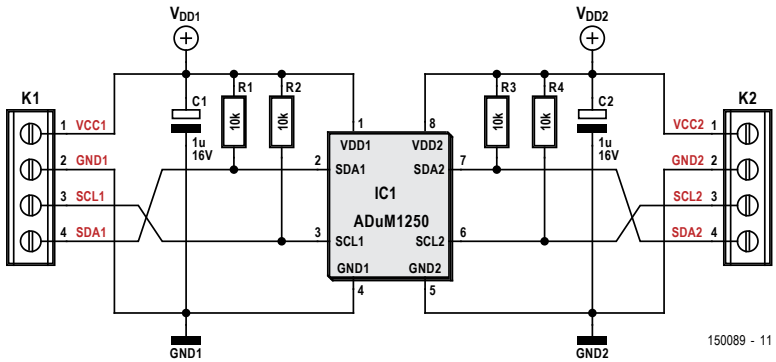


Table 1. Pinouts for K1 and K2.

Pin number	K1	K2
1	VCC1	VCC2
2	GND1	GND2
3	SCL1	SCL2
4	SDA1	SDA2

Figure 1. The schematic shows how easy it is to use the ADuM1250.

level-shifted is bidirectional, this also means that he would have to split the signal and work with optocouplers operating in both directions. On the Internet I read all sorts of reports about glitches and problems when using optocouplers on a databus, among other reasons because of small variations in propagation speed. Problems because of this would have been difficult for me to track down. So I searched for another solution.

I arrived at the ADuM1250 from Analog Devices. This IC is a double I²C isolator and level-shifter, with complete galvanic isolation between the two sides (up to 2.5 kV) and can handle bidirectional switching speeds up to 1000 kHz. The IC supports power supply voltages and logic signal levels between 3.0 V and 5.5 V and is 'non-latching'.

Using this IC it is therefore not necessary to split the bidirectional I²C signals into separate send and receive signals, as would have been the case when using optocouplers. An additional benefit is that the IC is 'hot-swappable'. In this case that means that when the IC is connected to an active bus and it does not have a supply voltage applied, it will not cause glitches on the data bus.

The schematic

The schematic shown in **Figure 1** has been designed in such a way that the power supply voltage, which also defines the signal levels (generally 3.3 or 5 V) can be arbitrarily chosen for either side.

The SDA and SCL signal lines are each fitted with pullup resistors of 10 kΩ (R1 through R4). The supply voltages on each side are buffered by C1 and C2. All pins of the IC are accessible via PCB terminal blocks K1 and K2. The pinouts for K1 and K2 can be looked up in **Table 1**. SCL1 is connected to SCL2 via the ADuM1250 and the same is true for SDA1, which is coupled to SDA2.

Hardware

A simple, compact printed circuit board has been designed for the Universal I²C bus isolator and level-shifter, see **Figure 2**. The printed circuit board layout can be downloaded from the project page on our website [1]. The circuit board is also available from the Elektor Store [2]. The soldering of the few SMD components is not all that difficult, but if you are not looking forward to that then a completely-built module is also available from the Store. Note: the PCB terminal blocks are supplied, but you will still need to fit those yourself.

In use

Although very little can go wrong with the ADuM1250, we will nevertheless describe a simple test procedure, which will allow the correct operation and connection of the Universal I²C bus isolator and level-shifter to be verified. This procedure tests the I²C communication with an Arduino Uno. The necessary Arduino sketches can be downloaded from [1].

Test procedure 1
using an DS1307 RTC (5 V):

- From within the Arduino development environment, open the sketch '_5V_test_program' and load it into the Arduino Uno.
- Connect the Uno via K1 to the circuit board.
- Connect the DS1307 to K2 and use a supply voltage of 5 V. Ensure that there is no common ground connection between the Arduino and the DS1307.
- Open the serial monitor from the Arduino development environment (data connection 9600 baud). This now indicates the date and time which is generated by the DS1307.

Test procedure 2
using an EEPROM (3.3 V):

- From within the Arduino development environment, open the sketch '3V_test_program' and load it into the Arduino Uno.
- Connect the Uno via K1 to the circuit board.
- Connect an EEPROM (24C02) with an operating voltage of 3.3 V via K2 of the bus isolator. Ensure that there is no common ground connection between the Arduino and the EEPROM.
- Open the serial monitor from the Arduino development environment (data connection 9600 baud).
- The sketch writes the value 90 to the EEPROM at address 1. This value also needs to be displayed in the serial monitor when this address from the EEPROM is read again. ⏮

(150089)

COMPONENT LIST

Resistors
R1,R2,R3,R4 = 10kΩ, 0.1W, SMD0603

Capacitors
C1,C2 = 1µF 16V, tantalum, CASE A

Semiconductors
IC1 = ADUM1250ARZ, SOIC-8

Miscellaneous
K1,K2 = 4-way PCB terminal block, 3.5mm lead pitch

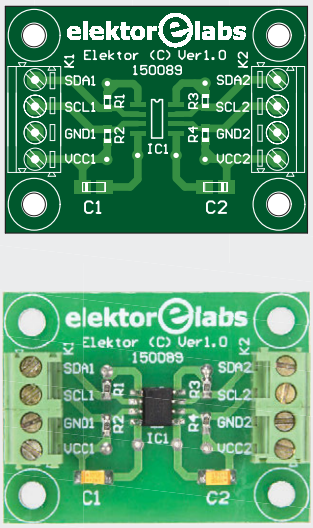


Figure 2. The compact circuit board reflects the simplicity of the schematic.

Web Links

[1] www.elektormagazine.com/150089
[2] www.elektor.com, Search: 150089

Close-to-Universal IR Remote Control

This remote control is a solution to an almost universal problem. All audio and video equipment have their own remote control, with the result that you need a handful of them just to listen to something or watch something. All of these devices also tend to be left in standby 24/7 and hence waste a non-trivial amount of energy in the background.

By **Goswin Visschers** (The Netherlands)

The first problem could obviously easily be solved by buying a programmable universal remote control. However, I wanted one that had a digital input, so that everything could be turned on or off simultaneously when somebody entered the room. This is easily done using a motion detector. But it turned out that there aren't any universal remote controls with a digital input, which led me to design this project.

Almost universal

The reason for calling this design an 'Close-to-Universal IR Remote Control', is that not all of the IR protocols could be

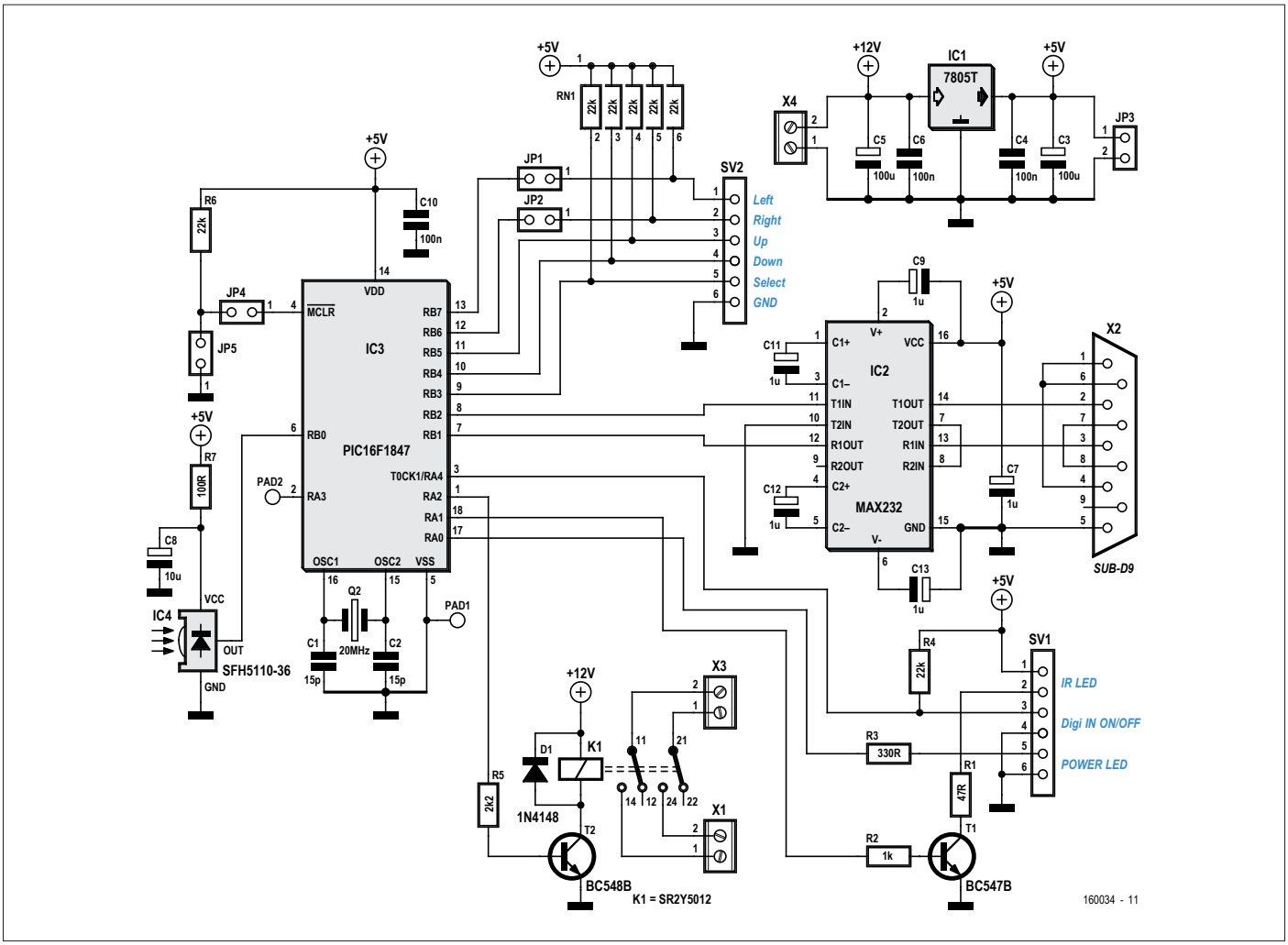


Figure 1. The circuit diagram for the Close-to-Universal IR Remote Control.

Specifications

- Supports the following IR protocols:
  - Philips RC5, RC5 Extended, RC6 Mode 0 and use of the toggle bit
  - NEC protocol and NEC extended protocol
  - Sony Sirc 12-bit, 15-bit and 20-bit (the last one has not been tested)
- Transmission of On and Off codes via a digital input
- Several IR transmitter LEDs can be connected in parallel
- Codes can be 'learnt' via a serial terminal (RS232)
- A relay circuit can completely isolate the equipment from the electricity supply (power saving)
- Selection between four sources (TV, radio, USB, CD, etc.) via the 'Select' button
- Equipment can be turned on and off using a maximum of 6 different IR codes
- Four programmable buttons can transmit different IR codes depending on which source has been selected

supported due to technical limitations. The specifications for the circuit are shown in the inset further down.

The circuit

There is nothing special about the circuit diagram (**Figure 1**) for the Almost Universal Remote Control. The supply voltage is provided by a 7805 5V linear voltage regulator, along with several capacitors. A MAX232 with its associated capacitors takes care of the level shifting between TTL and RS232, and a standard IR receiver ensures that the microcontroller can receive IR signals when the codes are being programmed.
The SFH5110-36 IR receiver used here has a 36 kHz bandpass filter, whereas Sony remote controls modulate the IR commands at 40 kHz and NEC at 38 kHz. This isn't a problem though; as long as the remote control is close enough to the receiver, the received signals will be strong enough to pass through the filter.
Microcontroller IC3 is in charge of the circuit. This PIC16F1847 is clocked at 20 MHz by crystal Q2. Jumpers JP1, JP2, JP3 and JP4 are only used during the in-circuit programming of the microcontroller. If this isn't required, you can replace JP1, JP2 and JP4 with wire links.
JP3 supplies power to the in-circuit programmer and could be left out if that facility isn't required. JP5 is used to set the Digi IN input as either active-high or active-low. With JP5 closed, the Digi IN input is active-low, and the on-codes will be transmitted by the IR LED when SV1-3 are SV1-4 connected together. If JP5 is left open, the function of Digi IN is the opposite.
The connections to PAD1 and PAD2 currently don't have a function, but they could be put to use by adapting the software.
Relay K1 is turned on after the on-codes have been transmitted and turned off after the off-codes have been transmitted.
The IR LED is connected to SV1-1 and SV1-2. When several LEDs are connected in parallel, resistor R1 should be replaced with a wire link and each IR LED should have its own series resistor, with a value of 47 ohms, for example.
A green LED can be connected to SV1-5 and SV1-6. It will flash when the circuit is active.
RN1 is a resistor network that ensures that inputs RB3 to RB7 will be at a defined high level when the buttons aren't pressed.
I have designed a PCB layout for this project using Eagle. All project files (Eagle files, PCB layouts, etc.) can be downloaded from the website for this project [1].

The software

With most of the universal remote controls, the IR codes to be learnt are sampled at high frequencies and stored in the memory of the microcontroller. The advantage with this method is that these remote controls can cope with all existing IR protocols. The disadvantage is that a relatively large amount of memory is required to store these codes. It just so happens that the limited amount of EEPROM memory is one of the disadvantages of the PIC16 series of microcontrollers from Microchip, which I prefer to use.
We could get round this limitation by using an external EEPROM to provide more memory, but I decided to make the software of the microcontroller more 'intelligent' instead. The disadvantage of this solution is that the controller can only support a limited set of protocols. This software can also be downloaded from the website for this project [1].

The 'intelligent' software recognizes the IR protocol that it is learning. The advantage of this is that the microcontroller only needs to store the type of IR protocol and the data contained in the IR code. The type of IR protocol can be stored in a byte and the data will never be more than 32 bits, therefore it can fit in 4 bytes. This 'intelligence' does require more program code, which is why I've chosen the PIC16F1847, which has 14 Kbytes of program memory. This memory has been filled up to the brim in order to support as many IR protocols as possible.
The protocols are recognized by measuring the pulse-width of the first IR pulse. NEC, for example, uses a pulse of 9 ms (**Figure 2**) and RC5 uses a start pulse of 0.9 ms (**Figure 3**). Sony's Sirc and Philips' RC6 use an almost identical start pulse of about 2.4 ms. In order to differentiate between them, the gap before the second pulse is also measured. With RC6 this gap is

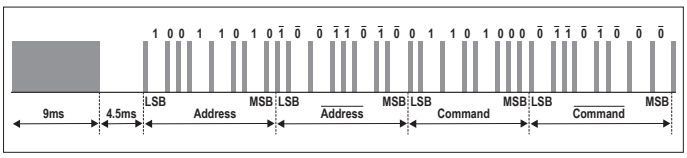


Figure 2. An example of a NEC protocol IR code.

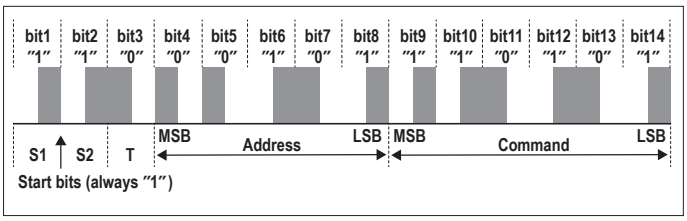


Figure 3. An example of a RC5 protocol IR code.

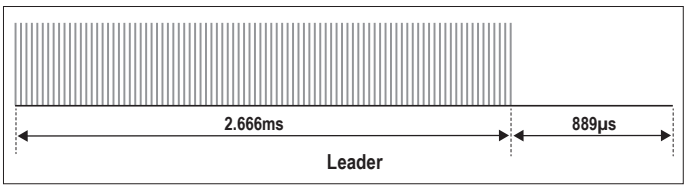


Figure 4. Start pulse of the RC6 Protocol.

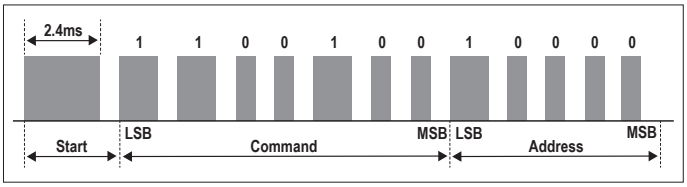


Figure 5. An example of a Sony SIRC protocol IR code.

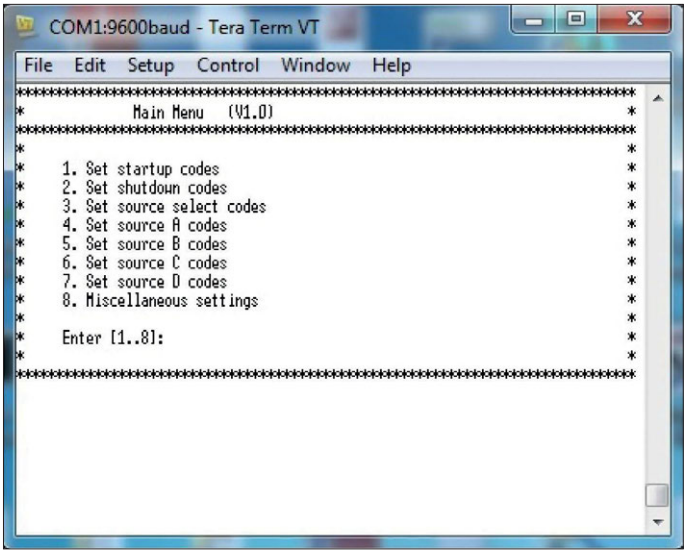


Figure 6. Once the serial port has been configured and the circuit has been turned on, you'll be greeted by this menu.

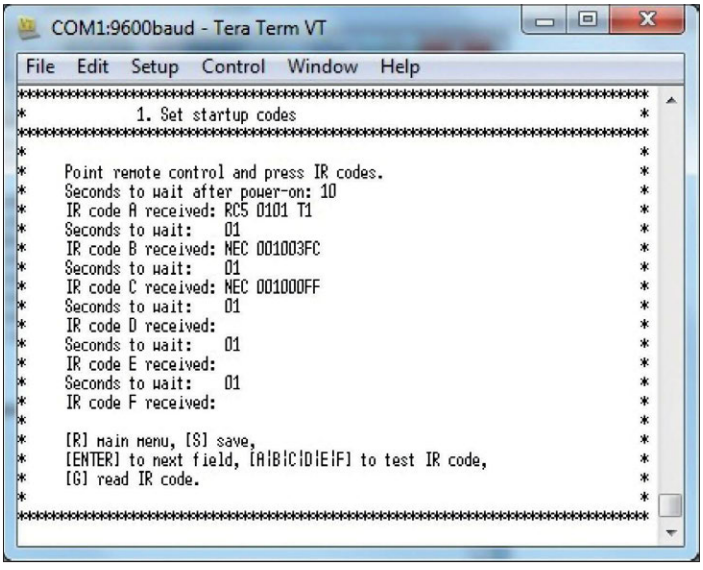


Figure 7. Option 1 is used to program the startup codes.

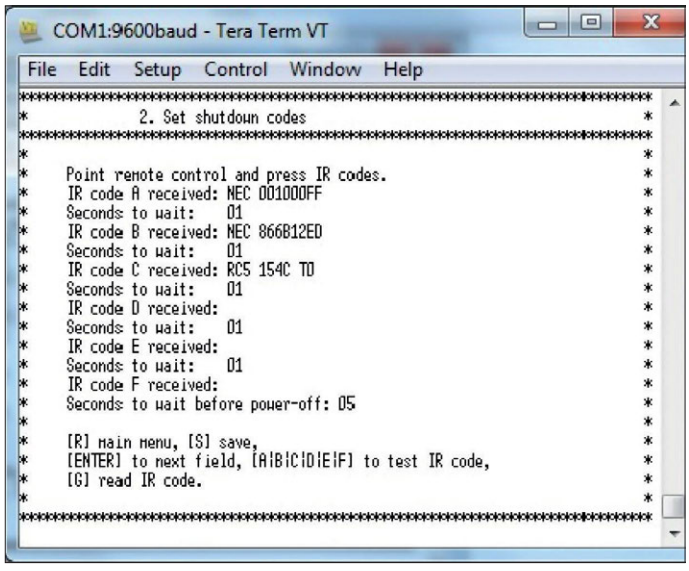


Figure 8. Option 2 is used to program the shutdown codes.

0.889 ms, whereas with Sirc it is 0.6 ms, see **Figures 4 and 5**. Once the type of IR protocol has been recognized, the software knows how the following data bits are coded. RC5 use RC6 use bi-phase coding, NEC uses pulse-duration coding and Sony uses pulse-width coding (you can read more about the various coding mechanisms via the Web Links shown at the end of this article, [2], [3], [4] and [5]).

Next, the microcontroller reads the IR data bits, puts them neatly into bytes and then stores them in the EEPROM memory. The transmitting of IR codes happens exactly the other way round. The first byte in the EEPROM memory contains the IR protocol, so the software knows how the data bits are coded and at which frequency they have to be modulated. The data bytes are then transmitted one by one in the same order as they were 'learnt'.

When IR codes are received, the carrier wave is removed by the IR receiver (SFH5110-36). This means that when the IR codes are transmitted via the IR LED, the IR code needs to be modulated using a carrier wave of 36, 38 or 40 kHz, depending on the type of IR protocol.
In an original Philips or Sony remote control, the IR code is repeatedly sent for as long as the button is pressed. The interval between two repeated IR codes depends on the type of the IR protocol. The Philips IR codes contain a toggle-bit. This bit remains the same for as long as the button is pressed. The toggle bit changes state after the button has been released and then pressed again.

NEC does it in a different way. The original IR code is transmitted just once, but then a IR repeat code is transmitted at fixed intervals for as long as the button is pressed. Some devices always expect a minimum of one IR repeat code before they will recognize the IR code. For this reason the repeat code is always transmitted at least once.
You can use a simple mobile phone camera to check that the IR codes are being transmitted. You should hold it a few cm away from the IR LED and straight in front of it. It works because

mobile phone cameras are sensitive to IR light, so you can see it on the screen. Note that an IR LED has a beam with a very narrow angle, so make sure that the camera is held directly in front of the IR LED.

Learning IR codes

Connect the Almost Universal Remote Control to a RS232 port on the PC using a null-modem cable (you may also use an RS232 to USB converter). Start a terminal program on the PC (such as Tera Term) and set the serial port to 9600 baud, 8 data bits, no parity and 1 stop bit (8n1). When power is applied to the Almost Universal Remote Control, you will immediately see a menu appear in the window of the terminal program (**Figure 6**). You should ensure that the IR receiver is in the shade, away from any light sources and flat screen TVs, before starting to 'learn' the IR codes. Light sources and flat screen TVs cause interference in the IR receiver, giving rise to possible errors when IR codes are being read.

In menu option 1 (**Figure 7**) the IR codes to switch the equipment on can be programmed and in option 2 (**Figure 8**) the IR codes to switch the equipment off can be programmed. In option 3 the Almost Universal Remote Control is 'taught' the various sources (**Figure 9**).

In menu-options 4 to 7 (**Figure 10**) the IR codes are programmed for the 'up', 'down', 'left' and 'right' buttons for each of the sources. Depending on the selected source, these four buttons can transmit a different IR code.

Press the 'G' key to program a new IR code (note that commands must always be in capital letters). Repeat the same IR code several times by pressing on the 'G' key again, and compare the received codes with the previous ones. The code that's been received most often is likely to be the correct one. Any changes can be saved using the 'S' key. The 'R' key is used to return to the main menu.

From the menu in option 8 the following functions can be set up:

- Select the number of times that an IR code is transmitted repeatedly. This option simulates the holding down of one of the remote control buttons.
- Turn the 'Auto power-on' function of the digital input on or off.

Day to day use

When the digital input is used, JP1 determines whether the start IR codes are transmitted when there is a logical high or logical low level at the digital input; the opposite obviously applies for the stop IR codes.

The 'Select' button can be used at any time to transmit the start and stop IR codes. To transmit the start IR codes, press briefly on the 'Select' button, and the relay switches on. After a programmable delay, the start IR codes are then transmitted. After the start IR codes have been transmitted, you can switch between the different sources. This is done by a quick press of the 'Select' button. The IR codes for the 'up', 'down', 'left' and 'right' buttons will change according to the source selected.

To switch everything off, hold down the 'Select' button for at least two seconds. The stop IR codes will then be transmitted, after which the relay will finally be turned off. ⏮

(160034)

Web Links

[1] www.elektormagazine.com/160034
[2] www.sbprojects.com/knowledge/ir/nec.php
[3] www.sbprojects.com/knowledge/ir/rc5.php
[4] www.sbprojects.com/knowledge/ir/sirc.php
[5] www.sbprojects.com/knowledge/ir/rc6.php

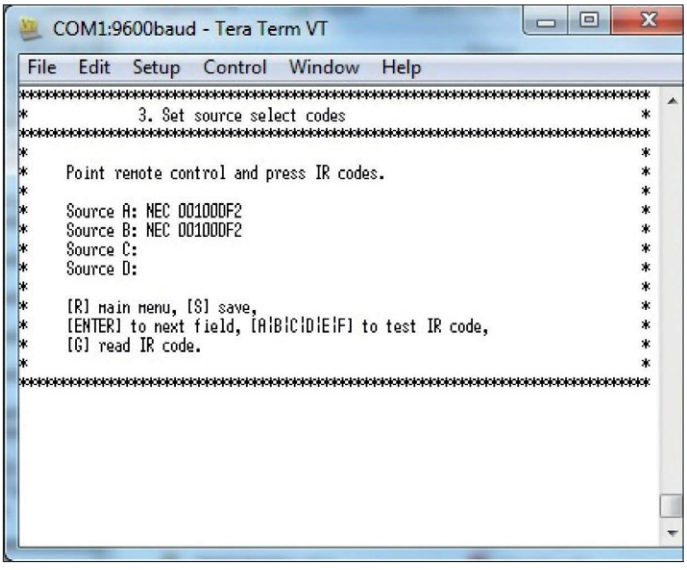


Figure 9. Option 3 is used to program the source selection codes.

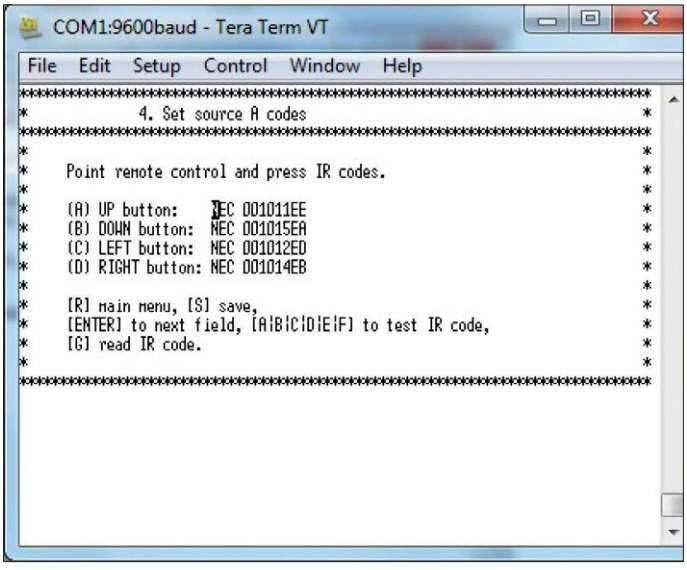


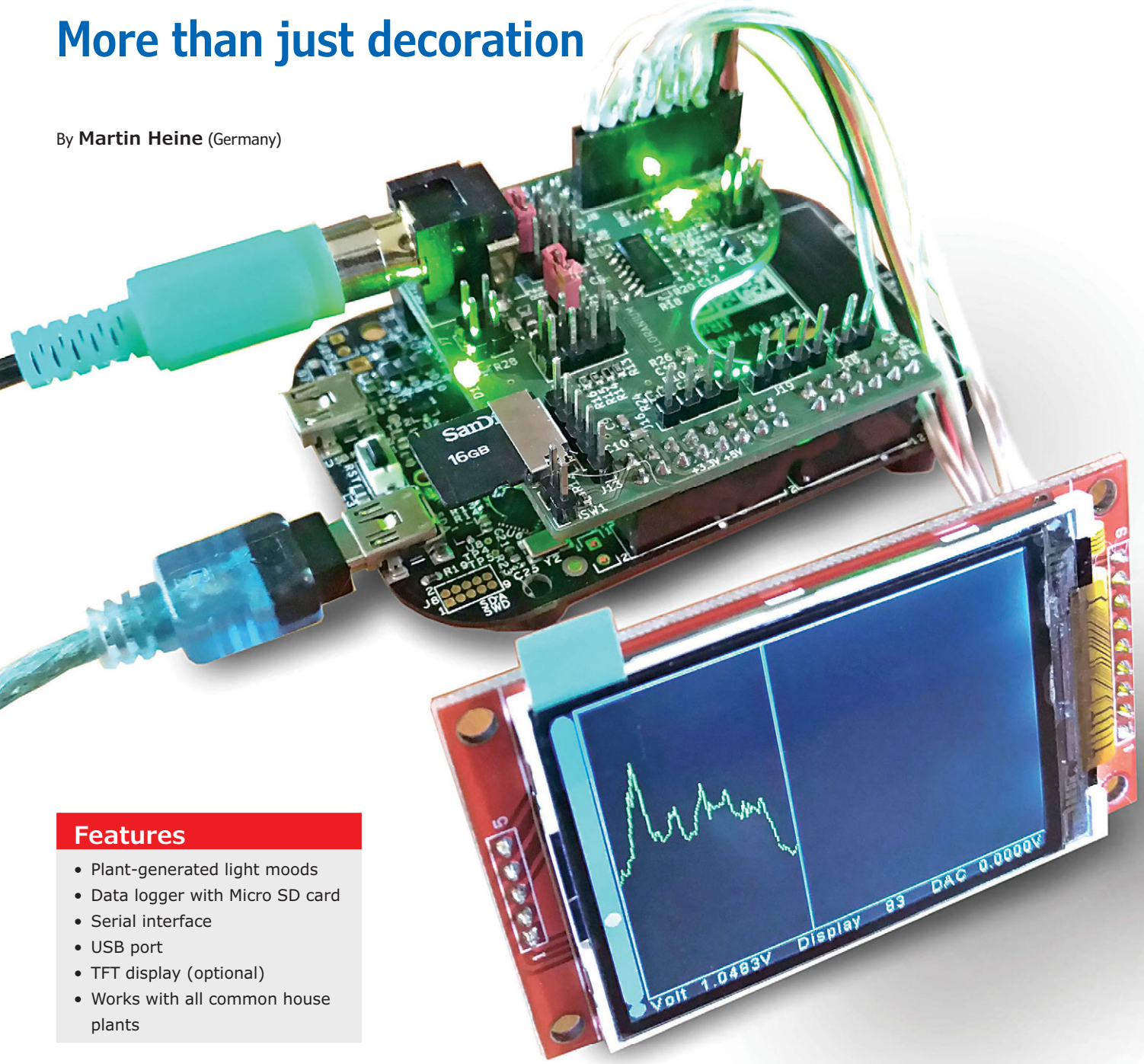
Figure 10. Via options 4 through 7 the arrow keys are programmed.

HOMELAB PROJECT

Floranium

More than just decoration

By **Martin Heine** (Germany)



Features

- Plant-generated light moods
- Data logger with Micro SD card
- Serial interface
- USB port
- TFT display (optional)
- Works with all common house plants

Shortly after publication of the Bio-Light article in issue 1/2018 of Elektor Magazine, we learned that a similar project has been available for many years: Floranium from Light Art Vision. That device is more than just a stylish plant light – it is also a complete and expandable system for measurements and experiments with plant physiology.

Plant signals are a very complex aspect of plant physiology. There are innumerable ways to observe the behaviour of plants, including the colours of the blossoms, emission of aromatic substances, biochemical changes, and electrical signals (action potentials). We can basically distinguish between two different types of activity: primary metabolic processes and secondary metabolic processes. The primary metabolic processes include photosynthesis, ion exchange at cell membranes (anions and cations), stimulus transmission via action potentials, and communication by electrical signals through the nutrient vessels (phloem). The secondary metabolic processes produce plant substances such as phenolic and isoprenoid compounds, alkaloids (caffeine, nicotine, etc.), and amino acids (alliin, canavanine, etc.)

Signals triggered by primary metabolic processes are very difficult to measure. That requires insertion of extremely fine electrodes directly into the nutrient vessels, and these electrodes must be able to withstand the moist conditions inside the plant without impairment. One example is the 'aphid bio-electrode' (that's not a joke). This involves connecting a very thin gold wire to a cabbage leaf aphid. The aphid wants to feed on the plant's nutrients, so it inserts its stylet into a nutrient vessel and starts sucking — thereby creating a direct electrical connection to the phloem. It is not known if the aphid notices that it is being used as an electrode.

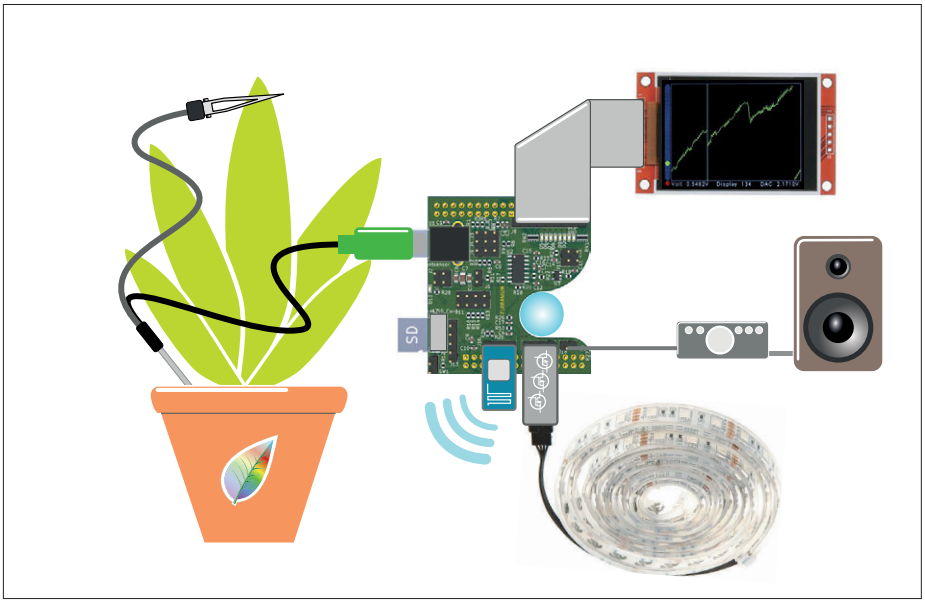


Figure 1. Overview of the Floranium system.

Skin resistance or galvanic potential?

When Cleve Backster carried out his famous (or infamous) experiments on his office plant (a philodendron) in 1966, he used a polygraph. A polygraph, commonly known as a lie detector, measures blood pressure, heart rate, respiration rate and skin resistance. If a person being interrogated lies, they perspire slightly and nearly imperceptibly. That lowers the skin resistance due to the conductivity of the salts and minerals in the perspiration. A plant also 'perspires' by exuding moisture through openings on the bottom of its leaves. This moisture contains phenolic and isoprenoid compounds, alkaloids and amino acids (secondary metabolic products), making the plant a sort of living electrolytic capacitor. The resistance or impedance can be measured, and there is a galvanic potential between the metal electrodes, the same as in a battery. The best example of this is the well-known lemon or apple clock, with a quartz clock powered by the electrochemical voltage between a zinc nail and a copper nail pushed into a lemon or an apple.

Plants are living organisms, and living organisms respond to their environment in order to give them an advantage (more sun, more water, more nutrients) or protect them against predators. That is accompanied by changes in the ion exchange and the primary action potential, which in turn stimulates secondary effects. For example, if a plant is attacked with a burning match, it responds with increased resin production so that any burn wounds can be closed quickly. That alters the impedance between the leaves and the soil,

The evolution of Floranium

Inspired by the book *The Secret Life of Plants* by Christopher Bird and Peter Tompkins as well as the experiments by Cleve Backster, in 1981 Martin Heine set about converting an old dishwasher into a measuring chamber for plants. At the same time he started experimenting with highly sensitive instrumentation amplifiers and resistance bridges in order to personally explore plant signals. Despite working in other areas after completion of his university studies in electronics, he never entirely lost interest in plant signals. In 2007 he had the inspired idea of converting plant signals into a play of colours from a small lamp. His prototype led to a small circuit that fitted in a tea light holder. He named his invention 'Floranium' [6]. A number of Floranium variants (called Floranions) arose over the course of time. They are produced in the Light Art Vision Studio in Cave Creek, Arizona (USA). The colourful lights of the plants in the film "Avatar" not only correspond exactly to the function of the Floranion, but are actually the avatars of the associated plants. It is therefore hardly surprising that the Floranions are being marketed in cooperation with the film producer as Avatar Lights or Avatar Lamps and a Floranium Avatar Edition has been created.

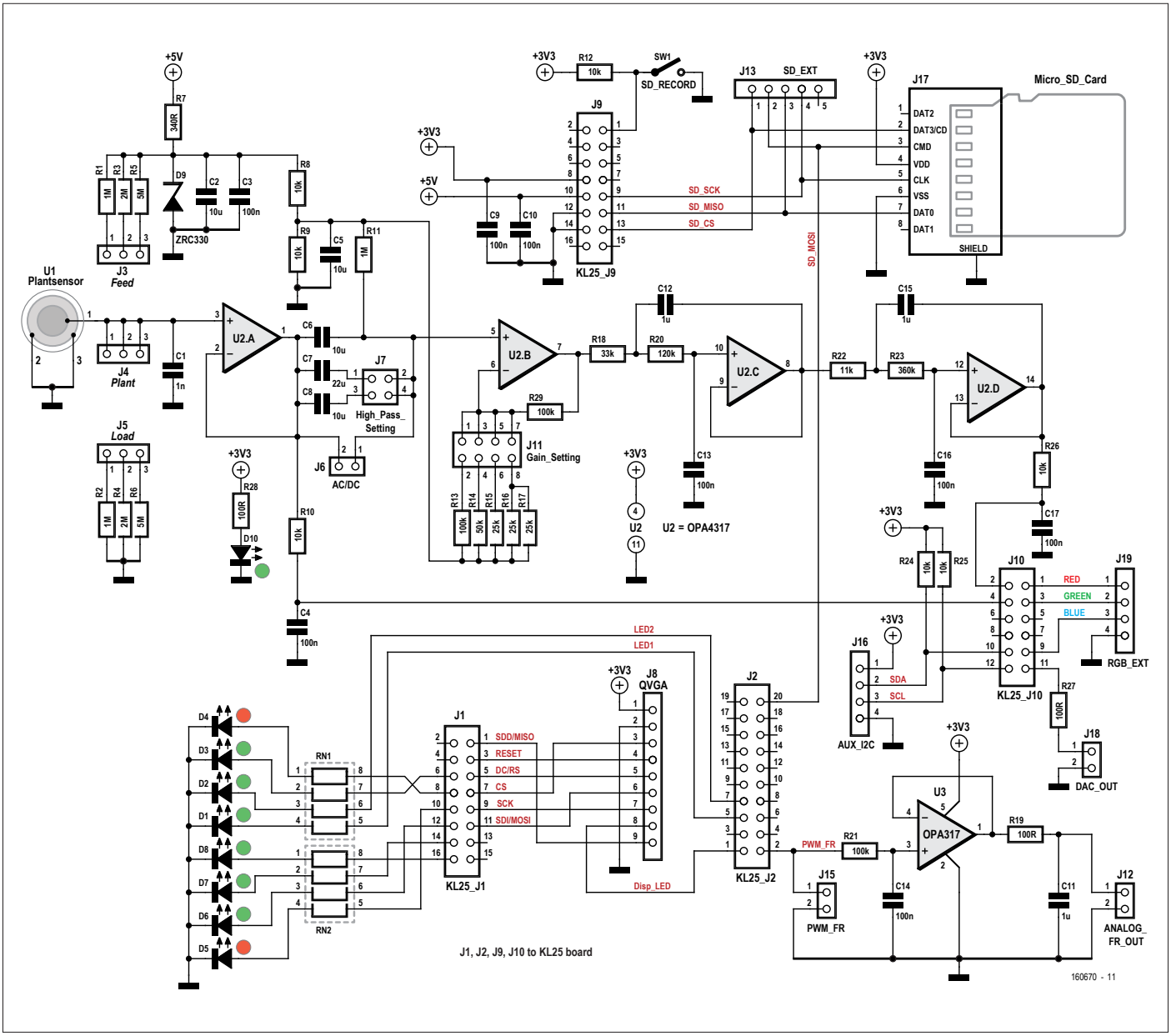


Figure 2. The analogue circuitry on the Floranium shield.

with a superimposed galvanic potential. The connected Floranium measures the sum of all these superimposed signals.

A shield for the microcontroller board

The Floranium described here is a plug-on board (shield) for a KL25Z microcontroller development board from NXP (formerly Freescale). **Figure 1** shows the general layout of the system. The plant electrode provide the input signal that is transformed into acoustic or visual signals by other devices, which are more or less optional: a graphical display, an audio system, RGB LEDs and RGB LED strips. The signal data can be saved on an SD card or in the cloud over a wireless link, and it the signal can be evaluated directly using measuring equipment.

First let's look at the core element of the system: the microcontroller board [1]. The ARM Cortex M0+ microcontroller is clocked at 48 MHz and has an integrated 16-bit SAR ADC, several PWM channels, a 10-bit DAC, and several other interfaces, including USB, serial (UART), SPI and I^2C. A bright RGB LED, a capacitive touch sensor, and additional user-accessible I/O lines are present on the board. The microcontroller board is equipped with a USB flash programmer and is ARM mbed enabled. An online compiler facilitates program development, with broad support by an online development forum. The only hardware items needed for programming are a USB cable and a PC with a web browser.

The circuitry on the shield (see **Figure 2**) is largely analogue and fairly sophisticated. The input signal changes very slowly and drift in the measurement circuitry should be avoided as much as possible, so filtering and amplification of the plant signal are performed by the four opamps of an OPA4317, which is advertised as having zero drift, rail-to-rail input and output, and precision performance.

The plant sensor signal U1 is input to opamp U2A, which is configured as an impedance converter to provide the highest possible input impedance for the signal to be measured. A pull-up resistor for

resistance measurement (galvanic skin response / skin conductance, abbreviated GSR/SC) can be selected with J3 and J4. That makes the plant the second resistor of a voltage divider. A pull-down resistor to act as a load for voltage measurements can be selected with J4 and J5.
In GSR/SC mode (pull-up resistor), the bandgap voltage reference ZRC330 (D9) provides a stable bias voltage source. It is connected to +5 V through series resistor R7.
At the output of the impedance converter U2A there is a simple high-pass filter. Its time constant can be adjusted by connecting capacitors in parallel through jumper J7. This filter can be used to mask slow drift in the plant signal, similar to the AC input of an oscilloscope. That is mainly necessary when the plant signal needs to be amplified.
The second opamp U2B amplifies the signal. The gain can be set by jumpers on J11. A fourth-order Sallen-Key filter built around opamps U2C and U2D suppresses any 50 Hz mains noise. The filter output is connected to analogue input A0 on the KL25Z microcontroller board to feed the plant signal to the 16-bit analogue to digital converter.
A total of eight LEDs are connected to headers J1 and J2 of the KL25Z to provide a dot-bar display of the voltage level of the plant signal. That is convenient for checking that the plant is properly connected and that the signal level is within the measuring range. The dot bar shows the voltage level (0–3.3 V) at the output of the impedance converter opamp U2A. This allows the absolute value of the voltage to be displayed even if the high-pass filter has been inserted in the signal path by removing jumper J6.
The RGB LED on the KL25Z board is not bright enough to light up a room or similar area, so three PWM channels are routed to socket header J19 for driving external RGB LEDs or LED strips.
Another PWM channel is fed out to J15. It outputs the signal voltage (0–3.3 V) as a PWM signal. A downstream low-pass filter and impedance converter (U3) convert this PWM signal into a DC voltage that can be connected to other devices, such as an analog chart recorder or a voltmeter with data logging.
The measurement data from the plant can also be recorded by the integrated data logger function. A Micro SD card holder (J17) is connected to the SPI interface of the KL25Z board (J9) for this purpose Data recording on the SD card can be started and stopped with switch SW1.
For local display of the plant signal as a curve, similar to an oscilloscope, a TFT colour display with QVGA resolution (320 × 240 pixels) can be connected to the SPI bus on J8. The output of the 10-bit DAC of the KL25Z is fed out to J18. If you can think of some way to get plants to make pleasant sounds, you can use this interface for that.

RGB colour wheel as an intuitive signal indicator

Visualisation of variations in the plant signal by variations in the colour of the light emitted by an RGB LED can be implemented without any need for offset

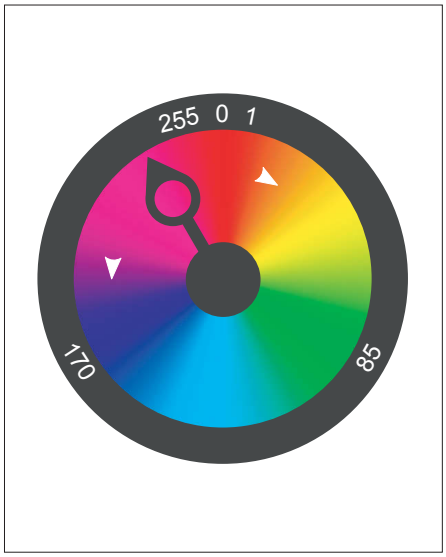


Figure 3. Plant signals on the colour wheel.

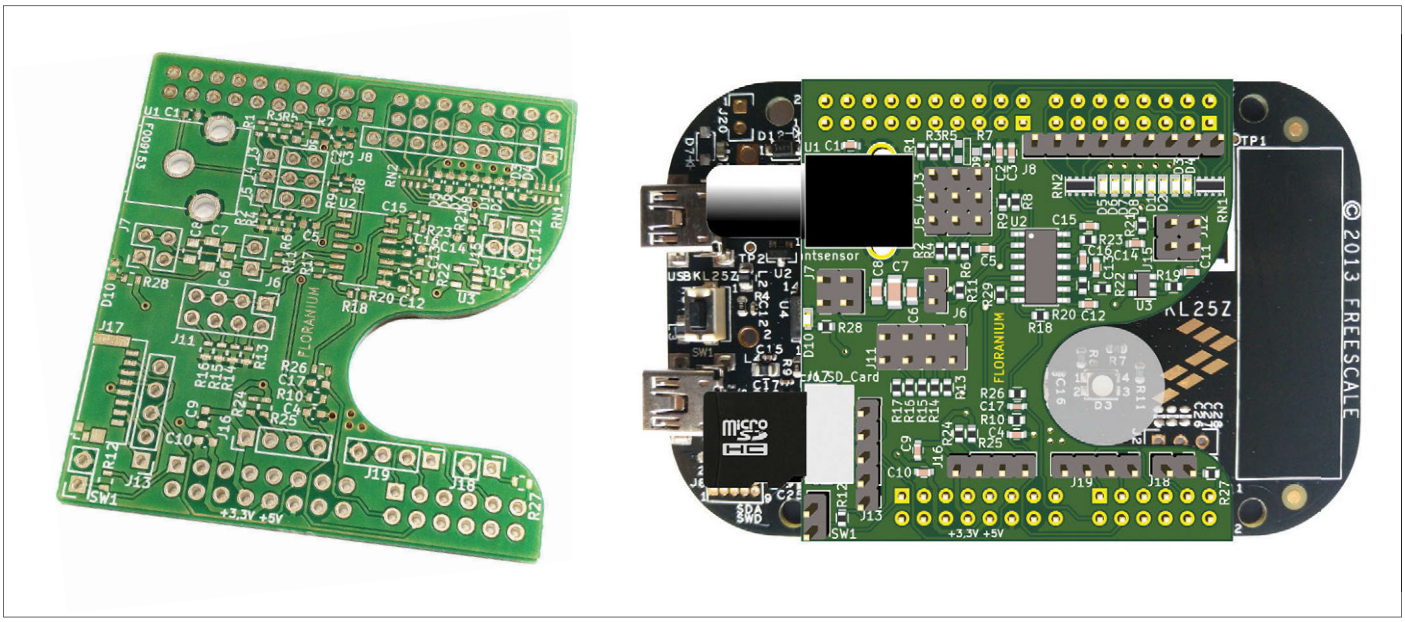


Figure 4. The shield board is shaped to reveal the RGB LED and important control elements of the microcontroller board.

compensation or signal amplification. The only requirement for this is a high-resolution analogue to digital converter, such as the 16-bit successive approximation (SAR) ADC of the MKL25Z microcontroller. The voltage measurement resolution with a reference voltage of 3.3 V is:

$3.3\ V \div 2^{16} = 50\ \mu V$

If the colour of an RGB LED is determined by an 8-bit signal, then 256 steps are needed for a full circuit of the colour wheel from red through yellow, green, cyan, blue, magenta and back to red (see **Figure 3**). This means that a full colour cycle corresponds to a measured voltage of:

$256 \times 50\ \mu V = 12.8\ mV$

If the lower 8 bits of the 16-bit ADC output are masked and the RGB LED PWM signal is controlled such that in each third, two primary colours mix in equal parts with a continuous gradient, the result is exactly this colour cycle that repeats 256 times between 0 V and 3.3 V (8-bit MSB):

```
if(plantRGB >= 0 && plantRGB <= 85
   ) { red = 1-(color*3); green =
   color*3; blue = 0; }

if(plantRGB > 85 && plantRGB
   <= 170) { red = 0; green
   = 2-(color*3); blue =
   (color*3)-1; }

if(plantRGB > 170 && plantRGB <=
   255) { red = (color*3)-2; green
   = 0; blue = (1-color)*3; }
```

This way even very small relative changes in the plant signal can be readily recognised, independent of the absolute signal level.

All on board

The PCB of the Floranion shield shown in **Figure 4** is designed with a contoured cutout to provide access to the reset button, RGB LED and touch pad on the KL25Z board. To enhance the visibility of the colour display from the LED, a small disc of opal acrylic can be glued into the notch. The jumper headers for the pull-up and pull-down resistors, high-pass filter and amplifier gain are located right next to the Cinch jack for the signal cable. The SD card slot is located at the lower left for easy access, with the LED dot bar display at the upper right and above it the connector for a TFT display module. Jumper J15 for the Fullscale PWM and jumper J12 for the Fullscale Analogue Out signals can be seen below the LED dot bar display. At the bottom there are connectors for external RGB lamps or LED strips, DAC Out and I²C. That allows other modules to be connected, such

Table 1. Binaries of various software versions and their features.

Firmware	Version	RGB	SD	TFT	TxD	DAC	Description
floranionShieldArt100_KL25Z.bin	1.00	X	–	–	–	X	Only RGB and external RGB. For lamps, lights, artworks and decoration
floranionShieldSD100_KL25Z.bin	1.00	–	X	–	X	–	Stand-alone data logger. RGB LED eliminated to reduce power consumption
floranionShieldTFT100_KL25Z.bin	1.00	X	X	X	X	X	For operation with TFT QVGA display and data logger function
floranionShieldSci100_KL25Z.bin	1.00	X	X	–	X	X	For operation without TFT display but with data logger function, RGB, DAC and TxD

Figure 5. A Faraday cage can also be attractive.

as a temperature sensor, a radio time signal receiver for precise time stamps, or a wireless module for remote operation. Among other things, that could be used to transmit plant signals directly from the forest to the cloud. The only limit here is your imagination. The project download [2] includes KiCAD PCB design files and detailed lists of jumper settings and connector pinouts.

Programming aspects

The Floranion shield is programmed in C. The source code is free (under GPL3) and is available on the ARM mbed website [3]. The code is compiled with the ARM mbed online compiler, which is compatible with the GCC open source compiler. An offline toolchain, such as the GNU ARM embedded toolchain from ARM, can also be used for compilation. To avoid the need for compilation, four precompiled firmware binaries (see **Table 1**) are available in the project download [2] on the Elektor website.

After fitting pinheaders on the KL25Z board, you can plug the Floranion shield onto the board. Before installing the Floranion firmware, the mbed firmware for the KL25Z board must be installed. This process is described in great detail on the ARM mbed developer website [4]. When you connect the board to your PC with a USB cable, it should be recognised as drive MBED.

Then you can download the Floranion firmware to the board by simply dragging the corresponding file to the MBED drive. The green LED of the KL25Z will blink during the download process. Once it stops blinking, just press the Reset button. Now your Floranium system is ready to run, and you can start your voyage of discovery in the sensory world of plants.

Connecting to the plant

To avoid having you plant start screaming in pain right away, it goes without saying that as a true plant lover you would never stick a needle in a stem or crush a leaf in an alligator clip. Always connect the electrode very gently so it just grasps the leaf. A good ionic connection to the inner part of the leaf can be achieved with a curved hair clip and a drop of conductive gel as used by doctors for EEG pads — a tried and tested solution. For connection to the soil, either outdoors or in a flower pot, a normal galvanised nail can be used.

The secret life of plants

The idea that plants are living organisms with feelings is probably as old as human thought. In 1848 the German psychologist, physicist and natural philosopher Gustav Theodor Fechner published the book *Nanna oder über das Seelenleben der Pflanzen* (Nana or the Souls of Plants) in Leipzig and espoused the idea that the entire universe is imbued with soul. The Indian scientist Jagadish Chandra Bose dedicated himself fully to the physiology of plants from 1900 onward and did pioneering work in that field. He reported on the influence of electromagnetic fields on plants and developed instruments for observing the growth of plants.

In 1966 the lie detector specialist Cleve Backster connected a polygraph to a plant to see what would happen. He watered the plant and touched a leaf, but the plant did not show any response. Then he wanted to singe a leaf with a lit match. But as he reported, when he held the match in his hand and was just about strike it, the polygraph went wild. After that he dedicated his life to plants, and in 2003 he published the book *Biocommunication with Plants, Living Foods, and Human Cells*.

Today the responses of plants to external influences are being investigated and studied by numerous projects, researchers and scientists. One example is the EU-funded project PLEASED, led by Andrea Vitaletti at the University of Rome. Research is also being carried out by another group at the University of Freiburg headed by Dr Edgar Wagner. Now plants are changing the colour of light (Floranium), making music (Damanhur), and controlling greenhouses as part of the ZINEG project. Nevertheless, the subject is still controversial and derided by many scientists.

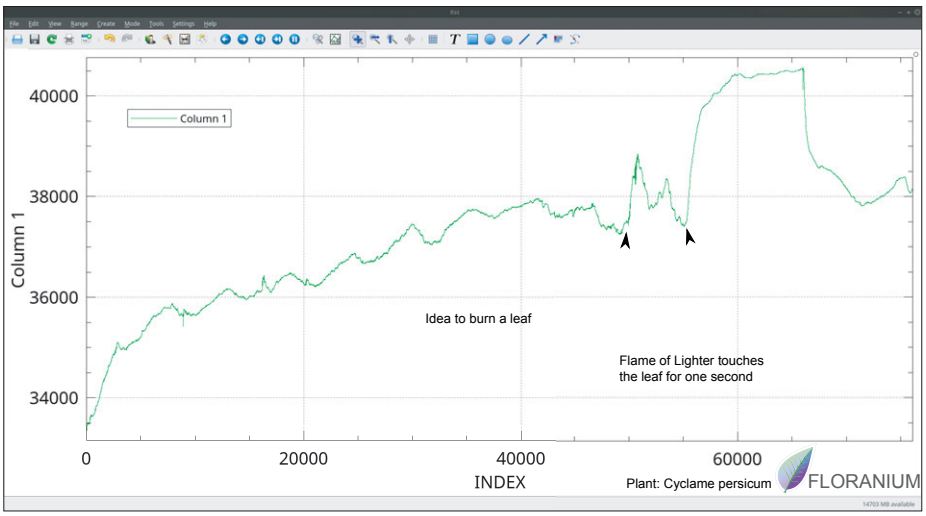


Figure 6. A cyclamen reacts to a fire attack.

After the power is first switched on, the RGB LED will slowly pulsate in blue. When the plant is connected the colour will start changing. These colour changes usually occur in rapid sequence at first. This is because an electrochemical reaction occurs at the electrode when it initially comes in contact with the moist leaf, but that will subside after a while. To avoid disturbances from electrostatic charges, it is recommended to place the plant inside a Faraday cage as shown in **Figure 5**. Not nice for the plant, maybe, but in its best interest.

Data visualisation with KST

The open source program KST [5] is a very attractive tool for real-time data visualisation and analysis. If the serial data from the Floranion transferred via the USB port (SDA) of the KL25Z is saved to a file (for which you can use the "Log to file" menu command or equivalent available in almost every terminal emulator program), you can open that file with the KST data wizard. Then KST will plot the data as a curve in real time. **Figure 6** shows an example of this sort of curve with data from an cyclamen that was threatened with the flame of a cigarette lighter.

Unfortunately the response of the plant cannot be reproduced at any given time. Cleve Backster was also confronted with that issue, which is why his research has repeatedly been ridiculed. Nevertheless, the potential for using plants as natural biosensors is immense.

TFT display and enclosure

Another way to visualise the plant signals is to connect a TFT display to the KL25Z shield – for example, a QVGA TFT display module with an ILI9341 display driver, for which numerous free libraries are available.

To allow all that to be packaged nicely, the project download includes an STL file with an enclosure model for a 3D printer. As the Micro SD card is no longer accessible after the boards are installed in the enclosure, an alternative is to use the SD card slot on the rear of the display module. For that you only need to solder a five-lead connecting cable between the two SD card connectors (on the Floranion shield that is J13, SD_EXT). Switch SW1 for starting and stopping data recording must also be repositioned to the enclosure. The opening in the enclosure directly above the slot for the SD card will hold a slide switch measuring 15.5 × 7.5 × 7.5 mm. There is also a small hole provided on the front of the enclosure for mounting the Reset switch. ⏮

(160670-I)

About the author

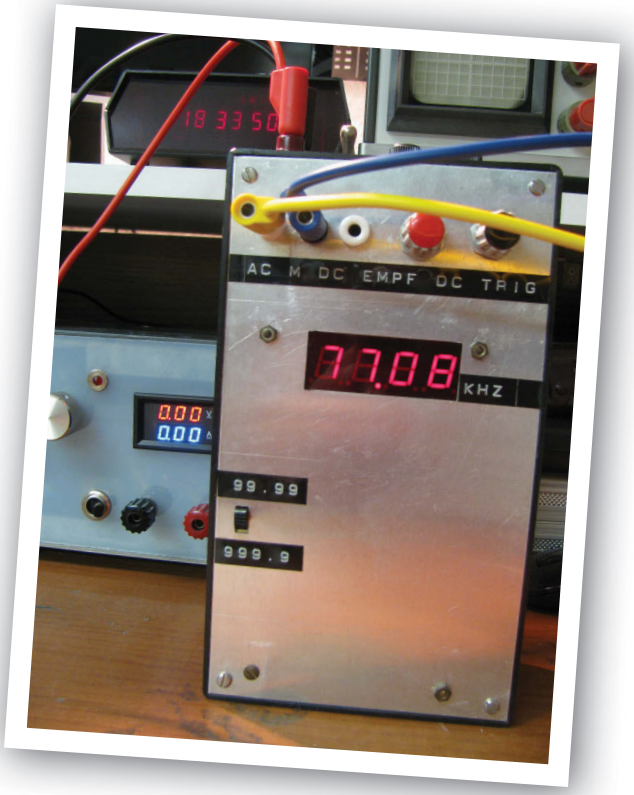
Martin Heine M.Sc. studied electronics at the Reutlingen University of Applied Sciences, where he received a Master's degree in Computer Based Engineering (CBE). As a freelancer he worked on numerous research projects of the University of Stuttgart, the German Aerospace Centre (DLR), and the Denkendorf Institute of Textile and Fibre Research (DITF). During his many years in the USA, he developed antennas and RF circuits for Valor Enterprises, Laird Technologies, Hirschmann Automotive, and Saunders & Associates (S&A). After returning to Germany, he first worked at Rohde & Schwarz on military Software Defined Radio (SDR) technology, followed by RAFI, and developed magnetic-bearing high speed drives for S&A (Saunders & Associates). His company Light Art Vision produces sensors for biosignals (Floranium) and ionising radiation. One of his products was honoured with the "Otto van Guericke" innovation prize from the AiF.

Web Links

[1] https://os.mbed.com/platforms/KL25Z
[2] www.elektormagazine.com/160670
[3] https://os.mbed.com/users/lasmahei/code/floranionSciShield/
[4] https://os.mbed.com/handbook/Firmware-FRDM-KL25Z
[5] https://kst-plot.kde.org/
[6] www.floranium.com

The Elektor Mini Counter

The frequency counter that thought it was a pocket calculator



By **Karl-Ludwig Butte** (Germany)

Just shy of forty years ago, Elektor published three frequency counter project articles in the June 1978 issue. The 'µP counter' concept was forward looking. For professionals there was the ¼ Gigahertz Counter, and for hobbyists the Mini Counter. Readers could purchase ready-made PCBs for both of these projects from the Elektor Printed-circuit Service (EPS).

Just imagine: your own frequency counter! In my mind's eye I could already see the six bright red 7-segment displays of the ¼ Gigahertz Counter glowing on my hobby bench. But after reading through the articles, I quickly realised that the Mini Counter would already put a significant dent in my limited budget. In my mind's eye, two of the six 7-segment displays went dark — because the Mini Counter only had four of those fascinating display modules. I consoled myself with the thought that I would probably never build a device that could operate at a frequency of ¼ gigahertz (250 MHz!). But I absolutely wanted to build the Mini Counter.

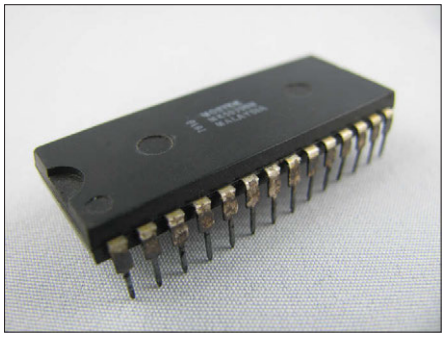


Figure 1. The Mostek MK50398N.

The Mostek MK50398N six-decade counter and display decoder IC

The ¼ Gigahertz Counter and the Mini Counter were both based on the MK50398N LSI counter IC from Mostek. The abbreviation 'LSI' stands for 'Large Scale Integration', which refers to the large number of transistors integrated into the IC. Mostek was founded in 1969 by former Texas Instruments employees. The first IC produced by Mostek was a simple shift register. It was followed by ICs for pocket calculators and telecommunications. However, Mostek's most successful product — with a market share of 85% — was DRAM ICs for the nascent microcomputer industry. In the late 1970s Mostek fell victim to the DRAM price dumping campaign of the Japanese semiconductor industry, and after passing through several hands it ended up as part of STMicroelectronics [1].

But let's get back to the MK50398N (**Figure 1**). This IC contained a six-digit BCD up/down counter and was available in two versions. The MK50398N was designed to drive six 7-sgment displays, while the MK50399N had six BCD outputs. That was highly attractive for circuit designers because they got not only virtually all of the counter circuitry, but also the drive circuitry for the display unit in a single 28-pin package. All that was necessary to connect the 7-segment displays was a suitable driver IC (e.g. the CD4049) and series resistors for the individual segments. A similar approach was already popular in other areas and with other manufacturers. For example, the ICL7107 from Intersil provided a complete 3½-digit A/D converter with integrated LED display driver, allowing a complete digital voltmeter to be built with minimal effort and expense.

The MK50398N had an internal oscillator, but it could also be used with an external oscillator. The IC also had a six-digit latch register to hold the displayed count and a multiplexer for the display modules. Further technical details can be found in the data sheet available at various places on the Internet, such as [2].

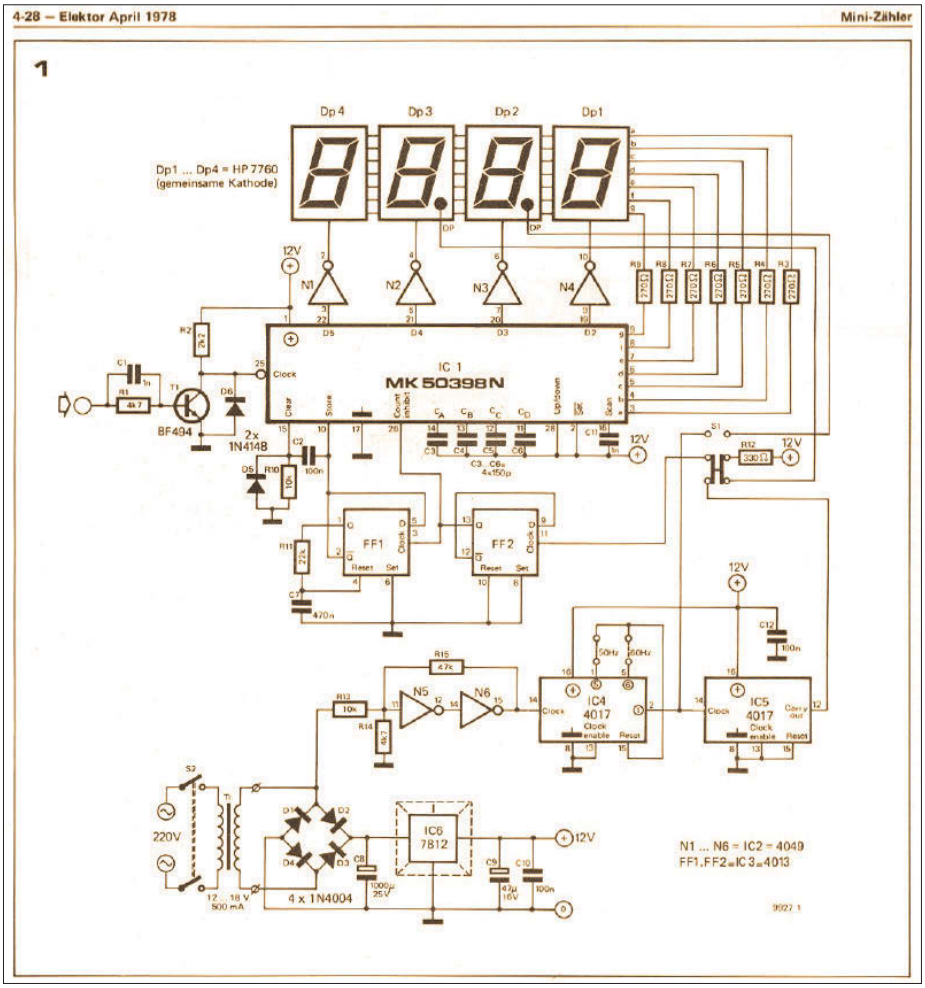


Figure 2. Schematic diagram of the Mini Counter from the April 1978 issue of Elektor magazine.

The Elektor Mini Counter

Figure 2 shows the circuit diagram of the Mini Counter. To start with, the MK50398N needed a power source. That was implemented very conventionally with a transformer, a full-wave rectifier and a 7812 voltage regulator, along with the customary filter capacitors. In my opinion I could do without the transformer — after all, I had an adjustable lab power supply sitting on my bench. That saved me a few Deutschmarks (DM).

However, that part of the circuit did more than just provide the supply voltage. The counter gate period signal was derived from the 50-Hz AC line frequency, saving the cost of a more accurate but more expensive crystal oscillator. A Schmitt trigger made from two CD4049 inverters converted the AC line frequency signal into a CMOS-compatible pulse waveform, which was then divided down to 10 Hz and 1 Hz in two stages with the aid of two CD4017 decade counter ICs. One of resulting 10 Hz or 1 Hz signals was selected by switch S1 and applied to the clock input of the D-type flip-flop FF2, which together with the second D-type flip-flop FF1 generated the control signals for the Count Inhibit, Store and Clear inputs of the MK50398N. The Q output of FF2 first went low for 100 ms or 1 s. Then the output went high, toggling FF1. The Q output of FF2 also applied a high level to the Count Inhibit input of the MK50398N, stopping the count process. At the same time, the /Q output of FF1 pulled the Store input of the counter IC low. That allowed the current count to be transferred to the latch register, from which it was shown on the display. Capacitor C7 on the FF1 Reset input caused FF1 to be reset a short time later; FF1 effectively acted as a monostable multivibrator. It also produced a high level on the Clear input to reset the internal count register. That did not affect the internal latch register, so the latest count was always visible on the display. Then the next count cycle started.

In addition to selecting the 10 Hz or 1 Hz clock signal for FF2, switch S1 controlled the decimal points of the 7-segment display modules. That gave a measuring range of either 0 to 99.99 kHz or 100.0 to 999.9 kHz.

Finally there was the input stage that fed the test signal to the counter IC. It basically consisted of a single BF494 transistor (T1) and its protection diode D6. With that the counter could measure signals of 1 V or above. I thought to myself: surely that can be improved.

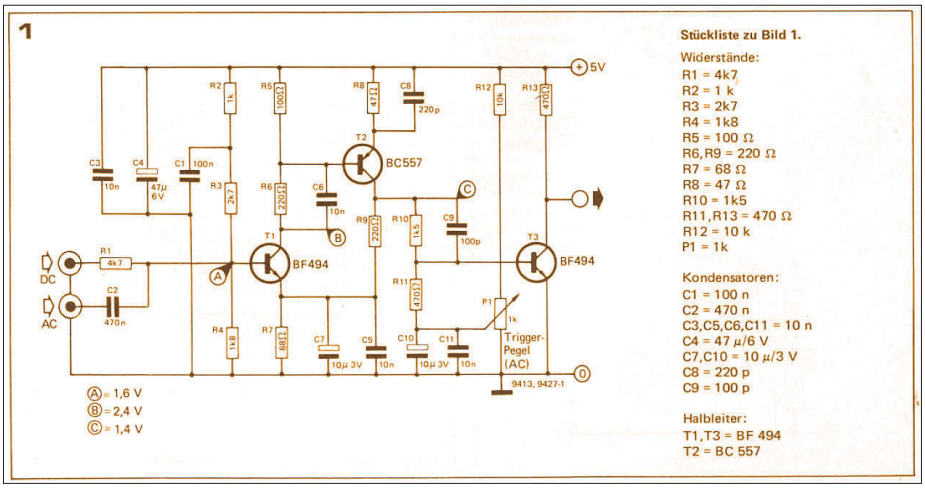


Figure 3. Schematic diagram of the preamplifier from the June 1976 issue of Elektor magazine.

The preamplifier

For another frequency counter project published in the November 1975 issue, Elektor developed a preamplifier with a type E420 dual FET, which was published in the German December 1975 issue. This dual FET was difficult to obtain, so a new preamplifier design was published in the June 1976 issue. It had even better sensitivity and bandwidth than the previous design and used only

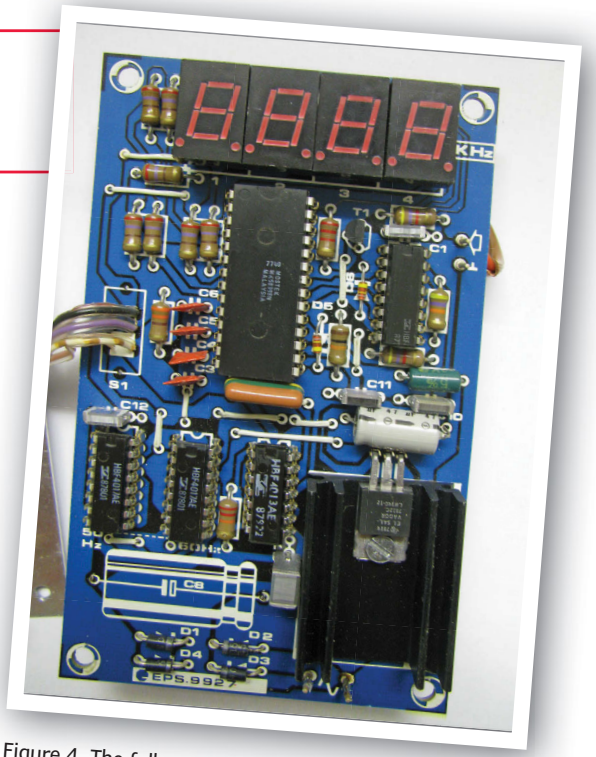


Figure 4. The fully assembled Mini Counter circuit board.

commonly available components. That preamplifier boosted the input sensitivity to approximately 4 mV. **Figure 3** shows the schematic diagram of the preamplifier (with thanks to Editor Jan for sending me a copy of the article).

Construction

After acquiring an MK50398N, four 7-segment displays, an enclosure and all the other components needed for the Mini Counter and the preamplifier, I decided to buy the ready-made PCB from Elektor to make sure that nothing could go wrong. That's because my prior experience with etching my own boards had been disappointing.

Mounting the components on the board was easy — there weren't any SMD components then ;-). **Figure 4** shows the fully assembled circuit board. However, installing the board into the enclosure was more difficult. The 7-segment displays were mounted directly on the main PCB, so everything hat to be fitted on the rear of the front panel. Only the jacks for the power supply, a hefty power switch and two fuse holders were mounted on the top of the enclosure. That way only a few wires were needed between the front panel and the rest of the enclosure. **Figure 5** shows the internal construction. A nice benefit of not using a power transformer was that I did not have to deal with hazardous (then) 220 V AC inside the enclosure.

Now everything was ready for the first test. I connected the Mini Counter to my lab power supply, switched it on, and was rewarded with a lovely bright red 00.00 display. For the first functional test, I decided to keep things simple and measure the frequency of a 1-V, 50-Hz sine-wave signal from a transformer connected to my lab power supply. Right after I connected the leads, the display changed to 00.05. Hurrah! It worked! But what was going on? Right after that I saw the display change to 00.10, and then to 00.15. It increased by 50 Hz every second. How could that happen? Did I build a pocket calculator instead of a frequency counter? I quickly disconnected everything and took the device apart. I checked and double-checked every component and every solder joint. Everything looked okay. I read the article again to see if I had missed some detail, but found nothing. There was no explanation for this strange behaviour. I simply couldn't understand it.

It seemed like the only remedy was to call the editorial office of Elektor. So that's what I did. They said, "Well, did you see the *Missing Link* item in the latest issue?"

Oops — no. I had missed the correction because I did not have a subscription at that time.

It turned out that there was a typo in the value for C2, and with the listed incorrect value of 100 nF (0.1 µF) there was effectively no Clear pulse for the six-digit BCD count register of the MK50398N. As a result, it added each new count to the previous one instead of starting at zero for each cycle. After C2 was replaced by a capacitor with the right value (150 nF/0.15 µF), everything worked the way it should — and it still does now, 40 years later. You can see that from the lead photo.

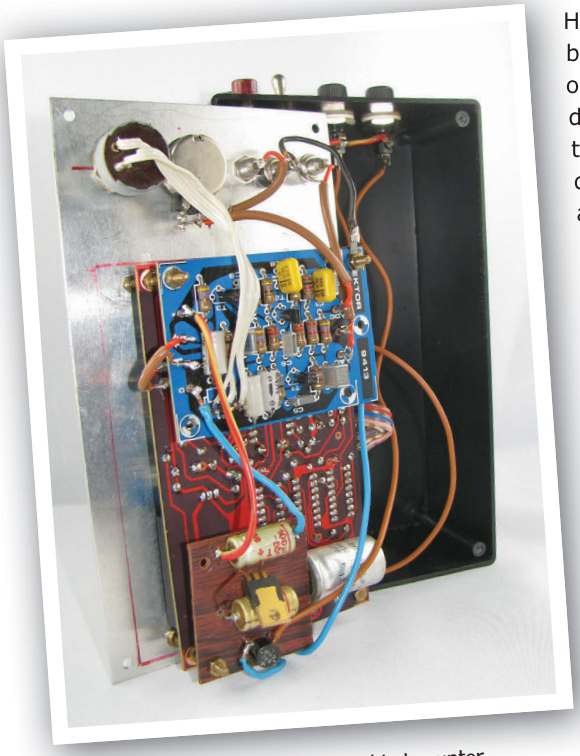


Figure 5. Inside view of the assembled counter.

Lessons learned

As everyone says, you learn from your mistakes, and from that mistake I learned to pay attention to corrections, then called *Missing Links* in Elektor. Since then I have manually copied every published correction and update to the corresponding original article, to avoid incorporating known errors into the projects I build. ⏮

(160657-I)

Web Links

[1] http://en.wikipedia.org/wiki/Mostek
[2] http://datasheet.octopart.com/MK50398N-Mostek-datasheet-11127392.pdf

ELEKTOR ETHICS

Our Vulnerable Digital Society

Is attack the best form of defence?

By **Tessel Renzenbrink** (Netherlands)

A dispute over a monumental statue led to what has come to be recognised as the world's first cyber war. In 2007 the government of Estonia relocated the "Bronze Soldier" war monument from a prominent location in the centre of the capital city, Tallinn, to a military cemetery. Many Estonians found the statue irritating because it reminded them of the Soviet occupation, which ended in 1991. However, the Russian-speaking minority felt insulted by the relocation, and Moscow was not amused. The conflict escalated on 26 April 2007, with public disturbances on the streets of Tallinn. The first cyberattacks began the next day.

Image: Wikimedia Commons

Distributed denial of service (DDoS) attacks effectively shut down large parts of the Internet. Many government services were inaccessible, the phone system stopped working, news media found it very difficult to reach the population, and payment transactions came to a standstill. The attacks persisted for over three weeks. Although the Estonian government immediately blamed Moscow for the attacks, the involvement of the Russian government or any other government has never been proven.

"Every country should have a cyber war"

The Estonians have made the most of their misfortune. As the first country to experience the vulnerability of digital infrastructure, they have developed into a frontrunner in cybersecurity [1]. The Estonian national cybersecurity strategy is regarded as a sterling example for countries that are now trying to catch up, In an interview with the *Quartz* news site [2], Taavi Kotka, former Chief Information Officer of the Estonian government, said "Every country should have a cyber war. That way citizens learn what an attack means, what phishing is, and how DDoS works. They start to understand it and learn how they have to deal with it."

It looks like Kotka was right: attacks on the power grid led to blackouts in the Ukraine in 2015 and 2016, British hospitals had to close down after being hit by WannaCry ransomware in 2017, and DDoS attacks briefly blocked financial and trade services in the Netherlands early this year. Now that critical infrastructure is increasingly falling victim to digital attacks, there is a growing awareness in other countries as well that online security is vitally important.

Attack methods

It's high time for that — for many years, experts have been warning that Internet security is in a sad state. Now that there is increasing willingness to take action, it's remarkable that so much attention is being given to enhancing methods of attack. The powers of investigative

and security services are being extended to give them more offensive capabilities. To mention a few examples: security services with unrestricted permission to monitor and save Internet traffic; police services allowed to carry out counterhacking, even if it goes via third parties that are not under suspicion (so-called stepping stones); intelligence services allowed to keep zero-day exploits to themselves — which means not reporting security gaps in software, but instead keeping them secret in order to penetrate the computer systems of their opponents, whereby software developers are not able to correct the flaws and all users of the software remain vulnerable.

This attack map from the Norse Corporation (USA) shows cyberattacks in real time (http://map.norsecorp.com).

But is attack the best form of defence?

Cooperation and sharing knowledge

If you examine Estonia's renowned cybersecurity strategy, you see little sign of offensive capabilities [3]. The Estonian approach is based on cooperation and sharing knowledge. "We realised that the old dividing lines between domestic and international conflicts, defence and security problems [...] do not hold up in cyperspace", wrote Jaak Aaviksoo, Minister of Defence at the time of the cyberattacks in 2007, in a publication [4]. "No single ministry or department can handle what is simultaneously a problem of infrastructure, defence, law enforcement, commerce, and civil liberties. Furthermore, 85% of web infrastructure is in private hands. Therefore, 80% of cyberattacks are launched against private companies, NGOs, and individuals. These challenges mandated a multisector approach and cooperation with the private sector."

Among other things, this cooperation between the public and business sectors can be seen in the Cyber Unit of the Estonian Defence League. This group of digital security experts share expertise and devote their efforts to improving the security of companies and government organisations. They can also call on the services of volunteers when digital attacks threaten the critical infrastructure.

Estonia is also fostering cooperation at the international level. They put digital security on the agenda at NATO, and they advocated the establishment of a Cooperative Cyber Defence Centre of Excellence. This was realised in 2008, with headquarters in Tallinn. Within the EU as well, Estonia is a pioneer in this area.

Cybercitizenship

A lot of effort is also going into informing citizens about digital security. Training is available for children of all ages. A project that educates and develops the skills of users, developers and vendors of smart devices was launched in 2013. There are also special Internet police now for people who have questions about matters such as online identity fraud. "The notion of good cybercitizenship is crucial", says Aaviksoo. "We believe that a precondition for securing cyberspace is that every owner of a computer, computer network or information system feels responsible for the expedient and prudent use of information and communications technology."

On the Estonian digital portal (*e-estonia.com*), Klaid Mägi, former head of the Estonian Computer Emergency Response Team, says: "Most importantly, there is a common understanding that cyber security can only be ensured through cooperation, and that a joint contribution is required at all levels – state, private sector and individuals." [5]

(160659-I)

Web Links

[1] www.itu.int/dms_pub/itu-d/opb/str/D-STR-GCI.01-2017-PDF-E.pdf

[2] https://qz.com/1052269/every-country-should-have-a-cyber-war-what-estonia-learned-from-russian-hacking/

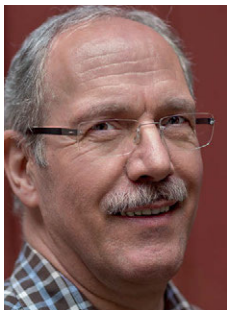
[3] www.mkm.ee/sites/default/files/cyber_security_strategy_2014-2017_public_version.pdf

[4] www.coedat.nato.int/publication/datr/volume6/02-Cyberattacks_Against_Estonia_Raised_Awareness_of_Cyberthreats.pdf

[5] https://e-estonia.com/how-estonia-became-a-global-heavyweight-in-cyber-security/

welcome in your ONLINE STORE

EDITOR'S CHOICE



Good SMD testers are not cheap. The one we have been using for several years now in the Elektor Labs cost well above 200 euros. The Chinese company Mastech has now launched the MS8911 SMD tester, which is able to detect resistors (R), capacitors (C) and inductors (L) and is priced under 40 euros. What I certainly did not expect from an instrument in this price range is that the display (with a maximum range of 6000) shows not only the value and the component type, but also the dissipation factor (D) for capacitors or the quality factor (Q) for inductors. The conclusion is brief and concise: For less than 40 euros, the Mastech MS8911 is an outstanding instrument that does exactly what it promises. Everyone who works regularly with SMD passives should have this tester on their bench.

Harry Baggen (Elektor Labs)

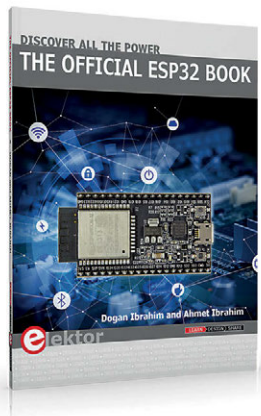
www.elektor.com/mastech-ms8911

Elektor Bestsellers

1. Raspberry Pi 3 (model B+)
   www.elektor.com/rpi3b-plus

2. Raspberry Pi 3 Basic to Advanced Projects
   www.elektor.com/rpi-projects-book
3. HDMI/AV Digital Microscope ADSM302
   www.elektor.com/adsm302
4. The Official ESP32 Book
   www.elektor.com/esp32-book
5. DVD Elektor 2017
   www.elektor.com/dvd-elektor-2017
6. Electronic Circuits for All
   www.elektor.com/electronic-circuits
7. pi-top 2 (DIY Laptop Kit)
   www.elektor.com/pi-top-v2

The Official ESP32 Book

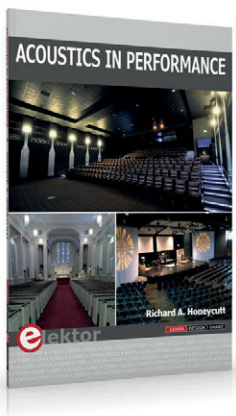


This book is an introduction to the ESP32 processor and describes the main hardware and software features of this chip. The book teaches the reader how to use the ESP32 hardware and software in practical projects. Many basic, simple, and intermediate level projects are given in the book based on the ESP32 DevKitC development board, using the highly popular Arduino IDE and also the MicroPython programming language.

member price: £27.95 • €31.46 • US $39

www.elektor.com/esp32-book

Acoustics in Performance



This book helps those responsible for providing good acoustics in performance and worship spaces to understand the variables and choices entailed in proper acoustic design for performance and worship. Practicing acoustical consultants will find the book a useful reference as well. The level of presentation is comfortable and straightforward without being simplistic.

member price: £19.95 • €22.46 • US $28

www.elektor.com/acoustics-in-performance

pi-top 2 DIY Laptop Kit

The new pi-top 2 turns a Raspberry Pi into a modular laptop. Learn to code, create your own devices, and take your knowledge to the next level. Inspired by makers, educators and inventors who love the original pi-top, this new version comes as a killer, new generation of pi-top laptops: 1920 x 1080 screen resolution, keyboard & trackpad, battery (up to 8h) and Polaris operating system. The included inventor's kit comprises several components including pi-topPROTO+.

member price: £237.95 • €269.96 • US $328

www.elektor.com/pi-top-v2

Anet E10 3D Printer

Quick Assembly Kit

(incl. 3 PLA Filaments)

The Anet E10 is a distinctive 3D printer, fashionable, easy and classy.

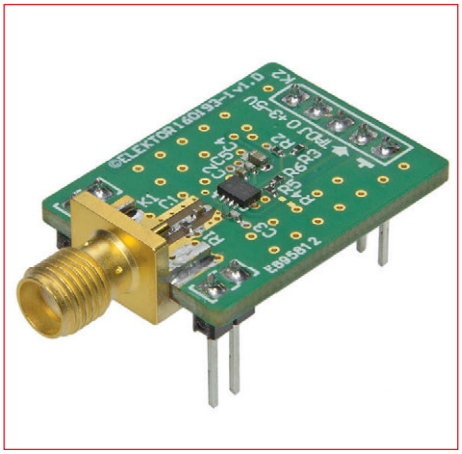
Main Features

- Rapid installation: only 10 minutes
- 12864 large LCD screen, one rotary button
- Aluminium alloy frame
- Low noise
- High printing precision
- Large printing size
- Unique belt adjustment design

SPECIAL PRICE: £280.95 • €319 • US $387

www.elektor.com/anet-e10

RF Power Meter

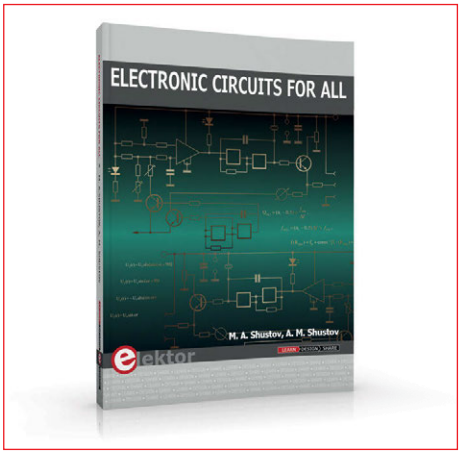


This compact RF Meter is composed of an RF log detector/controller AD8317 together with an Arduino Nano and an LCD. The RF front-end is a 4-layer break-out board with an SMA connector for the input signal. The main board holds the RF front-end, the Arduino Nano and the LCD. With this RF power meter you can easily check how much power you are transmitting, at frequencies up to 10 GHz.

member price: £25.95 • €29.25 • US $36

www.elektor.com/rf-power-meter

Electronic Circuits for All

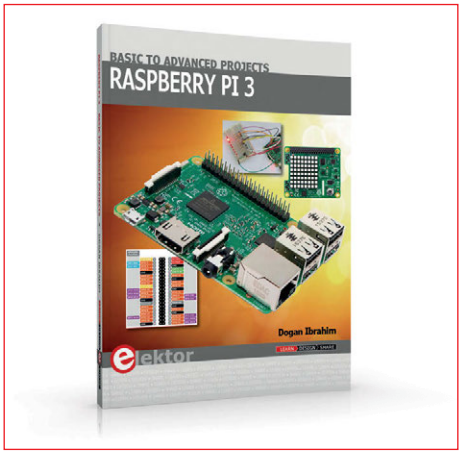


This book includes 400 new and original radio electronic multipurpose circuits. The technical solutions presented in the book are intended to stimulate the creative imagination of readers and broaden their area of thought. The chapters are devoted to power electronics and measuring equipment and contain numerous original circuits of generators, amplifiers, filters, electronic switches based on thyristors and CMOS switch elements.

member price: £31.95 • €35.95 • US $44

www.elektor.com/electronic-circuits

Raspberry Pi 3 Basic to Advanced Projects



This book is about the Raspberry Pi 3 computer and its use in various control and monitoring applications. The nice feature of this book is that it covers many Raspberry Pi 3 based hardware projects using the latest hardware modules such as the Sense HAT, Swiss Pi, MotoPi, Camera module, and many other state of the art analog and digital sensors. All the projects in the book are based on the Python programming language and they have been fully tested.

member price: £27.95 • €31.46 • US $39

www.elektor.com/rpi-projects-book

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 May 23, 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 2/2018 (March & April) is: **BA0C2**.

The €50 / £40 / $70 book vouchers have been awarded to: Jörg Steffensky (Germany); David Turnbull (UK); Alexandre Bourque (Canada); Vincent Pierdominici (USA); Jean-Baptiste Roulier (France).

Congratulations everyone!

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

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

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

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

electronica Fast Forward is brought to you by

electronica elektor

DO YOU WANT THE BEST ELECTRONICS DESIGN SOFTWARE

FEATURES

- Schematic Capture
- PCB Layout
- Gridless Autorouting
- 3D Visualization
- M-CAD Integration
- SPICE Simulation
- MCU Co-simulation
- Built in IDE
- Visual Programming

NOW INCLUDES:
Serpentine Routing, Layer Stackup Manager, and Assembly Variants.

Labcenter Electronics

www.labcenter.com

(+44) 01756 753440